
RequirementsLib Documentation

Release 1.5.17.dev0

Dan Ryan <dan@danryan.co>

November 12, 2020

Contents:

1	RequirementsLib: Requirement Management Library for Pip and Pipenv	1
1.1	Installation	1
1.2	Summary	1
1.3	Usage	2
1.3.1	Importing a lockfile into your <i>setup.py</i> file	2
1.3.2	Interacting with a <i>Pipfile</i> directly	2
1.3.3	Create a requirement object from <i>requirements.txt</i> format	2
1.3.4	Resolving Editable Package Dependencies	3
1.4	Integrations	4
1.5	Contributing	4
2	requirementslib package	7
2.1	Submodules	12
2.1.1	requirementslib.models package	12
2.1.1.1	Submodules	12
2.1.2	requirementslib.utils module	49
3	Indices and tables	53
	Python Module Index	55
	Index	57

RequirementsLib: Requirement Management Library for Pip and Pipenv

RequirementsLib: Requirement Management Library for Pip and Pipenv

1.1 Installation

Install from [PyPI](#):

```
$ pipenv install requirementslib
```

Install from [Github](#):

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

1.2 Summary

RequirementsLib provides a simple layer for building and interacting with requirements in both the [Pipfile](#) format and the [requirements.txt](#) format. This library was originally built for converting between these formats in [Pipenv](#).

1.3 Usage

1.3.1 Importing a lockfile into your *setup.py* file

You can use RequirementsLib to import your lockfile into your setup file for including your **install_requires** dependencies:

```
from requirementslib import Lockfile
lockfile = Lockfile.create('/path/to/project/dir')
install_requires = lockfile.as_requirements(dev=False)
```

1.3.2 Interacting with a *Pipfile* directly

You can also interact directly with a Pipfile:

```
>>> from requirementslib import Pipfile
>>> pf = Pipfile.load('/home/hawk/git/pypa-pipenv')
>>> pf.sections
[Section(name='packages', requirements=[]), Section(name='dev-packages',
↳requirements=[Requirement(name='pipenv', vcs=None, req=FileRequirement(setup_
↳path=None, path='.', editable=True, uri='file:///home/hawk/git/pypa-pipenv', link=
↳<Link file:///home/hawk/git/pypa-pipenv>, name='pipenv', req=<Requirement: "-e_
↳file:///home/hawk/git/pypa-pipenv">), markers='', specifiers=None, index=None,
↳editable=True, hashes=[], extras=None),...]
```

And you can even write it back out into Pipfile's native format:

```
>>> print(pf.dump(to_dict=False))
[packages]

[dev-packages]
pipenv = {path = ".", editable = true}
flake8 = ">=3.3.0,<4"
pytest = "*"
mock = "*"

[scripts]
tests = "bash ./run-tests.sh"

[pipenv]
allow_prereleases = true
```

1.3.3 Create a requirement object from *requirements.txt* format

```
>>> from requirementslib import Requirement
>>> r = Requirement.from_line('-e git+https://github.com/pypa/pipenv.git@master
↳#egg=pipenv')
>>> print(r)
Requirement(name='pipenv', vcs='git', req=VCSRequirement(editable=True, uri=
↳'git+https://github.com/pypa/pipenv.git', path=None, vcs='git', ref='master',
↳subdirectory=None, name='pipenv', link=<Link git+https://github.com/pypa/pipenv.
↳git@master#egg=pipenv>, req=<Requirement: "-e git+https://github.com/pypa/pipenv.
↳git@master#egg=pipenv">), markers=None, specifiers=None, index=None, editable=True,
↳hashes=[], extras=[])
```

(continues on next page)

(continued from previous page)

```
>>> r.as_pipfile()
{'pipenv': {'editable': True, 'ref': 'master', 'git': 'https://github.com/pypa/pipenv.'
↳git'}}
```

Or move from *Pipfile* format to *requirements.txt*:

```
>>> r = Requirement.from_pipfile(name='pythonfinder', indexes=[], pipfile={'path': '..
↳pythonfinder', 'editable': True})
>>> r.as_line()
'-e ../pythonfinder'
```

1.3.4 Resolving Editable Package Dependencies

Requirementslib also can resolve the dependencies of editable packages by calling the `run_requires` method. This method returns a detailed dictionary containing metadata parsed from the package built in a transient folder (unless it is already on the system or the call is run in a virtualenv).

The output of `run_requires` is very detailed and in most cases will be sufficient:

```
>>> from pprint import pprint
>>> from requirementslib.models.requirements import Requirement
>>> r = Requirement.from_line("-e git+git@github.com:sarugaku/vistir.git
↳#egg=vistir[spinner]")
>>> setup_info_dict = r.run_requires()
>>> from pprint import pprint
>>> pprint(setup_info_dict)
{'base_dir': '/tmp/requirementslib-t_ftl6no-src/src/vistir',
'build_backend': 'setuptools.build_meta',
'build_requires': ['setuptools>=36.2.2', 'wheel>=0.28.0'],
'extra_kwargs': {'build_dir': '/tmp/requirementslib-t_ftl6no-src/src',
                  'download_dir': '/home/hawk/.cache/pipenv/pkgs',
                  'src_dir': '/tmp/requirementslib-t_ftl6no-src/src',
                  'wheel_download_dir': '/home/hawk/.cache/pipenv/wheels'},
'extras': {'spinner': [Requirement.parse('cursor'),
                       Requirement.parse('yaspin')],
            'tests': [Requirement.parse('pytest'),
                      Requirement.parse('pytest-xdist'),
                      Requirement.parse('pytest-cov'),
                      Requirement.parse('pytest-timeout'),
                      Requirement.parse('hypothesis-fspaths'),
                      Requirement.parse('hypothesis')]},
'ireq': <InstallRequirement object: vistir[spinner] from git+ssh://git@github.com/
↳sarugaku/vistir.git#egg=vistir editable=True>,
'name': 'vistir',
'pyproject': PosixPath('/tmp/requirementslib-t_ftl6no-src/src/vistir/pyproject.toml'),
'python_requires': '>=2.6, !=3.0, !=3.1, !=3.2, !=3.3',
'requires': {'backports.functools_lru_cache;python_version<="3.4"': Requirement.parse(
↳backports.functools_lru_cache; python_version <= "3.4"),
              'backports.shutil_get_terminal_size;python_version<"3.3"': Requirement.
↳parse('backports.shutil_get_terminal_size; python_version < "3.3"),
              'backports.weakref;python_version<"3.3"': Requirement.parse('backports.
↳weakref; python_version < "3.3"),
              'colorama': Requirement.parse('colorama'),
              'pathlib2;python_version<"3.5"': Requirement.parse('pathlib2; python_
↳version < "3.5"),
```

(continues on next page)

(continued from previous page)

```
'requests': Requirement.parse('requests'),
'six': Requirement.parse('six'),
'spinner': [Requirement.parse('cursor'),
             Requirement.parse('yaspin')]],
'setup_cfg': PosixPath('/tmp/requirementslib-t_ftl6no-src/src/vistir/setup.cfg'),
'setup_py': PosixPath('/tmp/requirementslib-t_ftl6no-src/src/vistir/setup.py')}
```

As a side-effect of calls to `run_requires`, new metadata is made available on the requirement itself via the property `requirement.req.dependencies`:

```
>>> pprint(r.req.dependencies)
({ 'backports.functools_lru_cache;python_version<="3.4"': Requirement.parse('backports.
↳functools_lru_cache; python_version <= "3.4"'),
'backports.shutil_get_terminal_size;python_version<"3.3"': Requirement.parse(
↳'backports.shutil_get_terminal_size; python_version < "3.3"'),
'backports.weakref;python_version<"3.3"': Requirement.parse('backports.weakref;
↳python_version < "3.3"'),
'colorama': Requirement.parse('colorama'),
'pathlib2;python_version<"3.5"': Requirement.parse('pathlib2; python_version < "3.5"
↳'),
'requests': Requirement.parse('requests'),
'six': Requirement.parse('six'),
'spinner': [Requirement.parse('cursor'), Requirement.parse('yaspin')]],
[],
['setuptools>=36.2.2', 'wheel>=0.28.0'])
```

1.4 Integrations

- [Pip](#)
- [Pipenv](#)
- [Pipfile](#)

1.5 Contributing

1. Fork the repository and clone the fork to your local machine: `git clone git@github.com:yourusername/requirementslib.git`
2. Move into the repository directory and update the submodules: `git submodule update --init --recursive`
3. Install the package locally in a virtualenv using [pipenv](#): `pipenv install --dev`
 - a. You can also install the package into a [virtualenv](#) by running `pip install -e .[dev,tests,typing]` to ensure all the development and test dependencies are installed
4. Before making any changes to the code, make sure to file an issue. The best way to ensure a smooth collaboration is to communicate *before* investing significant time and energy into any changes! Make sure to consider not just your own use case but others who might be using the library
5. Create a new branch. For bugs, you can simply branch to `bugfix/<issuenumber>`. Features can be branched to `feature/<issuenumber>`. This convention is to streamline the branching process and to

encourage good practices around filing issues and associating pull requests with specific issues. If you find yourself addressing many issues in one pull request, that should give you pause

6. Make your desired changes. Don't forget to add additional tests to account for your new code – continuous integration **will** fail without it
7. Test your changes by running `pipenv run pytest -ra tests` or simply `pytest -ra tests` if you are inside an activated virtual environment
8. Create a corresponding `.rst` file in the `news` directory with a one sentence description of your change, e.g. Resolved an issue which sometimes prevented requirements from being converted from Pipfile entries to pip lines correctly
9. Commit your changes. The first line of your commit should be a summary of your changes, no longer than 72 characters, followed by a blank line, followed by a bulleted description of your changes. Don't forget to add separate lines with the phrase – Fixes #<issuenumber> for each issue you are addressing in your pull request
10. Before submitting your pull request, make sure to `git remote add upstream git@github.com:sarugaku/requirementslib.git` and then `git fetch upstream && git pull upstream master` to ensure your code is in sync with the latest version of the master branch,
11. Create a pull request describing your fix, referencing the issues in question. If your commit message from step 8 was detailed, you should be able to copy and paste it

requirementslib package

```
class requirementslib.Lockfile (path: pathlib.Path = NOTHING, requirements: list = NOTHING, dev_requirements: list = NOTHING, projectfile: requirementslib.models.project.ProjectFile = NOTHING, lockfile: plette.lockfiles.Lockfile = NOTHING, newlines: str = 'n')
```

Bases: `object`

```
as_requirements (include_hashes=False, dev=False)
```

Returns a list of requirements in pip-style format

```
classmethod create (path, create=True)
```

```
default
```

```
dev_requirements
```

```
dev_requirements_list
```

```
develop
```

```
extended_keys
```

```
classmethod from_data (path, data, meta_from_project=True)
```

Create a new lockfile instance from a dictionary.

Parameters

- **path** (*str*) – Path to the project root.
- **data** (*dict*) – Data to load into the lockfile.
- **meta_from_project** (*bool*) – Attempt to populate the meta section from the project root, default True.

```
get (k)
```

```
get_deps (dev=False, only=True)
```

```
get_requirements (dev=True, only=False)
```

Produces a generator which generates requirements from the desired section.

Parameters `dev` (*bool*) – Indicates whether to use dev requirements, defaults to False

Returns Requirements from the relevant the relevant pipfile

Return type *Requirement*

classmethod `load` (*path*, *create=True*)

Create a new lockfile instance.

Parameters

- **project_path** (*str* or `pathlib.Path`) – Path to project root or lockfile
- **lockfile_name** (*str*) – Name of the lockfile in the project root directory
- **pipfile_path** (`pathlib.Path`) – Path to the project pipfile

Returns A new lockfile representing the supplied project paths

Return type *Lockfile*

classmethod `load_projectfile` (*path*, *create=True*, *data=None*)

Given a path, load or create the necessary lockfile.

Parameters

- **path** (*str*) – Path to the project root or lockfile
- **create** (*bool*) – Whether to create the lockfile if not found, defaults to True

Raises

- **OSError** – Thrown if the project root directory doesn't exist
- **FileNotFoundError** – Thrown if the lockfile doesn't exist and `create=False`

Returns A project file instance for the supplied project

Return type *ProjectFile*

lockfile

classmethod `lockfile_from_pipfile` (*pipfile_path*)

newlines

path

projectfile

classmethod `read_projectfile` (*path*)

Read the specified project file and provide an interface for writing/updating.

Parameters **path** (*str*) – Path to the target file.

Returns A project file with the model and location for interaction

Return type *ProjectFile*

requirements

requirements_list

section_keys

write ()

```
class requirementslib.Pipfile(path: pathlib.Path = NOTHING, projectfile: require-
                             mentslib.models.project.ProjectFile = NOTHING, pipfile:
                             requirementslib.models.pipfile.PipfileLoader = NOTHING, pypro-
                             ject: tomlkit.toml_document.TOMLDocument = NOTHING,
                             build_system: dict = NOTHING, requirements: list = NOTHING,
                             dev_requirements: list = NOTHING)
```

Bases: `object`

`allow_prereleases`

`build_backend`

`build_requires`

`build_system`

`dev_packages`

`dev_requirements`

`extended_keys`

`get` (*k*)

`get_deps` (*dev=False*, *only=True*)

classmethod `load` (*path*, *create=False*)

Given a path, load or create the necessary pipfile.

Parameters

- **path** (*Text*) – Path to the project root or pipfile
- **create** (*bool*) – Whether to create the pipfile if not found, defaults to True

Raises

- **OSError** – Thrown if the project root directory doesn't exist
- **FileNotFoundError** – Thrown if the pipfile doesn't exist and *create=False*

Returns A pipfile instance pointing at the supplied project

`:rtype:: class:~requirementslib.models.pipfile.Pipfile`

classmethod `load_projectfile` (*path*, *create=False*)

Given a path, load or create the necessary pipfile.

Parameters

- **path** (*Text*) – Path to the project root or pipfile
- **create** (*bool*) – Whether to create the pipfile if not found, defaults to True

Raises

- **OSError** – Thrown if the project root directory doesn't exist
- **FileNotFoundError** – Thrown if the pipfile doesn't exist and *create=False*

Returns A project file instance for the supplied project

Return type `ProjectFile`

`packages`

`path`

`pipfile`

projectfile

classmethod read_projectfile (*path*)

Read the specified project file and provide an interface for writing/updating.

Parameters *path* (*Text*) – Path to the target file.

Returns A project file with the model and location for interaction

Return type *ProjectFile*

requirements

requires_python

root

write ()

class requirementslib.**Requirement** (*name=NOTHING*, *vcs=None*, *req=None*, *markers=None*, *specifiers=NOTHING*, *index=None*, *editable=None*, *hashes=NOTHING*, *extras=NOTHING*, *abstract_dep=None*, *line_instance=None*, *ireq=None*)

Bases: *object*

add_hashes (*hashes*)

as_ireq ()

as_line (*sources=None*, *include_hashes=True*, *include_extras=True*, *include_markers=True*, *as_list=False*)

Format this requirement as a line in requirements.txt.

If *sources* provided, it should be an sequence of mappings, containing all possible sources to be used for this requirement.

If *sources* is omitted or falsy, no index information will be included in the requirement line.

as_pipfile ()

build_backend

commit_hash

constraint_line

copy ()

extras_as_pip

find_all_matches (*sources=None*, *finder=None*)

Find all matching candidates for the current requirement.

Consults a finder to find all matching candidates.

Parameters

- **sources** – Pipfile-formatted sources, defaults to None
- **sources** – list[dict], optional
- **finder** (*PackageFinder*) – A **PackageFinder** instance from pip’s repository implementation

Returns A list of Installation Candidates

Return type list[*InstallationCandidate*]

classmethod from_ireq (*ireq*)

classmethod `from_line(line)`

classmethod `from_metadata(name, version, extras, markers)`

classmethod `from_pipfile(name, pipfile)`

get_abstract_dependencies (*sources=None*)

Retrieve the abstract dependencies of this requirement.

Returns the abstract dependencies of the current requirement in order to resolve.

Parameters

- **sources** – A list of sources (pipfile format), defaults to None
- **sources** – list, optional

Returns A list of abstract (unpinned) dependencies

Return type list[AbstractDependency]

get_dependencies (*sources=None*)

Retrieve the dependencies of the current requirement.

Retrieves dependencies of the current requirement. This only works on pinned requirements.

Parameters

- **sources** – Pipfile-formatted sources, defaults to None
- **sources** – list[dict], optional

Returns A set of requirement strings of the dependencies of this requirement.

Return type set(str)

get_hashes_as_pip (*as_list=False*)

get_line_instance ()

get_markers ()

get_name ()

get_requirement ()

get_specifier ()

get_specifiers ()

get_version ()

hashes_as_pip

ireq

is_direct_url

is_file_or_url

is_named

is_vcs

is_wheel

line_instance

markers_as_pip

merge_markers (*markers*)

name
normalized_name
pipfile_entry
requirement
run_requires (*sources=None, finder=None*)
specifiers
update_name_from_path (*path*)
uses_pep517

2.1 Submodules

2.1.1 requirementslib.models package

2.1.1.1 Submodules

requirementslib.models.cache module

exception requirementslib.models.cache.**CorruptCacheError** (*path*)
Bases: `Exception`

class requirementslib.models.cache.**DependencyCache** (*cache_dir=None*)
Bases: `object`

Creates a new persistent dependency cache for the current Python version. The cache file is written to the appropriate user cache dir for the current platform, i.e.

