scverse cookiecutter template

Follow along: scverse.org/cookiecutter-scverse-presentation

having impact with software

- Publish papers and get citations for academic success
- High quality software helps a lot with both
- Be part of the scverse ecosystem: ensure you meet minimum requirements for maximal impact
  - Get reviewed by PyOpenSci?
- Workshop goal: path to high quality software dev

scverse ecosystem

- Set of packages built around the scverse core packages
- Featured on the scverse ecosystem website

scverse ecosystem requirements

- Metadata: name, description, OSI-approved license, …
- Versioned releases installable from PyPI &/or Conda
- Tests automated via continuous integration (CI)
- Docs for API and use-cases
- Scverse datastructures used where appropriate

how to get into the scverse ecosystem

Instructions to add your package are on the scverse ecosystem repository

- Submit existing package, or
- Easiest way: scverse cookiecutter template

scverse cookiecutter template

- Project template for scverse packages[1]
- Best practice structure
- Many features such as continuous integration, documentation setup, …
- Automatically ticks all scverse ecosystem requirements

[1]Much is applicable elsewhere

steps

1. Set up global environment to create project
2. Create project using scverse cookiecutter template
3. Set up project specific development environment
4. Develop your package
5. Submit to scverse ecosystem
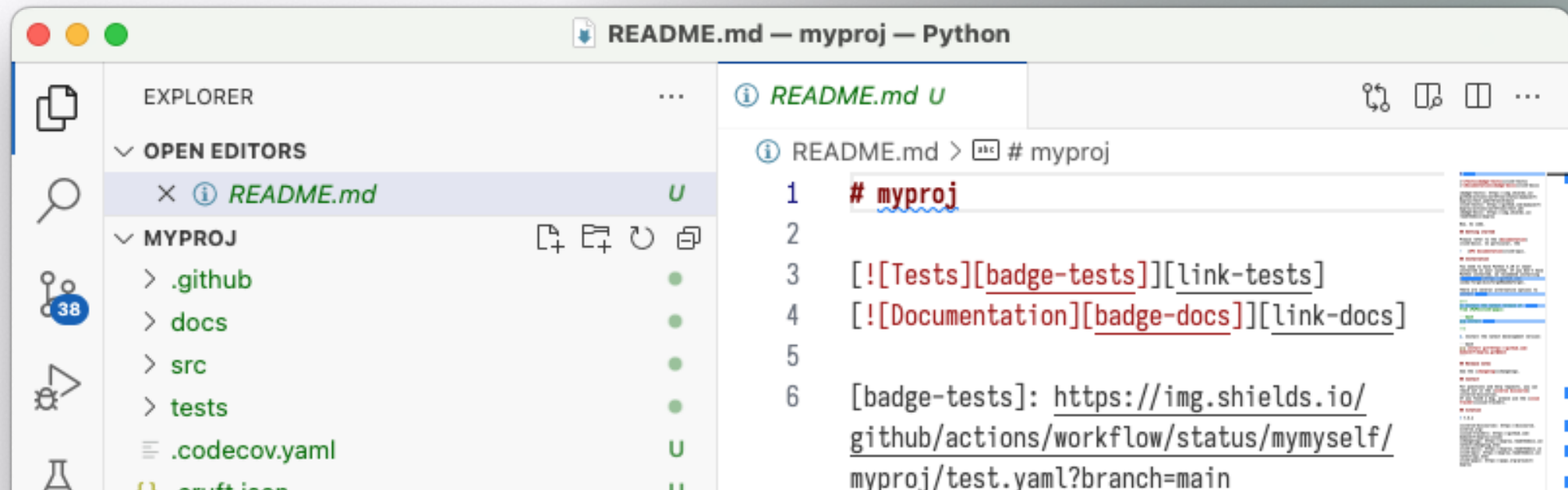
step 1: global environment

- Install using a package manager or installer:
  - `code` , `hatch` , & `git`
  - `uv` or `pipx`
- Install using a package manager, `pipx` , or `uv` :
  - `cruft` & `pre-commit`

```
$ pipx install cruft pre-commit  # or
$ uv tool install cruft pre-commit
```

step 2: creating the project

```
$ cruft create https://github.com/scverse/cookiecutter-scverse
  project_name (project-name): myproj
  […]
$ code myproj
```

# step 3: environment management

Hatch envs basic usage:

```
$ hatch run [env:]command [...args] # e.g. `… docs:build -T`
$ hatch test [...args]
$ hatch env remove <name> # or `hatch env prune` for all
$ hatch env find hatch-test
~/.local/share/hatch/env/virtual/myproj/FsejNibV/hatch-test.py3.12
[…]
```

