
pip_shims Documentation

Release 0.5.3

Dan Ryan <dan@danryan.co>

August 07, 2020

1	pip-shims: Shims for importing packages from pip's internals.	1
1.1	Warning	1
1.2	Installation	1
1.3	Summary	2
1.3.1	Importing a shim	2
1.3.2	Resolving Dependencies	2
1.4	Available Shims	3
2	pip_shims package	5
2.1	Submodules	5
2.1.1	pip_shims.models	5
2.1.2	pip_shims.compat	8
2.1.3	pip_shims.utils	21
2.1.4	pip_shims.shims	24
2.1.5	pip_shims.environment	55
2.2	Submodules	85
2.2.1	pip_shims.compat module	85
2.2.2	pip_shims.environment module	98
2.2.3	pip_shims.models module	98
2.2.4	pip_shims.shims module	101
2.2.5	pip_shims.utils module	132
3	0.5.3 (2020-08-08)	137
3.1	Bug Fixes	137
4	0.5.2 (2020-04-22)	139
4.1	Features	139
5	0.5.1 (2020-03-10)	141
5.1	Bug Fixes	141
6	0.5.0 (2020-01-28)	143
6.1	Features	143
6.2	Bug Fixes	143
7	0.4.0 (2019-11-22)	145
7.1	Features	145

8	0.3.4 (2019-11-18)	147
8.1	Features	147
8.2	Bug Fixes	147
9	0.3.3 (2019-06-16)	149
9.1	Features	149
9.2	Bug Fixes	149
10	0.3.2 (2018-10-27)	151
10.1	Features	151
11	0.3.1 (2018-10-06)	153
11.1	Features	153
12	0.3.0 (2018-10-06)	155
12.1	Features	155
12.2	Bug Fixes	155
13	0.2.0 (2018-10-05)	157
13.1	Features	157
14	0.1.2 (2018-08-18)	159
14.1	Features	159
14.2	Bug Fixes	159
15	0.1.1 (2018-08-14)	161
15.1	Bug Fixes	161
15.2	Documentation Updates	161
16	0.1.0 (2018-08-09)	163
16.1	Features	163
17	pip_shims	165
17.1	Submodules	165
17.1.1	pip_shims.models	165
17.1.2	pip_shims.compat	168
17.1.3	pip_shims.utils	181
17.1.4	pip_shims.shims	184
17.1.5	pip_shims.environment	215
18	pip_shims.shims	247
19	pip_shims.models	279
20	pip_shims.compat	283
21	pip_shims.environment	297
22	pip_shims.utils	299
23	Indices and tables	303
	Python Module Index	305
	Index	307

pip-shims: Shims for importing packages from pip's internals.

1.1 Warning

Warning: The authors of `pip` do not condone the use of this package. Relying on pip's internals is a **dangerous** idea for your software as they are broken intentionally and regularly. This package may not always be completely updated up PyPI, so relying on it may break your code! User beware!

1.2 Installation

Install from `PyPI`:

```
$ pipenv install pip-shims
```

Install from `Github`:

```
$ pipenv install -e git+https://github.com/sarugaku/pip-shims.git#egg=pip-  
→shims
```

1.3 Summary

pip-shims is a set of compatibility access shims to the [pip](#) internal API. **pip-shims** provides compatibility with pip versions 8.0 through the current release (18.x). The shims are provided using a lazy import strategy by hacking a module by overloading a class instance's `getattr` method. This library exists due to my constant writing of the same set of import shims across many different libraries, including [pipenv](#), [pip-tools](#), [requirementslib](#), and [passa](#).

1.3.1 Importing a shim

You can use **pip-shims** to expose elements of **pip**'s internal API by importing them:

```
>>> from pip_shims import Wheel
>>> mywheel = Wheel('/path/to/my/wheel.whl')
```

1.3.2 Resolving Dependencies

You can resolve the dependencies of a package using the shimmed resolver interface:

```
>>> from pip_shims.shims import resolve, InstallRequirement
>>> ireq = InstallRequirement.from_line("requests>=2.20")
>>> results = resolve(ireq)
>>> for k, v in results.items():
...     print("{0}: {1!r}".format(k, v))
requests: <InstallRequirement object: requests>=2.20 from https://files.pythonhosted.
↳org/packages/51/bd/23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/
↳requests-2.22.0-py2.py3-none-any.whl
↳#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9a590f48c010551dc6c4b31
↳editable=False>
idna: <InstallRequirement object: idna<2.9,>=2.5 from https://files.pythonhosted.org/
↳packages/14/2c/cd551d81dbe15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-2.
↳8-py2.py3-none-any.whl
↳#sha256=ea8b7f6188e6fa117537c3df7da9fc686d485087abf6ac197f9c46432f7e4a3c (from
↳requests>=2.20) editable=False>
urllib3: <InstallRequirement object: urllib3!=1.25.0,!<1.25.1,<1.26,>=1.21.1 from
↳https://files.pythonhosted.org/packages/b4/40/
↳a9837291310ee1ccc242ceb6ebfd9eb21539649f193a7c8c86ba15b98539/urllib3-1.25.7-py2.py3-
↳none-any.whl
↳#sha256=a8a318824cc77d1fd4b2bec2ded92646630d7fe8619497b142c84a9e6f5a7293 (from
↳requests>=2.20) editable=False>
chardet: <InstallRequirement object: chardet<3.1.0,>=3.0.2 from https://files.
↳pythonhosted.org/packages/bc/a9/
↳01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.py3-
↳none-any.whl
↳#sha256=fc323ffcaeaed0e0a02bf4d117757b98aed530d9ed4531e3e15460124c106691 (from
↳requests>=2.20) editable=False>
certifi: <InstallRequirement object: certifi>=2017.4.17 from https://files.
↳pythonhosted.org/packages/18/b0/
↳8146a4f8dd402f60744fa380bc73ca47303ccccf8b9190fd16a827281eac2/certifi-2019.9.11-py2.
↳py3-none-any.whl
↳#sha256=fd7c7c74727ddcf00e9acd26bba8da604ffec95bf1c2144e67aff7a8b50e6cef (from
↳requests>=2.20) editable=False>
```

1.4 Available Shims

pip-shims provides the following compatibility shims:

Import Path	Import Name	Former Path
__version__	pip_version	
<shimmed>	build_wheel	
<shimmed>	get_package_finder	
<shimmed>	get_requirement_set	
<shimmed>	get_resolver	
<shimmed>	is_archive_file	download
<shimmed>	is_file_url	download
<shimmed>	make_preparer	
<shimmed>	resolve	
<shimmed>	shim_unpack	
cache	WheelCache	wheel
cli	cmdoptions	cmdoptions
cli.base_command	Command	basecommand
cli.cmdoptions	index_group	cmdoptions
cli.cmdoptions	make_option_group	cmdoptions
cli.parser	ConfigOptionParser	baseparser
cli.req_command	SessionCommandMixin	
collector	LinkCollector	
commands	commands_dict	
commands.freeze	DEV_PKGS	
commands.install	InstallCommand	
distributions	make_distribution_for_install_requirement	operations.prepare.make_abstract_dist
distributions.base	AbstractDistribution	
distributions.installed	InstalledDistribution	
distributions.source	SourceDistribution	
distributions.wheel	WheelDistribution	
download	path_to_url	
download	unpack_url	
exceptions	BadCommand	
exceptions	BestVersionAlreadyInstalled	
exceptions	CommandError	
exceptions	DistributionNotFound	
exceptions	DistributionNotFound	
exceptions	InstallationError	
exceptions	PipError	
exceptions	PreviousBuildDirError	
exceptions	RequirementsFileParseError	
exceptions	UninstallationError	
index	CandidateEvaluator	
index	CandidatePreferences	
index	LinkEvaluator	
index	PackageFinder	
index	parse_version	
locations	USER_CACHE_DIR	
models	FormatControl	index

Continued on next page

Table 1 – continued from previous page

Import Path	Import Name	Former Path
models.index	PyPI	
models.link	Link	index
models.search_scope	SearchScope	
models.selection_prefs	SelectionPreferences	
models.target_python	TargetPython	
network.cache	SafeFileCache	download
operations.freeze	FrozenRequirement	<__init__>
operations.prepare	Downloader	
operations.prepare	make_abstract_dist	req.req_set
operations.prepare	RequirementPreparer	
pep425tags	get_supported	
pep425tags	get_tags	
req.constructors	_strip_extras	req.req_install
req.constructors	install_req_from_editable	req.req_install.InstallRequirement
req.constructors	install_req_from_line	req.req_install.InstallRequirement
req.constructors	install_req_from_req_string	
req.req_file	parse_requirements	
req.req_install	InstallRequirement	
req.req_set	RequirementSet	
req.req_tracker	get_requirement_tracker	
req.req_tracker	RequirementTracker	
req.req_uninstall	UninstallPathSet	
resolve	Resolver	
utils.compat	stdlib_pkgs	compat
utils.hashes	FAVORITE_HASH	
utils.misc	get_installed_distributions	utils
utils.misc	is_installable_dir	utils
utils.temp_dir	global_tempdir_manager	
utils.temp_dir	TempDirectory	
utils.urls	url_to_path	download
vcs.versioncontrol	VcsSupport	vcs.VcsSupport
wheel	Wheel	
wheel	WheelBuilder	
wheel_builder	build	
wheel_builder	build_one	
wheel_builder	build_one_inside_env	

pip_shims package

This library is a set of compatibility access shims to the `pip` internal API. It provides compatibility with `pip` versions 8.0 through the current release. The shims are provided using a lazy import strategy by hacking a module by overloading a class instance's `getattr` method. This library exists due to my constant writing of the same set of import shims.

2.1 Submodules

<code>pip_shims.models</code>	Helper module for shimming functionality across <code>pip</code> versions.
<code>pip_shims.compat</code>	Backports and helper functionality to support using new functionality.
<code>pip_shims.utils</code>	Shared utility functions which are not specific to any particular module.
<code>pip_shims.shims</code>	Main module with magic self-replacement mechanisms to handle import speedups.
<code>pip_shims.environment</code>	Module with functionality to learn about the environment.

2.1.1 pip_shims.models

Helper module for shimming functionality across `pip` versions.

```
class pip_shims.models.ImportTypes
```

```
    Bases: pip_shims.models.ImportTypes
```

```
    Create new instance of ImportTypes(FUNCTION, CLASS, MODULE, CONTEXTMANAGER)
```

```
    ATTRIBUTE = 5
```

```
    CLASS = 1
```

```
    CONTEXTMANAGER = 3
```

```

FUNCTION = 0
METHOD = 4
MODULE = 2

_asdict()
    Return a new OrderedDict which maps field names to their values.

_fields = ('FUNCTION', 'CLASS', 'MODULE', 'CONTEXTMANAGER')
_fields_defaults = {}

classmethod _make(iterable)
    Make a new ImportTypes object from a sequence or iterable

_replace(**kws)
    Return a new ImportTypes object replacing specified fields with new values

count()
    Return number of occurrences of value.

index()
    Return first index of value.

    Raises ValueError if the value is not present.

```

```

pip_shims.models.ImportTypesBase
    alias of pip_shims.models.ImportTypes

```

```

class pip_shims.models.PipVersion(version, round_prereleases_up=True,
                                   base_import_path=None, vendor_import_path='pip._vendor')

```

```

Bases: collections.abc.Sequence

```

```

_abc_impl = <_abc_data object>

```

```

_parse()

```

```

count(value) → integer – return number of occurrences of value

```

```

index(value[, start[, stop]]) → integer – return first index of value.
    Raises ValueError if the value is not present.

```

Supporting start and stop arguments is optional, but recommended.

```

is_valid(compared_to)

```

```

version_key

```

```

version_tuple

```

```

class pip_shims.models.PipVersionRange(start, end)

```

```

Bases: collections.abc.Sequence

```

```

_abc_impl = <_abc_data object>

```

```

base_import_paths

```

```

count(value) → integer – return number of occurrences of value

```

```

index(value[, start[, stop]]) → integer – return first index of value.
    Raises ValueError if the value is not present.

```

Supporting start and stop arguments is optional, but recommended.

```

is_valid()

```

vendor_import_paths

```
class pip_shims.models.ShimmedPath(name, import_target, import_type, version_range, provided_methods=None, provided_functions=None, provided_classmethods=None, provided_contextmanagers=None, provided_mixins=None, default_args=None)
```

Bases: `object`

```
_ShimmedPath__modules = {'pip._internal.cache': <module 'pip._internal.cache' from '/
```

```
_apply_aliases (imported, target)
```

```
_as_tuple ()
```

```
_ensure_functions (provided)
```

```
_ensure_methods (provided)
```

Given a base class, a new name, and any number of functions to attach, turns those functions into classmethods, attaches them, and returns an updated class object.

```
_import (prefix=None)
```

```
classmethod _import_module (module)
```

```
classmethod _parse_provides_dict (provides, prepend_arg_to_callables=None)
```

```
_shim_base (imported, attribute_name)
```

```
_shim_parent (imported, attribute_name)
```

```
_update_default_kwargs (parent, provided)
```

```
alias (aliases)
```

```
calculated_module_path
```

```
is_attribute
```

```
is_class
```

```
is_contextmanager
```

```
is_function
```

```
is_method
```

```
is_module
```

```
is_valid
```

```
shim ()
```

```
shim_attribute (imported, attribute_name)
```

```
shim_class (imported, attribute_name)
```

```
shim_contextmanager (imported, attribute_name)
```

```
shim_function (imported, attribute_name)
```

```
shim_module (imported, attribute_name)
```

```
shimmed
```

```
sort_order
```

```
update_sys_modules (imported)
```

```

class pip_shims.models.ShimmedPathCollection (name, import_type, paths=None)
    Bases: object

    __ShimmedPathCollection__registry = {'AbstractDistribution': <pip_shims.models.Shimmed
    __get_top_path ()
    __sort_paths ()
    add_mixin (mixin)
    add_path (path)
    alias (aliases)
        Takes a list of methods, functions, attributes, etc and ensures they all exist on the object pointing at the
        same referent.

        Parameters aliases (List[str]) – Names to map to the same functionality if they do not
            exist.

        Returns None

        Return type None

    create_path (import_path, version_start, version_end=None)
    classmethod get_registry ()
    pre_shim (fn)
    provide_function (name, fn)
    provide_method (name, fn)
    register ()
    set_default (default)
    set_default_args (callable_name, *args, **kwargs)
    shim ()
    classmethod traverse (shim)

pip_shims.models.import_pip ()
pip_shims.models.lookup_current_pip_version ()
pip_shims.models.pip_version_lookup (version, *args, **kwargs)

```

2.1.2 pip_shims.compat

Backports and helper functionality to support using new functionality.

```

class pip_shims.compat.CandidateEvaluator (project_name, supported_tags, spec-
                                        ifier, prefer_binary=False, al-
                                        low_all_prereleases=False, hashes=None)

    Bases: object

    classmethod create (project_name, target_python=None, prefer_binary=False, al-
                        low_all_prereleases=False, specifier=None, hashes=None)

class pip_shims.compat.CandidatePreferences (prefer_binary=False, al-
                                        low_all_prereleases=False)

    Bases: object

```

exception `pip_shims.compat.InvalidWheelFilename`

Bases: `Exception`

Wheel Filename is Invalid

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class `pip_shims.compat.LinkCollector` (*session=None, search_scope=None*)

Bases: `object`

class `pip_shims.compat.LinkEvaluator` (*allow_yanked, project_name, canonical_name, formats, target_python, ignore_requires_python=False, ignore_compatibility=True*)

Bases: `object`

class `pip_shims.compat.SearchScope` (*find_links=None, index_urls=None*)

Bases: `object`

classmethod `create` (*find_links=None, index_urls=None*)

class `pip_shims.compat.SelectionPreferences` (*allow_yanked=True, low_all_prereleases=False, format_control=None, prefer_binary=False, ignore_requires_python=False*)

Bases: `object`

class `pip_shims.compat.TargetPython` (*platform=None, py_version_info=None, abi=None, implementation=None*)

Bases: `object`

fallback_get_tags = `<pip_shims.models.ShimmedPathCollection object>`

get_tags()

class `pip_shims.compat.Wheel` (*filename*)

Bases: `object`

get_formatted_file_tags()

Return the wheel's tags as a sorted list of strings.

support_index_min (*tags*)

Return the lowest index that one of the wheel's file_tag combinations achieves in the given list of supported tags.

For example, if there are 8 supported tags and one of the file tags is first in the list, then return 0.

Parameters `tags` – the PEP 425 tags to check the wheel against, in order with most preferred first.

Raises `ValueError` – If none of the wheel's file tags match one of the supported tags.

supported (*tags*)

Return whether the wheel is compatible with one of the given tags.

Parameters `tags` – the PEP 425 tags to check the wheel against.

wheel_file_re = `re.compile('^(?P<namever>(?P<name>.+)-(?P<ver>.*))\n ((- (?P<build>\`

`pip_shims.compat._ensure_finder` (*finder=None, finder_provider=None, install_cmd=None, options=None, session=None*)

```
pip_shims.compat._ensure_wheel_cache (wheel_cache=None, wheel_cache_provider=None,
                                     format_control=None, format_control_provider=None,
                                     options=None, cache_dir=None)
```

```
pip_shims.compat.build_wheel (req=None, reqset=None, output_dir=None, pre-
                              parer=None, wheel_cache=None, build_options=None,
                              global_options=None, check_binary_allowed=None,
                              no_clean=False, session=None, finder=None, in-
                              stall_command=None, req_tracker=None, build_dir=None,
                              src_dir=None, download_dir=None, wheel_download_dir=None,
                              cache_dir=None, use_user_site=False, use_pep517=None,
                              format_control_provider=None, wheel_cache_provider=None,
                              preparer_provider=None, wheel_builder_provider=None,
                              build_one_provider=None, build_one_inside_env_provider=None,
                              build_many_provider=None, install_command_provider=None,
                              finder_provider=None, reqset_provider=None)
```

Build a wheel or a set of wheels

Raises `TypeError` – Raised when no requirements are provided

Parameters

- **req** (*Optional*[*TInstallRequirement*]) – An *InstallRequirement* to build
- **reqset** (*Optional*[*TReqSet*]) – A *RequirementSet* instance (*pip*<10) or an iterable of *InstallRequirement* instances (*pip*>=10) to build
- **output_dir** (*Optional*[*str*]) – Target output directory, only useful when building one wheel using *pip*>=20.0
- **preparer** (*Optional*[*TPreparer*]) – A preparer instance, defaults to None
- **wheel_cache** (*Optional*[*TWheelCache*]) – A wheel cache instance, defaults to None
- **build_options** (*Optional*[*List*[*str*]]) – A list of build options to pass in
- **global_options** (*Optional*[*List*[*str*]]) – A list of global options to pass in
- **bool]] check_binary_allowed** (*Optional*[*Callable*[*TInstallRequirement*,]]) – A callable to check whether we are allowed to build and cache wheels for an ireq
- **no_clean** (*bool*) – Whether to avoid cleaning up wheels
- **session** (*Optional*[*TSession*]) – A *PipSession* instance to pass to create a *finder* if necessary
- **finder** (*Optional*[*TFinder*]) – A *PackageFinder* instance to use for generating a *WheelBuilder* instance on *pip*<20
- **install_command** (*Optional*[*TCommandInstance*]) – The install command used to create the finder, session, and options if needed, defaults to None.
- **req_tracker** (*Optional*[*TReqTracker*]) – An optional requirement tracker instance, if one already exists
- **build_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **src_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **download_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **wheel_download_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer

- **cache_dir** (*Optional[str]*) – Passthrough cache directory for wheel cache options
- **use_user_site** (*bool*) – Whether to use the user site directory when preparing install requirements on *pip<20*
- **use_pep517** (*Optional[bool]*) – When set to *True* or *False*, prefers building with or without pep517 as specified, otherwise uses requirement preference. Only works for single requirements.
- **format_control_provider** (*Optional[TShimmedFunc]*) – A provider for the *FormatControl* class
- **wheel_cache_provider** (*Optional[TShimmedFunc]*) – A provider for the *WheelCache* class
- **preparer_provider** (*Optional[TShimmedFunc]*) – A provider for the *RequirementPreparer* class
- **wheel_builder_provider** (*Optional[TShimmedFunc]*) – A provider for the *WheelBuilder* class, if it exists
- **build_one_provider** (*Optional[TShimmedFunc]*) – A provider for the *_build_one* function, if it exists
- **build_one_inside_env_provider** (*Optional[TShimmedFunc]*) – A provider for the *_build_one_inside_env* function, if it exists
- **build_many_provider** (*Optional[TShimmedFunc]*) – A provider for the *build* function, if it exists
- **install_command_provider** (*Optional[TShimmedFunc]*) – A shim for providing new install command instances
- **finder_provider** (*TShimmedFunc*) – A provider to package finder instances
- **reqset_provider** (*TShimmedFunc*) – A provider for requirement set generation

Returns A tuple of successful and failed install requirements or else a path to a wheel

Return type *Optional[Union[str, Tuple[List[TInstallRequirement], List[TInstallRequirement]]]]*

`pip_shims.compat.ensure_resolution_dirs` (***kwargs*)

Ensures that the proper directories are scaffolded and present in the provided kwargs for performing dependency resolution via pip.

Returns A new kwargs dictionary with scaffolded directories for **build_dir**, **src_dir**, **download_dir**, and **wheel_download_dir** added to the key value pairs.

Return type *Dict[str, Any]*

`pip_shims.compat.get_ireq_output_path` (*wheel_cache, ireq*)

`pip_shims.compat.get_package_finder` (*install_cmd=None, options=None, session=None, platform=None, python_versions=None, abi=None, implementation=None, target_python=None, ignore_requires_python=None, target_python_builder=None, install_cmd_provider=None*)

Shim for compatibility to generate package finders.

Build and return a *PackageFinder* instance using the *InstallCommand* helper method to construct the finder, shimmed with backports as needed for compatibility.

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_cmd** – A `InstallCommand` instance which is used to generate the finder.
- **options** (*optparse.Values*) – An optional `optparse.Values` instance generated by calling `install_cmd.parser.parse_args()` typically.
- **session** – An optional session instance, can be created by the `install_cmd`.
- **platform** (*Optional[str]*) – An optional platform string, e.g. `linux_x86_64`
- **python_versions** (*Optional[Tuple[str, ...]]*) – A tuple of 2-digit strings representing python versions, e.g. (“27”, “35”, “36”, “37”...)
- **abi** (*Optional[str]*) – The target abi to support, e.g. “cp38”
- **implementation** (*Optional[str]*) – An optional implementation string for limiting searches to a specific implementation, e.g. “cp” or “py”
- **target_python** – A `TargetPython` instance (will be translated to alternate arguments if necessary on incompatible pip versions).
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore `requires_python` on resulting candidates, only valid after pip version 19.3.1
- **target_python_builder** – A ‘TargetPython’ builder (e.g. the class itself, uninstantiated)

Returns A `pip._internal.index.package_finder.PackageFinder` instance

Return type `pip._internal.index.package_finder.PackageFinder`

Example

```
>>> from pip_shims.shims import InstallCommand, get_package_finder
>>> install_cmd = InstallCommand()
>>> finder = get_package_finder(
...     install_cmd, python_versions=("27", "35", "36", "37", "38"),
...     implementation="
cp"
... )
>>> candidates = finder.find_all_candidates("requests")
>>> requests_222 = next(iter(c for c in candidates if c.version.public == "2.22.0
... ))
>>> requests_222
<InstallationCandidate('requests', <Version('2.22.0')>, <Link https://files.
...pythonhos
ted.org/packages/51/bd/
...23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/r
equests-2.22.0-py2.py3-none-any.whl
...#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9
a590f48c010551dc6c4b31 (from https://pypi.org/simple/requests/) (requires-python:>
...=2.
7, !=3.0.*, !=3.1.*, !=3.2.*, !=3.3.*, !=3.4.*)>>>
```



```

pip_shims.compat.get_requirement_set (install_command=None, req_set_provider=None,
                                     build_dir=None, src_dir=None, download_dir=None,
                                     wheel_download_dir=None, session=None,
                                     wheel_cache=None, upgrade=False, upgrade_strategy=None,
                                     ignore_installed=False, ignore_dependencies=False,
                                     force_reinstall=False, use_user_site=False,
                                     isolated=False, ignore_requires_python=False,
                                     require_hashes=None, cache_dir=None,
                                     options=None, install_cmd_provider=None,
                                     wheel_cache_provider=None)

```

Creates a requirement set from the supplied parameters.

Not all parameters are passed through for all pip versions, but any invalid parameters will be ignored if they are not needed to generate a requirement set on the current pip version.

:param *ShimmedPathCollection* wheel_cache_provider: A context manager provider which resolves to a *WheelCache* instance

Parameters install_command – A *InstallCommand* instance which is used to generate the finder.

:param *ShimmedPathCollection* req_set_provider: A provider to build requirement set instances.

Parameters

- **build_dir** (*str*) – The directory to build requirements in. Removed in pip 10, defaults to None
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*str*) – The directory to download requirement artifacts to. Removed in pip 10, defaults to None
- **wheel_download_dir** (*str*) – The directory to download wheels to. Removed in pip 10, defaults to None

:param *Session* session: The pip session to use. Removed in pip 10, defaults to None

Parameters

- **wheel_cache** (*WheelCache*) – The pip *WheelCache* instance to use for caching wheels. Removed in pip 10, defaults to None
- **upgrade** (*bool*) – Whether to try to upgrade existing requirements. Removed in pip 10, defaults to False.
- **upgrade_strategy** (*str*) – The upgrade strategy to use, e.g. “only-if-needed”. Removed in pip 10, defaults to None.
- **ignore_installed** (*bool*) – Whether to ignore installed packages when resolving. Removed in pip 10, defaults to False.
- **ignore_dependencies** (*bool*) – Whether to ignore dependencies of requirements when resolving. Removed in pip 10, defaults to False.
- **force_reinstall** (*bool*) – Whether to force reinstall of packages when resolving. Removed in pip 10, defaults to False.

- **use_user_site** (*bool*) – Whether to use user site packages when resolving. Removed in pip 10, defaults to False.
- **isolated** (*bool*) – Whether to resolve in isolation. Removed in pip 10, defaults to False.
- **ignore_requires_python** (*bool*) – Removed in pip 10, defaults to False.
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **options** (*Values*) – An *Values* instance from an install cmd
- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.

Returns A new requirement set instance

Return type RequirementSet

`pip_shims.compat.get_requirement_tracker` (*req_tracker_creator=None*)

`pip_shims.compat.get_resolver` (*resolver_fn, install_req_provider=None, format_control_provider=None, wheel_cache_provider=None, finder=None, upgrade_strategy='to-satisfy-only', force_reinstall=None, ignore_dependencies=None, ignore_requires_python=None, ignore_installed=True, use_user_site=False, isolated=None, wheel_cache=None, preparer=None, session=None, options=None, make_install_req=None, install_cmd_provider=None, install_cmd=None, use_pep517=True*)

A resolver creation compatibility shim for generating a resolver.

Consumes any argument that was previously used to instantiate a resolver, discards anything that is no longer valid.

Note: This is only valid for **pip >= 10.0.0**

Raises

- **ValueError** – A session is required but not provided and one cannot be created
- **ValueError** – A finder is required but not provided and one cannot be created
- **ValueError** – An install requirement provider is required and has not been provided

Parameters

- **resolver_fn** (*TShimmedFunc*) – The resolver function used to create new resolver instances.
- **install_req_provider** (*TShimmedFunc*) – The provider function to use to generate install requirements if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a *format_control* instance if needed.
- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.

- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to `None`
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to `None`
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to `None`
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to `True`
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to `False`
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to `None`
- **wheel_cache** (*Optional[WheelCache]*) – The wheel cache to use, defaults to `None`
- **preparer** (*Optional[TPreparer]*) – The requirement preparer to use, defaults to `None`
- **session** (*Optional[Session]*) – Existing session to use for getting requirements, defaults to `None`
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to `None`
- **make_install_req** (*Optional[functools.partial]*) – The partial function to pass in to the resolver for actually generating install requirements, if necessary
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to `None`.
- **use_pep517** (*bool*) – Whether to use the pep517 build process.

Returns A new resolver instance.

Return type Resolver

Example

```
>>> import os
>>> from tempdir import TemporaryDirectory
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_
↳ tracker,
...     get_resolver, InstallRequirement, RequirementSet
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> wheel_cache = WheelCache(USER_CACHE_DIR, FormatControl(None, None))
>>> with TemporaryDirectory() as temp_base:
...     reqset = RequirementSet()
...     ireq = InstallRequirement.from_line("requests")
```

(continues on next page)

(continued from previous page)

```

...     ireq.is_direct = True
...     build_dir = os.path.join(temp_base, "build")
...     src_dir = os.path.join(temp_base, "src")
...     ireq.build_location(build_dir)
...     with make_preparer(
...         options=pip_options, finder=finder, session=session,
...         build_dir=build_dir, install_cmd=install_cmd,
...     ) as preparer:
...         resolver = get_resolver(
...             finder=finder, ignore_dependencies=False, ignore_requires_
↪python=True,
...             preparer=preparer, session=session, options=pip_options,
...             install_cmd=install_cmd, wheel_cache=wheel_cache,
...         )
...         resolver.require_hashes = False
...         reqset.add_requirement(ireq)
...         results = resolver.resolve(reqset)
...         #reqset.cleanup_files()
...         for result_req in reqset.requirements:
...             print(result_req)
requests
charset
certifi
urllib3
idna

```

`pip_shims.compat.get_session` (*install_cmd_provider=None, install_cmd=None, options=None*)

`pip_shims.compat.make_preparer` (*preparer_fn, req_tracker_fn=None, build_dir=None, src_dir=None, download_dir=None, wheel_download_dir=None, progress_bar='off', build_isolation=False, session=None, finder=None, options=None, require_hashes=None, use_user_site=None, req_tracker=None, install_cmd_provider=None, downloader_provider=None, install_cmd=None, finder_provider=None*)

Creates a requirement preparer for preparing pip requirements.

Provides a compatibility shim that accepts all previously valid arguments and discards any that are no longer used.

Raises

- **`TypeError`** – No requirement tracker provided and one cannot be generated
- **`TypeError`** – No valid sessions provided and one cannot be generated
- **`TypeError`** – No valid finders provided and one cannot be generated

Parameters

- **`preparer_fn`** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **`req_tracker_fn`** (*Optional[TShimmedFunc]*) – Callable or shim for generating requirement trackers, defaults to None
- **`build_dir`** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **`src_dir`** (*Optional[str]*) – Directory to find or extract source files, defaults to None

- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **progress_bar** (*str*) – Whether to display a progress bar, defaults to off
- **build_isolation** (*bool*) – Whether to build requirements in isolation, defaults to False
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **require_hashes** (*Optional[bool]*) – Whether to require hashes for preparation
- **use_user_site** (*Optional[bool]*) – Whether to use the user site directory for preparing requirements
- **TShimmedFunc]] req_tracker** (*Optional[Union[TReqTracker,])*) – The requirement tracker to use for building packages, defaults to None
- **downloader_provider** (*Optional[TShimmedFunc]*) – A downloader provider
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None
- **finder_provider** (*Optional[TShimmedFunc]*) – A package finder provider

Yield A new requirement preparer instance

Return type ContextManager[RequirementPreparer]

Example

```
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_tracker
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> with make_preparer(
...     options=pip_options, finder=finder, session=session, install_cmd=ic
... ) as preparer:
...     print(preparer)
<pip._internal.operations.prepare.RequirementPreparer object at 0x7f8a2734be80>
```

`pip_shims.compat.partial_command` (*shimmed_path, cmd_mapping=None*)

Maps a default set of arguments across all members of a *ShimmedPath* instance, specifically for Command instances which need *summary* and *name* arguments.

:param ShimmedPath shimmed_path: A ShimmedCollection instance

Parameters cmd_mapping (*Any*) – A reference to use for mapping against, e.g. an import that depends on pip also

Returns A dictionary mapping new arguments to their default values

Return type Dict[str, str]

pip_shims.compat.**populate_options** (*install_command=None, options=None, **kwargs*)

```
pip_shims.compat.resolve (ireq, reqset_provider=None, req_tracker_provider=None,
                          install_cmd_provider=None, install_command=None,
                          finder_provider=None, resolver_provider=None,
                          wheel_cache_provider=None, format_control_provider=None,
                          make_preparer_provider=None, tempdir_manager_provider=None,
                          options=None, session=None, resolver=None, finder=None,
                          upgrade_strategy='to-satisfy-only', force_reinstall=None, ignore_dependencies=None,
                          ignore_requires_python=None, ignore_installed=True, use_user_site=False, isolated=None,
                          build_dir=None, source_dir=None, download_dir=None, cache_dir=None, wheel_download_dir=None,
                          wheel_cache=None, require_hashes=None, check_supported_wheels=True)
```

Resolves the provided **InstallRequirement**, returning a dictionary.

Maps a dictionary of names to corresponding `InstallRequirement` values.

:param InstallRequirement ireq: An `InstallRequirement` to initiate the resolution process

:param ShimmedPathCollection reqset_provider: A provider to build requirement set instances.

:param ShimmedPathCollection req_tracker_provider: A provider to build requirement tracker instances

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_command** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None.

:param ShimmedPathCollection finder_provider: A provider to package finder instances.

:param ShimmedPathCollection resolver_provider: A provider to build resolver instances

Parameters

- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a `format_control` instance if needed.
- **make_preparer_provider** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **tempdir_manager_provider** (*Optional[TShimmedFunc]*) – Shim for generating tempdir manager for pip temporary directories
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None

:param Resolver resolver: A pre-existing resolver instance to use for resolution

Parameters

- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to None
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to None
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to None
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to True
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to False
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **cache_dir** (*str*) – The cache directory to use for caching artifacts during resolution
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **wheel_cache** (*Optional[TWheelCache]*) – The wheel cache to use, defaults to None
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **check_supported_wheels** (*bool*) – Whether to check support of wheels before including them in resolution.

Returns A dictionary mapping requirements to corresponding `:class:`~pip._internal.req.req_install.InstallRequirement`'s`

Return type `InstallRequirement`

Example

```
>>> from pip_shims.shims import resolve, InstallRequirement
>>> ireq = InstallRequirement.from_line("requests>=2.20")
>>> results = resolve(ireq)
>>> for k, v in results.items():
...     print("{0}: {1!r}".format(k, v))
requests: <InstallRequirement object: requests>=2.20 from https://files.
pythonhosted.
org/packages/51/bd/23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/
reque
```

(continues on next page)

(continued from previous page)

```

sts-2.22.0-py2.py3-none-any.whl
↳#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9a590
f48c010551dc6c4b31 editable=False>
idna: <InstallRequirement object: idna<2.9,>=2.5 from https://files.pythonhosted.
↳org/
packages/14/2c/cd551d81dbe15200belcf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-
↳2.8-
py2.py3-none-any.whl
↳#sha256=ea8b7f6188e6fa117537c3df7da9fc686d485087abf6ac197f9c46432
f7e4a3c (from requests>=2.20) editable=False>
urllib3: <InstallRequirement object: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 from
↳htt
ps://files.pythonhosted.org/packages/b4/40/
↳a9837291310eelccc242ceb6ebfd9eb21539649f19
3a7c8c86ba15b98539/urllib3-1.25.7-py2.py3-none-any.whl
↳#sha256=a8a318824cc77d1fd4b2bec
2ded92646630d7fe8619497b142c84a9e6f5a7293 (from requests>=2.20) editable=False>
chardet: <InstallRequirement object: chardet<3.1.0,>=3.0.2 from https://files.
↳pythonh
osted.org/packages/bc/a9/
↳01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8
/chardet-3.0.4-py2.py3-none-any.whl
↳#sha256=fc323ffcaead0e0a02bf4d117757b98aed530d9ed
4531e3e15460124c106691 (from requests>=2.20) editable=False>
certifi: <InstallRequirement object: certifi>=2017.4.17 from https://files.
↳pythonhost
ed.org/packages/18/b0/
↳8146a4f8dd402f60744fa380bc73ca47303ccccf8b9190fd16a827281eac2/ce
rtifi-2019.9.11-py2.py3-none-any.whl
↳#sha256=fd7c7c74727ddcf00e9acd26bba8da604ffec95bf
1c2144e67aff7a8b50e6cef (from requests>=2.20) editable=False>

```

`pip_shims.compat.resolve_possible_shim(target)`

`pip_shims.compat.shim_unpack(unpack_fn, download_dir, tempdir_manager_provider, ireq=None, link=None, location=None, hashes=None, progress_bar='off', only_download=None, downloader_provider=None, session=None)`

Accepts all parameters that have been valid to pass to `pip._internal.download.unpack_url()` and selects or drops parameters as needed before invoking the provided callable.

Parameters

- **unpack_fn** (*Callable*) – A callable or shim referring to the pip implementation
- **download_dir** (*str*) – The directory to download the file to
- **tempdir_manager_provider** (*TShimmedFunc*) – A callable or shim referring to `global_tempdir_manager` function from pip or a shimmed no-op context manager

:param Optional[InstallRequirement] ireq: an Install Requirement instance, defaults to None

:param Optional[Link] link: A Link instance, defaults to None.

Parameters

- **location** (*Optional[str]*) – A location or source directory if the target is a VCS url, defaults to None.

- **hashes** (*Optional[Any]*) – A Hashes instance, defaults to None
- **progress_bar** (*str*) – Indicates progress bar usage during download, defaults to off.
- **only_download** (*Optional[bool]*) – Whether to skip install, defaults to None.
- **downloader_provider** (*Optional[ShimmedPathCollection]*) – A downloader class to instantiate, if applicable.
- **session** (*Optional[Session]*) – A PipSession instance, defaults to None.

Returns The result of unpacking the url.

Return type None

`pip_shims.compat.temp_environ()`

Allow the ability to set os.environ temporarily

`pip_shims.compat.wheel_cache(cache_dir=None, format_control=None, wheel_cache_provider=None, format_control_provider=None, tempdir_manager_provider=None)`

2.1.3 pip_shims.utils

Shared utility functions which are not specific to any particular module.

class `pip_shims.utils.BaseClassMethod` (*func_base, name, *args, **kwargs*)

Bases: `collections.abc.Callable`

`_abc_impl = <_abc_data object>`

class `pip_shims.utils.BaseMethod` (*func_base, name, *args, **kwargs*)

Bases: `collections.abc.Callable`

`_abc_impl = <_abc_data object>`

`pip_shims.utils._parse(version)`

`pip_shims.utils.add_mixin_to_class(basecls, mixins)`

Given a class, adds the provided mixin classes as base classes and gives a new class

Parameters

- **basecls** (*Type*) – An initial class to generate a new class from
- **mixins** (*List [Type]*) – A list of mixins to add as base classes

Returns A new class with the provided mixins as base classes

Return type `Type[basecls, *mixins]`

`pip_shims.utils.apply_alias(imported, target, *aliases)`

Given a target with attributes, point non-existent aliases at the first existing one

Parameters

- **Type] imported** (*Union[ModuleType,]*) – A Module or Class base
- **target** (*Any*) – The target which is a member of **imported** and will have aliases
- **aliases** (*str*) – A list of aliases, the first found attribute will be the basis for all non-existent names which will be created as pointers

Returns The original target

Return type Any

`pip_shims.utils.call_function_with_correct_args(fn, **provided_kwargs)`

Determines which arguments from `provided_kwargs` to call `fn` and calls it.

Consumes a list of allowed arguments (e.g. from `getargs()`) and uses it to determine which of the arguments in the provided kwarg should be passed through to the given callable.

Parameters

- **fn** (*Callable*) – A callable which has some dynamic arguments
- **allowed_args** (*List[str]*) – A list of allowed arguments which can be passed to the supplied function

Returns The result of calling the function

Return type Any

`pip_shims.utils.ensure_function(parent, funcname, func)`

Given a module, a function name, and a function object, attaches the given function to the module and ensures it is named properly according to the provided argument

Parameters

- **parent** (*Any*) – The parent to attach the function to
- **funcname** (*str*) – The name to give the function
- **func** (*Callable*) – The function to rename and attach to **parent**

Returns The function with its name, qualname, etc set to mirror **parent**

Return type Callable

`pip_shims.utils.fallback_is_artifact(self)`

`pip_shims.utils.fallback_is_file_url(link)`

`pip_shims.utils.fallback_is_vcs(self)`

`pip_shims.utils.filter_allowed_args(fn, **provided_kwargs)`

Given a function and a kwarg mapping, return only those kwargs used in the function.

Parameters

- **fn** (*Callable*) – A function to inspect
- **Any** **kwargs** (*Dict[str, Any]*) – A mapping of kwargs to filter

Returns A new, filtered kwarg mapping

Return type Tuple[List[Any], Dict[str, Any]]

`pip_shims.utils.get_allowed_args(fn_or_class)`

Given a callable or a class, returns the arguments and default kwargs passed in.

Parameters **Type** **fn_or_class** (*Union[Callable, Any]*) – A function, method or class to inspect.

Returns A 2-tuple with a list of arguments and a dictionary of keywords mapped to default values.

Return type Tuple[List[str], Dict[str, Any]]

`pip_shims.utils.get_method_args(target_method)`

Returns the arguments for a callable.

Parameters **target_method** (*Callable*) – A callable to retrieve arguments for

Returns A 2-tuple of the original callable and its resulting arguments

Return type Tuple[Callable, Optional[inspect.Arguments]]

`pip_shims.utils.has_property` (*target*, *name*)

`pip_shims.utils.make_classmethod` (*fn*)

`pip_shims.utils.make_method` (*fn*)

`pip_shims.utils.memoize` (*obj*)

`pip_shims.utils.nullcontext` (**args*, ***kwargs*)

`pip_shims.utils.parse_version` (*version*)

`pip_shims.utils.resolve_possible_shim` (*target*)

`pip_shims.utils.set_default_kwargs` (*basecls*, *method*, **args*, ***default_kwargs*)

`pip_shims.utils.split_package` (*module*, *subimport=None*)

Used to determine what target to import.

Either splits off the final segment or uses the provided sub-import to return a 2-tuple of the import path and the target module or sub-path.

Parameters

- **module** (*str*) – A package to import from
- **subimport** (*Optional[str]*) – A class, function, or subpackage to import

Returns A 2-tuple of the corresponding import package and sub-import path

Return type Tuple[str, str]

Example

```
>>> from pip_shims.utils import split_package
>>> split_package("pip._internal.req.req_install", subimport="InstallRequirement")
("pip._internal.req.req_install", "InstallRequirement")
>>> split_package("pip._internal.cli.base_command")
("pip._internal.cli", "base_command")
```

`pip_shims.utils.suppress_setattr` (*obj*, *attr*, *value*, *filter_none=False*)

Set an attribute, suppressing any exceptions and skipping the attempt on failure.

Parameters

- **obj** (*Any*) – Object to set the attribute on
- **attr** (*str*) – The attribute name to set
- **value** (*Any*) – The value to set the attribute to
- **filter_none** (*bool*) – [description], defaults to False

Returns Nothing

Return type None

Example

```
>>> class MyClass(object):
...     def __init__(self, name):
...         self.name = name
...         self.parent = None
...     def __repr__(self):
```

(continues on next page)

(continued from previous page)

```

...         return "<{0!r}> instance (name={1!r}, parent={2!r})>".format (
...             self.__class__.__name__, self.name, self.parent
...         )
...     def __str__(self):
...         return self.name
>>> me = MyClass("Dan")
>>> dad = MyClass("John")
>>> grandfather = MyClass("Joe")
>>> suppress_setattr(dad, "parent", grandfather)
>>> dad
<'MyClass' instance (name='John', parent=<'MyClass' instance (name='Joe',
↳parent=None
)>>>
>>> suppress_setattr(me, "parent", dad)
>>> me
<'MyClass' instance (name='Dan', parent=<'MyClass' instance (name='John', parent=<
↳'My
Class' instance (name='Joe', parent=None)>>>)>>
>>> suppress_setattr(me, "grandparent", grandfather)
>>> me
<'MyClass' instance (name='Dan', parent=<'MyClass' instance (name='John', parent=<
↳'My
Class' instance (name='Joe', parent=None)>>>)>>

```

2.1.4 pip_shims.shims

Main module with magic self-replacement mechanisms to handle import speedups.

`pip_shims.shims._strip_extras` (*path*)

class `pip_shims.shims.SessionCommandMixin`

Bases: `pip._internal.cli.command_context.CommandContextMixIn`

A class mixin for command classes needing `_build_session()`.

`_build_session` (*options*, *retries=None*, *timeout=None*)

classmethod `_get_index_urls` (*options*)

Return a list of index urls from user-provided options.

`enter_context` (*context_provider*)

`get_default_session` (*options*)

Get a default-managed session.

`main_context` ()

class `pip_shims.shims.Command` (*name='Default pip command.'*, *summary='PipCommand'*, *isolated='Default pip command.'*)

Bases: `pip._internal.cli.base_command.Command`, `pip._internal.cli.req_command.SessionCommandMixin`

`_build_session` (*options*, *retries=None*, *timeout=None*)

classmethod `_get_index_urls` (*options*)

Return a list of index urls from user-provided options.

`_main` (*args*)

`add_options` ()

enter_context (*context_provider*)

get_default_session (*options*)

Get a default-managed session.

handle_pip_version_check (*options*)

This is a no-op so that commands by default do not do the pip version check.

ignore_require_venv = **False**

main (*args*)

main_context ()

parse_args (*args*)

run (*options, args*)

usage = **None**

class pip_shims.shims.**ConfigOptionParser** (**args, **kwargs*)

Bases: pip._internal.cli.parser.CustomOptionParser

Custom option parser which updates its defaults by checking the configuration files and environmental variables

_add_help_option ()

_add_version_option ()

_check_conflict (*option*)

_create_option_list ()

_create_option_mappings ()

_get_all_options ()

_get_args (*args*)

_get_ordered_configuration_items ()

_init_parsing_state ()

_match_long_opt (*opt : string*) → string

Determine which long option string 'opt' matches, ie. which one it is an unambiguous abbreviation for.

Raises `BadOptionError` if 'opt' doesn't unambiguously match any long option string.

_populate_option_list (*option_list, add_help=True*)

_process_args (*largs, rargs, values*)

_process_args(*largs* [[string],] *rargs* : [string], *values* : Values)

Process command-line arguments and populate 'values', consuming options and arguments from 'rargs'.

If 'allow_interspersed_args' is false, stop at the first non-option argument. If true, accumulate any interspersed non-option arguments in 'largs'.

_process_long_opt (*rargs, values*)

_process_short_opts (*rargs, values*)

_share_option_mappings (*parser*)

_update_defaults (*defaults*)

Updates the given defaults with values from the config files and the environ. Does a little special handling for certain types of options (lists).

add_option (*Option*)
 add_option(opt_str, ..., kwarg=val, ...)

add_option_group (**args, **kwargs*)

add_options (*option_list*)

check_default (*option, key, val*)

check_values (*values : Values, args : [string]*)
 -> (*values : Values, args : [string]*)

Check that the supplied option values and leftover arguments are valid. Returns the option values and leftover arguments (possibly adjusted, possibly completely new – whatever you like). Default implementation just returns the passed-in values; subclasses may override as desired.

destroy ()

Declare that you are done with this OptionParser. This cleans up reference cycles so the OptionParser (and all objects referenced by it) can be garbage-collected promptly. After calling destroy(), the OptionParser is unusable.

disable_interspersed_args ()

Set parsing to stop on the first non-option. Use this if you have a command processor which runs another command that has options of its own and you want to make sure these options don't get confused.

enable_interspersed_args ()

Set parsing to not stop on the first non-option, allowing interspersing switches with command arguments. This is the default behavior. See also disable_interspersed_args() and the class documentation description of the attribute allow_interspersed_args.

error (*msg : string*)

Print a usage message incorporating 'msg' to stderr and exit. If you override this in a subclass, it should not return – it should either exit or raise an exception.

exit (*status=0, msg=None*)

expand_prog_name (*s*)

format_description (*formatter*)

format_epilog (*formatter*)

format_help (*formatter=None*)

format_option_help (*formatter=None*)

get_default_values ()

Overriding to make updating the defaults after instantiation of the option parser possible, _update_defaults() does the dirty work.

get_description ()

get_option (*opt_str*)

get_option_group (*opt_str*)

get_prog_name ()

get_usage ()

get_version ()

has_option (*opt_str*)

insert_option_group (*idx, *args, **kwargs*)

Insert an OptionGroup at a given position.

option_list_all

Get a list of all options, including those in option groups.

parse_args (*args=None, values=None*)

parse_args(args [[string] = sys.argv[1:], values : Values = None)

-> (values : Values, args : [string])

Parse the command-line options found in 'args' (default: sys.argv[1:]). Any errors result in a call to 'error()', which by default prints the usage message to stderr and calls sys.exit() with an error message. On success returns a pair (values, args) where 'values' is a Values instance (with all your option values) and 'args' is the list of arguments left over after parsing options.

print_help (*file : file = stdout*)

Print an extended help message, listing all options and any help text provided with them, to 'file' (default stdout).

print_usage (*file : file = stdout*)

Print the usage message for the current program (self.usage) to 'file' (default stdout). Any occurrence of the string "%prog" in self.usage is replaced with the name of the current program (basename of sys.argv[0]). Does nothing if self.usage is empty or not defined.

print_version (*file : file = stdout*)

Print the version message for this program (self.version) to 'file' (default stdout). As with print_usage(), any occurrence of "%prog" in self.version is replaced by the current program's name. Does nothing if self.version is empty or undefined.

remove_option (*opt_str*)

set_conflict_handler (*handler*)

set_default (*dest, value*)

set_defaults (***kwargs*)

set_description (*description*)

set_process_default_values (*process*)

set_usage (*usage*)

standard_option_list = []

exception pip_shims.shims.DistributionNotFound

Bases: pip._internal.exceptions.InstallationError

Raised when a distribution cannot be found to satisfy a requirement

args

with_traceback ()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class pip_shims.shims.FormatControl (*no_binary=None, only_binary=None*)

Bases: object

Helper for managing formats from which a package can be installed.

disallow_binaries ()

get_allowed_formats (*canonical_name*)

static **handle_mutual_excludes** (*value, target, other*)

no_binary

only_binary

class pip_shims.shims.FrozenRequirement (*name, req, editable, comments=()*)

Bases: object

classmethod from_dist (*dist*)

pip_shims.shims.get_installed_distributions (*local_only=True, skip={'argparse', 'python', 'wsgiref'}, include_editables=True, editables_only=False, user_only=False, paths=None*)

Return a list of installed Distribution objects.

If *local_only* is True (default), only return installations local to the current virtualenv, if in a virtualenv.

skip argument is an iterable of lower-case project names to ignore; defaults to `stdlib_pkgs`

If *include_editables* is False, don't report editables.

If *editables_only* is True, only report editables.

If *user_only* is True, only report installations in the user site directory.

If *paths* is set, only report the distributions present at the specified list of locations.

exception pip_shims.shims.InstallationError

Bases: pip._internal.exceptions.PipError

General exception during installation

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pip_shims.shims.UninstallationError

Bases: pip._internal.exceptions.PipError

General exception during uninstallation

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pip_shims.shims.RequirementsFileParseError

Bases: pip._internal.exceptions.InstallationError

Raised when a general error occurs parsing a requirements file line.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pip_shims.shims.BestVersionAlreadyInstalled

Bases: pip._internal.exceptions.PipError

Raised when the most up-to-date version of a package is already installed.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.shims.BadCommand`

Bases: `pip._internal.exceptions.PipError`

Raised when virtualenv or a command is not found

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.shims.CommandError`

Bases: `pip._internal.exceptions.PipError`

Raised when there is an error in command-line arguments

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.shims.PreviousBuildDirError`

Bases: `pip._internal.exceptions.PipError`

Raised when there's a previous conflicting build directory

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

`pip_shims.shims.install_req_from_editable` (*editable_req*, *comes_from=None*,
use_pep517=None, *isolated=False*,
options=None, *constraint=False*,
user_supplied=False)

`pip_shims.shims.install_req_from_line` (*name*, *comes_from=None*, *use_pep517=None*,
isolated=False, *options=None*, *constraint=False*,
line_source=None, *user_supplied=False*)

Creates an InstallRequirement from a name, which might be a requirement, directory containing 'setup.py', filename, or URL.

Parameters *line_source* – An optional string describing where the line is from, for logging purposes in case of an error.

`pip_shims.shims.install_req_from_req_string` (*req_string*, *comes_from=None*, *iso-*
lated=False, *use_pep517=None*,
user_supplied=False)

class `pip_shims.shims.InstallRequirement` (*req*, *comes_from*, *editable=False*, *link=None*,
markers=None, *use_pep517=None*, *iso-*
lated=False, *install_options=None*,
global_options=None, *hash_options=None*, *con-*
straint=False, *extras=()*, *user_supplied=False*)

Bases: `pip._internal.req.req_install.InstallRequirement`

_generate_metadata()

Invokes metadata generator functions, with the required arguments.

_get_archive_name (*path*, *parentdir*, *rootdir*)

_set_requirement()

Set requirement after generating metadata.

archive (*build_dir*)

Saves archive to provided build_dir.

Used for saving downloaded VCS requirements as part of *pip download*.

assert_source_matches_version ()

build_location (*build_dir, autodelete, parallel_builds*)

check_if_exists (*use_user_site*)

Find an installed distribution that satisfies or conflicts with this requirement, and set self.satisfied_by or self.should_reinstall appropriately.

ensure_build_location (*build_dir, autodelete, parallel_builds*)

ensure_has_source_dir (*parent_dir, autodelete=False, parallel_builds=False*)

Ensure that a source_dir is set.

This will create a temporary build dir if the name of the requirement isn't known yet.

Parameters **parent_dir** – The ideal pip parent_dir for the source_dir. Generally src_dir for editables and build_dir for sdist.

Returns self.source_dir

format_debug ()

An un-tested helper for getting state, for debugging.

from_editable = `<pip_shims.utils.BaseMethod object>`

from_line = `<pip_shims.utils.BaseMethod object>`

from_path ()

Format a nice indicator to show where this “comes from”

get_dist ()

has_hash_options

Return whether any known-good hashes are specified as options.

These activate `--require-hashes` mode; hashes specified as part of a URL do not.

hashes (*trust_internet=True*)

Return a hash-comparer that considers my option- and URL-based hashes to be known-good.

Hashes in URLs—ones embedded in the requirements file, not ones downloaded from an index server—are almost peers with ones from flags. They satisfy `--require-hashes` (whether it was implicitly or explicitly activated) but do not activate it. md5 and sha224 are not allowed in flags, which should nudge people toward good algos. We always OR all hashes together, even ones from URLs.

Parameters **trust_internet** – Whether to trust URL-based (`#md5=...`) hashes downloaded from the internet, as by `populate_link()`

install (*install_options, global_options=None, root=None, home=None, prefix=None, warn_script_location=True, use_user_site=False, pycompile=True*)

installed_version

is_pinned

Return whether I am pinned to an exact version.

For example, `some-package==1.2` is pinned; `some-package>1.2` is not.

is_wheel

load_pyproject_toml ()

Load the pyproject.toml file.

After calling this routine, all of the attributes related to PEP 517 processing for this requirement have been set. In particular, the `use_pep517` attribute can be used to determine whether we should follow the PEP 517 or legacy (`setup.py`) code path.

match_markers (*extras_requested=None*)

metadata

name

prepare_metadata ()

Ensure that project metadata is available.

Under PEP 517, call the backend hook to prepare the metadata. Under legacy processing, call `setup.py egg-info`.

pyproject_toml_path

setup_py_path

specifier

uninstall (*auto_confirm=False, verbose=False*)

Uninstall the distribution currently satisfying this requirement.

Prompts before removing or modifying files unless `auto_confirm` is `True`.

Refuses to delete or modify files outside of `sys.prefix` - thus uninstallation within a virtual environment can only modify that virtual environment, even if the virtualenv is linked to global site-packages.

unpacked_source_directory

update_editable (*obtain=True*)

warn_on_mismatching_name ()

`pip_shims.shims.is_archive_file` (*name*)

Return `True` if *name* is considered as an archive file.

`pip_shims.shims.is_file_url` (*link*)

class `pip_shims.shims.Downloader` (*session, progress_bar*)

Bases: `object`

`pip_shims.shims.unpack_url` (*link, location, downloader, download_dir=None, hashes=None*)

Unpack link into location, downloading if required.

Parameters `hashes` – A Hashes object, one of whose embedded hashes must match, or `HashMismatch` will be raised. If the Hashes is empty, no matches are required, and unhashable types of requirements (like VCS ones, which would ordinarily raise `HashUnsupported`) are allowed.

`pip_shims.shims.is_installable_dir` (*path*)

Is path a directory containing `setup.py` or `pyproject.toml`?

class `pip_shims.shims.Link` (*url, comes_from=None, requires_python=None, yanked_reason=None, cache_link_parsing=True*)

Bases: `pip._internal.models.link.Link`

Parameters

- **url** – url of the resource pointed to (href of the link)
- **comes_from** – instance of `HTMLPage` where the link was found, or string.

- **requires_python** – String containing the *Requires-Python* metadata field, specified in PEP 345. This may be specified by a `data-requires-python` attribute in the HTML link tag, as described in PEP 503.
- **yanked_reason** – the reason the file has been yanked, if the file has been yanked, or `None` if the file hasn't been yanked. This is the value of the “data-yanked” attribute, if present, in a simple repository HTML link. If the file has been yanked but no reason was provided, this should be the empty string. See PEP 592 for more information and the specification.
- **cache_link_parsing** – A flag that is used elsewhere to determine whether resources retrieved from this link should be cached. PyPI index urls should generally have this set to `False`, for example.

```

_compare (other, method)
_compare_key
_defining_class
_egg_fragment_re = re.compile('[#&]egg=(^[^&]*)')
_hash_re = re.compile('(sha1|sha224|sha384|sha256|sha512|md5)=([a-f0-9]+)')
_parsed_url
_subdirectory_fragment_re = re.compile('[#&]subdirectory=(^[^&]*)')
_url
cache_link_parsing
comes_from
egg_fragment
ext
file_path
filename
has_hash
hash
hash_name
is_artifact
is_existing_dir()
is_file
is_hash_allowed (hashes)
    Return True if the link has a hash and it is allowed.
is_vcs
is_wheel
is_yanked
netloc
    This can contain auth information.
path
requires_python

```

scheme
show_url
splitext()
subdirectory_fragment
url
url_without_fragment
yanked_reason

`pip_shims.shims.make_abstract_dist(install_req)`
 Returns a Distribution for the given InstallRequirement

`pip_shims.shims.make_distribution_for_install_requirement(install_req)`
 Returns a Distribution for the given InstallRequirement

`pip_shims.shims.make_option_group(group, parser)`
 Return an OptionGroup object group – assumed to be dict with ‘name’ and ‘options’ keys parser – an optparse Parser

class `pip_shims.shims.PackageFinder`(*link_collector, target_python, allow_yanked, format_control=None, candidate_prefs=None, ignore_requires_python=None*)

Bases: `object`

This finds packages.

This is meant to match easy_install’s technique for looking for packages, by reading pages and looking for appropriate links.

This constructor is primarily meant to be used by the create() class method and from tests.

Parameters

- **format_control** – A FormatControl object, used to control the selection of source packages / binary packages when consulting the index and links.
- **candidate_prefs** – Options to use when creating a CandidateEvaluator object.

`_log_skipped_link(link, reason)`

`_sort_links(links)`

Returns elements of links in order, non-egg links first, egg links second, while eliminating duplicates

allow_all_prereleases

classmethod `create(link_collector, selection_prefs, target_python=None)`

Create a PackageFinder.

Parameters

- **selection_prefs** – The candidate selection preferences, as a SelectionPreferences object.
- **target_python** – The target Python interpreter to use when checking compatibility. If None (the default), a TargetPython object will be constructed from the running Python.

`evaluate_links(link_evaluator, links)`

Convert links that are candidates to InstallationCandidate objects.

`find_all_candidates(project_name)`

Find all available InstallationCandidate for project_name

This checks `index_urls` and `find_links`. All versions found are returned as an `InstallationCandidate` list.

See `LinkEvaluator.evaluate_link()` for details on which files are accepted.

find_best_candidate (*project_name, specifier=None, hashes=None*)

Find matches for the given project and specifier.

Parameters **specifier** – An optional object implementing *filter* (e.g. `packaging.specifiers.SpecifierSet`) to filter applicable versions.

Returns A `BestCandidateResult` instance.

find_links

find_requirement (*req, upgrade*)

Try to find a `Link` matching `req`

Expects `req`, an `InstallRequirement` and `upgrade`, a boolean Returns a `InstallationCandidate` if found, Raises `DistributionNotFound` or `BestVersionAlreadyInstalled` otherwise

get_install_candidate (*link_evaluator, link*)

If the `link` is a candidate for install, convert it to an `InstallationCandidate` and return it. Otherwise, return `None`.

index_urls

make_candidate_evaluator (*project_name, specifier=None, hashes=None*)

Create a `CandidateEvaluator` object to use.

make_link_evaluator (*project_name*)

prefer_binary

process_project_url (*project_url, link_evaluator*)

search_scope

set_allow_all_prereleases ()

set_prefer_binary ()

target_python

trusted_hosts

class `pip_shims.shims.CandidateEvaluator` (*project_name, supported_tags, specifier, prefer_binary=False, allow_all_prereleases=False, hashes=None*)

Bases: `object`

Responsible for filtering and sorting candidates for installation based on what tags are valid.

Parameters **supported_tags** – The PEP 425 tags supported by the target Python in order of preference (most preferred first).

__sort_key (*candidate*)

Function to pass as the *key* argument to a call to `sorted()` to sort `InstallationCandidates` by preference.

Returns a tuple such that tuples sorting as greater using Python's default comparison operator are more preferred.

The preference is as follows:

First and foremost, candidates with allowed (matching) hashes are always preferred over candidates without matching hashes. This is because e.g. if the only candidate with an allowed hash is yanked, we still want to use that candidate.

Second, excepting hash considerations, candidates that have been yanked (in the sense of PEP 592) are always less preferred than candidates that haven't been yanked. Then:

If not finding wheels, they are sorted by version only. If finding wheels, then the sort order is by version, then:

1. existing installs
2. wheels ordered via `Wheel.support_index_min(self._supported_tags)`
3. source archives

If `prefer_binary` was set, then all wheels are sorted above sources.

Note: it was considered to embed this logic into the Link comparison operators, but then different sdist links with the same version, would have to be considered equal

compute_best_candidate (*candidates*)

Compute and return a *BestCandidateResult* instance.

classmethod create (*project_name*, *target_python=None*, *prefer_binary=False*, *allow_all_prereleases=False*, *specifier=None*, *hashes=None*)

Create a *CandidateEvaluator* object.

Parameters

- **target_python** – The target Python interpreter to use when checking compatibility. If None (the default), a *TargetPython* object will be constructed from the running Python.
- **specifier** – An optional object implementing *filter* (e.g. *packaging.specifiers.SpecifierSet*) to filter applicable versions.
- **hashes** – An optional collection of allowed hashes.

get_applicable_candidates (*candidates*)

Return the applicable candidates from a list of candidates.

sort_best_candidate (*candidates*)

Return the best candidate per the instance's sort order, or None if no candidate is acceptable.

class `pip_shims.shims.CandidatePreferences` (*prefer_binary=False*, *allow_all_prereleases=False*)

Bases: `object`

Encapsulates some of the preferences for filtering and sorting *InstallationCandidate* objects.

Parameters `allow_all_prereleases` – Whether to allow all pre-releases.

class `pip_shims.shims.LinkCollector` (*session*, *search_scope*)

Bases: `object`

Responsible for collecting *Link* objects from all configured locations, making network requests as needed.

The class's main method is its `collect_links()` method.

collect_links (*project_name*)

Find all available links for the given project name.

Returns All the *Link* objects (unfiltered), as a *CollectedLinks* object.

classmethod create (*session*, *options*, *suppress_no_index=False*)

Parameters

- **session** – The *Session* to use to make requests.

- **suppress_no_index** – Whether to ignore the `--no-index` option when constructing the `SearchScope` object.

fetch_page (*location*)

Fetch an HTML page containing package links.

find_links

class `pip_shims.shims.LinkEvaluator` (*project_name, canonical_name, formats, target_python, allow_yanked, ignore_requires_python=None*)

Bases: `object`

Responsible for evaluating links for a particular project.

Parameters

- **project_name** – The user supplied package name.
- **canonical_name** – The canonical package name.
- **formats** – The formats allowed for this package. Should be a set with ‘binary’ or ‘source’ or both in it.
- **target_python** – The target Python interpreter to use when evaluating link compatibility. This is used, for example, to check wheel compatibility, as well as when checking the Python version, e.g. the Python version embedded in a link filename (or egg fragment) and against an HTML link’s optional PEP 503 “data-requires-python” attribute.
- **allow_yanked** – Whether files marked as yanked (in the sense of PEP 592) are permitted to be candidates for install.
- **ignore_requires_python** – Whether to ignore incompatible PEP 503 “data-requires-python” values in HTML links. Defaults to False.

`_py_version_re = re.compile('-py([123]\\.|[0-9]?)$')`

evaluate_link (*link*)

Determine whether a link is a candidate for installation.

Returns A tuple (`is_candidate, result`), where *result* is (1) a version string if *is_candidate* is True, and (2) if *is_candidate* is False, an optional string to log the reason the link fails to qualify.

class `pip_shims.shims.TargetPython` (*platform=None, py_version_info=None, abi=None, implementation=None*)

Bases: `object`

Encapsulates the properties of a Python interpreter one is targeting for a package install, download, etc.

Parameters

- **platform** – A string or None. If None, searches for packages that are supported by the current system. Otherwise, will find packages that can be built on the platform passed in. These packages will only be downloaded for distribution: they will not be built locally.
- **py_version_info** – An optional tuple of ints representing the Python version information to use (e.g. `sys.version_info[:3]`). This can have length 1, 2, or 3 when provided.
- **abi** – A string or None. This is passed to `compatibility_tags.py`’s `get_supported()` function as is.
- **implementation** – A string or None. This is passed to `compatibility_tags.py`’s `get_supported()` function as is.

`_given_py_version_info`

`_valid_tags`

abi

format_given()

Format the given, non-None attributes for display.

get_tags()

Return the supported PEP 425 tags to check wheel candidates against.

The tags are returned in order of preference (most preferred first).

implementation

platform

py_version

py_version_info

class pip_shims.shims.**SearchScope** (*find_links, index_urls*)

Bases: `object`

Encapsulates the locations that pip is configured to search.

classmethod **create** (*find_links, index_urls*)

Create a SearchScope object after normalizing the *find_links*.

find_links

get_formatted_locations()

get_index_urls_locations (*project_name*)

Returns the locations found via self.index_urls

Checks the url_name on the main (first in the list) index and use this url_name to produce all locations

index_urls

class pip_shims.shims.**SelectionPreferences** (*allow_yanked, allow_all_prereleases=False, format_control=None, prefer_binary=False, ignore_requires_python=None*)

Bases: `object`

Encapsulates the candidate selection preferences for downloading and installing files.

Create a SelectionPreferences object.

Parameters

- **allow_yanked** – Whether files marked as yanked (in the sense of PEP 592) are permitted to be candidates for install.
- **format_control** – A FormatControl object or None. Used to control the selection of source packages / binary packages when consulting the index and links.
- **prefer_binary** – Whether to prefer an old, but valid, binary dist over a new source dist.
- **ignore_requires_python** – Whether to ignore incompatible “Requires-Python” values in links. Defaults to False.

allow_all_prereleases

allow_yanked

format_control

ignore_requires_python

prefer_binary

`pip_shims.shims.parse_requirements` (*filename*, *session*, *finder=None*, *comes_from=None*, *options=None*, *constraint=False*)

Parse a requirements file and yield `ParsedRequirement` instances.

Parameters

- **filename** – Path or url of requirements file.
- **session** – `PipSession` instance.
- **finder** – Instance of `pip.index.PackageFinder`.
- **comes_from** – Origin description of requirements.
- **options** – cli options.
- **constraint** – If true, parsing a constraint file rather than requirements file.

`pip_shims.shims.path_to_url` (*path*)

Convert a path to a file: URL. The path will be made absolute and have quoted path parts.

exception `pip_shims.shims.PipError`

Bases: `Exception`

Base pip exception

args

with_traceback ()

`Exception.with_traceback(tb)` – set `self.__traceback__` to `tb` and return `self`.

class `pip_shims.shims.RequirementPreparer` (*build_dir*, *download_dir*, *src_dir*,
wheel_download_dir, *build_isolation*,
req_tracker, *downloader*, *finder*, *require_hashes*, *use_user_site*)

Bases: `object`

Prepares a Requirement

`__download_should_save`

`__ensure_link_req_src_dir` (*req*, *download_dir*, *parallel_builds*)

Ensure `source_dir` of a linked `InstallRequirement`.

`__get_linked_req_hashes` (*req*)

`__log_preparing_link` (*req*)

Log the way the link prepared.

`prepare_editable_requirement` (*req*)

Prepare an editable requirement

`prepare_installed_requirement` (*req*, *skip_reason*)

Prepare an already-installed requirement

`prepare_linked_requirement` (*req*, *parallel_builds=False*)

Prepare a requirement to be obtained from `req.link`.

class `pip_shims.shims.RequirementSet` (*check_supported_wheels=True*)

Bases: `object`

Create a `RequirementSet`.

`add_named_requirement` (*install_req*)

`add_requirement` (*install_req*, *parent_req_name=None*, *extras_requested=None*)

Add `install_req` as a requirement to install.

Parameters

- **parent_req_name** – The name of the requirement that needed this added. The name is used because when multiple unnamed requirements resolve to the same name, we could otherwise end up with dependency links that point outside the Requirements set. `parent_req` must already be added. Note that `None` implies that this is a user supplied requirement, vs an inferred one.
- **extras_requested** – an iterable of extras used to evaluate the environment markers.

Returns Additional requirements to scan. That is either `[]` if the requirement is not applicable, or `[install_req]` if the requirement is applicable and has just been added.

add_unnamed_requirement (*install_req*)

all_requirements

get_requirement (*name*)

has_requirement (*name*)

class `pip_shims.shims.RequirementTracker` (*root*)

Bases: `object`

_entry_path (*link*)

add (*req*)

Add an `InstallRequirement` to build tracking.

cleanup ()

remove (*req*)

Remove an `InstallRequirement` from build tracking.

track (*req*)

class `pip_shims.shims.TempDirectory` (*path=None, delete=<pip._internal.utils.temp_dir._Default object>, kind='temp', globally_managed=False*)

Bases: `object`

Helper class that owns and cleans up a temporary directory.

This class can be used as a context manager or as an OO representation of a temporary directory.

Attributes:

path Location to the created temporary directory

delete Whether the directory should be deleted when exiting (when used as a contextmanager)

Methods:

cleanup() Deletes the temporary directory

When used as a context manager, if the `delete` attribute is `True`, on exiting the context the temporary directory is deleted.

_create (*kind*)

Create a temporary directory and store its path in `self.path`

cleanup ()

Remove the temporary directory created and reset state

path

`pip_shims.shims.global_tempdir_manager` ()

```
pip_shims.shims.shim_unpack(* , unpack_fn=<pip_shims.models.ShimmedPathCollection object>,
                             download_dir, tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection
                             object>, ireq=None, link=None, location=None, hashes=None,
                             progress_bar='off', only_download=None, down-
                             loader_provider=<pip_shims.models.ShimmedPathCollection
                             object>, session=None)
```

Accepts all parameters that have been valid to pass to `pip._internal.download.unpack_url()` and selects or drops parameters as needed before invoking the provided callable.