`~/.cache/pip-tools/depcache-pyX.Y.json`

Where X.Y indicates the Python version.

as_cache_key (*ireq*)

Given a requirement, return its cache key. This behavior is a little weird in order to allow backwards compatibility with cache files. For a requirement without extras, this will return, for example:

`("python", "2.1.0")`

For a requirement with extras, the extras will be comma-separated and appended to the version, inside brackets, like so:

`("python", "2.1.0[nbconvert,notebook]")`

cache

The dictionary that is the actual in-memory cache. This property lazily loads the cache from disk.

clear ()

get (*ireq, default=None*)

read_cache ()

Reads the cached contents into memory.

reverse_dependencies (*ireqs*)

Returns a lookup table of reverse dependencies for all the given ireqs.

Since this is all static, it only works if the dependency cache contains the complete data, otherwise you end up with a partial view. This is typically no problem if you use this function after the entire dependency tree is resolved.

write_cache()

Writes the cache to disk as JSON.

class requirementslib.models.cache.**HashCache** (*args, **kwargs)

Bases: `pip._internal.network.cache.SafeFileCache`

Caches hashes of PyPI artifacts so we do not need to re-download them.

Hashes are only cached when the URL appears to contain a hash in it and the cache key includes the hash value returned from the server). This ought to avoid issues where the location on the server changes.

get_hash (location)

class requirementslib.models.cache.**RequiresPythonCache** (cache_dir='/home/docs/.cache/pipenv')

Bases: `requirementslib.models.cache._JSONCache`

Cache a candidate's Requires-Python information.

filename_format = 'pyreqcache-py{python_version}.json'

`requirementslib.models.cache.read_cache_file` (cache_file_path)

requirementslib.models.dependencies module

class requirementslib.models.dependencies.**AbstractDependency** (name, specifiers, markers, candidates, requirement, parent, finder, dep_dict=NOTHING)

Bases: `object`

compatible_abstract_dep (other)

Merge this abstract dependency with another one.

Return the result of the merge as a new abstract dependency.

Parameters *other* (`AbstractDependency`) – An abstract dependency to merge with

Returns A new, combined abstract dependency

Return type `AbstractDependency`

compatible_versions (other)

Find compatible version numbers between this abstract dependency and another one.

Parameters *other* (`AbstractDependency`) – An abstract dependency to compare with.

Returns A set of compatible version strings

Return type `set(str)`

classmethod **from_requirement** (requirement, parent=None)

Creates a new `AbstractDependency` from a `Requirement` object.

This class is used to find all candidates matching a given set of specifiers and a given requirement.

Parameters *requirement* (`Requirement` object.) – A requirement for resolution

classmethod **from_string** (line, parent=None)

get_deps (*candidate*)

Get the dependencies of the supplied candidate.

Parameters **candidate** (`InstallRequirement`) – An installrequirement

Returns A list of abstract dependencies

Return type `list[AbstractDependency]`

version_set

Return the set of versions for the candidates in this abstract dependency.

Returns A set of matching versions

Return type `set(str)`

`requirementslib.models.dependencies.find_all_matches` (*finder, ireq, pre=False*)

Find all matching dependencies using the supplied finder and the given ireq.

Parameters

- **finder** (`PackageFinder`) – A package finder for discovering matching candidates.
- **ireq** (`InstallRequirement`) – An install requirement.

Returns A list of matching candidates.

Return type `list[InstallationCandidate]`

`requirementslib.models.dependencies.get_abstract_dependencies` (*reqs, sources=None, parent=None*)

Get all abstract dependencies for a given list of requirements.

Given a set of requirements, convert each requirement to an Abstract Dependency.

Parameters

- **reqs** (`list[Requirement]`) – A list of Requirements
- **sources** – Pipfile-formatted sources, defaults to None
- **sources** – `list[dict]`, optional
- **parent** – The parent of this list of dependencies, defaults to None
- **parent** – `Requirement`, optional

Returns A list of Abstract Dependencies

Return type `list[AbstractDependency]`

`requirementslib.models.dependencies.get_dependencies` (*ireq, sources=None, parent=None*)

Get all dependencies for a given install requirement.

Parameters

- **ireq** (`InstallRequirement`) – A single InstallRequirement
- **sources** (`list[dict]`, *optional*) – Pipfile-formatted sources, defaults to None
- **parent** (`InstallRequirement`) – The parent of this list of dependencies, defaults to None

Returns A set of dependency lines for generating new InstallRequirements.

Return type `set(str)`

`requirementslib.models.dependencies.get_dependencies_from_cache(ireq)`

Retrieves dependencies for the given install requirement from the dependency cache.

Parameters `ireq` (`InstallRequirement`) – A single `InstallRequirement`

Returns A set of dependency lines for generating new `InstallRequirements`.

Return type `set(str)` or `None`

`requirementslib.models.dependencies.get_dependencies_from_index(dep,`
`sources=None,`
`pip_options=None,`
`wheel_cache=None)`

Retrieves dependencies for the given install requirement from the pip resolver.

Parameters

- **dep** (`InstallRequirement`) – A single `InstallRequirement`
- **sources** (`list[dict]`, *optional*) – Pipfile-formatted sources, defaults to `None`

Returns A set of dependency lines for generating new `InstallRequirements`.

Return type `set(str)` or `None`

`requirementslib.models.dependencies.get_dependencies_from_json(ireq)`

Retrieves dependencies for the given install requirement from the json api.

Parameters `ireq` (`InstallRequirement`) – A single `InstallRequirement`

Returns A set of dependency lines for generating new `InstallRequirements`.

Return type `set(str)` or `None`

`requirementslib.models.dependencies.get_dependencies_from_wheel_cache(ireq)`

Retrieves dependencies for the given install requirement from the wheel cache.

Parameters `ireq` (`InstallRequirement`) – A single `InstallRequirement`

Returns A set of dependency lines for generating new `InstallRequirements`.

Return type `set(str)` or `None`

`requirementslib.models.dependencies.get_finder(sources=None, pip_command=None,`
`pip_options=None)`

Get a package finder for looking up candidates to install

Parameters

- **sources** – A list of pipfile-formatted sources, defaults to `None`
- **sources** – `list[dict]`, *optional*
- **pip_command** (`Command`) – A pip command instance, defaults to `None`
- **pip_options** (`cmdoptions`) – A pip options, defaults to `None`

Returns A package finder

Return type `PackageFinder`

`requirementslib.models.dependencies.get_grouped_dependencies(constraints)`

`requirementslib.models.dependencies.get_pip_command()`

`requirementslib.models.dependencies.get_pip_options(args=[],`
`sources=None,`
`pip_command=None)`

Build a pip command from a list of sources

Parameters

- **args** – positional arguments passed through to the pip parser
- **sources** – A list of pipfile-formatted sources, defaults to None
- **sources** – list[dict], optional
- **pip_command** (Command) – A pre-built pip command instance

Returns An instance of pip_options using the supplied arguments plus sane defaults

Return type cmdoptions

requirementslib.models.dependencies.**is_python** (*section*)

requirementslib.models.dependencies.**start_resolver** (*finder=None, session=None, wheel_cache=None*)

Context manager to produce a resolver.

Parameters **finder** (PackageFinder) – A package finder to use for searching the index

:param Session session: A session instance :param WheelCache wheel_cache: A pip WheelCache instance

:return: A 3-tuple of finder, preparer, resolver :type: (RequirementPreparer, Resolver)

requirementslib.models.lockfile module

```
class requirementslib.models.lockfile.Lockfile (path: pathlib.Path = NOTHING,  
                                              requirements: list = NOTHING,  
                                              dev_requirements: list =  
                                              NOTHING, projectfile: require-  
                                              mentslib.models.project.ProjectFile  
                                              = NOTHING, lockfile:  
                                              plette.lockfiles.Lockfile = NOTHING,  
                                              newlines: str = 'n')
```

Bases: object

as_requirements (*include_hashes=False, dev=False*)

Returns a list of requirements in pip-style format

classmethod create (*path, create=True*)

default

dev_requirements

dev_requirements_list

develop

extended_keys

classmethod from_data (*path, data, meta_from_project=True*)

Create a new lockfile instance from a dictionary.

Parameters

- **path** (*str*) – Path to the project root.
- **data** (*dict*) – Data to load into the lockfile.
- **meta_from_project** (*bool*) – Attempt to populate the meta section from the project root, default True.

get (*k*)

get_deps (*dev=False, only=True*)

get_requirements (*dev=True, only=False*)

Produces a generator which generates requirements from the desired section.

Parameters **dev** (*bool*) – Indicates whether to use dev requirements, defaults to False

Returns Requirements from the relevant the relevant pipfile

Return type *Requirement*

classmethod load (*path, create=True*)

Create a new lockfile instance.

Parameters

- **project_path** (*str* or *pathlib.Path*) – Path to project root or lockfile
- **lockfile_name** (*str*) – Name of the lockfile in the project root directory
- **pipfile_path** (*pathlib.Path*) – Path to the project pipfile

Returns A new lockfile representing the supplied project paths

Return type *Lockfile*

classmethod load_projectfile (*path, create=True, data=None*)

Given a path, load or create the necessary lockfile.

Parameters

- **path** (*str*) – Path to the project root or lockfile
- **create** (*bool*) – Whether to create the lockfile if not found, defaults to True

Raises

- **OSError** – Thrown if the project root directory doesn't exist
- **FileNotFoundError** – Thrown if the lockfile doesn't exist and *create=False*

Returns A project file instance for the supplied project

Return type *ProjectFile*

lockfile

classmethod lockfile_from_pipfile (*pipfile_path*)

newlines

path

projectfile

classmethod read_projectfile (*path*)

Read the specified project file and provide an interface for writing/updating.

Parameters **path** (*str*) – Path to the target file.

Returns A project file with the model and location for interaction

Return type *ProjectFile*

requirements

requirements_list

section_keys

```
write()
```

```
requirementslib.models.lockfile.preferred_newlines(f)
```

requirementslib.models.markers module

```
class requirementslib.models.markers.PipenvMarkers(os_name=None,
                                                    sys_platform=None,      plat-
                                                    form_machine=None,      plat-
                                                    form_python_implementation=None,
                                                    platform_release=None,
                                                    platform_system=None,
                                                    platform_version=None,
                                                    python_version=None,
                                                    python_full_version=None,
                                                    implementation_name=None,
                                                    implementation_version=None)
```

Bases: `object`

System-level requirements - see PEP508 for more detail

```
classmethod from_line(line)
```

```
classmethod from_pipfile(name, pipfile)
```

```
line_part
```

```
classmethod make_marker(marker_string)
```

```
pipfile_part
```

```
requirementslib.models.markers.cleanup_pyspecs
```

```
requirementslib.models.markers.contains_extra
```

Check whehter a marker contains an “extra == ...” operand.

```
requirementslib.models.markers.contains_pyversion
```

Check whether a marker contains a python_version operand.

```
requirementslib.models.markers.fix_version_tuple
```

```
requirementslib.models.markers.format_pyversion(parts)
```

```
requirementslib.models.markers.gen_marker(mkr)
```

```
requirementslib.models.markers.get_contained_extras
```

Collect “extra == ...” operands from a marker.

Returns a list of str. Each str is a speficied extra in this marker.

```
requirementslib.models.markers.get_contained_pyversions
```

Collect all *python_version* operands from a marker.

```
requirementslib.models.markers.get_sorted_version_string(version_set)
```

```
requirementslib.models.markers.get_specset(marker_list)
```

```
requirementslib.models.markers.get_versions
```

```
requirementslib.models.markers.get_without_extra(marker)
```

Build a new marker without the *extra == ...* part.

The implementation relies very deep into packaging’s internals, but I don’t have a better way now (except implementing the whole thing myself).

This could return *None* if the *extra == ...* part is the only one in the input marker.

```
requirementslib.models.markers.get_without_pyversion(marker)
```

Built a new marker without the *python_version* part.

This could return *None* if the *python_version* section is the only section in the marker.

```
requirementslib.models.markers.is_instance(item, cls)
requirementslib.models.markers.marker_from_specifier
requirementslib.models.markers.merge_markers(m1, m2)
requirementslib.models.markers.normalize_marker_str(marker)
requirementslib.models.markers.normalize_specifier_set(specs)
```

Given a specifier set, a string, or an iterable, normalize the specifiers

Note: This function exists largely to deal with *pyzmq* which handles the *requires_python* specifier incorrectly, using *3.7** rather than the correct form of *3.7.**. This workaround can likely go away if we ever introduce enforcement for metadata standards on PyPI.

Parameters **SpecifierSet** **specs** (*Union[str, ...]*) – Supplied specifiers to normalize

Returns A new set of specifiers or specifierset

Return type *Union[Set[Specifier], SpecifierSet]*

```
requirementslib.models.markers.parse_marker_dict(marker_dict)
```

requirementslib.models.metadata module

```
class requirementslib.models.metadata.Dependency(name: str, requirement: packaging.requirements.Requirement,
                                                specifier, extras=NOTHING,
                                                from_extras=None,
                                                python_version="", parent=None,
                                                markers=None, specset_str: str = "",
                                                python_version_str: str = "",
                                                marker_str: str = "")
```

Bases: *object*

add_parent (*parent*)

as_line ()

extras = *None*

Any extras this dependency declares

from_extras = *None*

The name of the extra meta-dependency this one came from (e.g. 'security')

classmethod from_info (*info*)

classmethod from_requirement (*req, parent=None*)

classmethod from_str (*depstr, parent=None*)

markers = *None*

The markers for this dependency

```
name = None
    The name of the dependency

parent = None
    The parent of this dependency (i.e. where it came from)

pin()

python_version = None
    The declared specifier set of allowable python versions for this dependency

requirement = None
    A requirement instance

specifier = None
    The specifier defined in the dependency definition

class requirementslib.models.metadata.Digest(algorithm: str, value: str)
    Bases: object

    algorithm = None
        The algorithm declared for the digest, e.g. 'sha256'

    classmethod collection_from_dict(digest_dict)

    classmethod create(algorithm, value)

    value = None
        The digest value

class requirementslib.models.metadata.ExtrasCollection(name: str, parent: De-
                                                         pendency, dependen-
                                                         cies=NOTHING)

    Bases: object

    add_dependency(dependency)

    dependencies = None
        The members of the collection

    name = None
        The name of the extras collection (e.g. 'security')

    parent = None
        The dependency the collection belongs to

class requirementslib.models.metadata.Package(info, last_serial: int, releases,
                                                         urls=NOTHING)
    Bases: object

    as_dict()

    dependencies

    classmethod from_json(package_json)

    get_dependencies()

    get_latest_lockfile()

    latest_sdist

    latest_wheels

    name

    pin_dependencies(include_extras=None)
```


`requirement``serialize()``version`

```
class requirementslib.models.metadata.PackageEncoder(*,
                                                    skipkeys=False,
                                                    ensure_ascii=True,
                                                    check_circular=True,
                                                    allow_nan=True,
                                                    sort_keys=False, indent=None,
                                                    separators=None,      de-
                                                    fault=None)
```

Bases: `json.encoder.JSONEncoder`**default** (*obj*)

Implement this method in a subclass such that it returns a serializable object for `o`, or calls the base implementation (to raise a `TypeError`).

For example, to support arbitrary iterators, you could implement default like this:

```
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
        return list(iterable)
    # Let the base class default method raise the TypeError
    return JSONEncoder.default(self, o)
```

```
class requirementslib.models.metadata.PackageInfo(name: str, version: str, pack-
age_url: str, summary: str =
None, author: str = None, key-
words=NOTHING, description:
str = "", download_url: str =
"", home_page: str = "", license:
str = "", maintainer: str = "",
maintainer_email: str = "", down-
loads=NOTHING, docs_url=None,
platform: str = "", project_url: str
= "", project_urls=NOTHING,
requires_python=None, re-
quires_dist=NOTHING, re-
lease_url=None, descrip-
tion_content_type: str =
'text/md', bugtrack_url=None,
classifiers=NOTHING, au-
thor_email=None, markers=None,
dependencies=None)
```

Bases: `object`**create_dependencies** (*force=False*)

Create values for `self.dependencies`.

Parameters `force` (*bool*) – Sets `self.dependencies` to an empty tuple if it would be `None`, defaults to `False`.

Returns An updated instance of the current object with `self.dependencies` updated accordingly.

Return type *PackageInfo*

classmethod `from_json(info_json)`

to_dependency()

class `requirementslib.models.metadata.ParsedTag(marker_string=None, python_version=None, platform_system=None, abi=None)`

Bases: `object`

abi = `None`

the ABI represented by the tag

marker_string = `None`

The marker string corresponding to the tag

platform_system = `None`

The platform represented by the tag

python_version = `None`

The python version represented by the tag

class `requirementslib.models.metadata.Release(version: str, urls, name=None)`

Bases: `collections.abc.Sequence`

latest

latest_timestamp

name = `None`

the name of the package

parsed_version

sdists

to_lockfile()

urls = `None`

The URL collection for the release

version = `None`

The version of the release

wheels

yanked

class `requirementslib.models.metadata.ReleaseCollection(releases=NOTHING)`

Bases: `object`

get_latest_lockfile()

latest

classmethod `load(releases, name=None)`

non_yanked_releases

sdists()

sort_releases()

wheels()

```

class requirementslib.models.metadata.ReleaseUrl (md5_digest:          require-
                                                    mentslib.models.metadata.Digest,
                                                    packagetype:    str, upload_time,
                                                    upload_time_iso_8601, size:    int,
                                                    url: str, digests, name: str = None,
                                                    comment_text: str = "", yanked:
                                                    bool = False, downloads: int = -1,
                                                    filename: str = "", has_sig: bool
                                                    = False, python_version: str =
                                                    'source', requires_python: str =
                                                    None, tags=NOTHING)

```

Bases: `object`

comment_text = None

The available comments of the given upload

classmethod create (release_dict, name=None)

digests = None

The digests of the package

downloads = None

The number of downloads (deprecated)

filename = None

The filename of the current upload

get_dependencies ()

get_markers_from_wheel ()

has_sig = None

Whether the upload has a signature

is_sdist

is_wheel

markers

md5_digest = None

The MD5 digest of the given release

name = None

The name of the package

packagetype = None

The package type of the url

pep508_url

python_version = None

The python_version attribute of the upload (e.g. 'source', 'py27', etc)

requires_python = None

The 'requires_python' restriction on the package

sha256

size = None

The size in bytes of the package

tags = None

A list of valid aprsed tags from the upload

upload_time = None

The upload timestamp from the package

upload_time_iso_8601 = None

The ISO8601 formatted upload timestamp of the package

url = None

The URL of the package

yanked = None

Whether the url has been yanked from the server

class requirementslib.models.metadata.**ReleaseUrlCollection** (*urls, name=None*)

Bases: `collections.abc.Sequence`

classmethod **create** (*urls, name=None*)

find_package_type (*type_*)

Given a package type (e.g. sdist, bdist_wheel), find the matching release.

