
xcookie Documentation

Release 0.2.1

Jon Crall

May 05, 2024

CONTENTS

1	xcookie package	3
1.1	Subpackages	3
1.1.1	xcookie.builders package	3
1.1.1.1	Submodules	3
1.1.1.1.1	xcookie.builders._builder module	3
1.1.1.1.2	xcookie.builders.common_ci module	3
1.1.1.1.3	xcookie.builders.docs module	4
1.1.1.1.4	xcookie.builders.docs_conf module	4
1.1.1.1.5	xcookie.builders.github_actions module	4
1.1.1.1.6	xcookie.builders.gitlab_ci module	6
1.1.1.1.7	xcookie.builders.pyproject module	6
1.1.1.1.8	xcookie.builders.readme module	7
1.1.1.1.9	xcookie.builders.readthedocs module	7
1.1.1.1.10	xcookie.builders.setup module	7
1.1.1.2	Module contents	7
1.1.2	xcookie.rc package	7
1.1.2.1	Submodules	7
1.1.2.1.1	xcookie.rc.conf_ext module	7
1.1.2.2	Module contents	9
1.2	Submodules	10
1.2.1	xcookie.__main__ module	10
1.2.2	xcookie.constants module	10
1.2.3	xcookie.directive module	10
1.2.4	xcookie.main module	12
1.2.5	xcookie.rich_ext module	18
1.2.6	xcookie.util_yaml module	19
1.2.7	xcookie.vcs_remotes module	22
1.3	Module contents	23
2	Indices and tables	25
	Bibliography	27
	Python Module Index	29
	Index	31

The xcookie module is a CLI tool for initializing and maintaining standardized repo infrastructure. In other words it is a Python project cookie cutter that attempts to help keep existing modules up to date with latest infrastructure developments.

Currently this module is specialized towards Erotemic / PyUtils / Kitware projects but the goal is to eventually generalize everything.

Read the docs	https://xcookie.readthedocs.io
Github	https://github.com/Erotemic/xcookie
Pypi	https://pypi.org/project/xcookie

XCOOKIE PACKAGE

1.1 Subpackages

1.1.1 xcookie.builders package

1.1.1.1 Submodules

1.1.1.1.1 xcookie.builders._builder module

class xcookie.builders._builder.Builder

Bases: `object`

1.1.1.1.2 xcookie.builders.common_ci module

Common subroutines for consistency between gitlab-ci / github actions / etc...

xcookie.builders.common_ci.make_mypy_check_parts(*self*)

xcookie.builders.common_ci.make_build_wheel_parts(*self*, wheelhouse_dpath='wheelhouse')

xcookie.builders.common_ci.make_install_and_test_wheel_parts(*self*, wheelhouse_dpath,
special_install_lines,
workspace_dname,
custom_before_test_lines=[],
custom_after_test_commands=[])

Builds the YAML common between github actions and gitlab CI to install and tests python packages.

References

<https://stackoverflow.com/questions/42019184/python-how-can-i-get-the-version-number-from-a-whl-file>

1.1.1.1.3 xcookie.builders.docs module

class xcookie.builders.docs.**DocsBuilder**(*config*)

Bases: `object`

Helper to build sphinx docs related files

Parameters

config (*XCookieConfig.shrinkuser()*)

property docs_dpath

property docs_auto_outdir

sphinx_apidoc_invocation(*shrinkuser=False*)

Instructions to invoke sphinx-apidoc. Xcookie calls this, but also writes it to the conf.py docstring for transparency.

xcookie.builders.docs.**build_docs_index**(*self*)

xcookie.builders.docs.**build_docs_conf**(*self*)

xcookie.builders.docs.**build_docs_requirements**(*self*)

1.1.1.1.4 xcookie.builders.docs_conf module

1.1.1.1.5 xcookie.builders.github_actions module

class xcookie.builders.github_actions.**Actions**

Bases: `object`

Help build Github Action JSON objects

Example

```
from xcookie.builders.github_actions import Actions
import types
for attr_name in dir(Actions):
```

```
    if not attr_name.startswith('_'):
        attr = getattr(Actions, attr_name)
        if isinstance(attr, types.MethodType):
            print(attr_name)
            action = attr()
```

```
...
```

```
action_versions = {'checkout': 'actions/checkout@v3', 'setup-python':
'actions/setup-python@v4'}
```

```
classmethod _available_action_methods()
```

```
classmethod _check_for_updates()
```

```
classmethod action(*args, **kwargs)
```

The generic action

```
classmethod checkout(*args, **kwargs)
```

```
classmethod setup_python(*args, **kwargs)
```

```
classmethod codecov_action(*args, **kwargs)
```


References

<https://github.com/codecov/codecov-action>

classmethod `combine_coverage(*args, **kwargs)`

classmethod `upload_artifact(*args, **kwargs)`

classmethod `download_artifact(*args, **kwargs)`

classmethod `msvc_dev_cmd(*args, osvar=None, bits=None, test_condition=None, **kwargs)`

classmethod `setup_qemu(*args, sensible=False, **kwargs)`

classmethod `setup_xcode(*args, sensible=False, **kwargs)`

classmethod `setup_ipfs(*args, **kwargs)`

classmethod `cibuildwheel(*args, sensible=False, **kwargs)`

`xcookie.builders.github_actions.build_github_actions(self)`

```
cat ~/code/xcookie/xcookie/rc/tests.yml.in | yq .jobs.lint cat ~/code/xcookie/xcookie/rc/tests.yml.in
| yq .jobs.build_and_test_sdist cat ~/code/xcookie/xcookie/rc/tests.yml.in | yq .jobs.deploy cat
~/code/xcookie/xcookie/rc/tests.yml.in | yq .
```

Example

```
>>> from xcookie.builders.github_actions import * # NOQA
>>> from xcookie.main import XCookieConfig
>>> from xcookie.main import TemplateApplier
>>> config = XCookieConfig(tags=['purepy'])
>>> self = TemplateApplier(config)
>>> text = build_github_actions(self)
>>> print(text)
```

`xcookie.builders.github_actions.lint_job(self)`

`xcookie.builders.github_actions.build_and_test_sdist_job(self)`

`xcookie.builders.github_actions.build_binpy_wheels_job(self)`

```
cat ~/code/xcookie/xcookie/rc/test_binaries.yml.in | yq .jobs.build_and_test_wheels
```

Notes

Supported Action platforms:

<https://raw.githubusercontent.com/actions/python-versions/main/versions-manifest.json>