Parameters

- **unpack_fn** (*Callable*) – A callable or shim referring to the pip implementation
- **download_dir** (*str*) – The directory to download the file to
- **tempdir_manager_provider** (*TShimmedFunc*) – A callable or shim referring to `global_tempdir_manager` function from pip or a shimmed no-op context manager

:param Optional[InstallRequirement] ireq: an Install Requirement instance, defaults to None

:param Optional[Link] link: A Link instance, defaults to None.

Parameters

- **location** (*Optional[str]*) – A location or source directory if the target is a VCS url, defaults to None.
- **hashes** (*Optional[Any]*) – A Hashes instance, defaults to None
- **progress_bar** (*str*) – Indicates progress bar usage during download, defaults to off.
- **only_download** (*Optional[bool]*) – Whether to skip install, defaults to None.
- **downloader_provider** (*Optional[ShimmedPathCollection]*) – A downloader class to instantiate, if applicable.
- **session** (*Optional[Session]*) – A PipSession instance, defaults to None.

Returns The result of unpacking the url.

Return type None

```
pip_shims.shims.get_requirement_tracker()
```

```
class pip_shims.shims.Resolver(preparer, finder, wheel_cache, make_install_req, use_user_site,
                             ignore_dependencies, ignore_installed, ignore_requires_python,
                             force_reinstall, upgrade_strategy, py_version_info=None)
```

Bases: `pip._internal.resolution.base.BaseResolver`

Resolves which packages need to be installed/uninstalled to perform the requested operation without breaking the requirements of any package.

```
_allowed_strategies = {'eager', 'only-if-needed', 'to-satisfy-only'}
```

```
_check_skip_installed(req_to_install)
```

Check if `req_to_install` should be skipped.

This will check if the req is installed, and whether we should upgrade or reinstall it, taking into account all the relevant user options.

After calling this `req_to_install` will only have `satisfied_by` set to None if the `req_to_install` is to be upgraded/reinstalled etc. Any other value will be a dist recording the current thing installed that satisfies the requirement.

Note that for vcs urls and the like we can't assess skipping in this routine - we simply identify that we need to pull the thing down, then later on it is pulled down and introspected to assess upgrade/ reinstalls etc.

Returns A text reason for why it was skipped, or None.

`_find_requirement_link` (*req*)

`_get_abstract_dist_for` (*req*)

Takes a InstallRequirement and returns a single AbstractDist representing a prepared variant of the same.

`_is_upgrade_allowed` (*req*)

`_populate_link` (*req*)

Ensure that if a link can be found for this, that it is found.

Note that req.link may still be None - if the requirement is already installed and not needed to be upgraded based on the return value of `_is_upgrade_allowed()`.

If preparer.require_hashes is True, don't use the wheel cache, because cached wheels, always built locally, have different hashes than the files downloaded from the index server and thus throw false hash mismatches. Furthermore, cached wheels at present have undeterministic contents due to file modification times.

`_resolve_one` (*requirement_set, req_to_install*)

Prepare a single requirements file.

Returns A list of additional InstallRequirements to also install.

`_set_req_to_reinstall` (*req*)

Set a requirement to be installed.

`get_installation_order` (*req_set*)

Create the installation order.

The installation order is topological - requirements are installed before the requiring thing. We break cycles at an arbitrary point, and make no other guarantees.

`resolve` (*root_reqs, check_supported_wheels*)

Resolve what operations need to be done

As a side-effect of this method, the packages (and their dependencies) are downloaded, unpacked and prepared for installation. This preparation is done by `pip.operations.prepare`.

Once PyPI has static dependency metadata available, it would be possible to move the preparation to become a step separated from dependency resolution.

class `pip_shims.shims.SafeFileCache` (*directory*)

Bases: `pip._vendor.cachecontrol.cache.BaseCache`

A file based cache which is safe to use even when the target directory may not be accessible or writable.

`_get_cache_path` (*name*)

`close` ()

`delete` (*key*)

`get` (*key*)

`set` (*key, value*)

class `pip_shims.shims.UninstallPathSet` (*dist*)

Bases: `object`

A set of file paths to be removed in the uninstallation of a requirement.

`_allowed_to_proceed` (*verbose*)
 Display which files would be deleted and prompt for confirmation

`_permitted` (*path*)
 Return True if the given path is one we are permitted to remove/modify, False otherwise.

`add` (*path*)

`add_pth` (*pth_file, entry*)

`commit` ()
 Remove temporary save dir: rollback will no longer be possible.

`classmethod from_dist` (*dist*)

`remove` (*auto_confirm=False, verbose=False*)
 Remove paths in `self.paths` with confirmation (unless `auto_confirm` is True).

`rollback` ()
 Rollback the changes previously made by `remove()`.

`pip_shims.shims.url_to_path` (*url*)
 Convert a file: URL to a path.

`class pip_shims.shims.VcsSupport`

Bases: `object`

`_registry` = {'bzzr': <pip._internal.vcs.bazaar.Bazaar object>, 'git': <pip._internal.vcs.git.Git object>, 'hg': <pip._internal.vcs.hg.Hg object>, 'ssh': <pip._internal.vcs.ssh.Ssh object>, 'svn': <pip._internal.vcs.svn.Svn object>, 'sftp': <pip._internal.vcs.sftp.Sftp object>}

`all_schemes`

`backends`

`dirname`

`get_backend` (*name*)
 Return a VersionControl object or None.

`get_backend_for_dir` (*location*)
 Return a VersionControl object if a repository of that type is found at the given directory.

`get_backend_for_scheme` (*scheme*)
 Return a VersionControl object or None.

`register` (*cls*)

`schemes` = ['ssh', 'git', 'hg', 'bzzr', 'sftp', 'svn']

`unregister` (*name*)

`class pip_shims.shims.Wheel` (*filename*)

Bases: `object`

`get_formatted_file_tags` ()
 Return the wheel's tags as a sorted list of strings.

`support_index_min` (*tags*)
 Return the lowest index that one of the wheel's file_tag combinations achieves in the given list of supported tags.

For example, if there are 8 supported tags and one of the file tags is first in the list, then return 0.

Parameters `tags` – the PEP 425 tags to check the wheel against, in order with most preferred first.

Raises `ValueError` – If none of the wheel's file tags match one of the supported tags.

supported (*tags*)

Return whether the wheel is compatible with one of the given tags.

Parameters *tags* – the PEP 425 tags to check the wheel against.

```
wheel_file_re = re.compile('^(?P<namever>(?!P<name>.+)ate(?P<ver>.*)?)\n ((?!P<build>\\
```

class `pip_shims.shims.WheelCache` (*cache_dir, format_control*)

Bases: `pip._internal.cache.Cache`

Wraps EphemWheelCache and SimpleWheelCache into a single Cache

This Cache allows for gracefully degradation, using the ephem wheel cache when a certain link is not found in the simple wheel cache first.

__get_cache_path_parts (*link*)

Get parts of part that must be `os.path.join`d with `cache_dir`

__get_cache_path_parts_legacy (*link*)

Get parts of part that must be `os.path.join`d with `cache_dir`

Legacy cache key (pip < 20) for compatibility with older caches.

__get_candidates (*link, canonical_package_name*)

get (*link, package_name, supported_tags*)

Returns a link to a cached item if it exists, otherwise returns the passed link.

get_cache_entry (*link, package_name, supported_tags*)

Returns a CacheEntry with a link to a cached item if it exists or None. The cache entry indicates if the item was found in the persistent or ephemeral cache.

get_ephem_path_for_link (*link*)

get_path_for_link (*link*)

Return a directory to store cached items in for link.

get_path_for_link_legacy (*link*)

`pip_shims.shims.build` (*requirements, wheel_cache, build_options, global_options*)

Build wheels.

Returns The list of InstallRequirement that succeeded to build and the list of InstallRequirement that failed to build.

`pip_shims.shims.build_one` (*req, output_dir, build_options, global_options*)

Build one wheel.

Returns The filename of the built wheel, or None if the build failed.

`pip_shims.shims.build_one_inside_env` (*req, output_dir, build_options, global_options*)

class `pip_shims.shims.AbstractDistribution` (*req*)

Bases: `object`

A base class for handling installable artifacts.

The requirements for anything installable are as follows:

- we must be able to determine the requirement name (or we can't correctly handle the non-upgrade case).
- for packages with setup requirements, we must also be able to determine their requirements without installing additional packages (for the same reason as run-time dependencies)
- we must be able to create a Distribution object exposing the above metadata.

__abc_impl = `<_abc_data object>`

```
get_pkg_resources_distribution()
```

```
prepare_distribution_metadata (finder, build_isolation)
```

```
class pip_shims.shims.InstalledDistribution (req)
```

```
Bases: pip._internal.distributions.base.AbstractDistribution
```

Represents an installed package.

This does not need any preparation as the required information has already been computed.

```
_abc_impl = <_abc_data object>
```

```
get_pkg_resources_distribution()
```

```
prepare_distribution_metadata (finder, build_isolation)
```

```
class pip_shims.shims.SourceDistribution (req)
```

```
Bases: pip._internal.distributions.base.AbstractDistribution
```

Represents a source distribution.

The preparation step for these needs metadata for the packages to be generated, either using PEP 517 or using the legacy *setup.py egg_info*.

```
_abc_impl = <_abc_data object>
```

```
_setup_isolation (finder)
```

```
get_pkg_resources_distribution()
```

```
prepare_distribution_metadata (finder, build_isolation)
```

```
class pip_shims.shims.WheelDistribution (req)
```

```
Bases: pip._internal.distributions.base.AbstractDistribution
```

Represents a wheel distribution.

This does not need any preparation as wheels can be directly unpacked.

```
_abc_impl = <_abc_data object>
```

```
get_pkg_resources_distribution()
```

Loads the metadata from the wheel file into memory and returns a Distribution that uses it, not relying on the wheel file or requirement.

```
prepare_distribution_metadata (finder, build_isolation)
```

```
pip_shims.shims.wheel_cache (cache_dir=None, format_control=None, *,
                             wheel_cache_provider=<pip_shims.models.ShimmedPathCollection
                             object>, format_control_provider=<pip_shims.models.ShimmedPathCollection
                             object>, tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection
                             object>)
```

```
pip_shims.shims.get_package_finder (install_cmd=None, options=None, session=None,
                                    platform=None, python_versions=None,
                                    abi=None, implementation=None, target_python=None,
                                    ignore_requires_python=None, *,
                                    target_python_builder=<class
                                    'pip._internal.models.target_python.TargetPython'>,
                                    install_cmd_provider=<pip_shims.models.ShimmedPathCollection
                                    object>)
```

Shim for compatibility to generate package finders.

Build and return a PackageFinder instance using the InstallCommand helper method to construct the finder, shimmed with backports as needed for compatibility.

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_cmd** – A `InstallCommand` instance which is used to generate the finder.
- **options** (*optparse.Values*) – An optional `optparse.Values` instance generated by calling `install_cmd.parser.parse_args()` typically.
- **session** – An optional session instance, can be created by the `install_cmd`.
- **platform** (*Optional[str]*) – An optional platform string, e.g. `linux_x86_64`
- **python_versions** (*Optional[Tuple[str, ...]]*) – A tuple of 2-digit strings representing python versions, e.g. `(“27”, “35”, “36”, “37”...)`
- **abi** (*Optional[str]*) – The target abi to support, e.g. `“cp38”`
- **implementation** (*Optional[str]*) – An optional implementation string for limiting searches to a specific implementation, e.g. `“cp”` or `“py”`
- **target_python** – A `TargetPython` instance (will be translated to alternate arguments if necessary on incompatible pip versions).
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore `requires_python` on resulting candidates, only valid after pip version 19.3.1
- **target_python_builder** – A ‘`TargetPython`’ builder (e.g. the class itself, uninstantiated)

Returns A `pip._internal.index.package_finder.PackageFinder` instance

Return type `pip._internal.index.package_finder.PackageFinder`

Example

```
>>> from pip_shims.shims import InstallCommand, get_package_finder
>>> install_cmd = InstallCommand()
>>> finder = get_package_finder(
...     install_cmd, python_versions=("27", "35", "36", "37", "38"),
...     implementation="
cp"
... )
>>> candidates = finder.find_all_candidates("requests")
>>> requests_222 = next(iter(c for c in candidates if c.version.public == "2.22.0
... ))
>>> requests_222
<InstallationCandidate('requests', <Version('2.22.0')>, <Link https://files.
pythonhos
ted.org/packages/51/bd/
23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/r
equests-2.22.0-py2.py3-none-any.whl
#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9
a590f48c010551dc6c4b31 (from https://pypi.org/simple/requests/) (requires-python:>
=2.
7, !=3.0.*, !=3.1.*, !=3.2.*, !=3.3.*, !=3.4.*)>>>
```

```

pip_shims.shims.make_preparer(*, preparer_fn=<pip_shims.models.ShimmedPathCollection object>, req_tracker_fn=<pip_shims.models.ShimmedPathCollection object>, build_dir=None, src_dir=None, download_dir=None, wheel_download_dir=None, progress_bar='off', build_isolation=False, session=None, finder=None, options=None, require_hashes=None, use_user_site=None, req_tracker=None, install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>, downloader_provider=<pip_shims.models.ShimmedPathCollection object>, install_cmd=None, finder_provider=<pip_shims.models.ShimmedPathCollection object>)

```

Creates a requirement preparer for preparing pip requirements.

Provides a compatibility shim that accepts all previously valid arguments and discards any that are no longer used.

Raises

- **TypeError** – No requirement tracker provided and one cannot be generated
- **TypeError** – No valid sessions provided and one cannot be generated
- **TypeError** – No valid finders provided and one cannot be generated

Parameters

- **preparer_fn** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **req_tracker_fn** (*Optional[TShimmedFunc]*) – Callable or shim for generating requirement trackers, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **src_dir** (*Optional[str]*) – Directory to find or extract source files, defaults to None
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **progress_bar** (*str*) – Whether to display a progress bar, defaults to off
- **build_isolation** (*bool*) – Whether to build requirements in isolation, defaults to False
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **require_hashes** (*Optional[bool]*) – Whether to require hashes for preparation
- **use_user_site** (*Optional[bool]*) – Whether to use the user site directory for preparing requirements
- **TShimmedFunc**] **req_tracker** (*Optional[Union[TReqTracker,]]*) – The requirement tracker to use for building packages, defaults to None
- **downloader_provider** (*Optional[TShimmedFunc]*) – A downloader provider

- **install_cmd** (*Optional*[*TCommandInstance*]) – The install command used to create the finder, session, and options if needed, defaults to None
- **finder_provider** (*Optional*[*TShimmedFunc*]) – A package finder provider

Yield A new requirement preparer instance

Return type `ContextManager[RequirementPreparer]`

Example

```
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_tracker
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> with make_preparer(
...     options=pip_options, finder=finder, session=session, install_cmd=ic
... ) as preparer:
...     print(preparer)
<pip._internal.operations.prepare.RequirementPreparer object at 0x7f8a2734be80>
```

`pip_shims.shims.get_resolver` (*, *resolver_fn*=<pip_shims.models.ShimmedPathCollection object>, *install_req_provider*=<pip_shims.models.ShimmedPathCollection object>, *format_control_provider*=<pip_shims.models.ShimmedPathCollection object>, *wheel_cache_provider*=<pip_shims.models.ShimmedPathCollection object>, *finder*=None, *upgrade_strategy*='to-satisfy-only', *force_reinstall*=None, *ignore_dependencies*=None, *ignore_requires_python*=None, *ignore_installed*=True, *use_user_site*=False, *isolated*=None, *wheel_cache*=None, *preparer*=None, *session*=None, *options*=None, *make_install_req*=None, *install_cmd_provider*=<pip_shims.models.ShimmedPathCollection object>, *install_cmd*=None, *use_pep517*=True)

A resolver creation compatibility shim for generating a resolver.

Consumes any argument that was previously used to instantiate a resolver, discards anything that is no longer valid.

Note: This is only valid for **pip >= 10.0.0**

Raises

- **ValueError** – A session is required but not provided and one cannot be created
- **ValueError** – A finder is required but not provided and one cannot be created
- **ValueError** – An install requirement provider is required and has not been provided

Parameters

- **resolver_fn** (*TShimmedFunc*) – The resolver function used to create new resolver instances.

- **install_req_provider** (*TShimmedFunc*) – The provider function to use to generate install requirements if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a `format_control` instance if needed.
- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to `None`.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to `None`
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to `None`
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to `None`
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to `True`
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to `False`
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to `None`
- **wheel_cache** (*Optional[TWheelCache]*) – The wheel cache to use, defaults to `None`
- **preparer** (*Optional[TPreparer]*) – The requirement preparer to use, defaults to `None`
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to `None`
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to `None`
- **make_install_req** (*Optional[functools.partial]*) – The partial function to pass in to the resolver for actually generating install requirements, if necessary
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to `None`.
- **use_pep517** (*bool*) – Whether to use the pep517 build process.

Returns A new resolver instance.

Return type `Resolver`

Example

```
>>> import os
>>> from tempdir import TemporaryDirectory
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_
↪ tracker,
...     get_resolver, InstallRequirement, RequirementSet
```

(continues on next page)

(continued from previous page)

```

... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> wheel_cache = WheelCache(USER_CACHE_DIR, FormatControl(None, None))
>>> with TemporaryDirectory() as temp_base:
...     reqset = RequirementSet()
...     ireq = InstallRequirement.from_line("requests")
...     ireq.is_direct = True
...     build_dir = os.path.join(temp_base, "build")
...     src_dir = os.path.join(temp_base, "src")
...     ireq.build_location(build_dir)
...     with make_preparer(
...         options=pip_options, finder=finder, session=session,
...         build_dir=build_dir, install_cmd=install_cmd,
...     ) as preparer:
...         resolver = get_resolver(
...             finder=finder, ignore_dependencies=False, ignore_requires_
↪python=True,
...             preparer=preparer, session=session, options=pip_options,
...             install_cmd=install_cmd, wheel_cache=wheel_cache,
...         )
...         resolver.require_hashes = False
...         reqset.add_requirement(ireq)
...         results = resolver.resolve(reqset)
...         #reqset.cleanup_files()
...         for result_req in reqset.requirements:
...             print(result_req)
requests
chardet
certifi
urllib3
idna

```

```

pip_shims.shims.get_requirement_set(install_command=None, *,
                                   req_set_provider=<pip_shims.models.ShimmedPathCollection
                                   object>, build_dir=None, src_dir=None, down-
                                   load_dir=None, wheel_download_dir=None, ses-
                                   sion=None, wheel_cache=None, upgrade=False,
                                   upgrade_strategy=None, ignore_installed=False,
                                   ignore_dependencies=False, force_reinstall=False,
                                   use_user_site=False, isolated=False, ig-
                                   nore_requires_python=False, require_hashes=None,
                                   cache_dir=None, options=None, in-
                                   stall_cmd_provider=<pip_shims.models.ShimmedPathCollection
                                   object>, wheel_cache_provider=<pip_shims.models.ShimmedPathCollection
                                   object>)

```

Creates a requirement set from the supplied parameters.

Not all parameters are passed through for all pip versions, but any invalid parameters will be ignored if they are not needed to generate a requirement set on the current pip version.

:param *ShimmedPathCollection* wheel_cache_provider: A context manager provider which resolves to a *WheelCache* instance

Parameters `install_command` – A `InstallCommand` instance which is used to generate the finder.

:param `ShimmedPathCollection req_set_provider`: A **provider** to build requirement set instances.

Parameters

- **build_dir** (`str`) – The directory to build requirements in. Removed in pip 10, defaults to None
- **source_dir** (`str`) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (`str`) – The directory to download requirement artifacts to. Removed in pip 10, defaults to None
- **wheel_download_dir** (`str`) – The directory to download wheels to. Removed in pip 10, defaults to None

:param `Session session`: **The pip session to use. Removed in pip 10**, defaults to None

Parameters

- **wheel_cache** (`WheelCache`) – The pip `WheelCache` instance to use for caching wheels. Removed in pip 10, defaults to None
- **upgrade** (`bool`) – Whether to try to upgrade existing requirements. Removed in pip 10, defaults to False.
- **upgrade_strategy** (`str`) – The upgrade strategy to use, e.g. “only-if-needed”. Removed in pip 10, defaults to None.
- **ignore_installed** (`bool`) – Whether to ignore installed packages when resolving. Removed in pip 10, defaults to False.
- **ignore_dependencies** (`bool`) – Whether to ignore dependencies of requirements when resolving. Removed in pip 10, defaults to False.
- **force_reinstall** (`bool`) – Whether to force reinstall of packages when resolving. Removed in pip 10, defaults to False.
- **use_user_site** (`bool`) – Whether to use user site packages when resolving. Removed in pip 10, defaults to False.
- **isolated** (`bool`) – Whether to resolve in isolation. Removed in pip 10, defaults to False.
- **ignore_requires_python** (`bool`) – Removed in pip 10, defaults to False.
- **require_hashes** (`bool`) – Whether to require hashes when resolving. Defaults to False.
- **options** (`Values`) – An `Values` instance from an install cmd
- **install_cmd_provider** (`ShimmedPathCollection`) – A shim for providing new install command instances.

Returns A new requirement set instance

Return type `RequirementSet`

```

pip_shims.shims.resolve(ireq, *, reqset_provider=<pip_shims.models.ShimmedPathCollection object>, req_tracker_provider=<pip_shims.models.ShimmedPathCollection object>, install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>, install_command=None, finder_provider=<pip_shims.models.ShimmedPathCollection object>, resolver_provider=<pip_shims.models.ShimmedPathCollection object>, wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>, format_control_provider=<pip_shims.models.ShimmedPathCollection object>, make_preparer_provider=<pip_shims.models.ShimmedPathCollection object>, tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection object>, options=None, session=None, resolver=None, finder=None, upgrade_strategy='to-satisfy-only', force_reinstall=None, ignore_dependencies=None, ignore_requires_python=None, ignore_installed=True, use_user_site=False, isolated=None, build_dir=None, source_dir=None, download_dir=None, cache_dir=None, wheel_download_dir=None, wheel_cache=None, require_hashes=None, check_supported_wheels=True)

```

Resolves the provided **InstallRequirement**, returning a dictionary.

Maps a dictionary of names to corresponding `InstallRequirement` values.

:param InstallRequirement ireq: An `InstallRequirement` to initiate the resolution process

:param *ShimmedPathCollection* reqset_provider: A provider to build requirement set instances.

:param *ShimmedPathCollection* req_tracker_provider: A provider to build requirement tracker instances

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_command** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None.

:param *ShimmedPathCollection* finder_provider: A provider to package finder instances.

:param *ShimmedPathCollection* resolver_provider: A provider to build resolver instances

Parameters

- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a `format_control` instance if needed.
- **make_preparer_provider** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **tempdir_manager_provider** (*Optional[TShimmedFunc]*) – Shim for generating tempdir manager for pip temporary directories
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None

:param Resolver resolver: A pre-existing resolver instance to use for resolution

Parameters

- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to None
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to None
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to None
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to True
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to False
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **cache_dir** (*str*) – The cache directory to use for caching artifacts during resolution
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **wheel_cache** (*Optional[TWheelCache]*) – The wheel cache to use, defaults to None
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **check_supported_wheels** (*bool*) – Whether to check support of wheels before including them in resolution.

Returns A dictionary mapping requirements to corresponding `:class:`~pip._internal.req.req_install.InstallRequirement`'s`

Return type `InstallRequirement`

Example

```
>>> from pip_shims.shims import resolve, InstallRequirement
>>> ireq = InstallRequirement.from_line("requests>=2.20")
>>> results = resolve(ireq)
>>> for k, v in results.items():
...     print("{0}: {1!r}".format(k, v))
requests: <InstallRequirement object: requests>=2.20 from https://files.
↳pythonhosted.
org/packages/51/bd/23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/
↳reque
sts-2.22.0-py2.py3-none-any.whl
↳#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9a590
```

(continues on next page)

(continued from previous page)

```
f48c010551dc6c4b31 editable=False>
idna: <InstallRequirement object: idna<2.9,>=2.5 from https://files.pythonhosted.
↳org/
packages/14/2c/cd551d81dbe15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-
↳2.8-
py2.py3-none-any.whl
↳#sha256=ea8b7f6188e6fa117537c3df7da9fc686d485087abf6ac197f9c46432
f7e4a3c (from requests>=2.20) editable=False>
urllib3: <InstallRequirement object: urllib3!=1.25.0,!<1.25.1,<1.26,>=1.21.1 from_
↳htt
ps://files.pythonhosted.org/packages/b4/40/
↳a9837291310ee1ccc242ceb6ebfd9eb21539649f19
3a7c8c86ba15b98539/urllib3-1.25.7-py2.py3-none-any.whl
↳#sha256=a8a318824cc77d1fd4b2bec
2ded92646630d7fe8619497b142c84a9e6f5a7293 (from requests>=2.20) editable=False>
chardet: <InstallRequirement object: chardet<3.1.0,>=3.0.2 from https://files.
↳pythonh
sted.org/packages/bc/a9/
↳01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8
/chardet-3.0.4-py2.py3-none-any.whl
↳#sha256=fc323ffcaeaed0e0a02bf4d117757b98aed530d9ed
4531e3e15460124c106691 (from requests>=2.20) editable=False>
certifi: <InstallRequirement object: certifi>=2017.4.17 from https://files.
↳pythonhost
ed.org/packages/18/b0/
↳8146a4f8dd402f60744fa380bc73ca47303cccfc8b9190fd16a827281eac2/ce
rtifi-2019.9.11-py2.py3-none-any.whl
↳#sha256=fd7c7c74727ddcf00e9acd26bba8da604ffec95bf
1c2144e67aff7a8b50e6cef (from requests>=2.20) editable=False>
```

`pip_shims.shims.build_wheel` (*req=None, reqset=None, output_dir=None, preparer=None, wheel_cache=None, build_options=None, global_options=None, check_binary_allowed=None, no_clean=False, session=None, finder=None, install_command=None, req_tracker=None, build_dir=None, src_dir=None, download_dir=None, wheel_download_dir=None, cache_dir=None, use_user_site=False, use_pep517=None, *, format_control_provider=<pip_shims.models.ShimmedPathCollection object>, wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>, preparer_provider=<pip_shims.models.ShimmedPathCollection object>, wheel_builder_provider=<pip_shims.models.ShimmedPathCollection object>, build_one_provider=<pip_shims.models.ShimmedPathCollection object>, build_one_inside_env_provider=<pip_shims.models.ShimmedPathCollection object>, build_many_provider=<pip_shims.models.ShimmedPathCollection object>, install_command_provider=<pip_shims.models.ShimmedPathCollection object>, finder_provider=None, reqset_provider=<pip_shims.models.ShimmedPathCollection object>*)

Build a wheel or a set of wheels

Raises `TypeError` – Raised when no requirements are provided

Parameters

- **req** (*Optional* [`TInstallRequirement`]) – An `InstallRequirement` to build
- **reqset** (*Optional* [`TReqSet`]) – A `RequirementSet` instance (`pip<10`) or an iterable

of *InstallRequirement* instances (*pip*≥10) to build

- **output_dir** (*Optional[str]*) – Target output directory, only useful when building one wheel using *pip*≥20.0
- **preparer** (*Optional[TPreparer]*) – A preparer instance, defaults to *None*
- **wheel_cache** (*Optional[TWheelCache]*) – A wheel cache instance, defaults to *None*
- **build_options** (*Optional[List[str]]*) – A list of build options to pass in
- **global_options** (*Optional[List[str]]*) – A list of global options to pass in
- **bool] check_binary_allowed** (*Optional[Callable[TInstallRequirement,]]*) – A callable to check whether we are allowed to build and cache wheels for an *ireq*
- **no_clean** (*bool*) – Whether to avoid cleaning up wheels
- **session** (*Optional[TSession]*) – A *PipSession* instance to pass to create a *finder* if necessary
- **finder** (*Optional[TFinder]*) – A *PackageFinder* instance to use for generating a *WheelBuilder* instance on *pip*<20
- **install_command** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to *None*.
- **req_tracker** (*Optional[TReqTracker]*) – An optional requirement tracker instance, if one already exists
- **build_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **src_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **download_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **wheel_download_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **cache_dir** (*Optional[str]*) – Passthrough cache directory for wheel cache options
- **use_user_site** (*bool*) – Whether to use the user site directory when preparing install requirements on *pip*<20
- **use_pep517** (*Optional[bool]*) – When set to *True* or *False*, prefers building with or without pep517 as specified, otherwise uses requirement preference. Only works for single requirements.
- **format_control_provider** (*Optional[TShimmedFunc]*) – A provider for the *FormatControl* class
- **wheel_cache_provider** (*Optional[TShimmedFunc]*) – A provider for the *WheelCache* class
- **preparer_provider** (*Optional[TShimmedFunc]*) – A provider for the *RequirementPreparer* class
- **wheel_builder_provider** (*Optional[TShimmedFunc]*) – A provider for the *WheelBuilder* class, if it exists
- **build_one_provider** (*Optional[TShimmedFunc]*) – A provider for the *_build_one* function, if it exists
- **build_one_inside_env_provider** (*Optional[TShimmedFunc]*) – A provider for the *_build_one_inside_env* function, if it exists

- **build_many_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *build* function, if it exists
- **install_command_provider** (*Optional*[*TShimmedFunc*]) – A shim for providing new install command instances
- **finder_provider** (*TShimmedFunc*) – A provider to package finder instances
- **reqset_provider** (*TShimmedFunc*) – A provider for requirement set generation

Returns A tuple of successful and failed install requirements or else a path to a wheel

Return type `Optional[Union[str, Tuple[List[TInstallRequirement], List[TInstallRequirement]]]]`

2.1.5 pip_shims.environment

Module with functionality to learn about the environment.

`pip_shims.environment.get_base_import_path()`

`pip_shims.environment.get_pip_version(import_path='pip')`

`pip_shims.environment.is_type_checking()`

`pip_shims._strip_extras(path)`

class `pip_shims.SessionCommandMixin`

Bases: `pip._internal.cli.command_context.CommandContextMixin`

A class mixin for command classes needing `_build_session()`.

__build_session (*options*, *retries=None*, *timeout=None*)

classmethod **__get_index_urls** (*options*)

Return a list of index urls from user-provided options.

enter_context (*context_provider*)

get_default_session (*options*)

Get a default-managed session.

main_context ()

class `pip_shims.Command` (*name='Default pip command'*, *summary='PipCommand'*, *isolated='Default pip command'*)

Bases: `pip._internal.cli.base_command.Command`, `pip._internal.cli.req_command.SessionCommandMixin`

__build_session (*options*, *retries=None*, *timeout=None*)

classmethod **__get_index_urls** (*options*)

Return a list of index urls from user-provided options.

__main (*args*)

add_options ()

enter_context (*context_provider*)

get_default_session (*options*)

Get a default-managed session.

handle_pip_version_check (*options*)

This is a no-op so that commands by default do not do the pip version check.

ignore_require_venv = **False**

main (*args*)

main_context ()

parse_args (*args*)

run (*options, args*)

usage = None

class pip_shims.**ConfigOptionParser** (**args, **kwargs*)

Bases: pip._internal.cli.parser.CustomOptionParser

Custom option parser which updates its defaults by checking the configuration files and environmental variables

_add_help_option ()

_add_version_option ()

_check_conflict (*option*)

_create_option_list ()

_create_option_mappings ()

_get_all_options ()

_get_args (*args*)

_get_ordered_configuration_items ()

_init_parsing_state ()

_match_long_opt (*opt : string*) → string

Determine which long option string 'opt' matches, ie. which one it is an unambiguous abbreviation for. Raises BadOptionError if 'opt' doesn't unambiguously match any long option string.

_populate_option_list (*option_list, add_help=True*)

_process_args (*largs, rargs, values*)

_process_args(*largs* [[string],] *rargs* : [string], *values* : Values)

Process command-line arguments and populate 'values', consuming options and arguments from 'rargs'. If 'allow_interspersed_args' is false, stop at the first non-option argument. If true, accumulate any interspersed non-option arguments in 'largs'.

_process_long_opt (*rargs, values*)

_process_short_opts (*rargs, values*)

_share_option_mappings (*parser*)

_update_defaults (*defaults*)

Updates the given defaults with values from the config files and the environ. Does a little special handling for certain types of options (lists).

add_option (*Option*)

add_option(*opt_str, ..., kwarg=val, ...*)

add_option_group (**args, **kwargs*)

add_options (*option_list*)

check_default (*option, key, val*)

check_values (*values : Values, args : [string]*)

-> (*values : Values, args : [string]*)

Check that the supplied option values and leftover arguments are valid. Returns the option values and leftover arguments (possibly adjusted, possibly completely new – whatever you like). Default implementation just returns the passed-in values; subclasses may override as desired.

destroy ()

Declare that you are done with this OptionParser. This cleans up reference cycles so the OptionParser (and all objects referenced by it) can be garbage-collected promptly. After calling destroy(), the OptionParser is unusable.

disable_interspersed_args ()

Set parsing to stop on the first non-option. Use this if you have a command processor which runs another command that has options of its own and you want to make sure these options don't get confused.

enable_interspersed_args ()

Set parsing to not stop on the first non-option, allowing interspersing switches with command arguments. This is the default behavior. See also disable_interspersed_args() and the class documentation description of the attribute allow_interspersed_args.

error (*msg : string*)

Print a usage message incorporating 'msg' to stderr and exit. If you override this in a subclass, it should not return – it should either exit or raise an exception.

exit (*status=0, msg=None*)

expand_prog_name (*s*)

format_description (*formatter*)

format_epilog (*formatter*)

format_help (*formatter=None*)

format_option_help (*formatter=None*)

get_default_values ()

Overriding to make updating the defaults after instantiation of the option parser possible, _update_defaults() does the dirty work.

get_description ()

get_option (*opt_str*)

get_option_group (*opt_str*)

get_prog_name ()

get_usage ()

get_version ()

has_option (*opt_str*)

insert_option_group (*idx, *args, **kwargs*)

Insert an OptionGroup at a given position.

option_list_all

Get a list of all options, including those in option groups.

parse_args (*args=None, values=None*)

parse_args(*args* [[string] = sys.argv[1:],] *values : Values = None*)

-> (values : Values, args : [string])

Parse the command-line options found in 'args' (default: sys.argv[1:]). Any errors result in a call to 'error()', which by default prints the usage message to stderr and calls sys.exit() with an error message. On success returns a pair (values, args) where 'values' is a Values instance (with all your option values) and 'args' is the list of arguments left over after parsing options.

print_help (*file* : *file* = *stdout*)

Print an extended help message, listing all options and any help text provided with them, to 'file' (default stdout).

print_usage (*file* : *file* = *stdout*)

Print the usage message for the current program (self.usage) to 'file' (default stdout). Any occurrence of the string "%prog" in self.usage is replaced with the name of the current program (basename of sys.argv[0]). Does nothing if self.usage is empty or not defined.

print_version (*file* : *file* = *stdout*)

Print the version message for this program (self.version) to 'file' (default stdout). As with print_usage(), any occurrence of "%prog" in self.version is replaced by the current program's name. Does nothing if self.version is empty or undefined.

remove_option (*opt_str*)

set_conflict_handler (*handler*)

set_default (*dest*, *value*)

set_defaults (***kwargs*)

set_description (*description*)

set_process_default_values (*process*)

set_usage (*usage*)

standard_option_list = []

exception pip_shims.DistributionNotFound

Bases: pip._internal.exceptions.InstallationError

Raised when a distribution cannot be found to satisfy a requirement

args

with_traceback ()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class pip_shims.FormatControl (*no_binary=None*, *only_binary=None*)

Bases: object

Helper for managing formats from which a package can be installed.

disallow_binaries ()

get_allowed_formats (*canonical_name*)

static handle_mutual_excludes (*value*, *target*, *other*)

no_binary

only_binary

class pip_shims.FrozenRequirement (*name*, *req*, *editable*, *comments=()*)

Bases: object

classmethod **from_dist** (*dist*)

`pip_shims.get_installed_distributions` (*local_only=True*, *skip={'argparse', 'python', 'wsgiref'}*, *include_editables=True*, *editables_only=False*, *user_only=False*, *paths=None*)

Return a list of installed Distribution objects.

If `local_only` is `True` (default), only return installations local to the current virtualenv, if in a virtualenv.

`skip` argument is an iterable of lower-case project names to ignore; defaults to `stdlib_pkgs`

If `include_editables` is `False`, don't report editables.

If `editables_only` is `True`, only report editables.

If `user_only` is `True`, only report installations in the user site directory.

If `paths` is set, only report the distributions present at the specified list of locations.

exception `pip_shims.InstallationError`

Bases: `pip._internal.exceptions.PipError`

General exception during installation

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.UninstallationError`

Bases: `pip._internal.exceptions.PipError`

General exception during uninstallation

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.RequirementsFileParseError`

Bases: `pip._internal.exceptions.InstallationError`

Raised when a general error occurs parsing a requirements file line.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.BestVersionAlreadyInstalled`

Bases: `pip._internal.exceptions.PipError`

Raised when the most up-to-date version of a package is already installed.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.BadCommand`

Bases: `pip._internal.exceptions.PipError`

Raised when virtualenv or a command is not found

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.CommandError`

Bases: `pip._internal.exceptions.PipError`

Raised when there is an error in command-line arguments

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.PreviousBuildDirError`

Bases: `pip._internal.exceptions.PipError`

Raised when there's a previous conflicting build directory

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

`pip_shims.install_req_from_editable` (*editable_req*, *comes_from=None*, *use_pep517=None*, *isolated=False*, *options=None*, *constraint=False*, *user_supplied=False*)

`pip_shims.install_req_from_line` (*name*, *comes_from=None*, *use_pep517=None*, *isolated=False*, *options=None*, *constraint=False*, *line_source=None*, *user_supplied=False*)

Creates an InstallRequirement from a name, which might be a requirement, directory containing 'setup.py', filename, or URL.

Parameters `line_source` – An optional string describing where the line is from, for logging purposes in case of an error.

`pip_shims.install_req_from_req_string` (*req_string*, *comes_from=None*, *isolated=False*, *use_pep517=None*, *user_supplied=False*)

class `pip_shims.InstallRequirement` (*req*, *comes_from*, *editable=False*, *link=None*, *markers=None*, *use_pep517=None*, *isolated=False*, *install_options=None*, *global_options=None*, *hash_options=None*, *constraint=False*, *extras=()*, *user_supplied=False*)

Bases: `pip._internal.req.req_install.InstallRequirement`

_generate_metadata()

Invokes metadata generator functions, with the required arguments.

_get_archive_name (*path*, *parentdir*, *rootdir*)

_set_requirement()

Set requirement after generating metadata.

archive (*build_dir*)

Saves archive to provided build_dir.

Used for saving downloaded VCS requirements as part of *pip download*.

assert_source_matches_version()

build_location (*build_dir*, *autodelete*, *parallel_builds*)

check_if_exists (*use_user_site*)

Find an installed distribution that satisfies or conflicts with this requirement, and set self.satisfied_by or self.should_reinstall appropriately.

ensure_build_location (*build_dir*, *autodelete*, *parallel_builds*)

ensure_has_source_dir (*parent_dir, autodelete=False, parallel_builds=False*)

Ensure that a source_dir is set.

This will create a temporary build dir if the name of the requirement isn't known yet.

Parameters **parent_dir** – The ideal pip parent_dir for the source_dir. Generally src_dir for editables and build_dir for sdist.

Returns self.source_dir

format_debug ()

An un-tested helper for getting state, for debugging.

from_editable = `<pip_shims.utils.BaseMethod object>`

from_line = `<pip_shims.utils.BaseMethod object>`

from_path ()

Format a nice indicator to show where this “comes from”

get_dist ()

has_hash_options

Return whether any known-good hashes are specified as options.

These activate --require-hashes mode; hashes specified as part of a URL do not.

hashes (*trust_internet=True*)

Return a hash-comparer that considers my option- and URL-based hashes to be known-good.

Hashes in URLs—ones embedded in the requirements file, not ones downloaded from an index server—are almost peers with ones from flags. They satisfy --require-hashes (whether it was implicitly or explicitly activated) but do not activate it. md5 and sha224 are not allowed in flags, which should nudge people toward good algos. We always OR all hashes together, even ones from URLs.

Parameters **trust_internet** – Whether to trust URL-based (#md5=...) hashes downloaded from the internet, as by populate_link()

install (*install_options, global_options=None, root=None, home=None, prefix=None, warn_script_location=True, use_user_site=False, pycompile=True*)

installed_version

is_pinned

Return whether I am pinned to an exact version.

For example, some-package==1.2 is pinned; some-package>1.2 is not.

is_wheel

load_pyproject_toml ()

Load the pyproject.toml file.

After calling this routine, all of the attributes related to PEP 517 processing for this requirement have been set. In particular, the use_pep517 attribute can be used to determine whether we should follow the PEP 517 or legacy (setup.py) code path.

match_markers (*extras_requested=None*)

metadata

name

prepare_metadata ()

Ensure that project metadata is available.

Under PEP 517, call the backend hook to prepare the metadata. Under legacy processing, call `setup.py egg-info`.

pyproject_toml_path

setup_py_path

specifier

uninstall (*auto_confirm=False, verbose=False*)

Uninstall the distribution currently satisfying this requirement.

Prompts before removing or modifying files unless `auto_confirm` is `True`.

Refuses to delete or modify files outside of `sys.prefix` - thus uninstallation within a virtual environment can only modify that virtual environment, even if the virtualenv is linked to global site-packages.

unpacked_source_directory

update_editable (*obtain=True*)

warn_on_mismatching_name ()

`pip_shims.is_archive_file` (*name*)

Return `True` if *name* is considered as an archive file.

`pip_shims.is_file_url` (*link*)

class `pip_shims.Downloader` (*session, progress_bar*)

Bases: `object`

`pip_shims.unpack_url` (*link, location, downloader, download_dir=None, hashes=None*)

Unpack link into location, downloading if required.

Parameters *hashes* – A Hashes object, one of whose embedded hashes must match, or `HashMismatch` will be raised. If the Hashes is empty, no matches are required, and unhashable types of requirements (like VCS ones, which would ordinarily raise `HashUnsupported`) are allowed.

`pip_shims.is_installable_dir` (*path*)

Is path a directory containing `setup.py` or `pyproject.toml`?

class `pip_shims.Link` (*url, comes_from=None, requires_python=None, yanked_reason=None, cache_link_parsing=True*)

Bases: `pip._internal.models.link.Link`

Parameters

- **url** – url of the resource pointed to (href of the link)
- **comes_from** – instance of `HTMLPage` where the link was found, or string.
- **requires_python** – String containing the *Requires-Python* metadata field, specified in PEP 345. This may be specified by a `data-requires-python` attribute in the HTML link tag, as described in PEP 503.
- **yanked_reason** – the reason the file has been yanked, if the file has been yanked, or `None` if the file hasn't been yanked. This is the value of the “data-yanked” attribute, if present, in a simple repository HTML link. If the file has been yanked but no reason was provided, this should be the empty string. See PEP 592 for more information and the specification.
- **cache_link_parsing** – A flag that is used elsewhere to determine whether resources retrieved from this link should be cached. PyPI index urls should generally have this set to `False`, for example.

_compare (*other, method*)

```

_compare_key
_defining_class
_egg_fragment_re = re.compile('[#&]egg=(^[&]*)')
_hash_re = re.compile('(sha1|sha224|sha384|sha256|sha512|md5)=[a-f0-9]+')
_parsed_url
_subdirectory_fragment_re = re.compile('[#&]subdirectory=(^[&]*)')
_url
cache_link_parsing
comes_from
egg_fragment
ext
file_path
filename
has_hash
hash
hash_name
is_artifact
is_existing_dir()
is_file
is_hash_allowed(hashes)
    Return True if the link has a hash and it is allowed.
is_vcs
is_wheel
is_yanked
netloc
    This can contain auth information.
path
requires_python
scheme
show_url
splittext()
subdirectory_fragment
url
url_without_fragment
yanked_reason

```

```

pip_shims.make_abstract_dist(install_req)
    Returns a Distribution for the given InstallRequirement

```

`pip_shims.make_distribution_for_install_requirement` (*install_req*)

Returns a Distribution for the given InstallRequirement

`pip_shims.make_option_group` (*group, parser*)

Return an OptionGroup object group – assumed to be dict with ‘name’ and ‘options’ keys parser – an optparse Parser

class `pip_shims.PackageFinder` (*link_collector, target_python, allow_yanked, format_control=**None, candidate_prefs=**None, ignore_requires_python=**None*)

Bases: `object`

This finds packages.

This is meant to match easy_install’s technique for looking for packages, by reading pages and looking for appropriate links.

This constructor is primarily meant to be used by the create() class method and from tests.

Parameters

- **format_control** – A FormatControl object, used to control the selection of source packages / binary packages when consulting the index and links.
- **candidate_prefs** – Options to use when creating a CandidateEvaluator object.

`_log_skipped_link` (*link, reason*)

`_sort_links` (*links*)

Returns elements of links in order, non-egg links first, egg links second, while eliminating duplicates

allow_all_prereleases

classmethod `create` (*link_collector, selection_prefs, target_python=**None*)

Create a PackageFinder.

Parameters

- **selection_prefs** – The candidate selection preferences, as a SelectionPreferences object.
- **target_python** – The target Python interpreter to use when checking compatibility. If None (the default), a TargetPython object will be constructed from the running Python.

`evaluate_links` (*link_evaluator, links*)

Convert links that are candidates to InstallationCandidate objects.

`find_all_candidates` (*project_name*)

Find all available InstallationCandidate for project_name

This checks index_urls and find_links. All versions found are returned as an InstallationCandidate list.

See LinkEvaluator.evaluate_link() for details on which files are accepted.

`find_best_candidate` (*project_name, specifier=**None, hashes=**None*)

Find matches for the given project and specifier.

Parameters **specifier** – An optional object implementing *filter* (e.g. *packaging.specifiers.SpecifierSet*) to filter applicable versions.

Returns A *BestCandidateResult* instance.

`find_links`

find_requirement (*req, upgrade*)

Try to find a Link matching req

Expects req, an InstallRequirement and upgrade, a boolean Returns a InstallationCandidate if found, Raises DistributionNotFound or BestVersionAlreadyInstalled otherwise

get_install_candidate (*link_evaluator, link*)

If the link is a candidate for install, convert it to an InstallationCandidate and return it. Otherwise, return None.

index_urls

make_candidate_evaluator (*project_name, specifier=None, hashes=None*)

Create a CandidateEvaluator object to use.

make_link_evaluator (*project_name*)

prefer_binary

process_project_url (*project_url, link_evaluator*)

search_scope

set_allow_all_prereleases ()

set_prefer_binary ()

target_python

trusted_hosts

```
class pip_shims.CandidateEvaluator(project_name, supported_tags, specifier, prefer_binary=False, allow_all_prereleases=False, hashes=None)
```

Bases: `object`

Responsible for filtering and sorting candidates for installation based on what tags are valid.

Parameters supported_tags – The PEP 425 tags supported by the target Python in order of preference (most preferred first).

_sort_key (*candidate*)

Function to pass as the *key* argument to a call to `sorted()` to sort InstallationCandidates by preference.

Returns a tuple such that tuples sorting as greater using Python’s default comparison operator are more preferred.

The preference is as follows:

First and foremost, candidates with allowed (matching) hashes are always preferred over candidates without matching hashes. This is because e.g. if the only candidate with an allowed hash is yanked, we still want to use that candidate.

Second, excepting hash considerations, candidates that have been yanked (in the sense of PEP 592) are always less preferred than candidates that haven’t been yanked. Then:

If not finding wheels, they are sorted by version only. If finding wheels, then the sort order is by version, then:

1. existing installs
2. wheels ordered via `Wheel.support_index_min(self._supported_tags)`
3. source archives

If `prefer_binary` was set, then all wheels are sorted above sources.

Note: it was considered to embed this logic into the Link comparison operators, but then different sdist links with the same version, would have to be considered equal

compute_best_candidate (*candidates*)

Compute and return a *BestCandidateResult* instance.

classmethod create (*project_name*, *target_python=None*, *prefer_binary=False*, *allow_all_prereleases=False*, *specifier=None*, *hashes=None*)

Create a *CandidateEvaluator* object.

Parameters

- **target_python** – The target Python interpreter to use when checking compatibility. If None (the default), a *TargetPython* object will be constructed from the running Python.
- **specifier** – An optional object implementing *filter* (e.g. *packaging.specifiers.SpecifierSet*) to filter applicable versions.
- **hashes** – An optional collection of allowed hashes.

get_applicable_candidates (*candidates*)

Return the applicable candidates from a list of candidates.

sort_best_candidate (*candidates*)

Return the best candidate per the instance’s sort order, or None if no candidate is acceptable.

class `pip_shims.CandidatePreferences` (*prefer_binary=False*, *allow_all_prereleases=False*)

Bases: `object`

Encapsulates some of the preferences for filtering and sorting *InstallationCandidate* objects.

Parameters **allow_all_prereleases** – Whether to allow all pre-releases.

class `pip_shims.LinkCollector` (*session*, *search_scope*)

Bases: `object`

Responsible for collecting *Link* objects from all configured locations, making network requests as needed.

The class’s main method is its `collect_links()` method.

collect_links (*project_name*)

Find all available links for the given project name.

Returns All the *Link* objects (unfiltered), as a *CollectedLinks* object.

classmethod create (*session*, *options*, *suppress_no_index=False*)

Parameters

- **session** – The *Session* to use to make requests.
- **suppress_no_index** – Whether to ignore the `–no-index` option when constructing the *SearchScope* object.

fetch_page (*location*)

Fetch an HTML page containing package links.

find_links

class `pip_shims.LinkEvaluator` (*project_name*, *canonical_name*, *formats*, *target_python*, *allow_yanked*, *ignore_requires_python=None*)

Bases: `object`

Responsible for evaluating links for a particular project.

Parameters

- **project_name** – The user supplied package name.
- **canonical_name** – The canonical package name.
- **formats** – The formats allowed for this package. Should be a set with ‘binary’ or ‘source’ or both in it.
- **target_python** – The target Python interpreter to use when evaluating link compatibility. This is used, for example, to check wheel compatibility, as well as when checking the Python version, e.g. the Python version embedded in a link filename (or egg fragment) and against an HTML link’s optional PEP 503 “data-requires-python” attribute.
- **allow_yanked** – Whether files marked as yanked (in the sense of PEP 592) are permitted to be candidates for install.
- **ignore_requires_python** – Whether to ignore incompatible PEP 503 “data-requires-python” values in HTML links. Defaults to False.

```
_py_version_re = re.compile('-py([123]\\.?[0-9]?)$')
```

evaluate_link (*link*)

Determine whether a link is a candidate for installation.

Returns A tuple (is_candidate, result), where *result* is (1) a version string if *is_candidate* is True, and (2) if *is_candidate* is False, an optional string to log the reason the link fails to qualify.

```
class pip_shims.TargetPython(platform=None, py_version_info=None, abi=None, implementation=None)
```

Bases: `object`

Encapsulates the properties of a Python interpreter one is targeting for a package install, download, etc.

Parameters

- **platform** – A string or None. If None, searches for packages that are supported by the current system. Otherwise, will find packages that can be built on the platform passed in. These packages will only be downloaded for distribution: they will not be built locally.
- **py_version_info** – An optional tuple of ints representing the Python version information to use (e.g. `sys.version_info[:3]`). This can have length 1, 2, or 3 when provided.
- **abi** – A string or None. This is passed to `compatibility_tags.py`’s `get_supported()` function as is.
- **implementation** – A string or None. This is passed to `compatibility_tags.py`’s `get_supported()` function as is.

_given_py_version_info

_valid_tags

abi

format_given ()

Format the given, non-None attributes for display.

get_tags ()

Return the supported PEP 425 tags to check wheel candidates against.

The tags are returned in order of preference (most preferred first).

implementation

platform

py_version

py_version_info

class pip_shims.**SearchScope** (*find_links, index_urls*)

Bases: `object`

Encapsulates the locations that pip is configured to search.

classmethod **create** (*find_links, index_urls*)

Create a SearchScope object after normalizing the *find_links*.

find_links

get_formatted_locations ()

get_index_urls_locations (*project_name*)

Returns the locations found via self.index_urls

Checks the url_name on the main (first in the list) index and use this url_name to produce all locations

index_urls

class pip_shims.**SelectionPreferences** (*allow_yanked, allow_all_prereleases=False, format_control=None, prefer_binary=False, ignore_requires_python=None*)

Bases: `object`

Encapsulates the candidate selection preferences for downloading and installing files.

Create a SelectionPreferences object.

Parameters

- **allow_yanked** – Whether files marked as yanked (in the sense of PEP 592) are permitted to be candidates for install.
- **format_control** – A FormatControl object or None. Used to control the selection of source packages / binary packages when consulting the index and links.
- **prefer_binary** – Whether to prefer an old, but valid, binary dist over a new source dist.
- **ignore_requires_python** – Whether to ignore incompatible “Requires-Python” values in links. Defaults to False.

allow_all_prereleases

allow_yanked

format_control

ignore_requires_python

prefer_binary

pip_shims.parse_requirements (*filename, session, finder=None, comes_from=None, options=None, constraint=False*)

Parse a requirements file and yield ParsedRequirement instances.

Parameters

- **filename** – Path or url of requirements file.
- **session** – PipSession instance.
- **finder** – Instance of pip.index.PackageFinder.
- **comes_from** – Origin description of requirements.
- **options** – cli options.

- **constraint** – If true, parsing a constraint file rather than requirements file.

`pip_shims.path_to_url` (*path*)

Convert a path to a file: URL. The path will be made absolute and have quoted path parts.

exception `pip_shims.PipError`

Bases: `Exception`

Base pip exception

args

with_traceback ()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class `pip_shims.RequirementPreparer` (*build_dir, download_dir, src_dir, wheel_download_dir, build_isolation, req_tracker, downloader, finder, require_hashes, use_user_site*)

Bases: `object`

Prepares a Requirement

_download_should_save

_ensure_link_req_src_dir (*req, download_dir, parallel_builds*)

Ensure source_dir of a linked InstallRequirement.

_get_linked_req_hashes (*req*)

_log_preparing_link (*req*)

Log the way the link prepared.

prepare_editable_requirement (*req*)

Prepare an editable requirement

prepare_installed_requirement (*req, skip_reason*)

Prepare an already-installed requirement

prepare_linked_requirement (*req, parallel_builds=False*)

Prepare a requirement to be obtained from req.link.

class `pip_shims.RequirementSet` (*check_supported_wheels=True*)

Bases: `object`

Create a RequirementSet.

add_named_requirement (*install_req*)

add_requirement (*install_req, parent_req_name=None, extras_requested=None*)

Add install_req as a requirement to install.

Parameters

- **parent_req_name** – The name of the requirement that needed this added. The name is used because when multiple unnamed requirements resolve to the same name, we could otherwise end up with dependency links that point outside the Requirements set. parent_req must already be added. Note that None implies that this is a user supplied requirement, vs an inferred one.
- **extras_requested** – an iterable of extras used to evaluate the environment markers.

Returns Additional requirements to scan. That is either [] if the requirement is not applicable, or [install_req] if the requirement is applicable and has just been added.

add_unnamed_requirement (*install_req*)

all_requirements

get_requirement (*name*)

has_requirement (*name*)

class pip_shims.**RequirementTracker** (*root*)

Bases: `object`

_entry_path (*link*)

add (*req*)

Add an InstallRequirement to build tracking.

cleanup ()

remove (*req*)

Remove an InstallRequirement from build tracking.

track (*req*)

class pip_shims.**TempDirectory** (*path=None, delete=<pip._internal.utils.temp_dir.Default object>, kind='temp', globally_managed=False*)

Bases: `object`

Helper class that owns and cleans up a temporary directory.

This class can be used as a context manager or as an OO representation of a temporary directory.

Attributes:

path Location to the created temporary directory

delete Whether the directory should be deleted when exiting (when used as a contextmanager)

Methods:

cleanup() Deletes the temporary directory

When used as a context manager, if the delete attribute is True, on exiting the context the temporary directory is deleted.

_create (*kind*)

Create a temporary directory and store its path in self.path

cleanup ()

Remove the temporary directory created and reset state

path

pip_shims.**global_tempdir_manager** ()

pip_shims.**shim_unpack** (*, *unpack_fn=<pip_shims.models.ShimmedPathCollection object>, download_dir, tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection object>, ireq=None, link=None, location=None, hashes=None, progress_bar='off', only_download=None, downloader_provider=<pip_shims.models.ShimmedPathCollection object>, session=None*)

Accepts all parameters that have been valid to pass to `pip._internal.download.unpack_url()` and selects or drops parameters as needed before invoking the provided callable.

Parameters

- **unpack_fn** (*Callable*) – A callable or shim referring to the pip implementation
- **download_dir** (*str*) – The directory to download the file to

- **tempdir_manager_provider** (*TShimmedFunc*) – A callable or shim referring to *global_tempdir_manager* function from pip or a shimmed no-op context manager

:param Optional[InstallRequirement] ireq: an Install Requirement instance, defaults to None

:param Optional[Link] link: A Link instance, defaults to None.

Parameters

- **location** (*Optional[str]*) – A location or source directory if the target is a VCS url, defaults to None.
- **hashes** (*Optional[Any]*) – A Hashes instance, defaults to None
- **progress_bar** (*str*) – Indicates progress par usage during download, defatuls to off.
- **only_download** (*Optional[bool]*) – Whether to skip install, defaults to None.
- **downloader_provider** (*Optional[ShimmedPathCollection]*) – A downloader class to instantiate, if applicable.
- **session** (*Optional[Session]*) – A PipSession instance, defaults to None.

Returns The result of unpacking the url.

Return type `None`

```
pip_shims.get_requirement_tracker()
```

```
class pip_shims.Resolver(preparer, finder, wheel_cache, make_install_req, use_user_site,
                        ignore_dependencies, ignore_installed, ignore_requires_python,
                        force_reinstall, upgrade_strategy, py_version_info=None)
```

Bases: `pip._internal.resolution.base.BaseResolver`

Resolves which packages need to be installed/uninstalled to perform the requested operation without breaking the requirements of any package.

```
_allowed_strategies = {'eager', 'only-if-needed', 'to-satisfy-only'}
```

```
_check_skip_installed(req_to_install)
```

Check if req_to_install should be skipped.

This will check if the req is installed, and whether we should upgrade or reinstall it, taking into account all the relevant user options.

After calling this req_to_install will only have satisfied_by set to None if the req_to_install is to be upgraded/reinstalled etc. Any other value will be a dist recording the current thing installed that satisfies the requirement.

Note that for vcs urls and the like we can't assess skipping in this routine - we simply identify that we need to pull the thing down, then later on it is pulled down and introspected to assess upgrade/ reinstalls etc.

Returns A text reason for why it was skipped, or None.

```
_find_requirement_link(req)
```

```
_get_abstract_dist_for(req)
```

Takes a InstallRequirement and returns a single AbstractDist representing a prepared variant of the same.

```
_is_upgrade_allowed(req)
```

```
_populate_link(req)
```

Ensure that if a link can be found for this, that it is found.

Note that `req.link` may still be `None` - if the requirement is already installed and not needed to be upgraded based on the return value of `_is_upgrade_allowed()`.

If `preparer.require_hashes` is `True`, don't use the wheel cache, because cached wheels, always built locally, have different hashes than the files downloaded from the index server and thus throw false hash mismatches. Furthermore, cached wheels at present have undeterministic contents due to file modification times.

`_resolve_one` (*requirement_set, req_to_install*)

Prepare a single requirements file.

Returns A list of additional `InstallRequirements` to also install.

`_set_req_to_reinstall` (*req*)

Set a requirement to be installed.

`get_installation_order` (*req_set*)

Create the installation order.

The installation order is topological - requirements are installed before the requiring thing. We break cycles at an arbitrary point, and make no other guarantees.

`resolve` (*root_reqs, check_supported_wheels*)

Resolve what operations need to be done

As a side-effect of this method, the packages (and their dependencies) are downloaded, unpacked and prepared for installation. This preparation is done by `pip.operations.prepare`.

Once PyPI has static dependency metadata available, it would be possible to move the preparation to become a step separated from dependency resolution.

class `pip_shims.SafeFileCache` (*directory*)

Bases: `pip._vendor.cachecontrol.cache.BaseCache`

A file based cache which is safe to use even when the target directory may not be accessible or writable.

`_get_cache_path` (*name*)

`close` ()

`delete` (*key*)

`get` (*key*)

`set` (*key, value*)

class `pip_shims.UninstallPathSet` (*dist*)

Bases: `object`

A set of file paths to be removed in the uninstallation of a requirement.

`_allowed_to_proceed` (*verbose*)

Display which files would be deleted and prompt for confirmation

`_permitted` (*path*)

Return `True` if the given path is one we are permitted to remove/modify, `False` otherwise.

`add` (*path*)

`add_pth` (*pth_file, entry*)

`commit` ()

Remove temporary save dir: rollback will no longer be possible.

classmethod `from_dist` (*dist*)

remove (*auto_confirm=False, verbose=False*)
Remove paths in `self.paths` with confirmation (unless `auto_confirm` is True).

rollback ()
Rollback the changes previously made by `remove()`.

`pip_shims.url_to_path(url)`
Convert a file: URL to a path.

class `pip_shims.VcsSupport`

Bases: `object`

`_registry = {'bzzr': <pip._internal.vcs.bazaar.Bazaar object>, 'git': <pip._internal.vcs.git.Git object>, 'hg': <pip._internal.vcs.hg.Hg object>, 'ssh': <pip._internal.vcs.ssh.Ssh object>, 'svn': <pip._internal.vcs.svn.Svn object>, 'vcs': <pip._internal.vcs.vcs.Vcs object>}`

`all_schemes`

`backends`

`dirnames`

`get_backend(name)`
Return a VersionControl object or None.

`get_backend_for_dir(location)`
Return a VersionControl object if a repository of that type is found at the given directory.

`get_backend_for_scheme(scheme)`
Return a VersionControl object or None.

`register(cls)`

`schemes = ['ssh', 'git', 'hg', 'bzzr', 'sftp', 'svn']`

`unregister(name)`

class `pip_shims.Wheel(filename)`

Bases: `object`

`get_formatted_file_tags()`
Return the wheel's tags as a sorted list of strings.

`support_index_min(tags)`
Return the lowest index that one of the wheel's file_tag combinations achieves in the given list of supported tags.

For example, if there are 8 supported tags and one of the file tags is first in the list, then return 0.

Parameters `tags` – the PEP 425 tags to check the wheel against, in order with most preferred first.

Raises `ValueError` – If none of the wheel's file tags match one of the supported tags.

`supported(tags)`
Return whether the wheel is compatible with one of the given tags.

Parameters `tags` – the PEP 425 tags to check the wheel against.

`wheel_file_re = re.compile('^(?P<namever>(?P<name>.+?)-(?P<ver>.*?))\n ((- ?P<build>\d+)?)`

class `pip_shims.WheelCache(cache_dir, format_control)`

Bases: `pip._internal.cache.Cache`

Wraps `EphemWheelCache` and `SimpleWheelCache` into a single `Cache`

This `Cache` allows for gracefully degradation, using the ephem wheel cache when a certain link is not found in the simple wheel cache first.

`_get_cache_path_parts` (*link*)

Get parts of part that must be `os.path.join`d with `cache_dir`

`_get_cache_path_parts_legacy` (*link*)

Get parts of part that must be `os.path.join`d with `cache_dir`

Legacy cache key (pip < 20) for compatibility with older caches.

`_get_candidates` (*link, canonical_package_name*)

`get` (*link, package_name, supported_tags*)

Returns a link to a cached item if it exists, otherwise returns the passed link.

`get_cache_entry` (*link, package_name, supported_tags*)

Returns a `CacheEntry` with a link to a cached item if it exists or `None`. The cache entry indicates if the item was found in the persistent or ephemeral cache.

`get_ephem_path_for_link` (*link*)

`get_path_for_link` (*link*)

Return a directory to store cached items in for link.

`get_path_for_link_legacy` (*link*)

`pip_shims.build` (*requirements, wheel_cache, build_options, global_options*)

Build wheels.

Returns The list of `InstallRequirement` that succeeded to build and the list of `InstallRequirement` that failed to build.

`pip_shims.build_one` (*req, output_dir, build_options, global_options*)

Build one wheel.

Returns The filename of the built wheel, or `None` if the build failed.

`pip_shims.build_one_inside_env` (*req, output_dir, build_options, global_options*)

class `pip_shims.AbstractDistribution` (*req*)

Bases: `object`

A base class for handling installable artifacts.

The requirements for anything installable are as follows:

- we must be able to determine the requirement name (or we can't correctly handle the non-upgrade case).
- for packages with setup requirements, we must also be able to determine their requirements without installing additional packages (for the same reason as run-time dependencies)
- we must be able to create a `Distribution` object exposing the above metadata.

`_abc_impl = <_abc_data object>`

`get_pkg_resources_distribution` ()

`prepare_distribution_metadata` (*finder, build_isolation*)

class `pip_shims.InstalledDistribution` (*req*)

Bases: `pip._internal.distributions.base.AbstractDistribution`

Represents an installed package.

This does not need any preparation as the required information has already been computed.

`_abc_impl = <_abc_data object>`

`get_pkg_resources_distribution` ()

`prepare_distribution_metadata` (*finder*, *build_isolation*)

class `pip_shims.SourceDistribution` (*req*)

Bases: `pip._internal.distributions.base.AbstractDistribution`

Represents a source distribution.

The preparation step for these needs metadata for the packages to be generated, either using PEP 517 or using the legacy `setup.py egg_info`.

`_abc_impl = <_abc_data object>`

`_setup_isolation` (*finder*)

`get_pkg_resources_distribution` ()

`prepare_distribution_metadata` (*finder*, *build_isolation*)

class `pip_shims.WheelDistribution` (*req*)

Bases: `pip._internal.distributions.base.AbstractDistribution`

Represents a wheel distribution.

This does not need any preparation as wheels can be directly unpacked.

`_abc_impl = <_abc_data object>`

`get_pkg_resources_distribution` ()

Loads the metadata from the wheel file into memory and returns a `Distribution` that uses it, not relying on the wheel file or requirement.

`prepare_distribution_metadata` (*finder*, *build_isolation*)

`pip_shims.wheel_cache` (*cache_dir=None*, *format_control=None*, ***,
wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>,
format_control_provider=<pip_shims.models.ShimmedPathCollection ob-
ject>, *tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection*
object>)

`pip_shims.get_package_finder` (*install_cmd=None*, *options=None*, *session=None*,
platform=None, *python_versions=None*, *abi=None*,
implementation=None, *target_python=None*, *ig-*
nore_requires_python=None, ***, *target_python_builder=<class*
'pip._internal.models.target_python.TargetPython'>, *in-*
stall_cmd_provider=<pip_shims.models.ShimmedPathCollection
object>)

Shim for compatibility to generate package finders.

Build and return a `PackageFinder` instance using the `InstallCommand` helper method to construct the finder, shimmed with backports as needed for compatibility.

Parameters

- **`install_cmd_provider`** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **`install_cmd`** – A `InstallCommand` instance which is used to generate the finder.
- **`options`** (*optparse.Values*) – An optional `optparse.Values` instance generated by calling `install_cmd.parser.parse_args()` typically.
- **`session`** – An optional session instance, can be created by the `install_cmd`.
- **`platform`** (*Optional[str]*) – An optional platform string, e.g. `linux_x86_64`

- `..]] python_versions` (*Optional[Tuple[str, ...]*) – A tuple of 2-digit strings representing python versions, e.g. (“27”, “35”, “36”, “37”...)
- `abi` (*Optional[str]*) – The target abi to support, e.g. “cp38”
- `implementation` (*Optional[str]*) – An optional implementation string for limiting searches to a specific implementation, e.g. “cp” or “py”
- `target_python` – A `TargetPython` instance (will be translated to alternate arguments if necessary on incompatible pip versions).
- `ignore_requires_python` (*Optional[bool]*) – Whether to ignore `requires_python` on resulting candidates, only valid after pip version 19.3.1
- `target_python_builder` – A ‘`TargetPython`’ builder (e.g. the class itself, uninstantiated)

Returns A `pip._internal.index.package_finder.PackageFinder` instance

Return type `pip._internal.index.package_finder.PackageFinder`

Example

```
>>> from pip_shims.shims import InstallCommand, get_package_finder
>>> install_cmd = InstallCommand()
>>> finder = get_package_finder(
...     install_cmd, python_versions=("27", "35", "36", "37", "38"),
...     implementation="
cp"
... )
>>> candidates = finder.find_all_candidates("requests")
>>> requests_222 = next(iter(c for c in candidates if c.version.public == "2.22.0
... ))
>>> requests_222
<InstallationCandidate('requests', <Version('2.22.0')>, <Link https://files.
pythonhos
ted.org/packages/51/bd/
23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/r
equests-2.22.0-py2.py3-none-any.whl
#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9
a590f48c010551dc6c4b31 (from https://pypi.org/simple/requests/) (requires-python:>
=2.
7, !=3.0.*, !=3.1.*, !=3.2.*, !=3.3.*, !=3.4.*)>>>
```

```
pip_shims.make_preparer(*, preparer_fn=<pip_shims.models.ShimmedPathCollection object>,
req_tracker_fn=<pip_shims.models.ShimmedPathCollection object>,
build_dir=None, src_dir=None, download_dir=None,
wheel_download_dir=None, progress_bar='off',
build_isolation=False, session=None, finder=None, options=None,
require_hashes=None, use_user_site=None, req_tracker=None,
install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>,
downloader_provider=<pip_shims.models.ShimmedPathCollection object>,
install_cmd=None, finder_provider=<pip_shims.models.ShimmedPathCollection object>)
```

Creates a requirement preparer for preparing pip requirements.

Provides a compatibility shim that accepts all previously valid arguments and discards any that are no longer used.

Raises

- **TypeError** – No requirement tracker provided and one cannot be generated
- **TypeError** – No valid sessions provided and one cannot be generated
- **TypeError** – No valid finders provided and one cannot be generated

Parameters

- **preparer_fn** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **req_tracker_fn** (*Optional[TShimmedFunc]*) – Callable or shim for generating requirement trackers, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **src_dir** (*Optional[str]*) – Directory to find or extract source files, defaults to None
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **progress_bar** (*str*) – Whether to display a progress bar, defaults to off
- **build_isolation** (*bool*) – Whether to build requirements in isolation, defaults to False
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **require_hashes** (*Optional[bool]*) – Whether to require hashes for preparation
- **use_user_site** (*Optional[bool]*) – Whether to use the user site directory for preparing requirements
- **TShimmedFunc**] **req_tracker** (*Optional[Union[TReqTracker,])*) – The requirement tracker to use for building packages, defaults to None
- **downloader_provider** (*Optional[TShimmedFunc]*) – A downloader provider
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None
- **finder_provider** (*Optional[TShimmedFunc]*) – A package finder provider

Yield A new requirement preparer instance

Return type ContextManager[RequirementPreparer]

Example

```
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_tracker
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
```

(continues on next page)

(continued from previous page)

```

...     install_cmd, session=session, options=pip_options
... )
>>> with make_preparer(
...     options=pip_options, finder=finder, session=session, install_cmd=ic
... ) as preparer:
...     print(preparer)
<pip._internal.operations.prepare.RequirementPreparer object at 0x7f8a2734be80>

```

```

pip_shims.get_resolver(*, resolver_fn=<pip_shims.models.ShimmedPathCollection object>,
install_req_provider=<pip_shims.models.ShimmedPathCollection object>,
format_control_provider=<pip_shims.models.ShimmedPathCollection object>,
wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>,
finder=None, upgrade_strategy='to-satisfy-only',
force_reinstall=None, ignore_dependencies=None, ignore_requires_python=None,
ignore_installed=True, use_user_site=False, isolated=None, wheel_cache=None,
preparer=None, session=None, options=None, make_install_req=None,
install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>,
install_cmd=None, use_pep517=True)

```

A resolver creation compatibility shim for generating a resolver.

