
repo-helper

Release 2021.6.13

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

Dominic Davis-Foster

Jun 23, 2021

Contents

1	Installation	3
1.1	from PyPI	3
1.2	from Anaconda	3
1.3	from GitHub	3
2	Configuration	5
2.1	Overview	5
2.2	Metadata	5
2.3	Optional features	9
2.4	Python versions	10
2.5	Packaging	12
2.6	Documentation	15
2.7	Testing	18
2.8	Travis	21
2.9	Conda & Anaconda	22
2.10	Other	23
3	Usage	27
3.1	repo-helper	27
3.2	repo-helper add	27
3.3	repo-helper broomstick	28
3.4	repo-helper build	29
3.5	repo-helper init	29
3.6	repo-helper make-recipe	30
3.7	repo-helper release	30
3.8	repo-helper show	31
3.9	repo-helper suggest	33
3.10	repo-helper wizard	33
4	Public API	35
4.1	repo_helper.blocks	35
4.2	repo_helper.build	39
4.3	repo_helper.conda	43
4.4	repo_helper.core	43
4.5	repo_helper.release	45
4.6	repo_helper.shields	47
4.7	repo_helper.templates	53
4.8	repo_helper.testing	54
4.9	repo_helper.utils	56
4.10	repo_helper.files	60
5	Contributing	79

5.1	Overview	79
5.2	Coding style	79
5.3	Automated tests	79
5.4	Type Annotations	79
5.5	Build documentation locally	80
5.6	Downloading source code	80
	Python Module Index	83
	Index	85

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.

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
```


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

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

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: 3.6

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.

Required: no

Default: The value of `default_python_versions`

Type: Sequence of String or Float

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 beginning of `setup.py`.

Example:

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

Required: no

Default: []

Type: Sequence of String

use_experimental_backend

Whether to use `repo-helper`'s experimental build backend, rather than `setuptools.build_meta`.

Required: no

Default: `False`

Type: Boolean

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

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:

```
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:

```
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 standalone page.

Example:

```
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:

```
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

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 \$VERSION: The default is now 0.812

Required: no

Default: 0.812

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:

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

Required: no

Default: []

Type: Sequence of String

min_coverage

The minimum permitted test coverage percentage.

Example:

```
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 \$VERSION.

Example:

```
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

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

pkginfo_extra

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.

```
pkginfo_extra:
- import datetime
- print(datetime.datetime.today())
```

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

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 \$VERSION.

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 build

Build a wheel for the given repository.

```
repo-helper build [OPTIONS] [REPOSITORY]
```

Options

- c, --conda**
Build a conda distribution.
- s, --source**
Build a source distribution.
- b, --binary**
Build a binary wheel.
- v, --verbose**
Enable verbose output.
- o, --out-dir <out_dir>**
The output directory.
- build-dir <build_dir>**
The temporary build directory.

Arguments

- REPOSITORY**
Optional argument. Default '.'

3.5 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.

3.6 repo-helper make-recipe

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

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

This command takes no options.

3.7 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.

3.8 repo-helper show

Show information about the repository.

3.8.1 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.

3.8.2 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.

3.8.3 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.

3.8.4 version

Show the repository version.

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

Options

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

3.9 repo-helper suggest

Suggest [trove classifiers](#) and keywords.

3.9.1 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.

3.10 repo-helper wizard

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