`xcookie.builders.github_actions.get_supported_platform_info(self)`

`xcookie.builders.github_actions.build_purewheel_job(self)`

`xcookie.builders.github_actions.test_wheels_job(self, needs=None)`

`xcookie.builders.github_actions.build_deploy(self, mode='live', needs=None)`

Example

```
>>> from xcookie.builders.github_actions import * # NOQA
>>> from xcookie.main import XCookieConfig
>>> from xcookie.main import TemplateApplier
>>> config = XCookieConfig(tags=['purepy'], remote_group='Org', repo_name='Repo')
>>> self = TemplateApplier(config)
>>> self._presetup()
>>> text = Yaml.dumps(build_deploy(self))
>>> print(text)
```

`xcookie.builders.github_actions.build_github_release(self, needs=None)`

References

<https://github.com/marketplace/actions/create-a-release-in-a-github-action> <https://github.com/softprops/action-gh-release> <https://github.com/softprops/action-gh-release/issues/20#issuecomment-572245945>

1.1.1.1.6 xcookie.builders.gitlab_ci module

`xcookie.builders.gitlab_ci.build_gitlab_ci(self)`

Example

```
>>> from xcookie.builders.gitlab_ci import * # NOQA
>>> from xcookie.main import XCookieConfig
>>> from xcookie.main import TemplateApplier
>>> config = XCookieConfig(tags=['purepy'])
>>> self = TemplateApplier(config)
>>> text = build_gitlab_ci(self)
>>> print(text)
```

`xcookie.builders.gitlab_ci.build_gitlab_rules(self)`

`xcookie.builders.gitlab_ci.make_purepy_ci_jobs(self)`

1.1.1.1.7 xcookie.builders.pyproject module

`xcookie.builders.pyproject.build_pyproject(self)`

Returns

templated code

Return type

str

1.1.1.1.8 xcookie.builders.readme module

class xcookie.builders.readme.BadgeBuilder

Bases: `object`

build_templates(*group, repo_name, main_branch, workflow*)

class xcookie.builders.readme.ReadmeBuilder

Bases: `object`

xcookie.builders.readme.**build_readme**(*self*)

xcookie.builders.readme.**_ibeis_badges**(*repo_names*)

```
repo_names = [  
    'ibeis', 'utool', 'vtool_ibeis', 'plottool_ibeis', 'guitool_ibeis', 'pyhesaff', 'pyflann_ibeis', 'vtool_ibeis_ext',  
    'graphid',  
]
```

1.1.1.1.9 xcookie.builders.readthedocs module

Build the readthedocs yaml file.

xcookie.builders.readthedocs.**build_readthedocs**(*self*)

1.1.1.1.10 xcookie.builders.setup module

xcookie.builders.setup.**build_setup**(*self*)

1.1.1.2 Module contents

1.1.2 xcookie.rc package

1.1.2.1 Submodules

1.1.2.1.1 xcookie.rc.conf_ext module

class xcookie.rc.conf_ext.PatchedPythonDomain(*env: BuildEnvironment*)

Bases: `PythonDomain`

References

<https://github.com/sphinx-doc/sphinx/issues/3866>

resolve_xref(*env, fromdocname, builder, typ, target, node, contnode*)

Helps to resolves cross-references

class xcookie.rc.conf_ext.GoogleStyleDocstringProcessor(*autobuild=1*)

Bases: `object`

A small extension that runs after napoleon and reformats erotemic-flavored google-style docstrings for sphinx.

register_section(tag, alias=None)

Decorator that adds a custom processing function for a non-standard google style tag. The decorated function should accept a list of docstring lines, where the first one will be the google-style tag that likely needs to be replaced, and then return the appropriate sphinx format (TODO what is the name? Is it just RST?).

_register_builtins()

Adds definitions I like of CommandLine, TextArt, and Ignore

process(lines)

Example

```
>>> import ubelt as ub
>>> self = GoogleStyleDocstringProcessor()
>>> lines = ['Hello world',
>>>          '',
>>>          'CommandLine:',
>>>          '    hi',
>>>          '',
>>>          'CommandLine:',
>>>          '',
>>>          '    bye',
>>>          '',
>>>          'TextArt:',
>>>          '',
>>>          '    1',
>>>          '    2',
>>>          '',
>>>          '    345',
>>>          '',
>>>          'Foobar:',
>>>          '',
>>>          'TextArt:']
>>> new_lines = self.process(lines[:])
>>> print(chr(10).join(new_lines))
```

process_docstring_callback(app, what_: str, name: str, obj: Any, options: Any, lines: List[str]) → None

Callback to be registered to autodoc-process-docstring

Custom process to transform docstring lines Remove “Ignore” blocks

Parameters

- **app** (*sphinx.application.Sphinx*) – the Sphinx application object
- **what** (*str*) – the type of the object which the docstring belongs to (one of “module”, “class”, “exception”, “function”, “method”, “attribute”)
- **name** (*str*) – the fully qualified name of the object
- **obj** – the object itself
- **options** – the options given to the directive: an object with attributes `inherited_members`, `undoc_members`, `show_inheritance` and `noindex` that are true if the flag option of same name was given to the auto directive

- **lines** (*List[str]*) – the lines of the docstring, see above

References

https://www.sphinx-doc.org/en/1.5.1/_modules/sphinx/ext/autodoc.html <https://www.sphinx-doc.org/en/master/usage/extensions/autodoc.html>

class xcookie.rc.conf_ext.**SphinxDocstring**(*lines*)

Bases: `object`

Helper to parse and modify sphinx docstrings

find_tagged_lines(*tag*)

xcookie.rc.conf_ext.**paragraph**(*text*)

Wraps multi-line strings and restructures the text to remove all newlines, heading, trailing, and double spaces.

Useful for writing log messages

Parameters

text (*str*) – typically a multiline string

Returns

the reduced text block

Return type

`str`

xcookie.rc.conf_ext.**create_doctest_figure**(*app, obj, name, lines*)

The idea is that each doctest that produces a figure should generate that and then that figure should be part of the docs.

xcookie.rc.conf_ext.**postprocess_hyperlinks**(*app, doctree, docname*)

Extension to fixup hyperlinks. This should be connected to the Sphinx application’s “autodoc-process-docstring” event.

xcookie.rc.conf_ext.**fix_rst_todo_section**(*lines*)

xcookie.rc.conf_ext.**setup**(*app*)

1.1.2.2 Module contents

xcookie.rc.**resource_fpath**(*fname*)

Example

```
>>> from xcookie.rc import * # NOQA
>>> fname = 'setup.py.in'
>>> fpath = resource_fpath(fname)
>>> assert fpath.name == fname
>>> assert fpath.exists()
```

1.2 Submodules

1.2.1 xcookie.__main__ module

1.2.2 xcookie.constants module

1.2.3 xcookie.directive module

Port and extension of xdoctest directives.

