Ruff

Ruff | pypi v0.7.0 | license MIT | python 3.7 | 3.8 | 3.9 | 3.10 | 3.11 | 3.12 | 3.13 | CI passing | Discord

Docs | **Playground**

An extremely fast Python linter and code formatter, written in Rust.

Linting the CPython codebase from scratch.

- ⊠ 10-100x faster than existing linters (like Flake8) and formatters (like Black)
- ⊠ Installable via `pip`
- ⊠ `pyproject.toml` support
- ⊠ Python 3.13 compatibility
- ⊠ Drop-in parity with Flake8, isort, and Black
- ⊠ Built-in caching, to avoid re-analyzing unchanged files
- ⊠ Fix support, for automatic error correction (e.g., automatically remove unused imports)
- ⊠ Over 800 built-in rules, with native re-implementations of popular Flake8 plugins, like flake8-bugbear
- ⊠ First-party editor integrations for VS Code and more
- ⊠ Monorepo-friendly, with hierarchical and cascading configuration

Ruff aims to be orders of magnitude faster than alternative tools while integrating more functionality behind a single, common interface.

Ruff can be used to replace Flake8 (plus dozens of plugins), Black, isort, pydocstyle, pyupgrade, autoflake, and more, all while executing tens or hundreds of times faster than any individual tool.

Ruff is extremely actively developed and used in major open-source projects like:

- Apache Airflow
- Apache Superset
- FastAPI
- Hugging Face
- Pandas
- SciPy

…and many more.

Ruff is backed by Astral. Read the launch post, or the original project announcement.

Testimonials

Sebastián Ramírez, creator of FastAPI:

> Ruff is so fast that sometimes I add an intentional bug in the code just to confirm it's actually running and checking the code.

Nick Schrock, founder of Elementl, co-creator of GraphQL:

> Why is Ruff a gamechanger? Primarily because it is nearly 1000x faster. Literally. Not a typo. On our largest module (dagster itself, 250k LOC) pylint takes about 2.5 minutes, parallelized across 4 cores on my M1. Running ruff against our *entire* codebase takes .4 seconds.

Bryan Van de Ven, co-creator of Bokeh, original author of Conda:

> Ruff is ~150-200x faster than flake8 on my machine, scanning the whole repo takes ~0.2s instead of ~20s. This is an enormous quality of life improvement for local dev. It's fast enough that I added it as an actual commit hook, which is terrific.

Timothy Crosley, creator of isort:

> Just switched my first project to Ruff. Only one downside so far: it's so fast I couldn't believe it was working till I intentionally introduced some errors.

Tim Abbott, lead developer of Zulip:

> This is just ridiculously fast… `ruff` is amazing.

Table of Contents

For more, see the documentation.

Getting Started

For more, see the documentation.

Installation

Ruff is available as `ruff` on PyPI:

```
# With pip.
pip install ruff

# With pipx.
pipx install ruff
```

Starting with version `0.5.0`, Ruff can be installed with our standalone installers:

```
# On macOS and Linux.
curl -LsSf https://astral.sh/ruff/install.sh | sh

# On Windows.
powershell -c "irm https://astral.sh/ruff/install.ps1 | iex"

# For a specific version.
curl -LsSf https://astral.sh/ruff/0.7.0/install.sh | sh
powershell -c "irm https://astral.sh/ruff/0.7.0/install.ps1 | iex"
```

You can also install Ruff via Homebrew, Conda, and with a variety of other package managers.

Usage

To run Ruff as a linter, try any of the following:

```
ruff check                          # Lint all files in the current directory (and any subdirectories).
ruff check path/to/code/            # Lint all files in `/path/to/code` (and any subdirectories).
ruff check path/to/code/*.py        # Lint all `.py` files in `/path/to/code`.
ruff check path/to/code/to/file.py  # Lint `file.py`.
ruff check @arguments.txt           # Lint using an input file, treating its contents as newline-delimited command-line
     arguments.
```

Or, to run Ruff as a formatter:

```
ruff format                          # Format all files in the current directory (and any subdirectories).
ruff format path/to/code/            # Format all files in `/path/to/code` (and any subdirectories).
ruff format path/to/code/*.py        # Format all `.py` files in `/path/to/code`.
ruff format path/to/code/to/file.py  # Format `file.py`.
ruff format @arguments.txt           # Format using an input file, treating its contents as newline-delimited command-line
     arguments.
```

Ruff can also be used as a pre-commit hook via `ruff-pre-commit`:

```
- repo: https://github.com/astral-sh/ruff-pre-commit
  # Ruff version.
  rev: v0.7.0
  hooks:
    # Run the linter.
    - id: ruff
      args: [ --fix ]
    # Run the formatter.
    - id: ruff-format
```

Ruff can also be used as a VS Code extension or with various other editors.