Consumes any argument that was previously used to instantiate a resolver, discards anything that is no longer valid.

Note: This is only valid for **pip >= 10.0.0**

Raises

- **ValueError** – A session is required but not provided and one cannot be created
- **ValueError** – A finder is required but not provided and one cannot be created
- **ValueError** – An install requirement provider is required and has not been provided

Parameters

- **resolver_fn** (*TShimmedFunc*) – The resolver function used to create new resolver instances.
- **install_req_provider** (*TShimmedFunc*) – The provider function to use to generate install requirements if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a format_control instance if needed.
- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to only-if-needed.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to None
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to None

- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to None
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to True
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to False
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to None
- **wheel_cache** (*Optional[WheelCache]*) – The wheel cache to use, defaults to None
- **preparer** (*Optional[TPreparer]*) – The requirement preparer to use, defaults to None
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **make_install_req** (*Optional[functools.partial]*) – The partial function to pass in to the resolver for actually generating install requirements, if necessary
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None.
- **use_pep517** (*bool*) – Whether to use the pep517 build process.

Returns A new resolver instance.

Return type Resolver

Example

```
>>> import os
>>> from tempdir import TemporaryDirectory
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_
↳ tracker,
...     get_resolver, InstallRequirement, RequirementSet
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> wheel_cache = WheelCache(USER_CACHE_DIR, FormatControl(None, None))
>>> with TemporaryDirectory() as temp_base:
...     reqset = RequirementSet()
...     ireq = InstallRequirement.from_line("requests")
...     ireq.is_direct = True
...     build_dir = os.path.join(temp_base, "build")
...     src_dir = os.path.join(temp_base, "src")
...     ireq.build_location(build_dir)
...     with make_preparer(
...         options=pip_options, finder=finder, session=session,
...         build_dir=build_dir, install_cmd=install_cmd,
```

(continues on next page)

(continued from previous page)

```

...     ) as preparer:
...         resolver = get_resolver(
...             finder=finder, ignore_dependencies=False, ignore_requires_
↪python=True,
...             preparer=preparer, session=session, options=pip_options,
...             install_cmd=install_cmd, wheel_cache=wheel_cache,
...         )
...         resolver.require_hashes = False
...         reqset.add_requirement(ireq)
...         results = resolver.resolve(reqset)
...         #reqset.cleanup_files()
...         for result_req in reqset.requirements:
...             print(result_req)
requests
chardet
certifi
urllib3
idna

```

`pip_shims.get_requirement_set` (*install_command=None*, *, *req_set_provider=<pip_shims.models.ShimmedPathCollection object>*, *build_dir=None*, *src_dir=None*, *download_dir=None*, *wheel_download_dir=None*, *session=None*, *wheel_cache=None*, *upgrade=False*, *upgrade_strategy=None*, *ignore_installed=False*, *ignore_dependencies=False*, *force_reinstall=False*, *use_user_site=False*, *isolated=False*, *ignore_requires_python=False*, *require_hashes=None*, *cache_dir=None*, *options=None*, *install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>*, *wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>*)

Creates a requirement set from the supplied parameters.

Not all parameters are passed through for all pip versions, but any invalid parameters will be ignored if they are not needed to generate a requirement set on the current pip version.

:param *ShimmedPathCollection* wheel_cache_provider: A context manager provider which resolves to a *WheelCache* instance

Parameters *install_command* – A *InstallCommand* instance which is used to generate the finder.

:param *ShimmedPathCollection* req_set_provider: A provider to build requirement set instances.

Parameters

- **build_dir** (*str*) – The directory to build requirements in. Removed in pip 10, defaults to None
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*str*) – The directory to download requirement artifacts to. Removed in pip 10, defaults to None
- **wheel_download_dir** (*str*) – The directory to download wheels to. Removed in pip 10, defaults to None

:param Session session: The pip session to use. Removed in pip 10, defaults to None

Parameters

- **wheel_cache** (*WheelCache*) – The pip WheelCache instance to use for caching wheels. Removed in pip 10, defaults to None
- **upgrade** (*bool*) – Whether to try to upgrade existing requirements. Removed in pip 10, defaults to False.
- **upgrade_strategy** (*str*) – The upgrade strategy to use, e.g. “only-if-needed”. Removed in pip 10, defaults to None.
- **ignore_installed** (*bool*) – Whether to ignore installed packages when resolving. Removed in pip 10, defaults to False.
- **ignore_dependencies** (*bool*) – Whether to ignore dependencies of requirements when resolving. Removed in pip 10, defaults to False.
- **force_reinstall** (*bool*) – Whether to force reinstall of packages when resolving. Removed in pip 10, defaults to False.
- **use_user_site** (*bool*) – Whether to use user site packages when resolving. Removed in pip 10, defaults to False.
- **isolated** (*bool*) – Whether to resolve in isolation. Removed in pip 10, defaults to False.
- **ignore_requires_python** (*bool*) – Removed in pip 10, defaults to False.
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **options** (*Values*) – An *Values* instance from an install cmd
- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.

Returns A new requirement set instance

Return type RequirementSet

```

pip_shims.resolve(ireq, *, reqset_provider=<pip_shims.models.ShimmedPathCollection object>,
                  req_tracker_provider=<pip_shims.models.ShimmedPathCollection object>,
                  install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>,
                  install_command=None, finder_provider=<pip_shims.models.ShimmedPathCollection
                  object>, resolver_provider=<pip_shims.models.ShimmedPathCollection object>,
                  wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>,
                  format_control_provider=<pip_shims.models.ShimmedPathCollection object>,
                  make_preparer_provider=<pip_shims.models.ShimmedPathCollection object>,
                  tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection object>,
                  options=None, session=None, resolver=None, finder=None, upgrade_strategy='to-
                  satisfy-only', force_reinstall=None, ignore_dependencies=None, ig-
                  nore_requires_python=None, ignore_installed=True, use_user_site=False,
                  isolated=None, build_dir=None, source_dir=None, download_dir=None,
                  cache_dir=None, wheel_download_dir=None, wheel_cache=None, re-
                  quire_hashes=None, check_supported_wheels=True)

```

Resolves the provided **InstallRequirement**, returning a dictionary.

Maps a dictionary of names to corresponding *InstallRequirement* values.

:param InstallRequirement ireq: An *InstallRequirement* to initiate the resolution process

:param ShimmedPathCollection reqset_provider: A **provider** to build requirement set instances.

:param *ShimmedPathCollection* req_tracker_provider: A provider to build requirement tracker instances

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_command** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None.

:param *ShimmedPathCollection* finder_provider: A provider to package finder instances.

:param *ShimmedPathCollection* resolver_provider: A provider to build resolver instances

Parameters

- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a format_control instance if needed.
- **make_preparer_provider** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **tempdir_manager_provider** (*Optional[TShimmedFunc]*) – Shim for generating tempdir manager for pip temporary directories
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None

:param Resolver resolver: A pre-existing resolver instance to use for resolution

Parameters

- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to None
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to None
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to None
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to True
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to False
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None

- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **cache_dir** (*str*) – The cache directory to use for caching artifacts during resolution
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **wheel_cache** (*Optional[WheelCache]*) – The wheel cache to use, defaults to None
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **check_supported_wheels** (*bool*) – Whether to check support of wheels before including them in resolution.

Returns A dictionary mapping requirements to corresponding `class:~pip._internal.req.req_install.InstallRequirement`'s

Return type `InstallRequirement`

Example

```
>>> from pip_shims.shims import resolve, InstallRequirement
>>> ireq = InstallRequirement.from_line("requests>=2.20")
>>> results = resolve(ireq)
>>> for k, v in results.items():
...     print("{0}: {1!r}".format(k, v))
requests: <InstallRequirement object: requests>=2.20 from https://files.
↳pythonhosted.
org/packages/51/bd/23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/
↳reque
sts-2.22.0-py2.py3-none-any.whl
↳#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9a590
f48c010551dc6c4b31 editable=False>
idna: <InstallRequirement object: idna<2.9,>=2.5 from https://files.pythonhosted.
↳org/
packages/14/2c/cd551d81dbe15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-
↳2.8-
py2.py3-none-any.whl
↳#sha256=ea8b7f6188e6fa117537c3df7da9fc686d485087abf6ac197f9c46432
f7e4a3c (from requests>=2.20) editable=False>
urllib3: <InstallRequirement object: urllib3!=1.25.0,!<1.25.1,<1.26,>=1.21.1 from
↳htt
↳ps://files.pythonhosted.org/packages/b4/40/
↳a9837291310ee1ccc242ceb6ebfd9eb21539649f19
3a7c8c86ba15b98539/urllib3-1.25.7-py2.py3-none-any.whl
↳#sha256=a8a318824cc77d1fd4b2bec
2ded92646630d7fe8619497b142c84a9e6f5a7293 (from requests>=2.20) editable=False>
chardet: <InstallRequirement object: chardet<3.1.0,>=3.0.2 from https://files.
↳pythonh
osted.org/packages/bc/a9/
↳01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8
/chardet-3.0.4-py2.py3-none-any.whl
↳#sha256=fc323ffcaeaed0e0a02bf4d117757b98aed530d9ed
4531e3e15460124c106691 (from requests>=2.20) editable=False>
```

(continues on next page)

(continued from previous page)

```
certifi: <InstallRequirement object: certifi>=2017.4.17 from https://files.
↳pythonhost
ed.org/packages/18/b0/
↳8146a4f8dd402f60744fa380bc73ca47303ccccf8b9190fd16a827281eac2/ce
rtifi-2019.9.11-py2.py3-none-any.whl
↳#sha256=fd7c7c74727ddcf00e9acd26bba8da604ffec95bf
1c2144e67aff7a8b50e6cef (from requests>=2.20) editable=False>
```

```
pip_shims.build_wheel(req=None, reqset=None, output_dir=None, preparer=None,
wheel_cache=None, build_options=None, global_options=None,
check_binary_allowed=None, no_clean=False, session=None,
finder=None, install_command=None, req_tracker=None, build_dir=None,
src_dir=None, download_dir=None, wheel_download_dir=None,
cache_dir=None, use_user_site=False, use_pep517=None, *, format_control_provider=<pip_shims.models.ShimmedPathCollection object>,
wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>,
preparer_provider=<pip_shims.models.ShimmedPathCollection object>,
wheel_builder_provider=<pip_shims.models.ShimmedPathCollection object>,
build_one_provider=<pip_shims.models.ShimmedPathCollection object>,
build_one_inside_env_provider=<pip_shims.models.ShimmedPathCollection object>,
build_many_provider=<pip_shims.models.ShimmedPathCollection object>,
install_command_provider=<pip_shims.models.ShimmedPathCollection object>,
finder_provider=None, reqset_provider=<pip_shims.models.ShimmedPathCollection object>)
```

Build a wheel or a set of wheels

Raises `TypeError` – Raised when no requirements are provided

Parameters

- **req** (*Optional*[*TInstallRequirement*]) – An *InstallRequirement* to build
- **reqset** (*Optional*[*TReqSet*]) – A *RequirementSet* instance (*pip*<10) or an iterable of *InstallRequirement* instances (*pip*>=10) to build
- **output_dir** (*Optional*[*str*]) – Target output directory, only useful when building one wheel using *pip*>=20.0
- **preparer** (*Optional*[*TPreparer*]) – A preparer instance, defaults to None
- **wheel_cache** (*Optional*[*TWheelCache*]) – A wheel cache instance, defaults to None
- **build_options** (*Optional*[*List*[*str*]]) – A list of build options to pass in
- **global_options** (*Optional*[*List*[*str*]]) – A list of global options to pass in
- **bool]] check_binary_allowed** (*Optional*[*Callable*[*TInstallRequirement*,]]) – A callable to check whether we are allowed to build and cache wheels for an ireq
- **no_clean** (*bool*) – Whether to avoid cleaning up wheels
- **session** (*Optional*[*TSession*]) – A *PipSession* instance to pass to create a *finder* if necessary
- **finder** (*Optional*[*TFinder*]) – A *PackageFinder* instance to use for generating a *WheelBuilder* instance on *pip*<20
- **install_command** (*Optional*[*TCommandInstance*]) – The install command used to create the finder, session, and options if needed, defaults to None.

- **req_tracker** (*Optional*[*TReqTracker*]) – An optional requirement tracker instance, if one already exists
- **build_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **src_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **download_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **wheel_download_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **cache_dir** (*Optional*[*str*]) – Passthrough cache directory for wheel cache options
- **use_user_site** (*bool*) – Whether to use the user site directory when preparing install requirements on *pip*<20
- **use_pep517** (*Optional*[*bool*]) – When set to *True* or *False*, prefers building with or without pep517 as specified, otherwise uses requirement preference. Only works for single requirements.
- **format_control_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *FormatControl* class
- **wheel_cache_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *WheelCache* class
- **preparer_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *RequirementPreparer* class
- **wheel_builder_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *WheelBuilder* class, if it exists
- **build_one_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *_build_one* function, if it exists
- **build_one_inside_env_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *_build_one_inside_env* function, if it exists
- **build_many_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *build* function, if it exists
- **install_command_provider** (*Optional*[*TShimmedFunc*]) – A shim for providing new install command instances
- **finder_provider** (*TShimmedFunc*) – A provider to package finder instances
- **reqset_provider** (*TShimmedFunc*) – A provider for requirement set generation

Returns A tuple of successful and failed install requirements or else a path to a wheel

Return type `Optional[Union[str, Tuple[List[TInstallRequirement], List[TInstallRequirement]]]]`

2.2 Submodules

2.2.1 pip_shims.compat module

Backports and helper functionality to support using new functionality.

```

class pip_shims.compat.CandidateEvaluator(project_name, supported_tags, specifier, prefer_binary=False, allow_all_prereleases=False, hashes=None)

    Bases: object

    classmethod create(project_name, target_python=None, prefer_binary=False, allow_all_prereleases=False, specifier=None, hashes=None)

class pip_shims.compat.CandidatePreferences(prefer_binary=False, allow_all_prereleases=False)

    Bases: object

exception pip_shims.compat.InvalidWheelFilename

    Bases: Exception

    Wheel Filename is Invalid

    args

    with_traceback()
        Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class pip_shims.compat.LinkCollector(session=None, search_scope=None)

    Bases: object

class pip_shims.compat.LinkEvaluator(allow_yanked, project_name, canonical_name, formats, target_python, ignore_requires_python=False, ignore_compatibility=True)

    Bases: object

class pip_shims.compat.SearchScope(find_links=None, index_urls=None)

    Bases: object

    classmethod create(find_links=None, index_urls=None)

class pip_shims.compat.SelectionPreferences(allow_yanked=True, allow_all_prereleases=False, format_control=None, prefer_binary=False, ignore_requires_python=False)

    Bases: object

class pip_shims.compat.TargetPython(platform=None, py_version_info=None, abi=None, implementation=None)

    Bases: object

    fallback_get_tags = <pip_shims.models.ShimmedPathCollection object>

    get_tags()

class pip_shims.compat.Wheel(filename)

    Bases: object

    get_formatted_file_tags()
        Return the wheel's tags as a sorted list of strings.

    support_index_min(tags)
        Return the lowest index that one of the wheel's file_tag combinations achieves in the given list of supported tags.

        For example, if there are 8 supported tags and one of the file tags is first in the list, then return 0.

        Parameters tags – the PEP 425 tags to check the wheel against, in order with most preferred first.

        Raises ValueError – If none of the wheel's file tags match one of the supported tags.

```

supported (*tags*)

Return whether the wheel is compatible with one of the given tags.

Parameters *tags* – the PEP 425 tags to check the wheel against.

wheel_file_re = `re.compile('^(?P<namever>(?P<name>.+?) - (?P<ver>. *?)) \n ((- (?P<build>\`

`pip_shims.compat._ensure_finder` (*finder=None, finder_provider=None, install_cmd=None, options=None, session=None*)

`pip_shims.compat._ensure_wheel_cache` (*wheel_cache=None, wheel_cache_provider=None, format_control=None, format_control_provider=None, options=None, cache_dir=None*)

`pip_shims.compat.build_wheel` (*req=None, reqset=None, output_dir=None, preparer=None, wheel_cache=None, build_options=None, global_options=None, check_binary_allowed=None, no_clean=False, session=None, finder=None, install_command=None, req_tracker=None, build_dir=None, src_dir=None, download_dir=None, wheel_download_dir=None, cache_dir=None, use_user_site=False, use_pep517=None, format_control_provider=None, wheel_cache_provider=None, preparer_provider=None, wheel_builder_provider=None, build_one_provider=None, build_one_inside_env_provider=None, build_many_provider=None, install_command_provider=None, finder_provider=None, reqset_provider=None*)

Build a wheel or a set of wheels

Raises **TypeError** – Raised when no requirements are provided

Parameters

- **req** (*Optional[`TInstallRequirement`]*) – An *InstallRequirement* to build
- **reqset** (*Optional[`TReqSet`]*) – A *RequirementSet* instance (*pip<10*) or an iterable of *InstallRequirement* instances (*pip>=10*) to build
- **output_dir** (*Optional[`str`]*) – Target output directory, only useful when building one wheel using *pip>=20.0*
- **preparer** (*Optional[`TPreparer`]*) – A preparer instance, defaults to `None`
- **wheel_cache** (*Optional[`TWheelCache`]*) – A wheel cache instance, defaults to `None`
- **build_options** (*Optional[`List[str]`]*) – A list of build options to pass in
- **global_options** (*Optional[`List[str]`]*) – A list of global options to pass in
- **bool]] check_binary_allowed** (*Optional[`Callable[TInstallRequirement,]`]*) – A callable to check whether we are allowed to build and cache wheels for an *ireq*
- **no_clean** (*`bool`*) – Whether to avoid cleaning up wheels
- **session** (*Optional[`TSession`]*) – A *PipSession* instance to pass to create a *finder* if necessary
- **finder** (*Optional[`TFinder`]*) – A *PackageFinder* instance to use for generating a *WheelBuilder* instance on *pip<20*
- **install_command** (*Optional[`TCommandInstance`]*) – The install command used to create the *finder*, *session*, and *options* if needed, defaults to `None`.

- **req_tracker** (*Optional*[*TReqTracker*]) – An optional requirement tracker instance, if one already exists
- **build_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **src_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **download_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **wheel_download_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **cache_dir** (*Optional*[*str*]) – Passthrough cache directory for wheel cache options
- **use_user_site** (*bool*) – Whether to use the user site directory when preparing install requirements on *pip*<20
- **use_pep517** (*Optional*[*bool*]) – When set to *True* or *False*, prefers building with or without pep517 as specified, otherwise uses requirement preference. Only works for single requirements.
- **format_control_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *FormatControl* class
- **wheel_cache_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *WheelCache* class
- **preparer_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *RequirementPreparer* class
- **wheel_builder_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *WheelBuilder* class, if it exists
- **build_one_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *_build_one* function, if it exists
- **build_one_inside_env_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *_build_one_inside_env* function, if it exists
- **build_many_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *build* function, if it exists
- **install_command_provider** (*Optional*[*TShimmedFunc*]) – A shim for providing new install command instances
- **finder_provider** (*TShimmedFunc*) – A provider to package finder instances
- **reqset_provider** (*TShimmedFunc*) – A provider for requirement set generation

Returns A tuple of successful and failed install requirements or else a path to a wheel

Return type *Optional*[*Union*[*str*, *Tuple*[*List*[*TInstallRequirement*], *List*[*TInstallRequirement*]]]

`pip_shims.compat.ensure_resolution_dirs (**kwargs)`

Ensures that the proper directories are scaffolded and present in the provided kwargs for performing dependency resolution via pip.

Returns A new kwargs dictionary with scaffolded directories for **build_dir**, **src_dir**, **download_dir**, and **wheel_download_dir** added to the key value pairs.

Return type *Dict*[*str*, *Any*]

`pip_shims.compat.get_ireq_output_path (wheel_cache, ireq)`

```
pip_shims.compat.get_package_finder(install_cmd=None, options=None, session=None,
                                   platform=None, python_versions=None,
                                   abi=None, implementation=None, target_python=None,
                                   ignore_requires_python=None, target_python_builder=None,
                                   install_cmd_provider=None)
```

Shim for compatibility to generate package finders.

Build and return a PackageFinder instance using the InstallCommand helper method to construct the finder, shimmed with backports as needed for compatibility.

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_cmd** – A `InstallCommand` instance which is used to generate the finder.
- **options** (*optparse.Values*) – An optional `optparse.Values` instance generated by calling `install_cmd.parser.parse_args()` typically.
- **session** – An optional session instance, can be created by the `install_cmd`.
- **platform** (*Optional[str]*) – An optional platform string, e.g. `linux_x86_64`
- **python_versions** (*Optional[Tuple[str, ...]]*) – A tuple of 2-digit strings representing python versions, e.g. `("27", "35", "36", "37"…)`
- **abi** (*Optional[str]*) – The target abi to support, e.g. `"cp38"`
- **implementation** (*Optional[str]*) – An optional implementation string for limiting searches to a specific implementation, e.g. `"cp"` or `"py"`
- **target_python** – A `TargetPython` instance (will be translated to alternate arguments if necessary on incompatible pip versions).
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore `requires_python` on resulting candidates, only valid after pip version 19.3.1
- **target_python_builder** – A ‘TargetPython’ builder (e.g. the class itself, uninstantiated)

Returns A `pip._internal.index.package_finder.PackageFinder` instance

Return type `pip._internal.index.package_finder.PackageFinder`

Example

```
>>> from pip_shims.shims import InstallCommand, get_package_finder
>>> install_cmd = InstallCommand()
>>> finder = get_package_finder(
...     install_cmd, python_versions=("27", "35", "36", "37", "38"),
...     implementation="
cp"
... )
>>> candidates = finder.find_all_candidates("requests")
>>> requests_222 = next(iter(c for c in candidates if c.version.public == "2.22.0
... ))
>>> requests_222
<InstallationCandidate('requests', <Version('2.22.0')>, <Link https://files.
pythonhos
ted.org/packages/51/bd/
23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/r
```

(continues on next page)

(continued from previous page)

```

requests-2.22.0-py2.py3-none-any.whl
↪#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9
a590f48c010551dc6c4b31 (from https://pypi.org/simple/requests/) (requires-python:>
↪=2.
7, !=3.0.*, !=3.1.*, !=3.2.*, !=3.3.*, !=3.4.*)>>>

```

```

pip_shims.compat.get_requirement_set (install_command=None, req_set_provider=None,
                                     build_dir=None, src_dir=None, download_dir=None,
                                     wheel_download_dir=None, session=None,
                                     wheel_cache=None, upgrade=False, upgrade_strategy=None,
                                     ignore_installed=False, ignore_dependencies=False,
                                     force_reinstall=False, use_user_site=False,
                                     isolated=False, ignore_requires_python=False,
                                     require_hashes=None, cache_dir=None,
                                     options=None, install_cmd_provider=None,
                                     wheel_cache_provider=None)

```

Creates a requirement set from the supplied parameters.

Not all parameters are passed through for all pip versions, but any invalid parameters will be ignored if they are not needed to generate a requirement set on the current pip version.

:param *ShimmedPathCollection* wheel_cache_provider: A context manager provider which resolves to a *WheelCache* instance

Parameters install_command – A *InstallCommand* instance which is used to generate the finder.

:param *ShimmedPathCollection* req_set_provider: A provider to build requirement set instances.

Parameters

- **build_dir** (*str*) – The directory to build requirements in. Removed in pip 10, defaults to None
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*str*) – The directory to download requirement artifacts to. Removed in pip 10, defaults to None
- **wheel_download_dir** (*str*) – The directory to download wheels to. Removed in pip 10, defaults to None

:param *Session* session: The pip session to use. Removed in pip 10, defaults to None

Parameters

- **wheel_cache** (*WheelCache*) – The pip *WheelCache* instance to use for caching wheels. Removed in pip 10, defaults to None
- **upgrade** (*bool*) – Whether to try to upgrade existing requirements. Removed in pip 10, defaults to False.
- **upgrade_strategy** (*str*) – The upgrade strategy to use, e.g. “only-if-needed”. Removed in pip 10, defaults to None.

- **ignore_installed** (*bool*) – Whether to ignore installed packages when resolving. Removed in pip 10, defaults to False.
- **ignore_dependencies** (*bool*) – Whether to ignore dependencies of requirements when resolving. Removed in pip 10, defaults to False.
- **force_reinstall** (*bool*) – Whether to force reinstall of packages when resolving. Removed in pip 10, defaults to False.
- **use_user_site** (*bool*) – Whether to use user site packages when resolving. Removed in pip 10, defaults to False.
- **isolated** (*bool*) – Whether to resolve in isolation. Removed in pip 10, defaults to False.
- **ignore_requires_python** (*bool*) – Removed in pip 10, defaults to False.
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **options** (*Values*) – An *Values* instance from an install cmd
- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.

Returns A new requirement set instance

Return type RequirementSet

`pip_shims.compat.get_requirement_tracker` (*req_tracker_creator=None*)

`pip_shims.compat.get_resolver` (*resolver_fn*, *install_req_provider=None*, *format_control_provider=None*, *wheel_cache_provider=None*, *finder=None*, *upgrade_strategy='to-satisfy-only'*, *force_reinstall=None*, *ignore_dependencies=None*, *ignore_requires_python=None*, *ignore_installed=True*, *use_user_site=False*, *isolated=None*, *wheel_cache=None*, *preparer=None*, *session=None*, *options=None*, *make_install_req=None*, *install_cmd_provider=None*, *install_cmd=None*, *use_pep517=True*)

A resolver creation compatibility shim for generating a resolver.

Consumes any argument that was previously used to instantiate a resolver, discards anything that is no longer valid.

Note: This is only valid for **pip >= 10.0.0**

Raises

- **ValueError** – A session is required but not provided and one cannot be created
- **ValueError** – A finder is required but not provided and one cannot be created
- **ValueError** – An install requirement provider is required and has not been provided

Parameters

- **resolver_fn** (*TShimmedFunc*) – The resolver function used to create new resolver instances.
- **install_req_provider** (*TShimmedFunc*) – The provider function to use to generate install requirements if needed.

- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a `format_control` instance if needed.
- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to `None`.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package reinstallation during resolution, defaults to `None`
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to `None`
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated `required_python` versions on packages, defaults to `None`
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to `True`
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to `False`
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to `None`
- **wheel_cache** (*Optional[TWheelCache]*) – The wheel cache to use, defaults to `None`
- **preparer** (*Optional[TPreparer]*) – The requirement preparer to use, defaults to `None`
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to `None`
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to `None`
- **make_install_req** (*Optional[functools.partial]*) – The partial function to pass in to the resolver for actually generating install requirements, if necessary
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to `None`.
- **use_pep517** (*bool*) – Whether to use the pep517 build process.

Returns A new resolver instance.

Return type `Resolver`

Example

```
>>> import os
>>> from tempdir import TemporaryDirectory
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_
↳ tracker,
...     get_resolver, InstallRequirement, RequirementSet
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([]
```

(continues on next page)

(continued from previous page)

```

>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> wheel_cache = WheelCache(USER_CACHE_DIR, FormatControl(None, None))
>>> with TemporaryDirectory() as temp_base:
...     reqset = RequirementSet()
...     ireq = InstallRequirement.from_line("requests")
...     ireq.is_direct = True
...     build_dir = os.path.join(temp_base, "build")
...     src_dir = os.path.join(temp_base, "src")
...     ireq.build_location(build_dir)
...     with make_preparer(
...         options=pip_options, finder=finder, session=session,
...         build_dir=build_dir, install_cmd=install_cmd,
...     ) as preparer:
...         resolver = get_resolver(
...             finder=finder, ignore_dependencies=False, ignore_requires_
↳python=True,
...             preparer=preparer, session=session, options=pip_options,
...             install_cmd=install_cmd, wheel_cache=wheel_cache,
...         )
...         resolver.require_hashes = False
...         reqset.add_requirement(ireq)
...         results = resolver.resolve(reqset)
...         #reqset.cleanup_files()
...         for result_req in reqset.requirements:
...             print(result_req)
requests
chardet
certifi
urllib3
idna

```

`pip_shims.compat.get_session` (*install_cmd_provider=None, install_cmd=None, options=None*)

`pip_shims.compat.make_preparer` (*preparer_fn, req_tracker_fn=None, build_dir=None, src_dir=None, download_dir=None, wheel_download_dir=None, progress_bar='off', build_isolation=False, session=None, finder=None, options=None, require_hashes=None, use_user_site=None, req_tracker=None, install_cmd_provider=None, downloader_provider=None, install_cmd=None, finder_provider=None*)

Creates a requirement preparer for preparing pip requirements.

Provides a compatibility shim that accepts all previously valid arguments and discards any that are no longer used.

Raises

- **TypeError** – No requirement tracker provided and one cannot be generated
- **TypeError** – No valid sessions provided and one cannot be generated
- **TypeError** – No valid finders provided and one cannot be generated

Parameters

- **preparer_fn** (*TShimmedFunc*) – Callable or shim for generating preparers.

- **req_tracker_fn** (*Optional*[*TShimmedFunc*]) – Callable or shim for generating requirement trackers, defaults to None
- **build_dir** (*Optional*[*str*]) – Directory for building packages and wheels, defaults to None
- **src_dir** (*Optional*[*str*]) – Directory to find or extract source files, defaults to None
- **download_dir** (*Optional*[*str*]) – Target directory to download files, defaults to None
- **wheel_download_dir** (*Optional*[*str*]) – Target directory to download wheels, defaults to None
- **progress_bar** (*str*) – Whether to display a progress bar, defaults to off
- **build_isolation** (*bool*) – Whether to build requirements in isolation, defaults to False
- **session** (*Optional*[*TSession*]) – Existing session to use for getting requirements, defaults to None
- **finder** (*Optional*[*TFinder*]) – The package finder to use during resolution, defaults to None
- **options** (*Optional*[*Values*]) – Pip options to use if needed, defaults to None
- **require_hashes** (*Optional*[*bool*]) – Whether to require hashes for preparation
- **use_user_site** (*Optional*[*bool*]) – Whether to use the user site directory for preparing requirements
- **TShimmedFunc**] **req_tracker** (*Optional*[*Union*[*TReqTracker*,)]) – The requirement tracker to use for building packages, defaults to None
- **downloader_provider** (*Optional*[*TShimmedFunc*]) – A downloader provider
- **install_cmd** (*Optional*[*TCommandInstance*]) – The install command used to create the finder, session, and options if needed, defaults to None
- **finder_provider** (*Optional*[*TShimmedFunc*]) – A package finder provider

Yield A new requirement preparer instance

Return type `ContextManager[RequirementPreparer]`

Example

```
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_tracker
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> with make_preparer(
...     options=pip_options, finder=finder, session=session, install_cmd=ic
... ) as preparer:
...     print(preparer)
<pip._internal.operations.prepare.RequirementPreparer object at 0x7f8a2734be80>
```

`pip_shims.compat.partial_command` (*shimmed_path*, *cmd_mapping=None*)

Maps a default set of arguments across all members of a *ShimmedPath* instance, specifically for Command instances which need *summary* and *name* arguments.

:param *ShimmedPath* shimmed_path: A *ShimmedCollection* instance

Parameters *cmd_mapping* (*Any*) – A reference to use for mapping against, e.g. an import that depends on pip also

Returns A dictionary mapping new arguments to their default values

Return type Dict[str, str]

`pip_shims.compat.populate_options` (*install_command=None*, *options=None*, ***kwargs*)

`pip_shims.compat.resolve` (*ireq*, *reqset_provider=None*, *req_tracker_provider=None*, *install_cmd_provider=None*, *install_command=None*, *finder_provider=None*, *resolver_provider=None*, *wheel_cache_provider=None*, *format_control_provider=None*, *make_preparer_provider=None*, *tempdir_manager_provider=None*, *options=None*, *session=None*, *resolver=None*, *finder=None*, *upgrade_strategy='to-satisfy-only'*, *force_reinstall=None*, *ignore_dependencies=None*, *ignore_requires_python=None*, *ignore_installed=True*, *use_user_site=False*, *isolated=None*, *build_dir=None*, *source_dir=None*, *download_dir=None*, *cache_dir=None*, *wheel_download_dir=None*, *wheel_cache=None*, *require_hashes=None*, *check_supported_wheels=True*)

Resolves the provided **InstallRequirement**, returning a dictionary.

Maps a dictionary of names to corresponding *InstallRequirement* values.

:param *InstallRequirement* ireq: An *InstallRequirement* to initiate the resolution process

:param *ShimmedPathCollection* reqset_provider: A **provider** to build requirement set instances.

:param *ShimmedPathCollection* req_tracker_provider: A **provider** to build requirement tracker instances

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_command** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None.

:param *ShimmedPathCollection* finder_provider: A **provider** to package finder instances.

:param *ShimmedPathCollection* resolver_provider: A **provider** to build resolver instances

Parameters

- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a *format_control* instance if needed.
- **make_preparer_provider** (*TShimmedFunc*) – Callable or shim for generating preparers.

- **tempdir_manager_provider** (*Optional* [*TShimmedFunc*]) – Shim for generating tempdir manager for pip temporary directories
- **options** (*Optional* [*Values*]) – Pip options to use if needed, defaults to None
- **session** (*Optional* [*TSession*]) – Existing session to use for getting requirements, defaults to None

:param Resolver resolver: A pre-existing resolver instance to use for resolution

Parameters

- **finder** (*Optional* [*TFinder*]) – The package finder to use during resolution, defaults to None.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to *only-if-needed*.
- **force_reinstall** (*Optional* [*bool*]) – Whether to simulate or assume package re-installation during resolution, defaults to None
- **ignore_dependencies** (*Optional* [*bool*]) – Whether to ignore package dependencies, defaults to None
- **ignore_requires_python** (*Optional* [*bool*]) – Whether to ignore indicated required_python versions on packages, defaults to None
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to True
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to False
- **isolated** (*Optional* [*bool*]) – Whether to isolate the resolution process, defaults to None
- **build_dir** (*Optional* [*str*]) – Directory for building packages and wheels, defaults to None
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*Optional* [*str*]) – Target directory to download files, defaults to None
- **cache_dir** (*str*) – The cache directory to use for caching artifacts during resolution
- **wheel_download_dir** (*Optional* [*str*]) – Target directory to download wheels, defaults to None
- **wheel_cache** (*Optional* [*TWheelCache*]) – The wheel cache to use, defaults to None
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **check_supported_wheels** (*bool*) – Whether to check support of wheels before including them in resolution.

Returns A dictionary mapping requirements to corresponding `:class:`~pip._internal.req.req_install.InstallRequirement`'s`

Return type `InstallRequirement`

Example

```

>>> from pip_shims.shims import resolve, InstallRequirement
>>> ireq = InstallRequirement.from_line("requests>=2.20")
>>> results = resolve(ireq)
>>> for k, v in results.items():
...     print("{0}: {1!r}".format(k, v))
requests: <InstallRequirement object: requests>=2.20 from https://files.
↳pythonhosted.
org/packages/51/bd/23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/
↳reque
sts-2.22.0-py2.py3-none-any.whl
↳#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9a590
f48c010551dc6c4b31 editable=False>
idna: <InstallRequirement object: idna<2.9,>=2.5 from https://files.pythonhosted.
↳org/
packages/14/2c/cd551d81dbe15200belcf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-
↳2.8-
py2.py3-none-any.whl
↳#sha256=ea8b7f6188e6fa117537c3df7da9fc686d485087abf6ac197f9c46432
f7e4a3c (from requests>=2.20) editable=False>
urllib3: <InstallRequirement object: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 from
↳htt
↳ps://files.pythonhosted.org/packages/b4/40/
↳a9837291310eelccc242ceb6ebfd9eb21539649f19
3a7c8c86ba15b98539/urllib3-1.25.7-py2.py3-none-any.whl
↳#sha256=a8a318824cc77d1fd4b2bec
2ded92646630d7fe8619497b142c84a9e6f5a7293 (from requests>=2.20) editable=False>
chardet: <InstallRequirement object: chardet<3.1.0,>=3.0.2 from https://files.
↳pythonh
↳sted.org/packages/bc/a9/
↳01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8
/chardet-3.0.4-py2.py3-none-any.whl
↳#sha256=fc323ffcaeaed0e0a02bf4d117757b98aed530d9ed
4531e3e15460124c106691 (from requests>=2.20) editable=False>
certifi: <InstallRequirement object: certifi>=2017.4.17 from https://files.
↳pythonhost
ed.org/packages/18/b0/
↳8146a4f8dd402f60744fa380bc73ca47303ccc8b9190fd16a827281eac2/ce
rtifi-2019.9.11-py2.py3-none-any.whl
↳#sha256=fd7c7c74727ddcf00e9acd26bba8da604ffec95bf
1c2144e67aff7a8b50e6cef (from requests>=2.20) editable=False>

```

`pip_shims.compat.resolve_possible_shim(target)`

`pip_shims.compat.shim_unpack(unpack_fn, download_dir, tempdir_manager_provider, ireq=None, link=None, location=None, hashes=None, progress_bar='off', only_download=None, downloader_provider=None, session=None)`

Accepts all parameters that have been valid to pass to `pip._internal.download.unpack_url()` and selects or drops parameters as needed before invoking the provided callable.

Parameters

- **unpack_fn** (*Callable*) – A callable or shim referring to the pip implementation
- **download_dir** (*str*) – The directory to download the file to
- **tempdir_manager_provider** (*TShimmedFunc*) – A callable or shim referring to *global_tempdir_manager* function from pip or a shimmed no-op context manager

:param Optional[InstallRequirement] ireq: an Install Requirement instance, defaults to None

:param Optional[Link] link: A Link instance, defaults to None.

Parameters

- **location** (*Optional[str]*) – A location or source directory if the target is a VCS url, defaults to None.
- **hashes** (*Optional[Any]*) – A Hashes instance, defaults to None
- **progress_bar** (*str*) – Indicates progress par usage during download, defatuls to off.
- **only_download** (*Optional[bool]*) – Whether to skip install, defaults to None.
- **downloader_provider** (*Optional[ShimmedPathCollection]*) – A downloader class to instantiate, if applicable.
- **session** (*Optional[Session]*) – A PipSession instance, defaults to None.

Returns The result of unpacking the url.

Return type `None`

```
pip_shims.compat.temp_environ()
```

Allow the ability to set os.environ temporarily

```
pip_shims.compat.wheel_cache(cache_dir=None, format_control=None,
                             wheel_cache_provider=None, format_control_provider=None,
                             tempdir_manager_provider=None)
```

2.2.2 pip_shims.environment module

Module with functionality to learn about the environment.

```
pip_shims.environment.get_base_import_path()
```

```
pip_shims.environment.get_pip_version(import_path='pip')
```

```
pip_shims.environment.is_type_checking()
```

2.2.3 pip_shims.models module

Helper module for shimming functionality across pip versions.

```
class pip_shims.models.ImportTypes
```

Bases: `pip_shims.models.ImportTypes`

Create new instance of ImportTypes(FUNCTION, CLASS, MODULE, CONTEXTMANAGER)

ATTRIBUTE = 5

CLASS = 1

CONTEXTMANAGER = 3

FUNCTION = 0

METHOD = 4

MODULE = 2

```

_asdict ()
    Return a new OrderedDict which maps field names to their values.

_fields = ('FUNCTION', 'CLASS', 'MODULE', 'CONTEXTMANAGER')

_fields_defaults = {}

classmethod _make (iterable)
    Make a new ImportTypes object from a sequence or iterable

_replace (**kws)
    Return a new ImportTypes object replacing specified fields with new values

count ()
    Return number of occurrences of value.

index ()
    Return first index of value.

    Raises ValueError if the value is not present.

```

```

pip_shims.models.ImportTypesBase
    alias of pip_shims.models.ImportTypes

```

```

class pip_shims.models.PipVersion (version, round_prereleases_up=True,
                                     base_import_path=None, vendor_import_path='pip._vendor')

```

```

Bases: collections.abc.Sequence

```

```

_abc_impl = <_abc_data object>

```

```

_parse ()

```

```

count (value) → integer – return number of occurrences of value

```

```

index (value[, start[, stop]]) → integer – return first index of value.
    Raises ValueError if the value is not present.

```

```

    Supporting start and stop arguments is optional, but recommended.

```

```

is_valid (compared_to)

```

```

version_key

```

```

version_tuple

```

```

class pip_shims.models.PipVersionRange (start, end)

```

```

Bases: collections.abc.Sequence

```

```

_abc_impl = <_abc_data object>

```

```

base_import_paths

```

```

count (value) → integer – return number of occurrences of value

```

```

index (value[, start[, stop]]) → integer – return first index of value.
    Raises ValueError if the value is not present.

```

```

    Supporting start and stop arguments is optional, but recommended.

```

```

is_valid ()

```

```

vendor_import_paths

```

```
class pip_shims.models.ShimmedPath(name, import_target, import_type, version_range,
provided_methods=None, provided_functions=None, provided_classmethods=None,
provided_contextmanagers=None, provided_mixins=None, default_args=None)
```

Bases: `object`

```
_ShimmedPath__modules = {'pip._internal.cache': <module 'pip._internal.cache' from '/
```

```
_apply_aliases (imported, target)
```

```
_as_tuple ()
```

```
_ensure_functions (provided)
```

```
_ensure_methods (provided)
```

Given a base class, a new name, and any number of functions to attach, turns those functions into class-methods, attaches them, and returns an updated class object.

```
_import (prefix=None)
```

```
classmethod _import_module (module)
```

```
classmethod _parse_provides_dict (provides, prepend_arg_to_callables=None)
```

```
_shim_base (imported, attribute_name)
```

```
_shim_parent (imported, attribute_name)
```

```
_update_default_kwargs (parent, provided)
```

```
alias (aliases)
```

```
calculated_module_path
```

```
is_attribute
```

```
is_class
```

```
is_contextmanager
```

```
is_function
```

```
is_method
```

```
is_module
```

```
is_valid
```

```
shim ()
```

```
shim_attribute (imported, attribute_name)
```

```
shim_class (imported, attribute_name)
```

```
shim_contextmanager (imported, attribute_name)
```

```
shim_function (imported, attribute_name)
```

```
shim_module (imported, attribute_name)
```

```
shimmed
```

```
sort_order
```

```
update_sys_modules (imported)
```

```
class pip_shims.models.ShimmedPathCollection(name, import_type, paths=None)
```

Bases: `object`


```

__ShimmedPathCollection__registry = {'AbstractDistribution': <pip_shims.models.Shimmed
__get_top_path()
__sort_paths()
add_mixin(mixin)
add_path(path)
alias(aliases)
    Takes a list of methods, functions, attributes, etc and ensures they all exist on the object pointing at the
    same referent.

    Parameters aliases (List[str]) – Names to map to the same functionality if they do not
        exist.

    Returns None

    Return type None

create_path(import_path, version_start, version_end=None)
classmethod get_registry()
pre_shim(fn)
provide_function(name, fn)
provide_method(name, fn)
register()
set_default(default)
set_default_args(callable_name, *args, **kwargs)
shim()
classmethod traverse(shim)

```

```

pip_shims.models.import_pip()
pip_shims.models.lookup_current_pip_version()
pip_shims.models.pip_version_lookup(version, *args, **kwargs)

```

2.2.4 pip_shims.shims module

Main module with magic self-replacement mechanisms to handle import speedups.

```

pip_shims.shims._strip_extras(path)

class pip_shims.shims.SessionCommandMixin
    Bases: pip._internal.cli.command_context.CommandContextMixin
    A class mixin for command classes needing _build_session().
    _build_session(options, retries=None, timeout=None)
    classmethod _get_index_urls(options)
        Return a list of index urls from user-provided options.
    enter_context(context_provider)
    get_default_session(options)
        Get a default-managed session.

```

```

    main_context ()

class pip_shims.shims.Command(name='Default pip command.', summary='PipCommand', iso-
                             lated='Default pip command.')
    Bases: pip._internal.cli.base_command.Command, pip._internal.cli.req_command.
           SessionCommandMixin

    _build_session (options, retries=None, timeout=None)

    classmethod _get_index_urls (options)
        Return a list of index urls from user-provided options.

    _main (args)

    add_options ()

    enter_context (context_provider)

    get_default_session (options)
        Get a default-managed session.

    handle_pip_version_check (options)
        This is a no-op so that commands by default do not do the pip version check.

    ignore_require_venv = False

    main (args)

    main_context ()

    parse_args (args)

    run (options, args)

    usage = None

class pip_shims.shims.ConfigOptionParser(*args, **kwargs)
    Bases: pip._internal.cli.parser.CustomOptionParser

    Custom option parser which updates its defaults by checking the configuration files and environmental variables

    _add_help_option ()

    _add_version_option ()

    _check_conflict (option)

    _create_option_list ()

    _create_option_mappings ()

    _get_all_options ()

    _get_args (args)

    _get_ordered_configuration_items ()

    _init_parsing_state ()

    _match_long_opt (opt : string) → string
        Determine which long option string 'opt' matches, ie. which one it is an unambiguous abbreviation for.
        Raises BadOptionError if 'opt' doesn't unambiguously match any long option string.

    _populate_option_list (option_list, add_help=True)

    _process_args (largs, rargs, values)

        _process_args(largs [[string],] rargs : [string], values : Values)

```

Process command-line arguments and populate ‘values’, consuming options and arguments from ‘rargs’. If ‘allow_interspersed_args’ is false, stop at the first non-option argument. If true, accumulate any interspersed non-option arguments in ‘larges’.

_process_long_opt (*rargs, values*)

_process_short_opts (*rargs, values*)

_share_option_mappings (*parser*)

_update_defaults (*defaults*)

Updates the given defaults with values from the config files and the environ. Does a little special handling for certain types of options (lists).

add_option (*Option*)

add_option(opt_str, ..., kwarg=val, ...)

add_option_group (**args, **kwargs*)

add_options (*option_list*)

check_default (*option, key, val*)

check_values (*values : Values, args : [string]*)

-> (values : Values, args : [string])

Check that the supplied option values and leftover arguments are valid. Returns the option values and leftover arguments (possibly adjusted, possibly completely new – whatever you like). Default implementation just returns the passed-in values; subclasses may override as desired.

destroy ()

Declare that you are done with this OptionParser. This cleans up reference cycles so the OptionParser (and all objects referenced by it) can be garbage-collected promptly. After calling destroy(), the OptionParser is unusable.

disable_interspersed_args ()

Set parsing to stop on the first non-option. Use this if you have a command processor which runs another command that has options of its own and you want to make sure these options don’t get confused.

enable_interspersed_args ()

Set parsing to not stop on the first non-option, allowing interspersing switches with command arguments. This is the default behavior. See also disable_interspersed_args() and the class documentation description of the attribute allow_interspersed_args.

error (*msg : string*)

Print a usage message incorporating ‘msg’ to stderr and exit. If you override this in a subclass, it should not return – it should either exit or raise an exception.

exit (*status=0, msg=None*)

expand_prog_name (*s*)

format_description (*formatter*)

format_epilog (*formatter*)

format_help (*formatter=None*)

format_option_help (*formatter=None*)

get_default_values ()

Overriding to make updating the defaults after instantiation of the option parser possible, _update_defaults() does the dirty work.

get_description ()

get_option (*opt_str*)

get_option_group (*opt_str*)

get_prog_name ()

get_usage ()

get_version ()

has_option (*opt_str*)

insert_option_group (*idx, *args, **kwargs*)

Insert an OptionGroup at a given position.

option_list_all

Get a list of all options, including those in option groups.

parse_args (*args=None, values=None*)

parse_args(*args* [[string] = sys.argv[1:],] *values* : Values = None)

-> (*values* : Values, *args* : [string])

Parse the command-line options found in 'args' (default: sys.argv[1:]). Any errors result in a call to 'error()', which by default prints the usage message to stderr and calls sys.exit() with an error message. On success returns a pair (values, args) where 'values' is a Values instance (with all your option values) and 'args' is the list of arguments left over after parsing options.

print_help (*file : file = stdout*)

Print an extended help message, listing all options and any help text provided with them, to 'file' (default stdout).

print_usage (*file : file = stdout*)

Print the usage message for the current program (self.usage) to 'file' (default stdout). Any occurrence of the string "%prog" in self.usage is replaced with the name of the current program (basename of sys.argv[0]). Does nothing if self.usage is empty or not defined.

print_version (*file : file = stdout*)

Print the version message for this program (self.version) to 'file' (default stdout). As with print_usage(), any occurrence of "%prog" in self.version is replaced by the current program's name. Does nothing if self.version is empty or undefined.

remove_option (*opt_str*)

set_conflict_handler (*handler*)

set_default (*dest, value*)

set_defaults (***kwargs*)

set_description (*description*)

set_process_default_values (*process*)

set_usage (*usage*)

standard_option_list = []

exception pip_shims.shims.DistributionNotFound

Bases: pip._internal.exceptions.InstallationError

Raised when a distribution cannot be found to satisfy a requirement

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class pip_shims.shims.**FormatControl** (*no_binary=None, only_binary=None*)

Bases: `object`

Helper for managing formats from which a package can be installed.

disallow_binaries()

get_allowed_formats (*canonical_name*)

static handle_mutual_excludes (*value, target, other*)

no_binary

only_binary

class pip_shims.shims.**FrozenRequirement** (*name, req, editable, comments=()*)

Bases: `object`

classmethod from_dist (*dist*)

pip_shims.shims.**get_installed_distributions** (*local_only=True, skip={'argparse', 'python', 'wsgiref'}, include_editables=True, editables_only=False, user_only=False, paths=None*)

Return a list of installed Distribution objects.

If `local_only` is True (default), only return installations local to the current virtualenv, if in a virtualenv.

`skip` argument is an iterable of lower-case project names to ignore; defaults to `stdlib_pkgs`

If `include_editables` is False, don't report editables.

If `editables_only` is True, only report editables.

If `user_only` is True, only report installations in the user site directory.

If `paths` is set, only report the distributions present at the specified list of locations.

exception pip_shims.shims.**InstallationError**

Bases: `pip._internal.exceptions.PipError`

General exception during installation

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pip_shims.shims.**UninstallationError**

Bases: `pip._internal.exceptions.PipError`

General exception during uninstallation

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pip_shims.shims.**RequirementsFileParseError**

Bases: `pip._internal.exceptions.InstallationError`

Raised when a general error occurs parsing a requirements file line.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pip_shims.shims.BestVersionAlreadyInstalled

Bases: pip._internal.exceptions.PipError

Raised when the most up-to-date version of a package is already installed.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pip_shims.shims.BadCommand

Bases: pip._internal.exceptions.PipError

Raised when virtualenv or a command is not found

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pip_shims.shims.CommandError

Bases: pip._internal.exceptions.PipError

Raised when there is an error in command-line arguments

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pip_shims.shims.PreviousBuildDirError

Bases: pip._internal.exceptions.PipError

Raised when there's a previous conflicting build directory

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

`pip_shims.shims.install_req_from_editable` (*editable_req*, *comes_from=None*,
use_pep517=None, *isolated=False*,
options=None, *constraint=False*,
user_supplied=False)

`pip_shims.shims.install_req_from_line` (*name*, *comes_from=None*, *use_pep517=None*,
isolated=False, *options=None*, *constraint=False*,
line_source=None, *user_supplied=False*)

Creates an InstallRequirement from a name, which might be a requirement, directory containing 'setup.py', filename, or URL.

Parameters line_source – An optional string describing where the line is from, for logging purposes in case of an error.

`pip_shims.shims.install_req_from_req_string` (*req_string*, *comes_from=None*, *iso-*
lated=False, *use_pep517=None*,
user_supplied=False)

```
class pip_shims.shims.InstallRequirement (req, comes_from, editable=False, link=None,
                                         markers=None, use_pep517=None, iso-
                                         lated=False, install_options=None,
                                         global_options=None, hash_options=None, con-
                                         straint=False, extras=(), user_supplied=False)
```

Bases: `pip._internal.req.req_install.InstallRequirement`

_generate_metadata ()

Invokes metadata generator functions, with the required arguments.

_get_archive_name (*path, parentdir, rootdir*)

_set_requirement ()

Set requirement after generating metadata.

archive (*build_dir*)

Saves archive to provided build_dir.

Used for saving downloaded VCS requirements as part of *pip download*.

assert_source_matches_version ()

build_location (*build_dir, autodelete, parallel_builds*)

check_if_exists (*use_user_site*)

Find an installed distribution that satisfies or conflicts with this requirement, and set `self.satisfied_by` or `self.should_reinstall` appropriately.

ensure_build_location (*build_dir, autodelete, parallel_builds*)

ensure_has_source_dir (*parent_dir, autodelete=False, parallel_builds=False*)

Ensure that a `source_dir` is set.

This will create a temporary build dir if the name of the requirement isn't known yet.

Parameters `parent_dir` – The ideal pip `parent_dir` for the `source_dir`. Generally `src_dir` for editables and `build_dir` for sdist.

Returns `self.source_dir`

format_debug ()

An un-tested helper for getting state, for debugging.

from_editable = `<pip_shims.utils.BaseMethod object>`

from_line = `<pip_shims.utils.BaseMethod object>`

from_path ()

Format a nice indicator to show where this “comes from”

get_dist ()

has_hash_options

Return whether any known-good hashes are specified as options.

These activate `--require-hashes` mode; hashes specified as part of a URL do not.

hashes (*trust_internet=True*)

Return a hash-comparer that considers my option- and URL-based hashes to be known-good.

Hashes in URLs—ones embedded in the requirements file, not ones downloaded from an index server—are almost peers with ones from flags. They satisfy `--require-hashes` (whether it was implicitly or explicitly activated) but do not activate it. `md5` and `sha224` are not allowed in flags, which should nudge people toward good algos. We always OR all hashes together, even ones from URLs.

Parameters **trust_internet** – Whether to trust URL-based (`#md5=...`) hashes downloaded from the internet, as by `populate_link()`

install (*install_options*, *global_options=None*, *root=None*, *home=None*, *prefix=None*, *warn_script_location=True*, *use_user_site=False*, *pycompile=True*)

installed_version

is_pinned

Return whether I am pinned to an exact version.

For example, `some-package==1.2` is pinned; `some-package>1.2` is not.

is_wheel

load_pyproject_toml ()

Load the `pyproject.toml` file.

After calling this routine, all of the attributes related to PEP 517 processing for this requirement have been set. In particular, the `use_pep517` attribute can be used to determine whether we should follow the PEP 517 or legacy (`setup.py`) code path.

match_markers (*extras_requested=None*)

metadata

name

prepare_metadata ()

Ensure that project metadata is available.

Under PEP 517, call the backend hook to prepare the metadata. Under legacy processing, call `setup.py egg-info`.

pyproject_toml_path

setup_py_path

specifier

uninstall (*auto_confirm=False*, *verbose=False*)

Uninstall the distribution currently satisfying this requirement.

Prompts before removing or modifying files unless `auto_confirm` is `True`.

Refuses to delete or modify files outside of `sys.prefix` - thus uninstallation within a virtual environment can only modify that virtual environment, even if the `virtualenv` is linked to global site-packages.

unpacked_source_directory

update_editable (*obtain=True*)

warn_on_mismatching_name ()

`pip_shims.shims.is_archive_file` (*name*)

Return `True` if *name* is considered as an archive file.

`pip_shims.shims.is_file_url` (*link*)

class `pip_shims.shims.Downloader` (*session*, *progress_bar*)

Bases: `object`

`pip_shims.shims.unpack_url` (*link*, *location*, *downloader*, *download_dir=None*, *hashes=None*)

Unpack *link* into *location*, downloading if required.

Parameters `hashes` – A Hashes object, one of whose embedded hashes must match, or HashMismatch will be raised. If the Hashes is empty, no matches are required, and unhashable types of requirements (like VCS ones, which would ordinarily raise HashUnsupported) are allowed.

`pip_shims.shims.is_installable_dir(path)`

Is path is a directory containing setup.py or pyproject.toml?

```
class pip_shims.shims.Link(url, comes_from=None, requires_python=None,
                           yanked_reason=None, cache_link_parsing=True)
```

Bases: `pip._internal.models.link.Link`

Parameters

- **url** – url of the resource pointed to (href of the link)
- **comes_from** – instance of HTMLPage where the link was found, or string.
- **requires_python** – String containing the *Requires-Python* metadata field, specified in PEP 345. This may be specified by a `data-requires-python` attribute in the HTML link tag, as described in PEP 503.
- **yanked_reason** – the reason the file has been yanked, if the file has been yanked, or None if the file hasn't been yanked. This is the value of the “data-yanked” attribute, if present, in a simple repository HTML link. If the file has been yanked but no reason was provided, this should be the empty string. See PEP 592 for more information and the specification.
- **cache_link_parsing** – A flag that is used elsewhere to determine whether resources retrieved from this link should be cached. PyPI index urls should generally have this set to False, for example.

`_compare(other, method)`

`_compare_key`

`_defining_class`

`_egg_fragment_re = re.compile('[#&]egg=(^[&]*)')`

`_hash_re = re.compile('(sha1|sha224|sha384|sha256|sha512|md5)=[a-f0-9+)]')`

`_parsed_url`

`_subdirectory_fragment_re = re.compile('[#&]subdirectory=(^[&]*)')`

`_url`

`cache_link_parsing`

`comes_from`

`egg_fragment`

`ext`

`file_path`

`filename`

`has_hash`

`hash`

`hash_name`

`is_artifact`

`is_existing_dir()`

is_file

is_hash_allowed (*hashes*)

Return True if the link has a hash and it is allowed.

is_vcs

is_wheel

is_yanked

netloc

This can contain auth information.

path

requires_python

scheme

show_url

splitext ()

subdirectory_fragment

url

url_without_fragment

yanked_reason

`pip_shims.shims.make_abstract_dist` (*install_req*)

Returns a Distribution for the given InstallRequirement

`pip_shims.shims.make_distribution_for_install_requirement` (*install_req*)

Returns a Distribution for the given InstallRequirement

`pip_shims.shims.make_option_group` (*group, parser*)

Return an OptionGroup object group – assumed to be dict with ‘name’ and ‘options’ keys parser – an optparse Parser

class `pip_shims.shims.PackageFinder` (*link_collector, target_python, allow_yanked, format_control=None, candidate_prefs=None, ignore_requires_python=None*)

Bases: `object`

This finds packages.

This is meant to match easy_install’s technique for looking for packages, by reading pages and looking for appropriate links.

This constructor is primarily meant to be used by the create() class method and from tests.

Parameters

- **format_control** – A FormatControl object, used to control the selection of source packages / binary packages when consulting the index and links.
- **candidate_prefs** – Options to use when creating a CandidateEvaluator object.

`_log_skipped_link` (*link, reason*)

`_sort_links` (*links*)

Returns elements of links in order, non-egg links first, egg links second, while eliminating duplicates

allow_all_prereleases

classmethod create (*link_collector, selection_prefs, target_python=None*)

Create a PackageFinder.

Parameters

- **selection_prefs** – The candidate selection preferences, as a SelectionPreferences object.
- **target_python** – The target Python interpreter to use when checking compatibility. If None (the default), a TargetPython object will be constructed from the running Python.

evaluate_links (*link_evaluator, links*)

Convert links that are candidates to InstallationCandidate objects.

find_all_candidates (*project_name*)

Find all available InstallationCandidate for project_name

This checks index_urls and find_links. All versions found are returned as an InstallationCandidate list.

See LinkEvaluator.evaluate_link() for details on which files are accepted.

find_best_candidate (*project_name, specifier=None, hashes=None*)

Find matches for the given project and specifier.

Parameters specifier – An optional object implementing *filter* (e.g. *packaging.specifiers.SpecifierSet*) to filter applicable versions.

Returns A *BestCandidateResult* instance.

find_links

find_requirement (*req, upgrade*)

Try to find a Link matching req

Expects req, an InstallRequirement and upgrade, a boolean Returns a InstallationCandidate if found, Raises DistributionNotFound or BestVersionAlreadyInstalled otherwise

get_install_candidate (*link_evaluator, link*)

If the link is a candidate for install, convert it to an InstallationCandidate and return it. Otherwise, return None.

index_urls

make_candidate_evaluator (*project_name, specifier=None, hashes=None*)

Create a CandidateEvaluator object to use.

make_link_evaluator (*project_name*)

prefer_binary

process_project_url (*project_url, link_evaluator*)

search_scope

set_allow_all_prereleases ()

set_prefer_binary ()

target_python

trusted_hosts

class pip_shims.shims.CandidateEvaluator (*project_name, supported_tags, specifier, prefer_binary=False, allow_all_prereleases=False, hashes=None*)

Bases: *object*

Responsible for filtering and sorting candidates for installation based on what tags are valid.

Parameters `supported_tags` – The PEP 425 tags supported by the target Python in order of preference (most preferred first).

`_sort_key` (*candidate*)

Function to pass as the *key* argument to a call to `sorted()` to sort `InstallationCandidates` by preference.

Returns a tuple such that tuples sorting as greater using Python’s default comparison operator are more preferred.

The preference is as follows:

First and foremost, candidates with allowed (matching) hashes are always preferred over candidates without matching hashes. This is because e.g. if the only candidate with an allowed hash is yanked, we still want to use that candidate.

Second, excepting hash considerations, candidates that have been yanked (in the sense of PEP 592) are always less preferred than candidates that haven’t been yanked. Then:

If not finding wheels, they are sorted by version only. If finding wheels, then the sort order is by version, then:

1. existing installs
2. wheels ordered via `Wheel.support_index_min(self._supported_tags)`
3. source archives

If `prefer_binary` was set, then all wheels are sorted above sources.

Note: it was considered to embed this logic into the `Link` comparison operators, but then different sdist links with the same version, would have to be considered equal

`compute_best_candidate` (*candidates*)

Compute and return a `BestCandidateResult` instance.

classmethod `create` (*project_name*, *target_python=None*, *prefer_binary=False*, *allow_all_prereleases=False*, *specifier=None*, *hashes=None*)

Create a `CandidateEvaluator` object.

Parameters

- **target_python** – The target Python interpreter to use when checking compatibility. If `None` (the default), a `TargetPython` object will be constructed from the running Python.
- **specifier** – An optional object implementing *filter* (e.g. `packaging.specifiers.SpecifierSet`) to filter applicable versions.
- **hashes** – An optional collection of allowed hashes.

`get_applicable_candidates` (*candidates*)

Return the applicable candidates from a list of candidates.

`sort_best_candidate` (*candidates*)

Return the best candidate per the instance’s sort order, or `None` if no candidate is acceptable.

class `pip_shims.shims.CandidatePreferences` (*prefer_binary=False*, *allow_all_prereleases=False*)

Bases: `object`

Encapsulates some of the preferences for filtering and sorting `InstallationCandidate` objects.

Parameters `allow_all_prereleases` – Whether to allow all pre-releases.

class `pip_shims.shims.LinkCollector` (*session, search_scope*)

Bases: `object`

Responsible for collecting Link objects from all configured locations, making network requests as needed.

The class's main method is its `collect_links()` method.

collect_links (*project_name*)

Find all available links for the given project name.

Returns All the Link objects (unfiltered), as a `CollectedLinks` object.

classmethod **create** (*session, options, suppress_no_index=False*)

Parameters

- **session** – The Session to use to make requests.
- **suppress_no_index** – Whether to ignore the `--no-index` option when constructing the `SearchScope` object.

fetch_page (*location*)

Fetch an HTML page containing package links.

find_links

class `pip_shims.shims.LinkEvaluator` (*project_name, canonical_name, formats, target_python, allow_yanked, ignore_requires_python=None*)

Bases: `object`

Responsible for evaluating links for a particular project.

Parameters

- **project_name** – The user supplied package name.
- **canonical_name** – The canonical package name.
- **formats** – The formats allowed for this package. Should be a set with ‘binary’ or ‘source’ or both in it.
- **target_python** – The target Python interpreter to use when evaluating link compatibility. This is used, for example, to check wheel compatibility, as well as when checking the Python version, e.g. the Python version embedded in a link filename (or egg fragment) and against an HTML link’s optional PEP 503 “data-requires-python” attribute.
- **allow_yanked** – Whether files marked as yanked (in the sense of PEP 592) are permitted to be candidates for install.
- **ignore_requires_python** – Whether to ignore incompatible PEP 503 “data-requires-python” values in HTML links. Defaults to False.

`_py_version_re = re.compile('-py([123]\\.|[0-9]?)$')`

evaluate_link (*link*)

Determine whether a link is a candidate for installation.

Returns A tuple (`is_candidate, result`), where *result* is (1) a version string if *is_candidate* is True, and (2) if *is_candidate* is False, an optional string to log the reason the link fails to qualify.

class `pip_shims.shims.TargetPython` (*platform=None, py_version_info=None, abi=None, implementation=None*)

Bases: `object`

Encapsulates the properties of a Python interpreter one is targeting for a package install, download, etc.

Parameters

- **platform** – A string or None. If None, searches for packages that are supported by the current system. Otherwise, will find packages that can be built on the platform passed in. These packages will only be downloaded for distribution: they will not be built locally.
- **py_version_info** – An optional tuple of ints representing the Python version information to use (e.g. `sys.version_info[:3]`). This can have length 1, 2, or 3 when provided.
- **abi** – A string or None. This is passed to `compatibility_tags.py`'s `get_supported()` function as is.
- **implementation** – A string or None. This is passed to `compatibility_tags.py`'s `get_supported()` function as is.

`_given_py_version_info`

`_valid_tags`

`abi`

`format_given()`

Format the given, non-None attributes for display.

`get_tags()`

Return the supported PEP 425 tags to check wheel candidates against.

The tags are returned in order of preference (most preferred first).

`implementation`

`platform`

`py_version`

`py_version_info`

`class pip_shims.shims.SearchScope` (*find_links, index_urls*)

Bases: `object`

Encapsulates the locations that pip is configured to search.

`classmethod create` (*find_links, index_urls*)

Create a `SearchScope` object after normalizing the *find_links*.

`find_links`

`get_formatted_locations()`

`get_index_urls_locations` (*project_name*)

Returns the locations found via `self.index_urls`

Checks the `url_name` on the main (first in the list) index and use this `url_name` to produce all locations

`index_urls`

`class pip_shims.shims.SelectionPreferences` (*allow_yanked, allow_all_prereleases=False, format_control=None, prefer_binary=False, ignore_requires_python=None*)

Bases: `object`

Encapsulates the candidate selection preferences for downloading and installing files.

Create a `SelectionPreferences` object.

Parameters

- **`allow_yanked`** – Whether files marked as yanked (in the sense of PEP 592) are permitted to be candidates for install.

- **format_control** – A FormatControl object or None. Used to control the selection of source packages / binary packages when consulting the index and links.
- **prefer_binary** – Whether to prefer an old, but valid, binary dist over a new source dist.
- **ignore_requires_python** – Whether to ignore incompatible “Requires-Python” values in links. Defaults to False.

allow_all_prereleases

allow_yanked

format_control

ignore_requires_python

prefer_binary

`pip_shims.shims.parse_requirements` (*filename, session, finder=None, comes_from=None, options=None, constraint=False*)

Parse a requirements file and yield ParsedRequirement instances.

Parameters

- **filename** – Path or url of requirements file.
- **session** – PipSession instance.
- **finder** – Instance of pip.index.PackageFinder.
- **comes_from** – Origin description of requirements.
- **options** – cli options.
- **constraint** – If true, parsing a constraint file rather than requirements file.

`pip_shims.shims.path_to_url` (*path*)

Convert a path to a file: URL. The path will be made absolute and have quoted path parts.

exception `pip_shims.shims.PipError`

Bases: `Exception`

Base pip exception

args

with_traceback ()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class `pip_shims.shims.RequirementPreparer` (*build_dir, download_dir, src_dir, wheel_download_dir, build_isolation, req_tracker, downloader, finder, require_hashes, use_user_site*)

Bases: `object`

Prepares a Requirement

`_download_should_save`

`_ensure_link_req_src_dir` (*req, download_dir, parallel_builds*)

Ensure source_dir of a linked InstallRequirement.

`_get_linked_req_hashes` (*req*)

`_log_preparing_link` (*req*)

Log the way the link prepared.

prepare_editable_requirement (*req*)

Prepare an editable requirement

prepare_installed_requirement (*req, skip_reason*)

Prepare an already-installed requirement

prepare_linked_requirement (*req, parallel_builds=False*)

Prepare a requirement to be obtained from req.link.

class pip_shims.shims.**RequirementSet** (*check_supported_wheels=True*)

Bases: `object`

Create a RequirementSet.

add_named_requirement (*install_req*)

add_requirement (*install_req, parent_req_name=None, extras_requested=None*)

Add install_req as a requirement to install.

Parameters

- **parent_req_name** – The name of the requirement that needed this added. The name is used because when multiple unnamed requirements resolve to the same name, we could otherwise end up with dependency links that point outside the Requirements set. `parent_req` must already be added. Note that `None` implies that this is a user supplied requirement, vs an inferred one.
- **extras_requested** – an iterable of extras used to evaluate the environment markers.

Returns Additional requirements to scan. That is either [] if the requirement is not applicable, or [install_req] if the requirement is applicable and has just been added.

add_unnamed_requirement (*install_req*)

all_requirements

get_requirement (*name*)

has_requirement (*name*)

class pip_shims.shims.**RequirementTracker** (*root*)

Bases: `object`

_entry_path (*link*)

add (*req*)

Add an InstallRequirement to build tracking.

cleanup ()

remove (*req*)

Remove an InstallRequirement from build tracking.

track (*req*)

class pip_shims.shims.**TempDirectory** (*path=None, delete=<pip._internal.utils.temp_dir._Default object>, kind='temp', globally_managed=False*)

Bases: `object`

Helper class that owns and cleans up a temporary directory.

This class can be used as a context manager or as an OO representation of a temporary directory.

Attributes:

path Location to the created temporary directory

delete Whether the directory should be deleted when exiting (when used as a contextmanager)

Methods:

cleanup() Deletes the temporary directory

When used as a context manager, if the delete attribute is True, on exiting the context the temporary directory is deleted.

_create (*kind*)

Create a temporary directory and store its path in self.path

cleanup ()

Remove the temporary directory created and reset state

path

`pip_shims.shims.global_tempdir_manager()`

`pip_shims.shims.shim_unpack` (*, *unpack_fn*=<pip_shims.models.ShimmedPathCollection object>, *download_dir*, *tempdir_manager_provider*=<pip_shims.models.ShimmedPathCollection object>, *ireq*=None, *link*=None, *location*=None, *hashes*=None, *progress_bar*='off', *only_download*=None, *downloader_provider*=<pip_shims.models.ShimmedPathCollection object>, *session*=None)

Accepts all parameters that have been valid to pass to `pip._internal.download.unpack_url()` and selects or drops parameters as needed before invoking the provided callable.