`xcookie.directive.named(key, pattern)`

helper for regex

class `xcookie.directive.Effect(action, key, value)`

Bases: `tuple`

Create new instance of Effect(action, key, value)

`_asdict()`

Return a new dict which maps field names to their values.

`_field_defaults = {}`

`_fields = ('action', 'key', 'value')`

classmethod `_make(iterable)`

Make a new Effect object from a sequence or iterable

`_replace(**kws)`

Return a new Effect object replacing specified fields with new values

action

Alias for field number 0

key

Alias for field number 1

value

Alias for field number 2

`xcookie.directive.extract_directive_comment(source)`

Different than the xdoctest version. Finds the last comment part of a line

`source = '# acommend # b comment' list(extract_directive_comment(source))`

TODO: lark grammar?

class `xcookie.directive.Directive(name, positive=True, args=[], inline=None)`

Bases: `NiceRepr`

Directives modify the runtime state.

classmethod `extract(text, directive_re, commands)`

Parses directives from a line or repl line

Parameters

- **text** (*str*) – must correspond to exactly one PS1 line and its PS2 followups.

- **prefix** (*str* | *None*) – The directive “namespace”. If *None*, this uses the xdoctest defaults of `DIRECTIVE_RE`. This will always be the case for xdoctest, but we are extending this class for use elsewhere.

Yields

Directive – directive: the parsed directives

_unpack_args(*num*)

effect(*argv=None, environ=None*)

effects(*argv=None, environ=None*)

Returns how this directive modifies a `RuntimeState` object

This is called by `RuntimeState.update()` to update itself

Parameters

- **argv** (*List[str]*, *default=None*) – if specified, overwrite `sys.argv`
- **environ** (*Dict[str, str]*, *default=None*) – if specified, overwrite `os.environ`

Returns

list of named tuples containing:

action (*str*): code indicating how to update key (*str*): name of runtime state item to modify
value (*object*): value to modify with

Return type

`List[Effect]`

`xcookie.directive._split_opstr(optstr)`

Simplified balanced paren logic to only split commas outside of parens

Example

```
>>> optstr = '+FOO, REQUIRES(foo,bar), +ELLIPSIS'
>>> _split_opstr(optstr)
['+FOO', 'REQUIRES(foo,bar)', '+ELLIPSIS']
```

`xcookie.directive._is_requires_satisfied(arg, argv=None, environ=None)`

Determines if the argument to a `REQUIRES` directive is satisfied

Parameters

- **arg** (*str*) – condition code
- **argv** (*List[str]*) – cmdline if arg is cmd code usually `sys.argv`
- **environ** (*Dict[str, str]*) – environment variables usually `os.environ`

Returns

flag - True if the requirement is met

Return type

`bool`

`class xcookie.directive.DirectiveExtractor(namespace, commands)`

Bases: `object`

Example

```
>>> from xcookie.directive import * # NOQA
>>> namespace = 'xcookie'
>>> commands = ['UNCOMMENT_IF', 'COMMENT_IF']
>>> self = DirectiveExtractor(namespace, commands)
>>> text = '- this line is not python # xcookie: +COMMENT_IF(cv2)'
>>> text = '# commented line # xcookie: +UNCOMMENT_IF(cv2)'
>>> extracted = self.extract(text)
>>> assert len(extracted) == 1
>>> directive = extracted[0]
```

extract(*text*)

`xcookie.directive._module_exists(modname)`

`xcookie.directive.parse_directive_optstr(optpart, commands, inline=None)`

Parses the information in the directive from the “optpart”

optstrs are:

optionally prefixed with + (default) or - comma separated may contain one paren enclosed argument (experimental) all spaces are ignored

Returns

the parsed directive

Return type

Directive

1.2.4 xcookie.main module

This is a Python script to apply the xcookie template to either create a new repo or update an existing one with the latest standards.

Todo: Port logic from ~/misc/make_new_python_package_repo.sh

ComamndLine:

~/code/xcookie/xcookie/main.py

python -m xcookie.main

ExampleUsage:

```
# Update my repos python -m xcookie.main --reporid=$HOME/code/pyflann_ibeis
--tags="eroticemic,github,binpy"
```

```
python -m xcookie.main --reporid=$HOME/code/whodat --tags="kitware,gitlab,purepy,cv2,gdal" python
-m xcookie.main --reporid=$HOME/code/whatdat --tags="kitware,gitlab,purepy,cv2,gdal" python -m
xcookie.main --reporid=$HOME/code/whendat --tags="kitware,gitlab,purepy,cv2,gdal" python -m xcookie.main
--reporid=$HOME/code/whydat --tags="kitware,gitlab,purepy,cv2,gdal" python -m xcookie.main --re-
porid=$HOME/code/howdat --tags="kitware,gitlab,purepy,cv2,gdal"
```

```
# Create this repo python -m xcookie.main --repo_name=xcookie --reporid=$HOME/code/xcookie
--tags="eroticemic,github,purepy"
```

```
# Create a new python repo python -m xcookie.main --repo_name=cookiecutter_purepy --re-
porid=$HOME/code/cookiecutter_purepy --tags="github,purepy"
```



```
# Create a new binary repo python -m xcookie.main --repo_name=cookiecutter_binpy --re-
podir=$HOME/code/cookiecutter_binpy --tags="github,binpy,gdal"

# Create a new binary gitlab kitware repo python -m xcookie.main --repo_name=kwimage_ext --re-
podir=$HOME/code/kwimage_ext --tags="kitware,gitlab,binpy"

# Create a new binary github repo python -m xcookie.main --repodir=$HOME/code/networkx_algo_common_subtree
--tags="github,erotemic,binpy"

# Create a new purepy github repo python -m xcookie.main --repodir=$HOME/code/googledoc
--tags="github,erotemic,purepy"

python -m xcookie.main --repodir=$HOME/code/networkx_algo_common_subtree_cython
--tags="github,erotemic,binpy"

python -m xcookie.main --repo_name=delayed_image --repodir=$HOME/code/delayed_image
--tags="kitware,gitlab,purepy,cv2,gdal"

HOST=https://gitlab.kitware.com export PRIVATE_GITLAB_TOKEN=$(git_token_for "$HOST") python -m
xcookie.main --repo_name=kwutil --repodir=$HOME/code/kwutil --tags="kitware,gitlab,purepy"

python -m xcookie.main --repo_name=geowatch --repodir=$HOME/code/geowatch
--tags="kitware,gitlab,purepy,cv2,gdal"

python -m xcookie.main --repo_name=stdx --repodir=$HOME/code/stdx --tags="github,purepy,erotemic"

python -m xcookie.main --repo_name=ustd --repodir=$HOME/code/ustd --tags="github,purepy,erotemic"

load_secrets export PRIVATE_GITLAB_TOKEN=$(git_token_for "https://gitlab.kitware.
com") python -m xcookie.main --repo_name=simple_dvc --repodir=$HOME/code/simple_dvc
--tags="gitlab,kitware,purepy,erotemic"
```

exception `xcookie.main.SkipFile`

Bases: `Exception`

class `xcookie.main.XCookieConfig(*args, **kwargs)`

Bases: `DataConfig`

Valid options: []