Parameters **type** (*str*) – A package type from `PACKAGE_TYPES`

Returns The package from this collection matching that type, if available

Return type Optional[[ReleaseUrl](#)]

latest

latest_timestamp

name = None

the name of the package

sdists

urls = None

A list of release URLs

wheels

requirementslib.models.metadata.**add_markers_to_dep** (*d, marker_str*)

requirementslib.models.metadata.**convert_package_info** (*info_json*)

requirementslib.models.metadata.**convert_release_urls_to_collection** (*urls=None, name=None*)

requirementslib.models.metadata.**convert_releases_to_collection** (*releases, name=None*)

requirementslib.models.metadata.**create_dependencies** (*requires_dist, parent=None*)

requirementslib.models.metadata.**create_digest_collection** (*digest_dict*)

requirementslib.models.metadata.**create_release_urls_from_list** (*urls, name=None*)

requirementslib.models.metadata.**create_specifierset** (*spec=None*)

requirementslib.models.metadata.**get_local_wheel_metadata** (*wheel_file*)

requirementslib.models.metadata.**get_package** (*name*)

requirementslib.models.metadata.**get_package_from_requirement** (*req*)

requirementslib.models.metadata.**get_package_version** (*name, version*)

requirementslib.models.metadata.**get_release** (*version, urls, name=None*)

`requirementslib.models.metadata.get_releases_from_package` (*releases, name=None*)

`requirementslib.models.metadata.get_remote_sdist_metadata` (*line*)

`requirementslib.models.metadata.get_remote_wheel_metadata` (*whl_file*)

`requirementslib.models.metadata.instance_check_converter` (*expected_type=None, converter=None*)

`requirementslib.models.metadata.parse_tag` (*tag*)

Parse a Tag instance.

:param Tag tag: A tag to parse :return: A parsed tag with combined markers, supported platform and python version :rtype: *ParsedTag*

`requirementslib.models.metadata.split_keywords` (*value*)

`requirementslib.models.metadata.validate_digest` (*inst, attrib, value*)

`requirementslib.models.metadata.validate_extras` (*inst, attrib, value*)

requirementslib.models.pipfile module

class `requirementslib.models.pipfile.Pipfile` (*path: pathlib.Path = NOTHING, projectfile: requirementslib.models.project.ProjectFile = NOTHING, pipfile: requirementslib.models.pipfile.PipfileLoader = NOTHING, pyproject: tomlkit.toml_document.TOMLDocument = NOTHING, build_system: dict = NOTHING, requirements: list = NOTHING, dev_requirements: list = NOTHING*)

Bases: `object`

allow_prereleases

build_backend

build_requires

build_system

dev_packages

dev_requirements

extended_keys

get (*k*)

get_deps (*dev=False, only=True*)

classmethod load (*path, create=False*)

Given a path, load or create the necessary pipfile.

Parameters

- **path** (*Text*) – Path to the project root or pipfile
- **create** (*bool*) – Whether to create the pipfile if not found, defaults to True

Raises

- **OSError** – Thrown if the project root directory doesn't exist

- **FileNotFoundError** – Thrown if the pipfile doesn't exist and `create=False`

Returns A pipfile instance pointing at the supplied project

rtype:: `class:~requirementslib.models.pipfile.Pipfile`

classmethod `load_projectfile` (*path*, *create=False*)

Given a path, load or create the necessary pipfile.

Parameters

- **path** (*Text*) – Path to the project root or pipfile
- **create** (*bool*) – Whether to create the pipfile if not found, defaults to True

Raises

- **OSError** – Thrown if the project root directory doesn't exist
- **FileNotFoundError** – Thrown if the pipfile doesn't exist and `create=False`

Returns A project file instance for the supplied project

Return type `ProjectFile`

packages

path

pipfile

projectfile

classmethod `read_projectfile` (*path*)

Read the specified project file and provide an interface for writing/updating.

Parameters **path** (*Text*) – Path to the target file.

Returns A project file with the model and location for interaction

Return type `ProjectFile`

requirements

requires_python

root

write()

class `requirementslib.models.pipfile.PipfileLoader` (*data*)

Bases: `plette.pipfiles.Pipfile`

classmethod `ensure_package_sections` (*data*)

Ensure that all pipfile package sections are present in the given toml document

:param TOMLDocument data: The toml document to ensure package sections are present on

Returns The updated toml document, ensuring `packages` and `dev-packages` sections are present

Return type `TOMLDocument`

classmethod `load` (*f*, *encoding=None*)

classmethod `populate_source` (*source*)

Derive missing values of source from the existing fields.

```
classmethod validate (data)
```

```
requirementslib.models.pipfile.reorder_source_keys (data)
```

requirementslib.models.project module

```
class requirementslib.models.project.FileDifference (default, develop)
```

```
    Bases: tuple
```

```
    default
```

```
        Alias for field number 0
```

```
    develop
```

```
        Alias for field number 1
```

```
class requirementslib.models.project.Project (root)
```

```
    Bases: object
```

```
    add_line_to_pipfile (line, develop)
```

```
    contains_key_in_pipfile (key)
```

```
    difference_lockfile (lockfile)
```

```
        Generate a difference between the current and given lockfiles.
```

```
        Returns a 2-tuple containing differences in default in develop sections.
```

```
        Each element is a 2-tuple of dicts. The first, inthis, contains entries only present in the current lockfile; the second, inthat, contains entries only present in the given one.
```

```
        If a key exists in both this and that, but the values differ, the key is present in both dicts, pointing to values from each file.
```

```
    is_synced ()
```

```
    lockfile
```

```
    lockfile_location
```

```
    pipfile
```

```
    pipfile_location
```

```
    remove_keys_from_lockfile (keys)
```

```
    remove_keys_from_pipfile (keys, default, develop)
```

```
class requirementslib.models.project.ProjectFile (location, line_ending, model)
```

```
    Bases: object
```

```
    A file in the Pipfile project.
```

```
    dumps ()
```

```
    classmethod read (location, model_cls, invalid_ok=False)
```

```
    write ()
```

```
class requirementslib.models.project.SectionDifference (inthis, inthat)
```

```
    Bases: tuple
```

```
    inthat
```

```
        Alias for field number 1
```

inthis

Alias for field number 0

`requirementslib.models.project.preferred_newlines(f)`

requirementslib.models.requirements module

```
class requirementslib.models.requirements.FileRequirement(setup_path=None,  
                                                         path=None,           ed-  
                                                         itable=False,       ex-  
                                                         tras=NOTHING,  
                                                         uri_scheme=None,  
                                                         uri=NOTHING,  
                                                         link=NOTHING, pypro-  
                                                         ject_requires=NOTHING,  
                                                         pypro-  
                                                         ject_backend=None,  
                                                         pyproject_path=None,  
                                                         subdirectory=None,  
                                                         setup_info=None,  
                                                         has_hashed_name=False,  
                                                         parsed_line=None,  
                                                         name=NOTHING,  
                                                         req=NOTHING)
```

Bases: `object`

File requirements for tar.gz installable files or wheels or setup.py containing directories.

dependencies

editable

Whether the package is editable

extras

Extras if applicable

formatted_path

classmethod from_line (*line*, *editable=None*, *extras=None*, *parsed_line=None*)

classmethod from_pipfile (*name*, *pipfile*)

get_link()

classmethod get_link_from_line (*line*)

Parse link information from given requirement line.

Return a 6-tuple:

- *vcs_type* indicates the VCS to use (e.g. “git”), or None.
- ***prefer*** is either “file”, “path” or “uri”, indicating how the information should be used in later stages.
- ***relpath*** is the relative path to use when recording the dependency, instead of the absolute path/URI used to perform installation. This can be None (to prefer the absolute path or URI).
- ***path*** is the absolute file path to the package. This will always use forward slashes. Can be None if the line is a remote URI.
- ***uri*** is the absolute URI to the package. Can be None if the line is not a URI.

- **link** is an instance of `pip._internal.index.Link`, representing a URI parse result based on the value of `uri`.

This function is provided to deal with edge cases concerning URIs without a valid netloc. Those URIs are problematic to a straight `urlsplit` call because they cannot be reliably reconstructed with `urlunsplit` due to a bug in the standard library:

```
>>> from urllib.parse import urlsplit, urlunsplit
>>> urlunsplit(urlsplit('git+file:///this/breaks'))
'git+file:/this/breaks'
>>> urlunsplit(urlsplit('file:///this/works'))
'file:///this/works'
```

See <https://bugs.python.org/issue23505#msg277350>.

get_name()

get_requirement()

get_uri()

is_direct_url

is_local

is_remote_artifact

line_part

link

Link object representing the package to clone

name

Package name

parsed_line

path

path to hit - without any of the VCS prefixes (like git+ / http+ / etc)

pipfile_part

pyproject_backend

PyProject Build System

pyproject_path

PyProject Path

pyproject_requires

PyProject Requirements

req

A Requirement instance

setup_info

setup_path

Path to the relevant `setup.py` location

setup_py_dir

subdirectory

uri

URI of the package

```
class requirementslib.models.requirements.Line(line, extras=None)
    Bases: object

    base_path
    get_ireq()
    get_line(with_prefix=False, with_markers=False, with_hashes=True, as_list=False)
    classmethod get_requirement_specs(specifierset)
    get_setup_info()
    get_url()
        Sets self.name if given a PEP-508 style URL.
    ireq
    is_artifact
    is_direct_url
    is_file
    is_file_url
    is_installable
    is_named
    is_path
    is_remote_url
    is_url
    is_vcs
    is_wheel
    line_for_ireq
    line_is_installable
        This is a safeguard against decoy requirements when a user installs a package whose name coincides with
        the name of a folder in the cwd, e.g. install alembic when there is a folder called alembic in the working
        directory.

        In this case we first need to check that the given requirement is a valid URL, VCS requirement, or instal-
        lable filesystem path before deciding to treat it as a file requirement over a named requirement.
    line_with_prefix
    link
    metadata
    name
    name_and_specifier
    parse()
    parse_extras()
        Parse extras from self.line and set them on the current object :returns: self :rtype: Line
    parse_hashes()
        Parse hashes from self.line and set them on the current object.

        Returns Self
```

Return type :class:`~Line`

parse_ireq()
parse_link()
parse_markers()
parse_name()
parse_requirement()
parsed_setup_cfg
parsed_setup_py
parsed_url
populate_setup_paths()
pyproject_backend
pyproject_requires
pyproject_toml
ref
requirement
requirement_info

Generates a 3-tuple of the requisite *name*, *extras* and *url* to generate a `Requirement` out of.

Returns A Tuple of an optional name, a Tuple of extras, and an optional URL.

Return type `Tuple[Optional[S], Tuple[Optional[S], ..], Optional[S]]`

setup_cfg
setup_info
setup_py
specifier
specifiers
classmethod split_hashes(line)
subdirectory
url
vcsrepo
wheel_kwargs

class `requirementslib.models.requirements.LinkInfo` (*vcs_type, prefer, relpath, path, uri, link*)

Bases: `tuple`

link
 Alias for field number 5
path
 Alias for field number 3
prefer
 Alias for field number 1

relpath
Alias for field number 2

uri
Alias for field number 4

vcs_type
Alias for field number 0

```
class requirementslib.models.requirements.NamedRequirement (name, version,  
                                                         req=NOTHING,  
                                                         extras=NOTHING,  
                                                         editable=False,  
                                                         parsed_line=None)
```

Bases: `object`

editable

extras

classmethod **from_line** (*line, parsed_line=None*)

classmethod **from_pipfile** (*name, pipfile*)

get_requirement ()

line_part

name

parsed_line

pipfile_part

req

version

```
class requirementslib.models.requirements.Requirement (name=NOTHING, vcs=None,  
                                                         req=None, markers=None,  
                                                         specifiers=NOTHING, in-  
                                                         dex=None, editable=None,  
                                                         hashes=NOTHING,  
                                                         extras=NOTHING,  
                                                         abstract_dep=None,  
                                                         line_instance=None,  
                                                         ireq=None)
```

Bases: `object`

add_hashes (*hashes*)

as_ireq ()

as_line (*sources=None, include_hashes=True, include_extras=True, include_markers=True,*
 as_list=False)

Format this requirement as a line in requirements.txt.

If *sources* provided, it should be an sequence of mappings, containing all possible sources to be used for this requirement.

If *sources* is omitted or falsy, no index information will be included in the requirement line.

as_pipfile ()

build_backend

`commit_hash`

`constraint_line`

`copy()`

`extras_as_pip`

`find_all_matches` (*sources=None, finder=None*)

Find all matching candidates for the current requirement.

Consults a finder to find all matching candidates.

Parameters

- **sources** – Pipfile-formatted sources, defaults to None
- **sources** – list[dict], optional
- **finder** (*PackageFinder*) – A **PackageFinder** instance from pip’s repository implementation

Returns A list of Installation Candidates

Return type list[*InstallationCandidate*]

`classmethod from_ireq` (*ireq*)

`classmethod from_line` (*line*)

`classmethod from_metadata` (*name, version, extras, markers*)

`classmethod from_pipfile` (*name, pipfile*)

`get_abstract_dependencies` (*sources=None*)

Retrieve the abstract dependencies of this requirement.

Returns the abstract dependencies of the current requirement in order to resolve.

Parameters

- **sources** – A list of sources (pipfile format), defaults to None
- **sources** – list, optional

Returns A list of abstract (unpinned) dependencies

Return type list[*AbstractDependency*]

`get_dependencies` (*sources=None*)

Retrieve the dependencies of the current requirement.

Retrieves dependencies of the current requirement. This only works on pinned requirements.

Parameters

- **sources** – Pipfile-formatted sources, defaults to None
- **sources** – list[dict], optional

Returns A set of requirement strings of the dependencies of this requirement.

Return type set(str)

`get_hashes_as_pip` (*as_list=False*)

`get_line_instance` ()

`get_markers` ()

```
get_name()
get_requirement()
get_specifier()
get_specifiers()
get_version()
hashes_as_pip
ireq
is_direct_url
is_file_or_url
is_named
is_vcs
is_wheel
line_instance
markers_as_pip
merge_markers(markers)
name
normalized_name
pipfile_entry
requirement
run_requires(sources=None, finder=None)
specifiers
update_name_from_path(path)
uses_pep517

class requirementslib.models.requirements.VCSRequirement(setup_path=None,
                                                         extras=NOTHING,
                                                         uri_scheme=None,
                                                         pypro-
                                                         ject_requires=NOTHING,
                                                         pypro-
                                                         ject_backend=None,
                                                         pyproject_path=None,
                                                         subdirectory=None,
                                                         setup_info=None,
                                                         has_hashed_name=False,
                                                         parsed_line=None, ed-
                                                         itable=None, uri=None,
                                                         path=None, vcs=None,
                                                         ref=None, repo=None,
                                                         base_line=None,
                                                         name=NOTHING,
                                                         link=NOTHING,
                                                         req=NOTHING)

Bases: requirementslib.models.requirements.FileRequirement
```

editable

Whether the repository is editable

classmethod **from_line** (*line*, *editable=None*, *extras=None*, *parsed_line=None*)

classmethod **from_pipfile** (*name*, *pipfile*)

get_checkout_dir (*src_dir=None*)

get_commit_hash ()

get_link ()

get_name ()

get_requirement ()

get_vcs_repo (*src_dir=None*, *checkout_dir=None*)

line_part

requirements.txt compatible line part sans-extras.

link

locked_vcs_repo (*src_dir=None*)

name**path**

path to the repository, if it's local

pipfile_part**ref**

vcs reference name (branch / commit / tag)

repo**req****setup_info**

update_repo (*src_dir=None*, *ref=None*)

uri

URI for the repository

url**vcs**

vcs type, i.e. git/hg/svn

vcs_uri

`requirementslib.models.requirements.file_req_from_parsed_line(parsed_line)`

`requirementslib.models.requirements.named_req_from_parsed_line(parsed_line)`

`requirementslib.models.requirements.vcs_req_from_parsed_line(parsed_line)`

requirementslib.models.resolvers module

```
class requirementslib.models.resolvers.DependencyResolver (pinned_deps=NOTHING,
                                                           dep_dict=NOTHING,
                                                           candi-
                                                           date_dict=NOTHING,
                                                           pin_history=NOTHING,
                                                           al-
                                                           low_prereleases=False,
                                                           hashes=NOTHING,
                                                           hash_cache=NOTHING,
                                                           finder=None,         in-
                                                           clude_incompatible_hashes=True,
                                                           avail-
                                                           able_candidates_cache=NOTHING)
```

Bases: `object`

add_abstract_dep (*dep*)

Add an abstract dependency by either creating a new entry or merging with an old one.

Parameters *dep* (`AbstractDependency`) – An abstract dependency to add

Raises `ResolutionError` – Raised when the given dependency is not compatible with an existing abstract dependency.

allow_all_wheels ()

Monkey patches pip.Wheel to allow wheels from all platforms and Python versions.