Parameters

- **unpack_fn** (*Callable*) – A callable or shim referring to the pip implementation
- **download_dir** (*str*) – The directory to download the file to
- **tempdir_manager_provider** (*TShimmedFunc*) – A callable or shim referring to *global_tempdir_manager* function from pip or a shimmed no-op context manager

:param Optional[InstallRequirement] ireq: an Install Requirement instance, defaults to None

:param Optional[Link] link: A Link instance, defaults to None.

Parameters

- **location** (*Optional[str]*) – A location or source directory if the target is a VCS url, defaults to None.
- **hashes** (*Optional[Any]*) – A Hashes instance, defaults to None
- **progress_bar** (*str*) – Indicates progress par usage during download, defatuls to off.
- **only_download** (*Optional[bool]*) – Whether to skip install, defaults to None.
- **downloader_provider** (*Optional[ShimmedPathCollection]*) – A downloader class to instantiate, if applicable.
- **session** (*Optional[Session]*) – A PipSession instance, defaults to None.

Returns The result of unpacking the url.

Return type None

`pip_shims.shims.get_requirement_tracker()`

```
class pip_shims.shims.Resolver (preparer, finder, wheel_cache, make_install_req, use_user_site,
                                ignore_dependencies, ignore_installed, ignore_requires_python,
                                force_reinstall, upgrade_strategy, py_version_info=None)
```

Bases: pip._internal.resolution.base.BaseResolver

Resolves which packages need to be installed/uninstalled to perform the requested operation without breaking the requirements of any package.

```
_allowed_strategies = {'eager', 'only-if-needed', 'to-satisfy-only'}
```

```
_check_skip_installed (req_to_install)
```

Check if req_to_install should be skipped.

This will check if the req is installed, and whether we should upgrade or reinstall it, taking into account all the relevant user options.

After calling this req_to_install will only have satisfied_by set to None if the req_to_install is to be upgraded/reinstalled etc. Any other value will be a dist recording the current thing installed that satisfies the requirement.

Note that for vcs urls and the like we can't assess skipping in this routine - we simply identify that we need to pull the thing down, then later on it is pulled down and introspected to assess upgrade/ reinstalls etc.

Returns A text reason for why it was skipped, or None.

```
_find_requirement_link (req)
```

```
_get_abstract_dist_for (req)
```

Takes a InstallRequirement and returns a single AbstractDist representing a prepared variant of the same.

```
_is_upgrade_allowed (req)
```

```
_populate_link (req)
```

Ensure that if a link can be found for this, that it is found.

Note that req.link may still be None - if the requirement is already installed and not needed to be upgraded based on the return value of _is_upgrade_allowed().

If preparer.require_hashes is True, don't use the wheel cache, because cached wheels, always built locally, have different hashes than the files downloaded from the index server and thus throw false hash mismatches. Furthermore, cached wheels at present have undeterministic contents due to file modification times.

```
_resolve_one (requirement_set, req_to_install)
```

Prepare a single requirements file.

Returns A list of additional InstallRequirements to also install.

```
_set_req_to_reinstall (req)
```

Set a requirement to be installed.

```
get_installation_order (req_set)
```

Create the installation order.

The installation order is topological - requirements are installed before the requiring thing. We break cycles at an arbitrary point, and make no other guarantees.

```
resolve (root_reqs, check_supported_wheels)
```

Resolve what operations need to be done

As a side-effect of this method, the packages (and their dependencies) are downloaded, unpacked and prepared for installation. This preparation is done by pip.operations.prepare.

Once PyPI has static dependency metadata available, it would be possible to move the preparation to become a step separated from dependency resolution.

```
class pip_shims.shims.SafeFileCache(directory)
```

```
Bases: pip._vendor.cachecontrol.cache.BaseCache
```

A file based cache which is safe to use even when the target directory may not be accessible or writable.

```
_get_cache_path(name)
```

```
close()
```

```
delete(key)
```

```
get(key)
```

```
set(key, value)
```

```
class pip_shims.shims.UninstallPathSet(dist)
```

```
Bases: object
```

A set of file paths to be removed in the uninstallation of a requirement.

```
_allowed_to_proceed(verbose)
```

Display which files would be deleted and prompt for confirmation

```
_permitted(path)
```

Return True if the given path is one we are permitted to remove/modify, False otherwise.

```
add(path)
```

```
add_pth(pth_file, entry)
```

```
commit()
```

Remove temporary save dir: rollback will no longer be possible.

```
classmethod from_dist(dist)
```

```
remove(auto_confirm=False, verbose=False)
```

Remove paths in `self.paths` with confirmation (unless `auto_confirm` is True).

```
rollback()
```

Rollback the changes previously made by `remove()`.

```
pip_shims.shims.url_to_path(url)
```

Convert a file: URL to a path.

```
class pip_shims.shims.VcsSupport
```

```
Bases: object
```

```
_registry = {'bzzr': <pip._internal.vcs.bazaar.Bazaar object>, 'git': <pip._internal.
```

```
all_schemes
```

```
backends
```

```
dirnames
```

```
get_backend(name)
```

Return a VersionControl object or None.

```
get_backend_for_dir(location)
```

Return a VersionControl object if a repository of that type is found at the given directory.

```
get_backend_for_scheme(scheme)
```

Return a VersionControl object or None.

```
register(cls)
```

```
schemes = ['ssh', 'git', 'hg', 'bzzr', 'sftp', 'svn']
```

`unregister` (*name*)

`class` `pip_shims.shims.Wheel` (*filename*)

Bases: `object`

`get_formatted_file_tags` ()

Return the wheel's tags as a sorted list of strings.

`support_index_min` (*tags*)

Return the lowest index that one of the wheel's file_tag combinations achieves in the given list of supported tags.

For example, if there are 8 supported tags and one of the file tags is first in the list, then return 0.

Parameters *tags* – the PEP 425 tags to check the wheel against, in order with most preferred first.

Raises `ValueError` – If none of the wheel's file tags match one of the supported tags.

`supported` (*tags*)

Return whether the wheel is compatible with one of the given tags.

Parameters *tags* – the PEP 425 tags to check the wheel against.

`wheel_file_re` = `re.compile ('^ (?P<namever> (?P<name> .+?) - (?P<ver> .+?)) \n (- (?P<build> \\\`

`class` `pip_shims.shims.WheelCache` (*cache_dir, format_control*)

Bases: `pip._internal.cache.Cache`

Wraps `EphemWheelCache` and `SimpleWheelCache` into a single `Cache`

This `Cache` allows for gracefully degradation, using the ephem wheel cache when a certain link is not found in the simple wheel cache first.

`__get_cache_path_parts` (*link*)

Get parts of part that must be `os.path`.joined with `cache_dir`

`__get_cache_path_parts_legacy` (*link*)

Get parts of part that must be `os.path`.joined with `cache_dir`

Legacy cache key (pip < 20) for compatibility with older caches.

`__get_candidates` (*link, canonical_package_name*)

`get` (*link, package_name, supported_tags*)

Returns a link to a cached item if it exists, otherwise returns the passed link.

`get_cache_entry` (*link, package_name, supported_tags*)

Returns a `CacheEntry` with a link to a cached item if it exists or `None`. The cache entry indicates if the item was found in the persistent or ephemeral cache.

`get_ephem_path_for_link` (*link*)

`get_path_for_link` (*link*)

Return a directory to store cached items in for link.

`get_path_for_link_legacy` (*link*)

`pip_shims.shims.build` (*requirements, wheel_cache, build_options, global_options*)

Build wheels.

Returns The list of `InstallRequirement` that succeeded to build and the list of `InstallRequirement` that failed to build.

`pip_shims.shims.build_one` (*req, output_dir, build_options, global_options*)

Build one wheel.

Returns The filename of the built wheel, or None if the build failed.

`pip_shims.shims.build_one_inside_env(req, output_dir, build_options, global_options)`

class `pip_shims.shims.AbstractDistribution` (*req*)

Bases: `object`

A base class for handling installable artifacts.

The requirements for anything installable are as follows:

- we must be able to determine the requirement name (or we can't correctly handle the non-upgrade case).
- for packages with setup requirements, we must also be able to determine their requirements without installing additional packages (for the same reason as run-time dependencies)
- we must be able to create a Distribution object exposing the above metadata.

`_abc_impl = <_abc_data object>`

`get_pkg_resources_distribution()`

`prepare_distribution_metadata(finder, build_isolation)`

class `pip_shims.shims.InstalledDistribution` (*req*)

Bases: `pip._internal.distributions.base.AbstractDistribution`

Represents an installed package.

This does not need any preparation as the required information has already been computed.

`_abc_impl = <_abc_data object>`

`get_pkg_resources_distribution()`

`prepare_distribution_metadata(finder, build_isolation)`

class `pip_shims.shims.SourceDistribution` (*req*)

Bases: `pip._internal.distributions.base.AbstractDistribution`

Represents a source distribution.

The preparation step for these needs metadata for the packages to be generated, either using PEP 517 or using the legacy `setup.py egg_info`.

`_abc_impl = <_abc_data object>`

`_setup_isolation(finder)`

`get_pkg_resources_distribution()`

`prepare_distribution_metadata(finder, build_isolation)`

class `pip_shims.shims.WheelDistribution` (*req*)

Bases: `pip._internal.distributions.base.AbstractDistribution`

Represents a wheel distribution.

This does not need any preparation as wheels can be directly unpacked.

`_abc_impl = <_abc_data object>`

`get_pkg_resources_distribution()`

Loads the metadata from the wheel file into memory and returns a Distribution that uses it, not relying on the wheel file or requirement.

`prepare_distribution_metadata(finder, build_isolation)`

```
pip_shims.shims.wheel_cache(cache_dir=None, format_control=None, *,
                             wheel_cache_provider=<pip_shims.models.ShimmedPathCollection
                             object>, format_control_provider=<pip_shims.models.ShimmedPathCollection
                             object>, tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection
                             object>)
```

```
pip_shims.shims.get_package_finder(install_cmd=None, options=None, session=None,
                                    platform=None, python_versions=None,
                                    abi=None, implementation=None, tar-
                                    get_python=None, ignore_requires_python=None,
                                    *, target_python_builder=<class
                                    'pip._internal.models.target_python.TargetPython'>, in-
                                    stall_cmd_provider=<pip_shims.models.ShimmedPathCollection
                                    object>)
```

Shim for compatibility to generate package finders.

Build and return a `PackageFinder` instance using the `InstallCommand` helper method to construct the finder, shimmed with backports as needed for compatibility.

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_cmd** – A `InstallCommand` instance which is used to generate the finder.
- **options** (*optparse.Values*) – An optional `optparse.Values` instance generated by calling `install_cmd.parser.parse_args()` typically.
- **session** – An optional session instance, can be created by the `install_cmd`.
- **platform** (*Optional[str]*) – An optional platform string, e.g. `linux_x86_64`
- **python_versions** (*Optional[Tuple[str, ...]]*) – A tuple of 2-digit strings representing python versions, e.g. (“27”, “35”, “36”, “37”...)
- **abi** (*Optional[str]*) – The target abi to support, e.g. “cp38”
- **implementation** (*Optional[str]*) – An optional implementation string for limiting searches to a specific implementation, e.g. “cp” or “py”
- **target_python** – A `TargetPython` instance (will be translated to alternate arguments if necessary on incompatible pip versions).
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore `requires_python` on resulting candidates, only valid after pip version 19.3.1
- **target_python_builder** – A ‘`TargetPython`’ builder (e.g. the class itself, uninstantiated)

Returns A `pip._internal.index.package_finder.PackageFinder` instance

Return type `pip._internal.index.package_finder.PackageFinder`

Example

```
>>> from pip_shims.shims import InstallCommand, get_package_finder
>>> install_cmd = InstallCommand()
>>> finder = get_package_finder(
...     install_cmd, python_versions=("27", "35", "36", "37", "38"),
...     implementation="
cp"
... )
```

(continues on next page)

(continued from previous page)

```

>>> candidates = finder.find_all_candidates("requests")
>>> requests_222 = next(iter(c for c in candidates if c.version.public == "2.22.0
↳"))
>>> requests_222
<InstallationCandidate('requests', <Version('2.22.0')>, <Link https://files.
↳pythonhos
ted.org/packages/51/bd/
↳23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/r
equests-2.22.0-py2.py3-none-any.whl
↳#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9
a590f48c010551dc6c4b31 (from https://pypi.org/simple/requests/) (requires-python:>
↳=2.
7, !=3.0.*, !=3.1.*, !=3.2.*, !=3.3.*, !=3.4.*)>>>

```

`pip_shims.shims.make_preparer` (*, *preparer_fn*=<pip_shims.models.ShimmedPathCollection object>, *req_tracker_fn*=<pip_shims.models.ShimmedPathCollection object>, *build_dir*=None, *src_dir*=None, *download_dir*=None, *wheel_download_dir*=None, *progress_bar*='off', *build_isolation*=False, *session*=None, *finder*=None, *options*=None, *require_hashes*=None, *use_user_site*=None, *req_tracker*=None, *install_cmd_provider*=<pip_shims.models.ShimmedPathCollection object>, *downloader_provider*=<pip_shims.models.ShimmedPathCollection object>, *install_cmd*=None, *finder_provider*=<pip_shims.models.ShimmedPathCollection object>)

Creates a requirement preparer for preparing pip requirements.

Provides a compatibility shim that accepts all previously valid arguments and discards any that are no longer used.

Raises

- **TypeError** – No requirement tracker provided and one cannot be generated
- **TypeError** – No valid sessions provided and one cannot be generated
- **TypeError** – No valid finders provided and one cannot be generated

Parameters

- **preparer_fn** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **req_tracker_fn** (*Optional[TShimmedFunc]*) – Callable or shim for generating requirement trackers, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **src_dir** (*Optional[str]*) – Directory to find or extract source files, defaults to None
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **progress_bar** (*str*) – Whether to display a progress bar, defaults to off
- **build_isolation** (*bool*) – Whether to build requirements in isolation, defaults to False

- **session** (*Optional*[*TSession*]) – Existing session to use for getting requirements, defaults to None
- **finder** (*Optional*[*TFinder*]) – The package finder to use during resolution, defaults to None
- **options** (*Optional*[*Values*]) – Pip options to use if needed, defaults to None
- **require_hashes** (*Optional*[*bool*]) – Whether to require hashes for preparation
- **use_user_site** (*Optional*[*bool*]) – Whether to use the user site directory for preparing requirements
- **TShimmedFunc]] req_tracker** (*Optional*[*Union*[*TReqTracker*,)]) – The requirement tracker to use for building packages, defaults to None
- **downloader_provider** (*Optional*[*TShimmedFunc*]) – A downloader provider
- **install_cmd** (*Optional*[*TCommandInstance*]) – The install command used to create the finder, session, and options if needed, defaults to None
- **finder_provider** (*Optional*[*TShimmedFunc*]) – A package finder provider

Yield A new requirement preparer instance

Return type `ContextManager[RequirementPreparer]`

Example

```
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_tracker
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> with make_preparer(
...     options=pip_options, finder=finder, session=session, install_cmd=ic
... ) as preparer:
...     print(preparer)
<pip._internal.operations.prepare.RequirementPreparer object at 0x7f8a2734be80>
```

`pip_shims.shims.get_resolver`(**, resolver_fn=<pip_shims.models.ShimmedPathCollection object>, install_req_provider=<pip_shims.models.ShimmedPathCollection object>, format_control_provider=<pip_shims.models.ShimmedPathCollection object>, wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>, finder=None, upgrade_strategy='to-satisfy-only', force_reinstall=None, ignore_dependencies=None, ignore_requires_python=None, ignore_installed=True, use_user_site=False, isolated=None, wheel_cache=None, preparer=None, session=None, options=None, make_install_req=None, install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>, install_cmd=None, use_pep517=True)*

A resolver creation compatibility shim for generating a resolver.

Consumes any argument that was previously used to instantiate a resolver, discards anything that is no longer valid.

Note: This is only valid for **pip >= 10.0.0**

Raises

- **ValueError** – A session is required but not provided and one cannot be created
- **ValueError** – A finder is required but not provided and one cannot be created
- **ValueError** – An install requirement provider is required and has not been provided

Parameters

- **resolver_fn** (*TShimmedFunc*) – The resolver function used to create new resolver instances.
- **install_req_provider** (*TShimmedFunc*) – The provider function to use to generate install requirements if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a format_control instance if needed.
- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to None
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to None
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to None
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to True
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to False
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to None
- **wheel_cache** (*Optional[TWheelCache]*) – The wheel cache to use, defaults to None
- **preparer** (*Optional[TPreparer]*) – The requirement preparer to use, defaults to None
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **make_install_req** (*Optional[functools.partial]*) – The partial function to pass in to the resolver for actually generating install requirements, if necessary
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None.

- `use_pep517` (*bool*) – Whether to use the pep517 build process.

Returns A new resolver instance.

Return type Resolver

Example

```
>>> import os
>>> from tempdir import TemporaryDirectory
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_
↳ tracker,
...     get_resolver, InstallRequirement, RequirementSet
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> wheel_cache = WheelCache(USER_CACHE_DIR, FormatControl(None, None))
>>> with TemporaryDirectory() as temp_base:
...     reqset = RequirementSet()
...     ireq = InstallRequirement.from_line("requests")
...     ireq.is_direct = True
...     build_dir = os.path.join(temp_base, "build")
...     src_dir = os.path.join(temp_base, "src")
...     ireq.build_location(build_dir)
...     with make_preparer(
...         options=pip_options, finder=finder, session=session,
...         build_dir=build_dir, install_cmd=install_cmd,
...     ) as preparer:
...         resolver = get_resolver(
...             finder=finder, ignore_dependencies=False, ignore_requires_
↳ python=True,
...             preparer=preparer, session=session, options=pip_options,
...             install_cmd=install_cmd, wheel_cache=wheel_cache,
...         )
...         resolver.require_hashes = False
...         reqset.add_requirement(ireq)
...         results = resolver.resolve(reqset)
...         #reqset.cleanup_files()
...         for result_req in reqset.requirements:
...             print(result_req)
requests
chardet
certifi
urllib3
idna
```

```

pip_shims.shims.get_requirement_set (install_command=None, *,
                                     req_set_provider=<pip_shims.models.ShimmedPathCollection
                                     object>, build_dir=None, src_dir=None, down-
                                     load_dir=None, wheel_download_dir=None, ses-
                                     sion=None, wheel_cache=None, upgrade=False,
                                     upgrade_strategy=None, ignore_installed=False,
                                     ignore_dependencies=False, force_reinstall=False,
                                     use_user_site=False, isolated=False, ig-
                                     nore_requires_python=False, require_hashes=None,
                                     cache_dir=None, options=None, in-
                                     stall_cmd_provider=<pip_shims.models.ShimmedPathCollection
                                     object>, wheel_cache_provider=<pip_shims.models.ShimmedPathCollection
                                     object>)

```

Creates a requirement set from the supplied parameters.

Not all parameters are passed through for all pip versions, but any invalid parameters will be ignored if they are not needed to generate a requirement set on the current pip version.

:param *ShimmedPathCollection* wheel_cache_provider: A context manager provider which resolves to a *WheelCache* instance

Parameters install_command – A *InstallCommand* instance which is used to generate the finder.

:param *ShimmedPathCollection* req_set_provider: A provider to build requirement set instances.

Parameters

- **build_dir** (*str*) – The directory to build requirements in. Removed in pip 10, defaults to None
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*str*) – The directory to download requirement artifacts to. Removed in pip 10, defaults to None
- **wheel_download_dir** (*str*) – The directory to download wheels to. Removed in pip 10, defaults to None

:param *Session* session: The pip session to use. Removed in pip 10, defaults to None

Parameters

- **wheel_cache** (*WheelCache*) – The pip *WheelCache* instance to use for caching wheels. Removed in pip 10, defaults to None
- **upgrade** (*bool*) – Whether to try to upgrade existing requirements. Removed in pip 10, defaults to False.
- **upgrade_strategy** (*str*) – The upgrade strategy to use, e.g. “only-if-needed”. Removed in pip 10, defaults to None.
- **ignore_installed** (*bool*) – Whether to ignore installed packages when resolving. Removed in pip 10, defaults to False.
- **ignore_dependencies** (*bool*) – Whether to ignore dependencies of requirements when resolving. Removed in pip 10, defaults to False.

- **force_reinstall** (*bool*) – Whether to force reinstall of packages when resolving. Removed in pip 10, defaults to False.
- **use_user_site** (*bool*) – Whether to use user site packages when resolving. Removed in pip 10, defaults to False.
- **isolated** (*bool*) – Whether to resolve in isolation. Removed in pip 10, defaults to False.
- **ignore_requires_python** (*bool*) – Removed in pip 10, defaults to False.
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **options** (*Values*) – An *Values* instance from an install cmd
- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.

Returns A new requirement set instance

Return type RequirementSet

```
pip_shims.shims.resolve(ireq, *, reqset_provider=<pip_shims.models.ShimmedPathCollection object>, req_tracker_provider=<pip_shims.models.ShimmedPathCollection object>, install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>, install_command=None, finder_provider=<pip_shims.models.ShimmedPathCollection object>, resolver_provider=<pip_shims.models.ShimmedPathCollection object>, wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>, format_control_provider=<pip_shims.models.ShimmedPathCollection object>, make_preparer_provider=<pip_shims.models.ShimmedPathCollection object>, tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection object>, options=None, session=None, resolver=None, finder=None, upgrade_strategy='to-satisfy-only', force_reinstall=None, ignore_dependencies=None, ignore_requires_python=None, ignore_installed=True, use_user_site=False, isolated=None, build_dir=None, source_dir=None, download_dir=None, cache_dir=None, wheel_download_dir=None, wheel_cache=None, require_hashes=None, check_supported_wheels=True)
```

Resolves the provided **InstallRequirement**, returning a dictionary.

Maps a dictionary of names to corresponding *InstallRequirement* values.

:param InstallRequirement ireq: An *InstallRequirement* to initiate the resolution process

:param ShimmedPathCollection reqset_provider: A **provider** to build requirement set instances.

:param ShimmedPathCollection req_tracker_provider: A **provider** to build requirement tracker instances

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_command** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None.

:param ShimmedPathCollection finder_provider: A **provider** to package finder instances.

:param ShimmedPathCollection resolver_provider: A **provider** to build resolver instances

Parameters

- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a `format_control` instance if needed.
- **make_preparer_provider** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **tempdir_manager_provider** (*Optional[TShimmedFunc]*) – Shim for generating tempdir manager for pip temporary directories
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to `None`
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to `None`

:param Resolver resolver: A pre-existing resolver instance to use for resolution

Parameters

- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to `None`.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package reinstallation during resolution, defaults to `None`
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to `None`
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to `None`
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to `True`
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to `False`
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to `None`
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to `None`
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to `None`
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to `None`
- **cache_dir** (*str*) – The cache directory to use for caching artifacts during resolution
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to `None`
- **wheel_cache** (*Optional[TWheelCache]*) – The wheel cache to use, defaults to `None`
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to `False`.

- **check_supported_wheels** (*bool*) – Whether to check support of wheels before including them in resolution.

Returns A dictionary mapping requirements to corresponding :class:`~pip._internal.req.req_install.InstallRequirement`s

Return type InstallRequirement

Example

```
>>> from pip_shims.shims import resolve, InstallRequirement
>>> ireq = InstallRequirement.from_line("requests>=2.20")
>>> results = resolve(ireq)
>>> for k, v in results.items():
...     print("{0}: {1!r}".format(k, v))
requests: <InstallRequirement object: requests>=2.20 from https://files.
↳pythonhosted.
org/packages/51/bd/23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/
↳reque
sts-2.22.0-py2.py3-none-any.whl
↳#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9a590
f48c010551dc6c4b31 editable=False>
idna: <InstallRequirement object: idna<2.9,>=2.5 from https://files.pythonhosted.
↳org/
packages/14/2c/cd551d81dbe15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-
↳2.8-
py2.py3-none-any.whl
↳#sha256=ea8b7f6188e6fa117537c3df7da9fc686d485087abf6ac197f9c46432
f7e4a3c (from requests>=2.20) editable=False>
urllib3: <InstallRequirement object: urllib3!=1.25.0,!<1.25.1,<1.26,>=1.21.1 from
↳htt
↳htt
ps://files.pythonhosted.org/packages/b4/40/
↳a9837291310eelccc242ceb6ebfd9eb21539649f19
3a7c8c86ba15b98539/urllib3-1.25.7-py2.py3-none-any.whl
↳#sha256=a8a318824cc77d1fd4b2bec
2ded92646630d7fe8619497b142c84a9e6f5a7293 (from requests>=2.20) editable=False>
chardet: <InstallRequirement object: chardet<3.1.0,>=3.0.2 from https://files.
↳pythonh
osted.org/packages/bc/a9/
↳01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8
/chardet-3.0.4-py2.py3-none-any.whl
↳#sha256=fc323ffcaeaed0e0a02bf4d117757b98aed530d9ed
4531e3e15460124c106691 (from requests>=2.20) editable=False>
certifi: <InstallRequirement object: certifi>=2017.4.17 from https://files.
↳pythonhost
ed.org/packages/18/b0/
↳8146a4f8dd402f60744fa380bc73ca47303ccccf8b9190fd16a827281eac2/ce
rtifi-2019.9.11-py2.py3-none-any.whl
↳#sha256=fd7c7c74727ddcf00e9acd26bba8da604ffec95bf
1c2144e67aff7a8b50e6cef (from requests>=2.20) editable=False>
```

```

pip_shims.shims.build_wheel(req=None, reqset=None, output_dir=None, preparer=None,
                               wheel_cache=None, build_options=None, global_options=None,
                               check_binary_allowed=None, no_clean=False, session=None,
                               finder=None, install_command=None, req_tracker=None,
                               build_dir=None, src_dir=None, download_dir=None,
                               wheel_download_dir=None, cache_dir=None, use_user_site=False,
                               use_pep517=None, *, format_control_provider=<pip_shims.models.ShimmedPathCollection
                               object>, wheel_cache_provider=<pip_shims.models.ShimmedPathCollection
                               object>, preparer_provider=<pip_shims.models.ShimmedPathCollection
                               object>, wheel_builder_provider=<pip_shims.models.ShimmedPathCollection
                               object>, build_one_provider=<pip_shims.models.ShimmedPathCollection
                               object>, build_one_inside_env_provider=<pip_shims.models.ShimmedPathCollection
                               object>, build_many_provider=<pip_shims.models.ShimmedPathCollection
                               object>, install_command_provider=<pip_shims.models.ShimmedPathCollection
                               object>, finder_provider=None, reqset_provider=<pip_shims.models.ShimmedPathCollection
                               object>)

```

Build a wheel or a set of wheels

Raises `TypeError` – Raised when no requirements are provided

Parameters

- **req** (*Optional*[*TInstallRequirement*]) – An *InstallRequirement* to build
- **reqset** (*Optional*[*TReqSet*]) – A *RequirementSet* instance (*pip*<10) or an iterable of *InstallRequirement* instances (*pip*>=10) to build
- **output_dir** (*Optional*[*str*]) – Target output directory, only useful when building one wheel using *pip*>=20.0
- **preparer** (*Optional*[*TPreparer*]) – A preparer instance, defaults to None
- **wheel_cache** (*Optional*[*TWheelCache*]) – A wheel cache instance, defaults to None
- **build_options** (*Optional*[*List*[*str*]]) – A list of build options to pass in
- **global_options** (*Optional*[*List*[*str*]]) – A list of global options to pass in
- **bool]] check_binary_allowed** (*Optional*[*Callable*[*TInstallRequirement*,]]) – A callable to check whether we are allowed to build and cache wheels for an ireq
- **no_clean** (*bool*) – Whether to avoid cleaning up wheels
- **session** (*Optional*[*TSession*]) – A *PipSession* instance to pass to create a *finder* if necessary
- **finder** (*Optional*[*TFinder*]) – A *PackageFinder* instance to use for generating a *WheelBuilder* instance on *pip*<20
- **install_command** (*Optional*[*TCommandInstance*]) – The install command used to create the finder, session, and options if needed, defaults to None.
- **req_tracker** (*Optional*[*TReqTracker*]) – An optional requirement tracker instance, if one already exists
- **build_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **src_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **download_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer

- **wheel_download_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **cache_dir** (*Optional[str]*) – Passthrough cache directory for wheel cache options
- **use_user_site** (*bool*) – Whether to use the user site directory when preparing install requirements on *pip<20*
- **use_pep517** (*Optional[bool]*) – When set to *True* or *False*, prefers building with or without pep517 as specified, otherwise uses requirement preference. Only works for single requirements.
- **format_control_provider** (*Optional[TShimmedFunc]*) – A provider for the *FormatControl* class
- **wheel_cache_provider** (*Optional[TShimmedFunc]*) – A provider for the *WheelCache* class
- **preparer_provider** (*Optional[TShimmedFunc]*) – A provider for the *RequirementPreparer* class
- **wheel_builder_provider** (*Optional[TShimmedFunc]*) – A provider for the *WheelBuilder* class, if it exists
- **build_one_provider** (*Optional[TShimmedFunc]*) – A provider for the *_build_one* function, if it exists
- **build_one_inside_env_provider** (*Optional[TShimmedFunc]*) – A provider for the *_build_one_inside_env* function, if it exists
- **build_many_provider** (*Optional[TShimmedFunc]*) – A provider for the *build* function, if it exists
- **install_command_provider** (*Optional[TShimmedFunc]*) – A shim for providing new install command instances
- **finder_provider** (*TShimmedFunc*) – A provider to package finder instances
- **reqset_provider** (*TShimmedFunc*) – A provider for requirement set generation

Returns A tuple of successful and failed install requirements or else a path to a wheel

Return type `Optional[Union[str, Tuple[List[TInstallRequirement], List[TInstallRequirement]]]]`

2.2.5 pip_shims.utils module

Shared utility functions which are not specific to any particular module.

```
class pip_shims.utils.BaseClassMethod(func_base, name, *args, **kwargs)
```

```
    Bases: collections.abc.Callable
```

```
    _abc_impl = <_abc_data object>
```

```
class pip_shims.utils.BaseMethod(func_base, name, *args, **kwargs)
```

```
    Bases: collections.abc.Callable
```

```
    _abc_impl = <_abc_data object>
```

```
pip_shims.utils._parse(version)
```

```
pip_shims.utils.add_mixin_to_class(basecls, mixins)
```

Given a class, adds the provided mixin classes as base classes and gives a new class

Parameters

- **basecls** (*Type*) – An initial class to generate a new class from
- **mixins** (*List [Type]*) – A list of mixins to add as base classes

Returns A new class with the provided mixins as base classes

Return type `Type[basecls, *mixins]`

`pip_shims.utils.apply_alias` (*imported, target, *aliases*)

Given a target with attributes, point non-existent aliases at the first existing one

Parameters

- **Type]** **imported** (*Union [ModuleType,]*) – A Module or Class base
- **target** (*Any*) – The target which is a member of **imported** and will have aliases
- **aliases** (*str*) – A list of aliases, the first found attribute will be the basis for all non-existent names which will be created as pointers

Returns The original target

Return type `Any`

`pip_shims.utils.call_function_with_correct_args` (*fn, **provided_kwargs*)

Determines which arguments from **provided_kwargs** to call **fn** and calls it.

Consumes a list of allowed arguments (e.g. from `getargs()`) and uses it to determine which of the arguments in the provided kwargs should be passed through to the given callable.

Parameters

- **fn** (*Callable*) – A callable which has some dynamic arguments
- **allowed_args** (*List [str]*) – A list of allowed arguments which can be passed to the supplied function

Returns The result of calling the function

Return type `Any`

`pip_shims.utils.ensure_function` (*parent, funcname, func*)

Given a module, a function name, and a function object, attaches the given function to the module and ensures it is named properly according to the provided argument

Parameters

- **parent** (*Any*) – The parent to attach the function to
- **funcname** (*str*) – The name to give the function
- **func** (*Callable*) – The function to rename and attach to **parent**

Returns The function with its name, qualname, etc set to mirror **parent**

Return type `Callable`

`pip_shims.utils.fallback_is_artifact` (*self*)

`pip_shims.utils.fallback_is_file_url` (*link*)

`pip_shims.utils.fallback_is_vcs` (*self*)

`pip_shims.utils.filter_allowed_args` (*fn, **provided_kwargs*)

Given a function and a kwarg mapping, return only those kwargs used in the function.

Parameters

- **fn** (*Callable*) – A function to inspect

- **Any** **kwargs** (*Dict[str, Any]*) – A mapping of kwargs to filter

Returns A new, filtered kwarg mapping

Return type Tuple[List[Any], Dict[str, Any]]

`pip_shims.utils.get_allowed_args(fn_or_class)`

Given a callable or a class, returns the arguments and default kwargs passed in.

Parameters **Type** **fn_or_class** (*Union[Callable, Any]*) – A function, method or class to inspect.

Returns A 2-tuple with a list of arguments and a dictionary of keywords mapped to default values.

Return type Tuple[List[str], Dict[str, Any]]

`pip_shims.utils.get_method_args(target_method)`

Returns the arguments for a callable.

Parameters **target_method** (*Callable*) – A callable to retrieve arguments for

Returns A 2-tuple of the original callable and its resulting arguments

Return type Tuple[Callable, Optional[inspect.Arguments]]

`pip_shims.utils.has_property(target, name)`

`pip_shims.utils.make_classmethod(fn)`

`pip_shims.utils.make_method(fn)`

`pip_shims.utils.memoize(obj)`

`pip_shims.utils.nullcontext(*args, **kwargs)`

`pip_shims.utils.parse_version(version)`

`pip_shims.utils.resolve_possible_shim(target)`

`pip_shims.utils.set_default_kwargs(basecls, method, *args, **default_kwargs)`

`pip_shims.utils.split_package(module, subimport=None)`

Used to determine what target to import.

Either splits off the final segment or uses the provided sub-import to return a 2-tuple of the import path and the target module or sub-path.

Parameters

- **module** (*str*) – A package to import from
- **subimport** (*Optional[str]*) – A class, function, or subpackage to import

Returns A 2-tuple of the corresponding import package and sub-import path

Return type Tuple[str, str]

Example

```
>>> from pip_shims.utils import split_package
>>> split_package("pip._internal.req.req_install", subimport="InstallRequirement")
("pip._internal.req.req_install", "InstallRequirement")
>>> split_package("pip._internal.cli.base_command")
("pip._internal.cli", "base_command")
```

`pip_shims.utils.suppress_setattr(obj, attr, value, filter_none=False)`

Set an attribute, suppressing any exceptions and skipping the attempt on failure.

Parameters

- **obj** (*Any*) – Object to set the attribute on
- **attr** (*str*) – The attribute name to set
- **value** (*Any*) – The value to set the attribute to
- **filter_none** (*bool*) – [description], defaults to False

Returns Nothing**Return type** `None`**Example**

```

>>> class MyClass(object):
...     def __init__(self, name):
...         self.name = name
...         self.parent = None
...     def __repr__(self):
...         return "<{0!r} instance (name={1!r}, parent={2!r})>".format(
...             self.__class__.__name__, self.name, self.parent
...         )
...     def __str__(self):
...         return self.name
>>> me = MyClass("Dan")
>>> dad = MyClass("John")
>>> grandfather = MyClass("Joe")
>>> suppress_setattr(dad, "parent", grandfather)
>>> dad
<'MyClass' instance (name='John', parent=<'MyClass' instance (name='Joe',
↳parent=None
)>)>
>>> suppress_setattr(me, "parent", dad)
>>> me
<'MyClass' instance (name='Dan', parent=<'MyClass' instance (name='John', parent=<
↳'My
Class' instance (name='Joe', parent=None)>)>)>
>>> suppress_setattr(me, "grandparent", grandfather)
>>> me
<'MyClass' instance (name='Dan', parent=<'MyClass' instance (name='John', parent=<
↳'My
Class' instance (name='Joe', parent=None)>)>)>

```


3.1 Bug Fixes

- Avoid overriding slot members when adding new methods to a class. #67
- Call `resolve()` with correct arguments for pip 20.2. #68

4.1 Features

- Added support for `pip==20.1`. - Added support for global temporary directory context management when generating wheel caches using the compatibility module; - Added wheel cache context management which now requires the temporary directory context in some cases; - Improved function argument introspection; - Updated test invocations to reflect shifting parameters. [#65](#)

5.1 Bug Fixes

- Fixed incorrect session creation via `pip_shims.compat.get_session` which inadvertently passed a tuple to pip when building a session instance. #56
- Added `wheel_cache` context manager helper for managing global context when creating wheel `wheel_cache` instances. #58
- **Fixed resolution failures due to `Resolver.resolve` signature updates in pip@master:**
 - Automatically check for and pass `check_supports_wheel` argument to `Resolver.resolve()` when expected
 - Check whether `Resolver.resolve()` expects a `RequirementSet` or `List[InstallRequirement]` and pass the appropriate input #59
- Fixed requirement build failures due to new `autodelete: bool required` argument in `InstallRequirement.ensure_build_location`. #60
- Updated `Resolver` import path to point at new location (`legacy_resolve` -> `resolution.legacy_resolver`). #61
- Fixed `AttributeError` caused by failed `RequirementSet.cleanup()` calls after `Resolver.resolve()` which is no longer valid in `pip>=20.1`. #62

6.1 Features

- Exposed `build`, `build_one`, and `build_one_inside_env` from `wheel_builder` module starting in `pip>=20`. #49
- Added a `build_wheel` shim function which can build either a single `InstallRequirement` or an iterable of `InstallRequirement` instances. #50
- Exposed `global_tempdir_manager` for handling `TempDirectory` instance contexts. #51

6.2 Bug Fixes

- Added `Downloader` class which is now passed to `shim_unpack` implementation. #42
- Updated references to the `Downloader` class to point at `pip._internal.network.download.Downloader` which is where it resides on `pip` master for `pip>19.3.1`. #46
- Added a compatibility shim to provide ongoing access to the `Wheel` class which is removed in `pip>19.3.1`. #47
- Added mapping for `distributions.make_distribution_for_install` to `make_abstract_dist` for `pip>=20.0`. #52

7.1 Features

- Improved documentation and added fundamentally re-architected the library
- Added improved docstrings and example usages
- Included type annotations for many types and shims
- Fully reimplemented critical functionality to abstract logic while improving maintainability and ability to reason about the core operations
- Added numerous helper functions to reduce maintenance burden
- Added fully backward compatible library native shims to call `pip` functions:
 - `populate_options`
 - `get_requirement_set`
 - `get_package_finder`
 - `shim_unpack`
 - `make_preparer`
 - `get_resolver`
 - `resolve`
- Added design drawings
- Implemented `ShimmedPath` and `ShimmedPathCollection` abstractions [#37](#)

8.1 Features

- Added `SessionCommandMixin`, `CandidateEvaluator`, `CandidatePreferences`, `LinkCollector`, `LinkEvaluator`, `TargetPython`, `SearchScope`, and `SelectionPreferences` to exposed classes and `install_req_from_req_string` to exposed functions. #33

8.2 Bug Fixes

- Added override to the `Command` class to automatically fill in default values for `name` and `summary` which are now required in `__init__`. - Added mixin to the `Command` class to continue supporting `_build_session` method. #32
- Shimmed functions for `is_file_url` and `is_archive_file`. #34
- Updated the paths for the following moved items: - `SafeFileCache` -> `network.cache` - `Link` -> `models.link.Link` - `path_to_url` -> `utils.url` - `url_to_path` -> `utils.url` - `SourceDistribution` -> `distributions.source.legacy` #35

9.1 Features

- Added `commands.freeze.DEV_PKGS` and `utils.compat.stdlib_pkgs` shims. #25
- Updated `PackageFinder` test and added `CandidateEvaluator` import starting with `pip>=19.1` for finding prerelease candidates. #27

9.2 Bug Fixes

- Fixed import paths for `VcsSupport` on `pip>19.1.1`. #28

CHAPTER 10

0.3.2 (2018-10-27)

10.1 Features

- Added access to `pip._internal.models.index.PyPI`. #21

11.1 Features

- **Added shims for the following:**
 - `InstallationError`
 - `UninstallationError`
 - `DistributionNotFound`
 - `RequirementsFileParseError`
 - `BestVersionAlreadyInstalled`
 - `BadCommand`
 - `CommandError`
 - `PreviousBuildDirError` #19

12.1 Features

- Added and exposed `FrozenRequirement` for consumption. #17

12.2 Bug Fixes

- Fixed a bug which caused usage of incorrect location for `_strip_extras`. #13
- Fixed a bug which caused `FormatControl` imports to fail in `pip>=18.1`. #15
- Fixed a bug which caused `InstallRequirement.from_line` and `InstallRequirement.from_editable` to fail in `pip>=18.1`. #16

CHAPTER 13

0.2.0 (2018-10-05)

13.1 Features

- Added a shim for `pip._internal.req.req_uninstall.UninstallPathSet`. #10
- Made all module loading lazy by replacing modules dynamically at runtime. #9

14.1 Features

- Added `WheelCache` and `unpack_url` functionality. #4

14.2 Bug Fixes

- Fixed a bug which caused failures in the detection and import on pip version 9 and below when using `modutils`. #5
- Fixed a bug with sort order logic which caused invalid import paths to be prioritized accidentally. #7

15.1 Bug Fixes

- Fixed tests failures for appveyor path comparisons. #2

15.2 Documentation Updates

- Added warning to documentation to discourage use of these shims for accessing the pip API. #1

16.1 Features

- Initial release of pip compatibility shims! #0

<i>pip_shims</i>	This library is a set of compatibility access shims to the <code>pip</code> internal API.
<i>pip_shims.shims</i>	Main module with magic self-replacement mechanisms to handle import speedups.
<i>pip_shims.models</i>	Helper module for shimming functionality across pip versions.
<i>pip_shims.compat</i>	Backports and helper functionality to support using new functionality.
<i>pip_shims.environment</i>	Module with functionality to learn about the environment.
<i>pip_shims.utils</i>	Shared utility functions which are not specific to any particular module.

- *Submodules*

This library is a set of compatibility access shims to the `pip` internal API. It provides compatibility with `pip` versions 8.0 through the current release. The shims are provided using a lazy import strategy by hacking a module by overloading a class instance's `getattr` method. This library exists due to my constant writing of the same set of import shims.

17.1 Submodules

<code>pip_shims.models</code>	Helper module for shimming functionality across <code>pip</code> versions.
<code>pip_shims.compat</code>	Backports and helper functionality to support using new functionality.
<code>pip_shims.utils</code>	Shared utility functions which are not specific to any particular module.
<code>pip_shims.shims</code>	Main module with magic self-replacement mechanisms to handle import speedups.
<code>pip_shims.environment</code>	Module with functionality to learn about the environment.

17.1.1 `pip_shims.models`

Helper module for shimming functionality across `pip` versions.

```
class pip_shims.models.ImportTypes
```

```
    Bases: pip_shims.models.ImportTypes
```

```
    Create new instance of ImportTypes(FUNCTION, CLASS, MODULE, CONTEXTMANAGER)
```

```
    ATTRIBUTE = 5
```

```

CLASS = 1
CONTEXTMANAGER = 3
FUNCTION = 0
METHOD = 4
MODULE = 2

_asdict()
    Return a new OrderedDict which maps field names to their values.

_fields = ('FUNCTION', 'CLASS', 'MODULE', 'CONTEXTMANAGER')

_fields_defaults = {}

classmethod _make(iterable)
    Make a new ImportTypes object from a sequence or iterable

_replace(**kws)
    Return a new ImportTypes object replacing specified fields with new values

count()
    Return number of occurrences of value.

index()
    Return first index of value.

    Raises ValueError if the value is not present.

```

```

pip_shims.models.ImportTypesBase
    alias of pip_shims.models.ImportTypes

```

```

class pip_shims.models.PipVersion(version, round_prereleases_up=True,
                                   base_import_path=None, vendor_import_path='pip._vendor')

```

```

Bases: collections.abc.Sequence

```

```

_abc_impl = <_abc_data object>

```

```

_parse()

```

```

count(value) → integer – return number of occurrences of value

```

```

index(value[, start[, stop]]) → integer – return first index of value.
    Raises ValueError if the value is not present.

```

```

    Supporting start and stop arguments is optional, but recommended.

```

```

is_valid(compared_to)

```

```

version_key

```

```

version_tuple

```

```

class pip_shims.models.PipVersionRange(start, end)

```

```

Bases: collections.abc.Sequence

```

```

_abc_impl = <_abc_data object>

```

```

base_import_paths

```

```

count(value) → integer – return number of occurrences of value

```

index (*value*[, *start*[, *stop*]]) → integer – return first index of value.
 Raises ValueError if the value is not present.

Supporting start and stop arguments is optional, but recommended.

is_valid()

vendor_import_paths

```
class pip_shims.models.ShimmedPath(name, import_target, import_type, ver-  

sion_range, provided_methods=None, pro-  

vided_functions=None, provided_classmethods=None,  

provided_contextmanagers=None, pro-  

vided_mixins=None, default_args=None)
```

Bases: object

_ShimmedPath__modules = {'pip._internal.cache': <module 'pip._internal.cache' from '...'>}

_apply_aliases (*imported*, *target*)

_as_tuple()

_ensure_functions (*provided*)

_ensure_methods (*provided*)

Given a base class, a new name, and any number of functions to attach, turns those functions into class-methods, attaches them, and returns an updated class object.

_import (*prefix*=None)

classmethod _import_module (*module*)

classmethod _parse_provides_dict (*provides*, *prepend_arg_to_callables*=None)

_shim_base (*imported*, *attribute_name*)

_shim_parent (*imported*, *attribute_name*)

_update_default_kwargs (*parent*, *provided*)

alias (*aliases*)

calculated_module_path

is_attribute

is_class

is_contextmanager

is_function

is_method

is_module

is_valid

shim()

shim_attribute (*imported*, *attribute_name*)

shim_class (*imported*, *attribute_name*)

shim_contextmanager (*imported*, *attribute_name*)

shim_function (*imported*, *attribute_name*)

shim_module (*imported*, *attribute_name*)

`shimmed`
`sort_order`
`update_sys_modules` (*imported*)

`class pip_shims.models.ShimmedPathCollection` (*name, import_type, paths=None*)

Bases: `object`

`_ShimmedPathCollection__registry` = {'AbstractDistribution': `<pip_shims.models.Shimmed`

`_get_top_path`()

`_sort_paths`()

`add_mixin` (*mixin*)

`add_path` (*path*)

`alias` (*aliases*)

Takes a list of methods, functions, attributes, etc and ensures they all exist on the object pointing at the same referent.

Parameters `aliases` (*List[str]*) – Names to map to the same functionality if they do not exist.

Returns `None`

Return type `None`

`create_path` (*import_path, version_start, version_end=None*)

`classmethod get_registry`()

`pre_shim` (*fn*)

`provide_function` (*name, fn*)

`provide_method` (*name, fn*)

`register`()

`set_default` (*default*)

`set_default_args` (*callable_name, *args, **kwargs*)

`shim`()

`classmethod traverse` (*shim*)

`pip_shims.models.import_pip`()

`pip_shims.models.lookup_current_pip_version`()

`pip_shims.models.pip_version_lookup` (*version, *args, **kwargs*)

17.1.2 pip_shims.compat

Backports and helper functionality to support using new functionality.

`class pip_shims.compat.CandidateEvaluator` (*project_name, supported_tags, specifier, prefer_binary=False, allow_all_prereleases=False, hashes=None*)

Bases: `object`

`classmethod create` (*project_name, target_python=None, prefer_binary=False, allow_all_prereleases=False, specifier=None, hashes=None*)

```
class pip_shims.compat.CandidatePreferences (prefer_binary=False, al-  

low_all_prereleases=False)
```

Bases: `object`

```
exception pip_shims.compat.InvalidWheelFilename
```

Bases: `Exception`

Wheel Filename is Invalid

args

```
with_traceback ()
```

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

```
class pip_shims.compat.LinkCollector (session=None, search_scope=None)
```

Bases: `object`

```
class pip_shims.compat.LinkEvaluator (allow_yanked, project_name, canonical_name, for-  

mats, target_python, ignore_requires_python=False,  

ignore_compatibility=True)
```

Bases: `object`

```
class pip_shims.compat.SearchScope (find_links=None, index_urls=None)
```

Bases: `object`

```
classmethod create (find_links=None, index_urls=None)
```

```
class pip_shims.compat.SelectionPreferences (allow_yanked=True, al-  

low_all_prereleases=False, for-  

mat_control=None, prefer_binary=False,  

ignore_requires_python=False)
```

Bases: `object`

```
class pip_shims.compat.TargetPython (platform=None, py_version_info=None, abi=None, im-  

plementation=None)
```

Bases: `object`

```
fallback_get_tags = <pip_shims.models.ShimmedPathCollection object>
```

```
get_tags ()
```

```
class pip_shims.compat.Wheel (filename)
```

Bases: `object`

```
get_formatted_file_tags ()
```

Return the wheel's tags as a sorted list of strings.

```
support_index_min (tags)
```

Return the lowest index that one of the wheel's file_tag combinations achieves in the given list of supported tags.

For example, if there are 8 supported tags and one of the file tags is first in the list, then return 0.

Parameters tags – the PEP 425 tags to check the wheel against, in order with most preferred first.

Raises ValueError – If none of the wheel's file tags match one of the supported tags.

```
supported (tags)
```

Return whether the wheel is compatible with one of the given tags.

Parameters tags – the PEP 425 tags to check the wheel against.

```
wheel_file_re = re.compile ('^(?P<namever> (?P<name> .+?) - (?P<ver> .*?)) \n ( (- (?P<build> \\\
```

`pip_shims.compat._ensure_finder` (*finder=None, finder_provider=None, install_cmd=None, options=None, session=None*)

`pip_shims.compat._ensure_wheel_cache` (*wheel_cache=None, wheel_cache_provider=None, format_control=None, format_control_provider=None, options=None, cache_dir=None*)

`pip_shims.compat.build_wheel` (*req=None, reqset=None, output_dir=None, preparer=None, wheel_cache=None, build_options=None, global_options=None, check_binary_allowed=None, no_clean=False, session=None, finder=None, install_command=None, req_tracker=None, build_dir=None, src_dir=None, download_dir=None, wheel_download_dir=None, cache_dir=None, use_user_site=False, use_pep517=None, format_control_provider=None, wheel_cache_provider=None, preparer_provider=None, wheel_builder_provider=None, build_one_provider=None, build_one_inside_env_provider=None, build_many_provider=None, install_command_provider=None, finder_provider=None, reqset_provider=None*)

Build a wheel or a set of wheels

Raises `TypeError` – Raised when no requirements are provided

Parameters

- **req** (*Optional[InstallRequirement]*) – An *InstallRequirement* to build
- **reqset** (*Optional[ReqSet]*) – A *RequirementSet* instance (*pip<10*) or an iterable of *InstallRequirement* instances (*pip>=10*) to build
- **output_dir** (*Optional[str]*) – Target output directory, only useful when building one wheel using *pip>=20.0*
- **preparer** (*Optional[Preparer]*) – A preparer instance, defaults to *None*
- **wheel_cache** (*Optional[WheelCache]*) – A wheel cache instance, defaults to *None*
- **build_options** (*Optional[List[str]]*) – A list of build options to pass in
- **global_options** (*Optional[List[str]]*) – A list of global options to pass in
- **bool[] check_binary_allowed** (*Optional[Callable[InstallRequirement, bool]]*) – A callable to check whether we are allowed to build and cache wheels for an *ireq*
- **no_clean** (*bool*) – Whether to avoid cleaning up wheels
- **session** (*Optional[Session]*) – A *PipSession* instance to pass to create a *finder* if necessary
- **finder** (*Optional[Finder]*) – A *PackageFinder* instance to use for generating a *WheelBuilder* instance on *pip<20*
- **install_command** (*Optional[CommandInstance]*) – The install command used to create the *finder*, *session*, and *options* if needed, defaults to *None*.
- **req_tracker** (*Optional[ReqTracker]*) – An optional requirement tracker instance, if one already exists
- **build_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **src_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **download_dir** (*Optional[str]*) – Passthrough parameter for building preparer

- **wheel_download_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **cache_dir** (*Optional[str]*) – Passthrough cache directory for wheel cache options
- **use_user_site** (*bool*) – Whether to use the user site directory when preparing install requirements on *pip<20*
- **use_pep517** (*Optional[bool]*) – When set to *True* or *False*, prefers building with or without pep517 as specified, otherwise uses requirement preference. Only works for single requirements.
- **format_control_provider** (*Optional[TShimmedFunc]*) – A provider for the *FormatControl* class
- **wheel_cache_provider** (*Optional[TShimmedFunc]*) – A provider for the *WheelCache* class
- **preparer_provider** (*Optional[TShimmedFunc]*) – A provider for the *RequirementPreparer* class
- **wheel_builder_provider** (*Optional[TShimmedFunc]*) – A provider for the *WheelBuilder* class, if it exists
- **build_one_provider** (*Optional[TShimmedFunc]*) – A provider for the *_build_one* function, if it exists
- **build_one_inside_env_provider** (*Optional[TShimmedFunc]*) – A provider for the *_build_one_inside_env* function, if it exists
- **build_many_provider** (*Optional[TShimmedFunc]*) – A provider for the *build* function, if it exists
- **install_command_provider** (*Optional[TShimmedFunc]*) – A shim for providing new install command instances
- **finder_provider** (*TShimmedFunc*) – A provider to package finder instances
- **reqset_provider** (*TShimmedFunc*) – A provider for requirement set generation

Returns A tuple of successful and failed install requirements or else a path to a wheel

Return type `Optional[Union[str, Tuple[List[TInstallRequirement], List[TInstallRequirement]]]]`

`pip_shims.compat.ensure_resolution_dirs` (***kwargs*)

Ensures that the proper directories are scaffolded and present in the provided kwargs for performing dependency resolution via pip.

Returns A new kwargs dictionary with scaffolded directories for **build_dir**, **src_dir**, **download_dir**, and **wheel_download_dir** added to the key value pairs.

Return type `Dict[str, Any]`

`pip_shims.compat.get_ireq_output_path` (*wheel_cache, ireq*)

`pip_shims.compat.get_package_finder` (*install_cmd=None, options=None, session=None, platform=None, python_versions=None, abi=None, implementation=None, target_python=None, ignore_requires_python=None, target_python_builder=None, install_cmd_provider=None*)

Shim for compatibility to generate package finders.

Build and return a `PackageFinder` instance using the `InstallCommand` helper method to construct the finder, shimmed with backports as needed for compatibility.

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_cmd** – A `InstallCommand` instance which is used to generate the finder.
- **options** (*optparse.Values*) – An optional `optparse.Values` instance generated by calling `install_cmd.parser.parse_args()` typically.
- **session** – An optional session instance, can be created by the `install_cmd`.
- **platform** (*Optional[str]*) – An optional platform string, e.g. `linux_x86_64`
- **python_versions** (*Optional[Tuple[str, ...]]*) – A tuple of 2-digit strings representing python versions, e.g. `("27", "35", "36", "37"…)`
- **abi** (*Optional[str]*) – The target abi to support, e.g. `"cp38"`
- **implementation** (*Optional[str]*) – An optional implementation string for limiting searches to a specific implementation, e.g. `"cp"` or `"py"`
- **target_python** – A `TargetPython` instance (will be translated to alternate arguments if necessary on incompatible pip versions).
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore `requires_python` on resulting candidates, only valid after pip version 19.3.1
- **target_python_builder** – A ‘`TargetPython`’ builder (e.g. the class itself, uninstantiated)

Returns A `pip._internal.index.package_finder.PackageFinder` instance

Return type `pip._internal.index.package_finder.PackageFinder`

Example

```
>>> from pip_shims.shims import InstallCommand, get_package_finder
>>> install_cmd = InstallCommand()
>>> finder = get_package_finder(
...     install_cmd, python_versions=("27", "35", "36", "37", "38"),
...     implementation="
cp"
... )
>>> candidates = finder.find_all_candidates("requests")
>>> requests_222 = next(iter(c for c in candidates if c.version.public == "2.22.0
... ))
>>> requests_222
<InstallationCandidate('requests', <Version('2.22.0')>, <Link https://files.
pythonhos
ted.org/packages/51/bd/
23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/r
equests-2.22.0-py2.py3-none-any.whl
#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9
a590f48c010551dc6c4b31 (from https://pypi.org/simple/requests/) (requires-python:>
=2.
7, !=3.0.*, !=3.1.*, !=3.2.*, !=3.3.*, !=3.4.*)>>>
```



```

pip_shims.compat.get_requirement_set (install_command=None, req_set_provider=None,
                                     build_dir=None, src_dir=None, download_dir=None,
                                     wheel_download_dir=None, session=None,
                                     wheel_cache=None, upgrade=False, upgrade_strategy=None,
                                     ignore_installed=False, ignore_dependencies=False,
                                     force_reinstall=False, use_user_site=False,
                                     isolated=False, ignore_requires_python=False,
                                     require_hashes=None, cache_dir=None,
                                     options=None, install_cmd_provider=None,
                                     wheel_cache_provider=None)

```

Creates a requirement set from the supplied parameters.

Not all parameters are passed through for all pip versions, but any invalid parameters will be ignored if they are not needed to generate a requirement set on the current pip version.

:param *ShimmedPathCollection* wheel_cache_provider: A context manager provider which resolves to a *WheelCache* instance

Parameters install_command – A *InstallCommand* instance which is used to generate the finder.

:param *ShimmedPathCollection* req_set_provider: A provider to build requirement set instances.

Parameters

- **build_dir** (*str*) – The directory to build requirements in. Removed in pip 10, defaults to None
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*str*) – The directory to download requirement artifacts to. Removed in pip 10, defaults to None
- **wheel_download_dir** (*str*) – The directory to download wheels to. Removed in pip 10, defaults to None

:param *Session* session: The pip session to use. Removed in pip 10, defaults to None

Parameters

- **wheel_cache** (*WheelCache*) – The pip *WheelCache* instance to use for caching wheels. Removed in pip 10, defaults to None
- **upgrade** (*bool*) – Whether to try to upgrade existing requirements. Removed in pip 10, defaults to False.
- **upgrade_strategy** (*str*) – The upgrade strategy to use, e.g. “only-if-needed”. Removed in pip 10, defaults to None.
- **ignore_installed** (*bool*) – Whether to ignore installed packages when resolving. Removed in pip 10, defaults to False.
- **ignore_dependencies** (*bool*) – Whether to ignore dependencies of requirements when resolving. Removed in pip 10, defaults to False.
- **force_reinstall** (*bool*) – Whether to force reinstall of packages when resolving. Removed in pip 10, defaults to False.

- **use_user_site** (*bool*) – Whether to use user site packages when resolving. Removed in pip 10, defaults to False.
- **isolated** (*bool*) – Whether to resolve in isolation. Removed in pip 10, defaults to False.
- **ignore_requires_python** (*bool*) – Removed in pip 10, defaults to False.
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **options** (*Values*) – An *Values* instance from an install cmd
- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.

Returns A new requirement set instance

Return type RequirementSet

`pip_shims.compat.get_requirement_tracker` (*req_tracker_creator=None*)

`pip_shims.compat.get_resolver` (*resolver_fn, install_req_provider=None, format_control_provider=None, wheel_cache_provider=None, finder=None, upgrade_strategy='to-satisfy-only', force_reinstall=None, ignore_dependencies=None, ignore_requires_python=None, ignore_installed=True, use_user_site=False, isolated=None, wheel_cache=None, preparer=None, session=None, options=None, make_install_req=None, install_cmd_provider=None, install_cmd=None, use_pep517=True*)

A resolver creation compatibility shim for generating a resolver.

Consumes any argument that was previously used to instantiate a resolver, discards anything that is no longer valid.

Note: This is only valid for **pip >= 10.0.0**

Raises

- **ValueError** – A session is required but not provided and one cannot be created
- **ValueError** – A finder is required but not provided and one cannot be created
- **ValueError** – An install requirement provider is required and has not been provided

Parameters

- **resolver_fn** (*TShimmedFunc*) – The resolver function used to create new resolver instances.
- **install_req_provider** (*TShimmedFunc*) – The provider function to use to generate install requirements if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a *format_control* instance if needed.
- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.

- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to `None`
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to `None`
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to `None`
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to `True`
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to `False`
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to `None`
- **wheel_cache** (*Optional[WheelCache]*) – The wheel cache to use, defaults to `None`
- **preparer** (*Optional[TPreparer]*) – The requirement preparer to use, defaults to `None`
- **session** (*Optional[Session]*) – Existing session to use for getting requirements, defaults to `None`
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to `None`
- **make_install_req** (*Optional[functools.partial]*) – The partial function to pass in to the resolver for actually generating install requirements, if necessary
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to `None`.
- **use_pep517** (*bool*) – Whether to use the pep517 build process.

Returns A new resolver instance.

Return type Resolver

Example

```
>>> import os
>>> from tempdir import TemporaryDirectory
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_
↳ tracker,
...     get_resolver, InstallRequirement, RequirementSet
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> wheel_cache = WheelCache(USER_CACHE_DIR, FormatControl(None, None))
>>> with TemporaryDirectory() as temp_base:
...     reqset = RequirementSet()
...     ireq = InstallRequirement.from_line("requests")
```

(continues on next page)

(continued from previous page)

```

...     ireq.is_direct = True
...     build_dir = os.path.join(temp_base, "build")
...     src_dir = os.path.join(temp_base, "src")
...     ireq.build_location(build_dir)
...     with make_preparer(
...         options=pip_options, finder=finder, session=session,
...         build_dir=build_dir, install_cmd=install_cmd,
...     ) as preparer:
...         resolver = get_resolver(
...             finder=finder, ignore_dependencies=False, ignore_requires_
↪python=True,
...             preparer=preparer, session=session, options=pip_options,
...             install_cmd=install_cmd, wheel_cache=wheel_cache,
...         )
...         resolver.require_hashes = False
...         reqset.add_requirement(ireq)
...         results = resolver.resolve(reqset)
...         #reqset.cleanup_files()
...         for result_req in reqset.requirements:
...             print(result_req)
requests
charset
certifi
urllib3
idna

```

`pip_shims.compat.get_session` (*install_cmd_provider=None, install_cmd=None, options=None*)

`pip_shims.compat.make_preparer` (*preparer_fn, req_tracker_fn=None, build_dir=None, src_dir=None, download_dir=None, wheel_download_dir=None, progress_bar='off', build_isolation=False, session=None, finder=None, options=None, require_hashes=None, use_user_site=None, req_tracker=None, install_cmd_provider=None, downloader_provider=None, install_cmd=None, finder_provider=None*)

Creates a requirement preparer for preparing pip requirements.

Provides a compatibility shim that accepts all previously valid arguments and discards any that are no longer used.

Raises

- **`TypeError`** – No requirement tracker provided and one cannot be generated
- **`TypeError`** – No valid sessions provided and one cannot be generated
- **`TypeError`** – No valid finders provided and one cannot be generated

Parameters

- **`preparer_fn`** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **`req_tracker_fn`** (*Optional[TShimmedFunc]*) – Callable or shim for generating requirement trackers, defaults to None
- **`build_dir`** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **`src_dir`** (*Optional[str]*) – Directory to find or extract source files, defaults to None

- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **progress_bar** (*str*) – Whether to display a progress bar, defaults to off
- **build_isolation** (*bool*) – Whether to build requirements in isolation, defaults to False
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **require_hashes** (*Optional[bool]*) – Whether to require hashes for preparation
- **use_user_site** (*Optional[bool]*) – Whether to use the user site directory for preparing requirements
- **TShimmedFunc]] req_tracker** (*Optional[Union[TReqTracker,])*) – The requirement tracker to use for building packages, defaults to None
- **downloader_provider** (*Optional[TShimmedFunc]*) – A downloader provider
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None
- **finder_provider** (*Optional[TShimmedFunc]*) – A package finder provider

Yield A new requirement preparer instance

Return type ContextManager[RequirementPreparer]

Example

```
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_tracker
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> with make_preparer(
...     options=pip_options, finder=finder, session=session, install_cmd=ic
... ) as preparer:
...     print(preparer)
<pip._internal.operations.prepare.RequirementPreparer object at 0x7f8a2734be80>
```

pip_shims.compat.**partial_command** (*shimmed_path, cmd_mapping=None*)

Maps a default set of arguments across all members of a *ShimmedPath* instance, specifically for Command instances which need *summary* and *name* arguments.

:param ShimmedPath shimmed_path: A ShimmedCollection instance

Parameters cmd_mapping (*Any*) – A reference to use for mapping against, e.g. an import that depends on pip also

Returns A dictionary mapping new arguments to their default values

Return type Dict[str, str]

pip_shims.compat.**populate_options** (*install_command=None, options=None, **kwargs*)

pip_shims.compat.**resolve** (*ireq, reqset_provider=None, req_tracker_provider=None, install_cmd_provider=None, install_command=None, finder_provider=None, resolver_provider=None, wheel_cache_provider=None, format_control_provider=None, make_preparer_provider=None, tempdir_manager_provider=None, options=None, session=None, resolver=None, finder=None, upgrade_strategy='to-satisfy-only', force_reinstall=None, ignore_dependencies=None, ignore_requires_python=None, ignore_installed=True, use_user_site=False, isolated=None, build_dir=None, source_dir=None, download_dir=None, cache_dir=None, wheel_download_dir=None, wheel_cache=None, require_hashes=None, check_supported_wheels=True*)

Resolves the provided **InstallRequirement**, returning a dictionary.

Maps a dictionary of names to corresponding `InstallRequirement` values.

:param InstallRequirement ireq: An `InstallRequirement` to initiate the resolution process

:param ShimmablePathCollection reqset_provider: A provider to build requirement set instances.

:param ShimmablePathCollection req_tracker_provider: A provider to build requirement tracker instances

Parameters

- **install_cmd_provider** (*ShimmablePathCollection*) – A shim for providing new install command instances.
- **install_command** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None.

:param ShimmablePathCollection finder_provider: A provider to package finder instances.

:param ShimmablePathCollection resolver_provider: A provider to build resolver instances

Parameters

- **wheel_cache_provider** (*TShimmableFunc*) – The provider function to use to generate a wheel cache if needed.
- **format_control_provider** (*TShimmableFunc*) – The provider function to use to generate a `format_control` instance if needed.
- **make_preparer_provider** (*TShimmableFunc*) – Callable or shim for generating preparers.
- **tempdir_manager_provider** (*Optional[TShimmableFunc]*) – Shim for generating tempdir manager for pip temporary directories
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None

:param Resolver resolver: A pre-existing resolver instance to use for resolution

Parameters

- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to None
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to None
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to None
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to True
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to False
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **cache_dir** (*str*) – The cache directory to use for caching artifacts during resolution
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **wheel_cache** (*Optional[TWheelCache]*) – The wheel cache to use, defaults to None
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **check_supported_wheels** (*bool*) – Whether to check support of wheels before including them in resolution.

Returns A dictionary mapping requirements to corresponding `:class:`~pip._internal.req.req_install.InstallRequirement`'s`

Return type `InstallRequirement`

Example

```
>>> from pip_shims.shims import resolve, InstallRequirement
>>> ireq = InstallRequirement.from_line("requests>=2.20")
>>> results = resolve(ireq)
>>> for k, v in results.items():
...     print("{0}: {1!r}".format(k, v))
requests: <InstallRequirement object: requests>=2.20 from https://files.
pythonhosted.org/packages/51/bd/23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/
requests
```

(continues on next page)

(continued from previous page)

```

sts-2.22.0-py2.py3-none-any.whl
↳#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9a590
f48c010551dc6c4b31 editable=False>
idna: <InstallRequirement object: idna<2.9,>=2.5 from https://files.pythonhosted.
↳org/
packages/14/2c/cd551d81dbe15200belcf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-
↳2.8-
py2.py3-none-any.whl
↳#sha256=ea8b7f6188e6fa117537c3df7da9fc686d485087abf6ac197f9c46432
f7e4a3c (from requests>=2.20) editable=False>
urllib3: <InstallRequirement object: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 from
↳htt
ps://files.pythonhosted.org/packages/b4/40/
↳a9837291310eelccc242ceb6ebfd9eb21539649f19
3a7c8c86ba15b98539/urllib3-1.25.7-py2.py3-none-any.whl
↳#sha256=a8a318824cc77d1fd4b2bec
2ded92646630d7fe8619497b142c84a9e6f5a7293 (from requests>=2.20) editable=False>
chardet: <InstallRequirement object: chardet<3.1.0,>=3.0.2 from https://files.
↳pythonh
osted.org/packages/bc/a9/
↳01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8
/chardet-3.0.4-py2.py3-none-any.whl
↳#sha256=fc323ffcaead0e0a02bf4d117757b98aed530d9ed
4531e3e15460124c106691 (from requests>=2.20) editable=False>
certifi: <InstallRequirement object: certifi>=2017.4.17 from https://files.
↳pythonhost
ed.org/packages/18/b0/
↳8146a4f8dd402f60744fa380bc73ca47303ccccf8b9190fd16a827281eac2/ce
rtifi-2019.9.11-py2.py3-none-any.whl
↳#sha256=fd7c7c74727ddcf00e9acd26bba8da604ffec95bf
1c2144e67aff7a8b50e6cef (from requests>=2.20) editable=False>

```

`pip_shims.compat.resolve_possible_shim(target)`

`pip_shims.compat.shim_unpack(unpack_fn, download_dir, tempdir_manager_provider, ireq=None, link=None, location=None, hashes=None, progress_bar='off', only_download=None, downloader_provider=None, session=None)`

Accepts all parameters that have been valid to pass to `pip._internal.download.unpack_url()` and selects or drops parameters as needed before invoking the provided callable.

Parameters

- **unpack_fn** (*Callable*) – A callable or shim referring to the pip implementation
- **download_dir** (*str*) – The directory to download the file to
- **tempdir_manager_provider** (*TShimmedFunc*) – A callable or shim referring to `global_tempdir_manager` function from pip or a shimmed no-op context manager

:param Optional[InstallRequirement] ireq: an Install Requirement instance, defaults to None

:param Optional[Link] link: A Link instance, defaults to None.

Parameters

- **location** (*Optional[str]*) – A location or source directory if the target is a VCS url, defaults to None.

- **hashes** (*Optional[Any]*) – A Hashes instance, defaults to None
- **progress_bar** (*str*) – Indicates progress bar usage during download, defaults to off.
- **only_download** (*Optional[bool]*) – Whether to skip install, defaults to None.
- **downloader_provider** (*Optional[ShimmedPathCollection]*) – A downloader class to instantiate, if applicable.
- **session** (*Optional[Session]*) – A PipSession instance, defaults to None.

Returns The result of unpacking the url.

Return type None

`pip_shims.compat.temp_environ()`

Allow the ability to set os.environ temporarily

`pip_shims.compat.wheel_cache(cache_dir=None, format_control=None, wheel_cache_provider=None, format_control_provider=None, tempdir_manager_provider=None)`

17.1.3 pip_shims.utils

Shared utility functions which are not specific to any particular module.

class `pip_shims.utils.BaseClassMethod` (*func_base, name, *args, **kwargs*)

Bases: `collections.abc.Callable`

`_abc_impl = <_abc_data object>`

class `pip_shims.utils.BaseMethod` (*func_base, name, *args, **kwargs*)

Bases: `collections.abc.Callable`

`_abc_impl = <_abc_data object>`

`pip_shims.utils._parse(version)`

`pip_shims.utils.add_mixin_to_class(basecls, mixins)`

Given a class, adds the provided mixin classes as base classes and gives a new class

Parameters

- **basecls** (*Type*) – An initial class to generate a new class from
- **mixins** (*List[Type]*) – A list of mixins to add as base classes

Returns A new class with the provided mixins as base classes

Return type `Type[basecls, *mixins]`

`pip_shims.utils.apply_alias(imported, target, *aliases)`

Given a target with attributes, point non-existent aliases at the first existing one

Parameters

- **Type]** **imported** (*Union[ModuleType,]*) – A Module or Class base
- **target** (*Any*) – The target which is a member of **imported** and will have aliases
- **aliases** (*str*) – A list of aliases, the first found attribute will be the basis for all non-existent names which will be created as pointers

Returns The original target

Return type Any

`pip_shims.utils.call_function_with_correct_args` (*fn*, ***provided_kwargs*)

Determines which arguments from **provided_kwargs** to call **fn** and calls it.

