

Things you didn't know that you
thought you knew that... something

Henry Schreiner

Python 3.13

Largest change to CPython ever

```
def pi(trials: int) -> float:
    Ncirc = 0
    rand = random.Random()

    for _ in range(trials):
        x = rand.uniform(-1, 1)
        y = rand.uniform(-1, 1)

        test = x * x + y * y
        if test <= 1:
            Ncirc += 1

    return 4.0 * (Ncirc / trials)
```

Caveats:

"Free" thread safety gone*

(*Python has always supported concurrency)

Have to rebuild all extensions

Extensions have always been able to release the GIL

Threads	Time (s)
1	6.48
2	3.28
4	1.74

Running with 10M

```
def pi_in_threads(threads: int, trials: int) -> float:
    with ThreadPoolExecutor(max_workers=threads) as executor:
        return statistics.mean(executor.map(pi, [trials // threads] * threads))
```

Learn some Rust

Even if you don't use it (yet)

Packaging

Modern without legacy (C++23 - C++98)

(AKA CrowdStrike)

Memory Safety (compile time)

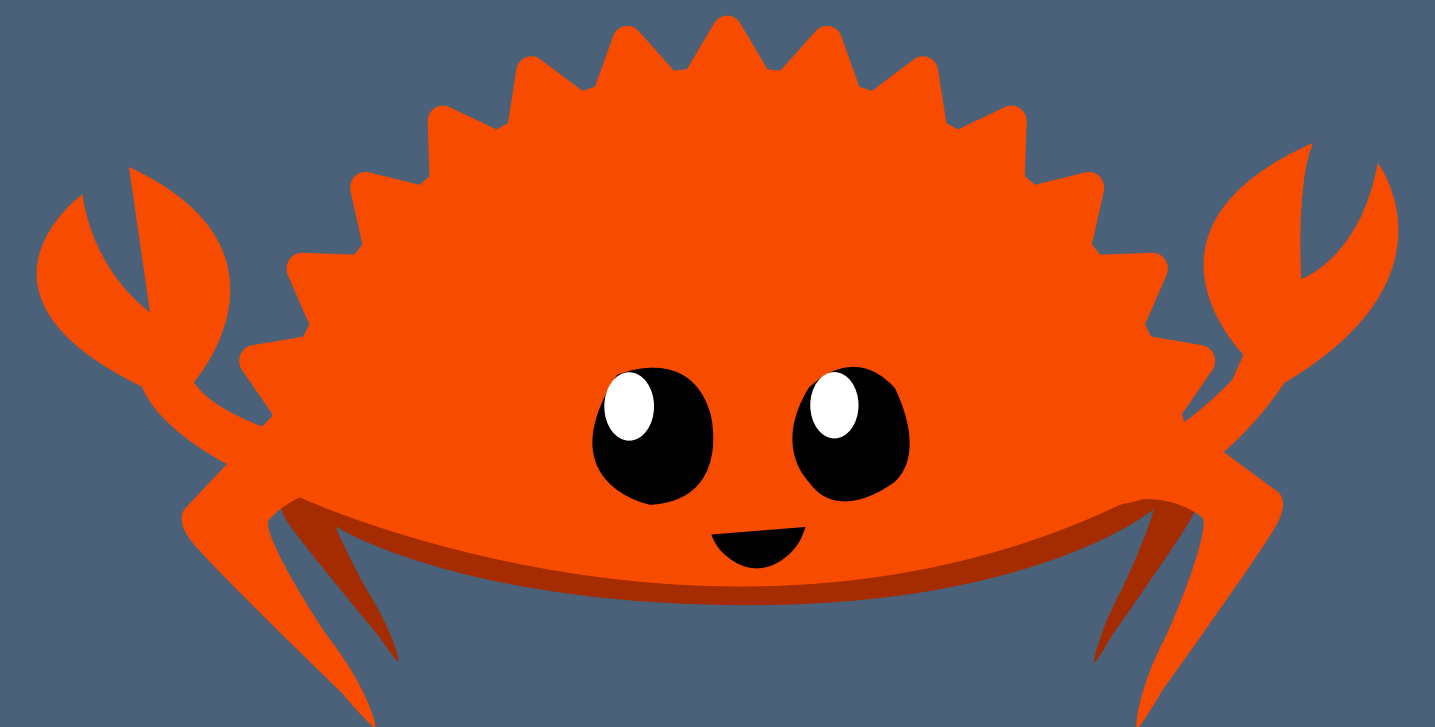
Explicit error handling

Great compiler messages

Traits

Declarative programming

Crabs are cuter than snakes



Some Rust projects

This is why it's interesting

BAT

Pretty cat replacement

PRESENTERM

Terminal presentations

PYPROJECT-FMT

Formatter for pyproject.toml

ZENITH

top/htop replacement

RUFF

Replacement for linters and formatters

10-100x faster

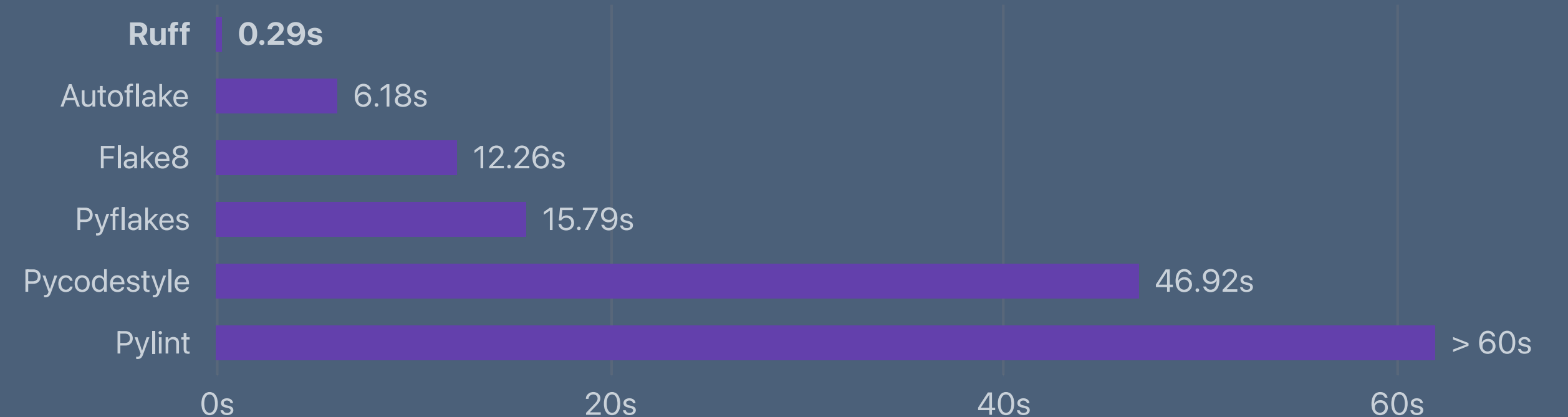
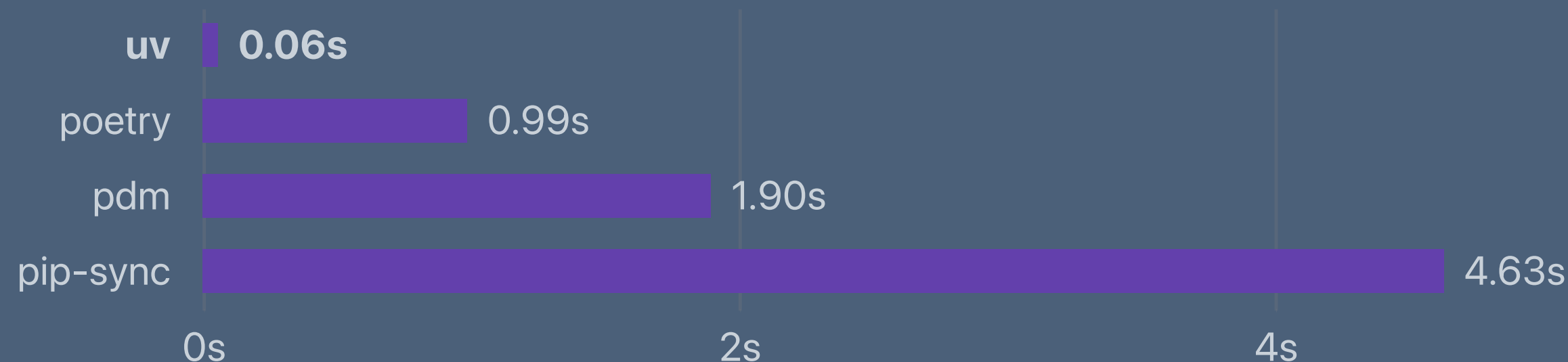
Single binary instead of 40+ packages

UV

Replacement for pip/venv/pip-tools

10-100x faster than pip & co.

