# repo-helper

*Release 2022.4.4*

**A tool to manage configuration files, build scripts etc. across multiple projects.**

**Dominic Davis-Foster**

**Apr 26, 2024**

# Contents

This project is in an early stage, and some things might not work correctly or break in a new release.

**Note:** The autocommit functionality is currently broken on Windows, but works OK on Linux and macOS.

Chapter

ONE

# Installation

## 1.1 from PyPI

```
$ python3 -m pip install repo_helper --user
```

## 1.2 from Anaconda

First add the required channels

```
$ conda config --add channels https://conda.anaconda.org/conda-forge
$ conda config --add channels https://conda.anaconda.org/domdfcoding
```

Then install

```
$ conda install repo_helper
```

## 1.3 from GitHub

```
$ python3 -m pip install git+https://github.com/repo-helper/repo_helper@master --user
```

Chapter

TWO

# Configuration

## 2.1 Overview

Place configuration options in a file called `repo_helper.yml` in the repository root.

Options are defined like so:

```
modname: repo_helper
copyright_years: "2020"
author: "Dominic Davis-Foster"
email: "dominic@example.com"
version: "0.0.1"
username: "domdfcoding"
license: 'LGPLv3+'
short_desc: 'Update multiple configuration files, build scripts etc. from a single␣
↪location'
```

## 2.2 Metadata

**author**

The name of the package author.

Example:

```
author: Dominic Davis-Foster
```

**Required**: yes

**Type**: String

**classifiers**

A list of "trove classifiers" for PyPI.

Example:

```
classifiers:
  - "Environment :: Console"
```

Classifiers are automatically added for the supported Python versions and implementations, and for most licenses.

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**copyright_years**
The copyright_years of the package.

Examples:

```
version: 2020
```

or

```
version: 2014-2019
```

**Required**: yes

**Type**: String or Integer

**email**
The email address of the author or maintainer.

Example:

```
email: dominic@example.com
```

**Required**: yes

**Type**: String

**import_name**
The name the package is imported with, if different to *modname*.

Example:

```
import_name: repo_helper
```

**Required**: no

**Default**: The value of *modname*

**Type**: String

**keywords**
A list of keywords for the project.

Example:

```
keywords:
  - version control
  - git
  - template
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**license**
The license for the project.

Example:

```
license: GPLv3+
```

Currently understands `LGPLv3`, `LGPLv3`, `GPLv3`, `GPLv3`, `GPLv2` and `BSD`.

**Required**: yes

**Type**: String

**modname**
The name of the package.

Example:

```
modname: repo_helper
```

**Required**: yes

**Type**: String

**on_pypi**
Flag to indicate the package is available on PyPI.

Example:

```
on_pypi: True
```

**Required**: no

**Default**: `True`

**Type**: Boolean

**pure_python**
Flag to indicate the package is pure Python.

Example:

```
pure_python: True
```

**Required**: no

**Default**: `True`

**Type**: Boolean

**pypi_name**
The name of project on PyPI, if different to *modname*.

Example:

```
pypi_name: git-helper
```

**Required**: no

**Default**: The value of *modname*

**Type**: String

**repo_name**
The name of GitHub repository, if different to *modname*.

Example:

```
repo_name: repo_helper
```

**Required**: no

**Default**: The value of *modname*

**Type**: String

**short_desc**
A short description of the project. Used by PyPI.

Example:

```
short_desc: This is a short description of my project.
```

**Required**: yes

**Type**: String

**source_dir**
The directory containing the source code of the project.

Example:

```
source_dir: src
```

By default this is the repository root

**Required**: no

**Default**: <blank>

**Type**: String

**stubs_package**
Flag to indicate the package is a PEP 561 stubs package.

Example:

```
stubs_package: True
```

**Required**: no

**Default**: `False`

**Type**: Boolean

**username**
The username of the GitHub account hosting the repository.

Example:

```
username: domdfcoding
```

**Required**: yes

**Type**: String

**version**
The version of the package.

Example:

```
version: 0.0.1
```

**Required**: yes

**Type**: String or Float

**assignee**
The username of the GitHub account to assign issues to.

Defaults to *username* if unset.

Example:

```
username: repo-helper
assignee: domdfcoding
```

New in version 2020.11.23.

**Required**: no

**Default**: The value of *username*

**Type**: String

## 2.3 Optional features

**docker_name**
The name of the docker image on dockerhub.

Example:

```
docker_name: domdfcoding/fancy_docker_image
```

**Required**: no

**Default**: <blank>

**Type**: String

**docker_shields**
Whether shields for docker container image size and build status should be shown.

Example:

```
docker_shields: True
```

**Required**: no

**Default**: `False`

**Type**: Boolean

**enable_pre_commit**
Whether pre-commit should be installed and configured.

Example:

```
enable_pre_commit: True
```

**Required**: no

**Default**: `True`

**Type**: Boolean

**enable_releases**
Whether packages should be copied from PyPI to GitHub Releases.

Example:

```
enable_releases: True
```

**Required**: no

**Default**: `True`

**Type**: Boolean

## 2.4 Python versions

**python_deploy_version**
The version of Python to use on Travis when deploying to PyPI, Anaconda and GitHub releases.

Example:

```
python_deploy_version: 3.8
```

**Required**: no

**Default**: 3.8

**Type**: String or Float

**python_versions**
A list of the version(s) of Python to use when performing tests with Tox, e.g.

```
python_versions:
  - 3.6
  - 3.7
  - 3.8
  - pypy3
```

If undefined the value of *python_deploy_version* is used instead.

For more advanced configuration, this may instead be a mapping of version number strings to mappings of options, e.g.:

```
python_versions:
  3.6:
  3.7:
  3.8:
  pypy3:
    experimental: false
```

PyPy 3.7 and prerelease versions of CPython are treated as experimental by default unless overridden.

Changed in version 2022.4.4: Added support for mappings as well as lists.

**Required**: no

**Default**: The value of `_default_python_versions`

**Type**: Sequence or Mapping

**requires_python**

The minimum required version of Python.

Example:

```
requires_python: 3.6.1
```

New in version 2021.2.18.

**Required**: no

**Default**: None

**Type**: String or Float

**third_party_version_matrix**

A mapping of third party library names to the version number(s) to test.

The special value "latest" indicates the latest version of the library should be used.

```
third_party_version_matrix:
  attrs:
  - 19.3
  - 20.1
  - 20.2
  - latest
```

This would translate into the following tox testenvs:

```
py36-attrs{19.3,20.1,20.2,latest}
```

and the following tox requirements:

```
attrs19.3: attrs~=19.3.0
attrs20.1: attrs~=20.1.0
attrs20.2: attrs~=20.2.0
attrslatest: attrs
```

which is `<name>~={<version>).0`.

New in version 2020.12.21.

**Note:** Currently matrices are only supported for a single third-party requirement.

**Required**: no

**Default**: { }

**Type**: Mapping of String to Sequence of String or Float

The lowest version of Python given above is used to set the minimum supported version for Pip, PyPI, setuptools etc.

## 2.5 Packaging

**additional_requirements_files**

A list of files containing additional requirements.

These may define "extras" (see *extras_require*). Used in `.readthedocs.yml`.

Example:

```
additional_requirements_files:
  - submodule/requirements.txt
```

This list is automatically populated with any filenames specified in *extras_require*.

Any files specified here are listed in `MANIFEST.in` for inclusion in distributions.

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**additional_setup_args**

A dictionary of additional keyword arguments for `setuptools.setup()`. The values can refer to variables in `__pkginfo__.py`. String values must be enclosed in quotes here.

Example:

```
additional_setup_args:
  author: "'Dominic Davis-Foster'"
  entry_points: "None"
```

**Required**: no

**Default**: { }

**Type**: Mapping of String to String

**console_scripts**

A list of entries for `console_scripts` in `setup.py`. Each entry must follow the same format as required in `setup.py`.

Example:

```
console_scripts:
  - "repo_helper = repo_helper.__main__:main"
  - "git-helper = repo_helper.__main__:main"
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**extras_require**

A dictionary of extra requirements, where the keys are the names of the extras and the values are a list of requirements.

Example:

```
extras_require:
  extra_a:
    - pytz >=2019.1
```

or

```
extras_require:
  extra_a: pytz >=2019.1
```

or

```
extras_require:
  extra_a: < a filename >
```

**Required**: no

**Default**: { }

**Type**: Mapping of String to String

**manifest_additional**

A list of additional entries for `MANIFEST.in`.

Example:

```
manifest_additional:
  - "recursive-include: repo_helper/templates *"
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**platforms**

A case-insensitive list of platforms to perform tests for.

Example:

```
platforms:
  - Windows
  - macOS
  - Linux
```

These values determine the GitHub test workflows to enable, and the Trove classifiers used on PyPI.

**Required**: no

**Default**: ['Windows', 'macOS', 'Linux']

**Type**: Sequence of 'Windows' or 'macOS' or 'Linux'

**Allowed values**: `Windows`, `macOS`, `Linux`

**py_modules**

A list of values for `py_modules` in `setup.py`, which indicate the single-file modules to include in the distributions.

Example:

```
py_modules:
  - domdf_spreadsheet_tools
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**setup_pre**

A list of additional python lines to insert at the beginnning of `setup.py`.

Example:

```
setup_pre:
  - import datetime
  - print(datetim.datetime.today())
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**entry_points**

A mapping of entry point categories to a list of entries for each category.

Each entry should be valid as per https://packaging.python.org/specifications/entry-points/

Example:

```
entry_points:
  pytest11:
    - "nbval = nbval.plugin"
```

**Required**: no

**Default**: { }

**Type**: Mapping of String to Sequence of String

**use_whey**

Whether to use whey to build distributions, rather than `setuptools.build_meta`.

New in version 2021.3.8.

**Required**: no

**Default**: `False`

**Type**: Boolean

**use_flit**
Whether to use flit to build distributions, rather than `setuptools.build_meta`.

New in version 2022.4.4.

**Note:** Support for flit is provisional and experimental.

**Required**: no

**Default**: `False`

**Type**: Boolean

**use_maturin**
Whether to use maturin to build distributions, rather than `setuptools.build_meta`.

New in version $VERSION.

**Note:** Support for maturin is provisional and experimental.

**Required**: no

**Default**: `False`

**Type**: Boolean

**use_hatch**
Whether to use hatch to build distributions, rather than `setuptools.build_meta`.

New in version $VERSION.

**Note:** Support for hatch is provisional and experimental.

**Required**: no

**Default**: `False`

**Type**: Boolean

## 2.6 Documentation

**docs_dir**
The directory containing the docs code of the project.

Example:

```
docs_dir: docs
```

**Required**: no

**Default**: `doc-source`

**Type**: String

**enable_docs**
Whether documentation should be built and deployed.

Example:

```
enable_docs: True
```

**Required**: no

**Default**: `True`

**Type**: Boolean

**extra_sphinx_extensions**
A list of additional extensions to enable for Sphinx.

Example:

```
extra_sphinx_extensions:
  - "sphinxcontrib.httpdomain"
```

These must also be listed in `doc-source/requirements.txt`.

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**html_context**
A dictionary of configuration values for the documentation HTML context.

String values must be encased in quotes.

Example:

```
html_context:
  display_github: True
  github_user: "'domdfcoding'"
```

**Required**: no

**Default**: { }

**Type**: Mapping of String to anything

**html_theme_options**
A dictionary of configuration values for the documentation HTML theme.

String values must be encased in quotes.

Example:

```
html_theme_options:
  logo_only: False
  fixed_sidebar: "'false'"
  github_type: "'star'"
```

**Required**: no

**Default**: { }

**Type**: Mapping of String to anything

**intersphinx_mapping**
A list of additional entries for `intersphinx_mapping` for Sphinx.

Each entry must be enclosed in double quotes.

Example:

```
intersphinx_mapping:
  - "'rtd': ('https://docs.readthedocs.io/en/latest/', None)"
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**preserve_custom_theme**
Whether custom documentation theme styling in `_static/style.css` and `_templates/layout.html` should be preserved.