```
repo-helper wizard
```

This command takes no options.

4.1 `repo_helper.blocks`

Reusable blocks of `reStructuredText`.

Classes:

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

Functions:

<code>create_docs_install_block(repo_name, username)</code>	Create the installation instructions for insertion into the documentation.
<code>create_docs_links_block(username, repo_name)</code>	Create the documentation links block.
<code>create_readme_install_block(modname, username)</code>	Create the installation instructions for insertion into the README.
<code>create_short_desc_block(short_desc)</code>	Creates the short description block insertion into the README, documentation etc.
<code>get_docs_installation_block_template()</code>	Loads the <code>docs_installation_block</code> template from file and returns a <code>jinj2 jinja2.environment.Template</code> for it.
<code>get_docs_links_block_template()</code>	Loads the <code>docs_links_block</code> template from file and returns a <code>jinj2 jinja2.environment.Template</code> for it.
<code>get_readme_installation_block_no_pypi_template()</code>	Loads the <code>readme_installation_block_no_pypi</code> template from file and returns a <code>jinj2 jinja2.environment.Template</code> for it.
<code>get_readme_installation_block_template()</code>	Loads the <code>readme_installation_block</code> template from file and returns a <code>jinj2 jinja2.environment.Template</code> for it.

Data:

<code>installation_regex</code>	Regular expression to match the installation block placeholder.
<code>links_regex</code>	Regular expression to match the links block placeholder.
<code>shields_regex</code>	Regular expression to match the shields block placeholder.
<code>short_desc_regex</code>	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,
                    pre_commit=False, 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:

<code>make()</code>	Constructs the contents of the shields block.
<code>set_docs_mode()</code>	Create shields for insertion into Sphinx documentation.
<code>set_readme_mode()</code>	Create shields for insertion into <code>README.rst</code> .

Attributes:

<code>sections</code>	This list controls which sections are included, and their order.
<code>substitutions</code>	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 `jinj2.jinja2.environment.Template` for it.

Return type `Template`

get_docs_links_block_template ()

Loads the `docs_links_block` template from file and returns a `jinj2.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 `jinj2.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 `jinj2.jinja2.environment.Template` for it.

Return type `Template`

installation_regex

Type: `Pattern`

Regular expression to match the installation block placeholder.

Pattern	<code>(.. start installation) (.*) (.. end installation)</code>
Flags	<code>re.DOTALL</code>

links_regex

Type: `Pattern`

Regular expression to match the links block placeholder.

Pattern	<code>(.. start links) (.*)? (.. end links)</code>
Flags	<code>re.DOTALL</code>

shields_regex

Type: `Pattern`

Regular expression to match the shields block placeholder.

Pattern	<code>(.. start shields) (.*)? (.. end shields)</code>
Flags	<code>re.DOTALL</code>

short_desc_regex

Type: `Pattern`

Regular expression to match the short description block placeholder.

Pattern	<code>(.. start short [-_]desc) (.*)? (.. end short [-_]desc)</code>
Flags	<code>re.DOTALL</code>

4.2 repo_helper.build

PEP 517 build backend.

Classes:

<code>Builder(repo_dir[, build_dir, out_dir, ...])</code>	Builds source and binary distributions using metadata read from <code>repo_helper.yml</code> .
---	--

Functions:

<code>build_sdist(sdist_directory[, config_settings])</code>	PEP 517 hook to build a source distribution.
<code>build_wheel(wheel_directory[, ...])</code>	PEP 517 hook to build a wheel binary distribution.

class Builder (*repo_dir, build_dir=None, out_dir=None, *, verbose=False, colour=None*)

Bases: `WheelBuilder`

Builds source and binary distributions using metadata read from `repo_helper.yml`.

Parameters

- **repo_dir** (`Path`) – The repository to build the distribution for.
- **build_dir** (`Union[str, Path, PathLike, None]`) – The temporary build directory. Default `<repo_dir>/build/repo_helper_build`.
- **out_dir** (`Union[str, Path, PathLike, None]`) – The output directory. Default `<repo_dir>/dist`.
- **verbose** (`bool`) – Enable verbose output. Default `False`.

Attributes:

<code>archive_name</code>	The archive name, without the tag
<code>build_dir</code>	The (temporary) build directory.
<code>colour</code>	Whether to use coloured output.
<code>config</code>	repo_helper's configuration dictionary.
<code>import_name</code>	The directory containing the source files.
<code>info_dir</code>	The <code>info</code> directory in the build directory for Conda builds.
<code>out_dir</code>	The output directory.
<code>pkg_dir</code>	The path of the package directory.
<code>project_dir</code>	The repository
<code>repo_dir</code>	The repository
<code>verbose</code>	Whether to enable verbose output.

Methods:

<code>build_conda()</code>	Build the Conda distribution.
<code>build_sdist()</code>	Build the source distribution.
<code>copy_license(dest_dir)</code>	Copy the any files matching <code>LICEN[CS]E</code> .
<code>copy_manifest_additional()</code>	Copy additional files to the build directory, as specified in <code>manifest_additional</code> .
<code>create_conda_archive(wheel_contents_dir[, ...])</code>	Create the conda archive.
<code>create_sdist_archive()</code>	Create the sdist archive.
<code>iter_source_files()</code>	Iterate over the files in the source directory.
<code>write_conda_about()</code>	Write the conda <code>about.json</code> file.
<code>write_conda_index([build_number])</code>	Write the conda <code>index.json</code> file.
<code>write_entry_points()</code>	Write the list of entry points to the wheel, as specified in <code>console_scripts</code> .
<code>write_license(dest_dir)</code>	Copy the any files matching <code>LICEN[CS]E</code> .
<code>write_metadata(metadata_file)</code>	Write <code>Core Metadata</code> to the given file.
<code>write_wheel()</code>	Write the metadata to the <code>WHEEL</code> file.

archive_name

The archive name, without the tag

build_conda()

Build the Conda distribution.

Return type `str`

Returns The filename of the created archive.

build_dir

The (temporary) build directory.

build_sdist()

Build the source distribution.

Return type `str`

Returns The filename of the created archive.

colour

Whether to use coloured output.

config

repo_helper's configuration dictionary.

copy_license (*dest_dir*)

Copy the any files matching LICEN[CS]E.

Parameters **dest_dir** (*PathPlus*) – The directory to copy the files into.

copy_manifest_additional ()

Copy additional files to the build directory, as specified in *manifest_additional*.

create_conda_archive (*wheel_contents_dir*, *build_number=1*)

Create the conda archive.

Parameters

- **wheel_contents_dir** (*Union[str, Path, PathLike]*) – The directory containing the installed contents of the wheel.
- **build_number** (*int*) – Default 1.

Return type *str*

Returns The filename of the created archive.

create_sdist_archive ()

Create the sdist archive.

Return type *str*

Returns The filename of the created archive.

property import_name

The directory containing the source files.

Return type *str*

property info_dir

The *info* directory in the build directory for Conda builds.

Return type *PathPlus*

iter_source_files ()

Iterate over the files in the source directory.

Return type *Iterator[PathPlus]*

out_dir

The output directory.

property pkg_dir

The path of the package directory.

Return type *str*

project_dir

The repository

repo_dir

Type: *PathPlus*

The repository

verbose

Whether to enable verbose output.

write_conda_about ()

Write the conda about . json file.

See also:

<https://docs.conda.io/projects/conda-build/en/latest/resources/package-spec.html#info-about-json>

write_conda_index (*build_number=1*)

Write the conda index . json file.

See also:

<https://docs.conda.io/projects/conda-build/en/latest/resources/package-spec.html#info-index-json>

Parameters **build_number** (*int*) – Default 1.

write_entry_points ()

Write the list of entry points to the wheel, as specified in *console_scripts*.

write_license (*dest_dir*)

Copy the any files matching LICEN[CS]E.

Parameters **dest_dir** (*PathPlus*) – The directory to copy the files into.

write_metadata (*metadata_file*)

Write Core Metadata to the given file.

Parameters **metadata_file** (*PathPlus*)

write_wheel ()

Write the metadata to the WHEEL file.

build_sdist (*sdist_directory, config_settings=None*)

PEP 517 hook to build a source distribution.

See also:

<https://www.python.org/dev/peps/pep-0517/#build-sdist>

Parameters

- **sdist_directory**
- **config_settings** – Default *None*.

build_wheel (*wheel_directory, config_settings=None, metadata_directory=None*)

PEP 517 hook to build a wheel binary distribution.

See also:

<https://www.python.org/dev/peps/pep-0517/#build-wheel>

Parameters

- **wheel_directory**

- **config_settings** – Default `None`.
- **metadata_directory** – Default `None`.

4.3 `repo_helper.conda`

Utilities for Conda packages.

Functions:

<code>make_recipe(repo_dir, recipe_file)</code>	Make a Conda <code>meta.yaml</code> 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.

4.4 `repo_helper.core`

Core functionality of `repo_helper`.

Classes:

<code>RepoHelper(target_repo[, managed_message])</code>	Repo Helper: Manage configuration files with ease.
---	--

Functions:

<code>import_registered_functions()</code>	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** – 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:

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

Methods:

<code>load_settings([allow_unknown_keys])</code>	Load settings from the <code>repo_helper.yml</code> file in the repository.
<code>run()</code>	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]`

4.5 repo_helper.release

Functions for making tagged releases.

New in version 2020.12.29.

Classes:

<code>Bumper(repo_path[, force])</code>	Class to bump the repository version.
<code>BumpversionFileConfig</code>	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:

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

Attributes:

<code>bumpversion_file</code>	The path to the bumpversion configuration file.
<code>current_version</code>	
<code>repo</code>	

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.

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: *dict*

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

Required Keys

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

4.6 repo_helper.shields

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

Functions:

<i>make_actions_linux_shield</i> (repo_name, username)	Create a shield to indicate the status of the tests on Linux.
<i>make_actions_macos_shield</i> (repo_name, username)	Create a shield to indicate the status of the tests on macOS.
<i>make_actions_shield</i> (repo_name, username, ...)	Create a shield to indicate the status of the tests on Linux.
<i>make_actions_windows_shield</i> (repo_name, username)	Create a shield to indicate the status of the tests on Windows.
<i>make_activity_shield</i> (repo_name, username, ...)	Create a shield to show the number of commits to the GitHub repository since the last release.
<i>make_codefactor_shield</i> (repo_name, username)	Create a shield to show the Codefactor code quality grade.
<i>make_conda_platform_shield</i> (pypi_name, username)	Create a shield to show the supported Conda platforms.

continues on next page

Table 18 – continued from previous page

<code>make_conda_version_shield(pypi_name, username)</code>	Create a shield to show the version on Conda.
<code>make_coveralls_shield(repo_name, username)</code>	Create a shield to show the code coverage from Coveralls .
<code>make_docker_automated_build_shield(...)</code>	Create a shield to indicate the Docker automated build status.
<code>make_docker_build_status_shield(docker_name, ...)</code>	Create a shield to indicate the Docker image build status.
<code>make_docker_size_shield(docker_name, username)</code>	Create a shield to indicate the size of a docker image.
<code>make_docs_check_shield(repo_name, username)</code>	Create a shield for the GitHub Actions “Docs Check” status.
<code>make_language_shield(repo_name, username)</code>	Create a shield to show the primary language of the GitHub repository.
<code>make_last_commit_shield(repo_name, username)</code>	Create a shield to indicate when the last commit to the GitHub repository occurred.
<code>make_license_shield(repo_name, username)</code>	Create a shield to show the license of the GitHub repository.
<code>make_maintained_shield()</code>	Create a shield to indicate that the project is maintained.
<code>make_pre_commit_ci_shield(repo_name, username)</code>	Create a shield to show the pre-commit.ci status.
<code>make_pre_commit_shield()</code>	Create a shield to show that a repository is configured for use with pre-commit.
<code>make_pypi_downloads_shield(pypi_name)</code>	Create a shield to show the PyPI download statistics.
<code>make_pypi_version_shield(pypi_name)</code>	Create a shield to show the version on PyPI.
<code>make_python_implementations_shield(pypi_name)</code>	Create a shield to show the supported Python implementations for the library.
<code>make_python_versions_shield(pypi_name)</code>	Create a shield to show the supported Python versions for the library.
<code>make_requires_shield(repo_name, username)</code>	Create a shield to show the requires.io requirements status.
<code>make_rtf_d_shield(repo_name[, target])</code>	Create a shield for the ReadTheDocs documentation build status.
<code>make_typing_shield()</code>	Create a shield to show that a library has PEP 484 Type Hints / Annotations.
<code>make_wheel_shield(pypi_name)</code>	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_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 requires.io 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_rtfld_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](https://pep.python.org/pep-0484/) 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.

4.7 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

4.8 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
docutils==0.16
pytest>=6.0.0
twine>=3.2.0
```