Parameters

- ***args** – positional arguments for this data config
- ****kwargs** – keyword arguments for this data config

`_load_pyproject_config()`

`_load_xcookie_pyproject_settings()`

`confirm(msg, default=True)`

Parameters

- **msg** (*str*) – display to the user
- **default** (*bool*) – default value if non-interactive

Return type

`bool`

`prompt(msg, choices, default=True)`

Parameters

- **msg** (*str*) – display to the user

- **default** (*bool*) – default value if non-interactive

Return type

bool

classmethod `main(cmdline=0, **kwargs)`

Main entry point

Example

```

repodir      =      ub.Path.appdir('pypkg/demo/my_new_repo')      import      sys,      ubelt
sys.path.append(ubelt.expandpath('~/.code/xcookie')) from xcookie.main import * # NOQA kwargs
= {
    'repodir': repodir,
} cmdline = 0

default = {'author': <Value(None)>, 'author_email': <Value(None)>, 'autostage':
<Value(False)>, 'ci_cpython_versions': <Value('auto')>,
'ci_pypi_live_password_varname': <Value('TWINE_PASSWORD')>,
'ci_pypi_test_password_varname': <Value('TEST_TWINE_PASSWORD')>,
'ci_pypy_versions': <Value('auto')>, 'ci_versions_full_loose': <Value('*')>,
'ci_versions_full_strict': <Value('max')>, 'ci_versions_minimal_loose':
<Value('max')>, 'ci_versions_minimal_strict': <Value('min')>, 'defaultbranch':
<Value('main')>, 'description': <Value(None)>, 'dev_status': <Value('planning')>,
'enable_gpg': <Value(True)>, 'init_new_remotes': <Value(True)>, 'interactive':
<Value(True)>, 'is_new': <Value('auto')>, 'license': <Value(None)>, 'linter':
<Value(True)>, 'max_python': <Value(None)>, 'min_python': <Value('3.7')>,
'mod_name': <Value(None)>, 'os': <Value('all')>, 'pkg_name': <Value(None)>,
'refresh_docs': <Value('auto')>, 'regen': <Value(None)>, 'rel_mod_parent_dpath':
<Value('.')>, 'remote_group': <Value(None)>, 'remote_host': <Value(None)>,
'render_doc_images': <Value(False)>, 'repo_name': <Value(None)>, 'repodir':
<Value('.')>, 'rotate_secrets': <Value('auto')>, 'supported_python_versions':
<Value('auto')>, 'tags': <Value('auto')>, 'test_command': <Value('auto')>,
'test_variants': <Value(['full-loose', 'full-strict', 'minimal-loose',
'minimal-strict'])>, 'typed': <Value(None)>, 'url': <Value(None)>, 'use_vcs':
<Value('auto')>, 'version': <Value(None)>, 'visibility': <Value('public')>,
'xdctest_style': <Value('google')>, 'yes': <Value(False)>}
```

`normalize()`

class `xcookie.main.TemplateApplier(config)`

Bases: `object`

The primary xcookie autogeneration class.

Note: this does not write any files unless you call `setup` (which just writes to a temporary directory) or `apply` (which can destructively clobber things).

`apply()`

Does the actual modification of the target repo.

Has special logic to handle building new repos versus updating repos.

`autostage()`

property `has_git`

property `rel_mod_dpath`

property `mod_dpath`

property `mod_name`

property `pkg_name`

`_build_template_registry()`

Take stock of the files in the template repo and ensure they all have appropriate properties.

property `tags`

`_presetup()`

This logic used to live in `setup`, but it doesn't have external side effects, so it would be good if we were able have these fields populated on initialization for tests.

`setup()`

Finalizes a few variables and writes the “clean” template to the staging directory.

`copy_staged_files()`

`vcs_checks()`

`_stage_file(info)`

Write a single file to the staging directory based on its template info.

Parameters

info (*dict*) – a template dictionary that defines how to construct a file

Returns

enriched information.

A side effect of this function is writing the data to temporary storage

Return type

`dict`

Example

```
>>> from xcookie.main import * # NOQA
>>> dpath = ub.Path.appdir('xcookie/tests/test-stage').delete().ensuredir()
>>> kwargs = {
>>>     'repodir': dpath / 'testrepo',
>>>     'tags': ['gitlab', 'kitware', 'purepy', 'cv2'],
>>>     'rotate_secrets': False,
>>>     'is_new': False,
>>>     'interactive': False,
>>> }
>>> config = XCookieConfig.cli(cmdline=0, data=kwargs)
>>> #config.__post_init__()
>>> print('config = {}'.format(ub.urepr(dict(config), nl=1)))
>>> self = TemplateApplier(config)
```

(continues on next page)

(continued from previous page)

```
>>> self._build_template_registry()
>>> info = [d for d in self.template_infos if d['fname'] == 'setup.py'][0]
>>> info = [d for d in self.template_infos if d['fname'] == '.gitlab-ci.yml'][0]
>>> self._stage_file(info)
```

`_apply_xcookie_directives(stage_fpath)`

`stage_files()`

`gather_tasks()`

`build_requirements()`

`refresh_docs()`

`rotate_secrets()`

`print_help_tips()`

`build_readthedocs()`

Returns

templated code

Return type

`str`

`build_setup()`

Returns

templated code

Return type

`str`

`build_pyproject()`

Returns

templated code

Return type

`str`

`build_github_actions()`

`build_gitlab_ci()`

`build_run_linter()`

`build_gitlab_rules()`

`build_readme()`

`build_docs_index()`

`build_docs_conf()`

`build_docs_requirements()`

`build_run_doctests()`

lut(*info*)

Hacked builders when template_info source is dynamic, but there is no corresponding explicit function.