This also saves the candidate cache and set a new one, or else the results from the previous non-patched calls will interfere.

allow_prereleases = `None`

Whether to allow prerelease dependencies

candidate_dict = `None`

A dictionary of sets of version numbers that are valid for a candidate currently

classmethod create (*finder=None, allow_prereleases=False, get_all_hashes=True*)

dep_dict = `None`

A dictionary of abstract dependencies by name

dependencies

finder = `None`

A finder for searching the index

get_hashes ()

get_hashes_for_one (*ireq*)

hash_cache = `None`

A hash cache

hashes = `None`

Stores hashes for each dependency

include_incompatible_hashes = `None`

Whether to include hashes even from incompatible wheels

pin_deps ()

Pins the current abstract dependencies and adds them to the history dict.

Adds any new dependencies to the abstract dependencies already present by merging them together to form new, compatible abstract dependencies.

pin_history = None

A historical record of pins

resolution

resolve (*root_nodes*, *max_rounds*=20)

Resolves dependencies using a backtracking resolver and multiple endpoints.

Note: this resolver caches aggressively. Runs for *max_rounds* or until any two pinning rounds yield the same outcome.

Parameters

- **root_nodes** (list[*Requirement*]) – A list of the root requirements.
- **max_rounds** – The max number of resolution rounds, defaults to 20
- **max_rounds** – int, optional

Raises **RuntimeError** – Raised when max rounds is exceeded without a resolution.

exception requirementslib.models.resolvers.**ResolutionError**

Bases: *Exception*

requirementslib.models.setup_info module

class requirementslib.models.setup_info.**Analyzer**

Bases: *ast.NodeVisitor*

generic_unparse (*item*)

generic_visit (*node*)

Called if no explicit visitor function exists for a node.

match_assignment_name (*match*)

match_assignment_str (*match*)

no_recurse ()

parse_function_names (*should_retry*=True, *function_map*=None)

parse_functions ()

parse_setup_function ()

unmap_binops ()

unparse (*item*)

unparse_Assign (*item*)

unparse_Attribute (*item*)

unparse_BinOp (*item*)

unparse_Call (*item*)

unparse_Compare (*item*)

unparse_Constant (*item*)

unparse_Dict (*item*)

```
unparse_Ellipsis (item)
unparse_IfExp (item)
unparse_List (item)
unparse_Mapping (item)
unparse_Name (item)
unparse_NameConstant (item)
unparse_Num (item)
unparse_Str (item)
unparse_Subscript (item)
unparse_Tuple (item)
unparse_keyword (item)
unparse_list (item)
unparse_str (item)
unparse_tuple (item)
visit_BinOp (node)

class requirementslib.models.setup_info.BaseRequirement (name="",          require-
                                                         ment=None)
    Bases: object
    as_dict ()
    as_tuple ()
    classmethod from_req (req)
    classmethod from_string (line)
    name
    requirement

class requirementslib.models.setup_info.BuildEnv (cleanup=True)
    Bases: pep517.envbuild.BuildEnvironment
    pip_install (reqs)
        Install dependencies into this env by calling pip in a subprocess

class requirementslib.models.setup_info.Extra (name=None, requirements: frozenset =
                                                         NOTHING)
    Bases: object
    add (req)
    as_dict ()
    name
    requirements

class requirementslib.models.setup_info.HookCaller (source_dir, build_backend, back-
                                                         end_path=None)
    Bases: pep517.wrappers.Pep517HookCaller

class requirementslib.models.setup_info.ScandirCloser (path)
    Bases: object
```

```

    close()
    next()
class requirementslib.models.setup_info.SetupInfo(name=None,      base_dir=None,
                                                    version=None,      require-
                                                    ments: frozenset = NOTH-
                                                    ING,      build_requires=None,
                                                    build_backend=NOTHING,
                                                    setup_requires=None,
                                                    python_requires=None,      ex-
                                                    tras_requirements=None,
                                                    setup_cfg: pathlib.Path = None,
                                                    setup_py:  pathlib.Path = None,
                                                    pyproject: pathlib.Path = None,
                                                    ireq=None, extra_kwargs: dict
                                                    = NOTHING, metadata=None,
                                                    stack=None, finalizer=None)

Bases: object
as_dict()
base_dir
build()
build_backend
build_requires
build_sdist()
build_wheel()
classmethod create(base_dir, subdirectory=None, ireq=None, kwargs=None, stack=None)
egg_base
extra_kwargs
extras
classmethod from_ireq(ireq, subdir=None, finder=None, session=None)
classmethod from_requirement(requirement, finder=None)
get_build_backend()
get_egg_metadata(metadata_dir=None, metadata_type=None)
    Given a metadata directory, return the corresponding metadata dictionary.

    Parameters
        • metadata_dir (Optional[str]) – Root metadata path, default: os.getcwd()
        • metadata_type (Optional[str]) – Type of metadata to search for, default None

    Returns A metadata dictionary built from the metadata in the given location

    Return type Dict[Any, Any]
get_extras_from_ireq()
get_info()
get_initial_info()

```

get_metadata_from_wheel (*wheel_path*)

Given a path to a wheel, return the metadata from that wheel.

Returns A dictionary of metadata from the provided wheel

Return type Dict[Any, Any]

ireq

metadata

name

parse_setup_cfg ()

parse_setup_py ()

pep517_config

populate_metadata (*metadata*)

Populates the metadata dictionary from the supplied metadata.

Returns The current instance.

Return type *SetupInfo*

pyproject

python_requires

reload ()

Wipe existing distribution info metadata for rebuilding.

Erases metadata from **self.egg_base** and unsets **self.requirements** and **self.extras**.

requires

run_pyproject ()

Populates the **pyproject.toml** metadata if available.

Returns The current instance

Return type *SetupInfo*

run_setup ()

setup_cfg

setup_py

setup_requires

stack

update_from_dict (*metadata*)

version

`requirementslib.models.setup_info.ast_parse_attribute_from_file` (*path*, *attribute*)

`requirementslib.models.setup_info.ast_parse_file` (*path*)

`requirementslib.models.setup_info.ast_parse_setup_py` (*path*)

`requirementslib.models.setup_info.ast_unparse` (*item*, *initial_mapping=False*, *analyzer=None*, *recurse=True*)

`requirementslib.models.setup_info.build_pep517` (*source_dir*, *build_dir*, *config_settings=None*, *dist_type='wheel'*)

```

requirementslib.models.setup_info.ensure_reqs
requirementslib.models.setup_info.find_distinfo(target, pkg_name=None)
requirementslib.models.setup_info.find_egginfo(target, pkg_name=None)
requirementslib.models.setup_info.get_distinfo_dist(path, pkg_name=None)
requirementslib.models.setup_info.get_egginfo_dist(path, pkg_name=None)
requirementslib.models.setup_info.get_extra_name_from_marker
requirementslib.models.setup_info.get_extras_from_setupcfg(parser)
requirementslib.models.setup_info.get_metadata(path, pkg_name=None, meta-
                                         data_type=None)
requirementslib.models.setup_info.get_metadata_from_dist(dist)
requirementslib.models.setup_info.get_metadata_from_wheel(wheel_path)
requirementslib.models.setup_info.get_name_and_version_from_setupcfg(parser,
                                         pack-
                                         age_dir)
requirementslib.models.setup_info.get_package_dir_from_setupcfg(parser,
                                         base_dir=None)
requirementslib.models.setup_info.iter_metadata(path, pkg_name=None,
                                         metadata_type='egg-info')
requirementslib.models.setup_info.make_base_requirements(reqs)
requirementslib.models.setup_info.parse_setup_cfg(setup_cfg_contents, base_dir)
requirementslib.models.setup_info.parse_special_directives(setup_entry, pack-
                                         age_dir=None)
requirementslib.models.setup_info.pep517_subprocess_runner(cmd, cwd=None, ex-
                                         tra_environs=None)

```

The default method of calling the wrapper subprocess.

```
requirementslib.models.setup_info.run_setup(script_path, egg_base=None)
```

Run a *setup.py* script with a target **egg_base** if provided.

Parameters

- **script_path** (*S*) – The path to the *setup.py* script to run
- **egg_base** (*Optional[S]*) – The metadata directory to build in

Raises `FileNotFoundError` – If the provided *script_path* does not exist

Returns The metadata dictionary

Return type Dict[Any, Any]

```
requirementslib.models.setup_info.setuptools_parse_setup_cfg(path)
```

requirementslib.models.utils module

```
requirementslib.models.utils.as_tuple(ireq)
```

Pulls out the (name: str, version:str, extras:(str)) tuple from the pinned InstallRequirement.

```
requirementslib.models.utils.build_vcs_uri(vcs, uri, name=None, ref=None, subdirec-
                                         tory=None, extras=None)
```

`requirementslib.models.utils.clean_requires_python(candidates)`

Get a cleaned list of all the candidates with valid specifiers in the *requires_python* attributes.

`requirementslib.models.utils.convert_direct_url_to_url(direct_url)`

Converts direct URLs to standard, link-style URLs

Given a direct url as defined by *PEP 508*, convert to a `Link` compatible URL by moving the name and extras into an **egg_fragment**.

Parameters `direct_url` (*str*) – A pep-508 compliant direct url.

Returns A reformatted URL for use with `Link` objects and `InstallRequirement` objects.

Return type `AnyStr`

`requirementslib.models.utils.convert_url_to_direct_url(url, name=None)`

Converts normal link-style URLs to direct urls.

Given a `Link` compatible URL, convert to a direct url as defined by *PEP 508* by extracting the name and extras from the **egg_fragment**.

Parameters

- **url** (*AnyStr*) – A `InstallRequirement` compliant URL.
- **name** (*Optional[AnyStr]*) – A name to use in case the supplied URL doesn't provide one.

Returns A pep-508 compliant direct url.

Return type `AnyStr`

Raises

- **ValueError** – Raised when the URL can't be parsed or a name can't be found.
- **TypeError** – When a non-string input is provided.

`requirementslib.models.utils.create_link(link)`

`requirementslib.models.utils.expand_env_variables(line)`

Expand the env vars in a line following pip's standard. https://pip.pypa.io/en/stable/reference/pip_install/#id10

Matches environment variable-style values in `'${MY_VARIABLE_1}'` with the variable name consisting of only uppercase letters, digits or the `'_'`

`requirementslib.models.utils.extras_to_string(extras)`

Turn a list of extras into a string

Parameters `extras` (*List[str]*) – a list of extras to format

Returns A string of extras

Return type `str`

`requirementslib.models.utils.filter_dict(dict_)`

`requirementslib.models.utils.filter_none(k, v)`

`requirementslib.models.utils.fix_requires_python_marker(requires_python)`

`requirementslib.models.utils.flat_map(fn, collection)`

Map a function over a collection and flatten the result by one-level

`requirementslib.models.utils.format_requirement(ireq)`

Formats an `InstallRequirement` instance as a string.

Generic formatter for pretty printing `InstallRequirements` to the terminal in a less verbose way than using its `__str__` method.

:param `InstallRequirement ireq`: A pip **InstallRequirement** instance. :return: A formatted string for prettyprinting :rtype: str

`requirementslib.models.utils.format_specifier(ireq)`

Generic formatter for pretty printing specifiers.

Pretty-prints specifiers from `InstallRequirements` for output to terminal.

:param `InstallRequirement ireq`: A pip **InstallRequirement** instance. :return: A string of specifiers in the given install requirement or <any> :rtype: str

`requirementslib.models.utils.full_groupby(iterable, key=None)`

Like `groupby()`, but sorts the input on the group key first.

`requirementslib.models.utils.get_default_pyproject_backend()`

`requirementslib.models.utils.get_name_variants(pkg)`

Given a packager name, get the variants of its name for both the canonicalized and “safe” forms.

Parameters `pkg` (*AnyStr*) – The package to lookup

Returns A list of names.

Return type Set

`requirementslib.models.utils.get_pinned_version(ireq)`

Get the pinned version of an `InstallRequirement`.

An `InstallRequirement` is considered pinned if:

- Is not editable
- It has exactly one specifier
- That specifier is “==”
- The version does not contain a wildcard

Examples: `django==1.8` # pinned `django>1.8` # NOT pinned `django~=1.8` # NOT pinned `django==1.*` # NOT pinned

Raises `TypeError` if the input is not a valid `InstallRequirement`, or `ValueError` if the `InstallRequirement` is not pinned.

`requirementslib.models.utils.get_pyproject(path)`

Given a base path, look for the corresponding `pyproject.toml` file and return its `build_requires` and `build_backend`.

Parameters `path` (*AnyStr*) – The root path of the project, should be a directory (will be truncated)

Returns A 2 tuple of build requirements and the build backend

Return type Optional[Tuple[List[*AnyStr*], *AnyStr*]]

`requirementslib.models.utils.get_setuptools_version`

`requirementslib.models.utils.get_url_name(url)`

Given a url, derive an appropriate name to use in a pipfile.

Parameters `url` (*str*) – A url to derive a string from

Returns The name of the corresponding pipfile entry

Return type Text`requirementslib.models.utils.get_version(pipfile_entry)``requirementslib.models.utils.init_requirement(name)``requirementslib.models.utils.is_pinned_requirement(ireq)`

Returns whether an InstallRequirement is a “pinned” requirement.

An InstallRequirement is considered pinned if:

- Is not editable
- It has exactly one specifier
- That specifier is “==”
- The version does not contain a wildcard

Examples: `django==1.8` # pinned `django>1.8` # NOT pinned `django~=1.8` # NOT pinned `django==1.*` # NOT pinned

`requirementslib.models.utils.key_from_ireq(ireq)`

Get a standardized key for an InstallRequirement.

`requirementslib.models.utils.key_from_req(req)`

Get an all-lowercase version of the requirement’s name.

`requirementslib.models.utils.lookup_table(values, key=None, keyval=None, unique=False, use_lists=False)`

Builds a dict-based lookup table (index) elegantly.

Supports building normal and unique lookup tables. For example:

```
>>> assert lookup_table(
...     ['foo', 'bar', 'baz', 'qux', 'quux'], lambda s: s[0]) == {
...     'b': {'bar', 'baz'},
...     'f': {'foo'},
...     'q': {'quux', 'qux'}
... }
```

For key functions that uniquely identify values, set `unique=True`:

```
>>> assert lookup_table(
...     ['foo', 'bar', 'baz', 'qux', 'quux'], lambda s: s[0], unique=True) == {
...     'b': 'baz',
...     'f': 'foo',
...     'q': 'quux'
... }
```

The values of the resulting lookup table will be values, not sets.

For extra power, you can even change the values while building up the LUT. To do so, use the `keyval` function instead of the `key` arg:

```
>>> assert lookup_table(
...     ['foo', 'bar', 'baz', 'qux', 'quux'],
...     keyval=lambda s: (s[0], s[1:])) == {
...     'b': {'ar', 'az'},
...     'f': {'oo'},
...     'q': {'uux', 'ux'}
... }
```


`requirementslib.models.utils.make_install_requirement` (*name*, *version=None*, *extras=None*, *markers=None*, *constraint=False*)

Generates an InstallRequirement.

Create an InstallRequirement from the supplied metadata.

Parameters

- **name** (*str*) – The requirement’s name.
- **version** (*str*.) – The requirement version (must be pinned).
- **extras** (*list[str]*) – The desired extras.
- **markers** (*str*) – The desired markers, without a preceding semicolon.
- **constraint** – Whether to flag the requirement as a constraint, defaults to False.
- **constraint** – bool, optional

Returns A generated InstallRequirement

Return type InstallRequirement

`requirementslib.models.utils.name_from_req` (*req*)

Get the name of the requirement

`requirementslib.models.utils.normalize_name` (*pkg*)

Given a package name, return its normalized, non-canonicalized form.

Parameters *pkg* (*AnyStr*) – The name of a package

Returns A normalized package name

Return type AnyStr

`requirementslib.models.utils.optional_instance_of` (*cls*)

`requirementslib.models.utils.parse_extras` (*extras_str*)

Turn a string of extras into a parsed extras list

Parameters *extras_str* (*str*) – An extras string

Returns A sorted list of extras

Return type List[str]

`requirementslib.models.utils.read_source` (*path*, *encoding='utf-8'*)

Read a source file and get the contents with proper encoding for Python 2/3.

Parameters

- **path** (*AnyStr*) – the file path
- **encoding** (*AnyStr*) – the encoding that defaults to UTF-8

Returns The contents of the source file

Return type AnyStr

`requirementslib.models.utils.specs_to_string` (*specs*)

Turn a list of specifier tuples into a string

Parameters *str*] **specs** (*List[Union[Specifier,]*) – a list of specifiers to format

Returns A string of specifiers

Return type str

`requirementslib.models.utils.split_markers_from_line(line)`

Split markers from a dependency

`requirementslib.models.utils.split_ref_from_uri(uri)`

Given a path or URI, check for a ref and split it from the path if it is present, returning a tuple of the original input and the ref or None.

Parameters `uri` (*AnyStr*) – The path or URI to split

Returns A 2-tuple of the path or URI and the ref

Return type `Tuple[AnyStr, Optional[AnyStr]]`

`requirementslib.models.utils.split_vcs_method_from_uri(uri)`

Split a vcs+uri formatted uri into (vcs, uri)

`requirementslib.models.utils.strip_extras_markers_from_requirement(req)`

Strips extras markers from requirement instances.