These can be installed as follows:

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

New in version 2020.11.17.

Functions:

<code>builder_smoke_test(target_dir, username, ...)</code>	Tests that the given repository can be successfully built with repo-helper's experimental PEP 517 backend.
<code>demo_environment()</code>	Pytest fixture to create a jinja2 environment for use in tests.
<code>example_config()</code>	Returns the contents of the example <code>repo_helper.yml</code> file.
<code>is_running_on_actions()</code>	Returns <code>True</code> if running on GitHub Actions.
<code>temp_empty_repo(tmp_pathplus, monkeypatch)</code>	Pytest fixture to return an empty git repository in a temporary location.
<code>temp_repo(temp_empty_repo, example_config)</code>	Pytest fixture to return a git repository in a temporary location.

builder_smoke_test (*target_dir, username, repository, *, actions=False, conda=False*)

Tests that the given repository can be successfully built with repo-helper's experimental **PEP 517** backend.

Parameters

- **target_dir** (`Union[str, Path, PathLike]`)
- **username** (`str`)
- **repository** (`str`)
- **actions** (`bool`) – Whether to create GitHub Actions groups. Default `False`.
- **conda** (`bool`) – Whether to test building a conda package. Default `False`.

Return type `Tuple[int, float]`

Returns A tuple comprising: * A return code for the build process. * The build duration.

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": []}},
  "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"],
  "pypi_name": "hello-world",
  "py_modules": [],
  "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": {}
}

```

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

Additional options can be set and values changed at the start of with:

```

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`

4.9 repo_helper.utils

General utilities.

Classes:

<code>IniConfigurator(base_path)</code>	Base class to generate <code>.ini</code> configuration files.
---	---

Functions:

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

Data:

<code>license_lookup</code>	Mapping of license short codes to license names used in trove classifiers.
<code>today</code>	Under normal circumstances returns <code>datetime.date.today()</code> .

class IniConfigurator (*base_path*)

Bases: `object`

Base class to generate `.ini` configuration files.

Parameters `base_path` (`Path`)

Methods:

<code>copy_existing_value</code> (<i>section</i> , <i>key</i>)	Copy the existing value for <i>key</i> , if present, to the new configuration.
<code>merge_existing</code> (<i>ini_file</i>)	Merge existing sections in the configuration file into the new configuration.
<code>write_out</code> ()	Write out to the <code>.ini</code> 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.**indent_join** (*iterable*)

Join an iterable of strings with newlines, and indent each line with a tab if there is more than 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 License'}
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`

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* (*Union[str, Path, PathLike]*)

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(2021, 6, 23)

Type: `date`

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

4.10 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:

<code>register(exclude_name[, ...])</code>	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`

4.10.4 bots

Manage configuration files for bots.

Functions:

<code>make_artefact_cleaner(repo_path, templates)</code>	Add configuration for https://github.com/marketplace/actions/github-actions-artifact-cleaner to the desired repo.
<code>make_auto_assign_action(repo_path, templates)</code>	Add configuration for auto-assign to the desired repo.
<code>make_dependabot(repo_path, templates)</code>	Add configuration for dependabot to the desired repo.
<code>make_dependabotv2(repo_path, templates)</code>	Add configuration for dependabot to the desired repo.
<code>make_imgbot(repo_path, templates)</code>	Add configuration for imgbot to the desired repo.
<code>make_stale_bot(repo_path, templates)</code>	Add configuration for stale to the desired repo.

make_artefact_cleaner (*repo_path, templates*)

Add configuration for <https://github.com/marketplace/actions/github-actions-artifact-cleaner> to the desired repo.

Parameters

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

New in version 2020.11.23.

Return type `List[str]`

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]`

4.10.5 ci_cd

Manage configuration files for continuous integration / continuous deployment.

Classes:

<code>ActionsManager(repo_path, templates)</code>	Responsible for creating, updating and removing GitHub Actions workflows.
---	---

Functions:

<code>ensure_bumpversion(repo_path, templates)</code>	Add configuration for bumpversion to the desired repo.
<code>get_bumpversion_filenames(templates)</code>	Returns an iterable of filenames to have the version number bumped in.
<code>make_actions_deploy_conda(repo_path, templates)</code>	Add script to build Conda package and deploy to Anaconda.

continues on next page

Table 27 – continued from previous page

<code>make_conda_actions_ci(repo_path, templates)</code>	Add configuration for testing conda paclages on <i>GitHub Actions</i> to the desired repo.
<code>make_github_ci(repo_path, templates)</code>	Add configuration for <i>GitHub Actions</i> to the desired repo.
<code>make_github_docs_test(repo_path, templates)</code>	Add configuration for GitHub Actions documentation check to the desired repo.
<code>make_github_flake8(repo_path, templates)</code>	Add configuration for the Flake8 GitHub Action.
<code>make_github_manylinux(repo_path, templates)</code>	Add configuration for <i>GitHub Actions</i> manylinux wheel builds the desired repo.
<code>make_github_mypy(repo_path, templates)</code>	Add configuration for the mypy GitHub Action.
<code>make_github_octocheese(repo_path, templates)</code>	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:

<code>get_gh_actions_python_versions()</code>	Prepares the mapping of Python versions to tox testenvs for use with GitHub Actions.
<code>get_linux_ci_requirements()</code>	Returns the Python requirements to run tests for on Linux.
<code>get_linux_ci_versions()</code>	Returns the Python versions to run tests for on Linux.
<code>get_linux_mypy_requirements()</code>	Returns the Python requirements to run tests for on Linux.
<code>get_macos_ci_requirements()</code>	Returns the Python requirements to run tests for on macOS.
<code>get_macos_ci_versions()</code>	Returns the Python versions to run tests for on macOS.
<code>get_windows_ci_requirements()</code>	Returns the Python requirements to run tests for on Windows.
<code>get_windows_ci_versions()</code>	Returns the Python versions to run tests for on Windows.
<code>make_flake8()</code>	Create, update or remove the flake8 action, as appropriate.
<code>make_linux()</code>	Create, update or remove the Linux action, as appropriate.
<code>make_macos()</code>	Create, update or remove the macOS action, as appropriate.
<code>make_mypy()</code>	Create, update or remove the mypy action, as appropriate.
<code>make_rustpython()</code>	Create, update or remove the RustPython action, as appropriate.

continues on next page

Table 28 – continued from previous page

<code>make_windows()</code>	Create, update or remove the Windows action, as appropriate.
-----------------------------	--

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_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 \$VERSION.
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_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]`

4.10.6 contributing

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

Functions:

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

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]

4.10.7 docs

Configuration for documentation with Sphinx and ReadTheDocs.

Functions:

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

4.10.8 gitignore

Configuration for the `.gitignore` file.

Functions:

<code>make_gitignore(repo_path, templates)</code>	Add <code>.gitignore</code> 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]`

4.10.9 linting

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

Functions:

<code>make_pylintrc(repo_path, templates)</code>	Copy <code>.pylintrc</code> 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]`

4.10.10 packaging

Manage configuration files for packaging tools.

Functions:

<code>make_manifest(repo_path, templates)</code>	Update the <code>MANIFEST.in</code> file for <code>setuptools</code> .
<code>make_pkginfo(repo_path, templates)</code>	Update the <code>__pkginfo__.py</code> file.
<code>make_pyproject(repo_path, templates)</code>	Create the <code>pyproject.toml</code> file for PEP 517 .
<code>make_setup(repo_path, templates)</code>	Update the <code>setup.py</code> script.
<code>make_setup_cfg(repo_path, templates)</code>	Update the <code>setup.py</code> 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]`

4.10.11 pre_commit

Configuration for pre-commit.

Data:

<code>GITHUB_COM</code>	Instance of <code>apeye.url.URL</code> that points to the GitHub website.
-------------------------	---

Classes:

<code>Hook</code>	Represents a pre-commit hook.
<code>Repo(repo, rev, hooks)</code>	Represents a repository providing a pre-commit hooks.

Functions:

<code>make_github_url(username, repository)</code>	Construct a URL to a GitHub repository from a username and repository name.
<code>make_pre_commit(repo_path, templates)</code>	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: `dict`

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:

<code>__eq__</code> (<i>other</i>)	Method generated by attrs for class Repo.
<code>__ge__</code> (<i>other</i>)	Method generated by attrs for class Repo.
<code>__gt__</code> (<i>other</i>)	Method generated by attrs for class Repo.
<code>__le__</code> (<i>other</i>)	Method generated by attrs for class Repo.
<code>__lt__</code> (<i>other</i>)	Method generated by attrs for class Repo.
<code>__ne__</code> (<i>other</i>)	Method generated by attrs for class Repo.
<code>__repr__</code> ()	Method generated by attrs for class Repo.
<code>to_dict</code> ()	Returns a dictionary representation of the <i>Repo</i> .

Attributes:

<i>repo</i>	The repository url to git clone from.
<i>rev</i>	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]`

4.10.12 readme

Functions to update README files.

Functions:

<code>rewrite_readme(repo_path, templates)</code>	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]`

4.10.13 testing

Configuration for testing and code formatting tools.

Classes:

<code>ToxConfig(repo_path, templates)</code>	Generates the <code>tox.ini</code> configuration file.
--	--

Functions:

<code>ensure_tests_requirements(repo_path, templates)</code>	Ensure <code>tests/requirements.txt</code> contains the required entries.
<code>make_formate_toml(repo_path, templates)</code>	Add configuration for <code>formate</code> .
<code>make_isort(repo_path, templates)</code>	Remove the <code>isort</code> configuration file.
<code>make_tox(repo_path, templates)</code>	Add configuration for <code>Tox</code> .
<code>make_yapf(repo_path, templates)</code>	Add configuration for <code>yapf</code> .

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:

<code>__getitem__(item)</code>	Passthrough to <code>templates.globals</code> .
<code>check_wheel_contents()</code>	[<code>check-wheel-contents</code>].
<code>coverage_report()</code>	[<code>coverage:report</code>].
<code>coverage_run()</code>	[<code>coverage:run</code>].
<code>envlists()</code>	[<code>envlists</code>].
<code>flake8()</code>	[<code>flake8</code>].
<code>get_mypy_commands()</code>	Compile the list of <code>mypy</code> commands.
<code>get_mypy_dependencies()</code>	Compile the list of <code>mypy</code> dependencies.
<code>get_source_files()</code>	Compile the list of source files.
<code>get_third_party_version_matrix()</code>	Returns information about the matrix of third party versions.
<code>merge_existing(ini_file)</code>	Merge existing sections in the configuration file into the new configuration.
<code>pytest()</code>	[<code>pytest</code>].
<code>testenv()</code>	[<code>testenv</code>].
<code>testenv_build()</code>	[<code>testenv:build</code>].
<code>testenv_coverage()</code>	[<code>testenv:coverage</code>].
<code>testenv_docs()</code>	[<code>testenv:docs</code>].
<code>testenv_lint()</code>	[<code>testenv:lint</code>].
<code>testenv_mypy()</code>	[<code>testenv:mypy</code>].
<code>testenv_pyup()</code>	[<code>testenv:pyup</code>].
<code>tox()</code>	[<code>tox</code>].

__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_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* – The existing `.ini` file.

