welcome to

MODERN TOOLS FOR PYTHON DEVELOPMENT: CODE FOR COLLABORATION

github.com/DrGabrielHarris/pydata-norwich-sep-2025

ABOUT ME

PhD from King’s College London in Medical Imaging Sciences in 2015

Over a decade of experience designing and implementing ML/AI solutions using MLOPs-first approach

PyData Norwich Co-organizer

Gabriel Harris Ph.D.

Principal Data Scientist @ NIQ | Designing and Implementing ML/AI solutions in Azure

ESTABLISH PROCESSES & BEST PRACTICES

Python HOW: Using Poetry, Make, and pre-commit-hooks to Setup a Repo Template for your Team

Bring consistency, rigour, and best practices to your messy data science team

Medium / Nov 15, 2022

PyData Cambridge Meetup 2023

welcome to

SETTING UP A REPOSITORY TEMPLATE FOR YOUR TEAM

with Consistency, Reproducibility, and Compliance in Mind

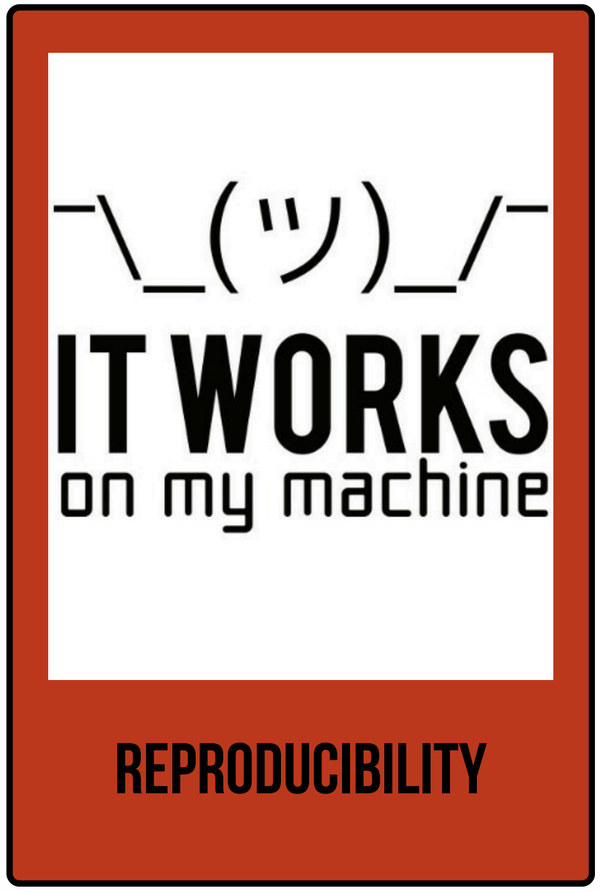
PyData London Conference 2023

welcome to

BRING BEST PRACTICES TO YOUR MESSY DATA SCIENCE TEAM!

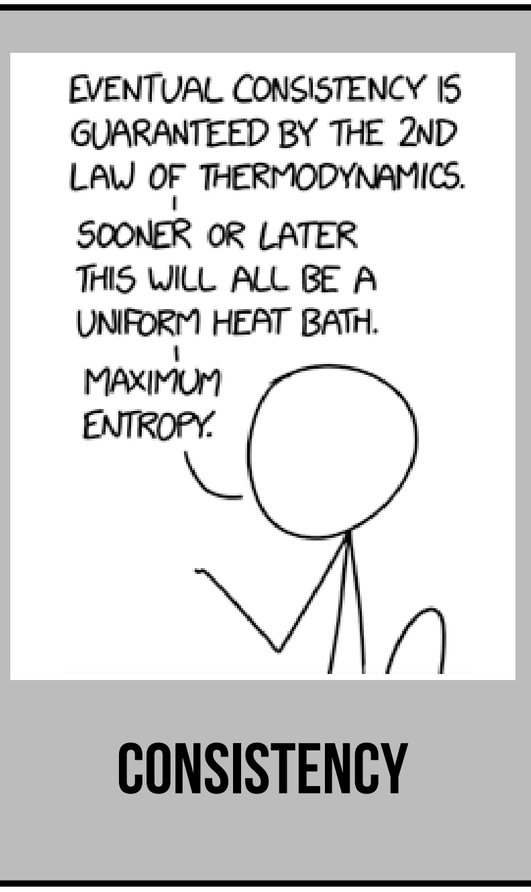
with Reproducibility, Compliance, and Consistency in Mind

WHY?