Example:

```
preserve_custom_theme: True
```

**Required**: no

**Default**: `False`

**Type**: Boolean

**rtfd_author**
The name of the author to show on ReadTheDocs, if different.

Example:

```
rtfd_author: Dominic Davis-Foster and Joe Bloggs
```

**Required**: no

**Default**: The value of *author*

**Type**: String

**sphinx_conf_epilogue**
Like *sphinx_conf_preamble*, but the lines are inserted at the end of the file.

Indent lines with a single tab to form part of the `setup` function.

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**sphinx_conf_preamble**
A list of lines of Python code to add to the top of `conf.py`.

These could be additional settings for Sphinx or calls to extra scripts that must be executed before building the documentation.

Example:

```yaml
sphinx_conf_preamble:
  - "import datetime"
  - "now = datetime.datetime.now()"
  - "strftime = now.strftime('%H:%M')"
  - "print(f'Starting building docs at {strftime}.')"
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

### `sphinx_html_theme`

The HTML theme to use for Sphinx.

Also adds the appropriate values to *extra_sphinx_extensions*, *html_theme_options*, and `html_context_options`.

Example:

```yaml
sphinx_html_theme: alabaster
```

Currently, the supported themes are sphinx_rtd_theme, domdf_sphinx_theme, alabaster, and furo.

**Required**: no

**Default**: `domdf-sphinx-theme`

**Type**: 'sphinx_rtd_theme' or 'sphinx-rtd-theme' or 'alabaster' or 'repo_helper_sphinx_theme' or 'repo-helper-sphinx-theme' or 'domdf_sphinx_theme' or 'domdf-sphinx-theme' or 'furo'

**Allowed values**: `sphinx_rtd_theme`, `sphinx-rtd-theme`, `alabaster`, `repo_helper_sphinx_theme`, `repo-helper-sphinx-theme`, `domdf_sphinx_theme`, `domdf-sphinx-theme`, `furo`

### `standalone_contrib_guide`

Whether the contributing guide for the documentation should be a standlone page.

Example:

```yaml
standalone_contrib_guide: True
```

**Required**: no

**Default**: `False`

**Type**: Boolean

### `docs_url`

The URL of the documentation, if it uses a custom domain. Default `https://{repo_name}.readthedocs.io`.

Example:

```yaml
docs_url: docs.repo-helper.uk
```

**Required**: no

**Default**: None

**Type**: String

**docs_fail_on_warning**
Whether the documentation check on GitHub Actions should fail on warnings.

Example:

```
docs_fail_on_warning: True
```

New in version 2021.2.18.

**Required**: no

**Default**: `False`

**Type**: Boolean

## 2.7 Testing

**enable_tests**
Whether tests should be performed with pytest.

Example:

```
enable_tests: True
```

**Required**: no

**Default**: `True`

**Type**: Boolean

**mypy_deps**
A list of additional packages to install in Tox when running mypy. Usually type stubs.

```
mypy_deps:
  - docutils-stubs
  - webcolors-stubs
  - gi-stubs
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**extra_lint_paths**
A list of additional files or directories to check with flake8 and mypy.

```
extra_lint_paths:
  - tools
  - utils
  - demo.py
```

New in version 2022.4.4.

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**extra_testenv_commands**
A list of additional commands to run in the primary testenv, after running the tests themselves.

```
extra_testenv_commands:
  - python coverage-fixup.py
```

New in version 2022.4.4.

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**mypy_plugins**
A list of plugins to enable for mypy.

Example:

```
mypy_plugins:
  - /one/plugin.py
  - other.plugin
  - custom_plugin:custom_entry_point
```

See https://mypy.readthedocs.io/en/stable/extending_mypy.html#extending-mypy-using-plugins for more info.

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**tests_dir**
The directory containing tests, relative to the repository root.

```
tests_dir: "tests"
```

If undefined it is assumed to be `tests`.

**Required**: no

**Default**: `tests`

**Type**: String

**tox_build_requirements**
A list of additional Python build requirements for Tox.

Example:

```
tox_build_requirements:
  - setuptools
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**tox_requirements**
A list of additional Python requirements for Tox.

Example:

```
tox_requirements:
  - flake8
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**tox_testenv_extras**
The "Extra" requirement to install when installing the package in the Tox testenv.

See https://setuptools.readthedocs.io/en/latest/setuptools.html#declaring-extras-optional-features-with-their-own-dependencies

Example:

```
tox_testenv_extras:
  - docs
```

**Required**: no

**Default**: <blank>

**Type**: String

**mypy_version**
The version of `mypy` to use.

Example:

```
mypy_version: 0.790
```

Changed in version 2021.2.18: The default is now `0.800`

Changed in version 2021.8.11: The default is now `0.910`

Changed in version 2022.4.4: The default is now `0.931`

Changed in version $VERSION: The default is now `0.971`

**Required**: no

**Default**: `0.971`

**Type**: String or Float

**enable_devmode**
Enable Python Development Mode when running tests.

Example:

```
enable_devmode: True
```

**Required**: no

**Default**: `True`

**Type**: Boolean

**tox_unmanaged**

A list of section names in `tox.ini` which should not be managed by `repo-helper`.

Example:

```yaml
tox_unmanaged:
  - "testenv"
  - "flake8"
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**min_coverage**

The minimum permitted test coverage percentage.

Example:

```yaml
mypy_version: 0.790
```

New in version 2020.1.27.

**Required**: no

**Default**: `80`

**Type**: String or Float

**github_ci_requirements**

Additional steps to run in GitHub actions before and after installing dependencies.

New in version 2021.8.11.

Example:

```yaml
github_ci_requirements:
  Linux:
    pre:
     - sudo apt update
     - sudo apt install python3-gi
  macOS:
    post:
     - "Installation Complete!"
```

**Required**: no

**Default**: { }

**Type**: Mapping of String to Mapping of String to Sequence of String

## 2.8 Travis

Options for configuring Travis.

https://travis-ci.com

**travis_additional_requirements**
A list of additional Python requirements for Travis.

Example:

```
travis_additional_requirements:
  - pbr
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**travis_extra_install_post**
Additional steps to run in Travis after installing dependencies.

Example:

```
travis_extra_install_post:
  - echo "Installation Complete!"
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**travis_extra_install_pre**
Additional steps to run in Travis before installing dependencies.

Example:

```
travis_extra_install_pre:
  - sudo apt update
  - sudo apt install python3-gi
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**travis_ubuntu_version**
The Travis Ubuntu version.

Example:

```
travis_ubuntu_version: "xenial"
```

**Required**: no

**Default**: `focal`

**Type**: 'focal' or 'bionic' or 'xenial' or 'trusty' or 'precise'

**Allowed values**: `focal`, `bionic`, `xenial`, `trusty`, `precise`

## 2.9 Conda & Anaconda

**conda_channels**
A list of Anaconda channels required to build and use the Conda package.

Example:

```
conda_channels:
  - domdfcoding
  - conda-forge
  - bioconda
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**on_conda_forge**
Flag to indicate the package is available on conda-forge.

If this flag is `True` the documentation will recommend installing from conda-forge over the *primary_conda_channel*.

Example:

```
on_conda_forge: True
```

New in version $VERSION.

**Required**: no

**Default**: `False`

**Type**: Boolean

**primary_conda_channel**
The Conda channel the package can be downloaded from.

This is automatically added to *conda_channels*.

Defaults to *username* if unset.

Example:

```
username: repo-helper
primary_conda_channel: domdfcoding
```

New in version 2020.12.17.

**Required**: no

**Default**: The value of *username*

**Type**: String

**conda_extras**
A list of extras (see *extras_require*) to include as requirements in the Conda package.

The special keyword `all` indicates all extras should be included.
The special keyword `none` indicates no extras should be included.

Example:

```
conda_extras:
  - plotting
  - xml
```

New in version 2020.11.12.

**Required**: no

**Default**: ['all']

**Type**: Sequence of String

**conda_description**
A short description of the project for Anaconda.

Example:

```
conda_description: This is a short description of my project.
```

A list of required Anaconda channels is automatically appended.

**Required**: no

**Default**: The value of *short_desc*

**Type**: String

**enable_conda**
Whether conda packages should be built and deployed.

Example:

```
enable_conda: True
```

**Required**: no

**Default**: `True`

**Type**: Boolean

## 2.10 Other

**additional_ignore**
A list of additional entries for `.gitignore`.

Example:

```
additional_ignore:
  - "*.pyc"
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**exclude_files**

A list of files not to manage with *repo_helper*.

```
exclude_files:
  - conf
  - tox
```

Valid values are as follows:

| Value | File(s) that will not be managed |
|---|---|
| lint_roller | `lint_roller.sh` |
| stale_bot | `.github/stale.yml` |
| auto_assign | `.github/workflow/assign.yml` and `.github/auto_assign.yml` |
| readme | `README.rst` |
| doc_requirements | `doc-source/requirements.txt` |
| pylintrc | `.pylintrc` |
| manifest | `MANIFEST.in` |
| setup | `setup.py` |
| pkginfo | `__pkginfo__.py` |
| conf | `doc-source/conf.py` |
| gitignore | `.gitignore` |
| rtfd | `.readthedocs.yml` |
| travis | `.travis.yml` |
| tox | `tox.ini` |
| test_requirements | *`tests_dir`*`/requirements.txt` |
| dependabot | `.dependabot/config.yml` |
| make_conda_recipe | `make_conda_recipe.py` |
| bumpversion | `.bumpversion.cfg` |
| issue_templates | `.github/ISSUE_TEMPLATE/bug_report.md` and `.github/ISSUE_TEMPLATE/feature_request.md` |
| 404 | `<docs_dir>/not-found.png` and `<docs_dir>/404.rst` |
| make_isort | `isort.cfg` |

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**imgbot_ignore**

A list of additional glob ignores for imgbot.

Example:

```
imgbot_ignore:
  - "**/*.svg"
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**yapf_exclude**

A list of regular expressions to use to exclude files and directories from autoformatting.

Example:

```
yapf_exclude:
  - ".*/templates/.*"
```

**Required**: no

**Default**: [ ]

**Type**: Sequence of String

**pre_commit_exclude**

Regular expression for files that should not be checked by pre_commit.

```
pre_commit_exclude: "^.*\\._py$"
```

**Required**: no

**Default**: `^$`

**Type**: String

**desktopfile**

A key value mapping of entries for a Linux `.desktop` file.

```
desktopfile:
  Exec: wxIconSaver
  Icon: document-save
```

`Version`, `Name` and `Comment` are pre-populated from *version*, *modname* and *short_desc*.

New in version 2020.11.15.

**Required**: no

**Default**: { }

**Type**: Mapping of String to String

Chapter

THREE

# Usage

## 3.1 repo-helper

Update files in the given repositories, based on settings in 'repo_helper.yml'.

```
repo-helper [OPTIONS] COMMAND [ARGS]...
```

### Options

**-f, --force**
Run 'repo_helper' even when the git working directory is not clean.

**-y, --commit, -n, --no-commit**
Commit or do not commit any changed files.

**Default** Ask first

**-m, --message** <message>
The commit message to use.

**Default** Updated files with 'repo_helper'.

**--version**
Show the version and exit.

## 3.2 repo-helper add

Add metadata.

New in version 2021.8.11.

### 3.2.1 requirement

Add a requirement.

```
repo-helper add requirement [OPTIONS] REQUIREMENT
```

#### Options

**--file** <file>
The file to add the requirement to.

#### Arguments

**REQUIREMENT**
Required argument.

### 3.2.2 typed

Add a 'py.typed' file and the associated trove classifier.

```
repo-helper add typed [OPTIONS]
```

### 3.2.3 version

Add a new Python version to test on.

```
repo-helper add version [OPTIONS] [VERSION]...
```

#### Arguments

**VERSION**
Optional argument(s). Default None

## 3.3 repo-helper broomstick

Clean up build and test artefacts .

Removes the following:

- build
- .mypy_cache
- .pytest_cache
- **/__pytest__
- *.egg-info

```
repo-helper broomstick [OPTIONS]
```

### Options

**-v, --verbose**
Show verbose output.

**--rm-tox**
Also remove the '.tox' directory

## 3.4 repo-helper init

Initialise the repository with some boilerplate files.