Returns

templated code

Return type

str

_docs_quickstart()

Todo:

- [] Auto-edit the index.py to include the magic reference to `__init__`
 - [] If this is a utility library, populate the “usefulness section”
 - [] Try and find out how to auto-expand the toc tree
 - [] Auto-run the sphinx-apidoc for the user
-

```
REPO_NAME=xcookie REPO_DPATH=$HOME/code/$REPO_NAME AUTHOR=$(git config --global
user.name) cd $REPO_DPATH/docs sphinx-quickstart -q --sep
```

```
--project="$REPO_NAME" --author="$AUTHOR" --ext-autodoc --ext-viewcode --ext-intersphinx
--ext-todo --extensions=sphinx.ext.autodoc,sphinx.ext.viewcode,sphinx.ext.napoleon,sphinx.ext.intersphinx,sphinx.ext.toctree
"$REPO_DPATH/docs"
```

```
# THEN NEED TO: REPO_NAME=kwarray REPO_DPATH=$HOME/code/$REPO_NAME
MOD_DPATH=$REPO_DPATH/$REPO_NAME sphinx-apidoc -f -o "$REPO_DPATH/docs/source"
"$MOD_DPATH" --separate cd "$REPO_DPATH/docs" make html
```

The user will need to enable the repo on their readthedocs account: <https://readthedocs.org/dashboard/import/manual/>

To enable the read-the-docs go to <https://readthedocs.org/dashboard/> and login

Make sure you have a .readthedocs.yml file

Click import project: (for github you can select, but gitlab you need to import manually)

Set the Repository NAME: \$REPO_NAME Set the Repository URL: \$REPO_URL

For gitlab you also need to setup an integrations and add gitlab incoming webhook Then go to \$REPO_URL/hooks and add the URL

Will also need to activate the main branch:

<https://readthedocs.org/projects/xcookie/versions/>

```
xcookie.main.main()
```

```
xcookie.main._parse_remote_url(url)
```

```
xcookie.main.find_git_root(dpath)
```

```
class xcookie.main.GitURL(data)
```

Bases: str

Represents a url to a git repo and can parse info about / modify the protocol

Example

```
>>> from git_well.git_remote_protocol import * # NOQA
>>> url1 = GitURL('https://foo.bar/user/repo.git')
>>> url2 = GitURL('git@foo.bar:group/repo.git')
>>> print(url1.to_git())
>>> print(url1.to_https())
>>> print(url2.to_git())
>>> print(url2.to_https())
git@foo.bar:user/repo.git
https://foo.bar/user/repo.git
git@foo.bar:group/repo.git
https://foo.bar/group/repo.git
>>> print(ub.urepr(url1.info))
>>> print(ub.urepr(url2.info))
{
  'host': 'foo.bar',
  'group': 'user',
  'repo_name': 'repo.git',
  'protocol': 'https',
  'url': 'https://foo.bar/user/repo.git',
}
{
  'host': 'foo.bar',
  'group': 'group',
  'repo_name': 'repo.git',
  'protocol': 'git',
  'url': 'git@foo.bar:group/repo.git',
}
```

property info

to_git()

to_https()

1.2.5 xcookie.rich_ext module

```
class xcookie.rich_ext.FuzzyPrompt(prompt: str | Text = "", *, console: Console | None = None, password:
                                   bool = False, choices: List[str] | None = None, show_default: bool =
                                   True, show_choices: bool = True)
```

Bases: `Prompt`

The user just needs to enter a non-ambiguous prefix

process_response(value: str) → str

Normalize the response

1.2.6 xcookie.util_yaml module

Vendored from kwutil.util_yaml

class xcookie.util_yaml._YamlRepresenter

Bases: `object`

static `str_presenter(dumper, data)`

`xcookie.util_yaml._custom_ruaml_loader()`

References

<https://stackoverflow.com/questions/59635900/ruamel-yaml-custom-commentedmapping-for-custom-tags>

<https://stackoverflow.com/questions/528281/how-can-i-include-a-yaml-file-inside-another>

`xcookie.util_yaml._custom_ruaml_dumper()`

References

<https://stackoverflow.com/questions/59635900/ruamel-yaml-custom-commentedmapping-for-custom-tags>

`xcookie.util_yaml._custom_pyaml_dumper()`

class xcookie.util_yaml.Yaml

Bases: `object`

Namespace for yaml functions

static `dumps(data, backend='ruamel')`

Dump yaml to a string representation (and account for some of our use-cases)

Parameters

- **data** (*Any*) – yaml representable data
- **backend** (*str*) – either ruamel or pyyaml

Returns

yaml text

Return type

`str`

Example

```
>>> import ubelt as ub
>>> data = {
>>>     'a': 'hello world',
>>>     'b': ub.udict({'a': 3})
>>> }
>>> text1 = Yaml.dumps(data, backend='ruamel')
>>> print(text1)
>>> text2 = Yaml.dumps(data, backend='pyyaml')
>>> print(text2)
>>> assert text1 == text2
```

static load(*file*, *backend*='ruamel')

Load yaml from a file

Parameters

- **file** (*io.TextIOBase* | *PathLike* | *str*) – yaml file path or file object
- **backend** (*str*) – either ruamel or pyyaml

Returns

object

static loads(*text*, *backend*='ruamel')

Load yaml from a text

Parameters

- **text** (*str*) – yaml text
- **backend** (*str*) – either ruamel or pyyaml

Returns

object

Example

```
>>> import ubelt as ub
>>> data = {
>>>     'a': 'hello world',
>>>     'b': ub.udict({'a': 3})
>>> }
>>> print('data = {}'.format(ub.urepr(data, nl=1)))
>>> print('---')
>>> text = Yaml.dumps(data)
>>> print(ub.highlight_code(text, 'yaml'))
>>> print('---')
>>> data2 = Yaml.loads(text)
>>> assert data == data2
>>> data3 = Yaml.loads(text, backend='pyyaml')
>>> print('data2 = {}'.format(ub.urepr(data2, nl=1)))
>>> print('data3 = {}'.format(ub.urepr(data3, nl=1)))
>>> assert data == data3
```

static coerce(*data*, *backend*='ruamel')

Attempt to convert input into a parsed yaml / json data structure. If the data looks like a path, it tries to load and parse file contents. If the data looks like a yaml/json string it tries to parse it. If the data looks like parsed data, then it returns it as-is.