REPRODUCIBILITY

CONVENTIONS



CONSISTENCY

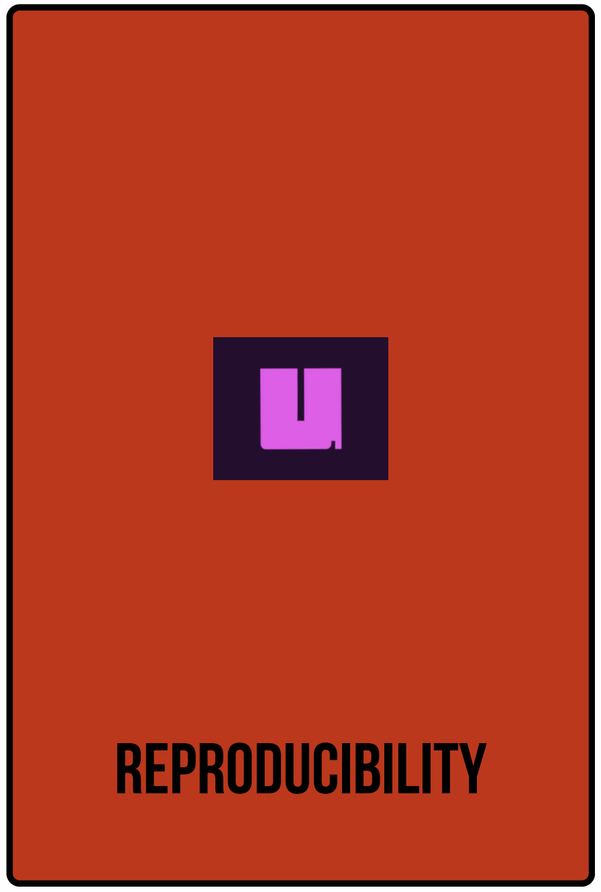
HOW?

REPRODUCIBILITY

CONVENTIONS

CONSISTENCY

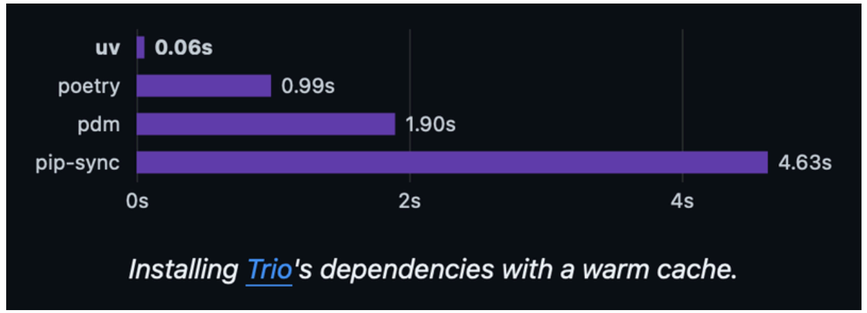
HOW?