Consumes a list of allowed arguments (e.g. from `getargs()`) and uses it to determine which of the arguments in the provided kwarg should be passed through to the given callable.

Parameters

- **fn** (*Callable*) – A callable which has some dynamic arguments
- **allowed_args** (*List[str]*) – A list of allowed arguments which can be passed to the supplied function

Returns The result of calling the function

Return type Any

`pip_shims.utils.ensure_function` (*parent*, *funcname*, *func*)

Given a module, a function name, and a function object, attaches the given function to the module and ensures it is named properly according to the provided argument

Parameters

- **parent** (*Any*) – The parent to attach the function to
- **funcname** (*str*) – The name to give the function
- **func** (*Callable*) – The function to rename and attach to **parent**

Returns The function with its name, qualname, etc set to mirror **parent**

Return type Callable

`pip_shims.utils.fallback_is_artifact` (*self*)

`pip_shims.utils.fallback_is_file_url` (*link*)

`pip_shims.utils.fallback_is_vcs` (*self*)

`pip_shims.utils.filter_allowed_args` (*fn*, ***provided_kwargs*)

Given a function and a kwarg mapping, return only those kwargs used in the function.

Parameters

- **fn** (*Callable*) – A function to inspect
- **Any** **kwargs** (*Dict[str, Any]*) – A mapping of kwargs to filter

Returns A new, filtered kwarg mapping

Return type Tuple[List[Any], Dict[str, Any]]

`pip_shims.utils.get_allowed_args` (*fn_or_class*)

Given a callable or a class, returns the arguments and default kwargs passed in.

Parameters **Type** **fn_or_class** (*Union[Callable, Any]*) – A function, method or class to inspect.

Returns A 2-tuple with a list of arguments and a dictionary of keywords mapped to default values.

Return type Tuple[List[str], Dict[str, Any]]

`pip_shims.utils.get_method_args` (*target_method*)

Returns the arguments for a callable.

Parameters **target_method** (*Callable*) – A callable to retrieve arguments for

Returns A 2-tuple of the original callable and its resulting arguments

Return type Tuple[Callable, Optional[inspect.Arguments]]

`pip_shims.utils.has_property` (*target*, *name*)

`pip_shims.utils.make_classmethod` (*fn*)

`pip_shims.utils.make_method` (*fn*)

`pip_shims.utils.memoize` (*obj*)

`pip_shims.utils.nullcontext` (**args*, ***kwargs*)

`pip_shims.utils.parse_version` (*version*)

`pip_shims.utils.resolve_possible_shim` (*target*)

`pip_shims.utils.set_default_kwargs` (*basecls*, *method*, **args*, ***default_kwargs*)

`pip_shims.utils.split_package` (*module*, *subimport=None*)

Used to determine what target to import.

Either splits off the final segment or uses the provided sub-import to return a 2-tuple of the import path and the target module or sub-path.

Parameters

- **module** (*str*) – A package to import from
- **subimport** (*Optional[str]*) – A class, function, or subpackage to import

Returns A 2-tuple of the corresponding import package and sub-import path

Return type Tuple[str, str]

Example

```
>>> from pip_shims.utils import split_package
>>> split_package("pip._internal.req.req_install", subimport="InstallRequirement")
("pip._internal.req.req_install", "InstallRequirement")
>>> split_package("pip._internal.cli.base_command")
("pip._internal.cli", "base_command")
```

`pip_shims.utils.suppress_setattr` (*obj*, *attr*, *value*, *filter_none=False*)

Set an attribute, suppressing any exceptions and skipping the attempt on failure.

Parameters

- **obj** (*Any*) – Object to set the attribute on
- **attr** (*str*) – The attribute name to set
- **value** (*Any*) – The value to set the attribute to
- **filter_none** (*bool*) – [description], defaults to False

Returns Nothing

Return type None

Example

```
>>> class MyClass(object):
...     def __init__(self, name):
...         self.name = name
...         self.parent = None
...     def __repr__(self):
```

(continues on next page)

(continued from previous page)

```

...         return "<{0!r}> instance (name={1!r}, parent={2!r})>".format (
...             self.__class__.__name__, self.name, self.parent
...         )
...     def __str__(self):
...         return self.name
>>> me = MyClass("Dan")
>>> dad = MyClass("John")
>>> grandfather = MyClass("Joe")
>>> suppress_setattr(dad, "parent", grandfather)
>>> dad
<'MyClass' instance (name='John', parent=<'MyClass' instance (name='Joe',
↳parent=None
)>>>
>>> suppress_setattr(me, "parent", dad)
>>> me
<'MyClass' instance (name='Dan', parent=<'MyClass' instance (name='John', parent=<
↳'My
Class' instance (name='Joe', parent=None)>>>)>>
>>> suppress_setattr(me, "grandparent", grandfather)
>>> me
<'MyClass' instance (name='Dan', parent=<'MyClass' instance (name='John', parent=<
↳'My
Class' instance (name='Joe', parent=None)>>>)>>

```

17.1.4 pip_shims.shims

Main module with magic self-replacement mechanisms to handle import speedups.

`pip_shims.shims._strip_extras` (*path*)

class `pip_shims.shims.SessionCommandMixin`

Bases: `pip._internal.cli.command_context.CommandContextMixIn`

A class mixin for command classes needing `_build_session()`.

`_build_session` (*options*, *retries=None*, *timeout=None*)

classmethod `_get_index_urls` (*options*)

Return a list of index urls from user-provided options.

`enter_context` (*context_provider*)

`get_default_session` (*options*)

Get a default-managed session.

`main_context` ()

class `pip_shims.shims.Command` (*name='Default pip command.'*, *summary='PipCommand'*, *isolated='Default pip command.'*)

Bases: `pip._internal.cli.base_command.Command`, `pip._internal.cli.req_command.SessionCommandMixin`

`_build_session` (*options*, *retries=None*, *timeout=None*)

classmethod `_get_index_urls` (*options*)

Return a list of index urls from user-provided options.

`_main` (*args*)

`add_options` ()

enter_context (*context_provider*)

get_default_session (*options*)

Get a default-managed session.

handle_pip_version_check (*options*)

This is a no-op so that commands by default do not do the pip version check.

ignore_require_venv = **False**

main (*args*)

main_context ()

parse_args (*args*)

run (*options, args*)

usage = **None**

class pip_shims.shims.**ConfigOptionParser** (**args, **kwargs*)

Bases: pip._internal.cli.parser.CustomOptionParser

Custom option parser which updates its defaults by checking the configuration files and environmental variables

_add_help_option ()

_add_version_option ()

_check_conflict (*option*)

_create_option_list ()

_create_option_mappings ()

_get_all_options ()

_get_args (*args*)

_get_ordered_configuration_items ()

_init_parsing_state ()

_match_long_opt (*opt : string*) → string

Determine which long option string 'opt' matches, ie. which one it is an unambiguous abbreviation for. Raises `BadOptionError` if 'opt' doesn't unambiguously match any long option string.

_populate_option_list (*option_list, add_help=True*)

_process_args (*largs, rargs, values*)

_process_args(*largs* [[string],] *rargs* : [string], *values* : Values)

Process command-line arguments and populate 'values', consuming options and arguments from 'rargs'. If 'allow_interspersed_args' is false, stop at the first non-option argument. If true, accumulate any interspersed non-option arguments in 'largs'.

_process_long_opt (*rargs, values*)

_process_short_opts (*rargs, values*)

_share_option_mappings (*parser*)

_update_defaults (*defaults*)

Updates the given defaults with values from the config files and the environ. Does a little special handling for certain types of options (lists).

add_option (*Option*)
 add_option(opt_str, ..., kwarg=val, ...)

add_option_group (**args, **kwargs*)

add_options (*option_list*)

check_default (*option, key, val*)

check_values (*values : Values, args : [string]*)
 -> (*values : Values, args : [string]*)

Check that the supplied option values and leftover arguments are valid. Returns the option values and leftover arguments (possibly adjusted, possibly completely new – whatever you like). Default implementation just returns the passed-in values; subclasses may override as desired.

destroy ()

Declare that you are done with this OptionParser. This cleans up reference cycles so the OptionParser (and all objects referenced by it) can be garbage-collected promptly. After calling destroy(), the OptionParser is unusable.

disable_interspersed_args ()

Set parsing to stop on the first non-option. Use this if you have a command processor which runs another command that has options of its own and you want to make sure these options don't get confused.

enable_interspersed_args ()

Set parsing to not stop on the first non-option, allowing interspersing switches with command arguments. This is the default behavior. See also disable_interspersed_args() and the class documentation description of the attribute allow_interspersed_args.

error (*msg : string*)

Print a usage message incorporating 'msg' to stderr and exit. If you override this in a subclass, it should not return – it should either exit or raise an exception.

exit (*status=0, msg=None*)

expand_prog_name (*s*)

format_description (*formatter*)

format_epilog (*formatter*)

format_help (*formatter=None*)

format_option_help (*formatter=None*)

get_default_values ()

Overriding to make updating the defaults after instantiation of the option parser possible, _update_defaults() does the dirty work.

get_description ()

get_option (*opt_str*)

get_option_group (*opt_str*)

get_prog_name ()

get_usage ()

get_version ()

has_option (*opt_str*)

insert_option_group (*idx, *args, **kwargs*)

Insert an OptionGroup at a given position.

option_list_all

Get a list of all options, including those in option groups.

parse_args (*args=None, values=None*)

parse_args(args [[string] = sys.argv[1:], values : Values = None)

-> (values : Values, args : [string])

Parse the command-line options found in 'args' (default: sys.argv[1:]). Any errors result in a call to 'error()', which by default prints the usage message to stderr and calls sys.exit() with an error message. On success returns a pair (values, args) where 'values' is a Values instance (with all your option values) and 'args' is the list of arguments left over after parsing options.

print_help (*file : file = stdout*)

Print an extended help message, listing all options and any help text provided with them, to 'file' (default stdout).

print_usage (*file : file = stdout*)

Print the usage message for the current program (self.usage) to 'file' (default stdout). Any occurrence of the string "%prog" in self.usage is replaced with the name of the current program (basename of sys.argv[0]). Does nothing if self.usage is empty or not defined.

print_version (*file : file = stdout*)

Print the version message for this program (self.version) to 'file' (default stdout). As with print_usage(), any occurrence of "%prog" in self.version is replaced by the current program's name. Does nothing if self.version is empty or undefined.

remove_option (*opt_str*)

set_conflict_handler (*handler*)

set_default (*dest, value*)

set_defaults (***kwargs*)

set_description (*description*)

set_process_default_values (*process*)

set_usage (*usage*)

standard_option_list = []

exception pip_shims.shims.DistributionNotFound

Bases: pip._internal.exceptions.InstallationError

Raised when a distribution cannot be found to satisfy a requirement

args

with_traceback ()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class pip_shims.shims.FormatControl (*no_binary=None, only_binary=None*)

Bases: object

Helper for managing formats from which a package can be installed.

disallow_binaries ()

get_allowed_formats (*canonical_name*)

static handle_mutual_excludes (*value, target, other*)

no_binary

only_binary

class pip_shims.shims.FrozenRequirement (*name, req, editable, comments=()*)
 Bases: object

classmethod from_dist (*dist*)

pip_shims.shims.get_installed_distributions (*local_only=True, skip={'argparse', 'python', 'wsgiref'}, include_editables=True, editables_only=False, user_only=False, paths=None*)

Return a list of installed Distribution objects.

If *local_only* is True (default), only return installations local to the current virtualenv, if in a virtualenv.

skip argument is an iterable of lower-case project names to ignore; defaults to `stdlib_pkgs`

If *include_editables* is False, don't report editables.

If *editables_only* is True, only report editables.

If *user_only* is True, only report installations in the user site directory.

If *paths* is set, only report the distributions present at the specified list of locations.

exception pip_shims.shims.InstallationError

Bases: pip._internal.exceptions.PipError

General exception during installation

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pip_shims.shims.UninstallationError

Bases: pip._internal.exceptions.PipError

General exception during uninstallation

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pip_shims.shims.RequirementsFileParseError

Bases: pip._internal.exceptions.InstallationError

Raised when a general error occurs parsing a requirements file line.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pip_shims.shims.BestVersionAlreadyInstalled

Bases: pip._internal.exceptions.PipError

Raised when the most up-to-date version of a package is already installed.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.shims.BadCommand`

Bases: `pip._internal.exceptions.PipError`

Raised when virtualenv or a command is not found

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.shims.CommandError`

Bases: `pip._internal.exceptions.PipError`

Raised when there is an error in command-line arguments

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.shims.PreviousBuildDirError`

Bases: `pip._internal.exceptions.PipError`

Raised when there's a previous conflicting build directory

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

`pip_shims.shims.install_req_from_editable` (*editable_req*, *comes_from=None*,
use_pep517=None, *isolated=False*,
options=None, *constraint=False*,
user_supplied=False)

`pip_shims.shims.install_req_from_line` (*name*, *comes_from=None*, *use_pep517=None*,
isolated=False, *options=None*, *constraint=False*,
line_source=None, *user_supplied=False*)

Creates an InstallRequirement from a name, which might be a requirement, directory containing 'setup.py', filename, or URL.

Parameters *line_source* – An optional string describing where the line is from, for logging purposes in case of an error.

`pip_shims.shims.install_req_from_req_string` (*req_string*, *comes_from=None*, *iso-*
lated=False, *use_pep517=None*,
user_supplied=False)

class `pip_shims.shims.InstallRequirement` (*req*, *comes_from*, *editable=False*, *link=None*,
markers=None, *use_pep517=None*, *iso-*
lated=False, *install_options=None*,
global_options=None, *hash_options=None*, *con-*
straint=False, *extras=()*, *user_supplied=False*)

Bases: `pip._internal.req.req_install.InstallRequirement`

_generate_metadata()

Invokes metadata generator functions, with the required arguments.

_get_archive_name (*path*, *parentdir*, *rootdir*)

_set_requirement()

Set requirement after generating metadata.

archive (*build_dir*)

Saves archive to provided build_dir.

Used for saving downloaded VCS requirements as part of *pip download*.

assert_source_matches_version ()

build_location (*build_dir, autodelete, parallel_builds*)

check_if_exists (*use_user_site*)

Find an installed distribution that satisfies or conflicts with this requirement, and set self.satisfied_by or self.should_reinstall appropriately.

ensure_build_location (*build_dir, autodelete, parallel_builds*)

ensure_has_source_dir (*parent_dir, autodelete=False, parallel_builds=False*)

Ensure that a source_dir is set.

This will create a temporary build dir if the name of the requirement isn't known yet.

Parameters **parent_dir** – The ideal pip parent_dir for the source_dir. Generally src_dir for editables and build_dir for sdist.

Returns self.source_dir

format_debug ()

An un-tested helper for getting state, for debugging.

from_editable = <pip_shims.utils.BaseMethod object>

from_line = <pip_shims.utils.BaseMethod object>

from_path ()

Format a nice indicator to show where this “comes from”

get_dist ()

has_hash_options

Return whether any known-good hashes are specified as options.

These activate `--require-hashes` mode; hashes specified as part of a URL do not.

hashes (*trust_internet=True*)

Return a hash-comparer that considers my option- and URL-based hashes to be known-good.

Hashes in URLs—ones embedded in the requirements file, not ones downloaded from an index server—are almost peers with ones from flags. They satisfy `--require-hashes` (whether it was implicitly or explicitly activated) but do not activate it. md5 and sha224 are not allowed in flags, which should nudge people toward good algos. We always OR all hashes together, even ones from URLs.

Parameters **trust_internet** – Whether to trust URL-based (`#md5=...`) hashes downloaded from the internet, as by `populate_link()`

install (*install_options, global_options=None, root=None, home=None, prefix=None, warn_script_location=True, use_user_site=False, pycompile=True*)

installed_version

is_pinned

Return whether I am pinned to an exact version.

For example, `some-package==1.2` is pinned; `some-package>1.2` is not.

is_wheel

load_pyproject_toml ()

Load the pyproject.toml file.

After calling this routine, all of the attributes related to PEP 517 processing for this requirement have been set. In particular, the `use_pep517` attribute can be used to determine whether we should follow the PEP 517 or legacy (`setup.py`) code path.

match_markers (*extras_requested=None*)

metadata

name

prepare_metadata ()

Ensure that project metadata is available.

Under PEP 517, call the backend hook to prepare the metadata. Under legacy processing, call `setup.py egg-info`.

pyproject_toml_path

setup_py_path

specifier

uninstall (*auto_confirm=False, verbose=False*)

Uninstall the distribution currently satisfying this requirement.

Prompts before removing or modifying files unless `auto_confirm` is `True`.

Refuses to delete or modify files outside of `sys.prefix` - thus uninstallation within a virtual environment can only modify that virtual environment, even if the virtualenv is linked to global site-packages.

unpacked_source_directory

update_editable (*obtain=True*)

warn_on_mismatching_name ()

`pip_shims.shims.is_archive_file` (*name*)

Return `True` if *name* is considered as an archive file.

`pip_shims.shims.is_file_url` (*link*)

class `pip_shims.shims.Downloader` (*session, progress_bar*)

Bases: `object`

`pip_shims.shims.unpack_url` (*link, location, downloader, download_dir=None, hashes=None*)

Unpack link into location, downloading if required.

Parameters *hashes* – A Hashes object, one of whose embedded hashes must match, or `HashMismatch` will be raised. If the Hashes is empty, no matches are required, and unhashable types of requirements (like VCS ones, which would ordinarily raise `HashUnsupported`) are allowed.

`pip_shims.shims.is_installable_dir` (*path*)

Is path a directory containing `setup.py` or `pyproject.toml`?

class `pip_shims.shims.Link` (*url, comes_from=None, requires_python=None, yanked_reason=None, cache_link_parsing=True*)

Bases: `pip._internal.models.link.Link`

Parameters

- **url** – url of the resource pointed to (href of the link)
- **comes_from** – instance of `HTMLPage` where the link was found, or string.

- **requires_python** – String containing the *Requires-Python* metadata field, specified in PEP 345. This may be specified by a `data-requires-python` attribute in the HTML link tag, as described in PEP 503.
- **yanked_reason** – the reason the file has been yanked, if the file has been yanked, or `None` if the file hasn't been yanked. This is the value of the “data-yanked” attribute, if present, in a simple repository HTML link. If the file has been yanked but no reason was provided, this should be the empty string. See PEP 592 for more information and the specification.
- **cache_link_parsing** – A flag that is used elsewhere to determine whether resources retrieved from this link should be cached. PyPI index urls should generally have this set to `False`, for example.

`_compare` (*other, method*)

`_compare_key`

`_defining_class`

`_egg_fragment_re = re.compile('[#&]egg=([^&]*)')`

`_hash_re = re.compile('(sha1|sha224|sha384|sha256|sha512|md5)=([a-f0-9]+)')`

`_parsed_url`

`_subdirectory_fragment_re = re.compile('[#&]subdirectory=([^&]*)')`

`_url`

`cache_link_parsing`

`comes_from`

`egg_fragment`

`ext`

`file_path`

`filename`

`has_hash`

`hash`

`hash_name`

`is_artifact`

`is_existing_dir()`

`is_file`

`is_hash_allowed` (*hashes*)

Return True if the link has a hash and it is allowed.

`is_vcs`

`is_wheel`

`is_yanked`

`netloc`

This can contain auth information.

`path`

`requires_python`

scheme
show_url
splitext()
subdirectory_fragment
url
url_without_fragment
yanked_reason

`pip_shims.shims.make_abstract_dist(install_req)`
 Returns a Distribution for the given InstallRequirement

`pip_shims.shims.make_distribution_for_install_requirement(install_req)`
 Returns a Distribution for the given InstallRequirement

`pip_shims.shims.make_option_group(group, parser)`
 Return an OptionGroup object group – assumed to be dict with ‘name’ and ‘options’ keys parser – an optparse Parser

class `pip_shims.shims.PackageFinder`(*link_collector*, *target_python*, *allow_yanked*, *format_control=None*, *candidate_prefs=None*, *ignore_requires_python=None*)

Bases: `object`

This finds packages.

This is meant to match easy_install’s technique for looking for packages, by reading pages and looking for appropriate links.

This constructor is primarily meant to be used by the create() class method and from tests.

Parameters

- **format_control** – A FormatControl object, used to control the selection of source packages / binary packages when consulting the index and links.
- **candidate_prefs** – Options to use when creating a CandidateEvaluator object.

`_log_skipped_link(link, reason)`

`_sort_links(links)`

Returns elements of links in order, non-egg links first, egg links second, while eliminating duplicates

allow_all_prereleases

classmethod `create(link_collector, selection_prefs, target_python=None)`

Create a PackageFinder.

Parameters

- **selection_prefs** – The candidate selection preferences, as a SelectionPreferences object.
- **target_python** – The target Python interpreter to use when checking compatibility. If None (the default), a TargetPython object will be constructed from the running Python.

`evaluate_links(link_evaluator, links)`

Convert links that are candidates to InstallationCandidate objects.

`find_all_candidates(project_name)`

Find all available InstallationCandidate for project_name

This checks `index_urls` and `find_links`. All versions found are returned as an `InstallationCandidate` list.

See `LinkEvaluator.evaluate_link()` for details on which files are accepted.

find_best_candidate (*project_name, specifier=None, hashes=None*)

Find matches for the given project and specifier.

Parameters **specifier** – An optional object implementing *filter* (e.g. `packaging.specifiers.SpecifierSet`) to filter applicable versions.

Returns A `BestCandidateResult` instance.

find_links

find_requirement (*req, upgrade*)

Try to find a `Link` matching `req`

Expects `req`, an `InstallRequirement` and `upgrade`, a boolean Returns a `InstallationCandidate` if found, Raises `DistributionNotFound` or `BestVersionAlreadyInstalled` otherwise

get_install_candidate (*link_evaluator, link*)

If the `link` is a candidate for install, convert it to an `InstallationCandidate` and return it. Otherwise, return `None`.

index_urls

make_candidate_evaluator (*project_name, specifier=None, hashes=None*)

Create a `CandidateEvaluator` object to use.

make_link_evaluator (*project_name*)

prefer_binary

process_project_url (*project_url, link_evaluator*)

search_scope

set_allow_all_prereleases ()

set_prefer_binary ()

target_python

trusted_hosts

class `pip_shims.shims.CandidateEvaluator` (*project_name, supported_tags, specifier, prefer_binary=False, allow_all_prereleases=False, hashes=None*)

Bases: `object`

Responsible for filtering and sorting candidates for installation based on what tags are valid.

Parameters **supported_tags** – The PEP 425 tags supported by the target Python in order of preference (most preferred first).

__sort_key (*candidate*)

Function to pass as the *key* argument to a call to `sorted()` to sort `InstallationCandidates` by preference.

Returns a tuple such that tuples sorting as greater using Python's default comparison operator are more preferred.

The preference is as follows:

First and foremost, candidates with allowed (matching) hashes are always preferred over candidates without matching hashes. This is because e.g. if the only candidate with an allowed hash is yanked, we still want to use that candidate.

Second, excepting hash considerations, candidates that have been yanked (in the sense of PEP 592) are always less preferred than candidates that haven't been yanked. Then:

If not finding wheels, they are sorted by version only. If finding wheels, then the sort order is by version, then:

1. existing installs
2. wheels ordered via `Wheel.support_index_min(self._supported_tags)`
3. source archives

If `prefer_binary` was set, then all wheels are sorted above sources.

Note: it was considered to embed this logic into the Link comparison operators, but then different sdist links with the same version, would have to be considered equal

compute_best_candidate (*candidates*)

Compute and return a *BestCandidateResult* instance.

classmethod create (*project_name*, *target_python=None*, *prefer_binary=False*, *allow_all_prereleases=False*, *specifier=None*, *hashes=None*)

Create a *CandidateEvaluator* object.

Parameters

- **target_python** – The target Python interpreter to use when checking compatibility. If None (the default), a *TargetPython* object will be constructed from the running Python.
- **specifier** – An optional object implementing *filter* (e.g. *packaging.specifiers.SpecifierSet*) to filter applicable versions.
- **hashes** – An optional collection of allowed hashes.

get_applicable_candidates (*candidates*)

Return the applicable candidates from a list of candidates.

sort_best_candidate (*candidates*)

Return the best candidate per the instance's sort order, or None if no candidate is acceptable.

class `pip_shims.shims.CandidatePreferences` (*prefer_binary=False*, *allow_all_prereleases=False*)

Bases: `object`

Encapsulates some of the preferences for filtering and sorting *InstallationCandidate* objects.

Parameters `allow_all_prereleases` – Whether to allow all pre-releases.

class `pip_shims.shims.LinkCollector` (*session*, *search_scope*)

Bases: `object`

Responsible for collecting *Link* objects from all configured locations, making network requests as needed.

The class's main method is its `collect_links()` method.

collect_links (*project_name*)

Find all available links for the given project name.

Returns All the *Link* objects (unfiltered), as a *CollectedLinks* object.

classmethod create (*session*, *options*, *suppress_no_index=False*)

Parameters

- **session** – The *Session* to use to make requests.

- **suppress_no_index** – Whether to ignore the `--no-index` option when constructing the `SearchScope` object.

fetch_page (*location*)

Fetch an HTML page containing package links.

find_links

class `pip_shims.shims.LinkEvaluator` (*project_name, canonical_name, formats, target_python, allow_yanked, ignore_requires_python=None*)

Bases: `object`

Responsible for evaluating links for a particular project.

Parameters

- **project_name** – The user supplied package name.
- **canonical_name** – The canonical package name.
- **formats** – The formats allowed for this package. Should be a set with ‘binary’ or ‘source’ or both in it.
- **target_python** – The target Python interpreter to use when evaluating link compatibility. This is used, for example, to check wheel compatibility, as well as when checking the Python version, e.g. the Python version embedded in a link filename (or egg fragment) and against an HTML link’s optional PEP 503 “data-requires-python” attribute.
- **allow_yanked** – Whether files marked as yanked (in the sense of PEP 592) are permitted to be candidates for install.
- **ignore_requires_python** – Whether to ignore incompatible PEP 503 “data-requires-python” values in HTML links. Defaults to False.

`_py_version_re = re.compile('-py([123]\\.|[0-9]?)$')`

evaluate_link (*link*)

Determine whether a link is a candidate for installation.

Returns A tuple (`is_candidate, result`), where *result* is (1) a version string if *is_candidate* is True, and (2) if *is_candidate* is False, an optional string to log the reason the link fails to qualify.

class `pip_shims.shims.TargetPython` (*platform=None, py_version_info=None, abi=None, implementation=None*)

Bases: `object`

Encapsulates the properties of a Python interpreter one is targeting for a package install, download, etc.

Parameters

- **platform** – A string or None. If None, searches for packages that are supported by the current system. Otherwise, will find packages that can be built on the platform passed in. These packages will only be downloaded for distribution: they will not be built locally.
- **py_version_info** – An optional tuple of ints representing the Python version information to use (e.g. `sys.version_info[:3]`). This can have length 1, 2, or 3 when provided.
- **abi** – A string or None. This is passed to `compatibility_tags.py`’s `get_supported()` function as is.
- **implementation** – A string or None. This is passed to `compatibility_tags.py`’s `get_supported()` function as is.

`_given_py_version_info`

`_valid_tags`

abi

format_given()

Format the given, non-None attributes for display.

get_tags()

Return the supported PEP 425 tags to check wheel candidates against.

The tags are returned in order of preference (most preferred first).

implementation

platform

py_version

py_version_info

class pip_shims.shims.**SearchScope** (*find_links, index_urls*)

Bases: `object`

Encapsulates the locations that pip is configured to search.

classmethod **create** (*find_links, index_urls*)

Create a SearchScope object after normalizing the *find_links*.

find_links

get_formatted_locations()

get_index_urls_locations (*project_name*)

Returns the locations found via self.index_urls

Checks the url_name on the main (first in the list) index and use this url_name to produce all locations

index_urls

class pip_shims.shims.**SelectionPreferences** (*allow_yanked, allow_all_prereleases=False, format_control=None, prefer_binary=False, ignore_requires_python=None*)

Bases: `object`

Encapsulates the candidate selection preferences for downloading and installing files.

Create a SelectionPreferences object.

Parameters

- **allow_yanked** – Whether files marked as yanked (in the sense of PEP 592) are permitted to be candidates for install.
- **format_control** – A FormatControl object or None. Used to control the selection of source packages / binary packages when consulting the index and links.
- **prefer_binary** – Whether to prefer an old, but valid, binary dist over a new source dist.
- **ignore_requires_python** – Whether to ignore incompatible “Requires-Python” values in links. Defaults to False.

allow_all_prereleases

allow_yanked

format_control

ignore_requires_python

prefer_binary

`pip_shims.shims.parse_requirements` (*filename*, *session*, *finder=None*, *comes_from=None*, *options=None*, *constraint=False*)

Parse a requirements file and yield `ParsedRequirement` instances.

Parameters

- **filename** – Path or url of requirements file.
- **session** – `PipSession` instance.
- **finder** – Instance of `pip.index.PackageFinder`.
- **comes_from** – Origin description of requirements.
- **options** – cli options.
- **constraint** – If true, parsing a constraint file rather than requirements file.

`pip_shims.shims.path_to_url` (*path*)

Convert a path to a file: URL. The path will be made absolute and have quoted path parts.

exception `pip_shims.shims.PipError`

Bases: `Exception`

Base pip exception

args

with_traceback ()

`Exception.with_traceback(tb)` – set `self.__traceback__` to `tb` and return `self`.

class `pip_shims.shims.RequirementPreparer` (*build_dir*, *download_dir*, *src_dir*, *wheel_download_dir*, *build_isolation*, *req_tracker*, *downloader*, *finder*, *require_hashes*, *use_user_site*)

Bases: `object`

Prepares a Requirement

`_download_should_save`

`_ensure_link_req_src_dir` (*req*, *download_dir*, *parallel_builds*)

Ensure `source_dir` of a linked `InstallRequirement`.

`_get_linked_req_hashes` (*req*)

`_log_preparing_link` (*req*)

Log the way the link prepared.

`prepare_editable_requirement` (*req*)

Prepare an editable requirement

`prepare_installed_requirement` (*req*, *skip_reason*)

Prepare an already-installed requirement

`prepare_linked_requirement` (*req*, *parallel_builds=False*)

Prepare a requirement to be obtained from `req.link`.

class `pip_shims.shims.RequirementSet` (*check_supported_wheels=True*)

Bases: `object`

Create a `RequirementSet`.

`add_named_requirement` (*install_req*)

`add_requirement` (*install_req*, *parent_req_name=None*, *extras_requested=None*)

Add `install_req` as a requirement to install.

Parameters

- **parent_req_name** – The name of the requirement that needed this added. The name is used because when multiple unnamed requirements resolve to the same name, we could otherwise end up with dependency links that point outside the Requirements set. `parent_req` must already be added. Note that `None` implies that this is a user supplied requirement, vs an inferred one.
- **extras_requested** – an iterable of extras used to evaluate the environment markers.

Returns Additional requirements to scan. That is either `[]` if the requirement is not applicable, or `[install_req]` if the requirement is applicable and has just been added.

add_unnamed_requirement (*install_req*)

all_requirements

get_requirement (*name*)

has_requirement (*name*)

class `pip_shims.shims.RequirementTracker` (*root*)

Bases: `object`

_entry_path (*link*)

add (*req*)

Add an `InstallRequirement` to build tracking.

cleanup ()

remove (*req*)

Remove an `InstallRequirement` from build tracking.

track (*req*)

class `pip_shims.shims.TempDirectory` (*path=None, delete=<pip._internal.utils.temp_dir._Default object>, kind='temp', globally_managed=False*)

Bases: `object`

Helper class that owns and cleans up a temporary directory.

This class can be used as a context manager or as an OO representation of a temporary directory.

Attributes:

path Location to the created temporary directory

delete Whether the directory should be deleted when exiting (when used as a contextmanager)

Methods:

cleanup() Deletes the temporary directory

When used as a context manager, if the `delete` attribute is `True`, on exiting the context the temporary directory is deleted.

_create (*kind*)

Create a temporary directory and store its path in `self.path`

cleanup ()

Remove the temporary directory created and reset state

path

`pip_shims.shims.global_tempdir_manager` ()

```

pip_shims.shims.shim_unpack(*, unpack_fn=<pip_shims.models.ShimmedPathCollection object>,
                             download_dir, tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection
                             object>, ireq=None, link=None, location=None, hashes=None,
                             progress_bar='off', only_download=None, down-
                             loader_provider=<pip_shims.models.ShimmedPathCollection
                             object>, session=None)

```

Accepts all parameters that have been valid to pass to `pip._internal.download.unpack_url()` and selects or drops parameters as needed before invoking the provided callable.

Parameters

- **unpack_fn** (*Callable*) – A callable or shim referring to the pip implementation
- **download_dir** (*str*) – The directory to download the file to
- **tempdir_manager_provider** (*TShimmedFunc*) – A callable or shim referring to `global_tempdir_manager` function from pip or a shimmed no-op context manager

:param Optional[InstallRequirement] ireq: an Install Requirement instance, defaults to None

:param Optional[Link] link: A Link instance, defaults to None.

Parameters

- **location** (*Optional[str]*) – A location or source directory if the target is a VCS url, defaults to None.
- **hashes** (*Optional[Any]*) – A Hashes instance, defaults to None
- **progress_bar** (*str*) – Indicates progress par usage during download, defatuls to off.
- **only_download** (*Optional[bool]*) – Whether to skip install, defaults to None.
- **downloader_provider** (*Optional[ShimmedPathCollection]*) – A down-loader class to instantiate, if applicable.
- **session** (*Optional[Session]*) – A PipSession instance, defaults to None.

Returns The result of unpacking the url.

Return type None

```

pip_shims.shims.get_requirement_tracker()

```

```

class pip_shims.shims.Resolver(preparer, finder, wheel_cache, make_install_req, use_user_site,
                               ignore_dependencies, ignore_installed, ignore_requires_python,
                               force_reinstall, upgrade_strategy, py_version_info=None)

```

Bases: `pip._internal.resolution.base.BaseResolver`

Resolves which packages need to be installed/uninstalled to perform the requested operation without breaking the requirements of any package.

```

_allowed_strategies = {'eager', 'only-if-needed', 'to-satisfy-only'}

```

```

_check_skip_installed(req_to_install)

```

Check if `req_to_install` should be skipped.

This will check if the req is installed, and whether we should upgrade or reinstall it, taking into account all the relevant user options.

After calling this `req_to_install` will only have `satisfied_by` set to None if the `req_to_install` is to be upgraded/reinstalled etc. Any other value will be a dist recording the current thing installed that satisfies the requirement.

Note that for vcs urls and the like we can't assess skipping in this routine - we simply identify that we need to pull the thing down, then later on it is pulled down and introspected to assess upgrade/ reinstalls etc.

Returns A text reason for why it was skipped, or None.

`_find_requirement_link` (*req*)

`_get_abstract_dist_for` (*req*)

Takes a InstallRequirement and returns a single AbstractDist representing a prepared variant of the same.

`_is_upgrade_allowed` (*req*)

`_populate_link` (*req*)

Ensure that if a link can be found for this, that it is found.

Note that req.link may still be None - if the requirement is already installed and not needed to be upgraded based on the return value of `_is_upgrade_allowed()`.

If preparer.require_hashes is True, don't use the wheel cache, because cached wheels, always built locally, have different hashes than the files downloaded from the index server and thus throw false hash mismatches. Furthermore, cached wheels at present have undeterministic contents due to file modification times.

`_resolve_one` (*requirement_set, req_to_install*)

Prepare a single requirements file.

Returns A list of additional InstallRequirements to also install.

`_set_req_to_reinstall` (*req*)

Set a requirement to be installed.

`get_installation_order` (*req_set*)

Create the installation order.

The installation order is topological - requirements are installed before the requiring thing. We break cycles at an arbitrary point, and make no other guarantees.

`resolve` (*root_reqs, check_supported_wheels*)

Resolve what operations need to be done

As a side-effect of this method, the packages (and their dependencies) are downloaded, unpacked and prepared for installation. This preparation is done by `pip.operations.prepare`.

Once PyPI has static dependency metadata available, it would be possible to move the preparation to become a step separated from dependency resolution.

class `pip_shims.shims.SafeFileCache` (*directory*)

Bases: `pip._vendor.cachecontrol.cache.BaseCache`

A file based cache which is safe to use even when the target directory may not be accessible or writable.

`_get_cache_path` (*name*)

`close` ()

`delete` (*key*)

`get` (*key*)

`set` (*key, value*)

class `pip_shims.shims.UninstallPathSet` (*dist*)

Bases: `object`

A set of file paths to be removed in the uninstallation of a requirement.

`_allowed_to_proceed` (*verbose*)
 Display which files would be deleted and prompt for confirmation

`_permitted` (*path*)
 Return True if the given path is one we are permitted to remove/modify, False otherwise.

`add` (*path*)

`add_pth` (*pth_file, entry*)

`commit` ()
 Remove temporary save dir: rollback will no longer be possible.

`classmethod from_dist` (*dist*)

`remove` (*auto_confirm=False, verbose=False*)
 Remove paths in `self.paths` with confirmation (unless `auto_confirm` is True).

`rollback` ()
 Rollback the changes previously made by `remove()`.

`pip_shims.shims.url_to_path` (*url*)
 Convert a file: URL to a path.

`class pip_shims.shims.VcsSupport`

Bases: `object`

`_registry` = {'bzzr': <pip._internal.vcs.bazaar.Bazaar object>, 'git': <pip._internal.vcs.git.Git object>, 'hg': <pip._internal.vcs.hg.Hg object>, 'ssh': <pip._internal.vcs.ssh.Ssh object>, 'svn': <pip._internal.vcs.svn.Svn object>, 'sftp': <pip._internal.vcs.sftp.Sftp object>}

`all_schemes`

`backends`

`dirname`

`get_backend` (*name*)
 Return a VersionControl object or None.

`get_backend_for_dir` (*location*)
 Return a VersionControl object if a repository of that type is found at the given directory.

`get_backend_for_scheme` (*scheme*)
 Return a VersionControl object or None.

`register` (*cls*)

`schemes` = ['ssh', 'git', 'hg', 'bzzr', 'sftp', 'svn']

`unregister` (*name*)

`class pip_shims.shims.Wheel` (*filename*)

Bases: `object`

`get_formatted_file_tags` ()
 Return the wheel's tags as a sorted list of strings.

`support_index_min` (*tags*)
 Return the lowest index that one of the wheel's file_tag combinations achieves in the given list of supported tags.

For example, if there are 8 supported tags and one of the file tags is first in the list, then return 0.

Parameters `tags` – the PEP 425 tags to check the wheel against, in order with most preferred first.

Raises `ValueError` – If none of the wheel's file tags match one of the supported tags.

supported (*tags*)

Return whether the wheel is compatible with one of the given tags.

Parameters *tags* – the PEP 425 tags to check the wheel against.

```
wheel_file_re = re.compile('^(?P<namever>( ?P<name>.+) - (?P<ver>. *?))\n ((- ?P<build>\
```

class `pip_shims.shims.WheelCache` (*cache_dir, format_control*)

Bases: `pip._internal.cache.Cache`

Wraps EphemWheelCache and SimpleWheelCache into a single Cache

This Cache allows for gracefully degradation, using the ephem wheel cache when a certain link is not found in the simple wheel cache first.

`_get_cache_path_parts` (*link*)

Get parts of part that must be `os.path.join`d with `cache_dir`

`_get_cache_path_parts_legacy` (*link*)

Get parts of part that must be `os.path.join`d with `cache_dir`

Legacy cache key (pip < 20) for compatibility with older caches.

`_get_candidates` (*link, canonical_package_name*)

`get` (*link, package_name, supported_tags*)

Returns a link to a cached item if it exists, otherwise returns the passed link.

`get_cache_entry` (*link, package_name, supported_tags*)

Returns a `CacheEntry` with a link to a cached item if it exists or `None`. The cache entry indicates if the item was found in the persistent or ephemeral cache.

`get_ephem_path_for_link` (*link*)

`get_path_for_link` (*link*)

Return a directory to store cached items in for link.

`get_path_for_link_legacy` (*link*)

`pip_shims.shims.build` (*requirements, wheel_cache, build_options, global_options*)

Build wheels.

Returns The list of `InstallRequirement` that succeeded to build and the list of `InstallRequirement` that failed to build.

`pip_shims.shims.build_one` (*req, output_dir, build_options, global_options*)

Build one wheel.

Returns The filename of the built wheel, or `None` if the build failed.

`pip_shims.shims.build_one_inside_env` (*req, output_dir, build_options, global_options*)

class `pip_shims.shims.AbstractDistribution` (*req*)

Bases: `object`

A base class for handling installable artifacts.

The requirements for anything installable are as follows:

- we must be able to determine the requirement name (or we can't correctly handle the non-upgrade case).
- for packages with setup requirements, we must also be able to determine their requirements without installing additional packages (for the same reason as run-time dependencies)
- we must be able to create a `Distribution` object exposing the above metadata.

`_abc_impl` = `<_abc_data object>`

```
get_pkg_resources_distribution()
```

```
prepare_distribution_metadata(finder, build_isolation)
```

```
class pip_shims.shims.InstalledDistribution(req)
```

```
Bases: pip._internal.distributions.base.AbstractDistribution
```

Represents an installed package.

This does not need any preparation as the required information has already been computed.

```
_abc_impl = <_abc_data object>
```

```
get_pkg_resources_distribution()
```

```
prepare_distribution_metadata(finder, build_isolation)
```

```
class pip_shims.shims.SourceDistribution(req)
```

```
Bases: pip._internal.distributions.base.AbstractDistribution
```

Represents a source distribution.

The preparation step for these needs metadata for the packages to be generated, either using PEP 517 or using the legacy *setup.py egg_info*.

```
_abc_impl = <_abc_data object>
```

```
_setup_isolation(finder)
```

```
get_pkg_resources_distribution()
```

```
prepare_distribution_metadata(finder, build_isolation)
```

```
class pip_shims.shims.WheelDistribution(req)
```

```
Bases: pip._internal.distributions.base.AbstractDistribution
```

Represents a wheel distribution.

This does not need any preparation as wheels can be directly unpacked.

```
_abc_impl = <_abc_data object>
```

```
get_pkg_resources_distribution()
```

Loads the metadata from the wheel file into memory and returns a Distribution that uses it, not relying on the wheel file or requirement.

```
prepare_distribution_metadata(finder, build_isolation)
```

```
pip_shims.shims.wheel_cache(cache_dir=None, format_control=None, *,
                           wheel_cache_provider=<pip_shims.models.ShimmedPathCollection
                           object>, format_control_provider=<pip_shims.models.ShimmedPathCollection
                           object>, tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection
                           object>)
```

```
pip_shims.shims.get_package_finder(install_cmd=None, options=None, session=None,
                                  platform=None, python_versions=None,
                                  abi=None, implementation=None, target
                                  get_python=None, ignore_requires_python=None,
                                  *, target_python_builder=<class
                                  'pip._internal.models.target_python.TargetPython'>, in
                                  install_cmd_provider=<pip_shims.models.ShimmedPathCollection
                                  object>)
```

Shim for compatibility to generate package finders.

Build and return a PackageFinder instance using the InstallCommand helper method to construct the finder, shimmed with backports as needed for compatibility.

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_cmd** – A `InstallCommand` instance which is used to generate the finder.
- **options** (*optparse.Values*) – An optional `optparse.Values` instance generated by calling `install_cmd.parser.parse_args()` typically.
- **session** – An optional session instance, can be created by the `install_cmd`.
- **platform** (*Optional[str]*) – An optional platform string, e.g. `linux_x86_64`
- **python_versions** (*Optional[Tuple[str, ...]]*) – A tuple of 2-digit strings representing python versions, e.g. (“27”, “35”, “36”, “37”...)
- **abi** (*Optional[str]*) – The target abi to support, e.g. “cp38”
- **implementation** (*Optional[str]*) – An optional implementation string for limiting searches to a specific implementation, e.g. “cp” or “py”
- **target_python** – A `TargetPython` instance (will be translated to alternate arguments if necessary on incompatible pip versions).
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore `requires_python` on resulting candidates, only valid after pip version 19.3.1
- **target_python_builder** – A ‘TargetPython’ builder (e.g. the class itself, uninstantiated)

Returns A `pip._internal.index.package_finder.PackageFinder` instance

Return type `pip._internal.index.package_finder.PackageFinder`

Example

```
>>> from pip_shims.shims import InstallCommand, get_package_finder
>>> install_cmd = InstallCommand()
>>> finder = get_package_finder(
...     install_cmd, python_versions=("27", "35", "36", "37", "38"),
...     implementation="
cp"
... )
>>> candidates = finder.find_all_candidates("requests")
>>> requests_222 = next(iter(c for c in candidates if c.version.public == "2.22.0
... ))
>>> requests_222
<InstallationCandidate('requests', <Version('2.22.0')>, <Link https://files.
pythonhos
ted.org/packages/51/bd/
23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/r
equests-2.22.0-py2.py3-none-any.whl
#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9
a590f48c010551dc6c4b31 (from https://pypi.org/simple/requests/) (requires-python:>
=2.
7, !=3.0.*, !=3.1.*, !=3.2.*, !=3.3.*, !=3.4.*)>>>
```

```

pip_shims.shims.make_preparer(*, preparer_fn=<pip_shims.models.ShimmedPathCollection object>, req_tracker_fn=<pip_shims.models.ShimmedPathCollection object>, build_dir=None, src_dir=None, download_dir=None, wheel_download_dir=None, progress_bar='off', build_isolation=False, session=None, finder=None, options=None, require_hashes=None, use_user_site=None, req_tracker=None, install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>, downloader_provider=<pip_shims.models.ShimmedPathCollection object>, install_cmd=None, finder_provider=<pip_shims.models.ShimmedPathCollection object>)

```

Creates a requirement preparer for preparing pip requirements.

Provides a compatibility shim that accepts all previously valid arguments and discards any that are no longer used.

Raises

- **TypeError** – No requirement tracker provided and one cannot be generated
- **TypeError** – No valid sessions provided and one cannot be generated
- **TypeError** – No valid finders provided and one cannot be generated

Parameters

- **preparer_fn** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **req_tracker_fn** (*Optional[TShimmedFunc]*) – Callable or shim for generating requirement trackers, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **src_dir** (*Optional[str]*) – Directory to find or extract source files, defaults to None
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **progress_bar** (*str*) – Whether to display a progress bar, defaults to off
- **build_isolation** (*bool*) – Whether to build requirements in isolation, defaults to False
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **require_hashes** (*Optional[bool]*) – Whether to require hashes for preparation
- **use_user_site** (*Optional[bool]*) – Whether to use the user site directory for preparing requirements
- **TShimmedFunc[] req_tracker** (*Optional[Union[TReqTracker,]]*) – The requirement tracker to use for building packages, defaults to None
- **downloader_provider** (*Optional[TShimmedFunc]*) – A downloader provider

- **install_cmd** (*Optional*[*TCommandInstance*]) – The install command used to create the finder, session, and options if needed, defaults to None
- **finder_provider** (*Optional*[*TShimmedFunc*]) – A package finder provider

Yield A new requirement preparer instance

Return type `ContextManager[RequirementPreparer]`

Example

```
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_tracker
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> with make_preparer(
...     options=pip_options, finder=finder, session=session, install_cmd=ic
... ) as preparer:
...     print(preparer)
<pip._internal.operations.prepare.RequirementPreparer object at 0x7f8a2734be80>
```

`pip_shims.shims.get_resolver` (*, *resolver_fn*=<*pip_shims.models.ShimmedPathCollection object*>, *install_req_provider*=<*pip_shims.models.ShimmedPathCollection object*>, *format_control_provider*=<*pip_shims.models.ShimmedPathCollection object*>, *wheel_cache_provider*=<*pip_shims.models.ShimmedPathCollection object*>, *finder*=None, *upgrade_strategy*='to-satisfy-only', *force_reinstall*=None, *ignore_dependencies*=None, *ignore_requires_python*=None, *ignore_installed*=True, *use_user_site*=False, *isolated*=None, *wheel_cache*=None, *preparer*=None, *session*=None, *options*=None, *make_install_req*=None, *install_cmd_provider*=<*pip_shims.models.ShimmedPathCollection object*>, *install_cmd*=None, *use_pep517*=True)

A resolver creation compatibility shim for generating a resolver.

Consumes any argument that was previously used to instantiate a resolver, discards anything that is no longer valid.

Note: This is only valid for **pip >= 10.0.0**

Raises

- **ValueError** – A session is required but not provided and one cannot be created
- **ValueError** – A finder is required but not provided and one cannot be created
- **ValueError** – An install requirement provider is required and has not been provided

Parameters

- **resolver_fn** (*TShimmedFunc*) – The resolver function used to create new resolver instances.

- **install_req_provider** (*TShimmedFunc*) – The provider function to use to generate install requirements if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a `format_control` instance if needed.
- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to `None`.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to `None`
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to `None`
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to `None`
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to `True`
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to `False`
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to `None`
- **wheel_cache** (*Optional[TWheelCache]*) – The wheel cache to use, defaults to `None`
- **preparer** (*Optional[TPreparer]*) – The requirement preparer to use, defaults to `None`
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to `None`
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to `None`
- **make_install_req** (*Optional[functools.partial]*) – The partial function to pass in to the resolver for actually generating install requirements, if necessary
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to `None`.
- **use_pep517** (*bool*) – Whether to use the pep517 build process.

Returns A new resolver instance.

Return type `Resolver`

Example

```
>>> import os
>>> from tempdir import TemporaryDirectory
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_
↪ tracker,
...     get_resolver, InstallRequirement, RequirementSet
```

(continues on next page)

(continued from previous page)

```

... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> wheel_cache = WheelCache(USER_CACHE_DIR, FormatControl(None, None))
>>> with TemporaryDirectory() as temp_base:
...     reqset = RequirementSet()
...     ireq = InstallRequirement.from_line("requests")
...     ireq.is_direct = True
...     build_dir = os.path.join(temp_base, "build")
...     src_dir = os.path.join(temp_base, "src")
...     ireq.build_location(build_dir)
...     with make_preparer(
...         options=pip_options, finder=finder, session=session,
...         build_dir=build_dir, install_cmd=install_cmd,
...     ) as preparer:
...         resolver = get_resolver(
...             finder=finder, ignore_dependencies=False, ignore_requires_
↪python=True,
...             preparer=preparer, session=session, options=pip_options,
...             install_cmd=install_cmd, wheel_cache=wheel_cache,
...         )
...         resolver.require_hashes = False
...         reqset.add_requirement(ireq)
...         results = resolver.resolve(reqset)
...         #reqset.cleanup_files()
...         for result_req in reqset.requirements:
...             print(result_req)
requests
chardet
certifi
urllib3
idna

```

```

pip_shims.shims.get_requirement_set(install_command=None, *,
                                   req_set_provider=<pip_shims.models.ShimmedPathCollection
                                   object>, build_dir=None, src_dir=None, down-
                                   load_dir=None, wheel_download_dir=None, ses-
                                   sion=None, wheel_cache=None, upgrade=False,
                                   upgrade_strategy=None, ignore_installed=False,
                                   ignore_dependencies=False, force_reinstall=False,
                                   use_user_site=False, isolated=False, ig-
                                   nore_requires_python=False, require_hashes=None,
                                   cache_dir=None, options=None, in-
                                   stall_cmd_provider=<pip_shims.models.ShimmedPathCollection
                                   object>, wheel_cache_provider=<pip_shims.models.ShimmedPathCollection
                                   object>)

```

Creates a requirement set from the supplied parameters.

Not all parameters are passed through for all pip versions, but any invalid parameters will be ignored if they are not needed to generate a requirement set on the current pip version.

:param *ShimmedPathCollection* wheel_cache_provider: A context manager provider which resolves to a *WheelCache* instance

Parameters `install_command` – A `InstallCommand` instance which is used to generate the finder.

:param `ShimmedPathCollection req_set_provider`: A **provider** to build requirement set instances.

Parameters

- **build_dir** (`str`) – The directory to build requirements in. Removed in pip 10, defaults to None
- **source_dir** (`str`) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (`str`) – The directory to download requirement artifacts to. Removed in pip 10, defaults to None
- **wheel_download_dir** (`str`) – The directory to download wheels to. Removed in pip 10, defaults to None

:param `Session session`: **The pip session to use. Removed in pip 10**, defaults to None

Parameters

- **wheel_cache** (`WheelCache`) – The pip `WheelCache` instance to use for caching wheels. Removed in pip 10, defaults to None
- **upgrade** (`bool`) – Whether to try to upgrade existing requirements. Removed in pip 10, defaults to False.
- **upgrade_strategy** (`str`) – The upgrade strategy to use, e.g. “only-if-needed”. Removed in pip 10, defaults to None.
- **ignore_installed** (`bool`) – Whether to ignore installed packages when resolving. Removed in pip 10, defaults to False.
- **ignore_dependencies** (`bool`) – Whether to ignore dependencies of requirements when resolving. Removed in pip 10, defaults to False.
- **force_reinstall** (`bool`) – Whether to force reinstall of packages when resolving. Removed in pip 10, defaults to False.
- **use_user_site** (`bool`) – Whether to use user site packages when resolving. Removed in pip 10, defaults to False.
- **isolated** (`bool`) – Whether to resolve in isolation. Removed in pip 10, defaults to False.
- **ignore_requires_python** (`bool`) – Removed in pip 10, defaults to False.
- **require_hashes** (`bool`) – Whether to require hashes when resolving. Defaults to False.
- **options** (`Values`) – An `Values` instance from an `install cmd`
- **install_cmd_provider** (`ShimmedPathCollection`) – A shim for providing new install command instances.

Returns A new requirement set instance

Return type `RequirementSet`

```

pip_shims.shims.resolve(ireq, *, reqset_provider=<pip_shims.models.ShimmedPathCollection object>, req_tracker_provider=<pip_shims.models.ShimmedPathCollection object>, install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>, install_command=None, finder_provider=<pip_shims.models.ShimmedPathCollection object>, resolver_provider=<pip_shims.models.ShimmedPathCollection object>, wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>, format_control_provider=<pip_shims.models.ShimmedPathCollection object>, make_preparer_provider=<pip_shims.models.ShimmedPathCollection object>, tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection object>, options=None, session=None, resolver=None, finder=None, upgrade_strategy='to-satisfy-only', force_reinstall=None, ignore_dependencies=None, ignore_requires_python=None, ignore_installed=True, use_user_site=False, isolated=None, build_dir=None, source_dir=None, download_dir=None, cache_dir=None, wheel_download_dir=None, wheel_cache=None, require_hashes=None, check_supported_wheels=True)

```

Resolves the provided **InstallRequirement**, returning a dictionary.

Maps a dictionary of names to corresponding `InstallRequirement` values.

:param InstallRequirement ireq: An `InstallRequirement` to initiate the resolution process

:param *ShimmedPathCollection* reqset_provider: A provider to build requirement set instances.

:param *ShimmedPathCollection* req_tracker_provider: A provider to build requirement tracker instances

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_command** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None.

:param *ShimmedPathCollection* finder_provider: A provider to package finder instances.

:param *ShimmedPathCollection* resolver_provider: A provider to build resolver instances

Parameters

- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a `format_control` instance if needed.
- **make_preparer_provider** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **tempdir_manager_provider** (*Optional[TShimmedFunc]*) – Shim for generating `tempdir` manager for pip temporary directories
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None

:param Resolver resolver: A pre-existing resolver instance to use for resolution

Parameters

- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to None
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to None
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to None
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to True
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to False
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **cache_dir** (*str*) – The cache directory to use for caching artifacts during resolution
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **wheel_cache** (*Optional[TWheelCache]*) – The wheel cache to use, defaults to None
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **check_supported_wheels** (*bool*) – Whether to check support of wheels before including them in resolution.

Returns A dictionary mapping requirements to corresponding `~pip._internal.req.req_install.InstallRequirement`'s

Return type `InstallRequirement`

Example

```
>>> from pip_shims.shims import resolve, InstallRequirement
>>> ireq = InstallRequirement.from_line("requests>=2.20")
>>> results = resolve(ireq)
>>> for k, v in results.items():
...     print("{0}: {1!r}".format(k, v))
requests: <InstallRequirement object: requests>=2.20 from https://files.
↳pythonhosted.
org/packages/51/bd/23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/
↳reque
sts-2.22.0-py2.py3-none-any.whl
↳#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9a590
```

(continues on next page)

(continued from previous page)

```

f48c010551dc6c4b31 editable=False>
idna: <InstallRequirement object: idna<2.9,>=2.5 from https://files.pythonhosted.
↳org/
packages/14/2c/cd551d81dbe15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-
↳2.8-
py2.py3-none-any.whl
↳#sha256=ea8b7f6188e6fa117537c3df7da9fc686d485087abf6ac197f9c46432
f7e4a3c (from requests>=2.20) editable=False>
urllib3: <InstallRequirement object: urllib3!=1.25.0,!<1.25.1,<1.26,>=1.21.1 from_
↳htt
ps://files.pythonhosted.org/packages/b4/40/
↳a9837291310ee1ccc242ceb6ebfd9eb21539649f19
3a7c8c86ba15b98539/urllib3-1.25.7-py2.py3-none-any.whl
↳#sha256=a8a318824cc77d1fd4b2bec
2ded92646630d7fe8619497b142c84a9e6f5a7293 (from requests>=2.20) editable=False>
chardet: <InstallRequirement object: chardet<3.1.0,>=3.0.2 from https://files.
↳pythonh
sted.org/packages/bc/a9/
↳01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8
/chardet-3.0.4-py2.py3-none-any.whl
↳#sha256=fc323ffcaeaed0e0a02bf4d117757b98aed530d9ed
4531e3e15460124c106691 (from requests>=2.20) editable=False>
certifi: <InstallRequirement object: certifi>=2017.4.17 from https://files.
↳pythonhost
ed.org/packages/18/b0/
↳8146a4f8dd402f60744fa380bc73ca47303cccfc8b9190fd16a827281eac2/ce
rtifi-2019.9.11-py2.py3-none-any.whl
↳#sha256=fd7c7c74727ddcf00e9acd26bba8da604ffec95bf
1c2144e67aff7a8b50e6cef (from requests>=2.20) editable=False>

```

`pip_shims.shims.build_wheel` (*req=None, reqset=None, output_dir=None, preparer=None, wheel_cache=None, build_options=None, global_options=None, check_binary_allowed=None, no_clean=False, session=None, finder=None, install_command=None, req_tracker=None, build_dir=None, src_dir=None, download_dir=None, wheel_download_dir=None, cache_dir=None, use_user_site=False, use_pep517=None, *, format_control_provider=<pip_shims.models.ShimmedPathCollection object>, wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>, preparer_provider=<pip_shims.models.ShimmedPathCollection object>, wheel_builder_provider=<pip_shims.models.ShimmedPathCollection object>, build_one_provider=<pip_shims.models.ShimmedPathCollection object>, build_one_inside_env_provider=<pip_shims.models.ShimmedPathCollection object>, build_many_provider=<pip_shims.models.ShimmedPathCollection object>, install_command_provider=<pip_shims.models.ShimmedPathCollection object>, finder_provider=None, reqset_provider=<pip_shims.models.ShimmedPathCollection object>*)

Build a wheel or a set of wheels

Raises `TypeError` – Raised when no requirements are provided

Parameters

- **req** (*Optional* [`TInstallRequirement`]) – An `InstallRequirement` to build
- **reqset** (*Optional* [`TReqSet`]) – A `RequirementSet` instance (`pip<10`) or an iterable

of *InstallRequirement* instances (*pip* ≥ 10) to build

- **output_dir** (*Optional[str]*) – Target output directory, only useful when building one wheel using *pip* ≥ 20.0
- **preparer** (*Optional[TPreparer]*) – A preparer instance, defaults to *None*
- **wheel_cache** (*Optional[TWheelCache]*) – A wheel cache instance, defaults to *None*
- **build_options** (*Optional[List[str]]*) – A list of build options to pass in
- **global_options** (*Optional[List[str]]*) – A list of global options to pass in
- **bool] check_binary_allowed** (*Optional[Callable[TInstallRequirement,]]*) – A callable to check whether we are allowed to build and cache wheels for an *ireq*
- **no_clean** (*bool*) – Whether to avoid cleaning up wheels
- **session** (*Optional[TSession]*) – A *PipSession* instance to pass to create a *finder* if necessary
- **finder** (*Optional[TFinder]*) – A *PackageFinder* instance to use for generating a *WheelBuilder* instance on *pip* < 20
- **install_command** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to *None*.
- **req_tracker** (*Optional[TReqTracker]*) – An optional requirement tracker instance, if one already exists
- **build_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **src_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **download_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **wheel_download_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **cache_dir** (*Optional[str]*) – Passthrough cache directory for wheel cache options
- **use_user_site** (*bool*) – Whether to use the user site directory when preparing install requirements on *pip* < 20
- **use_pep517** (*Optional[bool]*) – When set to *True* or *False*, prefers building with or without pep517 as specified, otherwise uses requirement preference. Only works for single requirements.
- **format_control_provider** (*Optional[TShimmedFunc]*) – A provider for the *FormatControl* class
- **wheel_cache_provider** (*Optional[TShimmedFunc]*) – A provider for the *WheelCache* class
- **preparer_provider** (*Optional[TShimmedFunc]*) – A provider for the *RequirementPreparer* class
- **wheel_builder_provider** (*Optional[TShimmedFunc]*) – A provider for the *WheelBuilder* class, if it exists
- **build_one_provider** (*Optional[TShimmedFunc]*) – A provider for the *_build_one* function, if it exists
- **build_one_inside_env_provider** (*Optional[TShimmedFunc]*) – A provider for the *_build_one_inside_env* function, if it exists

- **build_many_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *build* function, if it exists
- **install_command_provider** (*Optional*[*TShimmedFunc*]) – A shim for providing new install command instances
- **finder_provider** (*TShimmedFunc*) – A provider to package finder instances
- **reqset_provider** (*TShimmedFunc*) – A provider for requirement set generation

Returns A tuple of successful and failed install requirements or else a path to a wheel

Return type `Optional[Union[str, Tuple[List[TInstallRequirement], List[TInstallRequirement]]]]`

17.1.5 pip_shims.environment

Module with functionality to learn about the environment.

`pip_shims.environment.get_base_import_path()`

`pip_shims.environment.get_pip_version(import_path='pip')`

`pip_shims.environment.is_type_checking()`

`pip_shims._strip_extras(path)`

class `pip_shims.SessionCommandMixin`

Bases: `pip._internal.cli.command_context.CommandContextMixin`

A class mixin for command classes needing `_build_session()`.

method `_build_session(options, retries=None, timeout=None)`

classmethod `_get_index_urls(options)`

Return a list of index urls from user-provided options.

method `enter_context(context_provider)`

method `get_default_session(options)`

Get a default-managed session.

method `main_context()`

class `pip_shims.Command(name='Default pip command', summary='PipCommand', isolated='Default pip command')`

Bases: `pip._internal.cli.base_command.Command`, `pip._internal.cli.req_command.SessionCommandMixin`

method `_build_session(options, retries=None, timeout=None)`

classmethod `_get_index_urls(options)`

Return a list of index urls from user-provided options.

method `_main(args)`

method `add_options()`

method `enter_context(context_provider)`

method `get_default_session(options)`

Get a default-managed session.

method `handle_pip_version_check(options)`

This is a no-op so that commands by default do not do the pip version check.

attribute `ignore_require_venv = False`

```

main (args)
main_context ()
parse_args (args)
run (options, args)
usage = None

class pip_shims.ConfigOptionParser (*args, **kwargs)
    Bases: pip._internal.cli.parser.CustomOptionParser

    Custom option parser which updates its defaults by checking the configuration files and environmental variables

    _add_help_option ()
    _add_version_option ()
    _check_conflict (option)
    _create_option_list ()
    _create_option_mappings ()
    _get_all_options ()
    _get_args (args)
    _get_ordered_configuration_items ()
    _init_parsing_state ()
    _match_long_opt (opt : string) → string
        Determine which long option string ‘opt’ matches, ie. which one it is an unambiguous abbreviation for.
        Raises BadOptionError if ‘opt’ doesn’t unambiguously match any long option string.
    _populate_option_list (option_list, add_help=True)
    _process_args (largs, rargs, values)
        _process_args(largs [[string],] rargs : [string], values : Values)

        Process command-line arguments and populate ‘values’, consuming options and arguments from ‘rargs’.
        If ‘allow_interspersed_args’ is false, stop at the first non-option argument. If true, accumulate any interspersed non-option arguments in ‘largs’.
    _process_long_opt (rargs, values)
    _process_short_opts (rargs, values)
    _share_option_mappings (parser)
    _update_defaults (defaults)
        Updates the given defaults with values from the config files and the environ. Does a little special handling for certain types of options (lists).

    add_option (Option)
        add_option(opt_str, ..., kwarg=val, ...)
    add_option_group (*args, **kwargs)
    add_options (option_list)
    check_default (option, key, val)

```

check_values (*values* : *Values*, *args* : [*string*])

-> (*values* : *Values*, *args* : [*string*])

Check that the supplied option values and leftover arguments are valid. Returns the option values and leftover arguments (possibly adjusted, possibly completely new – whatever you like). Default implementation just returns the passed-in values; subclasses may override as desired.

destroy ()

Declare that you are done with this OptionParser. This cleans up reference cycles so the OptionParser (and all objects referenced by it) can be garbage-collected promptly. After calling destroy(), the OptionParser is unusable.

disable_interspersed_args ()

Set parsing to stop on the first non-option. Use this if you have a command processor which runs another command that has options of its own and you want to make sure these options don't get confused.

enable_interspersed_args ()

Set parsing to not stop on the first non-option, allowing interspersing switches with command arguments. This is the default behavior. See also disable_interspersed_args() and the class documentation description of the attribute allow_interspersed_args.

error (*msg* : *string*)

Print a usage message incorporating 'msg' to stderr and exit. If you override this in a subclass, it should not return – it should either exit or raise an exception.

exit (*status*=0, *msg*=None)

expand_prog_name (*s*)

format_description (*formatter*)

format_epilog (*formatter*)

format_help (*formatter*=None)

format_option_help (*formatter*=None)

get_default_values ()

Overriding to make updating the defaults after instantiation of the option parser possible, _update_defaults() does the dirty work.

get_description ()

get_option (*opt_str*)

get_option_group (*opt_str*)

get_prog_name ()

get_usage ()

get_version ()

has_option (*opt_str*)

insert_option_group (*idx*, **args*, ***kwargs*)

Insert an OptionGroup at a given position.

option_list_all

Get a list of all options, including those in option groups.

parse_args (*args*=None, *values*=None)

parse_args(*args* [[*string*] = sys.argv[1:],] *values* : *Values* = None)

-> (values : Values, args : [string])

Parse the command-line options found in ‘args’ (default: sys.argv[1:]). Any errors result in a call to ‘error()’, which by default prints the usage message to stderr and calls sys.exit() with an error message. On success returns a pair (values, args) where ‘values’ is a Values instance (with all your option values) and ‘args’ is the list of arguments left over after parsing options.

print_help (*file* : *file* = *stdout*)

Print an extended help message, listing all options and any help text provided with them, to ‘file’ (default stdout).

print_usage (*file* : *file* = *stdout*)

Print the usage message for the current program (self.usage) to ‘file’ (default stdout). Any occurrence of the string “%prog” in self.usage is replaced with the name of the current program (basename of sys.argv[0]). Does nothing if self.usage is empty or not defined.

print_version (*file* : *file* = *stdout*)

Print the version message for this program (self.version) to ‘file’ (default stdout). As with print_usage(), any occurrence of “%prog” in self.version is replaced by the current program’s name. Does nothing if self.version is empty or undefined.

remove_option (*opt_str*)

set_conflict_handler (*handler*)

set_default (*dest*, *value*)

set_defaults (***kwargs*)

set_description (*description*)

set_process_default_values (*process*)

set_usage (*usage*)

standard_option_list = []

exception pip_shims.DistributionNotFound

Bases: pip._internal.exceptions.InstallationError

Raised when a distribution cannot be found to satisfy a requirement

args

with_traceback ()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class pip_shims.FormatControl (*no_binary=None*, *only_binary=None*)

Bases: object

Helper for managing formats from which a package can be installed.

disallow_binaries ()

get_allowed_formats (*canonical_name*)

static handle_mutual_excludes (*value*, *target*, *other*)

no_binary

only_binary

class pip_shims.FrozenRequirement (*name*, *req*, *editable*, *comments=()*)

Bases: object

classmethod **from_dist** (*dist*)

`pip_shims.get_installed_distributions` (*local_only=True*, *skip={'argparse', 'python', 'wsgiref'}*, *include_editables=True*, *editables_only=False*, *user_only=False*, *paths=None*)

Return a list of installed Distribution objects.

If `local_only` is `True` (default), only return installations local to the current virtualenv, if in a virtualenv.

`skip` argument is an iterable of lower-case project names to ignore; defaults to `stdlib_pkgs`

If `include_editables` is `False`, don't report editables.

If `editables_only` is `True`, only report editables.

If `user_only` is `True`, only report installations in the user site directory.

If `paths` is set, only report the distributions present at the specified list of locations.

exception `pip_shims.InstallationError`

Bases: `pip._internal.exceptions.PipError`

General exception during installation

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.UninstallationError`

Bases: `pip._internal.exceptions.PipError`

General exception during uninstallation

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.RequirementsFileParseError`

Bases: `pip._internal.exceptions.InstallationError`

Raised when a general error occurs parsing a requirements file line.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.BestVersionAlreadyInstalled`

Bases: `pip._internal.exceptions.PipError`

Raised when the most up-to-date version of a package is already installed.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.BadCommand`

Bases: `pip._internal.exceptions.PipError`

Raised when virtualenv or a command is not found

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.CommandError`

Bases: `pip._internal.exceptions.PipError`

Raised when there is an error in command-line arguments

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.PreviousBuildDirError`

Bases: `pip._internal.exceptions.PipError`

Raised when there's a previous conflicting build directory

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

`pip_shims.install_req_from_editable` (*editable_req*, *comes_from=None*, *use_pep517=None*, *isolated=False*, *options=None*, *constraint=False*, *user_supplied=False*)

`pip_shims.install_req_from_line` (*name*, *comes_from=None*, *use_pep517=None*, *isolated=False*, *options=None*, *constraint=False*, *line_source=None*, *user_supplied=False*)

Creates an InstallRequirement from a name, which might be a requirement, directory containing 'setup.py', filename, or URL.

Parameters `line_source` – An optional string describing where the line is from, for logging purposes in case of an error.