pytest ()
[`pytest`].


```

testenv()
    [testenv].

testenv_build()
    [testenv:build].

testenv_coverage()
    [testenv:coverage].

testenv_docs()
    [testenv:docs].

testenv_lint()
    [testenv:lint].

testenv_mypy()
    [testenv:mypy].

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_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]*

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*

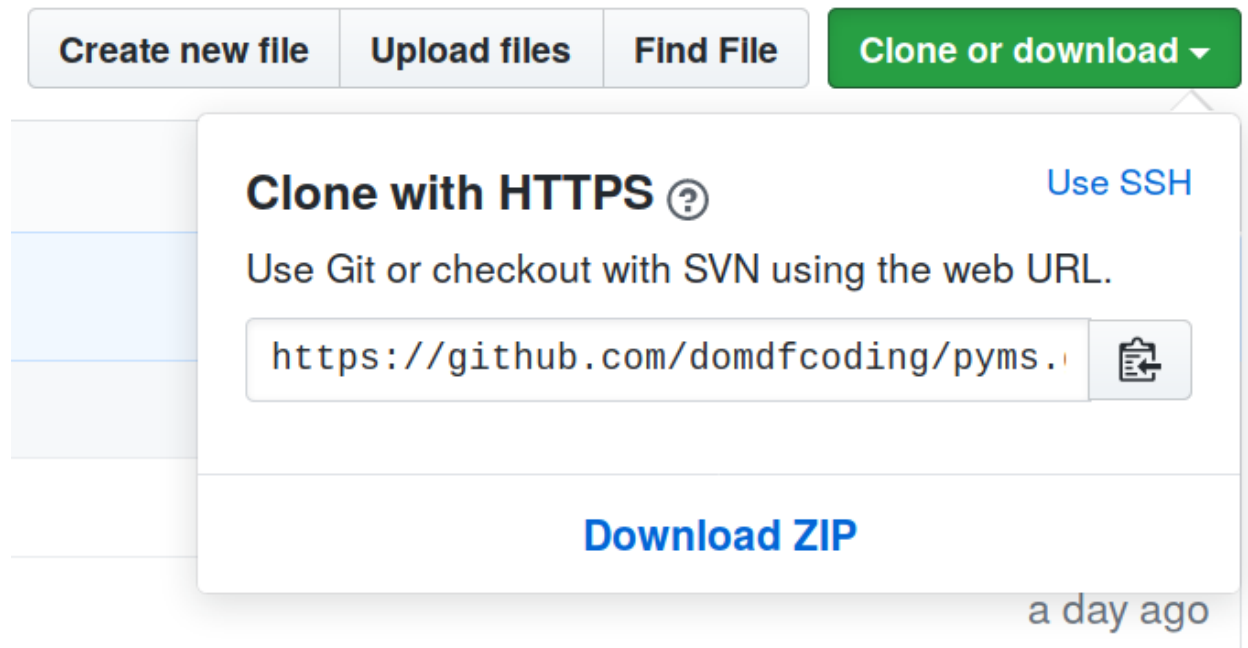


Fig. 1: Downloading a ‘zip’ file of the source code

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.

Python Module Index

r

- repo_helper.blocks, 35
- repo_helper.build, 39
- repo_helper.conda, 43
- repo_helper.core, 43
- repo_helper.files, 60
 - repo_helper.files.bots, 61
 - repo_helper.files.ci_cd, 62
 - repo_helper.files.contributing, 67
 - repo_helper.files.docs, 68
 - repo_helper.files.gitignore, 70
 - repo_helper.files.linting, 70
 - repo_helper.files.packaging, 71
 - repo_helper.files.pre_commit, 72
 - repo_helper.files.readme, 74
 - repo_helper.files.testing, 75
- repo_helper.release, 45
- repo_helper.shields, 47
- repo_helper.templates, 53
- repo_helper.testing, 54
- repo_helper.utils, 56

Symbols

__eq__ () (*Repo method*), 73
 __ge__ () (*Repo method*), 73
 __getitem__ () (*ToxConfig method*), 75
 __gt__ () (*Repo method*), 73
 __le__ () (*Repo method*), 73
 __lt__ () (*Repo method*), 73
 __ne__ () (*Repo method*), 73
 __repr__ () (*Repo method*), 73
 --add
 repo-helper-suggest-classifiers
 command line option, 33
 --binary
 repo-helper-build command line
 option, 29
 --build-dir <build_dir>
 repo-helper-build command line
 option, 29
 --colour
 repo-helper-show-changelog command
 line option, 32
 repo-helper-show-log command line
 option, 31
 --commit
 repo-helper command line option, 27
 repo-helper-init command line
 option, 30
 repo-helper-release command line
 option, 31
 --concise
 repo-helper-show-requirements
 command line option, 32
 --conda
 repo-helper-build command line
 option, 29
 --depth <depth>
 repo-helper-show-requirements
 command line option, 32
 --entries <entries>
 repo-helper-show-changelog command
 line option, 32
 repo-helper-show-log command line
 option, 31
 --file <file>
 repo-helper-add-requirement
 command line option, 28
 --force
 repo-helper command line option, 27
 repo-helper-init command line
 option, 30
 repo-helper-release command line
 option, 31
 --from-date <from_date>
 repo-helper-show-log command line
 option, 31
 --from-tag <from_tag>
 repo-helper-show-log command line
 option, 31
 --library
 repo-helper-suggest-classifiers
 command line option, 33
 --message <message>
 repo-helper command line option, 27
 repo-helper-init command line
 option, 30
 repo-helper-release command line
 option, 30
 --no-add
 repo-helper-suggest-classifiers
 command line option, 33
 --no-colour
 repo-helper-show-changelog command
 line option, 32
 repo-helper-show-log command line
 option, 31
 --no-commit
 repo-helper command line option, 27
 repo-helper-init command line
 option, 30
 repo-helper-release command line
 option, 31
 --no-pager
 repo-helper-show-changelog command
 line option, 32
 repo-helper-show-log command line
 option, 31
 repo-helper-show-requirements
 command line option, 32

```

--no-venv
  repo-helper-show-requirements
    command line option, 32
--not-library
  repo-helper-suggest-classifiers
    command line option, 33
--out-dir <out_dir>
  repo-helper-build command line
    option, 29
--quiet
  repo-helper-show-version command
    line option, 33
--reverse
  repo-helper-show-changelog command
    line option, 32
  repo-helper-show-log command line
    option, 31
--rm-tox
  repo-helper-broomstick command
    line option, 29
--source
  repo-helper-build command line
    option, 29
--status <status>
  repo-helper-suggest-classifiers
    command line option, 33
--verbose
  repo-helper-broomstick command
    line option, 29
  repo-helper-build command line
    option, 29
--version
  repo-helper command line option, 27
-b
  repo-helper-build command line
    option, 29
-c
  repo-helper-build command line
    option, 29
  repo-helper-show-requirements
    command line option, 32
-d
  repo-helper-show-requirements
    command line option, 32
-f
  repo-helper command line option, 27
  repo-helper-init command line
    option, 30
  repo-helper-release command line
    option, 31
-l
  repo-helper-suggest-classifiers
    command line option, 33
-m
  repo-helper command line option, 27
  repo-helper-init command line
    option, 30
  repo-helper-release command line
    option, 30
-n
  repo-helper command line option, 27
  repo-helper-init command line
    option, 30
  repo-helper-release command line
    option, 31
  repo-helper-show-changelog command
    line option, 32
  repo-helper-show-log command line
    option, 31
-o
  repo-helper-build command line
    option, 29
-q
  repo-helper-show-version command
    line option, 33
-r
  repo-helper-show-changelog command
    line option, 32
  repo-helper-show-log command line
    option, 31
-s
  repo-helper-build command line
    option, 29
  repo-helper-suggest-classifiers
    command line option, 33
-v
  repo-helper-broomstick command
    line option, 29
  repo-helper-build command line
    option, 29
-y
  repo-helper command line option, 27
  repo-helper-init command line
    option, 30
  repo-helper-release command line
    option, 31

```

A