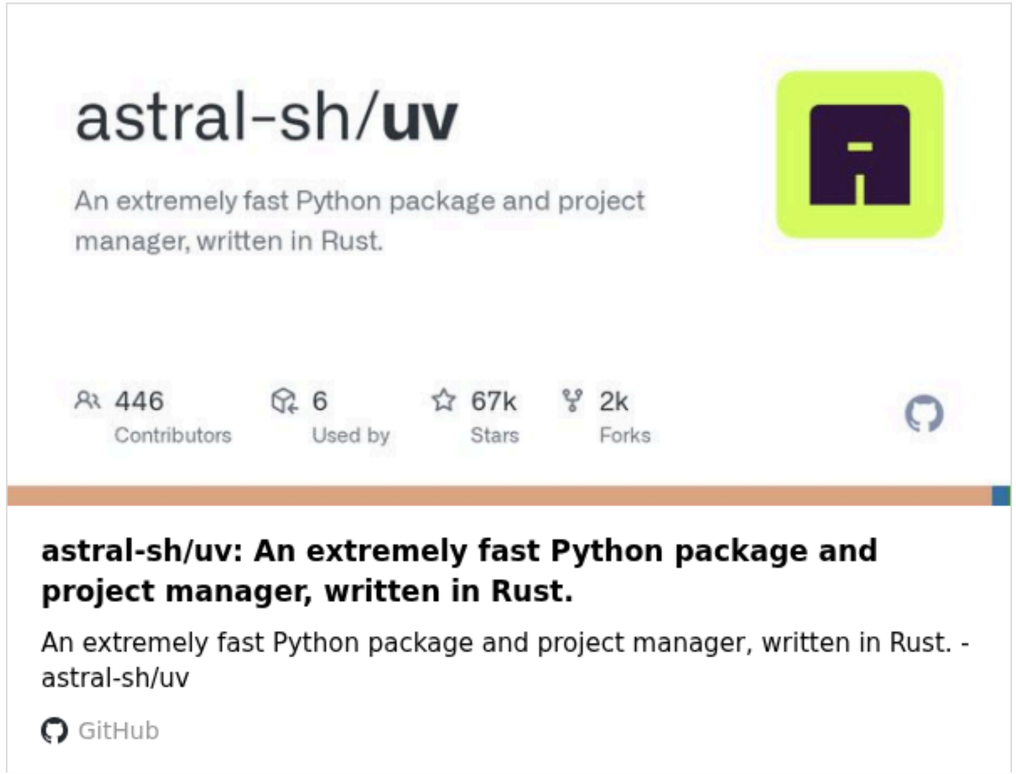


68.2K

Stars

"An extremely fast Python package and project manager, written in **Rust**. Replaces pip, pip-tools, pipx, poetry, pyenv, virtualenv, and more."





Install Python 3.12 from the the Astral python-build-standalone project

CLI

```
uv python install 3.12
uv init --python 3.12
uv venv
```

Create a virtual environment at .venv

Initialize a project, which creates:

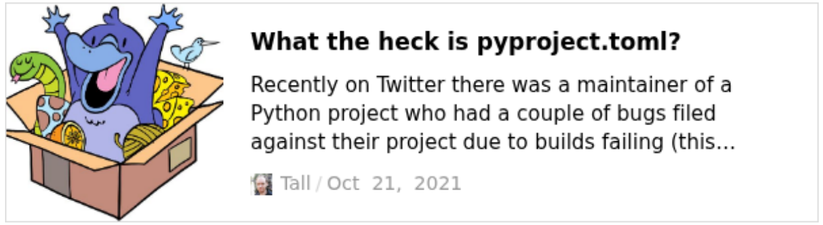
pyproject.toml (contains project's metadata)

```
[project]
name = "pydata-norwich-sep-2025"
version = "0.1.0"
description = "Add your description here"
readme = "README.md"
requires-python = ">=3.12"
dependencies = []
```

.python-version (contains project's default Python version that tells uv should use when creating virtual environment)

```
3.12
```

PYPROJECT.TOML



"the purpose of PEP 518 was to come up with a way for projects to specify what build tools they required … all projects should (eventually) have … development tools realized they … could put their own configuration … people liked the idea of centralizing configuration data in a single file"

- *Brett Cannon, Python Steering Council*

pyproject.toml

```toml
[project]
name = "pydata-norwich-sep-2025"
version = "0.1.0"
description = "Talk for Pydata Norwich Sep 2025"
readme = "README.md"
requires-python = ">=3.12"
dependencies = []
```

PYPROJECT.TOML

pyproject.toml

```toml
[project]
name = "pydata-norwich-sep-2025"
version = "0.1.0"
description = "Talk for Pydata Norwich Sep 2025"
readme = "README.md"
requires-python = ">=3.12"
dependencies = []

[tool.ruff]
line-length = 88
indent-width = 4
```

STRUCTURE

Three possible TOML tables:

- The [build-system] table … allows you to declare which build backend you use … to build your project
- The [project] table … to specify your project's basic metadata
- The [tool] table has tool-specific subtables, … contents are defined by each tool

Writing pyproject toml

+ PYPROJECT.TOML

pyproject.toml

```toml
[project]
name = "pydata-norwich-sep-2025"
version = "0.1.0"
description = "Talk for Pydata Norwich Sep 2025"
readme = "README.md"
requires-python = ">=3.12"
dependencies = []

[dependency-groups]

dev = ["ruff", "pre-commit"]
features = ["polars", "seaborn"]
training = ["spacy>=3"]
inference = ["pydantic>=2"]
```

ISOLAT DEPENDANCY

Update .venv with different options

CLI

```
uv sync --all-groups
uv sync --only-dev
uv sync --only-group inference
```