Parameters

- **data** (*str* | *PathLike* | *dict* | *list*)
- **backend** (*str*) – either ruamel or pyyaml

Returns

parsed yaml data

Return type

object

Note: The input to the function cannot distinguish a string that should be loaded and a string that should be parsed. If it looks like a file that exists it will read it. To avoid this corner case use this only for data where you expect the output is a List or Dict.

References

<https://stackoverflow.com/questions/528281/how-can-i-include-a-yaml-file-inside-another>

Example

```
>>> Yaml.coerce('[1, 2, 3]')
[1, 2, 3]
>>> fpath = ub.Path.appdir('cmd_queue/tests/util_yaml').ensuredir() / 'file.yaml'
↪
>>> fpath.write_text(Yaml.dumps([4, 5, 6]))
>>> Yaml.coerce(fpath)
[4, 5, 6]
>>> Yaml.coerce(str(fpath))
[4, 5, 6]
>>> dict(Yaml.coerce('{a: b, c: d}'))
{'a': 'b', 'c': 'd'}
>>> Yaml.coerce(None)
None
```

Example

```
>>> assert Yaml.coerce('') is None
```

Example

```
>>> dpath = ub.Path.appdir('cmd_queue/tests/util_yaml').ensuredir()
>>> fpath = dpath / 'external.yaml'
>>> fpath.write_text(Yaml.dumps({'foo': 'bar'}))
>>> text = ub.codeblock(
>>>     f'''
>>>     items:
>>>     - !include {dpath}/external.yaml
>>>     '''
>>> data = Yaml.coerce(text, backend='ruamel')
>>> print(Yaml.dumps(data, backend='ruamel'))
items:
- foo: bar
```

```
>>> text = ub.codeblock(
>>>     f'''
>>>     items:
```

(continues on next page)

(continued from previous page)

```
>>> !include [{dpath}/external.yaml, blah, 1, 2, 3]
>>> '''
>>> data = Yaml.coerce(text, backend='ruamel')
>>> print('data = {}'.format(ub.urepr(data, nl=1)))
>>> print(Yaml.dumps(data, backend='ruamel'))
```

static `InlineList(items)`

References

static `Dict(data)`

Get a ruamel-enhanced dictionary

Example

```
>>> data = {'a': 'avalue', 'b': 'bvalue'}
>>> data = Yaml.Dict(data)
>>> data.yaml_set_start_comment('hello')
>>> # Note: not working https://sourceforge.net/p/ruamel-yaml/tickets/400/
>>> data.yaml_set_comment_before_after_key('a', before='a comment', indent=2)
>>> data.yaml_set_comment_before_after_key('b', 'b comment')
>>> print(Yaml.dumps(data))
```

static `CodeBlock(text)`

`xcookie.util_yaml._dev()`

1.2.7 xcookie.vcs_remotes module

exception `xcookie.vcs_remotes.NotFound`

Bases: `Exception`

exception `xcookie.vcs_remotes.Ambiguous`

Bases: `Exception`

`xcookie.vcs_remotes._return_one(found)`

class `xcookie.vcs_remotes.GitlabRemote(proj_name, proj_group, url, visibility='public',
private_token='env:PRIVATE_GITLAB_TOKEN')`

Bases: `object`

`pip install python-gitlab`

auth()

property `group`

property `project`

new_project()

```
set_protected_branches()
```

```
class xcookie.vcs_remotes.GithubRemote(proj_name)
```

```
Bases: object
```

```
new_project()
```

```
publish_release()
```

```
POC for making a release script
```

References

https://cli.github.com/manual/gh_release_create

```
xcookie.vcs_remotes.version_bump()
```

```
xcookie.vcs_remotes._parse_changelog(fpath)
```

Helper to parse the changelog for the version to verify versions agree.

CommandLine

```
xdoctest -m dev/parse_changelog.py _parse_changelog --dev
fpath = "CHANGELOG.md"
```

1.3 Module contents

The xcookie module is a CLI tool for initializing and maintaining standardized repo infrastructure. In other words it is a Python project cookie cutter that attempts to help keep existing modules up to date with latest infrastructure developments.

Currently this module is specialized towards Erotemic / PyUtils / Kitware projects but the goal is to eventually generalize everything.

Read the docs	https://xcookie.readthedocs.io
Github	https://github.com/Erotemic/xcookie
Pypi	https://pypi.org/project/xcookie

INDICES AND TABLES

- `genindex`
- `modindex`

BIBLIOGRAPHY

[SO56937691] <https://stackoverflow.com/questions/56937691/making-yaml-ruamel-yaml-always-dump-lists-inline>

PYTHON MODULE INDEX

X

- `xcookie`, 23
- `xcookie.__init__`, 1
- `xcookie.__main__`, 10
- `xcookie.builders`, 7
 - `xcookie.builders._builder`, 3
 - `xcookie.builders.common_ci`, 3
 - `xcookie.builders.docs`, 4
 - `xcookie.builders.github_actions`, 4
 - `xcookie.builders.gitlab_ci`, 6
 - `xcookie.builders.pyproject`, 6
 - `xcookie.builders.readme`, 7
 - `xcookie.builders.readthedocs`, 7
 - `xcookie.builders.setup`, 7
- `xcookie.constants`, 10
- `xcookie.directive`, 10
- `xcookie.main`, 12
- `xcookie.rc`, 9
 - `xcookie.rc.conf_ext`, 7
- `xcookie.rich_ext`, 18
- `xcookie.util_yaml`, 19
- `xcookie.vcs_remotes`, 22

Symbols

_YamlRepresenter (class in xcookie.util_yaml), 19
 _apply_xcookie_directives() (xcookie.main.TemplateApplier method), 16
 _asdict() (xcookie.directive.Effect method), 10
 _available_action_methods() (xcookie.builders.github_actions.Actions class method), 4
 _build_template_registry() (xcookie.main.TemplateApplier method), 15
 _check_for_updates() (xcookie.builders.github_actions.Actions class method), 4
 _custom_pyaml_dumper() (in module xcookie.util_yaml), 19
 _custom_ruaml_dumper() (in module xcookie.util_yaml), 19
 _custom_ruaml_loader() (in module xcookie.util_yaml), 19
 _dev() (in module xcookie.util_yaml), 22
 _docs_quickstart() (xcookie.main.TemplateApplier method), 17
 _field_defaults (xcookie.directive.Effect attribute), 10
 _fields (xcookie.directive.Effect attribute), 10
 _ibeis_badges() (in module xcookie.builders.readme), 7
 _is_requires_satisfied() (in module xcookie.directive), 11
 _load_pyproject_config() (xcookie.main.XCookieConfig method), 13
 _load_xcookie_pyproject_settings() (xcookie.main.XCookieConfig method), 13
 _make() (xcookie.directive.Effect class method), 10
 _module_exists() (in module xcookie.directive), 12
 _parse_changelog() (in module xcookie.vcs_remotes), 23
 _parse_remote_url() (in module xcookie.main), 17
 _presetup() (xcookie.main.TemplateApplier method), 15