`pip_shims.install_req_from_req_string` (*req_string*, *comes_from=None*, *isolated=False*, *use_pep517=None*, *user_supplied=False*)

class `pip_shims.InstallRequirement` (*req*, *comes_from*, *editable=False*, *link=None*, *markers=None*, *use_pep517=None*, *isolated=False*, *install_options=None*, *global_options=None*, *hash_options=None*, *constraint=False*, *extras=()*, *user_supplied=False*)

Bases: `pip._internal.req.req_install.InstallRequirement`

_generate_metadata()

Invokes metadata generator functions, with the required arguments.

_get_archive_name (*path*, *parentdir*, *rootdir*)

_set_requirement()

Set requirement after generating metadata.

archive (*build_dir*)

Saves archive to provided build_dir.

Used for saving downloaded VCS requirements as part of *pip download*.

assert_source_matches_version()

build_location (*build_dir*, *autodelete*, *parallel_builds*)

check_if_exists (*use_user_site*)

Find an installed distribution that satisfies or conflicts with this requirement, and set self.satisfied_by or self.should_reinstall appropriately.

ensure_build_location (*build_dir*, *autodelete*, *parallel_builds*)

ensure_has_source_dir (*parent_dir*, *autodelete=False*, *parallel_builds=False*)

Ensure that a `source_dir` is set.

This will create a temporary build dir if the name of the requirement isn't known yet.

Parameters `parent_dir` – The ideal pip `parent_dir` for the `source_dir`. Generally `src_dir` for editables and `build_dir` for sdist.

Returns `self.source_dir`

format_debug ()

An un-tested helper for getting state, for debugging.

from_editable = `<pip_shims.utils.BaseMethod object>`

from_line = `<pip_shims.utils.BaseMethod object>`

from_path ()

Format a nice indicator to show where this “comes from”

get_dist ()

has_hash_options

Return whether any known-good hashes are specified as options.

These activate `--require-hashes` mode; hashes specified as part of a URL do not.

hashes (*trust_internet=True*)

Return a hash-comparer that considers my option- and URL-based hashes to be known-good.

Hashes in URLs—ones embedded in the requirements file, not ones downloaded from an index server—are almost peers with ones from flags. They satisfy `--require-hashes` (whether it was implicitly or explicitly activated) but do not activate it. `md5` and `sha224` are not allowed in flags, which should nudge people toward good algos. We always OR all hashes together, even ones from URLs.

Parameters `trust_internet` – Whether to trust URL-based (`#md5=...`) hashes downloaded from the internet, as by `populate_link()`

install (*install_options*, *global_options=None*, *root=None*, *home=None*, *prefix=None*, *warn_script_location=True*, *use_user_site=False*, *pycompile=True*)

installed_version

is_pinned

Return whether I am pinned to an exact version.

For example, `some-package==1.2` is pinned; `some-package>1.2` is not.

is_wheel

load_pyproject_toml ()

Load the `pyproject.toml` file.

After calling this routine, all of the attributes related to PEP 517 processing for this requirement have been set. In particular, the `use_pep517` attribute can be used to determine whether we should follow the PEP 517 or legacy (`setup.py`) code path.

match_markers (*extras_requested=None*)

metadata

name

prepare_metadata ()

Ensure that project metadata is available.

Under PEP 517, call the backend hook to prepare the metadata. Under legacy processing, call `setup.py egg-info`.

pyproject_toml_path

setup_py_path

specifier

uninstall (*auto_confirm=False, verbose=False*)

Uninstall the distribution currently satisfying this requirement.

Prompts before removing or modifying files unless `auto_confirm` is `True`.

Refuses to delete or modify files outside of `sys.prefix` - thus uninstallation within a virtual environment can only modify that virtual environment, even if the virtualenv is linked to global site-packages.

unpacked_source_directory

update_editable (*obtain=True*)

warn_on_mismatching_name ()

`pip_shims.is_archive_file` (*name*)

Return `True` if *name* is considered as an archive file.

`pip_shims.is_file_url` (*link*)

class `pip_shims.Downloader` (*session, progress_bar*)

Bases: `object`

`pip_shims.unpack_url` (*link, location, downloader, download_dir=None, hashes=None*)

Unpack link into *location*, downloading if required.

Parameters *hashes* – A Hashes object, one of whose embedded hashes must match, or `HashMismatch` will be raised. If the Hashes is empty, no matches are required, and unhashable types of requirements (like VCS ones, which would ordinarily raise `HashUnsupported`) are allowed.

`pip_shims.is_installable_dir` (*path*)

Is *path* a directory containing `setup.py` or `pyproject.toml`?

class `pip_shims.Link` (*url, comes_from=None, requires_python=None, yanked_reason=None, cache_link_parsing=True*)

Bases: `pip._internal.models.link.Link`

Parameters

- **url** – url of the resource pointed to (href of the link)
- **comes_from** – instance of `HTMLPage` where the link was found, or string.
- **requires_python** – String containing the *Requires-Python* metadata field, specified in PEP 345. This may be specified by a `data-requires-python` attribute in the HTML link tag, as described in PEP 503.
- **yanked_reason** – the reason the file has been yanked, if the file has been yanked, or `None` if the file hasn't been yanked. This is the value of the “data-yanked” attribute, if present, in a simple repository HTML link. If the file has been yanked but no reason was provided, this should be the empty string. See PEP 592 for more information and the specification.
- **cache_link_parsing** – A flag that is used elsewhere to determine whether resources retrieved from this link should be cached. PyPI index urls should generally have this set to `False`, for example.

_compare (*other, method*)

```

_compare_key
_defining_class
_egg_fragment_re = re.compile('[#&]egg=(^[&]*)')
_hash_re = re.compile('(sha1|sha224|sha384|sha256|sha512|md5)=[a-f0-9]+')
_parsed_url
_subdirectory_fragment_re = re.compile('[#&]subdirectory=(^[&]*)')
_url
cache_link_parsing
comes_from
egg_fragment
ext
file_path
filename
has_hash
hash
hash_name
is_artifact
is_existing_dir()
is_file
is_hash_allowed(hashes)
    Return True if the link has a hash and it is allowed.
is_vcs
is_wheel
is_yanked
netloc
    This can contain auth information.
path
requires_python
scheme
show_url
splittext()
subdirectory_fragment
url
url_without_fragment
yanked_reason

```

```

pip_shims.make_abstract_dist(install_req)
    Returns a Distribution for the given InstallRequirement

```

`pip_shims.make_distribution_for_install_requirement` (*install_req*)

Returns a Distribution for the given InstallRequirement

`pip_shims.make_option_group` (*group, parser*)

Return an OptionGroup object group – assumed to be dict with ‘name’ and ‘options’ keys parser – an optparse Parser

class `pip_shims.PackageFinder` (*link_collector, target_python, allow_yanked, format_control=None, candidate_prefs=None, ignore_requires_python=None*)

Bases: `object`

This finds packages.

This is meant to match easy_install’s technique for looking for packages, by reading pages and looking for appropriate links.

This constructor is primarily meant to be used by the create() class method and from tests.

Parameters

- **format_control** – A FormatControl object, used to control the selection of source packages / binary packages when consulting the index and links.
- **candidate_prefs** – Options to use when creating a CandidateEvaluator object.

`_log_skipped_link` (*link, reason*)

`_sort_links` (*links*)

Returns elements of links in order, non-egg links first, egg links second, while eliminating duplicates

allow_all_prereleases

classmethod `create` (*link_collector, selection_prefs, target_python=None*)

Create a PackageFinder.

Parameters

- **selection_prefs** – The candidate selection preferences, as a SelectionPreferences object.
- **target_python** – The target Python interpreter to use when checking compatibility. If None (the default), a TargetPython object will be constructed from the running Python.

`evaluate_links` (*link_evaluator, links*)

Convert links that are candidates to InstallationCandidate objects.

`find_all_candidates` (*project_name*)

Find all available InstallationCandidate for project_name

This checks index_urls and find_links. All versions found are returned as an InstallationCandidate list.

See LinkEvaluator.evaluate_link() for details on which files are accepted.

`find_best_candidate` (*project_name, specifier=None, hashes=None*)

Find matches for the given project and specifier.

Parameters **specifier** – An optional object implementing *filter* (e.g. `packaging.specifiers.SpecifierSet`) to filter applicable versions.

Returns A `BestCandidateResult` instance.

`find_links`

find_requirement (*req, upgrade*)

Try to find a Link matching req

Expects req, an InstallRequirement and upgrade, a boolean Returns a InstallationCandidate if found, Raises DistributionNotFound or BestVersionAlreadyInstalled otherwise

get_install_candidate (*link_evaluator, link*)

If the link is a candidate for install, convert it to an InstallationCandidate and return it. Otherwise, return None.

index_urls

make_candidate_evaluator (*project_name, specifier=None, hashes=None*)

Create a CandidateEvaluator object to use.

make_link_evaluator (*project_name*)

prefer_binary

process_project_url (*project_url, link_evaluator*)

search_scope

set_allow_all_prereleases ()

set_prefer_binary ()

target_python

trusted_hosts

```
class pip_shims.CandidateEvaluator(project_name, supported_tags, specifier, prefer_binary=False, allow_all_prereleases=False, hashes=None)
```

Bases: `object`

Responsible for filtering and sorting candidates for installation based on what tags are valid.

Parameters supported_tags – The PEP 425 tags supported by the target Python in order of preference (most preferred first).

_sort_key (*candidate*)

Function to pass as the *key* argument to a call to `sorted()` to sort InstallationCandidates by preference.

Returns a tuple such that tuples sorting as greater using Python’s default comparison operator are more preferred.

The preference is as follows:

First and foremost, candidates with allowed (matching) hashes are always preferred over candidates without matching hashes. This is because e.g. if the only candidate with an allowed hash is yanked, we still want to use that candidate.

Second, excepting hash considerations, candidates that have been yanked (in the sense of PEP 592) are always less preferred than candidates that haven’t been yanked. Then:

If not finding wheels, they are sorted by version only. If finding wheels, then the sort order is by version, then:

1. existing installs
2. wheels ordered via `Wheel.support_index_min(self._supported_tags)`
3. source archives

If `prefer_binary` was set, then all wheels are sorted above sources.

Note: it was considered to embed this logic into the Link comparison operators, but then different sdist links with the same version, would have to be considered equal

compute_best_candidate (*candidates*)

Compute and return a *BestCandidateResult* instance.

classmethod create (*project_name*, *target_python=None*, *prefer_binary=False*, *allow_all_prereleases=False*, *specifier=None*, *hashes=None*)

Create a *CandidateEvaluator* object.

Parameters

- **target_python** – The target Python interpreter to use when checking compatibility. If None (the default), a *TargetPython* object will be constructed from the running Python.
- **specifier** – An optional object implementing *filter* (e.g. *packaging.specifiers.SpecifierSet*) to filter applicable versions.
- **hashes** – An optional collection of allowed hashes.

get_applicable_candidates (*candidates*)

Return the applicable candidates from a list of candidates.

sort_best_candidate (*candidates*)

Return the best candidate per the instance’s sort order, or None if no candidate is acceptable.

class `pip_shims.CandidatePreferences` (*prefer_binary=False*, *allow_all_prereleases=False*)

Bases: `object`

Encapsulates some of the preferences for filtering and sorting *InstallationCandidate* objects.

Parameters **allow_all_prereleases** – Whether to allow all pre-releases.

class `pip_shims.LinkCollector` (*session*, *search_scope*)

Bases: `object`

Responsible for collecting *Link* objects from all configured locations, making network requests as needed.

The class’s main method is its `collect_links()` method.

collect_links (*project_name*)

Find all available links for the given project name.

Returns All the *Link* objects (unfiltered), as a *CollectedLinks* object.

classmethod create (*session*, *options*, *suppress_no_index=False*)

Parameters

- **session** – The *Session* to use to make requests.
- **suppress_no_index** – Whether to ignore the `–no-index` option when constructing the *SearchScope* object.

fetch_page (*location*)

Fetch an HTML page containing package links.

find_links

class `pip_shims.LinkEvaluator` (*project_name*, *canonical_name*, *formats*, *target_python*, *allow_yanked*, *ignore_requires_python=None*)

Bases: `object`

Responsible for evaluating links for a particular project.

Parameters

- **project_name** – The user supplied package name.
- **canonical_name** – The canonical package name.
- **formats** – The formats allowed for this package. Should be a set with ‘binary’ or ‘source’ or both in it.
- **target_python** – The target Python interpreter to use when evaluating link compatibility. This is used, for example, to check wheel compatibility, as well as when checking the Python version, e.g. the Python version embedded in a link filename (or egg fragment) and against an HTML link’s optional PEP 503 “data-requires-python” attribute.
- **allow_yanked** – Whether files marked as yanked (in the sense of PEP 592) are permitted to be candidates for install.
- **ignore_requires_python** – Whether to ignore incompatible PEP 503 “data-requires-python” values in HTML links. Defaults to False.

```
_py_version_re = re.compile('-py([123]\\.|[0-9]?)$')
```

evaluate_link (*link*)

Determine whether a link is a candidate for installation.

Returns A tuple (is_candidate, result), where *result* is (1) a version string if *is_candidate* is True, and (2) if *is_candidate* is False, an optional string to log the reason the link fails to qualify.

```
class pip_shims.TargetPython(platform=None, py_version_info=None, abi=None, implementation=None)
```

Bases: `object`

Encapsulates the properties of a Python interpreter one is targeting for a package install, download, etc.

Parameters

- **platform** – A string or None. If None, searches for packages that are supported by the current system. Otherwise, will find packages that can be built on the platform passed in. These packages will only be downloaded for distribution: they will not be built locally.
- **py_version_info** – An optional tuple of ints representing the Python version information to use (e.g. `sys.version_info[:3]`). This can have length 1, 2, or 3 when provided.
- **abi** – A string or None. This is passed to `compatibility_tags.py`’s `get_supported()` function as is.
- **implementation** – A string or None. This is passed to `compatibility_tags.py`’s `get_supported()` function as is.

_given_py_version_info

_valid_tags

abi

format_given ()

Format the given, non-None attributes for display.

get_tags ()

Return the supported PEP 425 tags to check wheel candidates against.

The tags are returned in order of preference (most preferred first).

implementation

platform

py_version

py_version_info

class pip_shims.**SearchScope** (*find_links, index_urls*)

Bases: `object`

Encapsulates the locations that pip is configured to search.

classmethod **create** (*find_links, index_urls*)

Create a SearchScope object after normalizing the *find_links*.

find_links

get_formatted_locations ()

get_index_urls_locations (*project_name*)

Returns the locations found via self.index_urls

Checks the url_name on the main (first in the list) index and use this url_name to produce all locations

index_urls

class pip_shims.**SelectionPreferences** (*allow_yanked, allow_all_prereleases=False, format_control=None, prefer_binary=False, ignore_requires_python=None*)

Bases: `object`

Encapsulates the candidate selection preferences for downloading and installing files.

Create a SelectionPreferences object.

Parameters

- **allow_yanked** – Whether files marked as yanked (in the sense of PEP 592) are permitted to be candidates for install.
- **format_control** – A FormatControl object or None. Used to control the selection of source packages / binary packages when consulting the index and links.
- **prefer_binary** – Whether to prefer an old, but valid, binary dist over a new source dist.
- **ignore_requires_python** – Whether to ignore incompatible “Requires-Python” values in links. Defaults to False.

allow_all_prereleases

allow_yanked

format_control

ignore_requires_python

prefer_binary

pip_shims.parse_requirements (*filename, session, finder=None, comes_from=None, options=None, constraint=False*)

Parse a requirements file and yield ParsedRequirement instances.

Parameters

- **filename** – Path or url of requirements file.
- **session** – PipSession instance.
- **finder** – Instance of pip.index.PackageFinder.
- **comes_from** – Origin description of requirements.
- **options** – cli options.

- **constraint** – If true, parsing a constraint file rather than requirements file.

`pip_shims.path_to_url` (*path*)

Convert a path to a file: URL. The path will be made absolute and have quoted path parts.

exception `pip_shims.PipError`

Bases: `Exception`

Base pip exception

args

with_traceback ()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class `pip_shims.RequirementPreparer` (*build_dir, download_dir, src_dir, wheel_download_dir, build_isolation, req_tracker, downloader, finder, require_hashes, use_user_site*)

Bases: `object`

Prepares a Requirement

_download_should_save

_ensure_link_req_src_dir (*req, download_dir, parallel_builds*)

Ensure source_dir of a linked InstallRequirement.

_get_linked_req_hashes (*req*)

_log_preparing_link (*req*)

Log the way the link prepared.

prepare_editable_requirement (*req*)

Prepare an editable requirement

prepare_installed_requirement (*req, skip_reason*)

Prepare an already-installed requirement

prepare_linked_requirement (*req, parallel_builds=False*)

Prepare a requirement to be obtained from req.link.

class `pip_shims.RequirementSet` (*check_supported_wheels=True*)

Bases: `object`

Create a RequirementSet.

add_named_requirement (*install_req*)

add_requirement (*install_req, parent_req_name=None, extras_requested=None*)

Add install_req as a requirement to install.

Parameters

- **parent_req_name** – The name of the requirement that needed this added. The name is used because when multiple unnamed requirements resolve to the same name, we could otherwise end up with dependency links that point outside the Requirements set. parent_req must already be added. Note that None implies that this is a user supplied requirement, vs an inferred one.
- **extras_requested** – an iterable of extras used to evaluate the environment markers.

Returns Additional requirements to scan. That is either [] if the requirement is not applicable, or [install_req] if the requirement is applicable and has just been added.

add_unnamed_requirement (*install_req*)

all_requirements

get_requirement (*name*)

has_requirement (*name*)

class pip_shims.**RequirementTracker** (*root*)

Bases: `object`

_entry_path (*link*)

add (*req*)

Add an InstallRequirement to build tracking.

cleanup ()

remove (*req*)

Remove an InstallRequirement from build tracking.

track (*req*)

class pip_shims.**TempDirectory** (*path=None, delete=<pip._internal.utils.temp_dir.Default object>, kind='temp', globally_managed=False*)

Bases: `object`

Helper class that owns and cleans up a temporary directory.

This class can be used as a context manager or as an OO representation of a temporary directory.

Attributes:

path Location to the created temporary directory

delete Whether the directory should be deleted when exiting (when used as a contextmanager)

Methods:

cleanup() Deletes the temporary directory

When used as a context manager, if the delete attribute is True, on exiting the context the temporary directory is deleted.

_create (*kind*)

Create a temporary directory and store its path in self.path

cleanup ()

Remove the temporary directory created and reset state

path

pip_shims.**global_tempdir_manager** ()

pip_shims.**shim_unpack** (*, *unpack_fn=<pip_shims.models.ShimmedPathCollection object>, download_dir, tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection object>, ireq=None, link=None, location=None, hashes=None, progress_bar='off', only_download=None, downloader_provider=<pip_shims.models.ShimmedPathCollection object>, session=None*)

Accepts all parameters that have been valid to pass to `pip._internal.download.unpack_url()` and selects or drops parameters as needed before invoking the provided callable.

Parameters

- **unpack_fn** (*Callable*) – A callable or shim referring to the pip implementation
- **download_dir** (*str*) – The directory to download the file to

- **tempdir_manager_provider** (*TShimmedFunc*) – A callable or shim referring to *global_tempdir_manager* function from pip or a shimmed no-op context manager

:param Optional[InstallRequirement] ireq: an Install Requirement instance, defaults to None

:param Optional[Link] link: A Link instance, defaults to None.

Parameters

- **location** (*Optional[str]*) – A location or source directory if the target is a VCS url, defaults to None.
- **hashes** (*Optional[Any]*) – A Hashes instance, defaults to None
- **progress_bar** (*str*) – Indicates progress par usage during download, defatuls to off.
- **only_download** (*Optional[bool]*) – Whether to skip install, defaults to None.
- **downloader_provider** (*Optional[ShimmedPathCollection]*) – A downloader class to instantiate, if applicable.
- **session** (*Optional[Session]*) – A PipSession instance, defaults to None.

Returns The result of unpacking the url.

Return type `None`

```
pip_shims.get_requirement_tracker()
```

```
class pip_shims.Resolver(preparer, finder, wheel_cache, make_install_req, use_user_site,
                        ignore_dependencies, ignore_installed, ignore_requires_python,
                        force_reinstall, upgrade_strategy, py_version_info=None)
```

Bases: `pip._internal.resolution.base.BaseResolver`

Resolves which packages need to be installed/uninstalled to perform the requested operation without breaking the requirements of any package.

```
_allowed_strategies = {'eager', 'only-if-needed', 'to-satisfy-only'}
```

```
_check_skip_installed(req_to_install)
```

Check if req_to_install should be skipped.

This will check if the req is installed, and whether we should upgrade or reinstall it, taking into account all the relevant user options.

After calling this req_to_install will only have satisfied_by set to None if the req_to_install is to be upgraded/reinstalled etc. Any other value will be a dist recording the current thing installed that satisfies the requirement.

Note that for vcs urls and the like we can't assess skipping in this routine - we simply identify that we need to pull the thing down, then later on it is pulled down and introspected to assess upgrade/ reinstalls etc.

Returns A text reason for why it was skipped, or None.

```
_find_requirement_link(req)
```

```
_get_abstract_dist_for(req)
```

Takes a InstallRequirement and returns a single AbstractDist representing a prepared variant of the same.

```
_is_upgrade_allowed(req)
```

```
_populate_link(req)
```

Ensure that if a link can be found for this, that it is found.

Note that `req.link` may still be `None` - if the requirement is already installed and not needed to be upgraded based on the return value of `_is_upgrade_allowed()`.

If `preparer.require_hashes` is `True`, don't use the wheel cache, because cached wheels, always built locally, have different hashes than the files downloaded from the index server and thus throw false hash mismatches. Furthermore, cached wheels at present have undeterministic contents due to file modification times.

`_resolve_one` (*requirement_set, req_to_install*)

Prepare a single requirements file.

Returns A list of additional `InstallRequirements` to also install.

`_set_req_to_reinstall` (*req*)

Set a requirement to be installed.

`get_installation_order` (*req_set*)

Create the installation order.

The installation order is topological - requirements are installed before the requiring thing. We break cycles at an arbitrary point, and make no other guarantees.

`resolve` (*root_reqs, check_supported_wheels*)

Resolve what operations need to be done

As a side-effect of this method, the packages (and their dependencies) are downloaded, unpacked and prepared for installation. This preparation is done by `pip.operations.prepare`.

Once PyPI has static dependency metadata available, it would be possible to move the preparation to become a step separated from dependency resolution.

class `pip_shims.SafeFileCache` (*directory*)

Bases: `pip._vendor.cachecontrol.cache.BaseCache`

A file based cache which is safe to use even when the target directory may not be accessible or writable.

`_get_cache_path` (*name*)

`close` ()

`delete` (*key*)

`get` (*key*)

`set` (*key, value*)

class `pip_shims.UninstallPathSet` (*dist*)

Bases: `object`

A set of file paths to be removed in the uninstallation of a requirement.

`_allowed_to_proceed` (*verbose*)

Display which files would be deleted and prompt for confirmation

`_permitted` (*path*)

Return `True` if the given path is one we are permitted to remove/modify, `False` otherwise.

`add` (*path*)

`add_pth` (*pth_file, entry*)

`commit` ()

Remove temporary save dir: rollback will no longer be possible.

classmethod `from_dist` (*dist*)

remove (*auto_confirm=False, verbose=False*)
Remove paths in `self.paths` with confirmation (unless `auto_confirm` is True).

rollback ()
Rollback the changes previously made by `remove()`.

`pip_shims.url_to_path(url)`
Convert a file: URL to a path.

class `pip_shims.VcsSupport`

Bases: `object`

`_registry = {'bZR': <pip._internal.vcs.bazaar.Bazaar object>, 'git': <pip._internal.vcs.git.Git object>, 'hg': <pip._internal.vcs.hg.Hg object>, 'ssh': <pip._internal.vcs.ssh.Ssh object>, 'svn': <pip._internal.vcs.svn.Svn object>, 'svn+ssh': <pip._internal.vcs.svn.Svn object>, 'vcs': <pip._internal.vcs.vcs.Vcs object>}`

`all_schemes`

`backends`

`dirnames`

`get_backend(name)`
Return a VersionControl object or None.

`get_backend_for_dir(location)`
Return a VersionControl object if a repository of that type is found at the given directory.

`get_backend_for_scheme(scheme)`
Return a VersionControl object or None.

`register(cls)`

`schemes = ['ssh', 'git', 'hg', 'bZR', 'sftp', 'svn']`

`unregister(name)`

class `pip_shims.Wheel(filename)`

Bases: `object`

`get_formatted_file_tags()`
Return the wheel's tags as a sorted list of strings.

`support_index_min(tags)`
Return the lowest index that one of the wheel's file_tag combinations achieves in the given list of supported tags.

For example, if there are 8 supported tags and one of the file tags is first in the list, then return 0.

Parameters `tags` – the PEP 425 tags to check the wheel against, in order with most preferred first.

Raises `ValueError` – If none of the wheel's file tags match one of the supported tags.

`supported(tags)`
Return whether the wheel is compatible with one of the given tags.

Parameters `tags` – the PEP 425 tags to check the wheel against.

`wheel_file_re = re.compile('^(?P<namever>(?P<name>.+?) - (?P<ver>. *?)) \n ((- (?P<build>\w+))?)$')`

class `pip_shims.WheelCache(cache_dir, format_control)`

Bases: `pip._internal.cache.Cache`

Wraps `EphemWheelCache` and `SimpleWheelCache` into a single `Cache`

This `Cache` allows for gracefully degradation, using the ephem wheel cache when a certain link is not found in the simple wheel cache first.

`_get_cache_path_parts` (*link*)

Get parts of part that must be `os.path.joined` with `cache_dir`

`_get_cache_path_parts_legacy` (*link*)

Get parts of part that must be `os.path.joined` with `cache_dir`

Legacy cache key (pip < 20) for compatibility with older caches.

`_get_candidates` (*link, canonical_package_name*)

`get` (*link, package_name, supported_tags*)

Returns a link to a cached item if it exists, otherwise returns the passed link.

`get_cache_entry` (*link, package_name, supported_tags*)

Returns a `CacheEntry` with a link to a cached item if it exists or `None`. The cache entry indicates if the item was found in the persistent or ephemeral cache.

`get_ephem_path_for_link` (*link*)

`get_path_for_link` (*link*)

Return a directory to store cached items in for link.

`get_path_for_link_legacy` (*link*)

`pip_shims.build` (*requirements, wheel_cache, build_options, global_options*)

Build wheels.

Returns The list of `InstallRequirement` that succeeded to build and the list of `InstallRequirement` that failed to build.

`pip_shims.build_one` (*req, output_dir, build_options, global_options*)

Build one wheel.

Returns The filename of the built wheel, or `None` if the build failed.

`pip_shims.build_one_inside_env` (*req, output_dir, build_options, global_options*)

class `pip_shims.AbstractDistribution` (*req*)

Bases: `object`

A base class for handling installable artifacts.

The requirements for anything installable are as follows:

- we must be able to determine the requirement name (or we can't correctly handle the non-upgrade case).
- for packages with setup requirements, we must also be able to determine their requirements without installing additional packages (for the same reason as run-time dependencies)
- we must be able to create a `Distribution` object exposing the above metadata.

`_abc_impl = <_abc_data object>`

`get_pkg_resources_distribution` ()

`prepare_distribution_metadata` (*finder, build_isolation*)

class `pip_shims.InstalledDistribution` (*req*)

Bases: `pip._internal.distributions.base.AbstractDistribution`

Represents an installed package.

This does not need any preparation as the required information has already been computed.

`_abc_impl = <_abc_data object>`

`get_pkg_resources_distribution` ()

`prepare_distribution_metadata` (*finder*, *build_isolation*)

class `pip_shims.SourceDistribution` (*req*)

Bases: `pip._internal.distributions.base.AbstractDistribution`

Represents a source distribution.

The preparation step for these needs metadata for the packages to be generated, either using PEP 517 or using the legacy `setup.py egg_info`.

`_abc_impl = <_abc_data object>`

`_setup_isolation` (*finder*)

`get_pkg_resources_distribution` ()

`prepare_distribution_metadata` (*finder*, *build_isolation*)

class `pip_shims.WheelDistribution` (*req*)

Bases: `pip._internal.distributions.base.AbstractDistribution`

Represents a wheel distribution.

This does not need any preparation as wheels can be directly unpacked.

`_abc_impl = <_abc_data object>`

`get_pkg_resources_distribution` ()

Loads the metadata from the wheel file into memory and returns a `Distribution` that uses it, not relying on the wheel file or requirement.

`prepare_distribution_metadata` (*finder*, *build_isolation*)

`pip_shims.wheel_cache` (*cache_dir=None*, *format_control=None*, ***,
wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>,
format_control_provider=<pip_shims.models.ShimmedPathCollection ob-
ject>, *tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection*
object>)

`pip_shims.get_package_finder` (*install_cmd=None*, *options=None*, *session=None*,
platform=None, *python_versions=None*, *abi=None*,
implementation=None, *target_python=None*, *ig-*
nore_requires_python=None, ***, *target_python_builder=<class*
'pip._internal.models.target_python.TargetPython'>, *in-*
stall_cmd_provider=<pip_shims.models.ShimmedPathCollection
object>)

Shim for compatibility to generate package finders.

Build and return a `PackageFinder` instance using the `InstallCommand` helper method to construct the finder, shimmed with backports as needed for compatibility.

Parameters

- **`install_cmd_provider`** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **`install_cmd`** – A `InstallCommand` instance which is used to generate the finder.
- **`options`** (*optparse.Values*) – An optional `optparse.Values` instance generated by calling `install_cmd.parser.parse_args()` typically.
- **`session`** – An optional session instance, can be created by the `install_cmd`.
- **`platform`** (*Optional[str]*) – An optional platform string, e.g. `linux_x86_64`

- `..]] python_versions` (*Optional[Tuple[str, ...]*) – A tuple of 2-digit strings representing python versions, e.g. (“27”, “35”, “36”, “37”...)
- `abi` (*Optional[str]*) – The target abi to support, e.g. “cp38”
- `implementation` (*Optional[str]*) – An optional implementation string for limiting searches to a specific implementation, e.g. “cp” or “py”
- `target_python` – A `TargetPython` instance (will be translated to alternate arguments if necessary on incompatible pip versions).
- `ignore_requires_python` (*Optional[bool]*) – Whether to ignore `requires_python` on resulting candidates, only valid after pip version 19.3.1
- `target_python_builder` – A ‘`TargetPython`’ builder (e.g. the class itself, uninstantiated)

Returns A `pip._internal.index.package_finder.PackageFinder` instance

Return type `pip._internal.index.package_finder.PackageFinder`

Example

```
>>> from pip_shims.shims import InstallCommand, get_package_finder
>>> install_cmd = InstallCommand()
>>> finder = get_package_finder(
...     install_cmd, python_versions=("27", "35", "36", "37", "38"),
...     implementation="
cp"
... )
>>> candidates = finder.find_all_candidates("requests")
>>> requests_222 = next(iter(c for c in candidates if c.version.public == "2.22.0
... ))
>>> requests_222
<InstallationCandidate('requests', <Version('2.22.0')>, <Link https://files.
pythonhos
ted.org/packages/51/bd/
23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/r
equests-2.22.0-py2.py3-none-any.whl
#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9
a590f48c010551dc6c4b31 (from https://pypi.org/simple/requests/) (requires-python:>
=2.
7, !=3.0.*, !=3.1.*, !=3.2.*, !=3.3.*, !=3.4.*)>>>
```

```
pip_shims.make_preparer(*, preparer_fn=<pip_shims.models.ShimmedPathCollection object>,
                        req_tracker_fn=<pip_shims.models.ShimmedPathCollection object>,
                        build_dir=None, src_dir=None, download_dir=None,
                        wheel_download_dir=None, progress_bar='off',
                        build_isolation=False, session=None, finder=None, options=None,
                        require_hashes=None, use_user_site=None, req_tracker=None,
                        install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>,
                        downloader_provider=<pip_shims.models.ShimmedPathCollection object>,
                        install_cmd=None, finder_provider=<pip_shims.models.ShimmedPathCollection object>)
```

Creates a requirement preparer for preparing pip requirements.

Provides a compatibility shim that accepts all previously valid arguments and discards any that are no longer used.

Raises

- **TypeError** – No requirement tracker provided and one cannot be generated
- **TypeError** – No valid sessions provided and one cannot be generated
- **TypeError** – No valid finders provided and one cannot be generated

Parameters

- **preparer_fn** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **req_tracker_fn** (*Optional[TShimmedFunc]*) – Callable or shim for generating requirement trackers, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **src_dir** (*Optional[str]*) – Directory to find or extract source files, defaults to None
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **progress_bar** (*str*) – Whether to display a progress bar, defaults to off
- **build_isolation** (*bool*) – Whether to build requirements in isolation, defaults to False
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **require_hashes** (*Optional[bool]*) – Whether to require hashes for preparation
- **use_user_site** (*Optional[bool]*) – Whether to use the user site directory for preparing requirements
- **TShimmedFunc]] req_tracker** (*Optional[Union[TReqTracker,])*) – The requirement tracker to use for building packages, defaults to None
- **downloader_provider** (*Optional[TShimmedFunc]*) – A downloader provider
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None
- **finder_provider** (*Optional[TShimmedFunc]*) – A package finder provider

Yield A new requirement preparer instance

Return type ContextManager[RequirementPreparer]

Example

```
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_tracker
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
```

(continues on next page)

(continued from previous page)

```

...     install_cmd, session=session, options=pip_options
... )
>>> with make_preparer(
...     options=pip_options, finder=finder, session=session, install_cmd=ic
... ) as preparer:
...     print(preparer)
<pip._internal.operations.prepare.RequirementPreparer object at 0x7f8a2734be80>

```

```

pip_shims.get_resolver(*, resolver_fn=<pip_shims.models.ShimmedPathCollection object>,
install_req_provider=<pip_shims.models.ShimmedPathCollection object>,
format_control_provider=<pip_shims.models.ShimmedPathCollection object>,
wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>,
finder=None, upgrade_strategy='to-satisfy-only',
force_reinstall=None, ignore_dependencies=None, ignore_requires_python=None,
ignore_installed=True, use_user_site=False, isolated=None, wheel_cache=None,
preparer=None, session=None, options=None, make_install_req=None,
install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>,
install_cmd=None, use_pep517=True)

```

A resolver creation compatibility shim for generating a resolver.

Consumes any argument that was previously used to instantiate a resolver, discards anything that is no longer valid.

Note: This is only valid for **pip >= 10.0.0**

Raises

- **ValueError** – A session is required but not provided and one cannot be created
- **ValueError** – A finder is required but not provided and one cannot be created
- **ValueError** – An install requirement provider is required and has not been provided

Parameters

- **resolver_fn** (*TShimmedFunc*) – The resolver function used to create new resolver instances.
- **install_req_provider** (*TShimmedFunc*) – The provider function to use to generate install requirements if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a format_control instance if needed.
- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to only-if-needed.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to None
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to None

- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to None
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to True
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to False
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to None
- **wheel_cache** (*Optional[WheelCache]*) – The wheel cache to use, defaults to None
- **preparer** (*Optional[TPreparer]*) – The requirement preparer to use, defaults to None
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **make_install_req** (*Optional[functools.partial]*) – The partial function to pass in to the resolver for actually generating install requirements, if necessary
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None.
- **use_pep517** (*bool*) – Whether to use the pep517 build process.

Returns A new resolver instance.

Return type Resolver

Example

```
>>> import os
>>> from tempdir import TemporaryDirectory
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_
↳ tracker,
...     get_resolver, InstallRequirement, RequirementSet
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> wheel_cache = WheelCache(USER_CACHE_DIR, FormatControl(None, None))
>>> with TemporaryDirectory() as temp_base:
...     reqset = RequirementSet()
...     ireq = InstallRequirement.from_line("requests")
...     ireq.is_direct = True
...     build_dir = os.path.join(temp_base, "build")
...     src_dir = os.path.join(temp_base, "src")
...     ireq.build_location(build_dir)
...     with make_preparer(
...         options=pip_options, finder=finder, session=session,
...         build_dir=build_dir, install_cmd=install_cmd,
```

(continues on next page)

(continued from previous page)

```

...     ) as preparer:
...         resolver = get_resolver(
...             finder=finder, ignore_dependencies=False, ignore_requires_
↪python=True,
...             preparer=preparer, session=session, options=pip_options,
...             install_cmd=install_cmd, wheel_cache=wheel_cache,
...         )
...         resolver.require_hashes = False
...         reqset.add_requirement(ireq)
...         results = resolver.resolve(reqset)
...         #reqset.cleanup_files()
...         for result_req in reqset.requirements:
...             print(result_req)
requests
chardet
certifi
urllib3
idna

```

`pip_shims.get_requirement_set` (*install_command=None*, *, *req_set_provider=<pip_shims.models.ShimmedPathCollection object>*, *build_dir=None*, *src_dir=None*, *download_dir=None*, *wheel_download_dir=None*, *session=None*, *wheel_cache=None*, *upgrade=False*, *upgrade_strategy=None*, *ignore_installed=False*, *ignore_dependencies=False*, *force_reinstall=False*, *use_user_site=False*, *isolated=False*, *ignore_requires_python=False*, *require_hashes=None*, *cache_dir=None*, *options=None*, *install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>*, *wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>*)

Creates a requirement set from the supplied parameters.

Not all parameters are passed through for all pip versions, but any invalid parameters will be ignored if they are not needed to generate a requirement set on the current pip version.

:param *ShimmedPathCollection* wheel_cache_provider: A context manager provider which resolves to a *WheelCache* instance

Parameters *install_command* – A *InstallCommand* instance which is used to generate the finder.

:param *ShimmedPathCollection* req_set_provider: A provider to build requirement set instances.

Parameters

- **build_dir** (*str*) – The directory to build requirements in. Removed in pip 10, defaults to None
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*str*) – The directory to download requirement artifacts to. Removed in pip 10, defaults to None
- **wheel_download_dir** (*str*) – The directory to download wheels to. Removed in pip 10, defaults to None

:param Session session: The pip session to use. Removed in pip 10, defaults to None

Parameters

- **wheel_cache** (*WheelCache*) – The pip WheelCache instance to use for caching wheels. Removed in pip 10, defaults to None
- **upgrade** (*bool*) – Whether to try to upgrade existing requirements. Removed in pip 10, defaults to False.
- **upgrade_strategy** (*str*) – The upgrade strategy to use, e.g. “only-if-needed”. Removed in pip 10, defaults to None.
- **ignore_installed** (*bool*) – Whether to ignore installed packages when resolving. Removed in pip 10, defaults to False.
- **ignore_dependencies** (*bool*) – Whether to ignore dependencies of requirements when resolving. Removed in pip 10, defaults to False.
- **force_reinstall** (*bool*) – Whether to force reinstall of packages when resolving. Removed in pip 10, defaults to False.
- **use_user_site** (*bool*) – Whether to use user site packages when resolving. Removed in pip 10, defaults to False.
- **isolated** (*bool*) – Whether to resolve in isolation. Removed in pip 10, defaults to False.
- **ignore_requires_python** (*bool*) – Removed in pip 10, defaults to False.
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **options** (*Values*) – An *Values* instance from an install cmd
- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.

Returns A new requirement set instance

Return type RequirementSet

```
pip_shims.resolve(ireq, *, reqset_provider=<pip_shims.models.ShimmedPathCollection object>,
                 req_tracker_provider=<pip_shims.models.ShimmedPathCollection object>,
                 install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>,
                 install_command=None, finder_provider=<pip_shims.models.ShimmedPathCollection
                 object>, resolver_provider=<pip_shims.models.ShimmedPathCollection object>,
                 wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>,
                 format_control_provider=<pip_shims.models.ShimmedPathCollection object>,
                 make_preparer_provider=<pip_shims.models.ShimmedPathCollection object>,
                 tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection object>,
                 options=None, session=None, resolver=None, finder=None, upgrade_strategy='to-
                 satisfy-only', force_reinstall=None, ignore_dependencies=None, ig-
                 nore_requires_python=None, ignore_installed=True, use_user_site=False,
                 isolated=None, build_dir=None, source_dir=None, download_dir=None,
                 cache_dir=None, wheel_download_dir=None, wheel_cache=None, re-
                 quire_hashes=None, check_supported_wheels=True)
```

Resolves the provided **InstallRequirement**, returning a dictionary.

Maps a dictionary of names to corresponding *InstallRequirement* values.

:param InstallRequirement ireq: An *InstallRequirement* to initiate the resolution process

:param ShimmedPathCollection reqset_provider: A *provider* to build requirement set instances.

:param *ShimmedPathCollection* req_tracker_provider: A provider to build requirement tracker instances

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_command** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None.

:param *ShimmedPathCollection* finder_provider: A provider to package finder instances.

:param *ShimmedPathCollection* resolver_provider: A provider to build resolver instances

Parameters

- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a format_control instance if needed.
- **make_preparer_provider** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **tempdir_manager_provider** (*Optional[TShimmedFunc]*) – Shim for generating tempdir manager for pip temporary directories
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None

:param Resolver resolver: A pre-existing resolver instance to use for resolution

Parameters

- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to None
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to None
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to None
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to True
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to False
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None

- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **cache_dir** (*str*) – The cache directory to use for caching artifacts during resolution
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **wheel_cache** (*Optional[WheelCache]*) – The wheel cache to use, defaults to None
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **check_supported_wheels** (*bool*) – Whether to check support of wheels before including them in resolution.

Returns A dictionary mapping requirements to corresponding `class:~pip._internal.req.req_install.InstallRequirement`'s

Return type `InstallRequirement`

Example

```
>>> from pip_shims.shims import resolve, InstallRequirement
>>> ireq = InstallRequirement.from_line("requests>=2.20")
>>> results = resolve(ireq)
>>> for k, v in results.items():
...     print("{0}: {1!r}".format(k, v))
requests: <InstallRequirement object: requests>=2.20 from https://files.
↳pythonhosted.
org/packages/51/bd/23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/
↳reque
sts-2.22.0-py2.py3-none-any.whl
↳#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9a590
f48c010551dc6c4b31 editable=False>
idna: <InstallRequirement object: idna<2.9,>=2.5 from https://files.pythonhosted.
↳org/
packages/14/2c/cd551d81dbe15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-
↳2.8-
py2.py3-none-any.whl
↳#sha256=ea8b7f6188e6fa117537c3df7da9fc686d485087abf6ac197f9c46432
f7e4a3c (from requests>=2.20) editable=False>
urllib3: <InstallRequirement object: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 from
↳htt
↳htt
ps://files.pythonhosted.org/packages/b4/40/
↳a9837291310ee1ccc242ceb6ebfd9eb21539649f19
3a7c8c86ba15b98539/urllib3-1.25.7-py2.py3-none-any.whl
↳#sha256=a8a318824cc77d1fd4b2bec
2ded92646630d7fe8619497b142c84a9e6f5a7293 (from requests>=2.20) editable=False>
chardet: <InstallRequirement object: chardet<3.1.0,>=3.0.2 from https://files.
↳pythonh
osted.org/packages/bc/a9/
↳01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8
/chardet-3.0.4-py2.py3-none-any.whl
↳#sha256=fc323ffcaeaed0e0a02bf4d117757b98aed530d9ed
4531e3e15460124c106691 (from requests>=2.20) editable=False>
```

(continues on next page)

(continued from previous page)

```
certifi: <InstallRequirement object: certifi>=2017.4.17 from https://files.
↳pythonhost
ed.org/packages/18/b0/
↳8146a4f8dd402f60744fa380bc73ca47303ccccf8b9190fd16a827281eac2/ce
rtifi-2019.9.11-py2.py3-none-any.whl
↳#sha256=fd7c7c74727ddcf00e9acd26bba8da604ffec95bf
1c2144e67aff7a8b50e6cef (from requests>=2.20) editable=False>
```

```
pip_shims.build_wheel(req=None, reqset=None, output_dir=None, preparer=None,
wheel_cache=None, build_options=None, global_options=None,
check_binary_allowed=None, no_clean=False, session=None,
finder=None, install_command=None, req_tracker=None, build_dir=None,
src_dir=None, download_dir=None, wheel_download_dir=None,
cache_dir=None, use_user_site=False, use_pep517=None, *, for-
mat_control_provider=<pip_shims.models.ShimmedPathCollection object>,
wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>,
preparer_provider=<pip_shims.models.ShimmedPathCollection object>,
wheel_builder_provider=<pip_shims.models.ShimmedPathCollection ob-
ject>, build_one_provider=<pip_shims.models.ShimmedPathCollection ob-
ject>, build_one_inside_env_provider=<pip_shims.models.ShimmedPathCollection
object>, build_many_provider=<pip_shims.models.ShimmedPathCollection
object>, install_command_provider=<pip_shims.models.ShimmedPathCollection
object>, finder_provider=None, reqset_provider=<pip_shims.models.ShimmedPathCollection
object>)
```

Build a wheel or a set of wheels

Raises `TypeError` – Raised when no requirements are provided

Parameters

- **req** (*Optional*[*TInstallRequirement*]) – An *InstallRequirement* to build
- **reqset** (*Optional*[*TReqSet*]) – A *RequirementSet* instance (*pip*<10) or an iterable of *InstallRequirement* instances (*pip*>=10) to build
- **output_dir** (*Optional*[*str*]) – Target output directory, only useful when building one wheel using *pip*>=20.0
- **preparer** (*Optional*[*TPreparer*]) – A preparer instance, defaults to None
- **wheel_cache** (*Optional*[*TWheelCache*]) – A wheel cache instance, defaults to None
- **build_options** (*Optional*[*List*[*str*]]) – A list of build options to pass in
- **global_options** (*Optional*[*List*[*str*]]) – A list of global options to pass in
- **bool]] check_binary_allowed** (*Optional*[*Callable*[*TInstallRequirement*,]]) – A callable to check whether we are allowed to build and cache wheels for an ireq
- **no_clean** (*bool*) – Whether to avoid cleaning up wheels
- **session** (*Optional*[*TSession*]) – A *PipSession* instance to pass to create a *finder* if necessary
- **finder** (*Optional*[*TFinder*]) – A *PackageFinder* instance to use for generating a *WheelBuilder* instance on *pip*<20
- **install_command** (*Optional*[*TCommandInstance*]) – The install command used to create the finder, session, and options if needed, defaults to None.

- **req_tracker** (*Optional*[*TReqTracker*]) – An optional requirement tracker instance, if one already exists
- **build_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **src_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **download_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **wheel_download_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **cache_dir** (*Optional*[*str*]) – Passthrough cache directory for wheel cache options
- **use_user_site** (*bool*) – Whether to use the user site directory when preparing install requirements on *pip*<20
- **use_pep517** (*Optional*[*bool*]) – When set to *True* or *False*, prefers building with or without pep517 as specified, otherwise uses requirement preference. Only works for single requirements.
- **format_control_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *FormatControl* class
- **wheel_cache_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *WheelCache* class
- **preparer_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *RequirementPreparer* class
- **wheel_builder_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *WheelBuilder* class, if it exists
- **build_one_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *_build_one* function, if it exists
- **build_one_inside_env_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *_build_one_inside_env* function, if it exists
- **build_many_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *build* function, if it exists
- **install_command_provider** (*Optional*[*TShimmedFunc*]) – A shim for providing new install command instances
- **finder_provider** (*TShimmedFunc*) – A provider to package finder instances
- **reqset_provider** (*TShimmedFunc*) – A provider for requirement set generation

Returns A tuple of successful and failed install requirements or else a path to a wheel

Return type `Optional[Union[str, Tuple[List[TInstallRequirement], List[TInstallRequirement]]]]`

Main module with magic self-replacement mechanisms to handle import speedups.

`pip_shims.shims._strip_extras` (*path*)

class `pip_shims.shims.SessionCommandMixin`

Bases: `pip._internal.cli.command_context.CommandContextMixin`

A class mixin for command classes needing `_build_session()`.

`_build_session` (*options*, *retries=None*, *timeout=None*)

classmethod `_get_index_urls` (*options*)

Return a list of index urls from user-provided options.

`enter_context` (*context_provider*)

`get_default_session` (*options*)

Get a default-managed session.

`main_context` ()

class `pip_shims.shims.Command` (*name='Default pip command.'*, *summary='PipCommand'*, *isolated='Default pip command.'*)

Bases: `pip._internal.cli.base_command.Command`, `pip._internal.cli.req_command.SessionCommandMixin`

`_build_session` (*options*, *retries=None*, *timeout=None*)

classmethod `_get_index_urls` (*options*)

Return a list of index urls from user-provided options.

`_main` (*args*)

`add_options` ()

`enter_context` (*context_provider*)

`get_default_session` (*options*)

Get a default-managed session.

handle_pip_version_check (*options*)

This is a no-op so that commands by default do not do the pip version check.

ignore_require_venv = **False**

main (*args*)

main_context ()

parse_args (*args*)

run (*options, args*)

usage = **None**

class pip_shims.shims.**ConfigOptionParser** (**args, **kwargs*)

Bases: pip._internal.cli.parser.CustomOptionParser

Custom option parser which updates its defaults by checking the configuration files and environmental variables

_add_help_option ()

_add_version_option ()

_check_conflict (*option*)

_create_option_list ()

_create_option_mappings ()

_get_all_options ()

_get_args (*args*)

_get_ordered_configuration_items ()

_init_parsing_state ()

_match_long_opt (*opt : string*) → string

Determine which long option string 'opt' matches, ie. which one it is an unambiguous abbreviation for.

Raises BadOptionError if 'opt' doesn't unambiguously match any long option string.

_populate_option_list (*option_list, add_help=True*)

_process_args (*largs, rargs, values*)

_process_args(*largs* [[string],] *rargs* : [string], *values* : Values)

Process command-line arguments and populate 'values', consuming options and arguments from 'rargs'.

If 'allow_interspersed_args' is false, stop at the first non-option argument. If true, accumulate any interspersed non-option arguments in 'largs'.

_process_long_opt (*rargs, values*)

_process_short_opts (*rargs, values*)

_share_option_mappings (*parser*)

_update_defaults (*defaults*)

Updates the given defaults with values from the config files and the environ. Does a little special handling for certain types of options (lists).

add_option (*Option*)

add_option(*opt_str, ..., kwarg=val, ...*)

add_option_group (**args, **kwargs*)

add_options (*option_list*)

check_default (*option, key, val*)

check_values (*values : Values, args : [string]*)

-> (*values : Values, args : [string]*)

Check that the supplied option values and leftover arguments are valid. Returns the option values and leftover arguments (possibly adjusted, possibly completely new – whatever you like). Default implementation just returns the passed-in values; subclasses may override as desired.

destroy ()

Declare that you are done with this OptionParser. This cleans up reference cycles so the OptionParser (and all objects referenced by it) can be garbage-collected promptly. After calling destroy(), the OptionParser is unusable.

disable_interspersed_args ()

Set parsing to stop on the first non-option. Use this if you have a command processor which runs another command that has options of its own and you want to make sure these options don't get confused.

enable_interspersed_args ()

Set parsing to not stop on the first non-option, allowing interspersing switches with command arguments. This is the default behavior. See also disable_interspersed_args() and the class documentation description of the attribute allow_interspersed_args.

error (*msg : string*)

Print a usage message incorporating 'msg' to stderr and exit. If you override this in a subclass, it should not return – it should either exit or raise an exception.

exit (*status=0, msg=None*)

expand_prog_name (*s*)

format_description (*formatter*)

format_epilog (*formatter*)

format_help (*formatter=None*)

format_option_help (*formatter=None*)

get_default_values ()

Overriding to make updating the defaults after instantiation of the option parser possible, _update_defaults() does the dirty work.

get_description ()

get_option (*opt_str*)

get_option_group (*opt_str*)

get_prog_name ()

get_usage ()

get_version ()

has_option (*opt_str*)

insert_option_group (*idx, *args, **kwargs*)

Insert an OptionGroup at a given position.

option_list_all

Get a list of all options, including those in option groups.

parse_args (*args=None, values=None*)

parse_args(args [[string] = sys.argv[1:], values : Values = None)

-> (values : Values, args : [string])

Parse the command-line options found in 'args' (default: sys.argv[1:]). Any errors result in a call to 'error()', which by default prints the usage message to stderr and calls sys.exit() with an error message. On success returns a pair (values, args) where 'values' is a Values instance (with all your option values) and 'args' is the list of arguments left over after parsing options.

print_help (file : file = stdout)

Print an extended help message, listing all options and any help text provided with them, to 'file' (default stdout).

print_usage (file : file = stdout)

Print the usage message for the current program (self.usage) to 'file' (default stdout). Any occurrence of the string "%prog" in self.usage is replaced with the name of the current program (basename of sys.argv[0]). Does nothing if self.usage is empty or not defined.

print_version (file : file = stdout)

Print the version message for this program (self.version) to 'file' (default stdout). As with print_usage(), any occurrence of "%prog" in self.version is replaced by the current program's name. Does nothing if self.version is empty or undefined.

remove_option (opt_str)

set_conflict_handler (handler)

set_default (dest, value)

set_defaults (**kwargs)

set_description (description)

set_process_default_values (process)

set_usage (usage)

standard_option_list = []

exception pip_shims.shims.DistributionNotFound

Bases: pip._internal.exceptions.InstallationError

Raised when a distribution cannot be found to satisfy a requirement

args

with_traceback ()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class pip_shims.shims.FormatControl (no_binary=None, only_binary=None)

Bases: object

Helper for managing formats from which a package can be installed.

disallow_binaries ()

get_allowed_formats (canonical_name)

static handle_mutual_excludes (value, target, other)

no_binary

only_binary

class pip_shims.shims.FrozenRequirement (name, req, editable, comments=())

Bases: object

classmethod `from_dist` (*dist*)

```
pip_shims.shims.get_installed_distributions(local_only=True, skip={'argparse',
                                         'python', 'wsgiref'}, include_editables=True,
                                         editables_only=False, user_only=False,
                                         paths=None)
```

Return a list of installed Distribution objects.

If `local_only` is `True` (default), only return installations local to the current virtualenv, if in a virtualenv.

`skip` argument is an iterable of lower-case project names to ignore; defaults to `stdlib_pkgs`

If `include_editables` is `False`, don't report editables.

If `editables_only` is `True`, only report editables.

If `user_only` is `True`, only report installations in the user site directory.

If `paths` is set, only report the distributions present at the specified list of locations.

exception `pip_shims.shims.InstallationError`

Bases: `pip._internal.exceptions.PipError`

General exception during installation

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.shims.UninstallationError`

Bases: `pip._internal.exceptions.PipError`

General exception during uninstallation

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.shims.RequirementsFileParseError`

Bases: `pip._internal.exceptions.InstallationError`

Raised when a general error occurs parsing a requirements file line.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.shims.BestVersionAlreadyInstalled`

Bases: `pip._internal.exceptions.PipError`

Raised when the most up-to-date version of a package is already installed.

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception `pip_shims.shims.BadCommand`

Bases: `pip._internal.exceptions.PipError`

Raised when virtualenv or a command is not found

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pip_shims.shims.CommandError

Bases: pip._internal.exceptions.PipError

Raised when there is an error in command-line arguments

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception pip_shims.shims.PreviousBuildDirError

Bases: pip._internal.exceptions.PipError

Raised when there's a previous conflicting build directory

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

pip_shims.shims.install_req_from_editable(*editable_req*, *comes_from=None*,
use_pep517=None, *isolated=False*,
options=None, *constraint=False*,
user_supplied=False)

pip_shims.shims.install_req_from_line(*name*, *comes_from=None*, *use_pep517=None*,
isolated=False, *options=None*, *constraint=False*,
line_source=None, *user_supplied=False*)

Creates an InstallRequirement from a name, which might be a requirement, directory containing 'setup.py', filename, or URL.

Parameters *line_source* – An optional string describing where the line is from, for logging purposes in case of an error.

pip_shims.shims.install_req_from_req_string(*req_string*, *comes_from=None*, *iso-*
lated=False, *use_pep517=None*,
user_supplied=False)

class pip_shims.shims.InstallRequirement(*req*, *comes_from*, *editable=False*, *link=None*,
markers=None, *use_pep517=None*, *iso-*
lated=False, *install_options=None*,
global_options=None, *hash_options=None*, *con-*
straint=False, *extras=()*, *user_supplied=False*)

Bases: pip._internal.req.req_install.InstallRequirement

_generate_metadata()

Invokes metadata generator functions, with the required arguments.

_get_archive_name(*path*, *parentdir*, *rootdir*)

_set_requirement()

Set requirement after generating metadata.

archive(*build_dir*)

Saves archive to provided build_dir.

Used for saving downloaded VCS requirements as part of *pip* download.

assert_source_matches_version()

build_location(*build_dir*, *autodelete*, *parallel_builds*)

check_if_exists (*use_user_site*)

Find an installed distribution that satisfies or conflicts with this requirement, and set `self.satisfied_by` or `self.should_reinstall` appropriately.

ensure_build_location (*build_dir, autodelete, parallel_builds*)

ensure_has_source_dir (*parent_dir, autodelete=False, parallel_builds=False*)

Ensure that a `source_dir` is set.

This will create a temporary build dir if the name of the requirement isn't known yet.

Parameters `parent_dir` – The ideal pip `parent_dir` for the `source_dir`. Generally `src_dir` for editables and `build_dir` for sdist.

Returns `self.source_dir`

format_debug ()

An un-tested helper for getting state, for debugging.

from_editable = `<pip_shims.utils.BaseMethod object>`

from_line = `<pip_shims.utils.BaseMethod object>`

from_path ()

Format a nice indicator to show where this “comes from”

get_dist ()

has_hash_options

Return whether any known-good hashes are specified as options.

These activate `--require-hashes` mode; hashes specified as part of a URL do not.

hashes (*trust_internet=True*)

Return a hash-comparer that considers my option- and URL-based hashes to be known-good.

Hashes in URLs—ones embedded in the requirements file, not ones downloaded from an index server—are almost peers with ones from flags. They satisfy `--require-hashes` (whether it was implicitly or explicitly activated) but do not activate it. `md5` and `sha224` are not allowed in flags, which should nudge people toward good algos. We always OR all hashes together, even ones from URLs.

Parameters `trust_internet` – Whether to trust URL-based (`#md5=...`) hashes downloaded from the internet, as by `populate_link()`

install (*install_options, global_options=None, root=None, home=None, prefix=None, warn_script_location=True, use_user_site=False, pycompile=True*)

installed_version

is_pinned

Return whether I am pinned to an exact version.

For example, `some-package==1.2` is pinned; `some-package>1.2` is not.

is_wheel

load_pyproject_toml ()

Load the `pyproject.toml` file.

After calling this routine, all of the attributes related to PEP 517 processing for this requirement have been set. In particular, the `use_pep517` attribute can be used to determine whether we should follow the PEP 517 or legacy (`setup.py`) code path.

match_markers (*extras_requested=None*)

metadata

name

prepare_metadata()

Ensure that project metadata is available.

Under PEP 517, call the backend hook to prepare the metadata. Under legacy processing, call `setup.py egg-info`.

pyproject_toml_path

setup_py_path

specifier

uninstall (*auto_confirm=False, verbose=False*)

Uninstall the distribution currently satisfying this requirement.

Prompts before removing or modifying files unless `auto_confirm` is `True`.

Refuses to delete or modify files outside of `sys.prefix` - thus uninstallation within a virtual environment can only modify that virtual environment, even if the virtualenv is linked to global site-packages.

unpacked_source_directory

update_editable (*obtain=True*)

warn_on_mismatching_name()

`pip_shims.shims.is_archive_file(name)`

Return `True` if `name` is considered as an archive file.

`pip_shims.shims.is_file_url(link)`

class `pip_shims.shims.Downloader` (*session, progress_bar*)

Bases: `object`

`pip_shims.shims.unpack_url(link, location, downloader, download_dir=None, hashes=None)`

Unpack link into location, downloading if required.

Parameters `hashes` – A Hashes object, one of whose embedded hashes must match, or `HashMismatch` will be raised. If the Hashes is empty, no matches are required, and unhashable types of requirements (like VCS ones, which would ordinarily raise `HashUnsupported`) are allowed.

`pip_shims.shims.is_installable_dir(path)`

Is path a directory containing `setup.py` or `pyproject.toml`?

class `pip_shims.shims.Link` (*url, comes_from=None, requires_python=None, yanked_reason=None, cache_link_parsing=True*)

Bases: `pip._internal.models.link.Link`

Parameters

- **url** – url of the resource pointed to (href of the link)
- **comes_from** – instance of `HTMLPage` where the link was found, or string.
- **requires_python** – String containing the *Requires-Python* metadata field, specified in PEP 345. This may be specified by a `data-requires-python` attribute in the HTML link tag, as described in PEP 503.
- **yanked_reason** – the reason the file has been yanked, if the file has been yanked, or `None` if the file hasn't been yanked. This is the value of the “data-yanked” attribute, if present, in a simple repository HTML link. If the file has been yanked but no reason was provided, this should be the empty string. See PEP 592 for more information and the specification.

- **cache_link_parsing** – A flag that is used elsewhere to determine whether resources retrieved from this link should be cached. PyPI index urls should generally have this set to False, for example.

```
_compare (other, method)  
_compare_key  
_defining_class  
_egg_fragment_re = re.compile('[#&]egg=(^[&]*)')  
_hash_re = re.compile('(sha1|sha224|sha384|sha256|sha512|md5)=[a-f0-9]+')  
_parsed_url  
_subdirectory_fragment_re = re.compile('[#&]subdirectory=(^[&]*)')  
_url  
cache_link_parsing  
comes_from  
egg_fragment  
ext  
file_path  
filename  
has_hash  
hash  
hash_name  
is_artifact  
is_existing_dir()  
is_file  
is_hash_allowed (hashes)  
    Return True if the link has a hash and it is allowed.  
is_vcs  
is_wheel  
is_yanked  
netloc  
    This can contain auth information.  
path  
requires_python  
scheme  
show_url  
splittext()  
subdirectory_fragment  
url
```

url_without_fragment

yanked_reason

`pip_shims.shims.make_abstract_dist` (*install_req*)

Returns a Distribution for the given InstallRequirement

`pip_shims.shims.make_distribution_for_install_requirement` (*install_req*)

Returns a Distribution for the given InstallRequirement

`pip_shims.shims.make_option_group` (*group, parser*)

Return an OptionGroup object group – assumed to be dict with ‘name’ and ‘options’ keys parser – an optparse Parser

class `pip_shims.shims.PackageFinder` (*link_collector, target_python, allow_yanked, format_control=None, candidate_prefs=None, ignore_requires_python=None*)

Bases: `object`

This finds packages.

This is meant to match easy_install’s technique for looking for packages, by reading pages and looking for appropriate links.

This constructor is primarily meant to be used by the create() class method and from tests.

Parameters

- **format_control** – A FormatControl object, used to control the selection of source packages / binary packages when consulting the index and links.
- **candidate_prefs** – Options to use when creating a CandidateEvaluator object.

`_log_skipped_link` (*link, reason*)

`_sort_links` (*links*)

Returns elements of links in order, non-egg links first, egg links second, while eliminating duplicates

allow_all_prereleases

classmethod `create` (*link_collector, selection_prefs, target_python=None*)

Create a PackageFinder.

Parameters

- **selection_prefs** – The candidate selection preferences, as a SelectionPreferences object.
- **target_python** – The target Python interpreter to use when checking compatibility. If None (the default), a TargetPython object will be constructed from the running Python.

`evaluate_links` (*link_evaluator, links*)

Convert links that are candidates to InstallationCandidate objects.

`find_all_candidates` (*project_name*)

Find all available InstallationCandidate for project_name

This checks index_urls and find_links. All versions found are returned as an InstallationCandidate list.

See LinkEvaluator.evaluate_link() for details on which files are accepted.

`find_best_candidate` (*project_name, specifier=None, hashes=None*)

Find matches for the given project and specifier.

Parameters specifier – An optional object implementing *filter* (e.g. `packaging.specifiers.SpecifierSet`) to filter applicable versions.

Returns A *BestCandidateResult* instance.

find_links

find_requirement (*req, upgrade*)

Try to find a Link matching req

Expects req, an InstallRequirement and upgrade, a boolean Returns a InstallationCandidate if found, Raises DistributionNotFound or BestVersionAlreadyInstalled otherwise

get_install_candidate (*link_evaluator, link*)

If the link is a candidate for install, convert it to an InstallationCandidate and return it. Otherwise, return None.

index_urls

make_candidate_evaluator (*project_name, specifier=None, hashes=None*)

Create a CandidateEvaluator object to use.

make_link_evaluator (*project_name*)

prefer_binary

process_project_url (*project_url, link_evaluator*)

search_scope

set_allow_all_prereleases ()

set_prefer_binary ()

target_python

trusted_hosts

```
class pip_shims.shims.CandidateEvaluator(project_name, supported_tags, specifier, prefer_binary=False, allow_all_prereleases=False, hashes=None)
```

Bases: `object`

Responsible for filtering and sorting candidates for installation based on what tags are valid.

Parameters supported_tags – The PEP 425 tags supported by the target Python in order of preference (most preferred first).

_sort_key (*candidate*)

Function to pass as the *key* argument to a call to sorted() to sort InstallationCandidates by preference.

Returns a tuple such that tuples sorting as greater using Python’s default comparison operator are more preferred.

The preference is as follows:

First and foremost, candidates with allowed (matching) hashes are always preferred over candidates without matching hashes. This is because e.g. if the only candidate with an allowed hash is yanked, we still want to use that candidate.

Second, excepting hash considerations, candidates that have been yanked (in the sense of PEP 592) are always less preferred than candidates that haven’t been yanked. Then:

If not finding wheels, they are sorted by version only. If finding wheels, then the sort order is by version, then:

1. existing installs
2. wheels ordered via `Wheel.support_index_min(self._supported_tags)`

3. source archives

If `prefer_binary` was set, then all wheels are sorted above sources.

Note: it was considered to embed this logic into the Link comparison operators, but then different sdist links with the same version, would have to be considered equal

compute_best_candidate (*candidates*)

Compute and return a *BestCandidateResult* instance.

classmethod create (*project_name*, *target_python=None*, *prefer_binary=False*, *allow_all_prereleases=False*, *specifier=None*, *hashes=None*)

Create a *CandidateEvaluator* object.

Parameters

- **target_python** – The target Python interpreter to use when checking compatibility. If None (the default), a *TargetPython* object will be constructed from the running Python.
- **specifier** – An optional object implementing *filter* (e.g. *packaging.specifiers.SpecifierSet*) to filter applicable versions.
- **hashes** – An optional collection of allowed hashes.

get_applicable_candidates (*candidates*)

Return the applicable candidates from a list of candidates.

sort_best_candidate (*candidates*)

Return the best candidate per the instance’s sort order, or None if no candidate is acceptable.

class `pip_shims.shims.CandidatePreferences` (*prefer_binary=False*, *allow_all_prereleases=False*)

Bases: `object`

Encapsulates some of the preferences for filtering and sorting *InstallationCandidate* objects.

Parameters `allow_all_prereleases` – Whether to allow all pre-releases.

class `pip_shims.shims.LinkCollector` (*session*, *search_scope*)

Bases: `object`

Responsible for collecting *Link* objects from all configured locations, making network requests as needed.

The class’s main method is its `collect_links()` method.

collect_links (*project_name*)

Find all available links for the given project name.

Returns All the *Link* objects (unfiltered), as a *CollectedLinks* object.

classmethod create (*session*, *options*, *suppress_no_index=False*)

Parameters

- **session** – The *Session* to use to make requests.
- **suppress_no_index** – Whether to ignore the `–no-index` option when constructing the *SearchScope* object.

fetch_page (*location*)

Fetch an HTML page containing package links.

find_links

class `pip_shims.shims.LinkEvaluator` (*project_name*, *canonical_name*, *formats*, *target_python*, *allow_yanked*, *ignore_requires_python=None*)

Bases: `object`

Responsible for evaluating links for a particular project.

Parameters

- **project_name** – The user supplied package name.
- **canonical_name** – The canonical package name.
- **formats** – The formats allowed for this package. Should be a set with ‘binary’ or ‘source’ or both in it.
- **target_python** – The target Python interpreter to use when evaluating link compatibility. This is used, for example, to check wheel compatibility, as well as when checking the Python version, e.g. the Python version embedded in a link filename (or egg fragment) and against an HTML link’s optional PEP 503 “data-requires-python” attribute.
- **allow_yanked** – Whether files marked as yanked (in the sense of PEP 592) are permitted to be candidates for install.
- **ignore_requires_python** – Whether to ignore incompatible PEP 503 “data-requires-python” values in HTML links. Defaults to False.

```
_py_version_re = re.compile('-py([123]\\.|[0-9]?)$')
```

evaluate_link (*link*)

Determine whether a link is a candidate for installation.

Returns A tuple (is_candidate, result), where *result* is (1) a version string if *is_candidate* is True, and (2) if *is_candidate* is False, an optional string to log the reason the link fails to qualify.

```
class pip_shims.shims.TargetPython (platform=None, py_version_info=None, abi=None, implementation=None)
```

Bases: `object`

Encapsulates the properties of a Python interpreter one is targeting for a package install, download, etc.

Parameters

- **platform** – A string or None. If None, searches for packages that are supported by the current system. Otherwise, will find packages that can be built on the platform passed in. These packages will only be downloaded for distribution: they will not be built locally.
- **py_version_info** – An optional tuple of ints representing the Python version information to use (e.g. `sys.version_info[:3]`). This can have length 1, 2, or 3 when provided.
- **abi** – A string or None. This is passed to `compatibility_tags.py`’s `get_supported()` function as is.
- **implementation** – A string or None. This is passed to `compatibility_tags.py`’s `get_supported()` function as is.

_given_py_version_info

_valid_tags

abi

format_given ()

Format the given, non-None attributes for display.

get_tags ()

Return the supported PEP 425 tags to check wheel candidates against.

The tags are returned in order of preference (most preferred first).

implementation

platform

py_version

py_version_info

class pip_shims.shims.**SearchScope** (*find_links, index_urls*)

Bases: `object`

Encapsulates the locations that pip is configured to search.

classmethod **create** (*find_links, index_urls*)

Create a SearchScope object after normalizing the *find_links*.

find_links

get_formatted_locations ()

get_index_urls_locations (*project_name*)

Returns the locations found via self.index_urls

Checks the url_name on the main (first in the list) index and use this url_name to produce all locations

index_urls

class pip_shims.shims.**SelectionPreferences** (*allow_yanked, allow_all_prereleases=False, format_control=None, prefer_binary=False, ignore_requires_python=None*)

Bases: `object`

Encapsulates the candidate selection preferences for downloading and installing files.

Create a SelectionPreferences object.

Parameters

- **allow_yanked** – Whether files marked as yanked (in the sense of PEP 592) are permitted to be candidates for install.
- **format_control** – A FormatControl object or None. Used to control the selection of source packages / binary packages when consulting the index and links.
- **prefer_binary** – Whether to prefer an old, but valid, binary dist over a new source dist.
- **ignore_requires_python** – Whether to ignore incompatible “Requires-Python” values in links. Defaults to False.

allow_all_prereleases

allow_yanked

format_control

ignore_requires_python

prefer_binary

pip_shims.shims.parse_requirements (*filename, session, finder=None, comes_from=None, options=None, constraint=False*)

Parse a requirements file and yield ParsedRequirement instances.

Parameters

- **filename** – Path or url of requirements file.
- **session** – PipSession instance.
- **finder** – Instance of pip.index.PackageFinder.

- **comes_from** – Origin description of requirements.
- **options** – cli options.
- **constraint** – If true, parsing a constraint file rather than requirements file.