```

ActionsManager (class in repo_helper.files.ci_cd),
  63
additional_ignore
  configuration value, 23
additional_requirements_files
  configuration value, 12
additional_setup_args
  configuration value, 12
archive_name (Builder attribute), 40
assignee

```

configuration value, 9
 author
 configuration value, 5

B

build_conda() (*Builder method*), 40
 build_dir (*Builder attribute*), 40
 build_sdist() (*Builder method*), 40
 build_sdist() (*in module repo_helper.build*), 42
 build_wheel() (*in module repo_helper.build*), 42
 Builder (*class in repo_helper.build*), 39
 builder_smoke_test() (*in module repo_helper.testing*), 54
 bump() (*Bumper method*), 46
 bump_version_for_file() (*Bumper method*), 46
 Bumper (*class in repo_helper.release*), 45
 bumpversion_file (*Bumper attribute*), 46

C

check_wheel_contents() (*ToxConfig method*), 76
 classifiers
 configuration value, 5
 colour (*Builder attribute*), 40
 commit_changes() (*in module repo_helper.utils*), 57
 conda_channels
 configuration value, 22
 conda_description
 configuration value, 23
 conda_extras
 configuration value, 22
 config (*Builder attribute*), 41
 configuration value
 additional_ignore, 23
 additional_requirements_files, 12
 additional_setup_args, 12
 assignee, 9
 author, 5
 classifiers, 5
 conda_channels, 22
 conda_description, 23
 conda_extras, 22
 console_scripts, 12
 copyright_years, 6
 desktopfile, 25
 docker_name, 9
 docker_shields, 9
 docs_dir, 15
 docs_fail_on_warning, 18
 docs_url, 18
 email, 6
 enable_conda, 23
 enable_devmode, 20
 enable_docs, 15
 enable_pre_commit, 10

enable_releases, 10
 enable_tests, 18
 entry_points, 14
 exclude_files, 23
 extra_sphinx_extensions, 15
 extras_require, 12
 github_ci_requirements, 21
 html_context, 15
 html_theme_options, 15
 imgbot_ignore, 24
 import_name, 6
 intersphinx_mapping, 16
 keywords, 6
 license, 7
 manifest_additional, 13
 min_coverage, 20
 modname, 7
 mypy_deps, 18
 mypy_plugins, 18
 mypy_version, 20
 on_pypi, 7
 pkginfo_extra, 24
 platforms, 13
 pre_commit_exclude, 25
 preserve_custom_theme, 16
 primary_conda_channel, 5
 pure_python, 7
 py_modules, 13
 pypi_name, 7
 python_deploy_version, 10
 python_versions, 10
 repo_name, 8
 requires_python, 11
 rtfld_author, 16
 setup_pre, 14
 short_desc, 8
 source_dir, 8
 sphinx_conf_epilogue, 16
 sphinx_conf_preamble, 17
 sphinx_html_theme, 17
 standalone_contrib_guide, 17
 stubs_package, 8
 tests_dir, 19
 third_party_version_matrix, 11
 tox_build_requirements, 19
 tox_requirements, 19
 tox_testenv_extras, 19
 tox_unmanaged, 20
 travis_additional_requirements, 21
 travis_extra_install_post, 21
 travis_extra_install_pre, 22
 travis_ubuntu_version, 22
 use_experimental_backend, 14
 use_whey, 14

- username, 9
- version, 9
- yapf_exclude, 25
- console_scripts
 - configuration value, 12
- copy_docs_styling() (in module *repo_helper.files.docs*), 68
- copy_existing_value() (*IniConfigurator method*), 57
- copy_license() (*Builder method*), 41
- copy_manifest_additional() (*Builder method*), 41
- copyright_years
 - configuration value, 6
- coverage_report() (*ToxConfig method*), 76
- coverage_run() (*ToxConfig method*), 76
- create_conda_archive() (*Builder method*), 41
- create_docs_install_block() (in module *repo_helper.blocks*), 37
- create_docs_links_block() (in module *repo_helper.blocks*), 37
- create_readme_install_block() (in module *repo_helper.blocks*), 37
- create_sdist_archive() (*Builder method*), 41
- create_short_desc_block() (in module *repo_helper.blocks*), 38
- current_version (*Bumper attribute*), 46

D

- desktopfile
 - configuration value, 25
- discover_entry_points() (in module *repo_helper.utils*), 57
- docker_name
 - configuration value, 9
- docker_shields
 - configuration value, 9
- docs_dir
 - configuration value, 15
- docs_fail_on_warning
 - configuration value, 18
- docs_url
 - configuration value, 18

E

- email
 - configuration value, 6
- enable_conda
 - configuration value, 23
- enable_devmode
 - configuration value, 20
- enable_docs
 - configuration value, 15
- enable_pre_commit

- configuration value, 10
- enable_releases
 - configuration value, 10
- enable_tests
 - configuration value, 18
- ensure_bumpversion() (in module *repo_helper.files.ci_cd*), 65
- ensure_doc_requirements() (in module *repo_helper.files.docs*), 68
- ensure_tests_requirements() (in module *repo_helper.files.testing*), 77
- entry_points
 - configuration value, 14
- envlists() (*ToxConfig method*), 76
- exclude_files
 - configuration value, 23
- exclude_files() (*RepoHelper property*), 44
- extra_sphinx_extensions
 - configuration value, 15
- extras_require
 - configuration value, 12

F

- files (*RepoHelper attribute*), 44
- flake8() (*ToxConfig method*), 76

G

- get_bumpversion_config() (*Bumper method*), 46
- get_bumpversion_filenames() (in module *repo_helper.files.ci_cd*), 65
- get_current_version() (*Bumper method*), 46
- get_docs_installation_block_template() (in module *repo_helper.blocks*), 38
- get_docs_links_block_template() (in module *repo_helper.blocks*), 38
- get_gh_actions_python_versions() (*ActionsManager method*), 64
- get_linux_ci_requirements() (*ActionsManager method*), 64
- get_linux_ci_versions() (*ActionsManager method*), 64
- get_linux_mypy_requirements() (*ActionsManager method*), 64
- get_macos_ci_requirements() (*ActionsManager method*), 64
- get_macos_ci_versions() (*ActionsManager method*), 64
- get_mypy_commands() (*ToxConfig method*), 76
- get_mypy_dependencies() (*ToxConfig method*), 76
- get_readme_installation_block_no_pypi_template() (in module *repo_helper.blocks*), 38

get_readme_installation_block_template() *links_regex (in module repo_helper.blocks), 38*
 (in module repo_helper.blocks), 38
 get_source_files() *(ToxConfig method), 76*
 get_third_party_version_matrix() *(ToxConfig method), 76*
 get_windows_ci_requirements() *(ActionsManager method), 64*
 get_windows_ci_versions() *(ActionsManager method), 64*
 github_bash_block() *(in module repo_helper.files.contributing), 67*
 github_ci_requirements configuration value, 21
 GITHUB_COM *(in module repo_helper.files.pre_commit), 72*

H

html_context configuration value, 15
 html_theme_options configuration value, 15

I

imgbot_ignore configuration value, 24
 import_name configuration value, 6
 import_name() *(Builder property), 41*
 import_registered_functions() *(in module repo_helper.core), 45*
 indent_join() *(in module repo_helper.utils), 58*
 indent_with_tab() *(in module repo_helper.utils), 58*
 info_dir() *(Builder property), 41*
 IniConfigurator *(class in repo_helper.utils), 57*
 init_repo_template_dir *(in module repo_helper.templates), 53*
 installation_regex *(in module repo_helper.blocks), 38*
 intersphinx_mapping configuration value, 16
 is_registered() *(in module repo_helper.files), 60*
 is_running_on_actions() *(in module repo_helper.testing), 55*
 iter_source_files() *(Builder method), 41*

K

keywords configuration value, 6

L

license configuration value, 7
 license_lookup *(in module repo_helper.utils), 58*

M

major() *(Bumper method), 46*
 make() *(ShieldsBlock method), 36*
 make_404_page() *(in module repo_helper.files.docs), 68*
 make_actions_deploy_conda() *(in module repo_helper.files.ci_cd), 65*
 make_actions_linux_shield() *(in module repo_helper.shields), 48*