Ruff can also be used as a GitHub Action via `ruff-action`:

```
name: Ruff
on: [ push, pull_request ]
jobs:
  ruff:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: astral-sh/ruff-action@v1
```

Configuration

Ruff can be configured through a `pyproject.toml`, `ruff.toml`, or `.ruff.toml` file (see: *Configuration*, or *Settings* for a complete list of all configuration options).

If left unspecified, Ruff's default configuration is equivalent to the following `ruff.toml` file:

```
# Exclude a variety of commonly ignored directories.
exclude = [
    ".bzr",
    ".direnv",
    ".eggs",
    ".git",
    ".git-rewrite",
    ".hg",
    ".ipynb_checkpoints",
    ".mypy_cache",
    ".nox",
    ".pants.d",
    ".pyenv",
    ".pytest_cache",
    ".pytype",
    ".ruff_cache",
    ".svn",
    ".tox",
    ".venv",
    ".vscode",
    "__pypackages__",
    "_build",
    "buck-out",
    "build",
    "dist",
    "node_modules",
    "site-packages",
    "venv",
]

# Same as Black.
line-length = 88
indent-width = 4

# Assume Python 3.8
target-version = "py38"

[lint]
# Enable Pyflakes (`F`) and a subset of the pycodestyle (`E`)  codes by default.
select = ["E4", "E7", "E9", "F"]
ignore = []

# Allow fix for all enabled rules (when `--fix`) is provided.
fixable = ["ALL"]
unfixable = []

# Allow unused variables when underscore-prefixed.
dummy-variable-rgx = "^(_+|(_+[a-zA-Z0-9_]*[a-zA-Z0-9]+?))$"

[format]
# Like Black, use double quotes for strings.
quote-style = "double"

# Like Black, indent with spaces, rather than tabs.
indent-style = "space"

# Like Black, respect magic trailing commas.
skip-magic-trailing-comma = false
```

```
# Like Black, automatically detect the appropriate line ending.
line-ending = "auto"
```

Note that, in a `pyproject.toml`, each section header should be prefixed with `tool.ruff`. For example, `[lint]` should be replaced with `[tool.ruff.lint]`.

Some configuration options can be provided via dedicated command-line arguments, such as those related to rule enablement and disablement, file discovery, and logging level:

```
ruff check --select F401 --select F403 --quiet
```

The remaining configuration options can be provided through a catch-all `--config` argument:

```
ruff check --config "lint.per-file-ignores = {'some_file.py' = ['F841']}"
```

To opt in to the latest lint rules, formatter style changes, interface updates, and more, enable preview mode by setting `preview = true` in your configuration file or passing `--preview` on the command line. Preview mode enables a collection of unstable features that may change prior to stabilization.

See `ruff help` for more on Ruff's top-level commands, or `ruff help check` and `ruff help format` for more on the linting and formatting commands, respectively.

Rules

Ruff supports over 800 lint rules, many of which are inspired by popular tools like Flake8, isort, pyupgrade, and others. Regardless of the rule's origin, Ruff re-implements every rule in Rust as a first-party feature.

By default, Ruff enables Flake8's `F` rules, along with a subset of the `E` rules, omitting any stylistic rules that overlap with the use of a formatter, like `ruff format` or Black.

If you're just getting started with Ruff, **the default rule set is a great place to start**: it catches a wide variety of common errors (like unused imports) with zero configuration.

Beyond the defaults, Ruff re-implements some of the most popular Flake8 plugins and related code quality tools, including:

- autoflake
- eradicate
- flake8-2020
- flake8-annotations
- flake8-async
- flake8-bandit (#1646)
- flake8-blind-except

- flake8-boolean-trap
- flake8-bugbear
- flake8-builtins
- flake8-commas
- flake8-comprehensions
- flake8-copyright
- flake8-datetimez
- flake8-debugger
- flake8-django
- flake8-docstrings
- flake8-eradicate
- flake8-errmsg
- flake8-executable
- flake8-future-annotations
- flake8-gettext
- flake8-implicit-str-concat
- flake8-import-conventions
- flake8-logging
- flake8-logging-format
- flake8-no-pep420
- flake8-pie
- flake8-print
- flake8-pyi
- flake8-pytest-style
- flake8-quotes
- flake8-raise
- flake8-return
- flake8-self
- flake8-simplify
- flake8-slots
- flake8-super
- flake8-tidy-imports
- flake8-todos
- flake8-type-checking
- flake8-use-pathlib
- flynt (#2102)
- isort
- mccabe

- pandas-vet
- pep8-naming
- pydocstyle
- pygrep-hooks
- pylint-airflow
- pyupgrade
- tryceratops
- yesqa

For a complete enumeration of the supported rules, see *Rules*.