Make sure it exists, then tell VS Code:

⌘|^+▢+P → Python: Select Interpreter

running tests

```
$ hatch test --help
[…]
Options:
  -r, --randomize
  -p, --parallel
  -c, --cover
  -a, --all
  -py, --python=X.Y
  -i, --include=VAR=VAL
  -x, --exclude=VAR=VAL
  -s, --show
…otherwise same as `pytest`
```
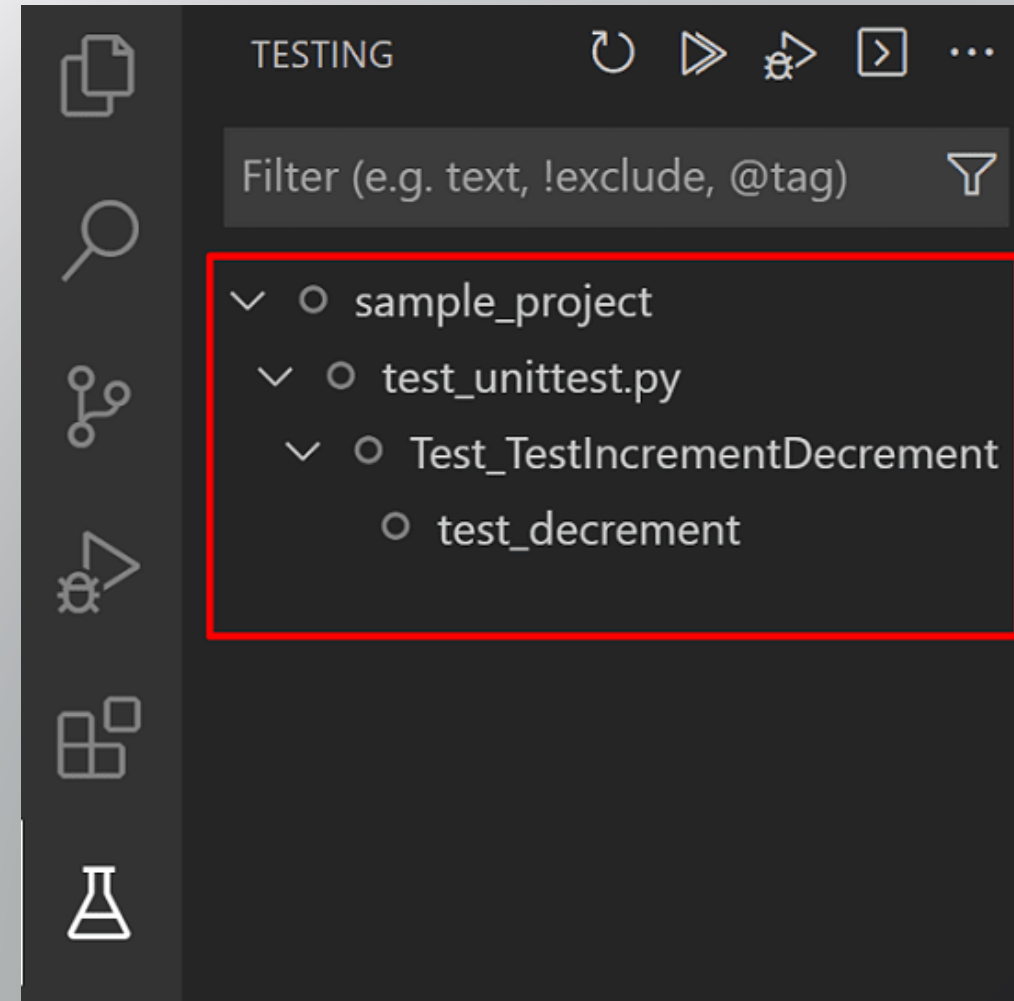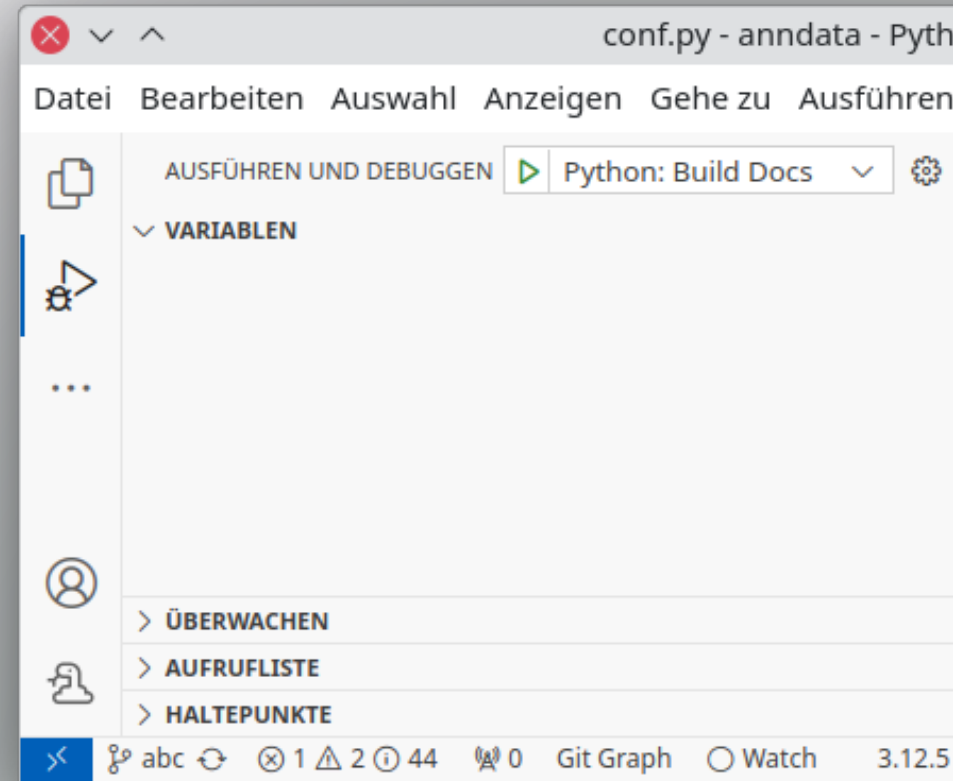
building docs

```
$ hatch run docs:build
$ hatch run docs:open
$ hatch run docs:clean
```

