Ruff Python lint and format

START HERE

`ruff check`
Lint the project

`ruff check --select I --fix`
Sort imports

`ruff check --fix`
Lint and auto-fix

`ruff rule <code>`
Explain a rule

`ruff format`
Format the project

`ruff check --add-noqa`
Silence existing hits

`ruff format --check`
Verify formatting (CI)

`ruff check --watch`
Re-lint on save

Config pyproject.toml / ruff.toml → `select rules` → **ruff check** violations → `--fix` → **ruff format** layout → `--check in CI` → **Clean tree** commit

CONCEPTS

Two tools in one binary: check lints (and fixes), format rewrites layout. Run both; they do not overlap.

Since 0.17 the default is a broad curated set (F, E4/E7/E9, B, UP, SIM, RUF, PL and more). See it with ruff check --show-settings --isolated; select replaces it, extend-select adds.

Ruff finds the nearest config upward from each file. --isolated ignores all config files.

CHECK

`ruff check`
Lint current directory

`ruff check <path>...`
Lint given files or dirs

`ruff check --fix`
Apply safe fixes

`ruff check --diff`
Preview fixes as a diff
Writes nothing; exit 0 when there is no diff.

`ruff check --unsafe-fixes --fix`
Also apply unsafe fixes
Unsafe fixes may change behaviour; review the diff.

`ruff check --show-fixes`
List what was fixed

`ruff check --fix-only`
Fix, do not report leftovers

`ruff check --watch`
Re-run on file changes

`ruff check -`
Lint code from stdin

`ruff check --exit-non-zero-on-fix`
Fail if fixes were applied

FORMAT

`ruff format`
Format current directory

`ruff format <path>...`
Format given files or dirs

`ruff format --check`
Fail if anything would change

`ruff format --diff`
Show changes, write nothing

FORMAT (cont.)

`ruff format --line-length <n>`
Override line length

`ruff format -`
Format stdin to stdout

Format suppression comments

Comment	Effect
`# fmt: off / # fmt: on`	skip a region
`# fmt: skip`	skip one statement

RULE SELECTION

`--select <codes>`
Enable only these rules
Replaces the default set; codes may be prefixes like E, F, I.

`--extend-select <codes>`
Add rules to the current set

`--ignore <codes>`
Disable rules

`--select ALL`
Enable every rule
Noisy and changes with each release; prefer an explicit list.

`--per-file-ignores <glob>:<code>`
Ignore rules per path

`--fixable <codes> / --unfixable <codes>`
Restrict which rules autofix

`--target-version <py3xx>`
Minimum Python to assume

`--statistics`
Count violations per rule

`ruff check --select I --fix`
Sort imports (isort rules)

`ruff check --select UP --fix`
Modernise syntax (pyupgrade)

Popular rule prefixes

Prefix	From
`E / W`	pycodestyle
`F`	Pyflakes
`I`	isort
`B`	bugbear
`UP`	pyupgrade
`SIM`	simplify
`N`	pep8-naming
`C4`	comprehensions

SUPPRESSING

`# noqa: <code>`
Ignore a rule on this line

`# noqa`
Ignore all rules on this line
Blanket noqa hides future rules too; name the code.

SUPPRESSING (cont.)

`# ruff: noqa: <code>`
Ignore a rule for the file

`ruff check --add-noqa`
Insert noqa for all hits

A # noqa that no longer matches a hit is flagged by RUF100 once that rule is enabled; --fix removes it.

EXPLORE

`ruff rule <code>`
Explain one rule

`ruff linter`
List supported linters

`ruff config`
List all settings

`ruff config <option>`
Describe one setting

`ruff check --show-settings <file>`
Show effective settings

`ruff check --show-files`
List files Ruff will scan

`ruff version`
Print the version

OUTPUT AND FILES

`--output-format <fmt>`
Choose report format
concise, full, json, github, gitlab, sarif, junit, and more.

`--exclude <pattern>`
Replace excluded paths

`--extend-exclude <pattern>`
Exclude more paths

`--force-exclude`
Honor excludes for explicit paths
Needed when a hook passes file names directly.

`--config <file|key=value>`
Use a file or override a setting

`--isolated`
Ignore all config files

CONFIG

pyproject.toml basics

```
[tool.ruff]
line-length = 100
target-version = "py312"
extend-exclude = ["migrations"]

[tool.ruff.lint]
select = ["E4", "E7", "E9", "F",
"I", "B"]
ignore = ["B008"]
```

CONFIG (cont.)

Per-file ignores

```
[tool.ruff.lint.per-file-
ignores]
"tests/**" = ["S101"]
"__init__.py" = ["F401"]
```

Formatter options

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

ruff.toml (no tool. prefix)

```
line-length = 100

[lint]
extend-select = ["I"]

[format]
quote-style = "single"
```

In ruff.toml and .ruff.toml drop the tool.ruff prefix: [lint] and [format] are top-level tables.

EDITORS AND CI

`ruff server`
Run the language server

pre-commit hook

```
repos:
- repo: https://github.com/
astral-sh/ruff-pre-commit
  rev: <tag>
  hooks:
  - id: ruff-check
    args: [--fix]
  - id: ruff-format
```

GitHub Actions step

```
- run: pip install ruff
- run: ruff check --output-
format=github
- run: ruff format --check
```

RECIPES

Adopt Ruff on an old codebase

```
ruff check --statistics
ruff check --add-noqa
ruff format
# then tighten rules gradually
```

Format and fix before commit

```
ruff check --fix
ruff format
git diff --stat
```

Try a rule set without editing config

```
ruff check --extend-select B,SIM
--statistics
ruff check --extend-select B --
diff
```

CI gate

```
ruff check --output-
format=concise
ruff format --check --diff
```