 make_actions_macos_shield() *(in module repo_helper.shields), 48*
 make_actions_shield() *(in module repo_helper.shields), 49*
 make_actions_windows_shield() *(in module repo_helper.shields), 49*
 make_activity_shield() *(in module repo_helper.shields), 49*
 make_alabaster_theming() *(in module repo_helper.files.docs), 68*
 make_artefact_cleaner() *(in module repo_helper.files.bots), 61*
 make_auto_assign_action() *(in module repo_helper.files.bots), 61*
 make_codefactor_shield() *(in module repo_helper.shields), 49*
 make_conda_actions_ci() *(in module repo_helper.files.ci_cd), 65*
 make_conda_platform_shield() *(in module repo_helper.shields), 49*
 make_conda_version_shield() *(in module repo_helper.shields), 50*
 make_conf() *(in module repo_helper.files.docs), 69*
 make_contributing() *(in module repo_helper.files.contributing), 67*
 make_coveralls_shield() *(in module repo_helper.shields), 50*
 make_dependabot() *(in module repo_helper.files.bots), 61*
 make_dependabotv2() *(in module repo_helper.files.bots), 61*
 make_docker_automated_build_shield() *(in module repo_helper.shields), 50*
 make_docker_build_status_shield() *(in module repo_helper.shields), 50*
 make_docker_size_shield() *(in module repo_helper.shields), 50*
 make_docs_check_shield() *(in module repo_helper.shields), 51*
 make_docs_contributing() *(in module repo_helper.files.contributing), 67*

make_docs_source_rst() (in module *repo_helper.files.docs*), 69
make_docutils_conf() (in module *repo_helper.files.docs*), 69
make_flake8() (*ActionsManager* method), 64
make_format_toml() (in module *repo_helper.files.testing*), 77
make_github_ci() (in module *repo_helper.files.ci_cd*), 66
make_github_docs_test() (in module *repo_helper.files.ci_cd*), 66
make_github_flake8() (in module *repo_helper.files.ci_cd*), 66
make_github_manylinux() (in module *repo_helper.files.ci_cd*), 66
make_github_mypy() (in module *repo_helper.files.ci_cd*), 66
make_github_octocheese() (in module *repo_helper.files.ci_cd*), 66
make_github_url() (in module *repo_helper.files.pre_commit*), 74
make_gitignore() (in module *repo_helper.files.gitignore*), 70
make_imgbot() (in module *repo_helper.files.bots*), 62
make_isort() (in module *repo_helper.files.testing*), 77
make_issue_templates() (in module *repo_helper.files.contributing*), 67
make_language_shield() (in module *repo_helper.shields*), 51
make_last_commit_shield() (in module *repo_helper.shields*), 51
make_license_shield() (in module *repo_helper.shields*), 51
make_linux() (*ActionsManager* method), 64
make_macos() (*ActionsManager* method), 64
make_maintained_shield() (in module *repo_helper.shields*), 51
make_manifest() (in module *repo_helper.files.packaging*), 71
make_mypy() (*ActionsManager* method), 65
make_pkginfo() (in module *repo_helper.files.packaging*), 71
make_pre_commit() (in module *repo_helper.files.pre_commit*), 74
make_pre_commit_ci_shield() (in module *repo_helper.shields*), 52
make_pre_commit_shield() (in module *repo_helper.shields*), 52
make_pylintrc() (in module *repo_helper.files.linting*), 70
make_pypi_downloads_shield() (in module *repo_helper.shields*), 52
make_pypi_version_shield() (in module *repo_helper.shields*), 52
make_pyproject() (in module *repo_helper.files.packaging*), 71
make_python_implementations_shield() (in module *repo_helper.shields*), 52
make_python_versions_shield() (in module *repo_helper.shields*), 52
make_readthedocs_theming() (in module *repo_helper.files.docs*), 69
make_recipe() (in module *repo_helper.conda*), 43
make_requires_shield() (in module *repo_helper.shields*), 52
make_rtfd() (in module *repo_helper.files.docs*), 69
make_rtfd_shield() (in module *repo_helper.shields*), 53
make_rustpython() (*ActionsManager* method), 65
make_setup() (in module *repo_helper.files.packaging*), 71
make_setup_cfg() (in module *repo_helper.files.packaging*), 71
make_stale_bot() (in module *repo_helper.files.bots*), 62
make_tox() (in module *repo_helper.files.testing*), 77
make_typing_shield() (in module *repo_helper.shields*), 53
make_wheel_shield() (in module *repo_helper.shields*), 53
make_windows() (*ActionsManager* method), 65
make_yapf() (in module *repo_helper.files.testing*), 78
managed_message() (*RepoHelper* property), 44
Management (class in *repo_helper.files*), 60
Manager (in module *repo_helper.files*), 60
manifest_additional
 configuration value, 13
merge_existing() (*IniConfigurator* method), 57
merge_existing() (*ToxConfig* method), 76
min_coverage
 configuration value, 20
minor() (*Bumper* method), 46
modname
 configuration value, 7
module
 repo_helper.blocks, 35
 repo_helper.build, 39
 repo_helper.conda, 43
 repo_helper.core, 43
 repo_helper.files, 60
 repo_helper.files.bots, 61
 repo_helper.files.ci_cd, 62
 repo_helper.files.contributing, 67
 repo_helper.files.docs, 68
 repo_helper.files.gitignore, 70
 repo_helper.files.linting, 70
 repo_helper.files.packaging, 71

- repo_helper.files.pre_commit, 72
- repo_helper.files.readme, 74
- repo_helper.files.testing, 75
- repo_helper.release, 45
- repo_helper.shields, 47
- repo_helper.templates, 53
- repo_helper.testing, 54
- repo_helper.utils, 56
- mypy_deps
 - configuration value, 18
- mypy_plugins
 - configuration value, 18
- mypy_version
 - configuration value, 20

N

- no_dev_versions() (*in module repo_helper.utils*), 58
- normalize() (*in module repo_helper.utils*), 58

O

- on_pypi
 - configuration value, 7
- out_dir (*Builder attribute*), 41

P

- patch() (*Bumper method*), 47
- pformat_tabs() (*in module repo_helper.utils*), 58
- pkg_dir() (*Builder property*), 41
- pkginfo_extra
 - configuration value, 24
- platforms
 - configuration value, 13
- pre_commit_exclude
 - configuration value, 25
- preserve_custom_theme
 - configuration value, 16
- primary_conda_channel
 - configuration value, 5
- project_dir (*Builder attribute*), 41
- pure_python
 - configuration value, 7
- py_modules
 - configuration value, 13
- pypi_name
 - configuration value, 7
- pytest() (*ToxConfig method*), 76
- Python Enhancement Proposals
 - PEP 484, 48, 53
 - PEP 503, 58
 - PEP 517, 39, 42, 54, 71, 81
- python_deploy_version
 - configuration value, 10
- python_versions

- configuration value, 10

R

- reformat_file() (*in module repo_helper.utils*), 59
- register() (*Management method*), 60
- repo (*Bumper attribute*), 47
- Repo (*class in repo_helper.files.pre_commit*), 73
- repo (*Repo attribute*), 73
- repo_dir (*Builder attribute*), 41
- repo_helper.blocks
 - module, 35
- repo_helper.build
 - module, 39
- repo_helper.conda
 - module, 43
- repo_helper.core
 - module, 43
- repo_helper.files
 - module, 60
 - repo_helper.files.bots
 - module, 61
 - repo_helper.files.ci_cd
 - module, 62
 - repo_helper.files.contributing
 - module, 67
 - repo_helper.files.docs
 - module, 68
 - repo_helper.files.gitignore
 - module, 70
 - repo_helper.files.linting
 - module, 70
 - repo_helper.files.packaging
 - module, 71
 - repo_helper.files.pre_commit
 - module, 72
 - repo_helper.files.readme
 - module, 74
 - repo_helper.files.testing
 - module, 75
- repo_helper.release
 - module, 45
- repo_helper.shields
 - module, 47
- repo_helper.templates
 - module, 53
- repo_helper.testing
 - module, 54
- repo_helper.utils
 - module, 56
- repo_name
 - configuration value, 8
- repo_name() (*RepoHelper property*), 44
- repo-helper command line option
 - commit, 27