```
repo-helper init [OPTIONS]
```

### Options

**-m, --message** <message>
The commit message to use.

**Default** Initialised repository with 'repo_helper'.

**-y, --commit, -n, --no-commit**
Commit or do not commit any changed files.

**Default** Commit automatically

**-f, --force**
Run 'repo_helper' even when the git working directory is not clean.

## 3.5 repo-helper make-recipe

Register the schema mapping for `repo_helper.yml` with PyCharm.

```
repo-helper make-recipe
```

This command takes no options.

## 3.6 repo-helper release

Make a release .

```
repo-helper release [OPTIONS] COMMAND [ARGS]...
```

#### Commands

**major**
Bump to the next major version.

**minor**
Bump to the next minor version.

**patch**
Bump to the next patch version.

**today**
Bump to the calver version for today's date, such as 2020.12.25.

**<version>**
Bump to the given version.

**Options**

Each command takes the following options:

**-m, --message** <message>
The commit message to use.

**Default** Bump version {current_version} -> {new_version}

**-y, --commit, -n, --no-commit**
Commit or do not commit any changed files.

**Default** Commit automatically

**-f, --force**
Make a release even when the git working directory is not clean.

## 3.7 repo-helper show

Show information about the repository.

### 3.7.1 log

Show git commit log.

```
repo-helper show log [OPTIONS]
```

#### Options

**--no-pager**
Disable the output pager.

**--colour, --no-colour**
Whether to use coloured output.

**--from-tag** <from_tag>
Show commits after the given tag.

**--from-date** <from_date>
Show commits after the given date.

**-r, --reverse**
Print entries in reverse order.

**-n, --entries** <entries>
Maximum number of entries to display.

### 3.7.2 changelog

Show commits since the last version tag.

```
repo-helper show changelog [OPTIONS]
```

#### Options

**--no-pager**
Disable the output pager.

**--colour, --no-colour**
Whether to use coloured output.

**-r, --reverse**
Print entries in reverse order.

**-n, --entries** <entries>
Maximum number of entries to display.

### 3.7.3 requirements

Lists the requirements of this library, and their dependencies.

```
repo-helper show requirements [OPTIONS]
```

#### Options

**--no-venv**
Don't search a 'venv' directory in the repository for the requirements.

**-c, --concise**
Show a consolidated list of all dependencies.

**-d, --depth** <depth>
The maximum depth to display. -1 means infinite depth.

**Default** -1

**--no-pager**
Disable the output pager.

Changed in version 2020.12.3: Added the `-c / --concise` option.

### 3.7.4 version

Show the repository version.

```
repo-helper show version [OPTIONS]
```

#### Options

**-q, --quiet**
Print only the version number.

## 3.8 repo-helper suggest

Suggest trove classifiers and keywords.

### 3.8.1 classifiers

Suggest trove classifiers based on repository metadata.

```
repo-helper suggest classifiers [OPTIONS]
```

#### Options

**-l, --library, --not-library**
Indicates this project is a library for developers.

**-s, --status** <status>
The Development Status of this project.

**--add, --no-add**
Add the classifiers to the 'repo_helper.yml' file.

## 3.9 repo-helper wizard

Run the wizard to create a `repo_helper.yml` file.