_register_builtins() (xcookie.rc.conf_ext.GoogleStyleDocstringProcessor method), 8
 _replace() (xcookie.directive.Effect method), 10
 _return_one() (in module xcookie.vcs_remotes), 22
 _split_opstr() (in module xcookie.directive), 11
 _stage_file() (xcookie.main.TemplateApplier method), 15
 _unpack_args() (xcookie.directive.Directive method), 11

A

action (xcookie.directive.Effect attribute), 10
 action() (xcookie.builders.github_actions.Actions class method), 4
 action_versions (xcookie.builders.github_actions.Actions attribute), 4
 Actions (class in xcookie.builders.github_actions), 4
 Ambiguous, 22
 apply() (xcookie.main.TemplateApplier method), 14
 auth() (xcookie.vcs_remotes.GitlabRemote method), 22
 autostage() (xcookie.main.TemplateApplier method), 14

B

BadgeBuilder (class in xcookie.builders.readme), 7
 build_and_test_sdist_job() (in module xcookie.builders.github_actions), 5
 build_binpy_wheels_job() (in module xcookie.builders.github_actions), 5
 build_deploy() (in module xcookie.builders.github_actions), 5
 build_docs_conf() (in module xcookie.builders.docs), 4
 build_docs_conf() (xcookie.main.TemplateApplier method), 16
 build_docs_index() (in module xcookie.builders.docs), 4
 build_docs_index() (xcookie.main.TemplateApplier method), 16
 build_docs_requirements() (in module xcookie.builders.docs), 4

`build_docs_requirements()` (*xcookie.main.TemplateApplier* method), 16
`build_github_actions()` (in module *xcookie.builders.github_actions*), 5
`build_github_actions()` (*xcookie.main.TemplateApplier* method), 16
`build_github_release()` (in module *xcookie.builders.github_actions*), 6
`build_gitlab_ci()` (in module *xcookie.builders.gitlab_ci*), 6
`build_gitlab_ci()` (*xcookie.main.TemplateApplier* method), 16
`build_gitlab_rules()` (in module *xcookie.builders.gitlab_ci*), 6
`build_gitlab_rules()` (*xcookie.main.TemplateApplier* method), 16
`build_purewheel_job()` (in module *xcookie.builders.github_actions*), 5
`build_pyproject()` (in module *xcookie.builders.pyproject*), 6
`build_pyproject()` (*xcookie.main.TemplateApplier* method), 16
`build_readme()` (in module *xcookie.builders.readme*), 7
`build_readme()` (*xcookie.main.TemplateApplier* method), 16
`build_readthedocs()` (in module *xcookie.builders.readthedocs*), 7
`build_readthedocs()` (*xcookie.main.TemplateApplier* method), 16
`build_requirements()` (*xcookie.main.TemplateApplier* method), 16
`build_run_doctests()` (*xcookie.main.TemplateApplier* method), 16
`build_run_linter()` (*xcookie.main.TemplateApplier* method), 16
`build_setup()` (in module *xcookie.builders.setup*), 7
`build_setup()` (*xcookie.main.TemplateApplier* method), 16
`build_templates()` (*xcookie.builders.readme.BadgeBuilder* method), 7
`Builder` (class in *xcookie.builders._builder*), 3

C

`checkout()` (*xcookie.builders.github_actions.Actions* class method), 4
`cibuildwheel()` (*xcookie.builders.github_actions.Actions* class method), 5
`CodeBlock()` (*xcookie.util_yaml.Yaml* static method), 22

`codecov_action()` (*xcookie.builders.github_actions.Actions* class method), 4
`coerce()` (*xcookie.util_yaml.Yaml* static method), 20
`combine_coverage()` (*xcookie.builders.github_actions.Actions* class method), 5
`confirm()` (*xcookie.main.XCookieConfig* method), 13
`copy_staged_files()` (*xcookie.main.TemplateApplier* method), 15
`create_doctest_figure()` (in module *xcookie.rc.conf_ext*), 9

D

`default` (*xcookie.main.XCookieConfig* attribute), 14
`Dict()` (*xcookie.util_yaml.Yaml* static method), 22
`Directive` (class in *xcookie.directive*), 10
`DirectiveExtractor` (class in *xcookie.directive*), 11
`docs_auto_outdir` (*xcookie.builders.docs.DocsBuilder* property), 4
`docs_dpath` (*xcookie.builders.docs.DocsBuilder* property), 4
`DocsBuilder` (class in *xcookie.builders.docs*), 4
`download_artifact()` (*xcookie.builders.github_actions.Actions* class method), 5
`dumps()` (*xcookie.util_yaml.Yaml* static method), 19

E

`Effect` (class in *xcookie.directive*), 10
`effect()` (*xcookie.directive.Directive* method), 11
`effects()` (*xcookie.directive.Directive* method), 11
`extract()` (*xcookie.directive.Directive* class method), 10
`extract()` (*xcookie.directive.DirectiveExtractor* method), 12
`extract_directive_comment()` (in module *xcookie.directive*), 10

F

`find_git_root()` (in module *xcookie.main*), 17
`find_tagged_lines()` (*xcookie.rc.conf_ext.SphinxDocstring* method), 9
`fix_rst_todo_section()` (in module *xcookie.rc.conf_ext*), 9
`FuzzyPrompt` (class in *xcookie.rich_ext*), 18

G

`gather_tasks()` (*xcookie.main.TemplateApplier* method), 16
`get_supported_platform_info()` (in module *xcookie.builders.github_actions*), 5
`GithubRemote` (class in *xcookie.vcs_remotes*), 23
`GitlabRemote` (class in *xcookie.vcs_remotes*), 22
`GitURL` (class in *xcookie.main*), 17

GoogleStyleDocstringProcessor (class in `xcookie.rc.conf_ext`), 7
 group (`xcookie.vcs_remotes.GitlabRemote` property), 22

H

has_git (`xcookie.main.TemplateApplier` property), 15

I

info (`xcookie.main.GitURL` property), 18
 InlineList() (`xcookie.util_yaml.Yaml` static method), 22

