**Engineer**

# Ruff

🖌 швидкий лінтер і форматувач для Python

## Що це таке

Ruff — це одна програма для командного рядка, написана на Rust, яка звіряє вихідний код Python із каталогом правил лінтингу й форматує його в єдиному послідовному стилі. За ним стоїть Astral, компанія, що створила uv і ty, а щоденну роботу виконують дві підкоманди: `ruff check` — це лінтер, а `ruff format` — форматувач. Ruff заявляє, що працює в 10–100 разів швидше за Flake8 і Black, зберігає кеш, щоб не аналізувати незмінені файли вдруге, і може замінити Flake8 із десятками його плагінів, Black, isort, pydocstyle, pyupgrade і autoflake.

Лінтер має понад 900 правил, і чимало з них — власні переписані версії популярних плагінів Flake8. Кожне правило має код із короткого префікса й трьох цифр, а префікс каже, звідки правило походить: F401, невживаний імпорт, узято з Pyflakes, а E501, задовгий рядок, — із pycodestyle. Конфігурація обирає правила за цими кодами, поодинці або за префіксом, тож `select = ["E", "F"]` вмикає кожне правило, чий код починається з E чи F, а особливий селектор `ALL` вмикає все. Багато правил мають виправлення, і їх поділено на безпечні, які мають зберегти поведінку під час виконання, і небезпечні, які можуть її змінити; `ruff check --fix` застосовує лише безпечні, якщо не додати ще й `--unsafe-fixes`.

Форматувач задумано як пряму заміну Black. На таких кодових базах, як Django і Zulip, понад 99,9% рядків виходять ідентичними, хоча документація прямо каже, що ці два інструменти не призначені для того, щоб перемикатися між ними туди й назад на тому самому коді. Як і Black, він пропускає ділянки між `# fmt: off` і `# fmt: on`.

Обидві половини читають ту саму конфігурацію, яка лежить у таблиці `[tool.ruff]` файлу `pyproject.toml` або у файлі `ruff.toml` чи `.ruff.toml`. Кожен файл Python перевіряють за найближчим файлом конфігурації над ним, і цей файл застосовують цілком: Ruff нічого не підмішує з батьківських тек, якщо найближчий файл не вкаже на батьківський через `extend`. Параметри з командного рядка мають перевагу над будь-яким файлом. Якщо нічого не задано, рядок може мати 88 символів, як у Black, а версію Python, з якої виходять правила, беруть із поля `requires-python` проєкту, коли жодна конфігурація її не називає.

Застосуйте поняття до рішення

Застосуйте «Що таке Ruff?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Одна команда веде ідею від ескізу до продукту, у який ваші клієнти вже можуть увійти, а не перекидає роботу між трьома підрядниками.

**Переглянути наші послуги** Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Збільшила дохід GIS-застосунку в 10 разів — 10x зростання виторгу завдяки картографічному продукту

**Збільшила дохід GIS-застосунку в 10 разів** Зв'язатися з нами →

## Чому це важливо

Швидкість вирішує, де перевірці дозволять працювати. Лінтер, повільний настільки, що перериває людину, витісняють у безперервну інтеграцію, де він звітує вже тоді, коли автор узявся за інше. Той, що відповідає швидко, може працювати в редакторі й у commit-хуку, поки зміна ще свіжа.

Об’єднання знімає суперечки між інструментами. Проєкт, який лінтить через Flake8 із плагінами, сортує імпорти через isort і форматує через Black, мусить тримати узгодженими кілька конфігурацій і кілька зафіксованих версій. Ruff дає одну конфігурацію, одну версію, яку треба зафіксувати, і одну нумерацію правил, а ще публікує перелік правил лінтера, що конфліктують із його власним форматувачем, щоб ці два інструменти можна було узгодити свідомо.

Коди виходу роблять Ruff перевіркою без обгортки. `ruff check` завершується з кодом 0, коли порушень не лишилося або всі виправлено, з кодом 1, коли порушення лишаються, і з кодом 2, коли конфігурація чи командний рядок недійсні або сам Ruff дав збій, так само як ESLint і Prettier.

`ruff format --check` завершується з кодом 1, коли хоч один файл було б переформатовано. Правило або обране, або вимкнене, і будь-яке обране правило, що спрацювало, ставить код виходу 1.

Ціна в тому, що набір правил — це рішення, яке хтось має взяти на себе. Власна порада Ruff — перелічувати правила явно через `select`, а не нашаровувати `extend-select`, починати з E і F, додавати по одній групі за раз, а `ALL` вмикати обачно, бо він приносить нові правила з кожним оновленням.

## Як це виглядає на практиці

Ruff відповідав за Python у планці нуля попереджень для шести мов, де кожна мова мала власний лінтер. Кожна діагностика вважалася помилкою, а inline-придушення були заборонені, і для Ruff це виключає коментар `# noqa`: знахідку в Python можна було виправити в коді й більше ніяк. Хуки pre-commit і pre-push запускали лінтери й тести, тож перевірка відбувалася ще до того, як з'являвся коміт.