Given a `Requirement` instance with markers defining `extra == 'name'`, strip out the extras from the markers and return the cleaned requirement

Parameters `req` (*PackagingRequirement*) – A packaging requirement to clean

Returns A cleaned requirement

Return type `PackagingRequirement`

`requirementslib.models.utils.tomlkit_dict_to_python(toml_dict)`

`requirementslib.models.utils.tomlkit_value_to_python(toml_value)`

`requirementslib.models.utils.validate_markers(instance, attr_, value)`

`requirementslib.models.utils.validate_path(instance, attr_, value)`

`requirementslib.models.utils.validate_specifiers(instance, attr_, value)`

`requirementslib.models.utils.validate_vcs(instance, attr_, value)`

`requirementslib.models.utils.version_from_ireq(ireq)`

version_from_ireq Extract the version from a supplied `InstallRequirement`

Parameters `ireq` (*InstallRequirement*) – An `InstallRequirement`

Returns The version of the `InstallRequirement`.

Return type `str`

requirementslib.models.url module

```
class requirementslib.models.url.URI(host: str, scheme: str = 'https', port: int = None,
    path: str = "", query: str = "", fragment: str = "",
    subdirectory: str = "", ref: str = "", username: str = "",
    password: str = "", query_dict: orderedmulti-
    dict.orderedmultidict.omdict = NOTHING, name: str =
    "", extras: tuple = NOTHING, is_direct_url: bool =
    False, is_implicit_ssh: bool = False, auth: str = None,
    fragment_dict: dict = NOTHING, username_is_quoted:
    bool = False, password_is_quoted: bool = False)
```

Bases: `object`

`as_link`

bare_url

base_url

extras = None
Any extras requested from the requirement

fragment = None
URL Fragments, e.g. *#fragment=value*

full_url

get_host_port_path (*strip_ref=False*)

classmethod get_parsed_url (*url*)

get_password (*unquote=False, include_token=True*)

get_username (*unquote=False*)

hidden_auth

host = None
The target hostname, e.g. *amazon.com*

is_direct_url = None
Whether the url was parsed as a direct pep508-style URL

is_file_url

is_implicit_ssh = None
Whether the url was an implicit *git+ssh* url (passed as *git+git@*)

is_installable

is_vcs

name = None
The name of the specified package in case it is a VCS URI with an egg fragment

name_with_extras

classmethod parse (*url*)

static parse_subdirectory (*url_part*)

password = None
Password parsed from *user:password@hostname*

path = None
The url path, e.g. */path/to/endpoint*

port = None
The numeric port of the url if specified

query = None
Query parameters, e.g. *?variable=value...*

query_dict = None
An *OrderedMultiDict* representing query fragments

ref = None
VCS ref this URI points at, if available

safe_string

scheme = None

The URI Scheme, e.g. *salesforce*

secret

subdirectory = None

Subdirectory fragment, e.g. *&subdirectory=blah...*

to_string (*escape_password=True, unquote=True, direct=None, strip_ssh=False, strip_ref=False, strip_name=False, strip_subdir=False*)

Converts the current URI to a string, unquoting or escaping the password as needed.

Parameters

- **escape_password** – Whether to replace password with ----, default True
- **escape_password** – bool, optional
- **unquote** – Whether to unquote url-escapes in the password, default False
- **unquote** – bool, optional
- **direct** (*bool*) – Whether to format as a direct URL
- **strip_ssh** (*bool*) – Whether to strip the SSH scheme from the url (git only)
- **strip_ref** (*bool*) – Whether to drop the VCS ref (if present)
- **strip_name** (*bool*) – Whether to drop the name and extras (if present)
- **strip_subdir** (*bool*) – Whether to drop the subdirectory (if present)

Returns The reconstructed string representing the URI

Return type `str`

unsafe_string

uri_escape

url_without_fragment

url_without_fragment_or_ref

url_without_ref

username = None

The username if provided, parsed from *user:password@hostname*

`requirementslib.models.url.remove_password_from_url(url)`

Given a url, remove the password and insert 4 dashes.

Parameters **url** (*S*) – The url to replace the authentication in

Returns The new URL without authentication

Return type `S`

`requirementslib.models.url.update_url_name_and_fragment(name_with_extras, ref, parsed_dict)`

requirementslib.models.vcs module

```
class requirementslib.models.vcs.VCSRepository(url, name, checkout_directory, vcs_type,
                                              parsed_url=NOTHING, subdirec-
                                              tory=None, commit_sha=None,
                                              ref=None, repo_backend=NOTHING,
                                              clone_log=None)
```

Bases: `object`

DEFAULT_RUN_ARGS = `None`

checkout_ref(*ref*)

get_commit_hash(*ref=None*)

get_parsed_url()

get_repo_backend()

is_local

classmethod monkeypatch_pip()

obtain()

update(*ref*)

2.1.2 requirementslib.utils module

```
exception requirementslib.utils.PathAccessError(exc, seg, path)
```

Bases: `KeyError, IndexError, TypeError`

An amalgamation of `KeyError`, `IndexError`, and `TypeError`, representing what can occur when looking up a path in a nested object.

```
requirementslib.utils.add_ssh_scheme_to_git_uri(uri)
```

Cleans VCS uris from pip format

```
requirementslib.utils.convert_entry_to_path(path)
```

Convert a pipfile entry to a string

```
requirementslib.utils.default_visit(path, key, value)
```

```
requirementslib.utils.dict_path_enter(path, key, value)
```

```
requirementslib.utils.dict_path_exit(path, key, old_parent, new_parent, new_items)
```

```
requirementslib.utils.get_dist_metadata(dist)
```

```
requirementslib.utils.get_path(root, path, default=<object object>)
```

Retrieve a value from a nested object via a tuple representing the lookup path. `>>> root = {'a': {'b': {'c': [[1], [2], [3]]}}}` `>>> get_path(root, ('a', 'b', 'c', 2, 0))` 3 The path format is intentionally consistent with that of `remap()`. One of `get_path`'s chief aims is improved error messaging. EAFP is great, but the error messages are not. For instance, `root['a']['b']['c'][2][1]` gives back `IndexError: list index out of range` What went out of range where? `get_path` currently raises `PathAccessError: could not access 2 from path ('a', 'b', 'c', 2, 1), got error: IndexError('list index out of range',)`, a subclass of `IndexError` and `KeyError`. You can also pass a default that covers the entire operation, should the lookup fail at any level. Args:

root: The target nesting of dictionaries, lists, or other objects supporting `__getitem__`.

path (tuple): A list of strings and integers to be successively looked up within *root*.

default: The value to be returned should any `PathAccessError` exceptions be raised.

`requirementslib.utils.get_setup_paths(base_path, subdirectory=None)`

`requirementslib.utils.is_editable(pipfile_entry)`

`requirementslib.utils.is_installable_dir(path)`

`requirementslib.utils.is_installable_file(path)`

Determine if a path can potentially be installed

`requirementslib.utils.is_star(val)`

`requirementslib.utils.is_vcs(pipfile_entry)`

Determine if dictionary entry from Pipfile is for a vcs dependency.

`requirementslib.utils.merge_items(target_list, sourced=False)`

`requirementslib.utils.prepare_pip_source_args(sources, pip_args=None)`

`requirementslib.utils.remap(root, visit=<function default_visit>, enter=<function dict_path_enter>, exit=<function dict_path_exit>, **kwargs)`

The `remap` (“recursive map”) function is used to traverse and transform nested structures. Lists, tuples, sets, and dictionaries are just a few of the data structures nested into heterogenous tree-like structures that are so common in programming. Unfortunately, Python’s built-in ways to manipulate collections are almost all flat. List comprehensions may be fast and succinct, but they do not recurse, making it tedious to apply quick changes or complex transforms to real-world data. `remap` goes where list comprehensions cannot. Here’s an example of removing all Nones from some data: `>>> from pprint import pprint >>> reviews = {'Star Trek': {'TNG': 10, 'DS9': 8.5, 'ENT': None}, ... 'Babylon 5': 6, 'Dr. Who': None} >>> pprint(remap(reviews, lambda p, k, v: v is not None)) {'Babylon 5': 6, 'Star Trek': {'DS9': 8.5, 'TNG': 10}}` Notice how both Nones have been removed despite the nesting in the dictionary. Not bad for a one-liner, and that’s just the beginning. See [‘this remap cookbook’](http://sedimental.org/remap.html) for more delicious recipes. .. `_this remap cookbook`: <http://sedimental.org/remap.html> `remap` takes four main arguments: the object to traverse and three optional callables which determine how the remapped object will be created. Args:

root: The target object to traverse. By default, `remap` supports iterables like `list`, `tuple`, `dict`, and `set`, but any object traversable by `enter` will work.

visit (callable): This function is called on every item in `root`. It must accept three positional arguments, `path`, `key`, and `value`. `path` is simply a tuple of parents’ keys. `visit` should return the new key-value pair. It may also return `True` as shorthand to keep the old item unmodified, or `False` to drop the item from the new structure. `visit` is called after `enter`, on the new parent. The `visit` function is called for every item in `root`, including duplicate items. For traversable values, it is called on the new parent object, after all its children have been visited. The default visit behavior simply returns the key-value pair unmodified.

enter (callable): This function controls which items in `root` are traversed. It accepts the same arguments as `visit`: the path, the key, and the value of the current item. It returns a pair of the blank new parent, and an iterator over the items which should be visited. If `False` is returned instead of an iterator, the value will not be traversed. The `enter` function is only called once per unique value. The default `enter` behavior support mappings, sequences, and sets. Strings and all other iterables will not be traversed.

exit (callable): This function determines how to handle items once they have been visited. It gets the same three arguments as the other functions – `path`, `key`, `value` – plus two more: the blank new parent object returned from `enter`, and a list of the new items, as remapped by `visit`. Like `enter`, the `exit` function is only called once per unique value. The default exit behavior is to simply add all new items to the new parent, e.g., using `list.extend()` and `dict.update()` to add to the new parent. Immutable objects, such as a `tuple` or `namedtuple`, must be recreated from scratch, but use the same type as the new parent passed back from the `enter` function.

reraise_visit (bool): A pragmatic convenience for the *visit* callable. When set to `False`, `remap` ignores any errors raised by the *visit* callback. Items causing exceptions are kept. See examples for more details.

`remap` is designed to cover the majority of cases with just the *visit* callable. While passing in multiple callables is very empowering, `remap` is designed so very few cases should require passing more than one function. When passing *enter* and *exit*, it's common and easiest to build on the default behavior. Simply add `from boltons.iterutils import default_enter (or default_exit)`, and have your *enter*/*exit* function call the default behavior before or after your custom logic. See [‘this example’_](#). Duplicate and self-referential objects (aka reference loops) are automatically handled internally, [‘as shown here’_](#). .. [_this example: http://sedimental.org/remap.html#sort_all_lists](#) .. [_as shown here: http://sedimental.org/remap.html#corner_cases](#)

```
requirementslib.utils.setup_logger()
```

```
requirementslib.utils.strip_ssh_from_git_uri(uri)
```

Return `git+ssh://` formatted URI to `git+git@` format

CHAPTER 3

Indices and tables

- `genindex`
- `modindex`
- `search`

r

- `requirementslib`, [7](#)
- `requirementslib.models`, [12](#)
- `requirementslib.models.cache`, [12](#)
- `requirementslib.models.dependencies`, [13](#)
- `requirementslib.models.lockfile`, [16](#)
- `requirementslib.models.markers`, [18](#)
- `requirementslib.models.metadata`, [19](#)
- `requirementslib.models.pipfile`, [25](#)
- `requirementslib.models.project`, [27](#)
- `requirementslib.models.requirements`, [28](#)
- `requirementslib.models.resolvers`, [36](#)
- `requirementslib.models.setup_info`, [37](#)
- `requirementslib.models.url`, [46](#)
- `requirementslib.models.utils`, [41](#)
- `requirementslib.models.vcs`, [49](#)
- `requirementslib.utils`, [49](#)

A

- `abi` (*requirementslib.models.metadata.ParsedTag attribute*), 22
- `AbstractDependency` (class in *requirementslib.models.dependencies*), 13
- `add()` (*requirementslib.models.setup_info.Extra method*), 38
- `add_abstract_dep()` (*requirementslib.models.resolvers.DependencyResolver method*), 36
- `add_dependency()` (*requirementslib.models.metadata.ExtrasCollection method*), 20
- `add_hashes()` (*requirementslib.models.requirements.Requirement method*), 32
- `add_hashes()` (*requirementslib.Requirement method*), 10
- `add_line_to_pipfile()` (*requirementslib.models.project.Project method*), 27
- `add_markers_to_dep()` (in module *requirementslib.models.metadata*), 24
- `add_parent()` (*requirementslib.models.metadata.Dependency method*), 19
- `add_ssh_scheme_to_git_uri()` (in module *requirementslib.utils*), 49
- `algorithm` (*requirementslib.models.metadata.Digest attribute*), 20
- `allow_all_wheels()` (*requirementslib.models.resolvers.DependencyResolver method*), 36
- `allow_prereleases` (*requirementslib.models.pipfile.Pipfile attribute*), 25
- `allow_prereleases` (*requirementslib.models.resolvers.DependencyResolver attribute*), 36
- `allow_prereleases` (*requirementslib.Pipfile attribute*), 9
- `Analyzer` (class in *requirementslib.models.setup_info*), 37
- `as_cache_key()` (*requirementslib.models.cache.DependencyCache method*), 12
- `as_dict()` (*requirementslib.models.metadata.Package method*), 20
- `as_dict()` (*requirementslib.models.setup_info.BaseRequirement method*), 38
- `as_dict()` (*requirementslib.models.setup_info.Extra method*), 38
- `as_dict()` (*requirementslib.models.setup_info.SetupInfo method*), 39
- `as_ireq()` (*requirementslib.models.requirements.Requirement method*), 32
- `as_ireq()` (*requirementslib.Requirement method*), 10
- `as_line()` (*requirementslib.models.metadata.Dependency method*), 19
- `as_line()` (*requirementslib.models.requirements.Requirement method*), 32
- `as_line()` (*requirementslib.Requirement method*), 10
- `as_link` (*requirementslib.models.url.URI attribute*), 46
- `as_pipfile()` (*requirementslib.models.requirements.Requirement method*), 32
- `as_pipfile()` (*requirementslib.Requirement method*), 10
- `as_requirements()` (*requirementslib.Lockfile method*), 7
- `as_requirements()` (*requirementslib.models.lockfile.Lockfile method*), 16
- `as_tuple()` (in module *requirementslib.models.utils*), 41
- `as_tuple()` (*requirementslib.models.setup_info.BaseRequirement method*), 38
- `ast_parse_attribute_from_file()` (in *mod-*

ule requirementslib.models.setup_info), 40
ast_parse_file() (in module *requirementslib.models.setup_info*), 40
ast_parse_setup_py() (in module *requirementslib.models.setup_info*), 40
ast_unparse() (in module *requirementslib.models.setup_info*), 40

B

bare_url (*requirementslib.models.url.URI* attribute), 46
base_dir (*requirementslib.models.setup_info.SetupInfo* attribute), 39
base_path (*requirementslib.models.requirements.Line* attribute), 30
base_url (*requirementslib.models.url.URI* attribute), 47
BaseRequirement (class in *requirementslib.models.setup_info*), 38
build() (*requirementslib.models.setup_info.SetupInfo* method), 39
build_backend (*requirementslib.models.pipfile.Pipfile* attribute), 25
build_backend (*requirementslib.models.requirements.Requirement* attribute), 32
build_backend (*requirementslib.models.setup_info.SetupInfo* attribute), 39
build_backend (*requirementslib.Pipfile* attribute), 9
build_backend (*requirementslib.Requirement* attribute), 10
build_pep517() (in module *requirementslib.models.setup_info*), 40
build_requires (*requirementslib.models.pipfile.Pipfile* attribute), 25
build_requires (*requirementslib.models.setup_info.SetupInfo* attribute), 39
build_requires (*requirementslib.Pipfile* attribute), 9
build_sdist() (*requirementslib.models.setup_info.SetupInfo* method), 39
build_system (*requirementslib.models.pipfile.Pipfile* attribute), 25
build_system (*requirementslib.Pipfile* attribute), 9
build_vcs_uri() (in module *requirementslib.models.utils*), 41
build_wheel() (*requirementslib.models.setup_info.SetupInfo* method), 39

BuildEnv (class in *requirementslib.models.setup_info*), 38

C

cache (*requirementslib.models.cache.DependencyCache* attribute), 12
candidate_dict (*requirementslib.models.resolvers.DependencyResolver* attribute), 36
checkout_ref() (*requirementslib.models.vcs.VCSRepository* method), 49
clean_requires_python() (in module *requirementslib.models.utils*), 41
cleanup_pyspecs (in module *requirementslib.models.markers*), 18
clear() (*requirementslib.models.cache.DependencyCache* method), 12
close() (*requirementslib.models.setup_info.ScandirCloser* method), 38
collection_from_dict() (*requirementslib.models.metadata.Digest* class method), 20
comment_text (*requirementslib.models.metadata.ReleaseUrl* attribute), 23
commit_hash (*requirementslib.models.requirements.Requirement* attribute), 32
commit_hash (*requirementslib.Requirement* attribute), 10
compatible_abstract_dep() (*requirementslib.models.dependencies.AbstractDependency* method), 13
compatible_versions() (*requirementslib.models.dependencies.AbstractDependency* method), 13
constraint_line (*requirementslib.models.requirements.Requirement* attribute), 33
constraint_line (*requirementslib.Requirement* attribute), 10
contains_extra (in module *requirementslib.models.markers*), 18
contains_key_in_pipfile() (*requirementslib.models.project.Project* method), 27
contains_pyversion (in module *requirementslib.models.markers*), 18
convert_direct_url_to_url() (in module *requirementslib.models.utils*), 42
convert_entry_to_path() (in module *requirementslib.utils*), 49