K

key (`xcookie.directive.Effect` attribute), 10

L

lint_job() (in module `xcookie.builders.github_actions`), 5
 load() (`xcookie.util_yaml.Yaml` static method), 19
 loads() (`xcookie.util_yaml.Yaml` static method), 20
 lut() (`xcookie.main.TemplateApplier` method), 16

M

main() (in module `xcookie.main`), 17
 main() (`xcookie.main.XCookieConfig` class method), 14
 make_build_wheel_parts() (in module `xcookie.builders.common_ci`), 3
 make_install_and_test_wheel_parts() (in module `xcookie.builders.common_ci`), 3
 make_mypy_check_parts() (in module `xcookie.builders.common_ci`), 3
 make_purepy_ci_jobs() (in module `xcookie.builders.gitlab_ci`), 6
 mod_dpath (`xcookie.main.TemplateApplier` property), 15
 mod_name (`xcookie.main.TemplateApplier` property), 15
 module
 `xcookie`, 23
 `xcookie.__init__`, 1
 `xcookie.__main__`, 10
 `xcookie.builders`, 7
 `xcookie.builders._builder`, 3
 `xcookie.builders.common_ci`, 3
 `xcookie.builders.docs`, 4
 `xcookie.builders.github_actions`, 4
 `xcookie.builders.gitlab_ci`, 6
 `xcookie.builders.pyproject`, 6
 `xcookie.builders.readme`, 7
 `xcookie.builders.readthedocs`, 7
 `xcookie.builders.setup`, 7
 `xcookie.constants`, 10
 `xcookie.directive`, 10
 `xcookie.main`, 12
 `xcookie.rc`, 9

`xcookie.rc.conf_ext`, 7
`xcookie.rich_ext`, 18
`xcookie.util_yaml`, 19
`xcookie.vcs_remotes`, 22

msvc_dev_cmd() (`xcookie.builders.github_actions.Actions` class method), 5

N

named() (in module `xcookie.directive`), 10
 new_project() (`xcookie.vcs_remotes.GithubRemote` method), 23
 new_project() (`xcookie.vcs_remotes.GitlabRemote` method), 22
 normalize() (`xcookie.main.XCookieConfig` method), 14
 NotFound, 22

P

paragraph() (in module `xcookie.rc.conf_ext`), 9
 parse_directive_optstr() (in module `xcookie.directive`), 12
 PatchedPythonDomain (class in `xcookie.rc.conf_ext`), 7
 pkg_name (`xcookie.main.TemplateApplier` property), 15
 postprocess_hyperlinks() (in module `xcookie.rc.conf_ext`), 9
 print_help_tips() (`xcookie.main.TemplateApplier` method), 16
 process() (`xcookie.rc.conf_ext.GoogleStyleDocstringProcessor` method), 8
 process_docstring_callback() (`xcookie.rc.conf_ext.GoogleStyleDocstringProcessor` method), 8
 process_response() (`xcookie.rich_ext.FuzzyPrompt` method), 18
 project (`xcookie.vcs_remotes.GitlabRemote` property), 22
 prompt() (`xcookie.main.XCookieConfig` method), 13
 publish_release() (`xcookie.vcs_remotes.GithubRemote` method), 23

R

ReadmeBuilder (class in `xcookie.builders.readme`), 7
 refresh_docs() (`xcookie.main.TemplateApplier` method), 16
 register_section() (`xcookie.rc.conf_ext.GoogleStyleDocstringProcessor` method), 7
 rel_mod_dpath (`xcookie.main.TemplateApplier` property), 15
 resolve_xref() (`xcookie.rc.conf_ext.PatchedPythonDomain` method), 7
 resource_fpath() (in module `xcookie.rc`), 9
 rotate_secrets() (`xcookie.main.TemplateApplier` method), 16

S

`set_protected_branches()`
 (*xcookie.vcs_remotes.GitlabRemote* method),
 22

`setup()` (*in module xcookie.rc.conf_ext*), 9

`setup()` (*xcookie.main.TemplateApplier* method), 15

`setup_ipfs()` (*xcookie.builders.github_actions.Actions*
 class method), 5

`setup_python()` (*xcookie.builders.github_actions.Actions*
 class method), 4

`setup_qemu()` (*xcookie.builders.github_actions.Actions*
 class method), 5

`setup_xcode()` (*xcookie.builders.github_actions.Actions*
 class method), 5

`SkipFile`, 13

`sphinx_apidoc_invocation()`
 (*xcookie.builders.docs.DocsBuilder* method), 4

`SphinxDocstring` (*class in xcookie.rc.conf_ext*), 9

`stage_files()` (*xcookie.main.TemplateApplier*
 method), 16

`str_presenter()` (*xcookie.util_yaml._YamlRepresenter*
 static method), 19

T

`tags` (*xcookie.main.TemplateApplier* property), 15

`TemplateApplier` (*class in xcookie.main*), 14

`test_wheels_job()` (*in module*
 xcookie.builders.github_actions), 5

`to_git()` (*xcookie.main.GitURL* method), 18

`to_https()` (*xcookie.main.GitURL* method), 18

U

`upload_artifact()` (*xcookie.builders.github_actions.Actions*
 class method), 5

V

`value` (*xcookie.directive.Effect* attribute), 10

`vcs_checks()` (*xcookie.main.TemplateApplier* method),
 15

`version_bump()` (*in module xcookie.vcs_remotes*), 23

X

`xcookie`
 module, 23

`xcookie.__init__`
 module, 1

`xcookie.__main__`
 module, 10

`xcookie.builders`
 module, 7

`xcookie.builders._builder`
 module, 3

`xcookie.builders.common_ci`

module, 3

`xcookie.builders.docs`
 module, 4

`xcookie.builders.github_actions`
 module, 4

`xcookie.builders.gitlab_ci`
 module, 6

`xcookie.builders.pyproject`
 module, 6

`xcookie.builders.readme`
 module, 7

`xcookie.builders.readthedocs`
 module, 7

`xcookie.builders.setup`
 module, 7

`xcookie.constants`
 module, 10

`xcookie.directive`
 module, 10

`xcookie.main`
 module, 12

`xcookie.rc`
 module, 9

`xcookie.rc.conf_ext`
 module, 7

`xcookie.rich_ext`
 module, 18

`xcookie.util_yaml`
 module, 19

`xcookie.vcs_remotes`
 module, 22

`XCookieConfig` (*class in xcookie.main*), 13

`Yaml` (*class in xcookie.util_yaml*), 19