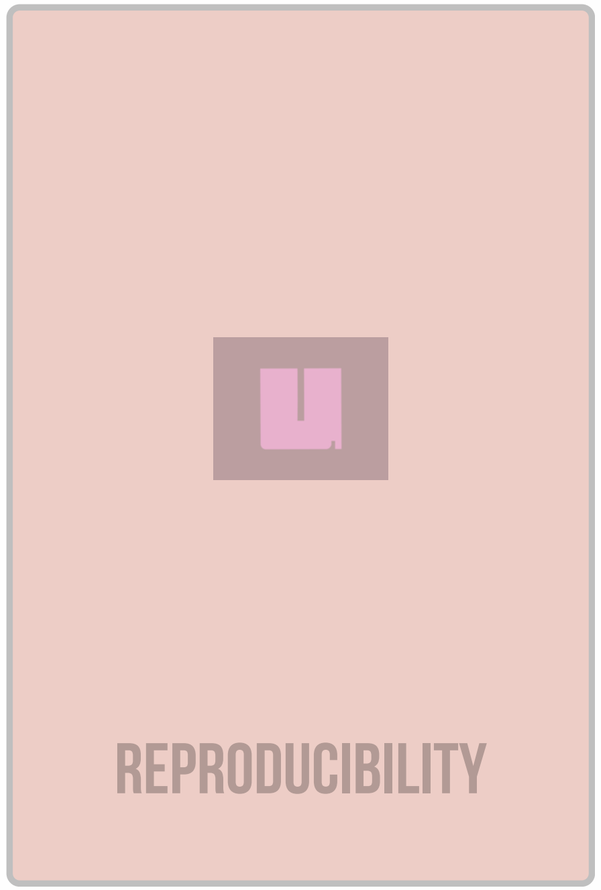
Export lockfile to another format

CLI

```
uv export --all-groups --output-file requirements.txt
uv export --only-group inference --output-file inference.txt
```


breathe

HOW?



Ruff

42.6K

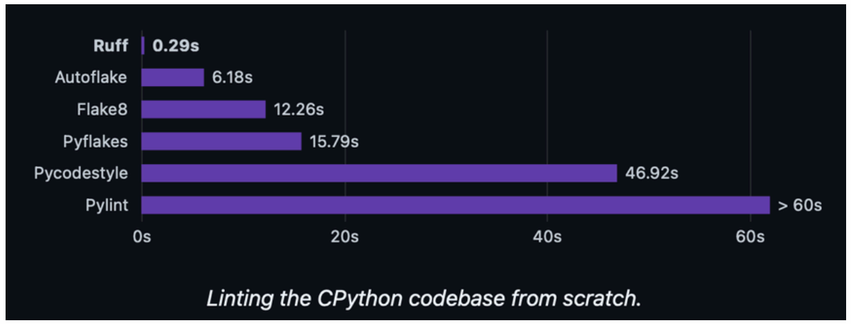
Stars

"An extremely fast Python **linter** and **formatter**, written in **Rust**. Replaces Flake8, isort, pydocstyle (plus dozens of plugins)."

astral-sh/ruff: An extremely fast Python linter and code formatter, written in...

An extremely fast Python linter and code formatter, written in Rust. - astral-sh/ruff

GitHub



Ruff INSTALL

OPTION 1

CLI

```
uv add ruff --dev
```

pyproject.toml

```
[dependency-groups]
dev = [
    "ruff>=0.13.2",
]
```

OPTION 2

pyproject.toml

```
[dependency-groups]
dev = [
    "ruff",
]
```

CLI

```
uv sync --only-dev
```

creates/updates

uv.lock

```
[package.metadata.requires-dev]
dev = [{ name = "ruff" }]

[[package]]
name = "ruff"
version = "0.13.2"
source = { registry = "https://pypi.org/simple" }
```

a cross-platform lockfile that contains exact information about dependencies. It should be checked into version control, allowing for consistent and reproducible installations across machines

Ruff LINTER

pyproject.toml

```toml
[tool.ruff.lint]
select = [
    "E",
    "F",
    "D",
    "I",
]
ignore = [
    "D100", "D101", "D102",
    "D103", "D104", "D105",
    "D106", "D107", "D206"
]

[tool.ruff.lint.pydocstyle]
convention = "google"
```