`pip_shims.shims.path_to_url` (*path*)

Convert a path to a file: URL. The path will be made absolute and have quoted path parts.

exception `pip_shims.shims.PipError`

Bases: `Exception`

Base pip exception

args

with_traceback ()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class `pip_shims.shims.RequirementPreparer` (*build_dir, download_dir, src_dir, wheel_download_dir, build_isolation, req_tracker, downloader, finder, require_hashes, use_user_site*)

Bases: `object`

Prepares a Requirement

_download_should_save

_ensure_link_req_src_dir (*req, download_dir, parallel_builds*)

Ensure source_dir of a linked InstallRequirement.

_get_linked_req_hashes (*req*)

_log_preparing_link (*req*)

Log the way the link prepared.

prepare_editable_requirement (*req*)

Prepare an editable requirement

prepare_installed_requirement (*req, skip_reason*)

Prepare an already-installed requirement

prepare_linked_requirement (*req, parallel_builds=False*)

Prepare a requirement to be obtained from req.link.

class `pip_shims.shims.RequirementSet` (*check_supported_wheels=True*)

Bases: `object`

Create a RequirementSet.

add_named_requirement (*install_req*)

add_requirement (*install_req, parent_req_name=None, extras_requested=None*)

Add install_req as a requirement to install.

Parameters

- **parent_req_name** – The name of the requirement that needed this added. The name is used because when multiple unnamed requirements resolve to the same name, we could otherwise end up with dependency links that point outside the Requirements set. parent_req must already be added. Note that None implies that this is a user supplied requirement, vs an inferred one.
- **extras_requested** – an iterable of extras used to evaluate the environment markers.

Returns Additional requirements to scan. That is either [] if the requirement is not applicable, or [install_req] if the requirement is applicable and has just been added.

add_unnamed_requirement (*install_req*)

all_requirements

get_requirement (*name*)

has_requirement (*name*)

class pip_shims.shims.**RequirementTracker** (*root*)

Bases: `object`

_entry_path (*link*)

add (*req*)

Add an InstallRequirement to build tracking.

cleanup ()

remove (*req*)

Remove an InstallRequirement from build tracking.

track (*req*)

class pip_shims.shims.**TempDirectory** (*path=None, delete=<pip._internal.utils.temp_dir._Default object>, kind='temp', globally_managed=False*)

Bases: `object`

Helper class that owns and cleans up a temporary directory.

This class can be used as a context manager or as an OO representation of a temporary directory.

Attributes:

path Location to the created temporary directory

delete Whether the directory should be deleted when exiting (when used as a contextmanager)

Methods:

cleanup() Deletes the temporary directory

When used as a context manager, if the delete attribute is True, on exiting the context the temporary directory is deleted.

_create (*kind*)

Create a temporary directory and store its path in self.path

cleanup ()

Remove the temporary directory created and reset state

path

pip_shims.shims.**global_tempdir_manager** ()

pip_shims.shims.**shim_unpack** (*, *unpack_fn=<pip_shims.models.ShimmedPathCollection object>, download_dir, tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection object>, ireq=None, link=None, location=None, hashes=None, progress_bar='off', only_download=None, downloader_provider=<pip_shims.models.ShimmedPathCollection object>, session=None*)

Accepts all parameters that have been valid to pass to `pip._internal.download.unpack_url()` and selects or drops parameters as needed before invoking the provided callable.

Parameters

- **unpack_fn** (*Callable*) – A callable or shim referring to the pip implementation
- **download_dir** (*str*) – The directory to download the file to
- **tempdir_manager_provider** (*TShimmedFunc*) – A callable or shim referring to *global_tempdir_manager* function from pip or a shimmed no-op context manager

:param Optional[InstallRequirement] ireq: an Install Requirement instance, defaults to None

:param Optional[Link] link: A Link instance, defaults to None.

Parameters

- **location** (*Optional[str]*) – A location or source directory if the target is a VCS url, defaults to None.
- **hashes** (*Optional[Any]*) – A Hashes instance, defaults to None
- **progress_bar** (*str*) – Indicates progress par usage during download, defaults to off.
- **only_download** (*Optional[bool]*) – Whether to skip install, defaults to None.
- **downloader_provider** (*Optional[ShimmedPathCollection]*) – A downloader class to instantiate, if applicable.
- **session** (*Optional[Session]*) – A PipSession instance, defaults to None.

Returns The result of unpacking the url.

Return type `None`

```
pip_shims.shims.get_requirement_tracker()
```

```
class pip_shims.shims.Resolver(preparer, finder, wheel_cache, make_install_req, use_user_site,
                               ignore_dependencies, ignore_installed, ignore_requires_python,
                               force_reinstall, upgrade_strategy, py_version_info=None)
```

Bases: `pip._internal.resolution.base.BaseResolver`

Resolves which packages need to be installed/uninstalled to perform the requested operation without breaking the requirements of any package.

```
_allowed_strategies = {'eager', 'only-if-needed', 'to-satisfy-only'}
```

```
_check_skip_installed(req_to_install)
```

Check if `req_to_install` should be skipped.

This will check if the req is installed, and whether we should upgrade or reinstall it, taking into account all the relevant user options.

After calling this `req_to_install` will only have `satisfied_by` set to `None` if the `req_to_install` is to be upgraded/reinstalled etc. Any other value will be a dist recording the current thing installed that satisfies the requirement.

Note that for vcs urls and the like we can't assess skipping in this routine - we simply identify that we need to pull the thing down, then later on it is pulled down and introspected to assess upgrade/ reinstalls etc.

Returns A text reason for why it was skipped, or `None`.

```
_find_requirement_link(req)
```

```
_get_abstract_dist_for(req)
```

Takes a `InstallRequirement` and returns a single `AbstractDist` representing a prepared variant of the same.

```
_is_upgrade_allowed(req)
```

`__populate_link` (*req*)

Ensure that if a link can be found for this, that it is found.

Note that `req.link` may still be `None` - if the requirement is already installed and not needed to be upgraded based on the return value of `__is_upgrade_allowed()`.

If `preparer.require_hashes` is `True`, don't use the wheel cache, because cached wheels, always built locally, have different hashes than the files downloaded from the index server and thus throw false hash mismatches. Furthermore, cached wheels at present have undeterministic contents due to file modification times.

`__resolve_one` (*requirement_set, req_to_install*)

Prepare a single requirements file.

Returns A list of additional `InstallRequirements` to also install.

`__set_req_to_reinstall` (*req*)

Set a requirement to be installed.

`get_installation_order` (*req_set*)

Create the installation order.

The installation order is topological - requirements are installed before the requiring thing. We break cycles at an arbitrary point, and make no other guarantees.

`resolve` (*root_reqs, check_supported_wheels*)

Resolve what operations need to be done

As a side-effect of this method, the packages (and their dependencies) are downloaded, unpacked and prepared for installation. This preparation is done by `pip.operations.prepare`.

Once PyPI has static dependency metadata available, it would be possible to move the preparation to become a step separated from dependency resolution.

class `pip_shims.shims.SafeFileCache` (*directory*)

Bases: `pip._vendor.cachecontrol.cache.BaseCache`

A file based cache which is safe to use even when the target directory may not be accessible or writable.

`__get_cache_path` (*name*)

`close` ()

`delete` (*key*)

`get` (*key*)

`set` (*key, value*)

class `pip_shims.shims.UninstallPathSet` (*dist*)

Bases: `object`

A set of file paths to be removed in the uninstallation of a requirement.

`__allowed_to_proceed` (*verbose*)

Display which files would be deleted and prompt for confirmation

`__permitted` (*path*)

Return `True` if the given path is one we are permitted to remove/modify, `False` otherwise.

`add` (*path*)

`add_pt` (*pth_file, entry*)

`commit` ()

Remove temporary save dir: rollback will no longer be possible.

```

classmethod from_dist (dist)

remove (auto_confirm=False, verbose=False)
    Remove paths in self.paths with confirmation (unless auto_confirm is True).

rollback ()
    Rollback the changes previously made by remove().

pip_shims.shims.url_to_path (url)
    Convert a file: URL to a path.

class pip_shims.shims.VcsSupport
    Bases: object

    _registry = {'bzzr': <pip._internal.vcs.bazaar.Bazaar object>, 'git': <pip._internal.vcs.git.Git object>, 'hg': <pip._internal.vcs.hg.Hg object>, 'ssh': <pip._internal.vcs.ssh.Ssh object>, 'svn': <pip._internal.vcs.svn.Svn object>}

    all_schemes

    backends

    dirnames

    get_backend (name)
        Return a VersionControl object or None.

    get_backend_for_dir (location)
        Return a VersionControl object if a repository of that type is found at the given directory.

    get_backend_for_scheme (scheme)
        Return a VersionControl object or None.

    register (cls)

    schemes = ['ssh', 'git', 'hg', 'bzzr', 'sftp', 'svn']

    unregister (name)

class pip_shims.shims.Wheel (filename)
    Bases: object

    get_formatted_file_tags ()
        Return the wheel's tags as a sorted list of strings.

    support_index_min (tags)
        Return the lowest index that one of the wheel's file_tag combinations achieves in the given list of supported tags.

        For example, if there are 8 supported tags and one of the file tags is first in the list, then return 0.

        Parameters tags – the PEP 425 tags to check the wheel against, in order with most preferred first.

        Raises ValueError – If none of the wheel's file tags match one of the supported tags.

    supported (tags)
        Return whether the wheel is compatible with one of the given tags.

        Parameters tags – the PEP 425 tags to check the wheel against.

    wheel_file_re = re.compile('^(?P<namever>(?!<name>.+)-(?P<ver>.*?))\n ((-?(?P<build>\d+)?

class pip_shims.shims.WheelCache (cache_dir, format_control)
    Bases: pip._internal.cache.Cache

    Wraps EphemWheelCache and SimpleWheelCache into a single Cache

```

This Cache allows for gracefully degradation, using the ephem wheel cache when a certain link is not found in the simple wheel cache first.

`_get_cache_path_parts` (*link*)

Get parts of part that must be `os.path.join`d with `cache_dir`

`_get_cache_path_parts_legacy` (*link*)

Get parts of part that must be `os.path.join`d with `cache_dir`

Legacy cache key (pip < 20) for compatibility with older caches.

`_get_candidates` (*link, canonical_package_name*)

`get` (*link, package_name, supported_tags*)

Returns a link to a cached item if it exists, otherwise returns the passed link.

`get_cache_entry` (*link, package_name, supported_tags*)

Returns a `CacheEntry` with a link to a cached item if it exists or `None`. The cache entry indicates if the item was found in the persistent or ephemeral cache.

`get_ephem_path_for_link` (*link*)

`get_path_for_link` (*link*)

Return a directory to store cached items in for link.

`get_path_for_link_legacy` (*link*)

`pip_shims.shims.build` (*requirements, wheel_cache, build_options, global_options*)

Build wheels.

Returns The list of `InstallRequirement` that succeeded to build and the list of `InstallRequirement` that failed to build.

`pip_shims.shims.build_one` (*req, output_dir, build_options, global_options*)

Build one wheel.

Returns The filename of the built wheel, or `None` if the build failed.

`pip_shims.shims.build_one_inside_env` (*req, output_dir, build_options, global_options*)

class `pip_shims.shims.AbstractDistribution` (*req*)

Bases: `object`

A base class for handling installable artifacts.

The requirements for anything installable are as follows:

- we must be able to determine the requirement name (or we can't correctly handle the non-upgrade case).
- for packages with setup requirements, we must also be able to determine their requirements without installing additional packages (for the same reason as run-time dependencies)
- we must be able to create a `Distribution` object exposing the above metadata.

`_abc_impl = <_abc_data object>`

`get_pkg_resources_distribution` ()

`prepare_distribution_metadata` (*finder, build_isolation*)

class `pip_shims.shims.InstalledDistribution` (*req*)

Bases: `pip._internal.distributions.base.AbstractDistribution`

Represents an installed package.

This does not need any preparation as the required information has already been computed.

```
_abc_impl = <_abc_data object>
get_pkg_resources_distribution()
prepare_distribution_metadata(finder, build_isolation)
```

```
class pip_shims.shims.SourceDistribution(req)
Bases: pip._internal.distributions.base.AbstractDistribution
```

Represents a source distribution.

The preparation step for these needs metadata for the packages to be generated, either using PEP 517 or using the legacy `setup.py egg_info`.

```
_abc_impl = <_abc_data object>
_setup_isolation(finder)
get_pkg_resources_distribution()
prepare_distribution_metadata(finder, build_isolation)
```

```
class pip_shims.shims.WheelDistribution(req)
Bases: pip._internal.distributions.base.AbstractDistribution
```

Represents a wheel distribution.

This does not need any preparation as wheels can be directly unpacked.

```
_abc_impl = <_abc_data object>
get_pkg_resources_distribution()
    Loads the metadata from the wheel file into memory and returns a Distribution that uses it, not relying on
    the wheel file or requirement.
prepare_distribution_metadata(finder, build_isolation)
```

```
pip_shims.shims.wheel_cache(cache_dir=None, format_control=None, *,
    wheel_cache_provider=<pip_shims.models.ShimmedPathCollection
    object>, format_control_provider=<pip_shims.models.ShimmedPathCollection
    object>, tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection
    object>)
```

```
pip_shims.shims.get_package_finder(install_cmd=None, options=None, session=None,
    platform=None, python_versions=None,
    abi=None, implementation=None, target_python=None,
    *, target_python_builder=<class
    'pip._internal.models.target_python.TargetPython'>, in-
    stall_cmd_provider=<pip_shims.models.ShimmedPathCollection
    object>)
```

Shim for compatibility to generate package finders.

Build and return a `PackageFinder` instance using the `InstallCommand` helper method to construct the finder, shimmed with backports as needed for compatibility.

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_cmd** – A `InstallCommand` instance which is used to generate the finder.
- **options** (*optparse.Values*) – An optional `optparse.Values` instance generated by calling `install_cmd.parser.parse_args()` typically.

- **session** – An optional session instance, can be created by the `install_cmd`.
- **platform** (*Optional[str]*) – An optional platform string, e.g. `linux_x86_64`
- **python_versions** (*Optional[Tuple[str, ...]]*) – A tuple of 2-digit strings representing python versions, e.g. (“27”, “35”, “36”, “37”...)
- **abi** (*Optional[str]*) – The target abi to support, e.g. “cp38”
- **implementation** (*Optional[str]*) – An optional implementation string for limiting searches to a specific implementation, e.g. “cp” or “py”
- **target_python** – A `TargetPython` instance (will be translated to alternate arguments if necessary on incompatible pip versions).
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore `requires_python` on resulting candidates, only valid after pip version 19.3.1
- **target_python_builder** – A ‘`TargetPython`’ builder (e.g. the class itself, uninstantiated)

Returns A `pip._internal.index.package_finder.PackageFinder` instance

Return type `pip._internal.index.package_finder.PackageFinder`

Example

```
>>> from pip_shims.shims import InstallCommand, get_package_finder
>>> install_cmd = InstallCommand()
>>> finder = get_package_finder(
...     install_cmd, python_versions=("27", "35", "36", "37", "38"),
...     implementation="
cp"
... )
>>> candidates = finder.find_all_candidates("requests")
>>> requests_222 = next(iter(c for c in candidates if c.version.public == "2.22.0
... ))
>>> requests_222
<InstallationCandidate('requests', <Version('2.22.0')>, <Link https://files.
pythonhos
ted.org/packages/51/bd/
23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/r
equests-2.22.0-py2.py3-none-any.whl
#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9
a590f48c010551dc6c4b31 (from https://pypi.org/simple/requests/) (requires-python:>
=2.
7, !=3.0.*, !=3.1.*, !=3.2.*, !=3.3.*, !=3.4.*)>>>
```

`pip_shims.shims.make_preparer` (*, `preparer_fn=<pip_shims.models.ShimmedPathCollection object>`, `req_tracker_fn=<pip_shims.models.ShimmedPathCollection object>`, `build_dir=None`, `src_dir=None`, `download_dir=None`, `wheel_download_dir=None`, `progress_bar='off'`, `build_isolation=False`, `session=None`, `finder=None`, `options=None`, `require_hashes=None`, `use_user_site=None`, `req_tracker=None`, `install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>`, `downloader_provider=<pip_shims.models.ShimmedPathCollection object>`, `install_cmd=None`, `finder_provider=<pip_shims.models.ShimmedPathCollection object>`)

Creates a requirement preparer for preparing pip requirements.

Provides a compatibility shim that accepts all previously valid arguments and discards any that are no longer used.

Raises

- **TypeError** – No requirement tracker provided and one cannot be generated
- **TypeError** – No valid sessions provided and one cannot be generated
- **TypeError** – No valid finders provided and one cannot be generated

Parameters

- **preparer_fn** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **req_tracker_fn** (*Optional[TShimmedFunc]*) – Callable or shim for generating requirement trackers, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **src_dir** (*Optional[str]*) – Directory to find or extract source files, defaults to None
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **progress_bar** (*str*) – Whether to display a progress bar, defaults to off
- **build_isolation** (*bool*) – Whether to build requirements in isolation, defaults to False
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **require_hashes** (*Optional[bool]*) – Whether to require hashes for preparation
- **use_user_site** (*Optional[bool]*) – Whether to use the user site directory for preparing requirements
- **TShimmedFunc**] **req_tracker** (*Optional[Union[TReqTracker,])*) – The requirement tracker to use for building packages, defaults to None
- **downloader_provider** (*Optional[TShimmedFunc]*) – A downloader provider
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None
- **finder_provider** (*Optional[TShimmedFunc]*) – A package finder provider

Yield A new requirement preparer instance

Return type ContextManager[RequirementPreparer]

Example

```
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_tracker
... )
```

(continues on next page)

(continued from previous page)

```
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> with make_preparer(
...     options=pip_options, finder=finder, session=session, install_cmd=ic
... ) as preparer:
...     print(preparer)
<pip._internal.operations.prepare.RequirementPreparer object at 0x7f8a2734be80>
```

```
pip_shims.shims.get_resolver(*, resolver_fn=<pip_shims.models.ShimmedPathCollection object>, install_req_provider=<pip_shims.models.ShimmedPathCollection object>, format_control_provider=<pip_shims.models.ShimmedPathCollection object>, wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>, finder=None, upgrade_strategy='to-satisfy-only', force_reinstall=None, ignore_dependencies=None, ignore_requires_python=None, ignore_installed=True, use_user_site=False, isolated=None, wheel_cache=None, preparer=None, session=None, options=None, make_install_req=None, install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>, install_cmd=None, use_pep517=True)
```

A resolver creation compatibility shim for generating a resolver.

Consumes any argument that was previously used to instantiate a resolver, discards anything that is no longer valid.

Note: This is only valid for **pip >= 10.0.0**

Raises

- **ValueError** – A session is required but not provided and one cannot be created
- **ValueError** – A finder is required but not provided and one cannot be created
- **ValueError** – An install requirement provider is required and has not been provided

Parameters

- **resolver_fn** (*TShimmedFunc*) – The resolver function used to create new resolver instances.
- **install_req_provider** (*TShimmedFunc*) – The provider function to use to generate install requirements if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a format_control instance if needed.
- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to only-if-needed.

- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to None
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to None
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to None
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to True
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to False
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to None
- **wheel_cache** (*Optional[WheelCache]*) – The wheel cache to use, defaults to None
- **preparer** (*Optional[TPreparer]*) – The requirement preparer to use, defaults to None
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **make_install_req** (*Optional[functools.partial]*) – The partial function to pass in to the resolver for actually generating install requirements, if necessary
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None.
- **use_pep517** (*bool*) – Whether to use the pep517 build process.

Returns A new resolver instance.

Return type Resolver

Example

```
>>> import os
>>> from tempdir import TemporaryDirectory
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_
↳ tracker,
...     get_resolver, InstallRequirement, RequirementSet
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> wheel_cache = WheelCache(USER_CACHE_DIR, FormatControl(None, None))
>>> with TemporaryDirectory() as temp_base:
...     reqset = RequirementSet()
...     ireq = InstallRequirement.from_line("requests")
...     ireq.is_direct = True
...     build_dir = os.path.join(temp_base, "build")
```

(continues on next page)

(continued from previous page)

```

...     src_dir = os.path.join(temp_base, "src")
...     ireq.build_location(build_dir)
...     with make_preparer(
...         options=pip_options, finder=finder, session=session,
...         build_dir=build_dir, install_cmd=install_cmd,
...     ) as preparer:
...         resolver = get_resolver(
...             finder=finder, ignore_dependencies=False, ignore_requires_
↪python=True,
...             preparer=preparer, session=session, options=pip_options,
...             install_cmd=install_cmd, wheel_cache=wheel_cache,
...         )
...         resolver.require_hashes = False
...         reqset.add_requirement(ireq)
...         results = resolver.resolve(reqset)
...         #reqset.cleanup_files()
...         for result_req in reqset.requirements:
...             print(result_req)
requests
chardet
certifi
urllib3
idna

```

```

pip_shims.shims.get_requirement_set(install_command=None, *,
                                   req_set_provider=<pip_shims.models.ShimmedPathCollection
                                   object>, build_dir=None, src_dir=None, down-
                                   load_dir=None, wheel_download_dir=None, ses-
                                   sion=None, wheel_cache=None, upgrade=False,
                                   upgrade_strategy=None, ignore_installed=False,
                                   ignore_dependencies=False, force_reinstall=False,
                                   use_user_site=False, isolated=False, ig-
                                   nore_requires_python=False, require_hashes=None,
                                   cache_dir=None, options=None, in-
                                   stall_cmd_provider=<pip_shims.models.ShimmedPathCollection
                                   object>, wheel_cache_provider=<pip_shims.models.ShimmedPathCollection
                                   object>)

```

Creates a requirement set from the supplied parameters.

Not all parameters are passed through for all pip versions, but any invalid parameters will be ignored if they are not needed to generate a requirement set on the current pip version.

:param *ShimmedPathCollection* wheel_cache_provider: A context manager provider which resolves to a *WheelCache* instance

Parameters install_command – A *InstallCommand* instance which is used to generate the finder.

:param *ShimmedPathCollection* req_set_provider: A provider to build requirement set instances.

Parameters

- **build_dir** (*str*) – The directory to build requirements in. Removed in pip 10, defaults to None
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None

- **download_dir** (*str*) – The directory to download requirement artifacts to. Removed in pip 10, defaults to None
- **wheel_download_dir** (*str*) – The directory to download wheels to. Removed in pip 10, defaults to None

:param Session session: The pip session to use. Removed in pip 10, defaults to None

Parameters

- **wheel_cache** (*WheelCache*) – The pip WheelCache instance to use for caching wheels. Removed in pip 10, defaults to None
- **upgrade** (*bool*) – Whether to try to upgrade existing requirements. Removed in pip 10, defaults to False.
- **upgrade_strategy** (*str*) – The upgrade strategy to use, e.g. “only-if-needed”. Removed in pip 10, defaults to None.
- **ignore_installed** (*bool*) – Whether to ignore installed packages when resolving. Removed in pip 10, defaults to False.
- **ignore_dependencies** (*bool*) – Whether to ignore dependencies of requirements when resolving. Removed in pip 10, defaults to False.
- **force_reinstall** (*bool*) – Whether to force reinstall of packages when resolving. Removed in pip 10, defaults to False.
- **use_user_site** (*bool*) – Whether to use user site packages when resolving. Removed in pip 10, defaults to False.
- **isolated** (*bool*) – Whether to resolve in isolation. Removed in pip 10, defaults to False.
- **ignore_requires_python** (*bool*) – Removed in pip 10, defaults to False.
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **options** (*Values*) – An *Values* instance from an install cmd
- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.

Returns A new requirement set instance

Return type RequirementSet

```

pip_shims.shims.resolve(ireq, *, reqset_provider=<pip_shims.models.ShimmedPathCollection object>, req_tracker_provider=<pip_shims.models.ShimmedPathCollection object>, install_cmd_provider=<pip_shims.models.ShimmedPathCollection object>, install_command=None, finder_provider=<pip_shims.models.ShimmedPathCollection object>, resolver_provider=<pip_shims.models.ShimmedPathCollection object>, wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>, format_control_provider=<pip_shims.models.ShimmedPathCollection object>, make_preparer_provider=<pip_shims.models.ShimmedPathCollection object>, tempdir_manager_provider=<pip_shims.models.ShimmedPathCollection object>, options=None, session=None, resolver=None, finder=None, upgrade_strategy='to-satisfy-only', force_reinstall=None, ignore_dependencies=None, ignore_requires_python=None, ignore_installed=True, use_user_site=False, isolated=None, build_dir=None, source_dir=None, download_dir=None, cache_dir=None, wheel_download_dir=None, wheel_cache=None, require_hashes=None, check_supported_wheels=True)

```

Resolves the provided **InstallRequirement**, returning a dictionary.

Maps a dictionary of names to corresponding `InstallRequirement` values.

:param InstallRequirement ireq: An `InstallRequirement` to initiate the resolution process

:param *ShimmedPathCollection* reqset_provider: A provider to build requirement set instances.

:param *ShimmedPathCollection* req_tracker_provider: A provider to build requirement tracker instances

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_command** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None.

:param *ShimmedPathCollection* finder_provider: A provider to package finder instances.

:param *ShimmedPathCollection* resolver_provider: A provider to build resolver instances

Parameters

- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a `format_control` instance if needed.
- **make_preparer_provider** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **tempdir_manager_provider** (*Optional[TShimmedFunc]*) – Shim for generating `tempdir` manager for pip temporary directories
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None

:param Resolver resolver: A pre-existing resolver instance to use for resolution

Parameters

- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to None
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to None
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to None
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to True
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to False
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **cache_dir** (*str*) – The cache directory to use for caching artifacts during resolution
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **wheel_cache** (*Optional[TWheelCache]*) – The wheel cache to use, defaults to None
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **check_supported_wheels** (*bool*) – Whether to check support of wheels before including them in resolution.

Returns A dictionary mapping requirements to corresponding `~pip._internal.req.req_install.InstallRequirement`'s

Return type `InstallRequirement`

Example

```
>>> from pip_shims.shims import resolve, InstallRequirement
>>> ireq = InstallRequirement.from_line("requests>=2.20")
>>> results = resolve(ireq)
>>> for k, v in results.items():
...     print("{0}: {1!r}".format(k, v))
requests: <InstallRequirement object: requests>=2.20 from https://files.
↳pythonhosted.
org/packages/51/bd/23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/
↳reque
sts-2.22.0-py2.py3-none-any.whl
↳#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9a590
```

(continues on next page)

(continued from previous page)

```
f48c010551dc6c4b31 editable=False>
idna: <InstallRequirement object: idna<2.9,>=2.5 from https://files.pythonhosted.
↳org/
packages/14/2c/cd551d81dbe15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-
↳2.8-
py2.py3-none-any.whl
↳#sha256=ea8b7f6188e6fa117537c3df7da9fc686d485087abf6ac197f9c46432
f7e4a3c (from requests>=2.20) editable=False>
urllib3: <InstallRequirement object: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 from_
↳htt
ps://files.pythonhosted.org/packages/b4/40/
↳a9837291310ee1ccc242ceb6ebfd9eb21539649f19
3a7c8c86ba15b98539/urllib3-1.25.7-py2.py3-none-any.whl
↳#sha256=a8a318824cc77d1fd4b2bec
2ded92646630d7fe8619497b142c84a9e6f5a7293 (from requests>=2.20) editable=False>
chardet: <InstallRequirement object: chardet<3.1.0,>=3.0.2 from https://files.
↳pythonh
sted.org/packages/bc/a9/
↳01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8
/chardet-3.0.4-py2.py3-none-any.whl
↳#sha256=fc323ffcaeaed0e0a02bf4d117757b98aed530d9ed
4531e3e15460124c106691 (from requests>=2.20) editable=False>
certifi: <InstallRequirement object: certifi>=2017.4.17 from https://files.
↳pythonhost
ed.org/packages/18/b0/
↳8146a4f8dd402f60744fa380bc73ca47303cccfc8b9190fd16a827281eac2/ce
rtifi-2019.9.11-py2.py3-none-any.whl
↳#sha256=fd7c7c74727ddcf00e9acd26bba8da604ffec95bf
1c2144e67aff7a8b50e6cef (from requests>=2.20) editable=False>
```

`pip_shims.shims.build_wheel` (*req=None, reqset=None, output_dir=None, preparer=None, wheel_cache=None, build_options=None, global_options=None, check_binary_allowed=None, no_clean=False, session=None, finder=None, install_command=None, req_tracker=None, build_dir=None, src_dir=None, download_dir=None, wheel_download_dir=None, cache_dir=None, use_user_site=False, use_pep517=None, *, format_control_provider=<pip_shims.models.ShimmedPathCollection object>, wheel_cache_provider=<pip_shims.models.ShimmedPathCollection object>, preparer_provider=<pip_shims.models.ShimmedPathCollection object>, wheel_builder_provider=<pip_shims.models.ShimmedPathCollection object>, build_one_provider=<pip_shims.models.ShimmedPathCollection object>, build_one_inside_env_provider=<pip_shims.models.ShimmedPathCollection object>, build_many_provider=<pip_shims.models.ShimmedPathCollection object>, install_command_provider=<pip_shims.models.ShimmedPathCollection object>, finder_provider=None, reqset_provider=<pip_shims.models.ShimmedPathCollection object>*)

Build a wheel or a set of wheels

Raises `TypeError` – Raised when no requirements are provided

Parameters

- **req** (*Optional* [`TInstallRequirement`]) – An `InstallRequirement` to build
- **reqset** (*Optional* [`TReqSet`]) – A `RequirementSet` instance (`pip<10`) or an iterable

of *InstallRequirement* instances (*pip* ≥ 10) to build

- **output_dir** (*Optional[str]*) – Target output directory, only useful when building one wheel using *pip* ≥ 20.0
- **preparer** (*Optional[TPreparer]*) – A preparer instance, defaults to *None*
- **wheel_cache** (*Optional[TWheelCache]*) – A wheel cache instance, defaults to *None*
- **build_options** (*Optional[List[str]]*) – A list of build options to pass in
- **global_options** (*Optional[List[str]]*) – A list of global options to pass in
- **bool] check_binary_allowed** (*Optional[Callable[TInstallRequirement,]]*) – A callable to check whether we are allowed to build and cache wheels for an *ireq*
- **no_clean** (*bool*) – Whether to avoid cleaning up wheels
- **session** (*Optional[TSession]*) – A *PipSession* instance to pass to create a *finder* if necessary
- **finder** (*Optional[TFinder]*) – A *PackageFinder* instance to use for generating a *WheelBuilder* instance on *pip* < 20
- **install_command** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to *None*.
- **req_tracker** (*Optional[TReqTracker]*) – An optional requirement tracker instance, if one already exists
- **build_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **src_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **download_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **wheel_download_dir** (*Optional[str]*) – Passthrough parameter for building preparer
- **cache_dir** (*Optional[str]*) – Passthrough cache directory for wheel cache options
- **use_user_site** (*bool*) – Whether to use the user site directory when preparing install requirements on *pip* < 20
- **use_pep517** (*Optional[bool]*) – When set to *True* or *False*, prefers building with or without pep517 as specified, otherwise uses requirement preference. Only works for single requirements.
- **format_control_provider** (*Optional[TShimmedFunc]*) – A provider for the *FormatControl* class
- **wheel_cache_provider** (*Optional[TShimmedFunc]*) – A provider for the *WheelCache* class
- **preparer_provider** (*Optional[TShimmedFunc]*) – A provider for the *RequirementPreparer* class
- **wheel_builder_provider** (*Optional[TShimmedFunc]*) – A provider for the *WheelBuilder* class, if it exists
- **build_one_provider** (*Optional[TShimmedFunc]*) – A provider for the *_build_one* function, if it exists
- **build_one_inside_env_provider** (*Optional[TShimmedFunc]*) – A provider for the *_build_one_inside_env* function, if it exists

- **build_many_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *build* function, if it exists
- **install_command_provider** (*Optional*[*TShimmedFunc*]) – A shim for providing new install command instances
- **finder_provider** (*TShimmedFunc*) – A provider to package finder instances
- **reqset_provider** (*TShimmedFunc*) – A provider for requirement set generation

Returns A tuple of successful and failed install requirements or else a path to a wheel

Return type `Optional[Union[str, Tuple[List[TInstallRequirement], List[TInstallRequirement]]]]`

Helper module for shimming functionality across pip versions.

```
class pip_shims.models.ImportTypes
```

```
    Bases: pip_shims.models.ImportTypes
```

```
    Create new instance of ImportTypes(FUNCTION, CLASS, MODULE, CONTEXTMANAGER)
```

```
    ATTRIBUTE = 5
```

```
    CLASS = 1
```

```
    CONTEXTMANAGER = 3
```

```
    FUNCTION = 0
```

```
    METHOD = 4
```

```
    MODULE = 2
```

```
    _asdict ()
```

```
        Return a new OrderedDict which maps field names to their values.
```

```
    _fields = ('FUNCTION', 'CLASS', 'MODULE', 'CONTEXTMANAGER')
```

```
    _fields_defaults = {}
```

```
    classmethod _make (iterable)
```

```
        Make a new ImportTypes object from a sequence or iterable
```

```
    _replace (**kws)
```

```
        Return a new ImportTypes object replacing specified fields with new values
```

```
    count ()
```

```
        Return number of occurrences of value.
```

```
    index ()
```

```
        Return first index of value.
```

```
        Raises ValueError if the value is not present.
```

`pip_shims.models.ImportTypesBase`
 alias of `pip_shims.models.ImportTypes`

```
class pip_shims.models.PipVersion(version, round_prereleases_up=True,
                                  base_import_path=None, vendor_import_path='pip._vendor')
```

Bases: `collections.abc.Sequence`

`_abc_impl = <_abc_data object>`

`_parse()`

`count(value)` → integer – return number of occurrences of value

`index(value[, start[, stop]])` → integer – return first index of value.
 Raises `ValueError` if the value is not present.

Supporting start and stop arguments is optional, but recommended.

`is_valid(compared_to)`

`version_key`

`version_tuple`

```
class pip_shims.models.PipVersionRange(start, end)
```

Bases: `collections.abc.Sequence`

`_abc_impl = <_abc_data object>`

`base_import_paths`

`count(value)` → integer – return number of occurrences of value

`index(value[, start[, stop]])` → integer – return first index of value.
 Raises `ValueError` if the value is not present.

Supporting start and stop arguments is optional, but recommended.

`is_valid()`

`vendor_import_paths`

```
class pip_shims.models.ShimmedPath(name, import_target, import_type, version_range,
                                   provided_methods=None, provided_functions=None,
                                   provided_classmethods=None, provided_contextmanagers=None,
                                   provided_mixins=None, default_args=None)
```

Bases: `object`

`_ShimmedPath__modules = {'pip._internal.cache': <module 'pip._internal.cache' from '/`

`_apply_aliases(imported, target)`

`_as_tuple()`

`_ensure_functions(provided)`

`_ensure_methods(provided)`

Given a base class, a new name, and any number of functions to attach, turns those functions into class-methods, attaches them, and returns an updated class object.

`_import(prefix=None)`

`classmethod _import_module(module)`

`classmethod _parse_provides_dict(provides, prepend_arg_to_callables=None)`

```

    _shim_base (imported, attribute_name)
    _shim_parent (imported, attribute_name)
    _update_default_kwargs (parent, provided)
    alias (aliases)
    calculated_module_path
    is_attribute
    is_class
    is_contextmanager
    is_function
    is_method
    is_module
    is_valid
    shim()
    shim_attribute (imported, attribute_name)
    shim_class (imported, attribute_name)
    shim_contextmanager (imported, attribute_name)
    shim_function (imported, attribute_name)
    shim_module (imported, attribute_name)
    shimmed
    sort_order
    update_sys_modules (imported)

```

```
class pip_shims.models.ShimmedPathCollection (name, import_type, paths=None)
```

```
Bases: object
```

```
    _ShimmedPathCollection__registry = {'AbstractDistribution': <pip_shims.models.Shimmed
```

```
    _get_top_path()
```

```
    _sort_paths()
```

```
    add_mixin (mixin)
```

```
    add_path (path)
```

```
    alias (aliases)
```

Takes a list of methods, functions, attributes, etc and ensures they all exist on the object pointing at the same referent.

Parameters `aliases` (`List[str]`) – Names to map to the same functionality if they do not exist.

Returns `None`

Return type `None`

```
    create_path (import_path, version_start, version_end=None)
```

```
    classmethod get_registry()
```

```
pre_shim (fn)  
provide_function (name, fn)  
provide_method (name, fn)  
register ()  
set_default (default)  
set_default_args (callable_name, *args, **kwargs)  
shim ()  
classmethod traverse (shim)  
  
pip_shims.models.import_pip ()  
pip_shims.models.lookup_current_pip_version ()  
pip_shims.models.pip_version_lookup (version, *args, **kwargs)
```

Backports and helper functionality to support using new functionality.

```
class pip_shims.compat.CandidateEvaluator (project_name, supported_tags, specifier, prefer_binary=False, allow_all_prereleases=False, hashes=None)
```

Bases: `object`

```
classmethod create (project_name, target_python=None, prefer_binary=False, allow_all_prereleases=False, specifier=None, hashes=None)
```

```
class pip_shims.compat.CandidatePreferences (prefer_binary=False, allow_all_prereleases=False)
```

Bases: `object`

```
exception pip_shims.compat.InvalidWheelFilename
```

Bases: `Exception`

Wheel Filename is Invalid

args

with_traceback ()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

```
class pip_shims.compat.LinkCollector (session=None, search_scope=None)
```

Bases: `object`

```
class pip_shims.compat.LinkEvaluator (allow_yanked, project_name, canonical_name, formats, target_python, ignore_requires_python=False, ignore_compatibility=True)
```

Bases: `object`

```
class pip_shims.compat.SearchScope (find_links=None, index_urls=None)
```

Bases: `object`

```
classmethod create (find_links=None, index_urls=None)
```

```
class pip_shims.compat.SelectionPreferences (allow_yanked=True,          al-
                                          low_all_prereleases=False,      for-
                                          mat_control=None,    prefer_binary=False,
                                          ignore_requires_python=False)
```

Bases: `object`

```
class pip_shims.compat.TargetPython (platform=None, py_version_info=None, abi=None, im-
                                     plementation=None)
```

Bases: `object`

```
fallback_get_tags = <pip_shims.models.ShimmedPathCollection object>
```

```
get_tags ()
```

```
class pip_shims.compat.Wheel (filename)
```

Bases: `object`

```
get_formatted_file_tags ()
```

Return the wheel's tags as a sorted list of strings.

```
support_index_min (tags)
```

Return the lowest index that one of the wheel's file_tag combinations achieves in the given list of supported tags.

For example, if there are 8 supported tags and one of the file tags is first in the list, then return 0.

Parameters `tags` – the PEP 425 tags to check the wheel against, in order with most preferred first.

Raises `ValueError` – If none of the wheel's file tags match one of the supported tags.

```
supported (tags)
```

Return whether the wheel is compatible with one of the given tags.

Parameters `tags` – the PEP 425 tags to check the wheel against.

```
wheel_file_re = re.compile ('^(?P<namever> (?P<name> .+?) - (?P<ver> .+?)) \n ( (- (?P<build> \\\
```

```
pip_shims.compat._ensure_finder (finder=None, finder_provider=None, install_cmd=None, op-
                                tions=None, session=None)
```

```
pip_shims.compat._ensure_wheel_cache (wheel_cache=None,    wheel_cache_provider=None,
                                       format_control=None, format_control_provider=None,
                                       options=None, cache_dir=None)
```

```
pip_shims.compat.build_wheel (req=None,    reqset=None,    output_dir=None,    pre-
                              parer=None,    wheel_cache=None,    build_options=None,
                              global_options=None,    check_binary_allowed=None,
                              no_clean=False,    session=None,    finder=None,    in-
                              stall_command=None,    req_tracker=None,    build_dir=None,
                              src_dir=None,    download_dir=None,    wheel_download_dir=None,
                              cache_dir=None,    use_user_site=False,    use_pep517=None,
                              format_control_provider=None,    wheel_cache_provider=None,
                              preparer_provider=None,    wheel_builder_provider=None,
                              build_one_provider=None,    build_one_inside_env_provider=None,
                              build_many_provider=None,    install_command_provider=None,
                              finder_provider=None,    reqset_provider=None)
```

Build a wheel or a set of wheels

Raises `TypeError` – Raised when no requirements are provided

Parameters

- `req` (`Optional [TInstallRequirement]`) – An `InstallRequirement` to build

- **reqset** (*Optional*[*TReqSet*]) – A *RequirementSet* instance (*pip*<10) or an iterable of *InstallRequirement* instances (*pip*>=10) to build
- **output_dir** (*Optional*[*str*]) – Target output directory, only useful when building one wheel using *pip*>=20.0
- **preparer** (*Optional*[*TPreparer*]) – A preparer instance, defaults to *None*
- **wheel_cache** (*Optional*[*TWheelCache*]) – A wheel cache instance, defaults to *None*
- **build_options** (*Optional*[*List*[*str*]]) – A list of build options to pass in
- **global_options** (*Optional*[*List*[*str*]]) – A list of global options to pass in
- **bool]] check_binary_allowed** (*Optional*[*Callable*[*TInstallRequirement*,]]) – A callable to check whether we are allowed to build and cache wheels for an *ireq*
- **no_clean** (*bool*) – Whether to avoid cleaning up wheels
- **session** (*Optional*[*TSession*]) – A *PipSession* instance to pass to create a *finder* if necessary
- **finder** (*Optional*[*TFinder*]) – A *PackageFinder* instance to use for generating a *WheelBuilder* instance on *pip*<20
- **install_command** (*Optional*[*TCommandInstance*]) – The install command used to create the finder, session, and options if needed, defaults to *None*.
- **req_tracker** (*Optional*[*TReqTracker*]) – An optional requirement tracker instance, if one already exists
- **build_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **src_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **download_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **wheel_download_dir** (*Optional*[*str*]) – Passthrough parameter for building preparer
- **cache_dir** (*Optional*[*str*]) – Passthrough cache directory for wheel cache options
- **use_user_site** (*bool*) – Whether to use the user site directory when preparing install requirements on *pip*<20
- **use_pep517** (*Optional*[*bool*]) – When set to *True* or *False*, prefers building with or without pep517 as specified, otherwise uses requirement preference. Only works for single requirements.
- **format_control_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *FormatControl* class
- **wheel_cache_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *WheelCache* class
- **preparer_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *RequirementPreparer* class
- **wheel_builder_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *WheelBuilder* class, if it exists
- **build_one_provider** (*Optional*[*TShimmedFunc*]) – A provider for the *_build_one* function, if it exists

- **build_one_inside_env_provider** (*Optional*[*TShimmedFunc*]) – A provider for the `_build_one_inside_env` function, if it exists
- **build_many_provider** (*Optional*[*TShimmedFunc*]) – A provider for the `build` function, if it exists
- **install_command_provider** (*Optional*[*TShimmedFunc*]) – A shim for providing new install command instances
- **finder_provider** (*TShimmedFunc*) – A provider to package finder instances
- **reqset_provider** (*TShimmedFunc*) – A provider for requirement set generation

Returns A tuple of successful and failed install requirements or else a path to a wheel

Return type *Optional*[*Union*[*str*, *Tuple*[*List*[*TInstallRequirement*], *List*[*TInstallRequirement*]]]

`pip_shims.compat.ensure_resolution_dirs` (***kwargs*)

Ensures that the proper directories are scaffolded and present in the provided `kwargs` for performing dependency resolution via `pip`.

Returns A new `kwargs` dictionary with scaffolded directories for `build_dir`, `src_dir`, `download_dir`, and `wheel_download_dir` added to the key value pairs.

Return type *Dict*[*str*, *Any*]

`pip_shims.compat.get_ireq_output_path` (*wheel_cache*, *ireq*)

`pip_shims.compat.get_package_finder` (*install_cmd=None*, *options=None*, *session=None*,
platform=None, *python_versions=None*,
abi=None, *implementation=None*, *target_python=None*, *ignore_requires_python=None*,
target_python_builder=None, *install_cmd_provider=None*)

Shim for compatibility to generate package finders.

Build and return a `PackageFinder` instance using the `InstallCommand` helper method to construct the finder, shimmed with backports as needed for compatibility.

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_cmd** – A `InstallCommand` instance which is used to generate the finder.
- **options** (*optparse.Values*) – An optional `optparse.Values` instance generated by calling `install_cmd.parser.parse_args()` typically.
- **session** – An optional `session` instance, can be created by the `install_cmd`.
- **platform** (*Optional*[*str*]) – An optional platform string, e.g. `linux_x86_64`
- **python_versions** (*Optional*[*Tuple*[*str*, ...]) – A tuple of 2-digit strings representing python versions, e.g. (“27”, “35”, “36”, “37”...)
- **abi** (*Optional*[*str*]) – The target abi to support, e.g. “cp38”
- **implementation** (*Optional*[*str*]) – An optional implementation string for limiting searches to a specific implementation, e.g. “cp” or “py”
- **target_python** – A `TargetPython` instance (will be translated to alternate arguments if necessary on incompatible pip versions).
- **ignore_requires_python** (*Optional*[*bool*]) – Whether to ignore `requires_python` on resulting candidates, only valid after pip version 19.3.1

- **target_python_builder** – A ‘TargetPython’ builder (e.g. the class itself, uninstantiated)

Returns A `pip._internal.index.package_finder.PackageFinder` instance

Return type `pip._internal.index.package_finder.PackageFinder`

Example

```
>>> from pip_shims.shims import InstallCommand, get_package_finder
>>> install_cmd = InstallCommand()
>>> finder = get_package_finder(
...     install_cmd, python_versions=("27", "35", "36", "37", "38"),
↳ implementation="
cp"
... )
>>> candidates = finder.find_all_candidates("requests")
>>> requests_222 = next(iter(c for c in candidates if c.version.public == "2.22.0
↳ "))
>>> requests_222
<InstallationCandidate('requests', <Version('2.22.0')>, <Link https://files.
↳ pythonhos
ted.org/packages/51/bd/
↳ 23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/r
equests-2.22.0-py2.py3-none-any.whl
↳ #sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9
a590f48c010551dc6c4b31 (from https://pypi.org/simple/requests/) (requires-python:>
↳ =2.
7, !=3.0.*, !=3.1.*, !=3.2.*, !=3.3.*, !=3.4.*)>>>
```

```
pip_shims.compat.get_requirement_set (install_command=None, req_set_provider=None,
build_dir=None, src_dir=None, down-
load_dir=None, wheel_download_dir=None,
session=None, wheel_cache=None, up-
grade=False, upgrade_strategy=None, ig-
nore_installed=False, ignore_dependencies=False,
force_reinstall=False, use_user_site=False, iso-
lated=False, ignore_requires_python=False,
require_hashes=None, cache_dir=None, op-
tions=None, install_cmd_provider=None,
wheel_cache_provider=None)
```

Creates a requirement set from the supplied parameters.

Not all parameters are passed through for all pip versions, but any invalid parameters will be ignored if they are not needed to generate a requirement set on the current pip version.

:param *ShimmedPathCollection* wheel_cache_provider: A context manager provider which resolves to a *WheelCache* instance

Parameters install_command – A *InstallCommand* instance which is used to generate the finder.

:param *ShimmedPathCollection* req_set_provider: A provider to build requirement set instances.

Parameters

- **build_dir** (*str*) – The directory to build requirements in. Removed in pip 10, defaults to None

- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None
- **download_dir** (*str*) – The directory to download requirement artifacts to. Removed in pip 10, defaults to None
- **wheel_download_dir** (*str*) – The directory to download wheels to. Removed in pip 10, defaults to None

:param Session session: The pip session to use. Removed in pip 10, defaults to None

Parameters

- **wheel_cache** (*WheelCache*) – The pip WheelCache instance to use for caching wheels. Removed in pip 10, defaults to None
- **upgrade** (*bool*) – Whether to try to upgrade existing requirements. Removed in pip 10, defaults to False.
- **upgrade_strategy** (*str*) – The upgrade strategy to use, e.g. “only-if-needed”. Removed in pip 10, defaults to None.
- **ignore_installed** (*bool*) – Whether to ignore installed packages when resolving. Removed in pip 10, defaults to False.
- **ignore_dependencies** (*bool*) – Whether to ignore dependencies of requirements when resolving. Removed in pip 10, defaults to False.
- **force_reinstall** (*bool*) – Whether to force reinstall of packages when resolving. Removed in pip 10, defaults to False.
- **use_user_site** (*bool*) – Whether to use user site packages when resolving. Removed in pip 10, defaults to False.
- **isolated** (*bool*) – Whether to resolve in isolation. Removed in pip 10, defaults to False.
- **ignore_requires_python** (*bool*) – Removed in pip 10, defaults to False.
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **options** (*Values*) – An Values instance from an install cmd
- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.

Returns A new requirement set instance

Return type RequirementSet

`pip_shims.compat.get_requirement_tracker` (*req_tracker_creator=None*)

`pip_shims.compat.get_resolver` (*resolver_fn*, *install_req_provider=None*, *format_control_provider=None*, *wheel_cache_provider=None*, *finder=None*, *upgrade_strategy='to-satisfy-only'*, *force_reinstall=None*, *ignore_dependencies=None*, *ignore_requires_python=None*, *ignore_installed=True*, *use_user_site=False*, *isolated=None*, *wheel_cache=None*, *preparer=None*, *session=None*, *options=None*, *make_install_req=None*, *install_cmd_provider=None*, *install_cmd=None*, *use_pep517=True*)

A resolver creation compatibility shim for generating a resolver.

Consumes any argument that was previously used to instantiate a resolver, discards anything that is no longer valid.

Note: This is only valid for **pip >= 10.0.0**

Raises

- **ValueError** – A session is required but not provided and one cannot be created
- **ValueError** – A finder is required but not provided and one cannot be created
- **ValueError** – An install requirement provider is required and has not been provided

Parameters

- **resolver_fn** (*TShimmedFunc*) – The resolver function used to create new resolver instances.
- **install_req_provider** (*TShimmedFunc*) – The provider function to use to generate install requirements if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a format_control instance if needed.
- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to `only-if-needed`.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to None
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to None
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to None
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to True
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to False
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to None
- **wheel_cache** (*Optional[TWheelCache]*) – The wheel cache to use, defaults to None
- **preparer** (*Optional[TPreparer]*) – The requirement preparer to use, defaults to None
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **make_install_req** (*Optional[functools.partial]*) – The partial function to pass in to the resolver for actually generating install requirements, if necessary

- `install_cmd` (*Optional*[*TCommandInstance*]) – The install command used to create the finder, session, and options if needed, defaults to `None`.
- `use_pep517` (*bool*) – Whether to use the pep517 build process.

Returns A new resolver instance.

Return type `Resolver`

Example

```
>>> import os
>>> from tempdir import TemporaryDirectory
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_
↳ tracker,
...     get_resolver, InstallRequirement, RequirementSet
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> wheel_cache = WheelCache(USER_CACHE_DIR, FormatControl(None, None))
>>> with TemporaryDirectory() as temp_base:
...     reqset = RequirementSet()
...     ireq = InstallRequirement.from_line("requests")
...     ireq.is_direct = True
...     build_dir = os.path.join(temp_base, "build")
...     src_dir = os.path.join(temp_base, "src")
...     ireq.build_location(build_dir)
...     with make_preparer(
...         options=pip_options, finder=finder, session=session,
...         build_dir=build_dir, install_cmd=install_cmd,
...     ) as preparer:
...         resolver = get_resolver(
...             finder=finder, ignore_dependencies=False, ignore_requires_
↳ python=True,
...             preparer=preparer, session=session, options=pip_options,
...             install_cmd=install_cmd, wheel_cache=wheel_cache,
...         )
...         resolver.require_hashes = False
...         reqset.add_requirement(ireq)
...         results = resolver.resolve(reqset)
...         #reqset.cleanup_files()
...         for result_req in reqset.requirements:
...             print(result_req)
requests
chardet
certifi
urllib3
idna
```

`pip_shims.compat.get_session` (*install_cmd_provider=None, install_cmd=None, options=None*)

```

pip_shims.compat.make_preparer(preparer_fn, req_tracker_fn=None, build_dir=None,
                               src_dir=None, download_dir=None,
                               wheel_download_dir=None, progress_bar='off',
                               build_isolation=False, session=None, finder=None, op-
                               tions=None, require_hashes=None, use_user_site=None,
                               req_tracker=None, install_cmd_provider=None,
                               downloader_provider=None, install_cmd=None,
                               finder_provider=None)

```

Creates a requirement preparer for preparing pip requirements.

Provides a compatibility shim that accepts all previously valid arguments and discards any that are no longer used.

Raises

- **TypeError** – No requirement tracker provided and one cannot be generated
- **TypeError** – No valid sessions provided and one cannot be generated
- **TypeError** – No valid finders provided and one cannot be generated

Parameters

- **preparer_fn** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **req_tracker_fn** (*Optional[TShimmedFunc]*) – Callable or shim for generating requirement trackers, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **src_dir** (*Optional[str]*) – Directory to find or extract source files, defaults to None
- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **progress_bar** (*str*) – Whether to display a progress bar, defaults to off
- **build_isolation** (*bool*) – Whether to build requirements in isolation, defaults to False
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None
- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **require_hashes** (*Optional[bool]*) – Whether to require hashes for preparation
- **use_user_site** (*Optional[bool]*) – Whether to use the user site directory for preparing requirements
- **TShimmedFunc]** **req_tracker** (*Optional[Union[TReqTracker,]]*) – The requirement tracker to use for building packages, defaults to None
- **downloader_provider** (*Optional[TShimmedFunc]*) – A downloader provider
- **install_cmd** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None
- **finder_provider** (*Optional[TShimmedFunc]*) – A package finder provider

Yield A new requirement preparer instance

Return type ContextManager[RequirementPreparer]

Example

```
>>> from pip_shims.shims import (
...     InstallCommand, get_package_finder, make_preparer, get_requirement_tracker
... )
>>> install_cmd = InstallCommand()
>>> pip_options, _ = install_cmd.parser.parse_args([])
>>> session = install_cmd._build_session(pip_options)
>>> finder = get_package_finder(
...     install_cmd, session=session, options=pip_options
... )
>>> with make_preparer(
...     options=pip_options, finder=finder, session=session, install_cmd=ic
... ) as preparer:
...     print(preparer)
<pip._internal.operations.prepare.RequirementPreparer object at 0x7f8a2734be80>
```

pip_shims.compat.**partial_command** (*shimmed_path*, *cmd_mapping=None*)

Maps a default set of arguments across all members of a *ShimmedPath* instance, specifically for Command instances which need *summary* and *name* arguments.

:param ShimmedPath shimmed_path: A ShimmedCollection instance

Parameters cmd_mapping (*Any*) – A reference to use for mapping against, e.g. an import that depends on pip also

Returns A dictionary mapping new arguments to their default values

Return type Dict[str, str]

pip_shims.compat.**populate_options** (*install_command=None*, *options=None*, ***kwargs*)

```
pip_shims.compat.resolve (ireq, reqset_provider=None, req_tracker_provider=None,
install_cmd_provider=None, install_command=None,
finder_provider=None, resolver_provider=None,
wheel_cache_provider=None, format_control_provider=None,
make_preparer_provider=None, tempdir_manager_provider=None,
options=None, session=None, resolver=None, finder=None,
upgrade_strategy='to-satisfy-only', force_reinstall=None, ignore_dependencies=None,
ignore_requires_python=None, ignore_installed=True, use_user_site=False, isolated=None,
build_dir=None, source_dir=None, download_dir=None, cache_dir=None, wheel_download_dir=None,
wheel_cache=None, require_hashes=None, check_supported_wheels=True)
```

Resolves the provided **InstallRequirement**, returning a dictionary.

Maps a dictionary of names to corresponding InstallRequirement values.

:param InstallRequirement ireq: An InstallRequirement to initiate the resolution process

:param ShimmedPathCollection reqset_provider: A provider to build requirement set instances.

:param ShimmedPathCollection req_tracker_provider: A provider to build requirement tracker instances

Parameters

- **install_cmd_provider** (*ShimmedPathCollection*) – A shim for providing new install command instances.
- **install_command** (*Optional[TCommandInstance]*) – The install command used to create the finder, session, and options if needed, defaults to None.

:param *ShimmedPathCollection* **finder_provider**: A provider to package finder instances.

:param *ShimmedPathCollection* **resolver_provider**: A provider to build resolver instances

Parameters

- **wheel_cache_provider** (*TShimmedFunc*) – The provider function to use to generate a wheel cache if needed.
- **format_control_provider** (*TShimmedFunc*) – The provider function to use to generate a format_control instance if needed.
- **make_preparer_provider** (*TShimmedFunc*) – Callable or shim for generating preparers.
- **tempdir_manager_provider** (*Optional[TShimmedFunc]*) – Shim for generating tempdir manager for pip temporary directories
- **options** (*Optional[Values]*) – Pip options to use if needed, defaults to None
- **session** (*Optional[TSession]*) – Existing session to use for getting requirements, defaults to None

:param **Resolver** **resolver**: A pre-existing resolver instance to use for resolution

Parameters

- **finder** (*Optional[TFinder]*) – The package finder to use during resolution, defaults to None.
- **upgrade_strategy** (*str*) – Upgrade strategy to use, defaults to *only-if-needed*.
- **force_reinstall** (*Optional[bool]*) – Whether to simulate or assume package re-installation during resolution, defaults to None
- **ignore_dependencies** (*Optional[bool]*) – Whether to ignore package dependencies, defaults to None
- **ignore_requires_python** (*Optional[bool]*) – Whether to ignore indicated required_python versions on packages, defaults to None
- **ignore_installed** (*bool*) – Whether to ignore installed packages during resolution, defaults to True
- **use_user_site** (*bool*) – Whether to use the user site location during resolution, defaults to False
- **isolated** (*Optional[bool]*) – Whether to isolate the resolution process, defaults to None
- **build_dir** (*Optional[str]*) – Directory for building packages and wheels, defaults to None
- **source_dir** (*str*) – The directory to use for source requirements. Removed in pip 10, defaults to None

- **download_dir** (*Optional[str]*) – Target directory to download files, defaults to None
- **cache_dir** (*str*) – The cache directory to use for caching artifacts during resolution
- **wheel_download_dir** (*Optional[str]*) – Target directory to download wheels, defaults to None
- **wheel_cache** (*Optional[WheelCache]*) – The wheel cache to use, defaults to None
- **require_hashes** (*bool*) – Whether to require hashes when resolving. Defaults to False.
- **check_supported_wheels** (*bool*) – Whether to check support of wheels before including them in resolution.

Returns A dictionary mapping requirements to corresponding `~pip._internal.req.req_install.InstallRequirement`'s

Return type `InstallRequirement`

Example

```
>>> from pip_shims.shims import resolve, InstallRequirement
>>> ireq = InstallRequirement.from_line("requests>=2.20")
>>> results = resolve(ireq)
>>> for k, v in results.items():
...     print("{0}: {1!r}".format(k, v))
requests: <InstallRequirement object: requests>=2.20 from https://files.
↳pythonhosted.
org/packages/51/bd/23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/
↳reque
sts-2.22.0-py2.py3-none-any.whl
↳#sha256=9cf5292fcd0f598c671cfc1e0d7d1a7f13bb8085e9a590
f48c010551dc6c4b31 editable=False>
idna: <InstallRequirement object: idna<2.9,>=2.5 from https://files.pythonhosted.
↳org/
packages/14/2c/cd551d81dbe15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-
↳2.8-
py2.py3-none-any.whl
↳#sha256=ea8b7f6188e6fa117537c3df7da9fc686d485087abf6ac197f9c46432
f7e4a3c (from requests>=2.20) editable=False>
urllib3: <InstallRequirement object: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 from_
↳htt
ps://files.pythonhosted.org/packages/b4/40/
↳a9837291310ee1ccc242ceb6ebfd9eb21539649f19
3a7c8c86ba15b98539/urllib3-1.25.7-py2.py3-none-any.whl
↳#sha256=a8a318824cc77d1fd4b2bec
2ded92646630d7fe8619497b142c84a9e6f5a7293 (from requests>=2.20) editable=False>
chardet: <InstallRequirement object: chardet<3.1.0,>=3.0.2 from https://files.
↳pythonh
osted.org/packages/bc/a9/
↳01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8
/chardet-3.0.4-py2.py3-none-any.whl
↳#sha256=fc323ffcaeaed0e0a02bf4d117757b98aed530d9ed
4531e3e15460124c106691 (from requests>=2.20) editable=False>
certifi: <InstallRequirement object: certifi>=2017.4.17 from https://files.
↳pythonhost
ed.org/packages/18/b0/
↳8146a4f8dd402f60744fa380bc73ca47303cccfc8b9190fd16a827281eac2/ce (continues on next page)
```

(continued from previous page)

```
rtifi-2019.9.11-py2.py3-none-any.whl
↪#sha256=fd7c7c74727ddcf00e9acd26bba8da604ffec95bf
1c2144e67aff7a8b50e6cef (from requests>=2.20) editable=False>
```

`pip_shims.compat.resolve_possible_shim(target)`

`pip_shims.compat.shim_unpack(unpack_fn, download_dir, tempdir_manager_provider, ireq=None, link=None, location=None, hashes=None, progress_bar='off', only_download=None, downloader_provider=None, session=None)`

Accepts all parameters that have been valid to pass to `pip._internal.download.unpack_url()` and selects or drops parameters as needed before invoking the provided callable.

Parameters

- **unpack_fn** (*Callable*) – A callable or shim referring to the pip implementation
- **download_dir** (*str*) – The directory to download the file to
- **tempdir_manager_provider** (*TShimmedFunc*) – A callable or shim referring to *global_tempdir_manager* function from pip or a shimmed no-op context manager

:param Optional[InstallRequirement] ireq: an Install Requirement instance, defaults to None

:param Optional[Link] link: A Link instance, defaults to None.

Parameters

- **location** (*Optional[str]*) – A location or source directory if the target is a VCS url, defaults to None.
- **hashes** (*Optional[Any]*) – A Hashes instance, defaults to None
- **progress_bar** (*str*) – Indicates progress par usage during download, defatuls to off.
- **only_download** (*Optional[bool]*) – Whether to skip install, defaults to None.
- **downloader_provider** (*Optional[ShimmedPathCollection]*) – A downloader class to instantiate, if applicable.
- **session** (*Optional[Session]*) – A PipSession instance, defaults to None.

Returns The result of unpacking the url.

Return type `None`

`pip_shims.compat.temp_environ()`

Allow the ability to set `os.environ` temporarily

`pip_shims.compat.wheel_cache(cache_dir=None, wheel_cache_provider=None, format_control=None, format_control_provider=None, tempdir_manager_provider=None)`

CHAPTER 21

pip_shims.environment

Module with functionality to learn about the environment.

```
pip_shims.environment.get_base_import_path()
```

```
pip_shims.environment.get_pip_version(import_path='pip')
```

```
pip_shims.environment.is_type_checking()
```


Shared utility functions which are not specific to any particular module.

```
class pip_shims.utils.BaseClassMethod(func_base, name, *args, **kwargs)  
    Bases: collections.abc.Callable
```

```
    _abc_impl = <_abc_data object>
```

```
class pip_shims.utils.BaseMethod(func_base, name, *args, **kwargs)  
    Bases: collections.abc.Callable
```

```
    _abc_impl = <_abc_data object>
```

```
pip_shims.utils._parse(version)
```

```
pip_shims.utils.add_mixin_to_class(basecls, mixins)
```

Given a class, adds the provided mixin classes as base classes and gives a new class

Parameters

- **basecls** (*Type*) – An initial class to generate a new class from
- **mixins** (*List [Type]*) – A list of mixins to add as base classes

Returns A new class with the provided mixins as base classes

Return type `Type[basecls, *mixins]`

```
pip_shims.utils.apply_alias(imported, target, *aliases)
```

Given a target with attributes, point non-existent aliases at the first existing one

Parameters

- **Type** **imported** (*Union [ModuleType,]*) – A Module or Class base
- **target** (*Any*) – The target which is a member of **imported** and will have aliases
- **aliases** (*str*) – A list of aliases, the first found attribute will be the basis for all non-existent names which will be created as pointers

Returns The original target

Return type Any

`pip_shims.utils.call_function_with_correct_args` (*fn*, ***provided_kwargs*)

Determines which arguments from **provided_kwargs** to call **fn** and calls it.

Consumes a list of allowed arguments (e.g. from `getargs()`) and uses it to determine which of the arguments in the provided kwarg should be passed through to the given callable.

Parameters

- **fn** (*Callable*) – A callable which has some dynamic arguments
- **allowed_args** (*List[str]*) – A list of allowed arguments which can be passed to the supplied function

Returns The result of calling the function

Return type Any

`pip_shims.utils.ensure_function` (*parent*, *funcname*, *func*)

Given a module, a function name, and a function object, attaches the given function to the module and ensures it is named properly according to the provided argument

Parameters

- **parent** (*Any*) – The parent to attach the function to
- **funcname** (*str*) – The name to give the function
- **func** (*Callable*) – The function to rename and attach to **parent**

Returns The function with its name, qualname, etc set to mirror **parent**

Return type Callable

`pip_shims.utils.fallback_is_artifact` (*self*)

`pip_shims.utils.fallback_is_file_url` (*link*)

`pip_shims.utils.fallback_is_vcs` (*self*)

`pip_shims.utils.filter_allowed_args` (*fn*, ***provided_kwargs*)

Given a function and a kwarg mapping, return only those kwarg used in the function.

Parameters

- **fn** (*Callable*) – A function to inspect
- **Any] kwargs** (*Dict[str, Any]*) – A mapping of kwarg to filter

Returns A new, filtered kwarg mapping

Return type Tuple[List[Any], Dict[str, Any]]

`pip_shims.utils.get_allowed_args` (*fn_or_class*)

Given a callable or a class, returns the arguments and default kwarg passed in.

Parameters **Type] fn_or_class** (*Union[Callable, Any]*) – A function, method or class to inspect.

Returns A 2-tuple with a list of arguments and a dictionary of keywords mapped to default values.

Return type Tuple[List[str], Dict[str, Any]]

`pip_shims.utils.get_method_args` (*target_method*)

Returns the arguments for a callable.

Parameters **target_method** (*Callable*) – A callable to retrieve arguments for

Returns A 2-tuple of the original callable and its resulting arguments

Return type Tuple[Callable, Optional[inspect.Arguments]]

`pip_shims.utils.has_property` (*target*, *name*)

`pip_shims.utils.make_classmethod` (*fn*)

`pip_shims.utils.make_method` (*fn*)

`pip_shims.utils.memoize` (*obj*)

`pip_shims.utils.nullcontext` (**args*, ***kwargs*)

`pip_shims.utils.parse_version` (*version*)

`pip_shims.utils.resolve_possible_shim` (*target*)

`pip_shims.utils.set_default_kwargs` (*basecls*, *method*, **args*, ***default_kwargs*)

`pip_shims.utils.split_package` (*module*, *subimport=None*)

Used to determine what target to import.

Either splits off the final segment or uses the provided sub-import to return a 2-tuple of the import path and the target module or sub-path.

Parameters

- **module** (*str*) – A package to import from
- **subimport** (*Optional[str]*) – A class, function, or subpackage to import

Returns A 2-tuple of the corresponding import package and sub-import path

Return type Tuple[str, str]

Example

```
>>> from pip_shims.utils import split_package
>>> split_package("pip._internal.req.req_install", subimport="InstallRequirement")
("pip._internal.req.req_install", "InstallRequirement")
>>> split_package("pip._internal.cli.base_command")
("pip._internal.cli", "base_command")
```

`pip_shims.utils.suppress_setattr` (*obj*, *attr*, *value*, *filter_none=False*)

Set an attribute, suppressing any exceptions and skipping the attempt on failure.