На зафіксованому етапі в жовтні 2026 року Ruff на цьому сайті мав менший обсяг роботи: 22 допоміжні скрипти поряд зі збіркою Hugo. Сторінка Подяки називає Ruff серед інструментів, що підтримують вихідний код у порядку. Набір правил у `pyproject.toml` виміряно, а не обрано. Кожну з 54 родин правил Ruff прогнали на цих скриптах 15 жовтня 2026 року, 39 із них нічого не повідомили, і ті мовчазні родини, чия тема — саме такий код, додали до початкових одинадцяти, тож тепер обрано 33. Родина безпеки, S, лишається ввімкненою, а п'ять її правил ігноровано за кодом, кожне з коментарем про те, чому позначений тут шаблон є безпечним двійником небезпечного, тож решта правил цієї родини й далі може зупинити коміт. Три специфічні для Ruff правила про неоднозначні символи Unicode ігноровано, бо текст сайту свідомо вживає справжні довгі й короткі тире та данські літери.

Конфігурація pre-commit запускає `ruff check` і `ruff format --check` на кожному файлі Python у `scripts/`, якого торкається коміт, а задача `fix` у Taskfile проєкту запускає ті самі дві команди в режимі, що виправляє. Поряд із Ruff стоїть basedpyright, перевіряч типів, із коментарем, що проводить межу між ними: лінтер не стежить за значенням через межу функції, і під час першого запуску перевіряч типів знайшов два справжні дефекти, які Ruff пропустив як чисті.

# Де виникають проблеми

Здебільшого клопіт із Ruff походить від конфігурації, яка буквальніша за людей, що її писали.

**Оновлення ламає гілку, якої ніхто не чіпав.** Код, що вчора проходив, після оновлення Ruff повідомляє нові порушення. Звична причина — `select = ["ALL"]`, що підхоплює кожне правило, яке додає випуск, разом із версією, яку ніхто не зафіксував. Зафіксуйте версію Ruff, щоб оновлення було комітом, який хтось переглядає, і перелічіть потрібні проєкту родини замість `ALL`, щоб нова родина з'являлася лише тоді, коли її хтось додасть.

**Одна тека перестає дотримуватися правил проєкту.** Порушення, які коренева конфігурація забороняє, проходять у підтеці, або там спрацьовує правило, яке розробники вимкнули. У цій теці лежить `ruff.toml` чи `pyproject.toml` із таблицею `[tool.ruff]`, і оскільки Ruff застосовує найближчий файл цілком, жодне кореневе налаштування нижче від нього вже не діє. `ruff check --show-settings` для файлу в цій теці друкує шлях до налаштувань, які він підхопив. Видаліть вкладений файл або зробіть так, щоб він успадковував налаштування через `extend = "../pyproject.toml"`.

**Придушення переживає свою причину.** Коментар `# noqa` з'явився з поважної причини, код змінився, і тепер коментар не ховає нічого або ховає наступну проблему в цьому рядку. Голий `# noqa` ще гірший: він заглушує всі правила в рядку. Оберіть RUF100, що повідомляє про коментарі `noqa`, які вже нічого не придушують, і PGH004, що вимагає конкретного коду, і дайте `--fix` прибрати мертві. Тут імпорт відтоді почали використовувати, тож коментар нічого не захищає, і RUF100 про нього повідомляє:

```
import json  # noqa: F401

print(json.dumps({"ok": True}))
```

**Форматувач і лінтер розходяться.** `ruff format` переписує файл, а `ruff check` потім позначає результат. Деякі правила лінтера стежать за тим, що вирішує форматувач, як-от кінцеві коми (COM812), відступи (W191, E111) і стиль лапок (правила Q), а довжини рядка форматувач дотримується лише в міру можливого, тож E501 може спрацювати й на відформатованому коді. Читайте попередження, які друкує `ruff format`, бо запуск без них — це

власний тест документації на те, що інструменти узгоджені, ігноруйте кожне правило, яке називає попередження, а E501 сприймайте як сигнал перебудувати рядок.

**Масове виправлення змінює поведінку.** Після `ruff check --fix --unsafe-fixes` по всьому дереву падають тести або обробник помилок перестає ловити те, що ловив раніше. Небезпечним виправленням дозволено змінювати поведінку під час виконання або прибирати коментарі, а власний приклад документації змінює, який виняток кидає фрагмент коду. Тримайте `--unsafe-fixes` поза хуками й скриптами, застосовуйте небезпечні виправлення по одному правилу через `--select` і читайте diff, а правило переводьте в безпечні через `extend-safe-fixes` лише тоді, коли розробники його перевірили.

## Що почитати далі

- Ruff — що Ruff замінює, наскільки швидким себе заявляє і хто його підтримує.
- The Ruff Linter — вибір правил і їхній пріоритет, безпечні й небезпечні виправлення, коментарі `noqa` і коди виходу.
- The Ruff Formatter — сумісність із Black і правила лінтера, що конфліктують із форматуванням.
- Rules — кожне правило за кодом, його походження і чи можна виправити його автоматично.
- Configuring Ruff — як Ruff знаходить файли, правило найближчого файлу, `extend` і параметри командного рядка.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

**Дивитися, скільки це коштує** Зв'язатися з нами →

# Зробіть термінологію практичною

Запитайте, як «Що таке Ruff?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

**Переглянути наші послуги**

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

**Дбайливо зроблено з любов'ю — D.**

**CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N**

**Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.**

**Надруковано з https://engineer.company/uk/glossary/ruff/**