DEFINE

- E (**pycodestyle**): checks code against some of the style conventions in PEP 8
- F (**pyflakes**) checks code for errors
- D(**pydocstyle**): check compliance with most docstring conventions in PEP 257
- I (**isort**) sorts imports

ignore specific rules for pyflakes

settings for pydocstyle

CLI

```
uv run ruff check
uv run ruff check --fix
uv run ruff check path/to/code/
```

RUN

all files in the current directory (dry)
all files in the current directory (in-place)
a single file (in-place)

Ruff FORMATTER

pyproject.toml

```toml
[tool.ruff]
line-length = 88
indent-width = 4

[tool.ruff.format]
quote-style = "double"
indent-style = "space"
docstring-code-format = false
```

DEFINE

Drop-in replacement for Black (and beyond)
configuration

CLI

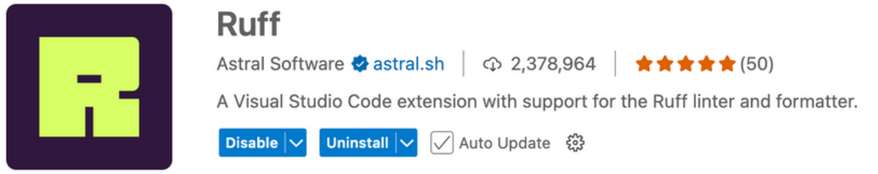
```
uv run ruff format
uv run ruff format path/to/code/

uv run ruff format path/to/file.py
uv run ruff format --check /path/to/file.py
```

RUN

all files in the current directory (in-place)
all files in `path/to/code` (in-place)
a single file (in-place)
a single file (dry)

VSCODE INTEGRATION



.vscode/settings.json

```
{
  "ruff.nativeServer": "on",
  "[python]": {
    "editor.formatOnSave": true,
    "editor.codeActionsOnSave": {
      "source.fixAll": "explicit",
      "source.organizeImports.ruff": "explicit"
    },
    "editor.defaultFormatter": "charliermarsh.ruff"
  },
  "notebook.formatOnSave.enabled": true,
  "notebook.codeActionsOnSave": {
    "notebook.source.fixAll": "explicit",
    "notebook.source.organizeImports": "explicit"
  }
}
```

DEFINE

Ruff can be **integrated** with various editors and IDEs

Install ruff extension for Visual Studio Code

Add configurations to **format**, **fix**, and **organize** imports on-save!

RUN

Ruff will **automatically execute** on save for Python or Jupyter Notebook files


breathe

HOW?