Will be replacing more soon!



UV

Faster pip

```
python3 -m venv .venv
```

1.06 s

```
uv venv
```

18 ms

```
.venv/bin/pip install torch
```

Cold cache: 39.8 s

Warm cache: 21.3 s

```
uv pip install torch
```

Cold cache: 10.5 s

Warm cache: 358 ms

Practical examples:

scientific-python/cookie's CI went from 12 minutes -> 6 minutes.

nox's docs build went from 22 seconds -> 4 seconds.

Use a build system

Describe what you want, not how to get it

- **Modern** CMake isn't bad! (3.15+ at least, maybe newer)
- Or you can try Meson, build2, Bazel, etc.
- Integrates with IDEs, profilers, etc.
- Integrates with the Python builds system via scikit-build-core (or meson-python for meson)
- Integrates with OpenMP via brew on macOS* ;)
- Try using Ninja generator (more parallel than Make)

```
# pyproject.toml
[build-system]
requires = ["scikit-build-core"]
build-backend = "scikit_build_core.build"

[project]
name = "example"
version = "0.0.1"
```

```
# CMakeLists.txt
cmake_minimum_required(VERSION 3.15...3.30)
project(example LANGUAGES C)

find_package(
    Python REQUIRED COMPONENTS
    Interpreter Development.Module)

python_add_library(example MODULE WITH_SOABI
    example.c)

install(TARGETS example DESTINATION .)
```

* Requires a path on Apple Silicon because Homebrew moved

Pre-commit

Check and formatter framework

Make a list of checks and formatters to run

Supports over a dozen languages

Thousands of hooks

Ultra fast

Also supports git hooks (like pre-commit)

```
pre-commit run -a
check for added large files.....Passed
check for case conflicts.....Passed
check for merge conflicts.....Passed
check for broken symlinks.....(no files to check)Skipped
check yaml.....Passed
debug statements (python).....Passed
fix end of files.....Passed
mixed line ending.....Passed
python tests naming.....Passed
fix requirements.txt.....(no files to check)Skipped
trim trailing whitespace.....Passed
ruff.....Passed
ruff-format.....Passed
rst ``code`` is two backticks.....Passed
rst directives end with two colons.....Passed
rst ``inline code`` next to normal text.....Passed
blacken-docs.....Passed
cmake-format.....Passed
mypy.....Passed
check sdist.....Passed
codespell.....Passed
shellcheck.....Passed
Disallow improper capitalization.....Passed
Disallow expressions.....Passed
Cog the documentation.....Passed
Validate pyproject.toml.....Passed
Validate Dependabot Config (v2).....Passed
Validate GitHub Workflows.....Passed
Validate ReadTheDocs Config.....Passed
Validate JSON Schema files.....Passed
```


Pre-commit setup

.pre-commit-config.yaml

repos:

- repo: <https://github.com/pre-commit/pre-commit-hooks>

Repo with .pre-commit-hook.yaml

rev: '4.4.0'

Pinned for reproducibility

hooks:

- id: check-added-large-files
- id: check-case-conflict
- id: check-merge-conflict
- id: detect-private-key
- id: end-of-file-fixer
- id: trailing-whitespace

Repo can have any number of hooks

- repo: <https://github.com/astral-sh/ruff-pre-commit>

rev: v0.5.2

hooks:

- id: ruff
- args: ["--fix", "--show-fixes"]
- id: ruff-format

Can override any field (like args)

Pre-commit: other features

More good stuff

- Every check globally cached (fast)
- pre-commit install -> will run as git hook on all commits (-n to skip)
- pre-commit autoupdate —parallel -> will update all pins
- Local hooks - can disallow greps for content or files, etc.
- pre-commit.ci: free-for-open-source service
 - Update your pins weekly/monthly/quarterly
 - Auto-commit fixes to PRs
 - Ultra fast with global caching

Scientific Python Development Guide

One stop for all about Python packaging

Home

Tutorials

Topical Guides

Testing with pytest

Code coverage

Writing documentation

Simple packaging

Compiled packaging

Classic packaging

Style & static checks

Static type checking

GHA: GitHub Actions intro

GHA: Pure Python wheels

GHA: Binary wheels

Task runners

Principles

Patterns

Repo-Review

Scientific Python Library Development Guide

This guide is maintained by the scientific Python community for the benefit of fellow scientists and research software engineers.