`convert_package_info()` (in module `requirementslib.models.metadata`), 24
`convert_release_urls_to_collection()` (in module `requirementslib.models.metadata`), 24
`convert_releases_to_collection()` (in module `requirementslib.models.metadata`), 24
`convert_url_to_direct_url()` (in module `requirementslib.models.utils`), 42
`copy()` (`requirementslib.models.requirements.Requirement` method), 33
`copy()` (`requirementslib.Requirement` method), 10
`CorruptCacheError`, 12
`create()` (`requirementslib.Lockfile` class method), 7
`create()` (`requirementslib.models.lockfile.Lockfile` class method), 16
`create()` (`requirementslib.models.metadata.Digest` class method), 20
`create()` (`requirementslib.models.metadata.ReleaseUrl` class method), 23
`create()` (`requirementslib.models.metadata.ReleaseUrlCollection` class method), 24
`create()` (`requirementslib.models.resolvers.DependencyResolver` class method), 36
`create()` (`requirementslib.models.setup_info.SetupInfo` class method), 39
`create_dependencies()` (in module `requirementslib.models.metadata`), 24
`create_dependencies()` (`requirementslib.models.metadata.PackageInfo` method), 21
`create_digest_collection()` (in module `requirementslib.models.metadata`), 24
`create_link()` (in module `requirementslib.models.utils`), 42
`create_release_urls_from_list()` (in module `requirementslib.models.metadata`), 24
`create_specifierset()` (in module `requirementslib.models.metadata`), 24

D

`default` (`requirementslib.Lockfile` attribute), 7
`default` (`requirementslib.models.lockfile.Lockfile` attribute), 16
`default` (`requirementslib.models.project.FileDifference` attribute), 27
`default()` (`requirementslib.models.metadata.PackageEncoder` method), 21
`DEFAULT_RUN_ARGS` (`requirementslib.models.vcs.VCSRepository` attribute), 49
`default_visit()` (in module `requirementslib.utils`), 49
`dep_dict` (`requirementslib.models.resolvers.DependencyResolver` attribute), 36
`dependencies` (`requirementslib.models.metadata.ExtrasCollection` attribute), 20
`dependencies` (`requirementslib.models.metadata.Package` attribute), 20
`dependencies` (`requirementslib.models.requirements.FileRequirement` attribute), 28
`dependencies` (`requirementslib.models.resolvers.DependencyResolver` attribute), 36
`Dependency` (class in `requirementslib.models.metadata`), 19
`DependencyCache` (class in `requirementslib.models.cache`), 12
`DependencyResolver` (class in `requirementslib.models.resolvers`), 36
`dev_packages` (`requirementslib.models.pipfile.Pipfile` attribute), 25
`dev_packages` (`requirementslib.Pipfile` attribute), 9
`dev_requirements` (`requirementslib.Lockfile` attribute), 7
`dev_requirements` (`requirementslib.models.lockfile.Lockfile` attribute), 16
`dev_requirements` (`requirementslib.models.pipfile.Pipfile` attribute), 25
`dev_requirements` (`requirementslib.Pipfile` attribute), 9
`dev_requirements_list` (`requirementslib.Lockfile` attribute), 7
`dev_requirements_list` (`requirementslib.models.lockfile.Lockfile` attribute), 16
`develop` (`requirementslib.Lockfile` attribute), 7
`develop` (`requirementslib.models.lockfile.Lockfile` attribute), 16
`develop` (`requirementslib.models.project.FileDifference` attribute), 27
`dict_path_enter()` (in module `requirementslib.utils`), 49
`dict_path_exit()` (in module `requirementslib.utils`), 49
`difference_lockfile()` (`requirementslib.models.project.Project` method), 27
`Digest` (class in `requirementslib.models.metadata`), 20
`digests` (`requirementslib.models.metadata.ReleaseUrl` attribute), 23
`downloads` (`requirementslib.models.metadata.ReleaseUrl` attribute), 23
`dumps()` (`requirementslib.models.project.ProjectFile` attribute), 23

- method), 27
- ## E
- editable (requirementslib.models.requirements.FileRequirement attribute), 28
- editable (requirementslib.models.requirements.NamedRequirement attribute), 32
- editable (requirementslib.models.requirements.VCSRequirement attribute), 34
- egg_base (requirementslib.models.setup_info.SetupInfo attribute), 39
- ensure_package_sections() (requirementslib.models.pipfile.PipfileLoader class method), 26
- ensure_reqs (in module requirementslib.models.setup_info), 40
- expand_env_variables() (in module requirementslib.models.utils), 42
- extended_keys (requirementslib.Lockfile attribute), 7
- extended_keys (requirementslib.models.lockfile.Lockfile attribute), 16
- extended_keys (requirementslib.models.pipfile.Pipfile attribute), 25
- extended_keys (requirementslib.Pipfile attribute), 9
- Extra (class in requirementslib.models.setup_info), 38
- extra_kwargs (requirementslib.models.setup_info.SetupInfo attribute), 39
- extras (requirementslib.models.metadata.Dependency attribute), 19
- extras (requirementslib.models.requirements.FileRequirement attribute), 28
- extras (requirementslib.models.requirements.NamedRequirement attribute), 32
- extras (requirementslib.models.setup_info.SetupInfo attribute), 39
- extras (requirementslib.models.url.URI attribute), 47
- extras_as_pip (requirementslib.models.requirements.Requirement attribute), 33
- extras_as_pip (requirementslib.Requirement attribute), 10
- extras_to_string() (in module requirementslib.models.utils), 42
- ExtrasCollection (class in requirementslib.models.metadata), 20
- filename (requirementslib.models.metadata.ReleaseUrl attribute), 23
- filename_format (requirementslib.models.cache.RequiresPythonCache attribute), 13
- FileRequirement (class in requirementslib.models.requirements), 28
- filter_dict() (in module requirementslib.models.utils), 42
- filter_none() (in module requirementslib.models.utils), 42
- find_all_matches() (in module requirementslib.models.dependencies), 14
- find_all_matches() (requirementslib.models.requirements.Requirement method), 33
- find_all_matches() (requirementslib.Requirement method), 10
- find_distinfo() (in module requirementslib.models.setup_info), 41
- find_egginfo() (in module requirementslib.models.setup_info), 41
- find_package_type() (requirementslib.models.metadata.ReleaseUrlCollection method), 24
- finder (requirementslib.models.resolvers.DependencyResolver attribute), 36
- fix_requires_python_marker() (in module requirementslib.models.utils), 42
- fix_version_tuple (in module requirementslib.models.markers), 18
- flat_map() (in module requirementslib.models.utils), 42
- format_pyversion() (in module requirementslib.models.markers), 18
- format_requirement() (in module requirementslib.models.utils), 42
- format_specifier() (in module requirementslib.models.utils), 43
- formatted_path (requirementslib.models.requirements.FileRequirement attribute), 28
- fragment (requirementslib.models.url.URI attribute), 47
- from_data() (requirementslib.Lockfile class method), 7
- from_data() (requirementslib.models.lockfile.Lockfile class method), 16
- from_extras (requirementslib.models.metadata.Dependency attribute), 19
- from_info() (requirementslib.models.metadata.Dependency class method), 19
- ## F
- file_req_from_parsed_line() (in module requirementslib.models.requirements), 35
- FileDifference (class in requirementslib.models.project), 27

[from_ireq\(\)](#) (*requirementslib.models.requirements.Requirement class method*), 33
[from_ireq\(\)](#) (*requirementslib.models.setup_info.SetupInfo class method*), 39
[from_ireq\(\)](#) (*requirementslib.Requirement class method*), 10
[from_json\(\)](#) (*requirementslib.models.metadata.Package class method*), 20
[from_json\(\)](#) (*requirementslib.models.metadata.PackageInfo class method*), 22
[from_line\(\)](#) (*requirementslib.models.markers.PipenvMarkers class method*), 18
[from_line\(\)](#) (*requirementslib.models.requirements.FileRequirement class method*), 28
[from_line\(\)](#) (*requirementslib.models.requirements.NamedRequirement class method*), 32
[from_line\(\)](#) (*requirementslib.models.requirements.Requirement class method*), 33
[from_line\(\)](#) (*requirementslib.models.requirements.VCSRequirement class method*), 35
[from_line\(\)](#) (*requirementslib.Requirement class method*), 10
[from_metadata\(\)](#) (*requirementslib.models.requirements.Requirement class method*), 33
[from_metadata\(\)](#) (*requirementslib.Requirement class method*), 11
[from_pipfile\(\)](#) (*requirementslib.models.markers.PipenvMarkers class method*), 18
[from_pipfile\(\)](#) (*requirementslib.models.requirements.FileRequirement class method*), 28
[from_pipfile\(\)](#) (*requirementslib.models.requirements.NamedRequirement class method*), 32
[from_pipfile\(\)](#) (*requirementslib.models.requirements.Requirement class method*), 33
[from_pipfile\(\)](#) (*requirementslib.models.requirements.VCSRequirement class method*), 35
[from_pipfile\(\)](#) (*requirementslib.Requirement class method*), 11
[from_req\(\)](#) (*requirementslib.models.setup_info.BaseRequirement class method*), 38
[from_requirement\(\)](#) (*requirementslib.models.dependencies.AbstractDependency class method*), 13
[from_requirement\(\)](#) (*requirementslib.models.metadata.Dependency class method*), 19
[from_requirement\(\)](#) (*requirementslib.models.setup_info.SetupInfo class method*), 39
[from_str\(\)](#) (*requirementslib.models.metadata.Dependency class method*), 19
[from_string\(\)](#) (*requirementslib.models.dependencies.AbstractDependency class method*), 13
[from_string\(\)](#) (*requirementslib.models.setup_info.BaseRequirement class method*), 38
[full_groupby\(\)](#) (*in module requirementslib.models.utils*), 43
[full_url](#) (*requirementslib.models.url.URI attribute*), 47

G

[gen_marker\(\)](#) (*in module requirementslib.models.markers*), 18
[generic_unparse\(\)](#) (*requirementslib.models.setup_info.Analyzer method*), 37
[generic_visit\(\)](#) (*requirementslib.models.setup_info.Analyzer method*), 37
[get\(\)](#) (*requirementslib.Lockfile method*), 7
[get\(\)](#) (*requirementslib.models.cache.DependencyCache method*), 12
[get\(\)](#) (*requirementslib.models.lockfile.Lockfile method*), 16
[get\(\)](#) (*requirementslib.models.pipfile.Pipfile method*), 25
[get\(\)](#) (*requirementslib.Pipfile method*), 9
[get_abstract_dependencies\(\)](#) (*in module requirementslib.models.dependencies*), 14
[get_abstract_dependencies\(\)](#) (*requirementslib.models.requirements.Requirement method*), 33
[get_abstract_dependencies\(\)](#) (*requirementslib.Requirement method*), 11
[get_build_backend\(\)](#) (*requirementslib.models.setup_info.SetupInfo method*), 39
[get_checkout_dir\(\)](#) (*requirementslib.models.requirements.VCSRequirement*

method), 35

get_commit_hash() (requirementslib.models.requirements.VCSRequirement method), 35

get_commit_hash() (requirementslib.models.vcs.VCSRepository method), 49

get_contained_extras (in module requirementslib.models.markers), 18

get_contained_pyversions (in module requirementslib.models.markers), 18

get_default_pyproject_backend() (in module requirementslib.models.utils), 43

get_dependencies() (in module requirementslib.models.dependencies), 14

get_dependencies() (requirementslib.models.metadata.Package method), 20

get_dependencies() (requirementslib.models.metadata.ReleaseUrl method), 23

get_dependencies() (requirementslib.models.requirements.Requirement method), 33

get_dependencies() (requirementslib.Requirement method), 11

get_dependencies_from_cache() (in module requirementslib.models.dependencies), 14

get_dependencies_from_index() (in module requirementslib.models.dependencies), 15

get_dependencies_from_json() (in module requirementslib.models.dependencies), 15

get_dependencies_from_wheel_cache() (in module requirementslib.models.dependencies), 15

get_deps() (requirementslib.Lockfile method), 7

get_deps() (requirementslib.models.dependencies.AbstractDependency method), 13

get_deps() (requirementslib.models.lockfile.Lockfile method), 16

get_deps() (requirementslib.models.pipfile.Pipfile method), 25

get_deps() (requirementslib.Pipfile method), 9

get_dist_metadata() (in module requirementslib.utils), 49

get_distinfo_dist() (in module requirementslib.models.setup_info), 41

get_egg_metadata() (requirementslib.models.setup_info.SetupInfo method), 39

get_egginfo_dist() (in module requirementslib.models.setup_info), 41

get_extra_name_from_marker (in module requirementslib.models.setup_info), 41

get_extras_from_ireq() (requirementslib.models.setup_info.SetupInfo method), 39

get_extras_from_setupcfg() (in module requirementslib.models.setup_info), 41

get_finder() (in module requirementslib.models.dependencies), 15

get_grouped_dependencies() (in module requirementslib.models.dependencies), 15

get_hash() (requirementslib.models.cache.HashCache method), 13

get_hashes() (requirementslib.models.resolvers.DependencyResolver method), 36

get_hashes_as_pip() (requirementslib.models.requirements.Requirement method), 33

get_hashes_as_pip() (requirementslib.Requirement method), 11

get_hashes_for_one() (requirementslib.models.resolvers.DependencyResolver method), 36

get_host_port_path() (requirementslib.models.url.URI method), 47

get_info() (requirementslib.models.setup_info.SetupInfo method), 39

get_initial_info() (requirementslib.models.setup_info.SetupInfo method), 39

get_ireq() (requirementslib.models.requirements.Line method), 30

get_latest_lockfile() (requirementslib.models.metadata.Package method), 20

get_latest_lockfile() (requirementslib.models.metadata.ReleaseCollection method), 22

get_line() (requirementslib.models.requirements.Line method), 30

get_line_instance() (requirementslib.models.requirements.Requirement method), 33

get_line_instance() (requirementslib.Requirement method), 11

get_link() (requirementslib.models.requirements.FileRequirement method), 28

get_link() (requirementslib.models.requirements.VCSRequirement

method), 35

get_link_from_line() (requirementslib.models.requirements.FileRequirement class method), 28

get_local_wheel_metadata() (in module requirementslib.models.metadata), 24

get_markers() (requirementslib.models.requirements.Requirement method), 33

get_markers() (requirementslib.Requirement method), 11

get_markers_from_wheel() (requirementslib.models.metadata.ReleaseUrl method), 23

get_metadata() (in module requirementslib.models.setup_info), 41

get_metadata_from_dist() (in module requirementslib.models.setup_info), 41

get_metadata_from_wheel() (in module requirementslib.models.setup_info), 41

get_metadata_from_wheel() (requirementslib.models.setup_info.SetupInfo method), 39

get_name() (requirementslib.models.requirements.FileRequirement method), 29

get_name() (requirementslib.models.requirements.Requirement method), 33

get_name() (requirementslib.models.requirements.VCSRequirement method), 35

get_name() (requirementslib.Requirement method), 11

get_name_and_version_from_setupcfg() (in module requirementslib.models.setup_info), 41

get_name_variants() (in module requirementslib.models.utils), 43

get_package() (in module requirementslib.models.metadata), 24

get_package_dir_from_setupcfg() (in module requirementslib.models.setup_info), 41

get_package_from_requirement() (in module requirementslib.models.metadata), 24

get_package_version() (in module requirementslib.models.metadata), 24

get_parsed_url() (requirementslib.models.url.URI class method), 47

get_parsed_url() (requirementslib.models.vcs.VCSRepository method), 49

get_password() (requirementslib.models.url.URI method), 47

get_path() (in module requirementslib.utils), 49

get_pinned_version() (in module requirementslib.models.utils), 43

get_pip_command() (in module requirementslib.models.dependencies), 15

get_pip_options() (in module requirementslib.models.dependencies), 15

get_pyproject() (in module requirementslib.models.utils), 43

get_release() (in module requirementslib.models.metadata), 24

get_releases_from_package() (in module requirementslib.models.metadata), 24

get_remote_sdist_metadata() (in module requirementslib.models.metadata), 25

get_remote_wheel_metadata() (in module requirementslib.models.metadata), 25

get_repo_backend() (requirementslib.models.vcs.VCSRepository method), 49

get_requirement() (requirementslib.models.requirements.FileRequirement method), 29

get_requirement() (requirementslib.models.requirements.NamedRequirement method), 32

get_requirement() (requirementslib.models.requirements.Requirement method), 34

get_requirement() (requirementslib.models.requirements.VCSRequirement method), 35

get_requirement() (requirementslib.Requirement method), 11

get_requirement_specs() (requirementslib.models.requirements.Line class method), 30

get_requirements() (requirementslib.Lockfile method), 7

get_requirements() (requirementslib.models.lockfile.Lockfile method), 17

get_setup_info() (requirementslib.models.requirements.Line method), 30

get_setup_paths() (in module requirementslib.utils), 50

get_setuptools_version() (in module requirementslib.models.utils), 43

get_sorted_version_string() (in module requirementslib.models.markers), 18

get_specifier() (requirementslib.models.requirements.Requirement method), 34

get_specifier() (requirementslib.Requirement

method), 11

get_specifiers() (requirementslib.models.requirements.Requirement method), 34

get_specifiers() (requirementslib.Requirement method), 11

get_specset() (in module requirementslib.models.markers), 18

get_uri() (requirementslib.models.requirements.FileRequirement method), 29

get_url() (requirementslib.models.requirements.Line method), 30

get_url_name() (in module requirementslib.models.utils), 43

get_username() (requirementslib.models.url.URI method), 47

get_vcs_repo() (requirementslib.models.requirements.VCSRequirement method), 35

get_version() (in module requirementslib.models.utils), 44

get_version() (requirementslib.models.requirements.Requirement method), 34

get_version() (requirementslib.Requirement method), 11

get_versions (in module requirementslib.models.markers), 18

get_without_extra() (in module requirementslib.models.markers), 18

get_without_pyversion() (in module requirementslib.models.markers), 19

H

has_sig (requirementslib.models.metadata.ReleaseUrl attribute), 23

hash_cache (requirementslib.models.resolvers.DependencyResolver attribute), 36

HashCode (class in requirementslib.models.cache), 13

hashes (requirementslib.models.resolvers.DependencyResolver attribute), 36

hashes_as_pip (requirementslib.models.requirements.Requirement attribute), 34

hashes_as_pip (requirementslib.Requirement attribute), 11

hidden_auth (requirementslib.models.url.URI attribute), 47

HookCaller (class in requirementslib.models.setup_info), 38

host (requirementslib.models.url.URI attribute), 47

include_incompatible_hashes (requirementslib.models.resolvers.DependencyResolver attribute), 36