See pyproject.toml:

```
[tools.hatch.envs.docs]
scripts.build = "..."
...
```

formatting and linting

VS Code:

```
{
  "[python]": {
    "editor.formatOnSave": true,
    "editor.defaultFormatter": "charliermarsh.ruff",
    "editor.codeActionsOnSave": { ... },
  }, ...
}
```

CLI: pre-commit (or hatch run pre-commit)

```
$ pre-commit install # `git commit` hook
$ pre-commit run --all-files
```
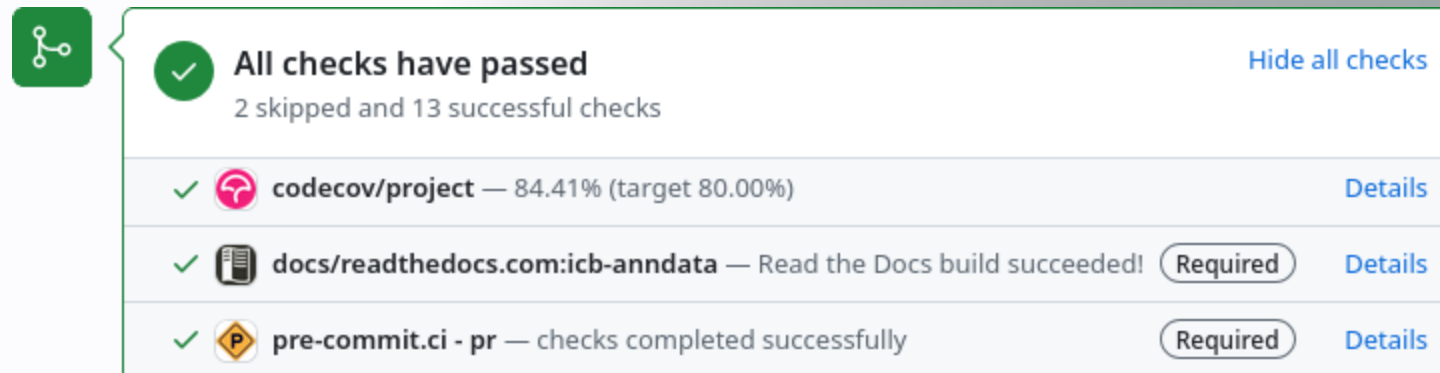
12

existing project

- Simple project

  1. Instantiate template
  2. Replace `src/*` directory with your package
  3. Edit `[project]` table in `pyproject.toml`

- Complex project

  1. Step by step PRs: formatter, …
  2. We can help!

follow along

step 4: committing code

- Use PRs, don't push to `main`
- Set up pre-commit.ci, codecov.io on github.com/<you>/<yourpackage>/settings/installation

ReadTheDocs

- Set up readthedocs.org and its PR previews:

⚠ **Warning**

This page was created from a pull request (#1616).

step 5: submit to scverse ecosystem

1. Follow the instructions
2. Submit a pull request
3. Maybe get reviewed by PyOpenSci

Thank You!