```
repo-helper wizard
```

This command takes no options.

Chapter

# FOUR

# Public API

## 4.1 repo_helper.blocks

Reusable blocks of reStructuredText.

**Classes:**

| | |
|---|---|
| *ShieldsBlock*(username, repo_name, version, *) | Create the shields block for insertion into the README, documentation etc. |

**Functions:**

| | |
|---|---|
| *create_docs_install_block*(repo_name, username) | Create the installation instructions for insertion into the documentation. |
| *create_docs_links_block*(username, repo_name) | Create the documentation links block. |
| *create_readme_install_block*(modname, username) | Create the installation instructions for insertion into the README. |
| *create_short_desc_block*(short_desc) | Creates the short description block insertion into the README, documentation etc. |
| *get_docs_installation_block_template*() | Loads the docs_installation_block template from file and returns a jinja2 `jinja2.environment.Template` for it. |
| *get_docs_links_block_template*() | Loads the docs_links_block template from file and returns a jinja2 `jinja2.environment.Template` for it. |
| *get_readme_installation_block_no_pypi_template*() | Loads the readme_installation_block_no_pypi template from file and returns a jinja2 `jinja2.environment.Template` for it. |
| *get_readme_installation_block_template*() | Loads the readme_installation_block template from file and returns a jinja2 `jinja2.environment.Template` for it. |

**Data:**

| | |
|---|---|
| *installation_regex* | Regular expression to match the installation block placeholder. |
| *links_regex* | Regular expression to match the links block placeholder. |
| *shields_regex* | Regular expression to match the shields block placeholder. |
| *short_desc_regex* | Regular expression to match the short description block placeholder. |

**class ShieldsBlock**(*username*, *repo_name*, *version*, *, *conda=True*, *tests=True*, *docs=True*, *docs_url='https://{}.readthedocs.io/en/latest/?badge=latest'*, *pypi_name=None*, *unique_name=''*, *docker_shields=False*, *docker_name=''*, *platforms=None*, *on_pypi=True*, *primary_conda_channel=None*)

Bases: `object`

Create the shields block for insertion into the README, documentation etc.

**Parameters**

- **username** (`str`) – The username of the GitHub account that owns the repository.
- **repo_name** (`str`) – The name of the repository.
- **version** (`Union[str, int]`)
- **conda** (`bool`) – Default `True`.
- **tests** (`bool`) – Default `True`.
- **docs** (`bool`) – Default `True`.
- **docs_url** (`str`) – Default `'https://{}.readthedocs.io/en/latest/?badge=latest'`.
- **pypi_name** (`Optional[str]`) – The name of the project on PyPI. Defaults to the value of `repo_name` if unset. Default `None`.
- **unique_name** (`str`) – An optional unique name for the reST substitutions. Default `''`.
- **docker_shields** (`bool`) – Whether to show shields for Docker. Default `False`. Default `False`.
- **docker_name** (`str`) – The name of the Docker image on DockerHub. Default `''`.
- **platforms** (`Optional[Iterable[str]]`) – List of supported platforms. Default `None`.
- **on_pypi** (`bool`) – Default `True`.
- **primary_conda_channel** (`Optional[str]`) – The Conda channel the package can be downloaded from. Default `None`.

New in version 2020.12.11.

**Methods:**

| | |
|---|---|
| *make*() | Constructs the contents of the shields block. |
| *set_docs_mode*() | Create shields for insertion into Sphinx documentation. |
| *set_readme_mode*() | Create shields for insertion into `README.rst`. |

**Attributes:**

| | |
|---|---|
| *sections* | This list controls which sections are included, and their order. |
| *substitutions* | This list controls which substitutions are included, and their order. |

**make**()

Constructs the contents of the shields block.

**Return type** StringList

**sections = ('Docs', 'Tests', 'PyPI', 'Anaconda', 'Activity', 'QA', 'Docker', 'Other')**
**Type:** tuple

This list controls which sections are included, and their order.

**set_docs_mode**()
Create shields for insertion into Sphinx documentation.

**set_readme_mode**()
Create shields for insertion into README.rst.

**substitutions = ('docs', 'docs_check', 'actions_linux', 'actions_windows', 'actions_ma**
**Type:** tuple

This list controls which substitutions are included, and their order.

**create_docs_install_block**(*repo_name*, *username*, *conda=True*, *pypi=True*, *pypi_name=None*, *conda_channels=None*)
Create the installation instructions for insertion into the documentation.

**Parameters**

- **repo_name** (str) – The name of the GitHub repository.
- **username** (str) – The username of the GitHub account that owns the repository. (Not used; ensures API compatibility with *create_readme_install_block()*)
- **conda** (bool) – Whether to show Anaconda installation instructions. Default True.
- **pypi** (bool) – Whether to show PyPI installation instructions. Default True.
- **pypi_name** (Optional[str]) – The name of the project on PyPI. Defaults to the value of repo_name if unset. Default None.
- **conda_channels** (Optional[Sequence[str]]) – List of required Conda channels. Default None.

**Return type** str

**Returns** The installation block created from the above settings.

**create_docs_links_block**(*username*, *repo_name*)
Create the documentation links block.

**Parameters**

- **username** (str) – The username of the GitHub account that owns the repository.
- **repo_name** (str) – The name of the GitHub repository.

**Return type** str

**Returns** The documentation links block created from the above settings.

**create_readme_install_block**(*modname*, *username*, *conda=True*, *pypi=True*, *pypi_name=None*, *conda_channels=None*)
Create the installation instructions for insertion into the README.

**Parameters**

- **modname** (str) – The name of the program / library.
- **username** (str) – The username of the GitHub account that owns the repository.
- **conda** (bool) – Whether to show Anaconda installation instructions. Default True.
- **pypi** (bool) – Whether to show PyPI installation instructions. Default True.
- **pypi_name** (Optional[str]) – The name of the project on PyPI. Defaults to the value of repo_name if unset. Default None.
- **conda_channels** (Optional[Sequence[str]]) – List of required Conda channels. Default None.

**Return type** str

**Returns** The installation block created from the above settings.

**create_short_desc_block**(*short_desc*)

Creates the short description block insertion into the README, documentation etc.

**Parameters** **short_desc** (str) – A short description of the program / library.

**Return type** str

**Returns** The short description block created from the above settings.

**get_docs_installation_block_template**()

Loads the docs_installation_block template from file and returns a jinja2 jinja2.environment.Template for it.

**Return type** Template

**get_docs_links_block_template**()

Loads the docs_links_block template from file and returns a jinja2 jinja2.environment.Template for it.

**Return type** Template

**get_readme_installation_block_no_pypi_template**()

Loads the readme_installation_block_no_pypi template from file and returns a jinja2 jinja2.environment.Template for it.

New in version 2020.12.1.

**Return type** Template

**get_readme_installation_block_template**()

Loads the readme_installation_block template from file and returns a jinja2 jinja2.environment.Template for it.

**Return type** Template

**installation_regex**

**Type:** Pattern

Regular expression to match the installation block placeholder.

| Pattern | (.. start installation)(.*?)(.. end installation) |
|---|---|
| Flags | re.DOTALL |

**links_regex**
**Type:** Pattern

Regular expression to match the links block placeholder.

| Pattern | (.. start links)(.*?)(.. end links) |
|---|---|
| Flags | re.DOTALL |

**shields_regex**
**Type:** Pattern

Regular expression to match the shields block placeholder.

| Pattern | (.. start shields)(.*?)(.. end shields) |
|---|---|
| Flags | re.DOTALL |

**short_desc_regex**
**Type:** Pattern

Regular expression to match the short description block placeholder.

| Pattern | (.. start short[-_]desc)(.*?)(.. end short[-_]desc) |
|---|---|
| Flags | re.DOTALL |

## 4.2 repo_helper.conda

Utilities for Conda packages.

**Functions:**

| *make_recipe*(repo_dir, recipe_file) | Make a Conda meta.yaml recipe. |
|---|---|

**make_recipe** (*repo_dir*, *recipe_file*)
Make a Conda meta.yaml recipe.

**Parameters**

- **repo_dir** (Union[str, Path, PathLike]) – The repository directory.
- **recipe_file** (Union[str, Path, PathLike]) – The file to save the recipe as.

New in version 2020.11.10.

## 4.3 repo_helper.core

Core functionality of repo_helper.

**Classes:**

| | |
|---|---|
| *RepoHelper*(target_repo[, managed_message]) | Repo Helper: Manage configuration files with ease. |

**Functions:**

| | |
|---|---|
| *import_registered_functions*() | Returns a list of all registered functions. |

**class RepoHelper**(*target_repo*, *managed_message="This file is managed by 'repo_helper'. Don't edit it directly."*)

Repo Helper: Manage configuration files with ease.

**Parameters**

- **target_repo** (Union[str, Path, PathLike]) – The path to the root of the repository to manage files for.
- **managed_message** (str) – Message placed at the top of files to indicate that they are managed by repo_helper. Default "This file is managed by 'repo_helper'. Don't edit it directly.".

**Attributes:**

| | |
|---|---|
| *exclude_files* | A tuple of excluded files that should **NOT** be managed by Git Helper. |
| *files* | List of functions to manage files. |
| *managed_message* | Message placed at the top of files to indicate that they are managed by repo_helper. |
| *repo_name* | The name of the repository being managed. |
| *target_repo* | The target repository |
| *templates* | Provides the templates and stores the configuration. |

**Methods:**

| | |
|---|---|
| *load_settings*([allow_unknown_keys]) | Load settings from the repo_helper.yml file in the repository. |
| *run*() | Run Git Helper for the repository and update all managed files. |

**property exclude_files**

A tuple of excluded files that should **NOT** be managed by Git Helper.

**Return type** Tuple[str, ...]

**files**

**Type:** *Management*

List of functions to manage files.

**load_settings**(*allow_unknown_keys=False*)
Load settings from the `repo_helper.yml` file in the repository.

**Parameters** **allow_unknown_keys** (`bool`) – Whether unknown keys should be allowed in the configuration file. Default `False`.

Changed in version 2021.2.18:

- This method is no longer called automatically when instantiating the *RepoHelper* class.
- Added the `allow_unknown_keys` argument.

**property managed_message**
Message placed at the top of files to indicate that they are managed by `repo_helper`.

**Return type** `str`

**property repo_name**
The name of the repository being managed.

**Return type** `str`

**run**()
Run Git Helper for the repository and update all managed files.

**Return type** `List[str]`

**Returns** A list of files managed by Git Helper, regardless of whether they were added, removed or modified.

**target_repo**
**Type:** `PathPlus`

The target repository

**templates**
**Type:** `Environment`

Provides the templates and stores the configuration.

**import_registered_functions**()
Returns a list of all registered functions.

**Return type** `List[Type]`

## 4.4 repo_helper.release

Functions for making tagged releases.

New in version 2020.12.29.

**Classes:**

| | |
|---|---|
| *Bumper*(repo_path[, force]) | Class to bump the repository version. |

continues on next page

Table 11 – continued from previous page

| | |
|---|---|
| *BumpversionFileConfig* | Represents the subset of `bumpversion` per-file configuration values used by `repo-helper`. |

**class Bumper**(*repo_path*, *force=False*)

Bases: `object`

Class to bump the repository version.

**Parameters**

- **repo_path** (`PathPlus`)
- **force** (`bool`) – Whether to force bumping the version when the repository is unclean. Default `False`.

**Methods:**

| | |
|---|---|
| *bump*(new_version, commit, message) | Bump to the given version. |
| *bump_version_for_file*(filename, config) | Bumps the version for the given file. |
| *get_bumpversion_config*(current_version, ...) | Returns the bumpversion config. |
| *get_current_version*() | Returns the current version from the `repo_helper.yml` configuration file. |
| *major*(commit, message) | Bump to the next major version. |
| *minor*(commit, message) | Bump to the next minor version. |
| *patch*(commit, message) | Bump to the next patch version. |
| *today*(commit, message) | Bump to the calver version for today's date. |

**Attributes:**

| | |
|---|---|
| *bumpversion_file* | The path to the bumpversion configuration file. |
| *current_version* | |
| *repo* | |

**bump**(*new_version*, *commit*, *message*)

Bump to the given version.

**Parameters**

- **new_version** (`Version`)
- **commit** (`Optional[bool]`) – Whether to commit automatically (`True`) or ask first (`None`).
- **message** (`str`) – The commit message.

Changed in version 2021.8.11: Now takes a `packaging.version.Version` rather than a `domdf_python_tools.versions.Version`.

**bump_version_for_file**(*filename*, *config*)

Bumps the version for the given file.

**Parameters**

- **filename** (`Union[str, Path, PathLike]`)
- **config** (*BumpversionFileConfig*)

**bumpversion_file**
The path to the bumpversion configuration file.

**current_version**

**get_bumpversion_config**(*current_version*, *new_version*)
Returns the bumpversion config.

**Parameters**

- **current_version** (str)
- **new_version** (str)

**Return type** Dict[str, *BumpversionFileConfig*]

**get_current_version**()
Returns the current version from the `repo_helper.yml` configuration file.

**Return type** Version

**major**(*commit*, *message*)
Bump to the next major version.

**Parameters**

- **commit** (Optional[bool]) – Whether to commit automatically (True) or ask first (None).
- **message** (str) – The commit message.

**minor**(*commit*, *message*)
Bump to the next minor version.

**Parameters**

- **commit** (Optional[bool]) – Whether to commit automatically (True) or ask first (None).
- **message** (str) – The commit message.

**patch**(*commit*, *message*)
Bump to the next patch version.

**Parameters**

- **commit** (Optional[bool]) – Whether to commit automatically (True) or ask first (None).
- **message** (str) – The commit message.

**repo**

**today**(*commit*, *message*)
Bump to the calver version for today's date.

E.g. 2020.12.25 for 25th December 2020

**Parameters**

- **commit** (Optional[bool]) – Whether to commit automatically (True) or ask first (None).
- **message** (str) – The commit message.

**typeddict BumpversionFileConfig**

Bases: TypedDict

Represents the subset of `bumpversion` per-file configuration values used by `repo-helper`.

**Required Keys**

- **search** (`str`)
- **replace** (`str`)

## 4.5 repo_helper.shields

Create a variety of shields, most powered by `https://shields.io/`.

**Functions:**

| | |
|---|---|
| *make_actions_linux_shield*(repo_name, username) | Create a shield to indicate the status of the tests on Linux. |
| *make_actions_linux_shield*(repo_name, username) | Create a shield to indicate the status of the tests on Linux. |
| *make_actions_macos_shield*(repo_name, username) | Create a shield to indicate the status of the tests on macOS. |
| *make_actions_shield*(repo_name, username, ...) | Create a shield to indicate the status of the tests on Linux. |
| *make_actions_windows_shield*(repo_name, username) | Create a shield to indicate the status of the tests on Windows. |
| *make_activity_shield*(repo_name, username, ...) | Create a shield to show the number of commits to the GitHub repository since the last release. |
| *make_codefactor_shield*(repo_name, username) | Create a shield to show the Codefactor code quality grade. |
| *make_conda_platform_shield*(pypi_name, username) | Create a shield to show the supported Conda platforms. |
| *make_conda_version_shield*(pypi_name, username) | Create a shield to show the version on Conda. |
| *make_coveralls_shield*(repo_name, username) | Create a shield to show the code coverage from Coveralls. |
| *make_docker_automated_build_shield*(...) | Create a shield to indicate the Docker automated build status. |
| *make_docker_build_status_shield*(docker_name, ...) | Create a shield to indicate the Docker image build status. |
| *make_docker_size_shield*(docker_name, username) | Create a shield to indicate the size of a docker image. |
| *make_docs_check_shield*(repo_name, username) | Create a shield for the GitHub Actions "Docs Check" status. |
| *make_language_shield*(repo_name, username) | Create a shield to show the primary language of the GitHub repository. |
| *make_last_commit_shield*(repo_name, username) | Create a shield to indicate when the last commit to the GitHub repository occurred. |
| *make_license_shield*(repo_name, username) | Create a shield to show the license of the GitHub repository. |
| *make_maintained_shield*() | Create a shield to indicate that the project is maintained. |

continues on next page

Table 14 – continued from previous page

| | |
|---|---|
| *make_pre_commit_ci_shield*(repo_name, username) | Create a shield to show the pre-commit.ci status. |
| *make_pre_commit_shield*() | Create a shield to show that a repository is configured for use with pre-commit. |
| *make_pypi_downloads_shield*(pypi_name) | Create a shield to show the PyPI download statistics. |
| *make_pypi_version_shield*(pypi_name) | Create a shield to show the version on PyPI. |
| *make_python_implementations_shield*(pypi_name) | Create a shield to show the supported Python implementations for the library. |
| *make_python_versions_shield*(pypi_name) | Create a shield to show the supported Python versions for the library. |
| *make_requires_shield*(repo_name, username) | Create a shield to show the dependency-dash requirements status. |
| *make_rtfd_shield*(repo_name[, target]) | Create a shield for the ReadTheDocs documentation build status. |
| *make_typing_shield*() | Create a shield to show that a library has **PEP 484** Type Hints / Annotations. |
| *make_wheel_shield*(pypi_name) | Create a shield to show whether the library has a wheel on PyPI. |

**make_actions_linux_shield**(*repo_name*, *username*)

Create a shield to indicate the status of the tests on Linux.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_actions_linux_shield**(*repo_name*, *username*)

Create a shield to indicate the status of the tests on Linux.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_actions_macos_shield**(*repo_name*, *username*)

Create a shield to indicate the status of the tests on macOS.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_actions_shield**(*repo_name*, *username*, *name*, *alt*)
Create a shield to indicate the status of the tests on Linux.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **username** (str) – The username of the GitHub account that owns the repository.
- **name** (str) – The name of the workflow.
- **alt** (str) – Alternative text for the image when it cannot be shown.

**Return type** str

**Returns** The shield.

New in version 2020.12.16.

**make_actions_windows_shield**(*repo_name*, *username*)
Create a shield to indicate the status of the tests on Windows.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_activity_shield**(*repo_name*, *username*, *version*)
Create a shield to show the number of commits to the GitHub repository since the last release.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **username** (str) – The username of the GitHub account that owns the repository.
- **version** (Union[str, float])

**Return type** str

**Returns** The shield.

**make_codefactor_shield**(*repo_name*, *username*)
Create a shield to show the Codefactor code quality grade.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_conda_platform_shield**(*pypi_name*, *username*)
Create a shield to show the supported Conda platforms.

**Parameters**

- **pypi_name** (str) – The name of the project on PyPI.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_conda_version_shield**(*pypi_name*, *username*)
Create a shield to show the version on Conda.

**Parameters**

- **pypi_name** (str) – The name of the project on PyPI.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_coveralls_shield**(*repo_name*, *username*)
Create a shield to show the code coverage from Coveralls.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_docker_automated_build_shield**(*docker_name*, *username*)
Create a shield to indicate the Docker automated build status.

**Parameters**

- **docker_name** (str)
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_docker_build_status_shield**(*docker_name*, *username*)
Create a shield to indicate the Docker image build status.

**Parameters**

- **docker_name** (str)
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_docker_size_shield**(*docker_name*, *username*)
Create a shield to indicate the size of a docker image.

**Parameters**

- **docker_name** (str) – The name of the Docker image on DockerHub.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_docs_check_shield**(*repo_name*, *username*)
Create a shield for the GitHub Actions "Docs Check" status.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_language_shield**(*repo_name*, *username*)
Create a shield to show the primary language of the GitHub repository.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_last_commit_shield**(*repo_name*, *username*)
Create a shield to indicate when the last commit to the GitHub repository occurred.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_license_shield**(*repo_name*, *username*)
Create a shield to show the license of the GitHub repository.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_maintained_shield**()
Create a shield to indicate that the project is maintained.

**Returns** The shield.

**Return type** str

**make_pre_commit_ci_shield**(*repo_name*, *username*)
Create a shield to show the pre-commit.ci status.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_pre_commit_shield**()
Create a shield to show that a repository is configured for use with pre-commit.

**Returns** The shield.

**Return type** str

**make_pypi_downloads_shield**(*pypi_name*)
Create a shield to show the PyPI download statistics.

**Parameters** **pypi_name** (str) – The name of the project on PyPI.

**Return type** str

**Returns** The shield.

**make_pypi_version_shield**(*pypi_name*)
Create a shield to show the version on PyPI.

**Parameters** **pypi_name** (str) – The name of the project on PyPI.

**Return type** str

**Returns** The shield.

**make_python_implementations_shield**(*pypi_name*)
Create a shield to show the supported Python implementations for the library.

**Parameters** **pypi_name** (str) – The name of the project on PyPI.

**Return type** str

**Returns** The shield.

**make_python_versions_shield**(*pypi_name*)
Create a shield to show the supported Python versions for the library.

**Parameters** **pypi_name** (str) – The name of the project on PyPI.

**Return type** str

**Returns** The shield.

**make_requires_shield**(*repo_name*, *username*)
Create a shield to show the dependency-dash requirements status.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **username** (str) – The username of the GitHub account that owns the repository.

**Return type** str

**Returns** The shield.

**make_rtfd_shield**(*repo_name*, *target='https://{}.readthedocs.io/en/latest/?badge=latest'*)
Create a shield for the ReadTheDocs documentation build status.

**Parameters**

- **repo_name** (str) – The name of the repository.
- **target** (str) – Default `'https://{}.readthedocs.io/en/latest/?badge=latest'`.

**Return type** str

**Returns** The shield.

**make_typing_shield**()
Create a shield to show that a library has **PEP 484** Type Hints / Annotations.

**Returns** The shield.

**Return type** str

**make_wheel_shield**(*pypi_name*)
Create a shield to show whether the library has a wheel on PyPI.

**Parameters** **pypi_name** (str) – The name of the project on PyPI.

**Return type** str

**Returns** The shield.

## 4.6 repo_helper.templates

Contains the pathlib.Path objects representing the templates directory (*template_dir*), and the directory representing the files used to initialise a new repository (*init_repo_template_dir*).

**template_dir**
**Type:** PosixPathPlus

The templates directory.

**init_repo_template_dir**
**Type:** PosixPathPlus

The directory representing the files used to initialise a new repository

## 4.7 repo_helper.testing

Helpers for running tests with pytest.

**Attention:** This module has the following additional requirements:

```
check-wheel-contents>=0.2.0
coincidence>=0.2.0
pytest>=6.2.0
```

These can be installed as follows:

```
$ python -m pip install repo-helper[testing]
```

New in version 2020.11.17.

**Functions:**

| | |
|---|---|
| *demo_environment*() | Pytest fixture to create a jinja2 environment for use in tests. |
| *example_config*() | Returns the contents of the example `repo_helper.yml` file. |
| *is_running_on_actions*() | Returns `True` if running on GitHub Actions. |
| *temp_empty_repo*(tmp_pathplus, monkeypatch) | Pytest fixture to return an empty git repository in a temporary location. |
| *temp_repo*(temp_empty_repo, example_config) | Pytest fixture to return a git repository in a temporary location. |

**fixture demo_environment**

**Scope:** function

Pytest fixture to create a jinja2 environment for use in tests.

The environment has the following variables available by default:

```
{
    "username": "octocat",
    "assignee": "octocat",
    "imgbot_ignore": [],
    "travis_ubuntu_version": "xenial",
    "github_ci_requirements": {"Linux": {"pre": [], "post": []}, "Windows": {"pre
↪": [], "post": []}},
    "travis_additional_requirements": [],
    "conda_channels": ["conda-forge"],
    "python_versions": ["3.6", "3.7"],
    "enable_tests": true,
    "enable_conda": true,
    "enable_docs": true,
    "enable_releases": true,
    "python_deploy_version": "3.6",
    "min_py_version": "3.6",
    "modname": "hello-world",
    "repo_name": "hello-world",
    "import_name": "hello_world",
    "platforms": ["Windows"],
```

(continues on next page)

(continued from previous page)

```
    "pypi_name": "hello-world",
    "py_modules": [],
    "extra_lint_paths": [],
    "extra_testenv_commands": [],
    "manifest_additional": [],
    "additional_requirements_files": [],
    "source_dir": "",
    "tests_dir": "tests",
    "additional_setup_args": {},
    "setup_pre": [],
    "docs_dir": "doc-source",
    "sphinx_html_theme": "alabaster",
    "additional_ignore": ["foo", "bar", "fuzz"],
    "join_path": "os.path.join",
    "pure_python": true,
    "stubs_package": false,
    "managed_message": "This file is managed by 'repo_helper'. Don't edit it␣
↪directly.",
    "short_desc": "a short description",
    "on_pypi": true,
    "docs_fail_on_warning": false,
    "requires_python": "3.6.1",
    "third_party_version_matrix": {},
    "use_whey": false,
    "use_flit": false
    "use_maturin": false
    "use_hatch": false
    "on_conda_forge": false
    "desktopfile": {}}
    }
```

plus `lint_warn_list` = `repo_helper.files.linting.lint_warn_list`.

Additional options can be set and values changed at the start of tests as follows:

```
def test(demo_environment):
    demo_environment.templates.globals["source_dir"] = "src"
```

**Return type** `Environment`

**fixture example_config**

**Scope:** session

Returns the contents of the example `repo_helper.yml` file.

**Return type** `str`

**is_running_on_actions**()

Returns `True` if running on GitHub Actions.

**Return type** `bool`

**fixture temp_empty_repo**

**Scope:** function

Pytest fixture to return an empty git repository in a temporary location.

*repo_helper.utils.today* is monkeypatched to return 25th July 2020.

**Return type** Repo

**fixture temp_repo**
**Scope:** function

Pytest fixture to return a git repository in a temporary location.

The repository will contain a `repo_helper.yml` yaml file, the contents of which can be seen at https://github.com/domdfcoding/repo_helper/blob/master/repo_helper/testing/repo_helper_example.yml.

*repo_helper.utils.today* is monkeypatched to return 25th July 2020.

**Return type** Repo

## 4.8 repo_helper.utils

General utilities.

**Classes:**

| | |
|---|---|
| *IniConfigurator*(base_path) | Base class to generate `.ini` configuration files. |

**Functions:**

| | |
|---|---|
| *commit_changes*(repo[, message]) | Commit staged changes. |
| *discover_entry_points*(group_name[, match_func]) | Returns a list of entry points in the given category, optionally filtered by `match_func`. |
| *get_license_text*(license_name, ...) | Obtain the license text for the given license. |
| *indent_join*(iterable) | Join an iterable of strings with newlines, and indent each line with a tab if there is more then one element. |
| *indent_with_tab*(text[, depth, predicate]) | Adds `'\t'` to the beginning of selected lines in 'text'. |
| *no_dev_versions*(versions) | Returns the subset of `versions` which does not end with `-dev`. |
| *normalize*(name) | Normalize the given name for PyPI et al. |
| *pformat_tabs*(obj[, width, depth, compact]) | Format a Python object into a pretty-printed representation. |
| *reformat_file*(filename, yapf_style, ...) | Reformat the given file. |
| *resource*(package, resource) | Retrieve the path to a resource inside a package. |
| *set_gh_actions_versions*(py_versions) | Convert development Python versions into the appropriate versions for GitHub Actions. |
| *sort_paths*(*paths) | Sort the `paths` by directory, then by file. |
| *stage_changes*(repo, files) | Stage any files that have been updated, added or removed. |

**Data:**

| | |
|---|---|
| *license_lookup* | Mapping of license short codes to license names used in trove classifiers. |

continues on next page

Table 18 – continued from previous page

| | |
|---|---|
| *today* | Under normal circumstances returns datetime.date.today(). |

**class IniConfigurator**(*base_path*)

Bases: object

Base class to generate .ini configuration files.

**Parameters** **base_path** (Path)

**Methods:**

| | |
|---|---|
| *copy_existing_value*(section, key) | Copy the existing value for key, if present, to the new configuration. |
| *merge_existing*(ini_file) | Merge existing sections in the configuration file into the new configuration. |
| *write_out*() | Write out to the .ini file. |

**copy_existing_value**(*section*, *key*)

Copy the existing value for key, if present, to the new configuration.

**Parameters**

- **section** (Section)
- **key** (str)

**merge_existing**(*ini_file*)

Merge existing sections in the configuration file into the new configuration.

**Parameters** **ini_file** (Path) – The existing .ini file.

**write_out**()

Write out to the .ini file.

**commit_changes**(*repo*, *message="Updated files with 'repo_helper'."*)

Commit staged changes.

**Parameters**

- **repo** (Union[str, Path, PathLike, Repo]) – The repository to commit in.
- **message** (str) – The commit message to use. Default "Updated files with 'repo_helper'.".

**Return type** str

**Returns** The SHA of the commit.

New in version 2020.11.23.

**discover_entry_points**(*group_name*, *match_func=None*)

Returns a list of entry points in the given category, optionally filtered by match_func.

New in version 1.1.0.

**Parameters**

- **group_name** (str) – The entry point group name, e.g. 'entry_points'.
- **match_func** (Optional[Callable[[Any], bool]]) – Function taking an object and returning True if the object is to be included in the output. Default None, which includes all objects.

**Return type** List[Any]

**Returns** List of matching objects.

**get_license_text**(*license_name*, *copyright_years*, *author*, *project_name*)
Obtain the license text for the given license.

New in version 2022.4.4.

**Parameters**

- **license_name** (str) – The name of the license.
- **copyright_years** (Union[str, int]) – The copyright years (e.g. '2019-2021') to display in the license. Not supported by all licenses.
- **author** (str) – The name of the author/copyright holder to display in the license. Not supported by all licenses.
- **project_name** (str) – The name of the project to display in the license. Not supported by all licenses.

Licenses are obtained from https://github.com/licenses/license-templates.

**Return type** str

**indent_join**(*iterable*)
Join an iterable of strings with newlines, and indent each line with a tab if there is more then one element.

**Parameters** **iterable** (Iterable[str])

**Return type** str

**indent_with_tab**(*text*, *depth=1*, *predicate=None*)
Adds '\t' to the beginning of selected lines in 'text'.

**Parameters**

- **text** (str) – The text to indent.
- **depth** (int) – The depth of the indentation. Default 1.
- **predicate** (Optional[Callable[[str], bool]]) – If given, '\t' will only be added to the lines where predicate(line) is :py:obj`True`. If predicate is not provided, it will default to adding '\t' to all non-empty lines that do not consist solely of whitespace characters. Default None.

**Return type** str

**license_lookup = {'AAL': 'Attribution Assurance License', 'AFL-1.1': 'Academic Free Licens**
**Type:** dict

Mapping of license short codes to license names used in trove classifiers.

**no_dev_versions**(*versions*)
Returns the subset of `versions` which does not end with `-dev`.

**Parameters** **versions** (`Iterable`[`str`])

**Return type** `List`[`str`]

**normalize**(*name*)
Normalize the given name for PyPI et al.

From **PEP 503** (public domain).

**Parameters** **name** (`str`) – The project name.

**Return type** `str`

**pformat_tabs**(*obj*, *width=80*, *depth=None*, *, *compact=False*)
Format a Python object into a pretty-printed representation.

Indentation is set at one tab.

**Parameters**

- **obj** (`object`) – The object to format.
- **width** (`int`) – The maximum width of the output. Default `80`.
- **depth** (`Optional`[`int`]) – Default `None`.
- **compact** (`bool`) – Default `False`.

**Return type** `str`

**reformat_file**(*filename*, *yapf_style*, *isort_config_file*)
Reformat the given file.

**Parameters**

- **filename** (`Union`[`str`, `Path`, `PathLike`])
- **yapf_style** (`str`) – The name of the yapf style, or the path to the yapf style file.
- **isort_config_file** (`str`) – The filename of the isort configuration file.

**Return type** `int`

**resource**(*package*, *resource*)
Retrieve the path to a resource inside a package.

New in version 2022.4.4.

**Parameters**

- **package** – The name of the package, or a module object representing it.
- **resource** – The name of the resource.

**set_gh_actions_versions**(*py_versions*)
Convert development Python versions into the appropriate versions for GitHub Actions.

**Parameters** **py_versions** (`Iterable`[`str`])

**Return type** `List`[`str`]

**sort_paths**(**paths*)
Sort the `paths` by directory, then by file.

New in version 2.6.0.

**Parameters** **paths**

**Return type** List[PathPlus]

**stage_changes**(*repo*, *files*)
Stage any files that have been updated, added or removed.

**Parameters**

- **repo** (Union[str, Path, PathLike, Repo]) – The repository.
- **files** (Iterable[Union[str, Path, PathLike]]) – List of files to stage.

**Return type** List[PathPlus]

**Returns** A list of staged files. Not all files in `files` will have been changed, and only changes are staged.

New in version 2020.11.23.

**today = datetime.date(2024, 4, 26)**
**Type:** date

Under normal circumstances returns `datetime.date.today()`.

## 4.9 repo_helper.files

The `files` subpackage contains most of the functions for managing files.

**Manager**
Type hint for a function that manages files.

Alias of Callable[[Path, Environment], List[str]]

**class Management**(**args*, ***kwargs*)
Bases: UserList[Tuple[Callable[[Path, Environment], List[str]], str, Sequence[str]]]

Class to store functions that manage files.

The syntax of each entry is:

- the function,
- a string to use in `exclude_files` to disable this function,
- a list of strings representing config values that must be true to call the function.

**Methods:**

| | |
|---|---|
| *register*(exclude_name[, ...]) | Decorator to register a function. |

**register**(*exclude_name*, *exclude_unless_true=()*, ***, *name=None*)
Decorator to register a function.

The function must have the following signature:

```
def function(
    repo_path: pathlib.Path,
    templates: jinja2.Environment,
    ) -> List[str]: ...
```

**Parameters**

- **exclude_name** (str) – A string to use in 'exclude_files' to disable this function.
- **exclude_unless_true** (Sequence[str]) – A list of strings representing config values that must be true to call the function. Default ().
- **name** (Optional[str]) – Optional name to use for the function in the output. Defaults to the name of the function.

**Return type** Callable

**Returns** The registered function.

**Raises** SyntaxError if the decorated function does not take the correct arguments.

**is_registered**(*obj*)

Return whether obj is a registered function.

**Parameters** **obj** (Any)

**Return type** bool

### 4.9.4 bots

Manage configuration files for bots.

**Functions:**

| | |
|---|---|
| *make_auto_assign_action*(repo_path, templates) | Add configuration for auto-assign to the desired repo. |
| *make_dependabot*(repo_path, templates) | Add configuration for dependabot to the desired repo. |
| *make_dependabotv2*(repo_path, templates) | Add configuration for dependabot to the desired repo. |
| *make_imgbot*(repo_path, templates) | Add configuration for imgbot to the desired repo. |
| *make_stale_bot*(repo_path, templates) | Add configuration for stale to the desired repo. |

**make_auto_assign_action**(*repo_path*, *templates*)

Add configuration for auto-assign to the desired repo.

https://github.com/kentaro-m/auto-assign

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_dependabot**(*repo_path*, *templates*)
Add configuration for `dependabot` to the desired repo.

`https://dependabot.com/`

**Parameters**

- **repo_path** (`Path`) – Path to the repository root.
- **templates** (`Environment`)

Deprecated since version 2020.12.11.

**Return type** `List[str]`

**make_dependabotv2**(*repo_path*, *templates*)
Add configuration for `dependabot` to the desired repo.

`https://dependabot.com/`

**Parameters**

- **repo_path** (`Path`) – Path to the repository root.
- **templates** (`Environment`)

New in version 2020.12.11.

**Return type** `List[str]`

**make_imgbot**(*repo_path*, *templates*)
Add configuration for `imgbot` to the desired repo.

`https://imgbot.net/`

**Parameters**

- **repo_path** (`Path`) – Path to the repository root.
- **templates** (`Environment`)

**Return type** `List[str]`

**make_stale_bot**(*repo_path*, *templates*)
Add configuration for `stale` to the desired repo.

`https://probot.github.io/apps/stale/`

**Parameters**

- **repo_path** (`Path`) – Path to the repository root.
- **templates** (`Environment`)

**Return type** `List[str]`

### 4.9.5 ci_cd

Manage configuration files for continuous integration / continuous deployment.

**Classes:**

| | |
|---|---|
| *ActionsManager*(repo_path, templates) | Responsible for creating, updating and removing GitHub Actions workflows. |

**Functions:**

| | |
|---|---|
| *ensure_bumpversion*(repo_path, templates) | Add configuration for `bumpversion` to the desired repo. |
| *get_bumpversion_filenames*(templates) | Returns an iterable of filenames to have the version number bumped in. |
| *make_actions_deploy_conda*(repo_path, templates) | Add script to build Conda package and deploy to Anaconda. |
| *make_actions_milestones*(repo_path, templates) | Add script to close milestones on release. |
| *make_conda_actions_ci*(repo_path, templates) | Add configuration for testing conda packages on *GitHub Actions* to the desired repo. |
| *make_github_ci*(repo_path, templates) | Add configuration for *GitHub Actions* to the desired repo. |
| *make_github_docs_test*(repo_path, templates) | Add configuration for GitHub Actions documentation check to the desired repo. |
| *make_github_flake8*(repo_path, templates) | Add configuration for the Flake8 GitHub Action. |
| *make_github_manylinux*(repo_path, templates) | Add configuration for *GitHub Actions* manylinux wheel builds the desired repo. |
| *make_github_mypy*(repo_path, templates) | Add configuration for the mypy GitHub Action. |
| *make_github_octocheese*(repo_path, templates) | Add configuration for the OctoCheese GitHub Action. |

**class ActionsManager**(*repo_path*, *templates*)

Bases: `object`

Responsible for creating, updating and removing GitHub Actions workflows.

**Parameters**

- **repo_path** (`Path`) – Path to the repository root.
- **templates** (`Environment`)

New in version 2020.12.18.

**Methods:**

| | |
|---|---|
| *get_gh_actions_matrix*() | Determines the matrix of Python versions used in GitHub Actions. |
| *get_gh_actions_python_versions*() | Prepares the mapping of Python versions to tox testenvs for use with GitHub Actions. |
| *get_linux_ci_requirements*() | Returns the Python requirements to run tests for on Linux. |
| *get_linux_ci_versions*() | Returns the Python versions to run tests for on Linux. |

continues on next page

Table 24 – continued from previous page

| | |
|---|---|
| *get_linux_mypy_requirements*() | Returns the Python requirements to run tests for on Linux. |
| *get_macos_ci_requirements*() | Returns the Python requirements to run tests for on macOS. |
| *get_macos_ci_versions*() | Returns the Python versions to run tests for on macOS. |
| *get_mypy_requirements*(platform) | Returns the Python requirements to run tests for on the given platform. |
| *get_windows_ci_requirements*() | Returns the Python requirements to run tests for on Windows. |
| *get_windows_ci_versions*() | Returns the Python versions to run tests for on Windows. |
| *make_flake8*() | Create, update or remove the flake8 action, as appropriate. |
| *make_linux*() | Create, update or remove the Linux action, as appropriate. |
| *make_macos*() | Create, update or remove the macOS action, as appropriate. |
| *make_mypy*() | Create, update or remove the mypy action, as appropriate. |
| *make_rustpython*() | Create, update or remove the RustPython action, as appropriate. |
| *make_windows*() | Create, update or remove the Windows action, as appropriate. |

**get_gh_actions_matrix**()
Determines the matrix of Python versions used in GitHub Actions.

New in version 2022.4.4.

**Return type** Dict[str, Tuple[str, bool]]

**get_gh_actions_python_versions**()
Prepares the mapping of Python versions to tox testenvs for use with GitHub Actions.

New in version 2020.12.21.

**Return type** Dict[str, str]

**get_linux_ci_requirements**()
Returns the Python requirements to run tests for on Linux.

**Return type** List[str]

**get_linux_ci_versions**()
Returns the Python versions to run tests for on Linux.

**Return type** List[str]

**get_linux_mypy_requirements**()
Returns the Python requirements to run tests for on Linux.

**Return type** List[str]

**get_macos_ci_requirements**()
Returns the Python requirements to run tests for on macOS.

**Return type** List[str]

**get_macos_ci_versions**()
Returns the Python versions to run tests for on macOS.

**Return type** List[str]

**get_mypy_requirements**(*platform*)
Returns the Python requirements to run tests for on the given platform.

**Return type** List[str]

**get_windows_ci_requirements**()
Returns the Python requirements to run tests for on Windows.

**Return type** List[str]

**get_windows_ci_versions**()
Returns the Python versions to run tests for on Windows.

**Return type** List[str]

**make_flake8**()
Create, update or remove the flake8 action, as appropriate.

New in version 2021.8.11.

**Return type** PathPlus

**make_linux**()
Create, update or remove the Linux action, as appropriate.

**Return type** PathPlus

**make_macos**()
Create, update or remove the macOS action, as appropriate.

**Return type** PathPlus

**make_mypy**()
Create, update or remove the mypy action, as appropriate.

New in version 2020.1.27.

**Return type** PathPlus

**make_rustpython**()
Create, update or remove the RustPython action, as appropriate.

**Return type** PathPlus

**make_windows**()
Create, update or remove the Windows action, as appropriate.

**Return type** PathPlus

**ensure_bumpversion**(*repo_path*, *templates*)
Add configuration for `bumpversion` to the desired repo.

https://pypi.org/project/bumpversion/

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**get_bumpversion_filenames**(*templates*)
Returns an iterable of filenames to have the version number bumped in.

New in version 2021.3.8.

**Parameters** **templates** (Environment)

**Return type** Iterable[str]

**make_actions_deploy_conda**(*repo_path*, *templates*)
Add script to build Conda package and deploy to Anaconda.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_actions_milestones**(*repo_path*, *templates*)
Add script to close milestones on release.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_conda_actions_ci**(*repo_path*, *templates*)
Add configuration for testing conda packages on *GitHub Actions* to the desired repo.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_github_ci**(*repo_path*, *templates*)
Add configuration for *GitHub Actions* to the desired repo.

**Parameters**

- **repo_path** (Path) – Path to the repository root.

- **templates** (Environment)

**Return type** List[str]

**make_github_docs_test**(*repo_path*, *templates*)
Add configuration for GitHub Actions documentation check to the desired repo.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_github_flake8**(*repo_path*, *templates*)
Add configuration for the Flake8 GitHub Action.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_github_manylinux**(*repo_path*, *templates*)
Add configuration for *GitHub Actions* manylinux wheel builds the desired repo.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_github_mypy**(*repo_path*, *templates*)
Add configuration for the mypy GitHub Action.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_github_octocheese**(*repo_path*, *templates*)
Add configuration for the OctoCheese GitHub Action.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

### 4.9.6 contributing

Contributing information for GitHub and documentation, plus GitHub issue templates.

**Functions:**

| | |
|---|---|
| *github_bash_block*(*commands) | Formats the given commands in a reStructuredText bash code block suitable for rendering on GitHub. |
| *make_contributing*(repo_path, templates) | Add `CONTRIBUTING.rst` to the desired repo. |
| *make_docs_contributing*(repo_path, templates) | Add CONTRIBUTING.rst to the documentation directory of the repo. |
| *make_issue_templates*(repo_path, templates) | Add issue templates for GitHub to the desired repo. |

**github_bash_block**(**commands*)
Formats the given commands in a reStructuredText bash code block suitable for rendering on GitHub.

**Parameters** **commands**

**Return type** str

**make_contributing**(*repo_path*, *templates*)
Add `CONTRIBUTING.rst` to the desired repo.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_docs_contributing**(*repo_path*, *templates*)
Add CONTRIBUTING.rst to the documentation directory of the repo.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_issue_templates**(*repo_path*, *templates*)
Add issue templates for GitHub to the desired repo.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

### 4.9.7 docs

Configuration for documentation with Sphinx and ReadTheDocs.

**Functions:**

| | |
|---|---|
| *copy_docs_styling*(repo_path, templates) | Copy custom styling for documentation to the desired repository. |
| *ensure_doc_requirements*(repo_path, templates) | Ensure <docs_dir>/requirements.txt contains the required entries. |
| *make_404_page*(repo_path, templates) | **type repo_path** Path |
| *make_alabaster_theming*() | Make the custom stylesheet for the alabaster Sphinx theme. |
| *make_conf*(repo_path, templates) | Add conf.py configuration file for Sphinx. |
| *make_docs_license_rst*(repo_path, templates) | Create the "License" page in the documentation. |
| *make_docs_source_rst*(repo_path, templates) | Create the "Source" page in the documentation, and add the associated image. |
| *make_docutils_conf*(repo_path, templates) | Add configuration for Docutils. |
| *make_readthedocs_theming*() | Make the custom stylesheet for the ReadTheDocs Sphinx theme. |
| *make_rtfd*(repo_path, templates) | Add configuration for ReadTheDocs. |
| *rewrite_docs_index*(repo_path, templates) | Update blocks in the documentation index.rst file. |

**copy_docs_styling**(*repo_path*, *templates*)

Copy custom styling for documentation to the desired repository.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**ensure_doc_requirements**(*repo_path*, *templates*)

Ensure <docs_dir>/requirements.txt contains the required entries.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_404_page**(*repo_path*, *templates*)

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_alabaster_theming**()
Make the custom stylesheet for the alabaster Sphinx theme.

**Return type** str

**Returns** The custom stylesheet.

**make_conf**(*repo_path*, *templates*)
Add `conf.py` configuration file for `Sphinx`.

https://www.sphinx-doc.org/en/master/index.html

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_docs_license_rst**(*repo_path*, *templates*)
Create the "License" page in the documentation.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_docs_source_rst**(*repo_path*, *templates*)