init_requirement() (in module requirementslib.models.utils), 44

instance_check_converter() (in module requirementslib.models.metadata), 25

inthat (requirementslib.models.project.SectionDifference attribute), 27

inthis (requirementslib.models.project.SectionDifference attribute), 27

ireq (requirementslib.models.requirements.Line attribute), 30

ireq (requirementslib.models.requirements.Requirement attribute), 34

ireq (requirementslib.models.setup_info.SetupInfo attribute), 40

ireq (requirementslib.Requirement attribute), 11

is_artifact (requirementslib.models.requirements.Line attribute), 30

is_direct_url (requirementslib.models.requirements.FileRequirement attribute), 29

is_direct_url (requirementslib.models.requirements.Line attribute), 30

is_direct_url (requirementslib.models.requirements.Requirement attribute), 34

is_direct_url (requirementslib.models.url.URI attribute), 47

is_direct_url (requirementslib.Requirement attribute), 11

is_editable() (in module requirementslib.utils), 50

is_file (requirementslib.models.requirements.Line attribute), 30

is_file_or_url (requirementslib.models.requirements.Requirement attribute), 34

is_file_or_url (requirementslib.Requirement attribute), 11

is_file_url (requirementslib.models.requirements.Line attribute), 30

is_file_url (requirementslib.models.url.URI attribute), 47

is_implicit_ssh (requirementslib.models.url.URI attribute), 47

is_installable (requirementslib.models.requirements.Line attribute), 30

is_installable (requirementslib.models.url.URI at-

tribute), 47

is_installable_dir() (in module require-mentslib.utils), 50

is_installable_file() (in module require-mentslib.utils), 50

is_instance() (in module require-mentslib.models.markers), 19

is_local (requirementslib.models.requirements.FileRequirement attribute), 29

is_local (requirementslib.models.vcs.VCSRepository attribute), 49

is_named (requirementslib.models.requirements.Line attribute), 30

is_named (requirementslib.models.requirements.Requirement attribute), 34

is_named (requirementslib.Requirement attribute), 11

is_path (requirementslib.models.requirements.Line attribute), 30

is_pinned_requirement() (in module require-mentslib.models.utils), 44

is_python() (in module require-mentslib.models.dependencies), 16

is_remote_artifact (requirementslib.models.requirements.FileRequirement attribute), 29

is_remote_url (requirementslib.models.requirements.Line attribute), 30

is_sdist (requirementslib.models.metadata.ReleaseUrl attribute), 23

is_star() (in module requirementslib.utils), 50

is_synced() (requirementslib.models.project.Project method), 27

is_url (requirementslib.models.requirements.Line attribute), 30

is_vcs (requirementslib.models.requirements.Line attribute), 30

is_vcs (requirementslib.models.requirements.Requirement attribute), 34

is_vcs (requirementslib.models.url.URI attribute), 47

is_vcs (requirementslib.Requirement attribute), 11

is_vcs() (in module requirementslib.utils), 50

is_wheel (requirementslib.models.metadata.ReleaseUrl attribute), 23

is_wheel (requirementslib.models.requirements.Line attribute), 30

is_wheel (requirementslib.models.requirements.Requirement attribute), 34

is_wheel (requirementslib.Requirement attribute), 11

iter_metadata() (in module require-mentslib.models.setup_info), 41

key_from_req() (in module require-mentslib.models.utils), 44

L

latest (requirementslib.models.metadata.Release attribute), 22

latest (requirementslib.models.metadata.ReleaseCollection attribute), 22

latest (requirementslib.models.metadata.ReleaseUrlCollection attribute), 24

latest_sdist (requirementslib.models.metadata.Package attribute), 20

latest_timestamp (requirementslib.models.metadata.Release attribute), 22

latest_timestamp (requirementslib.models.metadata.ReleaseUrlCollection attribute), 24

latest_wheels (requirementslib.models.metadata.Package attribute), 20

Line (class in requirementslib.models.requirements), 29

line_for_ireq (requirementslib.models.requirements.Line attribute), 30

line_instance (requirementslib.models.requirements.Requirement attribute), 34

line_instance (requirementslib.Requirement attribute), 11

line_is_installable (requirementslib.models.requirements.Line attribute), 30

line_part (requirementslib.models.markers.PipenvMarkers attribute), 18

line_part (requirementslib.models.requirements.FileRequirement attribute), 29

line_part (requirementslib.models.requirements.NamedRequirement attribute), 32

line_part (requirementslib.models.requirements.VCSRequirement attribute), 35

line_with_prefix (requirementslib.models.requirements.Line attribute), 30

link (requirementslib.models.requirements.FileRequirement attribute), 29

link (requirementslib.models.requirements.Line attribute), 30

link (requirementslib.models.requirements.LinkInfo attribute), 31

link (requirementslib.models.requirements.VCSRequirement attribute), 35

K

key_from_ireq() (in module require-

[LinkInfo](#) (class in [requirementslib.models.requirements](#)), 31
[load\(\)](#) ([requirementslib.Lockfile](#) class method), 8
[load\(\)](#) ([requirementslib.models.lockfile.Lockfile](#) class method), 17
[load\(\)](#) ([requirementslib.models.metadata.ReleaseCollection](#) class method), 22
[load\(\)](#) ([requirementslib.models.pipfile.Pipfile](#) class method), 25
[load\(\)](#) ([requirementslib.models.pipfile.PipfileLoader](#) class method), 26
[load\(\)](#) ([requirementslib.Pipfile](#) class method), 9
[load_projectfile\(\)](#) ([requirementslib.Lockfile](#) class method), 8
[load_projectfile\(\)](#) ([requirementslib.models.lockfile.Lockfile](#) class method), 17
[load_projectfile\(\)](#) ([requirementslib.models.pipfile.Pipfile](#) class method), 26
[load_projectfile\(\)](#) ([requirementslib.Pipfile](#) class method), 9
[locked_vcs_repo\(\)](#) ([requirementslib.models.requirements.VCSRequirement](#) method), 35
[Lockfile](#) (class in [requirementslib](#)), 7
[Lockfile](#) (class in [requirementslib.models.lockfile](#)), 16
[lockfile](#) ([requirementslib.Lockfile](#) attribute), 8
[lockfile](#) ([requirementslib.models.lockfile.Lockfile](#) attribute), 17
[lockfile](#) ([requirementslib.models.project.Project](#) attribute), 27
[lockfile_from_pipfile\(\)](#) ([requirementslib.Lockfile](#) class method), 8
[lockfile_from_pipfile\(\)](#) ([requirementslib.models.lockfile.Lockfile](#) class method), 17
[lockfile_location](#) ([requirementslib.models.project.Project](#) attribute), 27
[lookup_table\(\)](#) (in module [requirementslib.models.utils](#)), 44

M

[make_base_requirements\(\)](#) (in module [requirementslib.models.setup_info](#)), 41
[make_install_requirement\(\)](#) (in module [requirementslib.models.utils](#)), 44
[make_marker\(\)](#) ([requirementslib.models.markers.PipenvMarkers](#) class method), 18
[marker_from_specifier](#) (in module [requirementslib.models.markers](#)), 19

[marker_string](#) ([requirementslib.models.metadata.ParsedTag](#) attribute), 22
[markers](#) ([requirementslib.models.metadata.Dependency](#) attribute), 19
[markers](#) ([requirementslib.models.metadata.ReleaseUrl](#) attribute), 23
[markers_as_pip](#) ([requirementslib.models.requirements.Requirement](#) attribute), 34
[markers_as_pip](#) ([requirementslib.Requirement](#) attribute), 11
[match_assignment_name\(\)](#) ([requirementslib.models.setup_info.Analyzer](#) method), 37
[match_assignment_str\(\)](#) ([requirementslib.models.setup_info.Analyzer](#) method), 37
[md5_digest](#) ([requirementslib.models.metadata.ReleaseUrl](#) attribute), 23
[merge_items\(\)](#) (in module [requirementslib.utils](#)), 50
[merge_markers\(\)](#) (in module [requirementslib.models.markers](#)), 19
[merge_markers\(\)](#) ([requirementslib.models.requirements.Requirement](#) method), 34
[merge_markers\(\)](#) ([requirementslib.Requirement](#) method), 11
[metadata](#) ([requirementslib.models.requirements.Line](#) attribute), 30
[metadata](#) ([requirementslib.models.setup_info.SetupInfo](#) attribute), 40
[monkeypatch_pip\(\)](#) ([requirementslib.models.vcs.VCSRepository](#) class method), 49

N

[name](#) ([requirementslib.models.metadata.Dependency](#) attribute), 19
[name](#) ([requirementslib.models.metadata.ExtrasCollection](#) attribute), 20
[name](#) ([requirementslib.models.metadata.Package](#) attribute), 20
[name](#) ([requirementslib.models.metadata.Release](#) attribute), 22
[name](#) ([requirementslib.models.metadata.ReleaseUrl](#) attribute), 23
[name](#) ([requirementslib.models.metadata.ReleaseUrlCollection](#) attribute), 24
[name](#) ([requirementslib.models.requirements.FileRequirement](#) attribute), 29
[name](#) ([requirementslib.models.requirements.Line](#) attribute), 30

name (*requirementslib.models.requirements.NamedRequirement* attribute), 32

name (*requirementslib.models.requirements.Requirement* attribute), 34

name (*requirementslib.models.requirements.VCSRequirement* attribute), 35

name (*requirementslib.models.setup_info.BaseRequirement* attribute), 38

name (*requirementslib.models.setup_info.Extra* attribute), 38

name (*requirementslib.models.setup_info.SetupInfo* attribute), 40

name (*requirementslib.models.url.URI* attribute), 47

name (*requirementslib.Requirement* attribute), 11

name_and_specifier (*requirementslib.models.requirements.Line* attribute), 30

name_from_req() (in module *requirementslib.models.utils*), 45

name_with_extras (*requirementslib.models.url.URI* attribute), 47

named_req_from_parsed_line() (in module *requirementslib.models.requirements*), 35

NamedRequirement (class in *requirementslib.models.requirements*), 32

newlines (*requirementslib.Lockfile* attribute), 8

newlines (*requirementslib.models.lockfile.Lockfile* attribute), 17

next() (*requirementslib.models.setup_info.ScandirCloser* method), 39

no_recurse() (*requirementslib.models.setup_info.Analyzer* method), 37

non_yanked_releases (*requirementslib.models.metadata.ReleaseCollection* attribute), 22

normalize_marker_str() (in module *requirementslib.models.markers*), 19

normalize_name() (in module *requirementslib.models.utils*), 45

normalize_specifier_set() (in module *requirementslib.models.markers*), 19

normalized_name (*requirementslib.models.requirements.Requirement* attribute), 34

normalized_name (*requirementslib.Requirement* attribute), 12

O

obtain() (*requirementslib.models.vcs.VCSRepository* method), 49

optional_instance_of() (in module *requirementslib.models.utils*), 45

Package (class in *requirementslib.models.metadata*), 20

PackageEncoder (class in *requirementslib.models.metadata*), 21

PackageInfo (class in *requirementslib.models.metadata*), 21

packages (*requirementslib.models.pipfile.Pipfile* attribute), 26

packages (*requirementslib.Pipfile* attribute), 9

packagetype (*requirementslib.models.metadata.ReleaseUrl* attribute), 23

parent (*requirementslib.models.metadata.Dependency* attribute), 20

parent (*requirementslib.models.metadata.ExtrasCollection* attribute), 20

parse() (*requirementslib.models.requirements.Line* method), 30

parse() (*requirementslib.models.url.URI* class method), 47

parse_extras() (in module *requirementslib.models.utils*), 45

parse_extras() (*requirementslib.models.requirements.Line* method), 30

parse_function_names() (*requirementslib.models.setup_info.Analyzer* method), 37

parse_functions() (*requirementslib.models.setup_info.Analyzer* method), 37

parse_hashes() (*requirementslib.models.requirements.Line* method), 30

parse_ireq() (*requirementslib.models.requirements.Line* method), 31

parse_link() (*requirementslib.models.requirements.Line* method), 31

parse_marker_dict() (in module *requirementslib.models.markers*), 19

parse_markers() (*requirementslib.models.requirements.Line* method), 31

parse_name() (*requirementslib.models.requirements.Line* method), 31

parse_requirement() (*requirementslib.models.requirements.Line* method), 31

parse_setup_cfg() (in module *requirementslib.models.setup_info*), 41

parse_setup_cfg() (*requirementslib.models.setup_info* method), 41

[mentstlib.models.setup_info.SetupInfo](#) method), 40
[parse_setup_function\(\)](#) ([requirementstlib.models.setup_info.Analyzer](#) method), 37
[parse_setup_py\(\)](#) ([requirementstlib.models.setup_info.SetupInfo](#) method), 40
[parse_special_directives\(\)](#) (in module [requirementstlib.models.setup_info](#)), 41
[parse_subdirectory\(\)](#) ([requirementstlib.models.url.URI](#) static method), 47
[parse_tag\(\)](#) (in module [requirementstlib.models.metadata](#)), 25
[parsed_line](#) ([requirementstlib.models.requirements.FileRequirement](#) attribute), 29
[parsed_line](#) ([requirementstlib.models.requirements.NamedRequirement](#) attribute), 32
[parsed_setup_cfg](#) ([requirementstlib.models.requirements.Line](#) attribute), 31
[parsed_setup_py](#) ([requirementstlib.models.requirements.Line](#) attribute), 31
[parsed_url](#) ([requirementstlib.models.requirements.Line](#) attribute), 31
[parsed_version](#) ([requirementstlib.models.metadata.Release](#) attribute), 22
[ParsedTag](#) (class in [requirementstlib.models.metadata](#)), 22
[password](#) ([requirementstlib.models.url.URI](#) attribute), 47
[path](#) ([requirementstlib.Lockfile](#) attribute), 8
[path](#) ([requirementstlib.models.lockfile.Lockfile](#) attribute), 17
[path](#) ([requirementstlib.models.pipfile.Pipfile](#) attribute), 26
[path](#) ([requirementstlib.models.requirements.FileRequirement](#) attribute), 29
[path](#) ([requirementstlib.models.requirements.LinkInfo](#) attribute), 31
[path](#) ([requirementstlib.models.requirements.VCSRequirement](#) attribute), 35
[path](#) ([requirementstlib.models.url.URI](#) attribute), 47
[path](#) ([requirementstlib.Pipfile](#) attribute), 9
[PathAccessError](#), 49
[pep508_url](#) ([requirementstlib.models.metadata.ReleaseUrl](#) attribute), 23
[pep517_config](#) ([requirementstlib.models.setup_info.SetupInfo](#) attribute), 40
[pep517_subprocess_runner\(\)](#) (in module [requirementstlib.models.setup_info](#)), 41
[pin\(\)](#) ([requirementstlib.models.metadata.Dependency](#) method), 20
[pin_dependencies\(\)](#) ([requirementstlib.models.metadata.Package](#) method), 20
[pin_deps\(\)](#) ([requirementstlib.models.resolvers.DependencyResolver](#) method), 36
[pin_history](#) ([requirementstlib.models.resolvers.DependencyResolver](#) attribute), 37
[pip_install\(\)](#) ([requirementstlib.models.setup_info.BuildEnv](#) method), 38
[PipenvMarkers](#) (class in [requirementstlib.models.markers](#)), 18
[Pipfile](#) (class in [requirementstlib](#)), 8
[Pipfile](#) (class in [requirementstlib.models.pipfile](#)), 25
[pipfile](#) ([requirementstlib.models.pipfile.Pipfile](#) attribute), 26
[pipfile](#) ([requirementstlib.models.project.Project](#) attribute), 27
[pipfile](#) ([requirementstlib.Pipfile](#) attribute), 9
[pipfile_entry](#) ([requirementstlib.models.requirements.Requirement](#) attribute), 34
[pipfile_entry](#) ([requirementstlib.Requirement](#) attribute), 12
[pipfile_location](#) ([requirementstlib.models.project.Project](#) attribute), 27
[pipfile_part](#) ([requirementstlib.models.markers.PipenvMarkers](#) attribute), 18
[pipfile_part](#) ([requirementstlib.models.requirements.FileRequirement](#) attribute), 29
[pipfile_part](#) ([requirementstlib.models.requirements.NamedRequirement](#) attribute), 32
[pipfile_part](#) ([requirementstlib.models.requirements.VCSRequirement](#) attribute), 35
[PipfileLoader](#) (class in [requirementstlib.models.pipfile](#)), 26
[platform_system](#) ([requirementstlib.models.metadata.ParsedTag](#) attribute), 22
[populate_metadata\(\)](#) ([requirementstlib.models.metadata.ParsedTag](#) attribute), 22