REPRODUCIBILITY

CONVENTIONS

CONSISTENCY

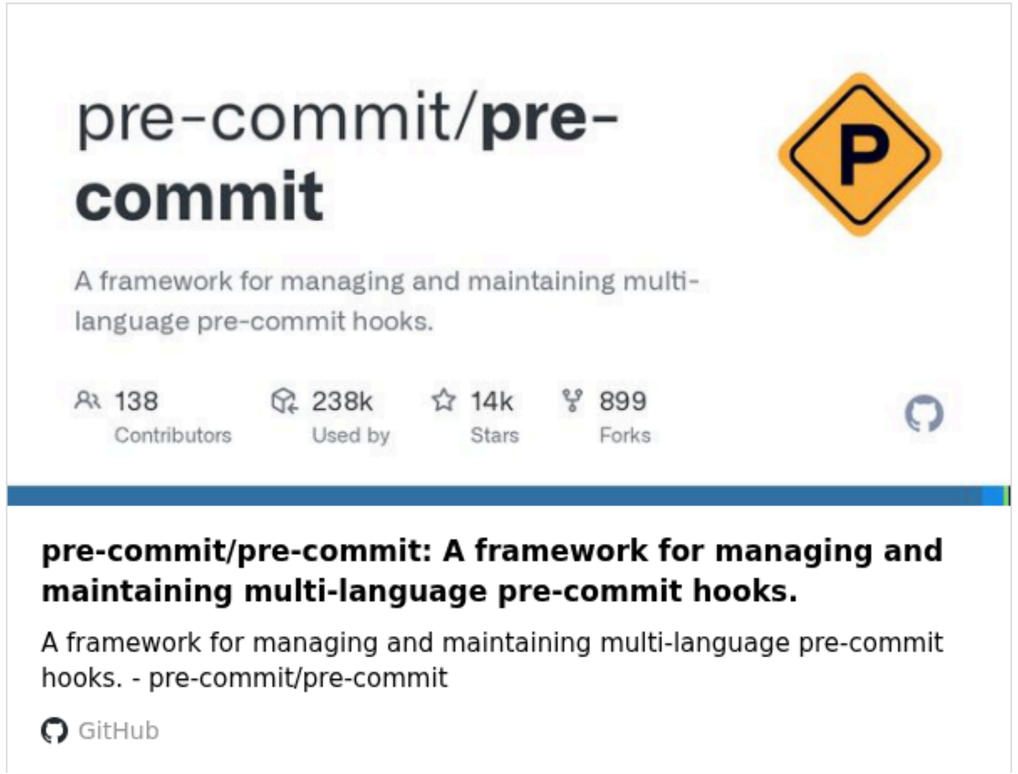
PRE-COMMIT

14.4K

Stars

"A framework for managing and maintaining multi-language **pre-commit hooks**."

"**Git hook scripts** are useful for identifying simple issues before submission to code review. By pointing these issues out before code review, this allows a code reviewer to focus on the architecture of a change while not wasting time with trivial style nitpicks."



INSTALL

OPTION 1

CLI

```
uv add pre-commit --dev
```

pyproject.toml

```
[dependency-groups]
dev = [
    "ruff>=0.13.2",
    "pre-commit>=4.3.0",

]
```

OPTION 2

pyproject.toml

```
[dependency-groups]
dev = [
    "ruff",
    "pre-commit",

]
```

CLI

```
uv sync --only-dev
```

creates/updates

uv.lock

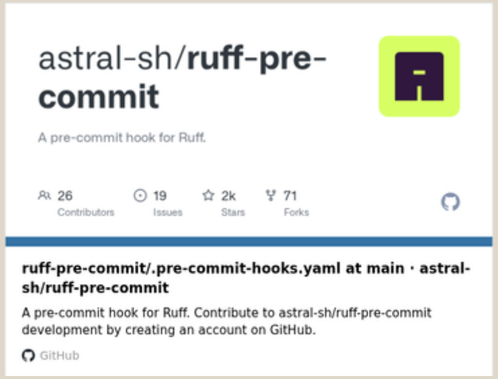
ADD CONFIGURATION

HOOK STRUCTURE

Add a. *.pre-commit-config.yaml* file with repository mapping to tell pre-commit where to get the code for the hook from

```
repos:
- repo: <link to a git package with a .pre-commit-hooks.yaml>
  rev: <package version>
  hooks:
  -   id: hook id found in .pre-commit-hooks.yaml
  -   arg: list of parameters to pass to the hoo  (default [])
```

HOOK FOR RUFF



.pre-commit-config.yaml

```
-   repo: https://github.com/astral-sh/ruff-pre-commit
    rev: 'v0.13.2'
    hooks:
    -   id: ruff-check
        types_or: [ python, pyi, jupyter ]
        args: [ --fix ]
    -   id: ruff-format
        types_or: [ python, pyi, jupyter ]
```

ADD CONFIGURATION

.pre-commit-config.yaml

Add out-of-the-box hooks from pre-commit:

- Checks whether the files parse as valid python
- Ensures that a file is either empty, or ends with one newline
- Trim trailing whitespace
- checks yaml files for parseable syntax
- checks toml files for parseable syntax
- checks toml files for parseable syntax

HOOK FOR PRE-COMMIT

```yaml
-   repo: https://github.com/pre-commit/pre-commit-hooks
    rev: 'v6.0.0'
    hooks:
    -   id: check-ast
    -   id: end-of-file-fixer
    -   id: trailing-whitespace
    -   id: check-yaml
    -   id: check-toml
    -   id: check-json
```

INSTALL HOOK SCRIPTS

RUN

Set up the git hook scripts locally

CLI

```
uv run pre-commit install
```

creates

.git/hooks/pre-commit

```
#!/usr/bin/env bash
# File generated by pre-commit: https://pre-commit.com
# ID: 138fd403232d2ddd5efb44317e38bf03

# start templated
INSTALL_PYTHON=/Users/gabrielharris/Documents/GitHub/DrGabrielH
arris/pydata-norwich-sep-2025/.venv/bin/python3
ARGS=(hook-impl --config=.pre-commit-config.yaml --hook-
type=pre-commit)
# end templated

HERE="$(cd "$(dirname "$0")" && pwd)"
ARGS+=(--hook-dir "$HERE" -- "$@")

if [ -x "$INSTALL_PYTHON" ]; then
    exec "$INSTALL_PYTHON" -mpre_commit "${ARGS[@]}"
elif command -v pre-commit > /dev/null; then
    exec pre-commit "${ARGS[@]}"
else
    echo '`pre-commit` not found.  Did you forget to activate
your virtualenv?' 1>&2
    exit
fi
```