```

--force, 27
--message <message>, 27
--no-commit, 27
--version, 27
-f, 27
-m, 27
-n, 27
-y, 27
repo-helper-add-requirement command
  line option
  --file <file>, 28
  REQUIREMENT, 28
repo-helper-add-version command line
  option
  VERSION, 28
repo-helper-broomstick command line
  option
  --rm-tox, 29
  --verbose, 29
  -v, 29
repo-helper-build command line option
  --binary, 29
  --build-dir <build_dir>, 29
  --conda, 29
  --out-dir <out_dir>, 29
  --source, 29
  --verbose, 29
  -b, 29
  -c, 29
  -o, 29
  -s, 29
  -v, 29
  REPOSITORY, 29
repo-helper-init command line option
  --commit, 30
  --force, 30
  --message <message>, 30
  --no-commit, 30
  -f, 30
  -m, 30
  -n, 30
  -y, 30
repo-helper-release command line
  option
  --commit, 31
  --force, 31
  --message <message>, 30
  --no-commit, 31
  -f, 31
  -m, 30
  -n, 31
  -y, 31
repo-helper-show-changelog command
  line option
  --colour, 32
  --entries <entries>, 32
  --no-colour, 32
  --no-pager, 32
  --reverse, 32
  -n, 32
  -r, 32
repo-helper-show-log command line
  option
  --colour, 31
  --entries <entries>, 31
  --from-date <from_date>, 31
  --from-tag <from_tag>, 31
  --no-colour, 31
  --no-pager, 31
  --reverse, 31
  -n, 31
  -r, 31
repo-helper-show-requirements command
  line option
  --concise, 32
  --depth <depth>, 32
  --no-pager, 32
  --no-venv, 32
  -c, 32
  -d, 32
repo-helper-show-version command line
  option
  --quiet, 33
  -q, 33
repo-helper-suggest-classifiers
  command line option
  --add, 33
  --library, 33
  --no-add, 33
  --not-library, 33
  --status <status>, 33
  -l, 33
  -s, 33
RepoHelper (class in repo_helper.core), 43
REPOSITORY
  repo-helper-build command line
    option, 29
REQUIREMENT
  repo-helper-add-requirement
    command line option, 28
requires_python
  configuration value, 11
rev (Repo attribute), 74
rewrite_docs_index() (in module
  repo_helper.files.docs), 69
rewrite_readme() (in module
  repo_helper.files.readme), 74
rtfd_author

```


configuration value, 16
run() (*RepoHelper* method), 44

S

sections (*ShieldsBlock* attribute), 37
set_docs_mode() (*ShieldsBlock* method), 37
set_gh_actions_versions() (in module *repo_helper.utils*), 59
set_readme_mode() (*ShieldsBlock* method), 37
setup_pre
 configuration value, 14
shields_regex (in module *repo_helper.blocks*), 39
ShieldsBlock (class in *repo_helper.blocks*), 35
short_desc
 configuration value, 8
short_desc_regex (in module *repo_helper.blocks*), 39
sort_paths() (in module *repo_helper.utils*), 59
source_dir
 configuration value, 8
sphinx_conf_epilogue
 configuration value, 16
sphinx_conf_preamble
 configuration value, 17
sphinx_html_theme
 configuration value, 17
stage_changes() (in module *repo_helper.utils*), 59
standalone_contrib_guide
 configuration value, 17
stubs_package
 configuration value, 8
substitutions (*ShieldsBlock* attribute), 37

T

target_repo (*RepoHelper* attribute), 44
template_dir (in module *repo_helper.templates*), 53
templates (*RepoHelper* attribute), 45
testenv() (*ToxConfig* method), 76
testenv_build() (*ToxConfig* method), 77
testenv_coverage() (*ToxConfig* method), 77
testenv_docs() (*ToxConfig* method), 77
testenv_lint() (*ToxConfig* method), 77
testenv_mypy() (*ToxConfig* method), 77
testenv_pyup() (*ToxConfig* method), 77
tests_dir
 configuration value, 19
third_party_version_matrix
 configuration value, 11
to_dict() (*Repo* method), 74
today (in module *repo_helper.utils*), 59
today() (*Bumper* method), 47
tox() (*ToxConfig* method), 77
tox_build_requirements
 configuration value, 19

tox_requirements
 configuration value, 19
tox_testenv_extras
 configuration value, 19
tox_unmanaged
 configuration value, 20
ToxConfig (class in *repo_helper.files.testing*), 75
travis_additional_requirements
 configuration value, 21
travis_extra_install_post
 configuration value, 21
travis_extra_install_pre
 configuration value, 22
travis_ubuntu_version
 configuration value, 22

U

use_experimental_backend
 configuration value, 14
use_whey
 configuration value, 14
username
 configuration value, 9

V

verbose (*Builder* attribute), 42
VERSION
 repo-helper-add-version command
 line option, 28
version
 configuration value, 9

W

write_conda_about() (*Builder* method), 42
write_conda_index() (*Builder* method), 42
write_entry_points() (*Builder* method), 42
write_license() (*Builder* method), 42
write_metadata() (*Builder* method), 42
write_out() (*IniConfigurator* method), 57
write_wheel() (*Builder* method), 42

Y

yapf_exclude
 configuration value, 25