Start at the basics. Do you have a pile of scientific Python scripts or Jupyter notebooks that are becoming unwieldy? Are changes to some parts of your code accidentally breaking other parts of your code? Do you want to more maintainable, reusable, and shareable form? Start at the [tutorial](#).

Learn recommended tools and best practices. [Topical guides](#) provide task-based instruction on topics that scientists and research software engineers may encounter as their projects evolve and grow. This covers modern packaging ([simple](#) or [compiled](#)), [style checking](#), [testing](#), [documentation](#), [static typing](#), [CI](#), and much more!

NEW PROJECT TEMPLATE

This guide comes with a [copier/cookiecutter/cruft](#) template for making new repos, [scientific-python/cookie](#). Eleven build backends including compiled backends, generation tested in Nox, and kept in-sync with the guide.

CHECKING AN EXISTING PROJECT

We provide [sp-repo-review](#), a set of [repo-review](#) checks for comparing your repository with the guidelines, runnable [right in the guide](#) via WebAssembly! All checks point to a linked badge in the guide.

Repo-Review

You can check the style of a GitHub repository below. Enter any repository, such as `scikit-hep/hist`, and the branch you want to check, such as `main` (it must exist). This will produce a list of results - green checkmarks mean this rule is followed, red errors mean the rule is not. A yellow warning sign means that the check was skipped because a previous required check failed. Some checks will fail, that's okay - the goal is bring all possible issues to your attention, not to force compliance with arbitrary checks.

You can also run [this tool](#) locally (Python 3.10+ required):

```
pipx run 'sp-repo-review[cli]' <path to repo>
```

Org/Repo

`scikit-hep/hist`

e.g. scikit-hep/hist

Branch

`main`

e.g. main



Results for scikit-hep/hist@main

General

Detected build backend: `hatchling.build`

Detected license(s): BSD License

- ✓ PY001: Has a pyproject.toml
- ✓ PY002: Has a README.(md|rst) file
- ✓ PY003: Has a LICENSE* file
- ✓ PY004: Has docs folder

<https://learn.scientific-python.org/development>

More info

Links to projects / tutorials



ISciNumPy.dev

A random collection of Science, Numerics, C++, CMake, and Python related topics.

Favorite posts and series

[C++ 11 14 17 20 23](#) • [macOS \(AS\) / Windows Setup](#) • [Azure DevOps \(Python Wheels\)](#) • [Conda-Forge ROOT](#) • [CLI11](#) • [GooFit](#) • [cibuildwheel](#) • [Hist](#) • [Python Bindings](#) • [Python 2→3 3.7 3.8 3.9 3.10 3.11 3.12 3.13](#) • [SSH](#)

My classes and books

[Modern CMake](#) • [CompClass](#) • [se-for-sci](#)

My workshops

[CMake Workshop](#) • [Python CPU, GPU, Compiled minicourses](#) • [Level Up Your Python](#) • [Packaging](#)

My projects

[pybind11 \(python_example, cmake_example, scikit_build_example\)](#) • [cibuildwheel](#) • [build](#) • [pipx](#) • [nox](#) • [pyproject-metadata](#) • [scikit-build \(core, cmake, ninja, moderncmakedomain\)](#) • [boost-histogram](#) • [Hist](#) • [UHI](#) • [Vector](#) • [GooFit](#) • [Particle DecayLanguage](#) • [Conda-Forge ROOT](#) • [Jekyll-Indico](#) • [uproot-browser](#) • [Scientific-Python/cookie](#) • [repo-review](#) • [CLI11](#) • [meson-python](#) • [Plumbum](#) • [validate-pyproject\(-schema-store\)](#) • [pytest GHA annotate-failures](#) • [flake8-errmsg](#) • [check-sdist](#) • [beautifulhugo](#) • [POVM](#) • [hypernewsviewer](#)

My sites

[Scientific-Python Development Guide](#) • [IRIS-HEP](#) • [Scikit-HEP](#) • [CLARIPHY](#)