Create the "Source" page in the documentation, and add the associated image.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_docutils_conf**(*repo_path*, *templates*)
Add configuration for `Docutils`.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_readthedocs_theming**()
Make the custom stylesheet for the ReadTheDocs Sphinx theme.

**Return type** str

**Returns** The custom stylesheet.

**make_rtfd**(*repo_path*, *templates*)
Add configuration for `ReadTheDocs`.

https://readthedocs.org/

See https://docs.readthedocs.io/en/stable/config-file/v2.html for details

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**rewrite_docs_index**(*repo_path*, *templates*)
Update blocks in the documentation index.rst file.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

### 4.9.8 gitignore

Configuration for the .gitignore file.

**Functions:**

| | |
|---|---|
| *make_gitignore*(repo_path, templates) | Add .gitignore file to the given repository. |

**make_gitignore**(*repo_path*, *templates*)
Add .gitignore file to the given repository.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

### 4.9.9 linting

Configuration for various linting tools, such as Flake8 and Pylint.

**Functions:**

| | |
|---|---|
| *make_pylintrc*(repo_path, templates) | Copy .pylintrc into the desired repository. |

**make_pylintrc**(*repo_path*, *templates*)
Copy .pylintrc into the desired repository.

**Parameters**

- **repo_path** (Path) – Path to the repository root.

- **templates** (Environment)

**Return type** List[str]

### 4.9.10 packaging

Manage configuration files for packaging tools.

**Functions:**

| | |
|---|---|
| *make_manifest*(repo_path, templates) | Update the MANIFEST.in file for setuptools. |
| *make_pkginfo*(repo_path, templates) | Update the __pkginfo__.py file. |
| *make_pyproject*(repo_path, templates) | Create the pyproject.toml file for **PEP 517**. |
| *make_setup*(repo_path, templates) | Update the setup.py script. |
| *make_setup_cfg*(repo_path, templates) | Update the setup.py script. |

**make_manifest**(*repo_path*, *templates*)
Update the MANIFEST.in file for setuptools.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_pkginfo**(*repo_path*, *templates*)
Update the __pkginfo__.py file.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_pyproject**(*repo_path*, *templates*)
Create the pyproject.toml file for **PEP 517**.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_setup**(*repo_path*, *templates*)
Update the setup.py script.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_setup_cfg**(*repo_path*, *templates*)
Update the `setup.py` script.

**Parameters**

- **repo_path** (`Path`) – Path to the repository root.
- **templates** (`Environment`)

**Return type** `List[str]`

### 4.9.11 pre_commit

Configuration for pre-commit.

**Data:**

| | |
|---|---|
| *GITHUB_COM* | Instance of `apeye.url.URL` that points to the GitHub website. |

**Classes:**

| | |
|---|---|
| *Hook* | Represents a pre-commit hook. |
| *Repo*(repo, rev, hooks) | Represents a repository providing a pre-commit hooks. |

**Functions:**

| | |
|---|---|
| *make_github_url*(username, repository) | Construct a URL to a GitHub repository from a username and repository name. |
| *make_pre_commit*(repo_path, templates) | Add configuration for `pre-commit`. |

**GITHUB_COM = URL('https://github.com')**
**Type:** `URL`

Instance of `apeye.url.URL` that points to the GitHub website.

**typeddict Hook**
Bases: `TypedDict`

Represents a pre-commit hook.

**Required Keys**

- **id** (`str`) – Which hook from the repository to use.

**Optional Keys**

- **alias** (`str`) – Allows the hook to be referenced using an additional id when using pre-commit run <hookid>.
- **name** (`str`) – Override the name of the hook - shown during hook execution.
- **language_version** (`str`) – Override the language version for the hook. See `https://pre-commit.com/#overriding-language-version`
- **files** (`str`) – Override the default pattern for files to run on.
- **exclude** (`str`) – File exclude pattern.

- **types** (List[str]) – Override the default file types to run on. See https://pre-commit.com/#filtering-files-with-types.
- **exclude_types** (List[str]) – File types to exclude.
- **args** (List[str]) – List of additional parameters to pass to the hook.
- **stages** (List[Literal['commit', 'merge-commit', 'push', 'prepare-commit-msg', 'commit-msg', 'manual']]) – Confines the hook to the commit, merge-commit, push, prepare-commit-msg, commit-msg, post-checkout, or manual stage. See https://pre-commit.com/#confining-hooks-to-run-at-certain-stages.
- **additional_dependencies** (List[str]) – A list of dependencies that will be installed in the environment where this hook gets run. One useful application is to install plugins for hooks such as eslint.
- **always_run** (bool) – If True, this hook will run even if there are no matching files.
- **verbose** (bool) – If True, forces the output of the hook to be printed even when the hook passes.
- **log_file** (str) – If present, the hook output will additionally be written to a file.

**class Repo**(*repo*, *rev*, *hooks*)

Bases: object

Represents a repository providing a pre-commit hooks.

**Methods:**

| | |
|---|---|
| *__eq__*(other) | Method generated by attrs for class Repo. |
| *__ge__*(other) | Method generated by attrs for class Repo. |
| *__gt__*(other) | Method generated by attrs for class Repo. |
| *__le__*(other) | Method generated by attrs for class Repo. |
| *__lt__*(other) | Method generated by attrs for class Repo. |
| *__ne__*(other) | Method generated by attrs for class Repo. |
| *__repr__*() | Method generated by attrs for class Repo. |
| *to_dict*() | Returns a dictionary representation of the *Repo*. |

**Attributes:**

| | |
|---|---|
| *repo* | The repository url to git clone from. |
| *rev* | The revision or tag to clone at. |

**__eq__**(*other*)

Method generated by attrs for class Repo.

**__ge__**(*other*)

Method generated by attrs for class Repo.

**__gt__**(*other*)

Method generated by attrs for class Repo.

**__le__**(*other*)

Method generated by attrs for class Repo.

**__lt__**(*other*)
Method generated by attrs for class Repo.

**__ne__**(*other*)
Method generated by attrs for class Repo.

**__repr__**()
Method generated by attrs for class Repo.

**repo**
**Type:** URL

The repository url to git clone from.

**rev**
**Type:** str

The revision or tag to clone at.

**to_dict**()
Returns a dictionary representation of the *Repo*.

**Return type** MutableMapping[str, Union[str, List[*Hook*]]]

**make_github_url**(*username*, *repository*)
Construct a URL to a GitHub repository from a username and repository name.

**Parameters**

- **username** (str) – The username of the GitHub account that owns the repository.
- **repository** (str) – The name of the repository.

**Return type** URL

**make_pre_commit**(*repo_path*, *templates*)
Add configuration for pre-commit.

https://github.com/pre-commit/pre-commit

# See https://pre-commit.com for more information # See https://pre-commit.com/hooks.html for more hooks

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

### 4.9.12 readme

Functions to update README files.

**Functions:**

| | |
|---|---|
| *rewrite_readme*(repo_path, templates) | Update blocks in the `README.rst` file. |

**rewrite_readme**(*repo_path*, *templates*)
Update blocks in the `README.rst` file.

**Parameters**

- **repo_path** (`Path`) – Path to the repository root.
- **templates** (`Environment`)

**Return type** `List[str]`

### 4.9.13 testing

Configuration for testing and code formatting tools.

**Classes:**

| | |
|---|---|
| *ToxConfig*(repo_path, templates) | Generates the `tox.ini` configuration file. |

**Functions:**

| | |
|---|---|
| *ensure_tests_requirements*(repo_path, templates) | Ensure `tests/requirements.txt` contains the required entries. |
| *make_formate_toml*(repo_path, templates) | Add configuration for `formate`. |
| *make_isort*(repo_path, templates) | Remove the `isort` configuration file. |
| *make_justfile*(repo_path, templates) | Add configuration for `just`. |
| *make_tox*(repo_path, templates) | Add configuration for `Tox`. |
| *make_yapf*(repo_path, templates) | Add configuration for `yapf`. |

**class ToxConfig**(*repo_path*, *templates*)
Bases: *IniConfigurator*

Generates the `tox.ini` configuration file.

**Parameters**

- **repo_path** (`Path`) – Path to the repository root.
- **templates** (`Environment`)

**Methods:**

| | |
|---|---|
| *__getitem__*(item) | Passthrough to `templates.globals`. |
| *check_wheel_contents*() | `[check-wheel-contents]`. |
| *coverage_report*() | `[coverage:report]`. |

continues on next page

Table 38 – continued from previous page

| | |
|---|---|
| *coverage_run*() | [coverage:run]. |
| *envlists*() | [envlists]. |
| *flake8*() | [flake8]. |
| *get_mypy_commands*() | Compile the list of mypy commands. |
| *get_mypy_dependencies*() | Compile the list of mypy dependencies. |
| *get_setenv*([prefer_binary, setuptools_stdlib]) | Return environment variables to be set in the testenv. |
| *get_source_files*() | Compile the list of source files. |
| *get_third_party_version_matrix*() | Returns information about the matrix of third party versions. |
| *merge_existing*(ini_file) | Merge existing sections in the configuration file into the new configuration. |
| *pytest*() | [pytest]. |
| *testenv*() | [testenv]. |
| *testenv__package*() | [testenv:.package]. |
| *testenv_build*() | [testenv:build]. |
| *testenv_coverage*() | [testenv:coverage]. |
| *testenv_docs*() | [testenv:docs]. |
| *testenv_lint*() | [testenv:lint]. |
| *testenv_mypy*() | [testenv:mypy]. |
| *testenv_perflint*() | [testenv:perflint]. |
| *testenv_py312_dev*() | [testenv:py312-dev]. |
| *testenv_pyup*() | [testenv:pyup]. |
| *tox*() | [tox]. |

**__getitem__**(*item*)
Passthrough to `templates.globals`.

**Parameters** **item** (`str`)

**Return type** `Any`

**check_wheel_contents**()
[check-wheel-contents].

**coverage_report**()
[coverage:report].

**coverage_run**()
[coverage:run].

**envlists**()
[envlists].

**flake8**()
[flake8].

**get_mypy_commands**()
Compile the list of mypy commands.

**Return type** `List[str]`

**get_mypy_dependencies**()
Compile the list of mypy dependencies.

**Return type** List[str]

**get_setenv**(*prefer_binary=True*, *setuptools_stdlib=True*)
Return environment variables to be set in the testenv.

**Parameters**

- **prefer_binary** (bool) – Whether pip should be configured to prefer older binary packages over newer source packages. Default True.
- **setuptools_stdlib** – Whether setuptools should be configured to use the stdlib distutils. Default True.

**Return type** List[str]

**get_source_files**()
Compile the list of source files.

**Return type** List[str]

**get_third_party_version_matrix**()
Returns information about the matrix of third party versions.

The returned object is a three-element tuple, comprising:

- The name of the third party library.
- A list of version strings.
- The testenv suffix, e.g. `-attrs{19.3,20.1}`.

**Return type** Tuple[str, DelimitedList, str]

**merge_existing**(*ini_file*)
Merge existing sections in the configuration file into the new configuration.

**Parameters** **ini_file** (Path) – The existing `.ini` file.

**pytest**()
[pytest].

**testenv**()
[testenv].

**testenv__package**()
[testenv:.package].

**testenv_build**()
[testenv:build].

**testenv_coverage**()
[testenv:coverage].

**testenv_docs**()
[testenv:docs].

**testenv_lint**()
[testenv:lint].

**testenv_mypy**()
[testenv:mypy].

**testenv_perflint**()
[testenv:perflint].

**testenv_py312_dev**()
[testenv:py312-dev].

**testenv_pyup**()
[testenv:pyup].

**tox**()
[tox].

**ensure_tests_requirements**(*repo_path*, *templates*)
Ensure tests/requirements.txt contains the required entries.

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_formate_toml**(*repo_path*, *templates*)
Add configuration for formate.

https://formate.readthedocs.io

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_isort**(*repo_path*, *templates*)
Remove the isort configuration file.

https://github.com/timothycrosley/isort

**Parameters**

- **repo_path** (Path) – Path to the repository root.
- **templates** (Environment)

**Return type** List[str]

**make_justfile**(*repo_path*, *templates*)
Add configuration for `just`.

`https://github.com/casey/just`

**Parameters**

- **repo_path** (`Path`) – Path to the repository root.
- **templates** (`Environment`)

**Return type** `List[str]`

**make_tox**(*repo_path*, *templates*)
Add configuration for `Tox`.

`https://tox.readthedocs.io`

**Parameters**

- **repo_path** (`Path`) – Path to the repository root.
- **templates** (`Environment`)

**Return type** `List[str]`

**make_yapf**(*repo_path*, *templates*)
Add configuration for `yapf`.

`https://github.com/google/yapf`

**Parameters**

- **repo_path** (`Path`) – Path to the repository root.
- **templates** (`Environment`)

**Return type** `List[str]`

Chapter

FIVE

# Contributing

## 5.1 Overview

`repo_helper` uses tox to automate testing and packaging, and pre-commit to maintain code quality.

Install `pre-commit` with `pip` and install the git hook:

```
$ python -m pip install pre-commit
$ pre-commit install
```

## 5.2 Coding style

formate is used for code formatting.

It can be run manually via `pre-commit`:

```
$ pre-commit run formate -a
```

Or, to run the complete autoformatting suite:

```
$ pre-commit run -a
```

## 5.3 Automated tests

Tests are run with `tox` and `pytest`. To run tests for a specific Python version, such as Python 3.6:

```
$ tox -e py36
```

To run tests for all Python versions, simply run:

```
$ tox
```

## 5.4 Type Annotations

Type annotations are checked using `mypy`. Run `mypy` using `tox`:

```
$ tox -e mypy
```

## 5.5 Build documentation locally

The documentation is powered by Sphinx. A local copy of the documentation can be built with `tox`:

```
$ tox -e docs
```

## 5.6 Downloading source code

The `repo_helper` source code is available on GitHub, and can be accessed from the following URL: https://github.com/repo-helper/repo_helper

If you have `git` installed, you can clone the repository with the following command:

```
$ git clone https://github.com/repo-helper/repo_helper
```

```
Cloning into 'repo_helper'...
remote: Enumerating objects: 47, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 173 (delta 16), reused 17 (delta 6), pack-reused 126
Receiving objects: 100% (173/173), 126.56 KiB | 678.00 KiB/s, done.
Resolving deltas: 100% (66/66), done.
```

Alternatively, the code can be downloaded in a 'zip' file by clicking:
*Clone or download –> Download Zip*

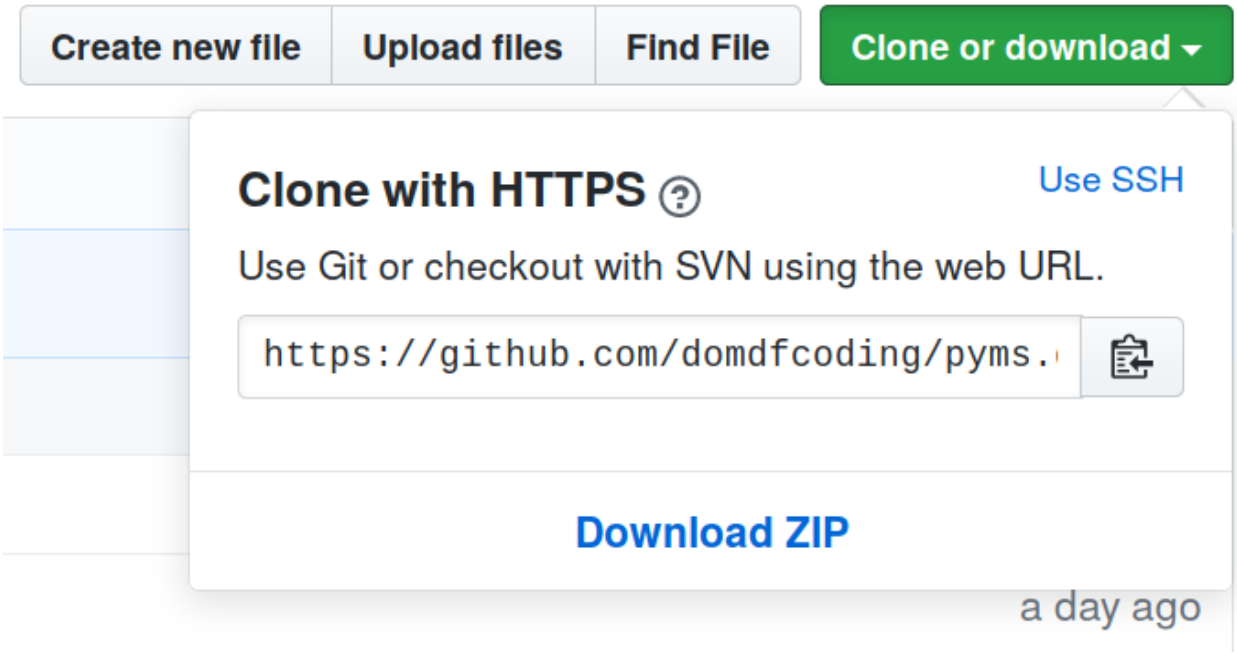


Fig. 1: Downloading a 'zip' file of the source code

### 5.6.1 Building from source

The recommended way to build `repo_helper` is to use tox:

```
$ tox -e build
```

The source and wheel distributions will be in the directory `dist`.

If you wish, you may also use pep517.build or another **PEP 517**-compatible build tool.

## 5.7 License

`repo_helper` is licensed under the GNU Lesser General Public License v3.0

---

Permissions of this copyleft license are conditioned on making available complete source code of licensed works and modifications under the same license or the GNU GPLv3. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights. However, a larger work using the licensed work through interfaces provided by the licensed work may be distributed under different terms and without source code for the larger work.

**Permissions**

- Commercial use – The licensed material and derivatives may be used for commercial purposes.
- Modification – The licensed material may be modified.
- Distribution – The licensed material may be distributed.
- Patent use – This license provides an express grant of patent rights from contributors.
- Private use – The licensed material may be used and modified in private.

**Conditions**

- License and copyright notice – A copy of the license and copyright notice must be included with the licensed material.
- Disclose source – Source code must be made available when the licensed material is distributed.
- State changes – Changes made to the licensed material must be documented.
- Same license (library) – Modifications must be released under the same license when distributing the licensed material. In some cases a similar or related license may be used, or this condition may not apply to works that use the licensed material as a library.

**Limitations**

- Liability – This license includes a limitation of liability.
- Warranty – This license explicitly states that it does NOT provide any warranty.

See more information on choosealicense.com ⇒

---

```
                   GNU LESSER GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.


  This version of the GNU Lesser General Public License incorporates
the terms and conditions of version 3 of the GNU General Public
License, supplemented by the additional permissions listed below.

  0. Additional Definitions.

  As used herein, "this License" refers to version 3 of the GNU Lesser
General Public License, and the "GNU GPL" refers to version 3 of the GNU
General Public License.

  "The Library" refers to a covered work governed by this License,
other than an Application or a Combined Work as defined below.

  An "Application" is any work that makes use of an interface provided
by the Library, but which is not otherwise based on the Library.
Defining a subclass of a class defined by the Library is deemed a mode
of using an interface provided by the Library.

  A "Combined Work" is a work produced by combining or linking an
Application with the Library.  The particular version of the Library
with which the Combined Work was made is also called the "Linked
Version".

  The "Minimal Corresponding Source" for a Combined Work means the
Corresponding Source for the Combined Work, excluding any source code
for portions of the Combined Work that, considered in isolation, are
based on the Application, and not on the Linked Version.

  The "Corresponding Application Code" for a Combined Work means the
object code and/or source code for the Application, including any data
and utility programs needed for reproducing the Combined Work from the
Application, but excluding the System Libraries of the Combined Work.

  1. Exception to Section 3 of the GNU GPL.

  You may convey a covered work under sections 3 and 4 of this License
without being bound by section 3 of the GNU GPL.

  2. Conveying Modified Versions.

  If you modify a copy of the Library, and, in your modifications, a
facility refers to a function or data to be supplied by an Application
that uses the facility (other than as an argument passed when the
facility is invoked), then you may convey a copy of the modified
version:

   a) under this License, provided that you make a good faith effort to
   ensure that, in the event an Application does not supply the
   function or data, the facility still operates, and performs
```

(continues on next page)

(continued from previous page)

```
   whatever part of its purpose remains meaningful, or

   b) under the GNU GPL, with none of the additional permissions of
   this License applicable to that copy.

  3. Object Code Incorporating Material from Library Header Files.

  The object code form of an Application may incorporate material from
a header file that is part of the Library.  You may convey such object
code under terms of your choice, provided that, if the incorporated
material is not limited to numerical parameters, data structure
layouts and accessors, or small macros, inline functions and templates
(ten or fewer lines in length), you do both of the following:

   a) Give prominent notice with each copy of the object code that the
   Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the object code with a copy of the GNU GPL and this license
   document.

  4. Combined Works.

  You may convey a Combined Work under terms of your choice that,
taken together, effectively do not restrict modification of the
portions of the Library contained in the Combined Work and reverse
engineering for debugging such modifications, if you also do each of
the following:

   a) Give prominent notice with each copy of the Combined Work that
   the Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the Combined Work with a copy of the GNU GPL and this license
   document.

   c) For a Combined Work that displays copyright notices during
   execution, include the copyright notice for the Library among
   these notices, as well as a reference directing the user to the
   copies of the GNU GPL and this license document.

   d) Do one of the following:

       0) Convey the Minimal Corresponding Source under the terms of this
       License, and the Corresponding Application Code in a form
       suitable for, and under terms that permit, the user to
       recombine or relink the Application with a modified version of
       the Linked Version to produce a modified Combined Work, in the
       manner specified by section 6 of the GNU GPL for conveying
       Corresponding Source.

       1) Use a suitable shared library mechanism for linking with the
       Library.  A suitable mechanism is one that (a) uses at run time
       a copy of the Library already present on the user's computer
       system, and (b) will operate properly with a modified version
       of the Library that is interface-compatible with the Linked
       Version.
```

(continues on next page)

(continued from previous page)

```
   e) Provide Installation Information, but only if you would otherwise
   be required to provide such information under section 6 of the
   GNU GPL, and only to the extent that such information is
   necessary to install and execute a modified version of the
   Combined Work produced by recombining or relinking the
   Application with a modified version of the Linked Version. (If
   you use option 4d0, the Installation Information must accompany
   the Minimal Corresponding Source and Corresponding Application
   Code. If you use option 4d1, you must provide the Installation
   Information in the manner specified by section 6 of the GNU GPL
   for conveying Corresponding Source.)

  5. Combined Libraries.

  You may place library facilities that are a work based on the
Library side by side in a single library together with other library
facilities that are not Applications and are not covered by this
License, and convey such a combined library under terms of your
choice, if you do both of the following:

   a) Accompany the combined library with a copy of the same work based
   on the Library, uncombined with any other library facilities,
   conveyed under the terms of this License.

   b) Give prominent notice with the combined library that part of it
   is a work based on the Library, and explaining where to find the
   accompanying uncombined form of the same work.

  6. Revised Versions of the GNU Lesser General Public License.

  The Free Software Foundation may publish revised and/or new versions
of the GNU Lesser General Public License from time to time. Such new
versions will be similar in spirit to the present version, but may
differ in detail to address new problems or concerns.

  Each version is given a distinguishing version number. If the
Library as you received it specifies that a certain numbered version
of the GNU Lesser General Public License "or any later version"
applies to it, you have the option of following the terms and
conditions either of that published version or of any later version
published by the Free Software Foundation. If the Library as you
received it does not specify a version number of the GNU Lesser
General Public License, you may choose any version of the GNU Lesser
General Public License ever published by the Free Software Foundation.

  If the Library as you received it specifies that a proxy can decide
whether future versions of the GNU Lesser General Public License shall
apply, that proxy's public statement of acceptance of any version is
permanent authorization for you to choose that version for the
Library.
```

# Python Module Index

## r

# Index

## Symbols

## A



## B



## C

## D

## K



## L



## M

## N



## O

## P



## R

## S



## T

## U



## V



## W



## Y