Parameters

- **obj** (*Any*) – Object to set the attribute on
- **attr** (*str*) – The attribute name to set
- **value** (*Any*) – The value to set the attribute to
- **filter_none** (*bool*) – [description], defaults to False

Returns Nothing

Return type None

Example

```
>>> class MyClass(object):
...     def __init__(self, name):
...         self.name = name
...         self.parent = None
```

(continues on next page)

(continued from previous page)

```
...     def __repr__(self):
...         return "<{0!r} instance (name={1!r}, parent={2!r})>".format(
...             self.__class__.__name__, self.name, self.parent
...         )
...     def __str__(self):
...         return self.name
>>> me = MyClass("Dan")
>>> dad = MyClass("John")
>>> grandfather = MyClass("Joe")
>>> suppress_setattr(dad, "parent", grandfather)
>>> dad
<'MyClass' instance (name='John', parent=<'MyClass' instance (name='Joe',
↳parent=None
)>>>
>>> suppress_setattr(me, "parent", dad)
>>> me
<'MyClass' instance (name='Dan', parent=<'MyClass' instance (name='John', parent=<
↳'My
Class' instance (name='Joe', parent=None)>>>)>>>
>>> suppress_setattr(me, "grandparent", grandfather)
>>> me
<'MyClass' instance (name='Dan', parent=<'MyClass' instance (name='John', parent=<
↳'My
Class' instance (name='Joe', parent=None)>>>)>>>
```

CHAPTER 23

Indices and tables

- `genindex`
- `modindex`
- `search`

p

pip_shims, 5
pip_shims.compat, 85
pip_shims.environment, 98
pip_shims.models, 98
pip_shims.shims, 101
pip_shims.utils, 132

Symbols

- `_ShimmedPathCollection__registry`
(*pip_shims.models.ShimmedPathCollection* attribute), 8, 100, 168, 281
- `_ShimmedPath__modules`
(*pip_shims.models.ShimmedPath* attribute), 7, 100, 167, 280
- `_abc_impl` (*pip_shims.AbstractDistribution* attribute), 74, 234
- `_abc_impl` (*pip_shims.InstalledDistribution* attribute), 74, 234
- `_abc_impl` (*pip_shims.SourceDistribution* attribute), 75, 235
- `_abc_impl` (*pip_shims.WheelDistribution* attribute), 75, 235
- `_abc_impl` (*pip_shims.models.PipVersion* attribute), 6, 99, 166, 280
- `_abc_impl` (*pip_shims.models.PipVersionRange* attribute), 6, 99, 166, 280
- `_abc_impl` (*pip_shims.shims.AbstractDistribution* attribute), 43, 121, 203, 266
- `_abc_impl` (*pip_shims.shims.InstalledDistribution* attribute), 44, 121, 204, 266
- `_abc_impl` (*pip_shims.shims.SourceDistribution* attribute), 44, 121, 204, 267
- `_abc_impl` (*pip_shims.shims.WheelDistribution* attribute), 44, 121, 204, 267
- `_abc_impl` (*pip_shims.utils.BaseClassMethod* attribute), 21, 132, 181, 299
- `_abc_impl` (*pip_shims.utils.BaseMethod* attribute), 21, 132, 181, 299
- `_add_help_option()`
(*pip_shims.ConfigOptionParser* method), 56, 216
- `_add_help_option()`
(*pip_shims.shims.ConfigOptionParser* method), 25, 102, 185, 248
- `_add_version_option()`
(*pip_shims.ConfigOptionParser* method), 56, 216
- `_add_version_option()`
(*pip_shims.shims.ConfigOptionParser* method), 25, 102, 185, 248
- `_allowed_strategies` (*pip_shims.Resolver* attribute), 71, 231
- `_allowed_strategies` (*pip_shims.shims.Resolver* attribute), 40, 118, 200, 263
- `_allowed_to_proceed()`
(*pip_shims.UninstallPathSet* method), 72, 232
- `_allowed_to_proceed()`
(*pip_shims.shims.UninstallPathSet* method), 41, 119, 201, 264
- `_apply_aliases()` (*pip_shims.models.ShimmedPath* method), 7, 100, 167, 280
- `_as_tuple()` (*pip_shims.models.ShimmedPath* method), 7, 100, 167, 280
- `_asdict()` (*pip_shims.models.ImportTypes* method), 6, 98, 166, 279
- `_build_session()` (*pip_shims.Command* method), 55, 215
- `_build_session()` (*pip_shims.SessionCommandMixin* method), 55, 215
- `_build_session()` (*pip_shims.shims.Command* method), 24, 102, 184, 247
- `_build_session()` (*pip_shims.shims.SessionCommandMixin* method), 24, 101, 184, 247
- `_check_conflict()`
(*pip_shims.ConfigOptionParser* method), 56, 216
- `_check_conflict()`
(*pip_shims.shims.ConfigOptionParser* method), 25, 102, 185, 248
- `_check_skip_installed()` (*pip_shims.Resolver* method), 71, 231
- `_check_skip_installed()`
(*pip_shims.shims.Resolver* method), 40, 118, 200, 263
- `_compare()` (*pip_shims.Link* method), 62, 222

- `_compare()` (*pip_shims.shims.Link* method), 32, 109, 192, 255
- `_compare_key` (*pip_shims.Link* attribute), 62, 222
- `_compare_key` (*pip_shims.shims.Link* attribute), 32, 109, 192, 255
- `_create()` (*pip_shims.TempDirectory* method), 70, 230
- `_create()` (*pip_shims.shims.TempDirectory* method), 39, 117, 199, 262
- `_create_option_list()` (*pip_shims.ConfigOptionParser* method), 56, 216
- `_create_option_list()` (*pip_shims.shims.ConfigOptionParser* method), 25, 102, 185, 248
- `_create_option_mappings()` (*pip_shims.ConfigOptionParser* method), 56, 216
- `_create_option_mappings()` (*pip_shims.shims.ConfigOptionParser* method), 25, 102, 185, 248
- `_defining_class` (*pip_shims.Link* attribute), 63, 223
- `_defining_class` (*pip_shims.shims.Link* attribute), 32, 109, 192, 255
- `_download_should_save` (*pip_shims.RequirementPreparer* attribute), 69, 229
- `_download_should_save` (*pip_shims.shims.RequirementPreparer* attribute), 38, 115, 198, 261
- `_egg_fragment_re` (*pip_shims.Link* attribute), 63, 223
- `_egg_fragment_re` (*pip_shims.shims.Link* attribute), 32, 109, 192, 255
- `_ensure_finder()` (in module *pip_shims.compat*), 9, 87, 169, 284
- `_ensure_functions()` (*pip_shims.models.ShimmedPath* method), 7, 100, 167, 280
- `_ensure_link_req_src_dir()` (*pip_shims.RequirementPreparer* method), 69, 229
- `_ensure_link_req_src_dir()` (*pip_shims.shims.RequirementPreparer* method), 38, 115, 198, 261
- `_ensure_methods()` (*pip_shims.models.ShimmedPath* method), 7, 100, 167, 280
- `_ensure_wheel_cache()` (in module *pip_shims.compat*), 9, 87, 170, 284
- `_entry_path()` (*pip_shims.RequirementTracker* method), 70, 230
- `_entry_path()` (*pip_shims.shims.RequirementTracker* method), 39, 116, 199, 262
- `_fields` (*pip_shims.models.ImportTypes* attribute), 6, 99, 166, 279
- `_fields_defaults` (*pip_shims.models.ImportTypes* attribute), 6, 99, 166, 279
- `_find_requirement_link()` (*pip_shims.Resolver* method), 71, 231
- `_find_requirement_link()` (*pip_shims.shims.Resolver* method), 41, 118, 201, 263
- `_generate_metadata()` (*pip_shims.InstallRequirement* method), 60, 220
- `_generate_metadata()` (*pip_shims.shims.InstallRequirement* method), 29, 107, 189, 252
- `_get_abstract_dist_for()` (*pip_shims.Resolver* method), 71, 231
- `_get_abstract_dist_for()` (*pip_shims.shims.Resolver* method), 41, 118, 201, 263
- `_get_all_options()` (*pip_shims.ConfigOptionParser* method), 56, 216
- `_get_all_options()` (*pip_shims.shims.ConfigOptionParser* method), 25, 102, 185, 248
- `_get_archive_name()` (*pip_shims.InstallRequirement* method), 60, 220
- `_get_archive_name()` (*pip_shims.shims.InstallRequirement* method), 29, 107, 189, 252
- `_get_args()` (*pip_shims.ConfigOptionParser* method), 56, 216
- `_get_args()` (*pip_shims.shims.ConfigOptionParser* method), 25, 102, 185, 248
- `_get_cache_path()` (*pip_shims.SafeFileCache* method), 72, 232
- `_get_cache_path()` (*pip_shims.shims.SafeFileCache* method), 41, 119, 201, 264
- `_get_cache_path_parts()` (*pip_shims.WheelCache* method), 73, 233
- `_get_cache_path_parts()` (*pip_shims.shims.WheelCache* method), 43, 120, 203, 266
- `_get_cache_path_parts_legacy()` (*pip_shims.WheelCache* method), 74, 234
- `_get_cache_path_parts_legacy()` (*pip_shims.shims.WheelCache* method), 43, 120, 203, 266
- `_get_candidates()` (*pip_shims.WheelCache* method), 74, 234
- `_get_candidates()` (*pip_shims.shims.WheelCache* method), 43, 120, 203, 266

method), 43, 120, 203, 266
 _get_index_urls() (*pip_shims.Command class method*), 55, 215
 _get_index_urls() (*pip_shims.SessionCommandMixin class method*), 55, 215
 _get_index_urls() (*pip_shims.shims.Command class method*), 24, 102, 184, 247
 _get_index_urls() (*pip_shims.shims.SessionCommandMixin class method*), 24, 101, 184, 247
 _get_linked_req_hashes() (*pip_shims.RequirementPreparer method*), 69, 229
 _get_linked_req_hashes() (*pip_shims.shims.RequirementPreparer method*), 38, 115, 198, 261
 _get_ordered_configuration_items() (*pip_shims.ConfigOptionParser method*), 56, 216
 _get_ordered_configuration_items() (*pip_shims.shims.ConfigOptionParser method*), 25, 102, 185, 248
 _get_top_path() (*pip_shims.models.ShimmedPathCollection method*), 8, 101, 168, 281
 _given_py_version_info (*pip_shims.TargetPython attribute*), 67, 227
 _given_py_version_info (*pip_shims.shims.TargetPython attribute*), 36, 114, 196, 259
 _hash_re (*pip_shims.Link attribute*), 63, 223
 _hash_re (*pip_shims.shims.Link attribute*), 32, 109, 192, 255
 _import() (*pip_shims.models.ShimmedPath method*), 7, 100, 167, 280
 _import_module() (*pip_shims.models.ShimmedPath class method*), 7, 100, 167, 280
 _init_parsing_state() (*pip_shims.ConfigOptionParser method*), 56, 216
 _init_parsing_state() (*pip_shims.shims.ConfigOptionParser method*), 25, 102, 185, 248
 _is_upgrade_allowed() (*pip_shims.Resolver method*), 71, 231
 _is_upgrade_allowed() (*pip_shims.shims.Resolver method*), 41, 118, 201, 263
 _log_preparing_link() (*pip_shims.RequirementPreparer method*), 69, 229
 _log_preparing_link() (*pip_shims.shims.RequirementPreparer method*), 38, 115, 198, 261
 _log_skipped_link() (*pip_shims.PackageFinder method*), 64, 224
 _log_skipped_link() (*pip_shims.shims.PackageFinder method*), 33, 110, 193, 256
 _main() (*pip_shims.Command method*), 55, 215
 _main() (*pip_shims.shims.Command method*), 24, 102, 184, 247
 _make() (*pip_shims.models.ImportTypes class method*), 6, 99, 166, 279
 _match_long_opt() (*pip_shims.ConfigOptionParser method*), 56, 216
 _match_long_opt() (*pip_shims.shims.ConfigOptionParser method*), 25, 102, 185, 248
 _parse() (*in module pip_shims.utils*), 21, 132, 181, 299
 _parse() (*pip_shims.models.PipVersion method*), 6, 99, 166, 280
 _parse_provides_dict() (*pip_shims.models.ShimmedPath class method*), 7, 100, 167, 280
 _parsed_url (*pip_shims.Link attribute*), 63, 223
 _parsed_url (*pip_shims.shims.Link attribute*), 32, 109, 192, 255
 _permitted() (*pip_shims.UninstallPathSet method*), 72, 232
 _permitted() (*pip_shims.shims.UninstallPathSet method*), 42, 119, 202, 264
 _populate_link() (*pip_shims.Resolver method*), 71, 231
 _populate_link() (*pip_shims.shims.Resolver method*), 41, 118, 201, 263
 _populate_option_list() (*pip_shims.ConfigOptionParser method*), 56, 216
 _populate_option_list() (*pip_shims.shims.ConfigOptionParser method*), 25, 102, 185, 248
 _process_args() (*pip_shims.ConfigOptionParser method*), 56, 216
 _process_args() (*pip_shims.shims.ConfigOptionParser method*), 25, 102, 185, 248
 _process_long_opt() (*pip_shims.ConfigOptionParser method*), 56, 216
 _process_long_opt() (*pip_shims.shims.ConfigOptionParser method*), 25, 103, 185, 248
 _process_short_opts() (*pip_shims.ConfigOptionParser method*), 56, 216

- `_process_short_opts()` (*pip_shims.shims.ConfigOptionParser method*), 25, 103, 185, 248
 - `_py_version_re` (*pip_shims.LinkEvaluator attribute*), 67, 227
 - `_py_version_re` (*pip_shims.shims.LinkEvaluator attribute*), 36, 113, 196, 259
 - `_registry` (*pip_shims.VcsSupport attribute*), 73, 233
 - `_registry` (*pip_shims.shims.VcsSupport attribute*), 42, 119, 202, 265
 - `_replace()` (*pip_shims.models.ImportTypes method*), 6, 99, 166, 279
 - `_resolve_one()` (*pip_shims.Resolver method*), 72, 232
 - `_resolve_one()` (*pip_shims.shims.Resolver method*), 41, 118, 201, 264
 - `_set_req_to_reinstall()` (*pip_shims.Resolver method*), 72, 232
 - `_set_req_to_reinstall()` (*pip_shims.shims.Resolver method*), 41, 118, 201, 264
 - `_set_requirement()` (*pip_shims.InstallRequirement method*), 60, 220
 - `_set_requirement()` (*pip_shims.shims.InstallRequirement method*), 29, 107, 189, 252
 - `_setup_isolation()` (*pip_shims.SourceDistribution method*), 75, 235
 - `_setup_isolation()` (*pip_shims.shims.SourceDistribution method*), 44, 121, 204, 267
 - `_share_option_mappings()` (*pip_shims.ConfigOptionParser method*), 56, 216
 - `_share_option_mappings()` (*pip_shims.shims.ConfigOptionParser method*), 25, 103, 185, 248
 - `_shim_base()` (*pip_shims.models.ShimmedPath method*), 7, 100, 167, 280
 - `_shim_parent()` (*pip_shims.models.ShimmedPath method*), 7, 100, 167, 281
 - `_sort_key()` (*pip_shims.CandidateEvaluator method*), 65, 225
 - `_sort_key()` (*pip_shims.shims.CandidateEvaluator method*), 34, 112, 194, 257
 - `_sort_links()` (*pip_shims.PackageFinder method*), 64, 224
 - `_sort_links()` (*pip_shims.shims.PackageFinder method*), 33, 110, 193, 256
 - `_sort_paths()` (*pip_shims.models.ShimmedPathCollection method*), 8, 101, 168, 281
 - `_strip_extras()` (*in module pip_shims*), 55, 215
 - `_strip_extras()` (*in module pip_shims.shims*), 24, 101, 184, 247
 - `_subdirectory_fragment_re` (*pip_shims.Link attribute*), 63, 223
 - `_subdirectory_fragment_re` (*pip_shims.shims.Link attribute*), 32, 109, 192, 255
 - `_update_default_kwargs()` (*pip_shims.models.ShimmedPath method*), 7, 100, 167, 281
 - `_update_defaults()` (*pip_shims.ConfigOptionParser method*), 56, 216
 - `_update_defaults()` (*pip_shims.shims.ConfigOptionParser method*), 25, 103, 185, 248
 - `_url` (*pip_shims.Link attribute*), 63, 223
 - `_url` (*pip_shims.shims.Link attribute*), 32, 109, 192, 255
 - `_valid_tags` (*pip_shims.TargetPython attribute*), 67, 227
 - `_valid_tags` (*pip_shims.shims.TargetPython attribute*), 36, 114, 196, 259
- ## A
- `abi` (*pip_shims.shims.TargetPython attribute*), 36, 114, 196, 259
 - `abi` (*pip_shims.TargetPython attribute*), 67, 227
 - `AbstractDistribution` (*class in pip_shims*), 74, 234
 - `AbstractDistribution` (*class in pip_shims.shims*), 43, 121, 203, 266
 - `add()` (*pip_shims.RequirementTracker method*), 70, 230
 - `add()` (*pip_shims.shims.RequirementTracker method*), 39, 116, 199, 262
 - `add()` (*pip_shims.shims.UninstallPathSet method*), 42, 119, 202, 264
 - `add()` (*pip_shims.UninstallPathSet method*), 72, 232
 - `add_mixin()` (*pip_shims.models.ShimmedPathCollection method*), 8, 101, 168, 281
 - `add_mixin_to_class()` (*in module pip_shims.utils*), 21, 132, 181, 299
 - `add_named_requirement()` (*pip_shims.RequirementSet method*), 69, 229
 - `add_named_requirement()` (*pip_shims.shims.RequirementSet method*), 38, 116, 198, 261
 - `add_option()` (*pip_shims.ConfigOptionParser method*), 56, 216
 - `add_option()` (*pip_shims.shims.ConfigOptionParser method*), 25, 103, 185, 248
 - `add_option_group()` (*pip_shims.ConfigOptionParser method*), 56, 216

- add_option_group() (*pip_shims.shims.ConfigOptionParser method*), 26, 103, 186, 248
 add_options() (*pip_shims.Command method*), 55, 215
 add_options() (*pip_shims.ConfigOptionParser method*), 56, 216
 add_options() (*pip_shims.shims.Command method*), 24, 102, 184, 247
 add_options() (*pip_shims.shims.ConfigOptionParser method*), 26, 103, 186, 248
 add_path() (*pip_shims.models.ShimmedPathCollection method*), 8, 101, 168, 281
 add_ptth() (*pip_shims.shims.UninstallPathSet method*), 42, 119, 202, 264
 add_ptth() (*pip_shims.UninstallPathSet method*), 72, 232
 add_requirement() (*pip_shims.RequirementSet method*), 69, 229
 add_requirement() (*pip_shims.shims.RequirementSet method*), 38, 116, 198, 261
 add_unnamed_requirement() (*pip_shims.RequirementSet method*), 69, 229
 add_unnamed_requirement() (*pip_shims.shims.RequirementSet method*), 39, 116, 199, 262
 alias() (*pip_shims.models.ShimmedPath method*), 7, 100, 167, 281
 alias() (*pip_shims.models.ShimmedPathCollection method*), 8, 101, 168, 281
 all_requirements (*pip_shims.RequirementSet attribute*), 69, 229
 all_requirements (*pip_shims.shims.RequirementSet attribute*), 39, 116, 199, 262
 all_schemes (*pip_shims.shims.VcsSupport attribute*), 42, 119, 202, 265
 all_schemes (*pip_shims.VcsSupport attribute*), 73, 233
 allow_all_prereleases (*pip_shims.PackageFinder attribute*), 64, 224
 allow_all_prereleases (*pip_shims.SelectionPreferences attribute*), 68, 228
 allow_all_prereleases (*pip_shims.shims.PackageFinder attribute*), 33, 110, 193, 256
 allow_all_prereleases (*pip_shims.shims.SelectionPreferences attribute*), 37, 115, 197, 260
 allow_yanked (*pip_shims.SelectionPreferences attribute*), 68, 228
 allow_yanked (*pip_shims.shims.SelectionPreferences attribute*), 37, 115, 197, 260
 apply_alias() (*in module pip_shims.utils*), 21, 133, 181, 299
 archive() (*pip_shims.InstallRequirement method*), 60, 220
 archive() (*pip_shims.shims.InstallRequirement method*), 29, 107, 189, 252
 args (*pip_shims.BadCommand attribute*), 59, 219
 args (*pip_shims.BestVersionAlreadyInstalled attribute*), 59, 219
 args (*pip_shims.CommandError attribute*), 60, 220
 args (*pip_shims.compat.InvalidWheelFilename attribute*), 9, 86, 169, 283
 args (*pip_shims.DistributionNotFound attribute*), 58, 218
 args (*pip_shims.InstallationError attribute*), 59, 219
 args (*pip_shims.PipError attribute*), 69, 229
 args (*pip_shims.PreviousBuildDirError attribute*), 60, 220
 args (*pip_shims.RequirementsFileParseError attribute*), 59, 219
 args (*pip_shims.shims.BadCommand attribute*), 29, 106, 189, 251
 args (*pip_shims.shims.BestVersionAlreadyInstalled attribute*), 28, 106, 188, 251
 args (*pip_shims.shims.CommandError attribute*), 29, 106, 189, 252
 args (*pip_shims.shims.DistributionNotFound attribute*), 27, 104, 187, 250
 args (*pip_shims.shims.InstallationError attribute*), 28, 105, 188, 251
 args (*pip_shims.shims.PipError attribute*), 38, 115, 198, 261
 args (*pip_shims.shims.PreviousBuildDirError attribute*), 29, 106, 189, 252
 args (*pip_shims.shims.RequirementsFileParseError attribute*), 28, 105, 188, 251
 args (*pip_shims.shims.UninstallationError attribute*), 28, 105, 188, 251
 args (*pip_shims.UninstallationError attribute*), 59, 219
 assert_source_matches_version() (*pip_shims.InstallRequirement method*), 60, 220
 assert_source_matches_version() (*pip_shims.shims.InstallRequirement method*), 30, 107, 190, 252
 ATTRIBUTE (*pip_shims.models.ImportTypes attribute*), 5, 98, 165, 279

B

- backends (*pip_shims.shims.VcsSupport attribute*), 42, 119, 202, 265
 backends (*pip_shims.VcsSupport attribute*), 73, 233

- BadCommand, 28, 59, 106, 188, 219, 251
 - base_import_paths
 - (*pip_shims.models.PipVersionRange* attribute), 6, 99, 166, 280
 - BaseClassMethod (class in *pip_shims.utils*), 21, 132, 181, 299
 - BaseMethod (class in *pip_shims.utils*), 21, 132, 181, 299
 - BestVersionAlreadyInstalled, 28, 59, 106, 188, 219, 251
 - build() (in module *pip_shims*), 74, 234
 - build() (in module *pip_shims.shims*), 43, 120, 203, 266
 - build_location() (*pip_shims.InstallRequirement* method), 60, 220
 - build_location() (*pip_shims.shims.InstallRequirement* method), 30, 107, 190, 252
 - build_one() (in module *pip_shims*), 74, 234
 - build_one() (in module *pip_shims.shims*), 43, 120, 203, 266
 - build_one_inside_env() (in module *pip_shims*), 74, 234
 - build_one_inside_env() (in module *pip_shims.shims*), 43, 121, 203, 266
 - build_wheel() (in module *pip_shims*), 84, 244
 - build_wheel() (in module *pip_shims.compat*), 10, 87, 170, 284
 - build_wheel() (in module *pip_shims.shims*), 53, 130, 213, 276
- C**
- cache_link_parsing (*pip_shims.Link* attribute), 63, 223
 - cache_link_parsing (*pip_shims.shims.Link* attribute), 32, 109, 192, 255
 - calculated_module_path
 - (*pip_shims.models.ShimmedPath* attribute), 7, 100, 167, 281
 - call_function_with_correct_args() (in module *pip_shims.utils*), 22, 133, 182, 300
 - CandidateEvaluator (class in *pip_shims*), 65, 225
 - CandidateEvaluator (class in *pip_shims.compat*), 8, 85, 168, 283
 - CandidateEvaluator (class in *pip_shims.shims*), 34, 111, 194, 257
 - CandidatePreferences (class in *pip_shims*), 66, 226
 - CandidatePreferences (class in *pip_shims.compat*), 8, 86, 168, 283
 - CandidatePreferences (class in *pip_shims.shims*), 35, 112, 195, 258
 - check_default() (*pip_shims.ConfigOptionParser* method), 56, 216
 - check_default() (*pip_shims.shims.ConfigOptionParser* method), 26, 103, 186, 249
 - check_if_exists() (*pip_shims.InstallRequirement* method), 60, 220
 - check_if_exists() (*pip_shims.shims.InstallRequirement* method), 30, 107, 190, 252
 - check_values() (*pip_shims.ConfigOptionParser* method), 56, 216
 - check_values() (*pip_shims.shims.ConfigOptionParser* method), 26, 103, 186, 249
 - CLASS (*pip_shims.models.ImportTypes* attribute), 5, 98, 165, 279
 - cleanup() (*pip_shims.RequirementTracker* method), 70, 230
 - cleanup() (*pip_shims.shims.RequirementTracker* method), 39, 116, 199, 262
 - cleanup() (*pip_shims.shims.TempDirectory* method), 39, 117, 199, 262
 - cleanup() (*pip_shims.TempDirectory* method), 70, 230
 - close() (*pip_shims.SafeFileCache* method), 72, 232
 - close() (*pip_shims.shims.SafeFileCache* method), 41, 119, 201, 264
 - collect_links() (*pip_shims.LinkCollector* method), 66, 226
 - collect_links() (*pip_shims.shims.LinkCollector* method), 35, 113, 195, 258
 - comes_from (*pip_shims.Link* attribute), 63, 223
 - comes_from (*pip_shims.shims.Link* attribute), 32, 109, 192, 255
 - Command (class in *pip_shims*), 55, 215
 - Command (class in *pip_shims.shims*), 24, 102, 184, 247
 - CommandError, 29, 59, 106, 189, 219, 252
 - commit() (*pip_shims.shims.UninstallPathSet* method), 42, 119, 202, 264
 - commit() (*pip_shims.UninstallPathSet* method), 72, 232
 - compute_best_candidate() (*pip_shims.CandidateEvaluator* method), 66, 226
 - compute_best_candidate() (*pip_shims.shims.CandidateEvaluator* method), 35, 112, 195, 258
 - ConfigOptionParser (class in *pip_shims*), 56, 216
 - ConfigOptionParser (class in *pip_shims.shims*), 25, 102, 185, 248
 - CONTEXTMANAGER (*pip_shims.models.ImportTypes* attribute), 5, 98, 166, 279
 - count() (*pip_shims.models.ImportTypes* method), 6, 99, 166, 279
 - count() (*pip_shims.models.PipVersion* method), 6, 99, 166, 280
 - count() (*pip_shims.models.PipVersionRange* method),

- 6, 99, 166, 280
 - create() (*pip_shims.CandidateEvaluator* class method), 66, 226
 - create() (*pip_shims.compat.CandidateEvaluator* class method), 8, 86, 168, 283
 - create() (*pip_shims.compat.SearchScope* class method), 9, 86, 169, 283
 - create() (*pip_shims.LinkCollector* class method), 66, 226
 - create() (*pip_shims.PackageFinder* class method), 64, 224
 - create() (*pip_shims.SearchScope* class method), 68, 228
 - create() (*pip_shims.shims.CandidateEvaluator* class method), 35, 112, 195, 258
 - create() (*pip_shims.shims.LinkCollector* class method), 35, 113, 195, 258
 - create() (*pip_shims.shims.PackageFinder* class method), 33, 110, 193, 256
 - create() (*pip_shims.shims.SearchScope* class method), 37, 114, 197, 260
 - create_path() (*pip_shims.models.ShimmedPathCollection* method), 8, 101, 168, 281
- ## D
- delete() (*pip_shims.SafeFileCache* method), 72, 232
 - delete() (*pip_shims.shims.SafeFileCache* method), 41, 119, 201, 264
 - destroy() (*pip_shims.ConfigOptionParser* method), 57, 217
 - destroy() (*pip_shims.shims.ConfigOptionParser* method), 26, 103, 186, 249
 - dirname() (*pip_shims.shims.VcsSupport* attribute), 42, 119, 202, 265
 - dirname() (*pip_shims.VcsSupport* attribute), 73, 233
 - disable_interspersed_args() (*pip_shims.ConfigOptionParser* method), 57, 217
 - disable_interspersed_args() (*pip_shims.shims.ConfigOptionParser* method), 26, 103, 186, 249
 - disallow_binaries() (*pip_shims.FormatControl* method), 58, 218
 - disallow_binaries() (*pip_shims.shims.FormatControl* method), 27, 105, 187, 250
 - DistributionNotFound, 27, 58, 104, 187, 218, 250
 - Downloader (class in *pip_shims*), 62, 222
 - Downloader (class in *pip_shims.shims*), 31, 108, 191, 254
- ## E
- egg_fragment (*pip_shims.Link* attribute), 63, 223
 - egg_fragment (*pip_shims.shims.Link* attribute), 32, 109, 192, 255
 - enable_interspersed_args() (*pip_shims.ConfigOptionParser* method), 57, 217
 - enable_interspersed_args() (*pip_shims.shims.ConfigOptionParser* method), 26, 103, 186, 249
 - ensure_build_location() (*pip_shims.InstallRequirement* method), 60, 220
 - ensure_build_location() (*pip_shims.shims.InstallRequirement* method), 30, 107, 190, 253
 - ensure_function() (in module *pip_shims.utils*), 22, 133, 182, 300
 - ensure_has_source_dir() (*pip_shims.InstallRequirement* method), 60, 220
 - ensure_has_source_dir() (*pip_shims.shims.InstallRequirement* method), 30, 107, 190, 253
 - ensure_resolution_dirs() (in module *pip_shims.compat*), 11, 88, 171, 286
 - enter_context() (*pip_shims.Command* method), 55, 215
 - enter_context() (*pip_shims.SessionCommandMixin* method), 55, 215
 - enter_context() (*pip_shims.shims.Command* method), 24, 102, 184, 247
 - enter_context() (*pip_shims.shims.SessionCommandMixin* method), 24, 101, 184, 247
 - error() (*pip_shims.ConfigOptionParser* method), 57, 217
 - error() (*pip_shims.shims.ConfigOptionParser* method), 26, 103, 186, 249
 - evaluate_link() (*pip_shims.LinkEvaluator* method), 67, 227
 - evaluate_link() (*pip_shims.shims.LinkEvaluator* method), 36, 113, 196, 259
 - evaluate_links() (*pip_shims.PackageFinder* method), 64, 224
 - evaluate_links() (*pip_shims.shims.PackageFinder* method), 33, 111, 193, 256
 - exit() (*pip_shims.ConfigOptionParser* method), 57, 217
 - exit() (*pip_shims.shims.ConfigOptionParser* method), 26, 103, 186, 249
 - expand_prog_name() (*pip_shims.ConfigOptionParser* method), 57, 217
 - expand_prog_name() (*pip_shims.shims.ConfigOptionParser* method), 26, 103, 186, 249

ext (*pip_shims.Link* attribute), 63, 223

ext (*pip_shims.shims.Link* attribute), 32, 109, 192, 255

F

fallback_get_tags

(*pip_shims.compat.TargetPython* attribute), 9, 86, 169, 284

fallback_is_artifact() (in module *pip_shims.utils*), 22, 133, 182, 300

fallback_is_file_url() (in module *pip_shims.utils*), 22, 133, 182, 300

fallback_is_vcs() (in module *pip_shims.utils*), 22, 133, 182, 300

fetch_page() (*pip_shims.LinkCollector* method), 66, 226

fetch_page() (*pip_shims.shims.LinkCollector* method), 36, 113, 196, 258

file_path (*pip_shims.Link* attribute), 63, 223

file_path (*pip_shims.shims.Link* attribute), 32, 109, 192, 255

filename (*pip_shims.Link* attribute), 63, 223

filename (*pip_shims.shims.Link* attribute), 32, 109, 192, 255

filter_allowed_args() (in module *pip_shims.utils*), 22, 133, 182, 300

find_all_candidates() (*pip_shims.PackageFinder* method), 64, 224

find_all_candidates() (*pip_shims.shims.PackageFinder* method), 33, 111, 193, 256

find_best_candidate() (*pip_shims.PackageFinder* method), 64, 224

find_best_candidate() (*pip_shims.shims.PackageFinder* method), 34, 111, 194, 256

find_links (*pip_shims.LinkCollector* attribute), 66, 226

find_links (*pip_shims.PackageFinder* attribute), 64, 224

find_links (*pip_shims.SearchScope* attribute), 68, 228

find_links (*pip_shims.shims.LinkCollector* attribute), 36, 113, 196, 258

find_links (*pip_shims.shims.PackageFinder* attribute), 34, 111, 194, 257

find_links (*pip_shims.shims.SearchScope* attribute), 37, 114, 197, 260

find_requirement() (*pip_shims.PackageFinder* method), 64, 224

find_requirement() (*pip_shims.shims.PackageFinder* method), 34, 111, 194, 257

format_control (*pip_shims.SelectionPreferences* attribute), 68, 228

format_control (*pip_shims.shims.SelectionPreferences* attribute), 37, 115, 197, 260

format_debug() (*pip_shims.InstallRequirement* method), 61, 221

format_debug() (*pip_shims.shims.InstallRequirement* method), 30, 107, 190, 253

format_description() (*pip_shims.ConfigOptionParser* method), 57, 217

format_description() (*pip_shims.shims.ConfigOptionParser* method), 26, 103, 186, 249

format_epilog() (*pip_shims.ConfigOptionParser* method), 57, 217

format_epilog() (*pip_shims.shims.ConfigOptionParser* method), 26, 103, 186, 249

format_given() (*pip_shims.shims.TargetPython* method), 37, 114, 197, 259

format_given() (*pip_shims.TargetPython* method), 67, 227

format_help() (*pip_shims.ConfigOptionParser* method), 57, 217

format_help() (*pip_shims.shims.ConfigOptionParser* method), 26, 103, 186, 249

format_option_help() (*pip_shims.ConfigOptionParser* method), 57, 217

format_option_help() (*pip_shims.shims.ConfigOptionParser* method), 26, 103, 186, 249

FormatControl (class in *pip_shims*), 58, 218

FormatControl (class in *pip_shims.shims*), 27, 105, 187, 250

from_dist() (*pip_shims.FrozenRequirement* class method), 58, 218

from_dist() (*pip_shims.shims.FrozenRequirement* class method), 28, 105, 188, 250

from_dist() (*pip_shims.shims.UninstallPathSet* class method), 42, 119, 202, 264

from_dist() (*pip_shims.UninstallPathSet* class method), 72, 232

from_editable (*pip_shims.InstallRequirement* attribute), 61, 221

from_editable (*pip_shims.shims.InstallRequirement* attribute), 30, 107, 190, 253

from_line (*pip_shims.InstallRequirement* attribute), 61, 221

from_line (*pip_shims.shims.InstallRequirement* attribute), 30, 107, 190, 253

from_path() (*pip_shims.InstallRequirement* method), 61, 221

from_path() (*pip_shims.shims.InstallRequirement*

method), 30, 107, 190, 253
 FrozenRequirement (class in pip_shims), 58, 218
 FrozenRequirement (class in pip_shims.shims), 28, 105, 188, 250
 FUNCTION (pip_shims.models.ImportTypes attribute), 5, 98, 166, 279

G

get () (pip_shims.SafeFileCache method), 72, 232
 get () (pip_shims.shims.SafeFileCache method), 41, 119, 201, 264
 get () (pip_shims.shims.WheelCache method), 43, 120, 203, 266
 get () (pip_shims.WheelCache method), 74, 234
 get_allowed_args () (in module pip_shims.utils), 22, 134, 182, 300
 get_allowed_formats () (pip_shims.FormatControl method), 58, 218
 get_allowed_formats () (pip_shims.shims.FormatControl method), 27, 105, 187, 250
 get_applicable_candidates () (pip_shims.CandidateEvaluator method), 66, 226
 get_applicable_candidates () (pip_shims.shims.CandidateEvaluator method), 35, 112, 195, 258
 get_backend () (pip_shims.shims.VcsSupport method), 42, 119, 202, 265
 get_backend () (pip_shims.VcsSupport method), 73, 233
 get_backend_for_dir () (pip_shims.shims.VcsSupport method), 42, 119, 202, 265
 get_backend_for_dir () (pip_shims.VcsSupport method), 73, 233
 get_backend_for_scheme () (pip_shims.shims.VcsSupport method), 42, 119, 202, 265
 get_backend_for_scheme () (pip_shims.VcsSupport method), 73, 233
 get_base_import_path () (in module pip_shims.environment), 55, 98, 215, 297
 get_cache_entry () (pip_shims.shims.WheelCache method), 43, 120, 203, 266
 get_cache_entry () (pip_shims.WheelCache method), 74, 234
 get_default_session () (pip_shims.Command method), 55, 215
 get_default_session () (pip_shims.SessionCommandMixin method), 55, 215
 get_default_session ()

(pip_shims.shims.Command method), 25, 102, 185, 247
 get_default_session () (pip_shims.shims.SessionCommandMixin method), 24, 101, 184, 247
 get_default_values () (pip_shims.ConfigOptionParser method), 57, 217
 get_default_values () (pip_shims.shims.ConfigOptionParser method), 26, 103, 186, 249
 get_description () (pip_shims.ConfigOptionParser method), 57, 217
 get_description () (pip_shims.shims.ConfigOptionParser method), 26, 103, 186, 249
 get_dist () (pip_shims.InstallRequirement method), 61, 221
 get_dist () (pip_shims.shims.InstallRequirement method), 30, 107, 190, 253
 get_ephem_path_for_link () (pip_shims.shims.WheelCache method), 43, 120, 203, 266
 get_ephem_path_for_link () (pip_shims.WheelCache method), 74, 234
 get_formatted_file_tags () (pip_shims.compat.Wheel method), 9, 86, 169, 284
 get_formatted_file_tags () (pip_shims.shims.Wheel method), 42, 120, 202, 265
 get_formatted_file_tags () (pip_shims.Wheel method), 73, 233
 get_formatted_locations () (pip_shims.SearchScope method), 68, 228
 get_formatted_locations () (pip_shims.shims.SearchScope method), 37, 114, 197, 260
 get_index_urls_locations () (pip_shims.SearchScope method), 68, 228
 get_index_urls_locations () (pip_shims.shims.SearchScope method), 37, 114, 197, 260
 get_install_candidate () (pip_shims.PackageFinder method), 65, 225
 get_install_candidate () (pip_shims.shims.PackageFinder method), 34, 111, 194, 257
 get_installation_order () (pip_shims.Resolver method), 72, 232
 get_installation_order () (pip_shims.shims.Resolver method), 41,

- 118, 201, 264
- `get_installed_distributions()` (in module *pip_shims*), 58, 218
- `get_installed_distributions()` (in module *pip_shims.shims*), 28, 105, 188, 251
- `get_ireq_output_path()` (in module *pip_shims.compat*), 11, 88, 171, 286
- `get_method_args()` (in module *pip_shims.utils*), 22, 134, 182, 300
- `get_option()` (*pip_shims.ConfigOptionParser* method), 57, 217
- `get_option()` (*pip_shims.shims.ConfigOptionParser* method), 26, 103, 186, 249
- `get_option_group()` (*pip_shims.ConfigOptionParser* method), 57, 217
- `get_option_group()` (*pip_shims.shims.ConfigOptionParser* method), 26, 104, 186, 249
- `get_package_finder()` (in module *pip_shims*), 75, 235
- `get_package_finder()` (in module *pip_shims.compat*), 11, 88, 171, 286
- `get_package_finder()` (in module *pip_shims.shims*), 44, 122, 204, 267
- `get_path_for_link()` (*pip_shims.shims.WheelCache* method), 43, 120, 203, 266
- `get_path_for_link()` (*pip_shims.WheelCache* method), 74, 234
- `get_path_for_link_legacy()` (*pip_shims.shims.WheelCache* method), 43, 120, 203, 266
- `get_path_for_link_legacy()` (*pip_shims.WheelCache* method), 74, 234
- `get_pip_version()` (in module *pip_shims.environment*), 55, 98, 215, 297
- `get_pkg_resources_distribution()` (*pip_shims.AbstractDistribution* method), 74, 234
- `get_pkg_resources_distribution()` (*pip_shims.InstalledDistribution* method), 74, 234
- `get_pkg_resources_distribution()` (*pip_shims.shims.AbstractDistribution* method), 43, 121, 203, 266
- `get_pkg_resources_distribution()` (*pip_shims.shims.InstalledDistribution* method), 44, 121, 204, 267
- `get_pkg_resources_distribution()` (*pip_shims.shims.SourceDistribution* method), 44, 121, 204, 267
- `get_pkg_resources_distribution()` (*pip_shims.shims.WheelDistribution* method), 44, 121, 204, 267
- `get_pkg_resources_distribution()` (*pip_shims.SourceDistribution* method), 75, 235
- `get_pkg_resources_distribution()` (*pip_shims.WheelDistribution* method), 75, 235
- `get_prog_name()` (*pip_shims.ConfigOptionParser* method), 57, 217
- `get_prog_name()` (*pip_shims.shims.ConfigOptionParser* method), 26, 104, 186, 249
- `get_registry()` (*pip_shims.models.ShimmedPathCollection* class method), 8, 101, 168, 281
- `get_requirement()` (*pip_shims.RequirementSet* method), 70, 230
- `get_requirement()` (*pip_shims.shims.RequirementSet* method), 39, 116, 199, 262
- `get_requirement_set()` (in module *pip_shims*), 80, 240
- `get_requirement_set()` (in module *pip_shims.compat*), 12, 90, 172, 287
- `get_requirement_set()` (in module *pip_shims.shims*), 49, 126, 209, 272
- `get_requirement_tracker()` (in module *pip_shims*), 71, 231
- `get_requirement_tracker()` (in module *pip_shims.compat*), 14, 91, 174, 288
- `get_requirement_tracker()` (in module *pip_shims.shims*), 40, 117, 200, 263
- `get_resolver()` (in module *pip_shims*), 78, 238
- `get_resolver()` (in module *pip_shims.compat*), 14, 91, 174, 288
- `get_resolver()` (in module *pip_shims.shims*), 47, 124, 207, 270
- `get_session()` (in module *pip_shims.compat*), 16, 93, 176, 290
- `get_tags()` (*pip_shims.compat.TargetPython* method), 9, 86, 169, 284
- `get_tags()` (*pip_shims.shims.TargetPython* method), 37, 114, 197, 259
- `get_tags()` (*pip_shims.TargetPython* method), 67, 227
- `get_usage()` (*pip_shims.ConfigOptionParser* method), 57, 217
- `get_usage()` (*pip_shims.shims.ConfigOptionParser* method), 26, 104, 186, 249
- `get_version()` (*pip_shims.ConfigOptionParser* method), 57, 217
- `get_version()` (*pip_shims.shims.ConfigOptionParser* method), 26, 104, 186, 249
- `global_tempdir_manager()` (in module *pip_shims*), 70, 230
- `global_tempdir_manager()` (in module

pip_shims.shims), 39, 117, 199, 262

H

- `handle_mutual_excludes()` (*pip_shims.FormatControl* static method), 58, 218
- `handle_mutual_excludes()` (*pip_shims.shims.FormatControl* static method), 27, 105, 187, 250
- `handle_pip_version_check()` (*pip_shims.Command* method), 55, 215
- `handle_pip_version_check()` (*pip_shims.shims.Command* method), 25, 102, 185, 247
- `has_hash()` (*pip_shims.Link* attribute), 63, 223
- `has_hash()` (*pip_shims.shims.Link* attribute), 32, 109, 192, 255
- `has_hash_options()` (*pip_shims.InstallRequirement* attribute), 61, 221
- `has_hash_options()` (*pip_shims.shims.InstallRequirement* attribute), 30, 107, 190, 253
- `has_option()` (*pip_shims.ConfigOptionParser* method), 57, 217
- `has_option()` (*pip_shims.shims.ConfigOptionParser* method), 26, 104, 186, 249
- `has_property()` (in module *pip_shims.utils*), 23, 134, 183, 301
- `has_requirement()` (*pip_shims.RequirementSet* method), 70, 230
- `has_requirement()` (*pip_shims.shims.RequirementSet* method), 39, 116, 199, 262
- `hash()` (*pip_shims.Link* attribute), 63, 223
- `hash()` (*pip_shims.shims.Link* attribute), 32, 109, 192, 255
- `hash_name()` (*pip_shims.Link* attribute), 63, 223
- `hash_name()` (*pip_shims.shims.Link* attribute), 32, 109, 192, 255
- `hashes()` (*pip_shims.InstallRequirement* method), 61, 221
- `hashes()` (*pip_shims.shims.InstallRequirement* method), 30, 107, 190, 253
- ## I
- `ignore_require_venv()` (*pip_shims.Command* attribute), 55, 215
- `ignore_require_venv()` (*pip_shims.shims.Command* attribute), 25, 102, 185, 248
- `ignore_requires_python()` (*pip_shims.SelectionPreferences* attribute), 68, 228
- `ignore_requires_python()` (*pip_shims.shims.SelectionPreferences* attribute), 37, 115, 197, 260
- implementation (*pip_shims.shims.TargetPython* attribute), 37, 114, 197, 259
- implementation (*pip_shims.TargetPython* attribute), 67, 227
- `import_pip()` (in module *pip_shims.models*), 8, 101, 168, 282
- `ImportTypes` (class in *pip_shims.models*), 5, 98, 165, 279
- `ImportTypesBase` (in module *pip_shims.models*), 6, 99, 166, 279
- `index()` (*pip_shims.models.ImportTypes* method), 6, 99, 166, 279
- `index()` (*pip_shims.models.PipVersion* method), 6, 99, 166, 280
- `index()` (*pip_shims.models.PipVersionRange* method), 6, 99, 166, 280
- `index_urls` (*pip_shims.PackageFinder* attribute), 65, 225
- `index_urls` (*pip_shims.SearchScope* attribute), 68, 228
- `index_urls` (*pip_shims.shims.PackageFinder* attribute), 34, 111, 194, 257
- `index_urls` (*pip_shims.shims.SearchScope* attribute), 37, 114, 197, 260
- `insert_option_group()` (*pip_shims.ConfigOptionParser* method), 57, 217
- `insert_option_group()` (*pip_shims.shims.ConfigOptionParser* method), 26, 104, 186, 249
- `install()` (*pip_shims.InstallRequirement* method), 61, 221
- `install()` (*pip_shims.shims.InstallRequirement* method), 30, 108, 190, 253
- `install_req_from_editable()` (in module *pip_shims*), 60, 220
- `install_req_from_editable()` (in module *pip_shims.shims*), 29, 106, 189, 252
- `install_req_from_line()` (in module *pip_shims*), 60, 220
- `install_req_from_line()` (in module *pip_shims.shims*), 29, 106, 189, 252
- `install_req_from_req_string()` (in module *pip_shims*), 60, 220
- `install_req_from_req_string()` (in module *pip_shims.shims*), 29, 106, 189, 252
- `InstallationError`, 28, 59, 105, 188, 219, 251
- `installed_version` (*pip_shims.InstallRequirement* attribute), 61, 221
- `installed_version` (*pip_shims.shims.InstallRequirement* attribute), 30, 108, 190, 253
- `InstalledDistribution` (class in *pip_shims*), 74, 234

- InstalledDistribution (class in *pip_shims.shims*), 44, 121, 204, 266
 - InstallRequirement (class in *pip_shims*), 60, 220
 - InstallRequirement (class in *pip_shims.shims*), 29, 106, 189, 252
 - InvalidWheelFilename, 8, 86, 169, 283
 - is_archive_file() (in module *pip_shims*), 62, 222
 - is_archive_file() (in module *pip_shims.shims*), 31, 108, 191, 254
 - is_artifact (*pip_shims.Link* attribute), 63, 223
 - is_artifact (*pip_shims.shims.Link* attribute), 32, 109, 192, 255
 - is_attribute (*pip_shims.models.ShimmedPath* attribute), 7, 100, 167, 281
 - is_class (*pip_shims.models.ShimmedPath* attribute), 7, 100, 167, 281
 - is_contextmanager (*pip_shims.models.ShimmedPath* attribute), 7, 100, 167, 281
 - is_existing_dir() (*pip_shims.Link* method), 63, 223
 - is_existing_dir() (*pip_shims.shims.Link* method), 32, 109, 192, 255
 - is_file (*pip_shims.Link* attribute), 63, 223
 - is_file (*pip_shims.shims.Link* attribute), 32, 109, 192, 255
 - is_file_url() (in module *pip_shims*), 62, 222
 - is_file_url() (in module *pip_shims.shims*), 31, 108, 191, 254
 - is_function (*pip_shims.models.ShimmedPath* attribute), 7, 100, 167, 281
 - is_hash_allowed() (*pip_shims.Link* method), 63, 223
 - is_hash_allowed() (*pip_shims.shims.Link* method), 32, 110, 192, 255
 - is_installable_dir() (in module *pip_shims*), 62, 222
 - is_installable_dir() (in module *pip_shims.shims*), 31, 109, 191, 254
 - is_method (*pip_shims.models.ShimmedPath* attribute), 7, 100, 167, 281
 - is_module (*pip_shims.models.ShimmedPath* attribute), 7, 100, 167, 281
 - is_pinned (*pip_shims.InstallRequirement* attribute), 61, 221
 - is_pinned (*pip_shims.shims.InstallRequirement* attribute), 30, 108, 190, 253
 - is_type_checking() (in module *pip_shims.environment*), 55, 98, 215, 297
 - is_valid (*pip_shims.models.ShimmedPath* attribute), 7, 100, 167, 281
 - is_valid() (*pip_shims.models.PipVersion* method), 6, 99, 166, 280
 - is_valid() (*pip_shims.models.PipVersionRange* method), 6, 99, 167, 280
 - is_vcs (*pip_shims.Link* attribute), 63, 223
 - is_vcs (*pip_shims.shims.Link* attribute), 32, 110, 192, 255
 - is_wheel (*pip_shims.InstallRequirement* attribute), 61, 221
 - is_wheel (*pip_shims.Link* attribute), 63, 223
 - is_wheel (*pip_shims.shims.InstallRequirement* attribute), 30, 108, 190, 253
 - is_wheel (*pip_shims.shims.Link* attribute), 32, 110, 192, 255
 - is_yanked (*pip_shims.Link* attribute), 63, 223
 - is_yanked (*pip_shims.shims.Link* attribute), 32, 110, 192, 255
- ## L
- Link (class in *pip_shims*), 62, 222
 - Link (class in *pip_shims.shims*), 31, 109, 191, 254
 - LinkCollector (class in *pip_shims*), 66, 226
 - LinkCollector (class in *pip_shims.compat*), 9, 86, 169, 283
 - LinkCollector (class in *pip_shims.shims*), 35, 112, 195, 258
 - LinkEvaluator (class in *pip_shims*), 66, 226
 - LinkEvaluator (class in *pip_shims.compat*), 9, 86, 169, 283
 - LinkEvaluator (class in *pip_shims.shims*), 36, 113, 196, 258
 - load_pyproject_toml() (*pip_shims.InstallRequirement* method), 61, 221
 - load_pyproject_toml() (*pip_shims.shims.InstallRequirement* method), 30, 108, 190, 253
 - lookup_current_pip_version() (in module *pip_shims.models*), 8, 101, 168, 282
- ## M
- main() (*pip_shims.Command* method), 55, 215
 - main() (*pip_shims.shims.Command* method), 25, 102, 185, 248
 - main_context() (*pip_shims.Command* method), 56, 216
 - main_context() (*pip_shims.SessionCommandMixin* method), 55, 215
 - main_context() (*pip_shims.shims.Command* method), 25, 102, 185, 248
 - main_context() (*pip_shims.shims.SessionCommandMixin* method), 24, 101, 184, 247
 - make_abstract_dist() (in module *pip_shims*), 63, 223
 - make_abstract_dist() (in module *pip_shims.shims*), 33, 110, 193, 256

- make_candidate_evaluator() (*pip_shims.PackageFinder* method), 65, 225
 make_candidate_evaluator() (*pip_shims.shims.PackageFinder* method), 34, 111, 194, 257
 make_classmethod() (*in module pip_shims.utils*), 23, 134, 183, 301
 make_distribution_for_install_requirements() (*in module pip_shims*), 63, 223
 make_distribution_for_install_requirements() (*in module pip_shims.shims*), 33, 110, 193, 256
 make_link_evaluator() (*pip_shims.PackageFinder* method), 65, 225
 make_link_evaluator() (*pip_shims.shims.PackageFinder* method), 34, 111, 194, 257
 make_method() (*in module pip_shims.utils*), 23, 134, 183, 301
 make_option_group() (*in module pip_shims*), 64, 224
 make_option_group() (*in module pip_shims.shims*), 33, 110, 193, 256
 make_preparer() (*in module pip_shims*), 76, 236
 make_preparer() (*in module pip_shims.compat*), 16, 93, 176, 290
 make_preparer() (*in module pip_shims.shims*), 45, 123, 205, 268
 match_markers() (*pip_shims.InstallRequirement* method), 61, 221
 match_markers() (*pip_shims.shims.InstallRequirement* method), 31, 108, 191, 253
 memoize() (*in module pip_shims.utils*), 23, 134, 183, 301
 metadata (*pip_shims.InstallRequirement* attribute), 61, 221
 metadata (*pip_shims.shims.InstallRequirement* attribute), 31, 108, 191, 253
 METHOD (*pip_shims.models.ImportTypes* attribute), 6, 98, 166, 279
 MODULE (*pip_shims.models.ImportTypes* attribute), 6, 98, 166, 279
- N**
- name (*pip_shims.InstallRequirement* attribute), 61, 221
 name (*pip_shims.shims.InstallRequirement* attribute), 31, 108, 191, 253
 netloc (*pip_shims.Link* attribute), 63, 223
 netloc (*pip_shims.shims.Link* attribute), 32, 110, 192, 255
 no_binary (*pip_shims.FormatControl* attribute), 58, 218
 no_binary (*pip_shims.shims.FormatControl* attribute), 27, 105, 187, 250
 nullcontext() (*in module pip_shims.utils*), 23, 134, 183, 301
- O**
- only_binary (*pip_shims.FormatControl* attribute), 58, 218
 only_binary (*pip_shims.shims.FormatControl* attribute), 27, 105, 187, 250
 option_list_all (*pip_shims.ConfigOptionParser* attribute), 57, 217
 option_list_all (*pip_shims.shims.ConfigOptionParser* attribute), 26, 104, 186, 249
- P**
- PackageFinder (class *in pip_shims*), 64, 224
 PackageFinder (class *in pip_shims.shims*), 33, 110, 193, 256
 parse_args() (*pip_shims.Command* method), 56, 216
 parse_args() (*pip_shims.ConfigOptionParser* method), 57, 217
 parse_args() (*pip_shims.shims.Command* method), 25, 102, 185, 248
 parse_args() (*pip_shims.shims.ConfigOptionParser* method), 27, 104, 187, 249
 parse_requirements() (*in module pip_shims*), 68, 228
 parse_requirements() (*in module pip_shims.shims*), 37, 115, 197, 260
 parse_version() (*in module pip_shims.utils*), 23, 134, 183, 301
 partial_command() (*in module pip_shims.compat*), 17, 94, 177, 292
 path (*pip_shims.Link* attribute), 63, 223
 path (*pip_shims.shims.Link* attribute), 32, 110, 192, 255
 path (*pip_shims.shims.TempDirectory* attribute), 39, 117, 199, 262
 path (*pip_shims.TempDirectory* attribute), 70, 230
 path_to_url() (*in module pip_shims*), 69, 229
 path_to_url() (*in module pip_shims.shims*), 38, 115, 198, 261
 pip_shims (module), 5, 165
 pip_shims.compat (module), 8, 85, 168, 283
 pip_shims.environment (module), 55, 98, 215, 297
 pip_shims.models (module), 5, 98, 165, 279
 pip_shims.shims (module), 24, 101, 184, 247
 pip_shims.utils (module), 21, 132, 181, 299
 pip_version_lookup() (*in module pip_shims.models*), 8, 101, 168, 282
 PipError, 38, 69, 115, 198, 229, 261

PipVersion (class in *pip_shims.models*), 6, 99, 166, 280
 PipVersionRange (class in *pip_shims.models*), 6, 99, 166, 280
 platform (*pip_shims.shims.TargetPython* attribute), 37, 114, 197, 259
 platform (*pip_shims.TargetPython* attribute), 67, 227
 populate_options() (in module *pip_shims.compat*), 18, 95, 178, 292
 pre_shim() (*pip_shims.models.ShimmedPathCollection* method), 8, 101, 168, 281
 prefer_binary (*pip_shims.PackageFinder* attribute), 65, 225
 prefer_binary (*pip_shims.SelectionPreferences* attribute), 68, 228
 prefer_binary (*pip_shims.shims.PackageFinder* attribute), 34, 111, 194, 257
 prefer_binary (*pip_shims.shims.SelectionPreferences* attribute), 37, 115, 197, 260
 prepare_distribution_metadata() (*pip_shims.AbstractDistribution* method), 74, 234
 prepare_distribution_metadata() (*pip_shims.InstalledDistribution* method), 74, 234
 prepare_distribution_metadata() (*pip_shims.shims.AbstractDistribution* method), 44, 121, 204, 266
 prepare_distribution_metadata() (*pip_shims.shims.InstalledDistribution* method), 44, 121, 204, 267
 prepare_distribution_metadata() (*pip_shims.shims.SourceDistribution* method), 44, 121, 204, 267
 prepare_distribution_metadata() (*pip_shims.shims.WheelDistribution* method), 44, 121, 204, 267
 prepare_distribution_metadata() (*pip_shims.SourceDistribution* method), 75, 235
 prepare_distribution_metadata() (*pip_shims.WheelDistribution* method), 75, 235
 prepare_editable_requirement() (*pip_shims.RequirementPreparer* method), 69, 229
 prepare_editable_requirement() (*pip_shims.shims.RequirementPreparer* method), 38, 115, 198, 261
 prepare_installed_requirement() (*pip_shims.RequirementPreparer* method), 69, 229
 prepare_installed_requirement() (*pip_shims.shims.RequirementPreparer* method), 38, 116, 198, 261
 prepare_linked_requirement() (*pip_shims.RequirementPreparer* method), 69, 229
 prepare_linked_requirement() (*pip_shims.shims.RequirementPreparer* method), 38, 116, 198, 261
 prepare_metadata() (*pip_shims.InstallRequirement* method), 61, 221
 prepare_metadata() (*pip_shims.shims.InstallRequirement* method), 31, 108, 191, 254
 PreviousBuildDirError, 29, 60, 106, 189, 220, 252
 print_help() (*pip_shims.ConfigOptionParser* method), 58, 218
 print_help() (*pip_shims.shims.ConfigOptionParser* method), 27, 104, 187, 250
 print_usage() (*pip_shims.ConfigOptionParser* method), 58, 218
 print_usage() (*pip_shims.shims.ConfigOptionParser* method), 27, 104, 187, 250
 print_version() (*pip_shims.ConfigOptionParser* method), 58, 218
 print_version() (*pip_shims.shims.ConfigOptionParser* method), 27, 104, 187, 250
 process_project_url() (*pip_shims.PackageFinder* method), 65, 225
 process_project_url() (*pip_shims.shims.PackageFinder* method), 34, 111, 194, 257
 provide_function() (*pip_shims.models.ShimmedPathCollection* method), 8, 101, 168, 282
 provide_method() (*pip_shims.models.ShimmedPathCollection* method), 8, 101, 168, 282
 py_version (*pip_shims.shims.TargetPython* attribute), 37, 114, 197, 260
 py_version (*pip_shims.TargetPython* attribute), 67, 227
 py_version_info (*pip_shims.shims.TargetPython* attribute), 37, 114, 197, 260
 py_version_info (*pip_shims.TargetPython* attribute), 67, 227
 pyproject_toml_path (*pip_shims.InstallRequirement* attribute), 62, 222
 pyproject_toml_path (*pip_shims.shims.InstallRequirement* attribute), 31, 108, 191, 254

R

register() (*pip_shims.models.ShimmedPathCollection* method), 8, 101, 168, 282
 register() (*pip_shims.shims.VcsSupport* method), 42, 119, 202, 265
 register() (*pip_shims.VcsSupport* method), 73, 233
 remove() (*pip_shims.RequirementTracker* method), 70, 230
 remove() (*pip_shims.shims.RequirementTracker* method), 39, 116, 199, 262
 remove() (*pip_shims.shims.UninstallPathSet* method), 42, 119, 202, 265
 remove() (*pip_shims.UninstallPathSet* method), 72, 232
 remove_option() (*pip_shims.ConfigOptionParser* method), 58, 218
 remove_option() (*pip_shims.shims.ConfigOptionParser* method), 27, 104, 187, 250
 RequirementPreparer (*class in pip_shims*), 69, 229
 RequirementPreparer (*class in pip_shims.shims*), 38, 115, 198, 261
 RequirementSet (*class in pip_shims*), 69, 229
 RequirementSet (*class in pip_shims.shims*), 38, 116, 198, 261
 RequirementsFileParseError, 28, 59, 105, 188, 219, 251
 RequirementTracker (*class in pip_shims*), 70, 230
 RequirementTracker (*class in pip_shims.shims*), 39, 116, 199, 262
 requires_python (*pip_shims.Link* attribute), 63, 223
 requires_python (*pip_shims.shims.Link* attribute), 32, 110, 192, 255
 resolve() (*in module pip_shims*), 81, 241
 resolve() (*in module pip_shims.compat*), 18, 95, 178, 292
 resolve() (*in module pip_shims.shims*), 50, 128, 210, 273
 resolve() (*pip_shims.Resolver* method), 72, 232
 resolve() (*pip_shims.shims.Resolver* method), 41, 118, 201, 264
 resolve_possible_shim() (*in module pip_shims.compat*), 20, 97, 180, 295
 resolve_possible_shim() (*in module pip_shims.utils*), 23, 134, 183, 301
 Resolver (*class in pip_shims*), 71, 231
 Resolver (*class in pip_shims.shims*), 40, 117, 200, 263
 rollback() (*pip_shims.shims.UninstallPathSet* method), 42, 119, 202, 265
 rollback() (*pip_shims.UninstallPathSet* method), 73, 233
 run() (*pip_shims.Command* method), 56, 216
 run() (*pip_shims.shims.Command* method), 25, 102, 185, 248

S

SafeFileCache (*class in pip_shims*), 72, 232
 SafeFileCache (*class in pip_shims.shims*), 41, 119, 201, 264
 scheme (*pip_shims.Link* attribute), 63, 223
 scheme (*pip_shims.shims.Link* attribute), 32, 110, 192, 255
 schemes (*pip_shims.shims.VcsSupport* attribute), 42, 119, 202, 265
 schemes (*pip_shims.VcsSupport* attribute), 73, 233
 search_scope (*pip_shims.PackageFinder* attribute), 65, 225
 search_scope (*pip_shims.shims.PackageFinder* attribute), 34, 111, 194, 257
 SearchScope (*class in pip_shims*), 68, 228
 SearchScope (*class in pip_shims.compat*), 9, 86, 169, 283
 SearchScope (*class in pip_shims.shims*), 37, 114, 197, 260
 SelectionPreferences (*class in pip_shims*), 68, 228
 SelectionPreferences (*class in pip_shims.compat*), 9, 86, 169, 283
 SelectionPreferences (*class in pip_shims.shims*), 37, 114, 197, 260
 SessionCommandMixin (*class in pip_shims*), 55, 215
 SessionCommandMixin (*class in pip_shims.shims*), 24, 101, 184, 247
 set() (*pip_shims.SafeFileCache* method), 72, 232
 set() (*pip_shims.shims.SafeFileCache* method), 41, 119, 201, 264
 set_allow_all_prereleases() (*pip_shims.PackageFinder* method), 65, 225
 set_allow_all_prereleases() (*pip_shims.shims.PackageFinder* method), 34, 111, 194, 257
 set_conflict_handler() (*pip_shims.ConfigOptionParser* method), 58, 218
 set_conflict_handler() (*pip_shims.shims.ConfigOptionParser* method), 27, 104, 187, 250
 set_default() (*pip_shims.ConfigOptionParser* method), 58, 218
 set_default() (*pip_shims.models.ShimmedPathCollection* method), 8, 101, 168, 282
 set_default() (*pip_shims.shims.ConfigOptionParser* method), 27, 104, 187, 250
 set_default_args() (*pip_shims.models.ShimmedPathCollection* method), 8, 101, 168, 282
 set_default_kwargs() (*in module pip_shims.utils*), 23, 134, 183, 301

set_defaults() (*pip_shims.ConfigOptionParser method*), 58, 218
 set_defaults() (*pip_shims.shims.ConfigOptionParser method*), 27, 104, 187, 250
 set_description() (*pip_shims.ConfigOptionParser method*), 58, 218
 set_description() (*pip_shims.shims.ConfigOptionParser method*), 27, 104, 187, 250
 set_prefer_binary() (*pip_shims.PackageFinder method*), 65, 225
 set_prefer_binary() (*pip_shims.shims.PackageFinder method*), 34, 111, 194, 257
 set_process_default_values() (*pip_shims.ConfigOptionParser method*), 58, 218
 set_process_default_values() (*pip_shims.shims.ConfigOptionParser method*), 27, 104, 187, 250
 set_usage() (*pip_shims.ConfigOptionParser method*), 58, 218
 set_usage() (*pip_shims.shims.ConfigOptionParser method*), 27, 104, 187, 250
 setup_py_path (*pip_shims.InstallRequirement attribute*), 62, 222
 setup_py_path (*pip_shims.shims.InstallRequirement attribute*), 31, 108, 191, 254
 shim() (*pip_shims.models.ShimmedPath method*), 7, 100, 167, 281
 shim() (*pip_shims.models.ShimmedPathCollection method*), 8, 101, 168, 282
 shim_attribute() (*pip_shims.models.ShimmedPath method*), 7, 100, 167, 281
 shim_class() (*pip_shims.models.ShimmedPath method*), 7, 100, 167, 281
 shim_contextmanager() (*pip_shims.models.ShimmedPath method*), 7, 100, 167, 281
 shim_function() (*pip_shims.models.ShimmedPath method*), 7, 100, 167, 281
 shim_module() (*pip_shims.models.ShimmedPath method*), 7, 100, 167, 281
 shim_unpack() (*in module pip_shims*), 70, 230
 shim_unpack() (*in module pip_shims.compat*), 20, 97, 180, 295
 shim_unpack() (*in module pip_shims.shims*), 39, 117, 199, 262
 shimmed (*pip_shims.models.ShimmedPath attribute*), 7, 100, 167, 281
 ShimmedPath (*class in pip_shims.models*), 7, 99, 167, 280
 ShimmedPathCollection (*class in pip_shims.models*), 7, 100, 167, 281
 pip_shims.models), 7, 100, 168, 281
 show_url (*pip_shims.Link attribute*), 63, 223
 show_url (*pip_shims.shims.Link attribute*), 33, 110, 193, 255
 sort_best_candidate() (*pip_shims.CandidateEvaluator method*), 66, 226
 sort_best_candidate() (*pip_shims.shims.CandidateEvaluator method*), 35, 112, 195, 258
 sort_order (*pip_shims.models.ShimmedPath attribute*), 7, 100, 168, 281
 SourceDistribution (*class in pip_shims*), 75, 235
 SourceDistribution (*class in pip_shims.shims*), 44, 121, 204, 267
 specifier (*pip_shims.InstallRequirement attribute*), 62, 222
 specifier (*pip_shims.shims.InstallRequirement attribute*), 31, 108, 191, 254
 split_package() (*in module pip_shims.utils*), 23, 134, 183, 301
 splitext() (*pip_shims.Link method*), 63, 223
 splitext() (*pip_shims.shims.Link method*), 33, 110, 193, 255
 standard_option_list (*pip_shims.ConfigOptionParser attribute*), 58, 218
 standard_option_list (*pip_shims.shims.ConfigOptionParser attribute*), 27, 104, 187, 250
 subdirectory_fragment (*pip_shims.Link attribute*), 63, 223
 subdirectory_fragment (*pip_shims.shims.Link attribute*), 33, 110, 193, 255
 support_index_min() (*pip_shims.compat.Wheel method*), 9, 86, 169, 284
 support_index_min() (*pip_shims.shims.Wheel method*), 42, 120, 202, 265
 support_index_min() (*pip_shims.Wheel method*), 73, 233
 supported() (*pip_shims.compat.Wheel method*), 9, 86, 169, 284
 supported() (*pip_shims.shims.Wheel method*), 42, 120, 202, 265
 supported() (*pip_shims.Wheel method*), 73, 233
 suppress_setattr() (*in module pip_shims.utils*), 23, 134, 183, 301

T

target_python (*pip_shims.PackageFinder attribute*), 65, 225
 target_python (*pip_shims.shims.PackageFinder attribute*), 34, 111, 194, 257
 TargetPython (*class in pip_shims*), 67, 227

- TargetPython (class in *pip_shims.compat*), 9, 86, 169, 284
- TargetPython (class in *pip_shims.shims*), 36, 113, 196, 259
- temp_environ() (in module *pip_shims.compat*), 21, 98, 181, 295
- TempDirectory (class in *pip_shims*), 70, 230
- TempDirectory (class in *pip_shims.shims*), 39, 116, 199, 262
- track() (*pip_shims.RequirementTracker* method), 70, 230
- track() (*pip_shims.shims.RequirementTracker* method), 39, 116, 199, 262
- traverse() (*pip_shims.models.ShimmedPathCollection* class method), 8, 101, 168, 282
- trusted_hosts (*pip_shims.PackageFinder* attribute), 65, 225
- trusted_hosts (*pip_shims.shims.PackageFinder* attribute), 34, 111, 194, 257
- ## U
- uninstall() (*pip_shims.InstallRequirement* method), 62, 222
- uninstall() (*pip_shims.shims.InstallRequirement* method), 31, 108, 191, 254
- UninstallationError, 28, 59, 105, 188, 219, 251
- UninstallPathSet (class in *pip_shims*), 72, 232
- UninstallPathSet (class in *pip_shims.shims*), 41, 119, 201, 264
- unpack_url() (in module *pip_shims*), 62, 222
- unpack_url() (in module *pip_shims.shims*), 31, 108, 191, 254
- unpacked_source_directory (*pip_shims.InstallRequirement* attribute), 62, 222
- unpacked_source_directory (*pip_shims.shims.InstallRequirement* attribute), 31, 108, 191, 254
- unregister() (*pip_shims.shims.VcsSupport* method), 42, 119, 202, 265
- unregister() (*pip_shims.VcsSupport* method), 73, 233
- update_editable() (*pip_shims.InstallRequirement* method), 62, 222
- update_editable() (*pip_shims.shims.InstallRequirement* method), 31, 108, 191, 254
- update_sys_modules() (*pip_shims.models.ShimmedPath* method), 7, 100, 168, 281
- url (*pip_shims.Link* attribute), 63, 223
- url (*pip_shims.shims.Link* attribute), 33, 110, 193, 255
- url_to_path() (in module *pip_shims*), 73, 233
- url_to_path() (in module *pip_shims.shims*), 42, 119, 202, 265
- url_without_fragment (*pip_shims.Link* attribute), 63, 223
- url_without_fragment (*pip_shims.shims.Link* attribute), 33, 110, 193, 255
- usage (*pip_shims.Command* attribute), 56, 216
- usage (*pip_shims.shims.Command* attribute), 25, 102, 185, 248
- ## V
- VcsSupport (class in *pip_shims*), 73, 233
- VcsSupport (class in *pip_shims.shims*), 42, 119, 202, 265
- vendor_import_paths (*pip_shims.models.PipVersionRange* attribute), 6, 99, 167, 280
- version_key (*pip_shims.models.PipVersion* attribute), 6, 99, 166, 280
- version_tuple (*pip_shims.models.PipVersion* attribute), 6, 99, 166, 280
- ## W
- warn_on_mismatching_name() (*pip_shims.InstallRequirement* method), 62, 222
- warn_on_mismatching_name() (*pip_shims.shims.InstallRequirement* method), 31, 108, 191, 254
- Wheel (class in *pip_shims*), 73, 233
- Wheel (class in *pip_shims.compat*), 9, 86, 169, 284
- Wheel (class in *pip_shims.shims*), 42, 120, 202, 265
- wheel_cache() (in module *pip_shims*), 75, 235
- wheel_cache() (in module *pip_shims.compat*), 21, 98, 181, 295
- wheel_cache() (in module *pip_shims.shims*), 44, 121, 204, 267
- wheel_file_re (*pip_shims.compat.Wheel* attribute), 9, 87, 169, 284
- wheel_file_re (*pip_shims.shims.Wheel* attribute), 43, 120, 203, 265
- wheel_file_re (*pip_shims.Wheel* attribute), 73, 233
- WheelCache (class in *pip_shims*), 73, 233
- WheelCache (class in *pip_shims.shims*), 43, 120, 203, 265
- WheelDistribution (class in *pip_shims*), 75, 235
- WheelDistribution (class in *pip_shims.shims*), 44, 121, 204, 267
- with_traceback() (*pip_shims.BadCommand* method), 59, 219
- with_traceback() (*pip_shims.BestVersionAlreadyInstalled* method), 59, 219
- with_traceback() (*pip_shims.CommandError* method), 60, 220

`with_traceback()` (*pip_shims.compat.InvalidWheelFilename method*), 9, 86, 169, 283

`with_traceback()` (*pip_shims.DistributionNotFound method*), 58, 218

`with_traceback()` (*pip_shims.InstallationError method*), 59, 219

`with_traceback()` (*pip_shims.PipError method*), 69, 229

`with_traceback()` (*pip_shims.PreviousBuildDirError method*), 60, 220

`with_traceback()` (*pip_shims.RequirementsFileParseError method*), 59, 219

`with_traceback()` (*pip_shims.shims.BadCommand method*), 29, 106, 189, 251

`with_traceback()` (*pip_shims.shims.BestVersionAlreadyInstalled method*), 28, 106, 188, 251

`with_traceback()` (*pip_shims.shims.CommandError method*), 29, 106, 189, 252

`with_traceback()` (*pip_shims.shims.DistributionNotFound method*), 27, 104, 187, 250

`with_traceback()` (*pip_shims.shims.InstallationError method*), 28, 105, 188, 251

`with_traceback()` (*pip_shims.shims.PipError method*), 38, 115, 198, 261

`with_traceback()` (*pip_shims.shims.PreviousBuildDirError method*), 29, 106, 189, 252

`with_traceback()` (*pip_shims.shims.RequirementsFileParseError method*), 28, 106, 188, 251

`with_traceback()` (*pip_shims.shims.UninstallationError method*), 28, 105, 188, 251

`with_traceback()` (*pip_shims.UninstallationError method*), 59, 219

Y

`yanked_reason` (*pip_shims.Link attribute*), 63, 223

`yanked_reason` (*pip_shims.shims.Link attribute*), 33, 110, 193, 256