Update hooks to latest version automatically (bring hooks to latest tag on default branch)

CLI

```
uv run pre-commit autoupdate
```

Run hooks against all files when adding new hooks (usually only run on changed files)

CLI

```
uv run pre-commit run --all-files
```

Now pre-commit will run automatically on **git commit**

```
> git commit -m "Update Readme"
[INFO] Initializing environment for https://github.com/astral-sh/ruff-pre-commit.
[INFO] Installing environment for https://github.com/astral-sh/ruff-pre-commit.
[INFO] Once installed this environment will be reused.
[INFO] This may take a few minutes...
ruff check.....(no files to check)Skipped
ruff format.....(no files to check)Skipped
check python ast.....(no files to check)Skipped
fix end of files.....Failed
- hook id: end-of-file-fixer
- exit code: 1
- files were modified by this hook

Fixing README.md

trim trailing whitespace.....Passed
check yaml.....(no files to check)Skipped
check toml.....(no files to check)Skipped
check json.....(no files to check)Skipped
```

Running hooks

Fix end of line

Hooks are either:

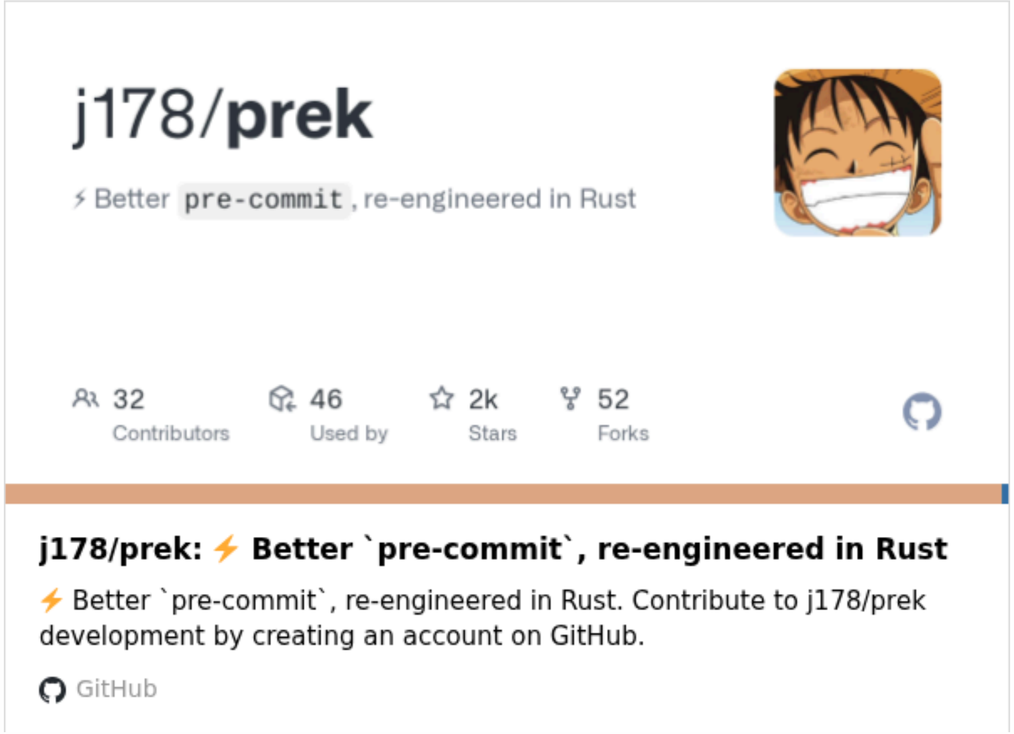
- Skipped: no files changes
- Passed: no violations
- Failed: violations > commit cancelled and fixes might be applied > new uncommented changes

PREK

1.6K

Stars

“prek is a reimagined version of pre-commit, built in **Rust**. It is designed to be a faster, dependency-free and drop-in alternative for it”



Thank you!