Contributing

Contributions are welcome and highly appreciated. To get started, check out the **contributing guidelines**.

You can also join us on **Discord**.

Support

Having trouble? Check out the existing issues on **GitHub**, or feel free to **open a new one**.

You can also ask for help on **Discord**.

Acknowledgements

Ruff's linter draws on both the APIs and implementation details of many other tools in the Python ecosystem, especially Flake8, Pyflakes, pycodestyle, pydocstyle, pyupgrade, and isort.

In some cases, Ruff includes a "direct" Rust port of the corresponding tool. We're grateful to the maintainers of these tools for their work, and for all the value they've provided to the Python community.

Ruff's formatter is built on a fork of Rome's `rome_formatter`, and again draws on both API and implementation details from Rome, Prettier, and Black.

Ruff's import resolver is based on the import resolution algorithm from Pyright.

Ruff is also influenced by a number of tools outside the Python ecosystem, like Clippy and ESLint.

Ruff is the beneficiary of a large number of contributors.

Ruff is released under the MIT license.

Who’s Using Ruff?

Ruff is used by a number of major open-source projects and companies, including:

- Albumentations
- Amazon (AWS SAM)
- Anthropic (Python SDK)
- Apache Airflow
- AstraZeneca (Magnus)
- Babel
- Benchling (Refac)
- Bokeh
- Cryptography (PyCA)
- CERN (Indico)
- DVC
- Dagger
- Dagster
- Databricks (MLflow)
- Dify
- FastAPI
- Godot
- Gradio
- Great Expectations
- HTTPX
- Hatch
- Home Assistant
- Hugging Face (Transformers, Datasets, Diffusers)
- IBM (Qiskit)
- ING Bank (popmon, probatus)
- Ibis
- ivy
- Jupyter
- Kraken Tech
- LangChain
- Litestar
- LlamaIndex
- Matrix (Synapse)
- MegaLinter

- Meltano (Meltano CLI, Singer SDK)
- Microsoft (Semantic Kernel, ONNX Runtime, LightGBM)
- Modern Treasury (Python SDK)
- Mozilla (Firefox)
- Mypy
- Nautobot
- Netflix (Dispatch)
- Neon
- Nokia
- NoneBot
- NumPyro
- ONNX
- OpenBB
- Open Wine Components
- PDM
- PaddlePaddle
- Pandas
- Pillow
- Poetry
- Polars
- PostHog
- Prefect (Python SDK, Marvin)
- PyInstaller
- PyMC
- PyMC-Marketing
- pytest
- PyTorch
- Pydantic
- Pylint
- PyVista
- Reflex
- River
- Rippling
- Robyn
- Saleor
- Scale AI (Launch SDK)
- SciPy
- Snowflake (SnowCLI)

- Sphinx
- Stable Baselines3
- Starlette
- Streamlit
- The Algorithms
- Vega-Altair
- WordPress (Openverse)
- ZenML
- Zulip
- build (PyPA)
- cibuildwheel (PyPA)
- delta-rs
- featuretools
- meson-python
- nox
- pip

Show Your Support

If you're using Ruff, consider adding the Ruff badge to your project's `README.md`:

```
[![Ruff](https://img.shields.io/endpoint?url=https://raw.githubusercontent.com/astral-sh/ruff/main/assets/badge/v2.json)](
    https://github.com/astral-sh/ruff)
```

…or `README.rst`:

```
.. image:: https://img.shields.io/endpoint?url=https://raw.githubusercontent.com/astral-sh/ruff/main/assets/badge/v2.json
    :target: https://github.com/astral-sh/ruff
    :alt: Ruff
```

…or, as HTML:

```
<a href="https://github.com/astral-sh/ruff"><img src="https://img.shields.io/endpoint?url=https://raw.githubusercontent.com/
    astral-sh/ruff/main/assets/badge/v2.json" alt="Ruff" style="max-width:100%;"></a>
```

License

This repository is licensed under the MIT License