[mentstlib.models.setup_info.SetupInfo](#) method), 40
[populate_setup_paths\(\)](#) ([requirementstlib.models.requirements.Line](#) method), 31
[populate_source\(\)](#) ([requirementstlib.models.pipfile.PipfileLoader](#) class method), 26
[port](#) ([requirementstlib.models.url.URI](#) attribute), 47
[prefer](#) ([requirementstlib.models.requirements.LinkInfo](#) attribute), 31
[preferred_newlines\(\)](#) (in module [requirementstlib.models.lockfile](#)), 18
[preferred_newlines\(\)](#) (in module [requirementstlib.models.project](#)), 28
[prepare_pip_source_args\(\)](#) (in module [requirementstlib.utils](#)), 50
[Project](#) (class in [requirementstlib.models.project](#)), 27
[ProjectFile](#) (class in [requirementstlib.models.project](#)), 27
[projectfile](#) ([requirementstlib.Lockfile](#) attribute), 8
[projectfile](#) ([requirementstlib.models.lockfile.Lockfile](#) attribute), 17
[projectfile](#) ([requirementstlib.models.pipfile.Pipfile](#) attribute), 26
[projectfile](#) ([requirementstlib.Pipfile](#) attribute), 9
[pyproject](#) ([requirementstlib.models.setup_info.SetupInfo](#) attribute), 40
[pyproject_backend](#) ([requirementstlib.models.requirements.FileRequirement](#) attribute), 29
[pyproject_backend](#) ([requirementstlib.models.requirements.Line](#) attribute), 31
[pyproject_path](#) ([requirementstlib.models.requirements.FileRequirement](#) attribute), 29
[pyproject_requires](#) ([requirementstlib.models.requirements.FileRequirement](#) attribute), 29
[pyproject_requires](#) ([requirementstlib.models.requirements.Line](#) attribute), 31
[pyproject_toml](#) ([requirementstlib.models.requirements.Line](#) attribute), 31
[python_requires](#) ([requirementstlib.models.setup_info.SetupInfo](#) attribute), 40
[python_version](#) ([requirementstlib.models.metadata.Dependency](#) attribute), 20
[python_version](#) ([requirementstlib.models.metadata.ParsedTag](#) attribute), 22
[python_version](#) ([requirementstlib.models.metadata.ReleaseUrl](#) attribute), 23
Q
[query](#) ([requirementstlib.models.url.URI](#) attribute), 47
[query_dict](#) ([requirementstlib.models.url.URI](#) attribute), 47
R
[read\(\)](#) ([requirementstlib.models.project.ProjectFile](#) class method), 27
[read_cache\(\)](#) ([requirementstlib.models.cache.DependencyCache](#) method), 12
[read_cache_file\(\)](#) (in module [requirementstlib.models.cache](#)), 13
[read_projectfile\(\)](#) ([requirementstlib.Lockfile](#) class method), 8
[read_projectfile\(\)](#) ([requirementstlib.models.lockfile.Lockfile](#) class method), 17
[read_projectfile\(\)](#) ([requirementstlib.models.pipfile.Pipfile](#) class method), 26
[read_projectfile\(\)](#) ([requirementstlib.Pipfile](#) class method), 10
[read_source\(\)](#) (in module [requirementstlib.models.utils](#)), 45
[ref](#) ([requirementstlib.models.requirements.Line](#) attribute), 31
[ref](#) ([requirementstlib.models.requirements.VCSRequirement](#) attribute), 35
[ref](#) ([requirementstlib.models.url.URI](#) attribute), 47
[Release](#) (class in [requirementstlib.models.metadata](#)), 22
[ReleaseCollection](#) (class in [requirementstlib.models.metadata](#)), 22
[ReleaseUrl](#) (class in [requirementstlib.models.metadata](#)), 22
[ReleaseUrlCollection](#) (class in [requirementstlib.models.metadata](#)), 24
[reload\(\)](#) ([requirementstlib.models.setup_info.SetupInfo](#) method), 40
[relpath](#) ([requirementstlib.models.requirements.LinkInfo](#) attribute), 31
[remap\(\)](#) (in module [requirementstlib.utils](#)), 50
[remove_keys_from_lockfile\(\)](#) ([requirementstlib.models.project.Project](#) method), 27
[remove_keys_from_pipfile\(\)](#) ([requirementstlib.models.project.Project](#) method), 27

[remove_password_from_url\(\)](#) (in module [requirementslib.models.url](#)), 48
[reorder_source_keys\(\)](#) (in module [requirementslib.models.pipfile](#)), 27
[repo](#) ([requirementslib.models.requirements.VCSRequirement](#) attribute), 35
[req](#) ([requirementslib.models.requirements.FileRequirement](#) attribute), 29
[req](#) ([requirementslib.models.requirements.NamedRequirement](#) attribute), 32
[req](#) ([requirementslib.models.requirements.VCSRequirement](#) attribute), 35
[Requirement](#) (class in [requirementslib](#)), 10
[Requirement](#) (class in [requirementslib.models.requirements](#)), 32
[requirement](#) ([requirementslib.models.metadata.Dependency](#) attribute), 20
[requirement](#) ([requirementslib.models.metadata.Package](#) attribute), 20
[requirement](#) ([requirementslib.models.requirements.Line](#) attribute), 31
[requirement](#) ([requirementslib.models.requirements.Requirement](#) attribute), 34
[requirement](#) ([requirementslib.models.setup_info.BaseRequirement](#) attribute), 38
[requirement](#) ([requirementslib.Requirement](#) attribute), 12
[requirement_info](#) ([requirementslib.models.requirements.Line](#) attribute), 31
[requirements](#) ([requirementslib.Lockfile](#) attribute), 8
[requirements](#) ([requirementslib.models.lockfile.Lockfile](#) attribute), 17
[requirements](#) ([requirementslib.models.pipfile.Pipfile](#) attribute), 26
[requirements](#) ([requirementslib.models.setup_info.Extra](#) attribute), 38
[requirements](#) ([requirementslib.Pipfile](#) attribute), 10
[requirements_list](#) ([requirementslib.Lockfile](#) attribute), 8
[requirements_list](#) ([requirementslib.models.lockfile.Lockfile](#) attribute), 17
[requirementslib](#) (module), 7
[requirementslib.models](#) (module), 12
[requirementslib.models.cache](#) (module), 12
[requirementslib.models.dependencies](#) (module), 13
[requirementslib.models.lockfile](#) (module), 16
[requirementslib.models.markers](#) (module), 18
[requirementslib.models.metadata](#) (module), 19
[requirementslib.models.pipfile](#) (module), 25
[requirementslib.models.project](#) (module), 27
[requirementslib.models.requirements](#) (module), 28
[requirementslib.models.resolvers](#) (module), 36
[requirementslib.models.setup_info](#) (module), 37
[requirementslib.models.url](#) (module), 46
[requirementslib.models.utils](#) (module), 41
[requirementslib.models.vcs](#) (module), 49
[requirementslib.utils](#) (module), 49
[requires](#) ([requirementslib.models.setup_info.SetupInfo](#) attribute), 40
[requires_python](#) ([requirementslib.models.metadata.ReleaseUrl](#) attribute), 23
[requires_python](#) ([requirementslib.models.pipfile.Pipfile](#) attribute), 26
[requires_python](#) ([requirementslib.Pipfile](#) attribute), 10
[RequiresPythonCache](#) (class in [requirementslib.models.cache](#)), 13
[resolution](#) ([requirementslib.models.resolvers.DependencyResolver](#) attribute), 37
[ResolutionError](#), 37
[resolve\(\)](#) ([requirementslib.models.resolvers.DependencyResolver](#) method), 37
[reverse_dependencies\(\)](#) ([requirementslib.models.cache.DependencyCache](#) method), 12
[root](#) ([requirementslib.models.pipfile.Pipfile](#) attribute), 26
[root](#) ([requirementslib.Pipfile](#) attribute), 10
[run_pyproject\(\)](#) ([requirementslib.models.setup_info.SetupInfo](#) method), 40
[run_requires\(\)](#) ([requirementslib.models.requirements.Requirement](#) method), 34
[run_requires\(\)](#) ([requirementslib.Requirement](#) method), 12
[run_setup\(\)](#) (in module [requirementslib](#)), 13

mentslib.models.setup_info), 41

run_setup() (requirementslib.models.setup_info.SetupInfo method), 40

S

safe_string (requirementslib.models.url.URI attribute), 47

ScandirCloser (class in requirementslib.models.setup_info), 38

scheme (requirementslib.models.url.URI attribute), 47

sdists (requirementslib.models.metadata.Release attribute), 22

sdists (requirementslib.models.metadata.ReleaseUrlCollection attribute), 24

sdists() (requirementslib.models.metadata.ReleaseCollection attribute), 22

secret (requirementslib.models.url.URI attribute), 48

section_keys (requirementslib.Lockfile attribute), 8

section_keys (requirementslib.models.lockfile.Lockfile attribute), 17

SectionDifference (class in requirementslib.models.project), 27

serialize() (requirementslib.models.metadata.Package method), 21

setup_cfg (requirementslib.models.requirements.Line attribute), 31

setup_cfg (requirementslib.models.setup_info.SetupInfo attribute), 40

setup_info (requirementslib.models.requirements.FileRequirement attribute), 29

setup_info (requirementslib.models.requirements.Line attribute), 31

setup_info (requirementslib.models.requirements.VCSRequirement attribute), 35

setup_logger() (in module requirementslib.utils), 51

setup_path (requirementslib.models.requirements.FileRequirement attribute), 29

setup_py (requirementslib.models.requirements.Line attribute), 31

setup_py (requirementslib.models.setup_info.SetupInfo attribute), 40

setup_py_dir (requirementslib.models.requirements.FileRequirement attribute), 29

setup_requires (requirementslib.models.setup_info.SetupInfo attribute), 40

SetupInfo (class in requirementslib.models.setup_info), 39

setuptools_parse_setup_cfg() (in module requirementslib.models.setup_info), 41

sha256 (requirementslib.models.metadata.ReleaseUrl attribute), 23

size (requirementslib.models.metadata.ReleaseUrl attribute), 23

sort_releases() (requirementslib.models.metadata.ReleaseCollection method), 22

specifier (requirementslib.models.metadata.Dependency attribute), 20

specifier (requirementslib.models.requirements.Line attribute), 31

specifiers (requirementslib.models.requirements.Line attribute), 31

specifiers (requirementslib.models.requirements.Requirement attribute), 34

specifiers (requirementslib.Requirement attribute), 12

specs_to_string() (in module requirementslib.models.utils), 45

split_hashes() (requirementslib.models.requirements.Line class method), 31

split_keywords() (in module requirementslib.models.metadata), 25

split_markers_from_line() (in module requirementslib.models.utils), 45

split_ref_from_uri() (in module requirementslib.models.utils), 46

split_vcs_method_from_uri() (in module requirementslib.models.utils), 46

stack (requirementslib.models.setup_info.SetupInfo attribute), 40

start_resolver() (in module requirementslib.models.dependencies), 16

strip_extras_markers_from_requirement() (in module requirementslib.models.utils), 46

strip_ssh_from_git_uri() (in module requirementslib.utils), 51

subdirectory (requirementslib.models.requirements.FileRequirement attribute), 29

subdirectory (requirementslib.models.requirements.Line attribute), 31

subdirectory (requirementslib.models.url.URI attribute), 48

T

tags (*requirementslib.models.metadata.ReleaseUrl* attribute), 23

to_dependency() (*requirementslib.models.metadata.PackageInfo* method), 22

to_lockfile() (*requirementslib.models.metadata.Release* method), 22

to_string() (*requirementslib.models.url.URI* method), 48

tomlkit_dict_to_python() (in module *requirementslib.models.utils*), 46

tomlkit_value_to_python() (in module *requirementslib.models.utils*), 46

U

unmap_binops() (*requirementslib.models.setup_info.Analyzer* method), 37

unparse() (*requirementslib.models.setup_info.Analyzer* method), 37

unparse_Assign() (*requirementslib.models.setup_info.Analyzer* method), 37

unparse_Attribute() (*requirementslib.models.setup_info.Analyzer* method), 37

unparse_BinOp() (*requirementslib.models.setup_info.Analyzer* method), 37

unparse_Call() (*requirementslib.models.setup_info.Analyzer* method), 37

unparse_Compare() (*requirementslib.models.setup_info.Analyzer* method), 37

unparse_Constant() (*requirementslib.models.setup_info.Analyzer* method), 37

unparse_Dict() (*requirementslib.models.setup_info.Analyzer* method), 37

unparse_Ellipsis() (*requirementslib.models.setup_info.Analyzer* method), 37

unparse_IfExp() (*requirementslib.models.setup_info.Analyzer* method), 38

unparse_keyword() (*requirementslib.models.setup_info.Analyzer* method), 38

unparse_List() (*requirementslib.models.setup_info.Analyzer* method),

38

unparse_list() (*requirementslib.models.setup_info.Analyzer* method), 38

unparse_Mapping() (*requirementslib.models.setup_info.Analyzer* method), 38

unparse_Name() (*requirementslib.models.setup_info.Analyzer* method), 38

unparse_NameConstant() (*requirementslib.models.setup_info.Analyzer* method), 38

unparse_Num() (*requirementslib.models.setup_info.Analyzer* method), 38

unparse_Str() (*requirementslib.models.setup_info.Analyzer* method), 38

unparse_str() (*requirementslib.models.setup_info.Analyzer* method), 38

unparse_Subscript() (*requirementslib.models.setup_info.Analyzer* method), 38

unparse_Tuple() (*requirementslib.models.setup_info.Analyzer* method), 38

unparse_tuple() (*requirementslib.models.setup_info.Analyzer* method), 38

unsafe_string (*requirementslib.models.url.URI* attribute), 48

update() (*requirementslib.models.vcs.VCSRepository* method), 49

update_from_dict() (*requirementslib.models.setup_info.SetupInfo* method), 40

update_name_from_path() (*requirementslib.models.requirements.Requirement* method), 34

update_name_from_path() (*requirementslib.Requirement* method), 12

update_repo() (*requirementslib.models.requirements.VCSRequirement* method), 35

update_url_name_and_fragment() (in module *requirementslib.models.url*), 48

upload_time (*requirementslib.models.metadata.ReleaseUrl* attribute), 23

upload_time_iso_8601 (*requirementslib.models.metadata.ReleaseUrl* attribute), 24

- URI (class in `requirementslib.models.url`), 46
- `uri` (`requirementslib.models.requirements.FileRequirement` attribute), 29
- `uri` (`requirementslib.models.requirements.LinkInfo` attribute), 32
- `uri` (`requirementslib.models.requirements.VCSRequirement` attribute), 35
- `uri_escape` (`requirementslib.models.url.URI` attribute), 48
- `url` (`requirementslib.models.metadata.ReleaseUrl` attribute), 24
- `url` (`requirementslib.models.requirements.Line` attribute), 31
- `url` (`requirementslib.models.requirements.VCSRequirement` attribute), 35
- `url_without_fragment` (`requirementslib.models.url.URI` attribute), 48
- `url_without_fragment_or_ref` (`requirementslib.models.url.URI` attribute), 48
- `url_without_ref` (`requirementslib.models.url.URI` attribute), 48
- `urls` (`requirementslib.models.metadata.Release` attribute), 22
- `urls` (`requirementslib.models.metadata.ReleaseUrlCollection` attribute), 24
- `username` (`requirementslib.models.url.URI` attribute), 48
- `uses_pep517` (`requirementslib.models.requirements.Requirement` attribute), 34
- `uses_pep517` (`requirementslib.Requirement` attribute), 12
- ## V
- `validate()` (`requirementslib.models.pipfile.PipfileLoader` class method), 26
- `validate_digest()` (in module `requirementslib.models.metadata`), 25
- `validate_extras()` (in module `requirementslib.models.metadata`), 25
- `validate_markers()` (in module `requirementslib.models.utils`), 46
- `validate_path()` (in module `requirementslib.models.utils`), 46
- `validate_specifiers()` (in module `requirementslib.models.utils`), 46
- `validate_vcs()` (in module `requirementslib.models.utils`), 46
- `value` (`requirementslib.models.metadata.Digest` attribute), 20
- `vcs` (`requirementslib.models.requirements.VCSRequirement` attribute), 35
- `vcs_req_from_parsed_line()` (in module `requirementslib.models.requirements`), 35
- `vcs_type` (`requirementslib.models.requirements.LinkInfo` attribute), 32
- `vcs_uri` (`requirementslib.models.requirements.VCSRequirement` attribute), 35
- `vcsrepo` (`requirementslib.models.requirements.Line` attribute), 31
- `VCSRepository` (class in `requirementslib.models.vcs`), 49
- `VCSRequirement` (class in `requirementslib.models.requirements`), 34
- `version` (`requirementslib.models.metadata.Package` attribute), 21
- `version` (`requirementslib.models.metadata.Release` attribute), 22
- `version` (`requirementslib.models.requirements.NamedRequirement` attribute), 32
- `version` (`requirementslib.models.setup_info.SetupInfo` attribute), 40
- `version_from_ireq()` (in module `requirementslib.models.utils`), 46
- `version_set` (`requirementslib.models.dependencies.AbstractDependency` attribute), 14
- `visit_BinOp()` (`requirementslib.models.setup_info.Analyzer` method), 38
- ## W
- `wheel_kwargs` (`requirementslib.models.requirements.Line` attribute), 31
- `wheels` (`requirementslib.models.metadata.Release` attribute), 22
- `wheels` (`requirementslib.models.metadata.ReleaseUrlCollection` attribute), 24
- `wheels()` (`requirementslib.models.metadata.ReleaseCollection` method), 22
- `write()` (`requirementslib.Lockfile` method), 8
- `write()` (`requirementslib.models.lockfile.Lockfile` method), 17
- `write()` (`requirementslib.models.pipfile.Pipfile` method), 26
- `write()` (`requirementslib.models.project.ProjectFile` method), 27
- `write()` (`requirementslib.Pipfile` method), 10
- `write_cache()` (`requirementslib.models.cache.DependencyCache` method), 13
- ## Y
- `yanked` (`requirementslib.models.metadata.Release` attribute), 22

yanked (*requirementslib.models.metadata.ReleaseUrl*
attribute), [24](#)