Engineer

Ruff

🖌 en hurtig linter og formatter til Python

Hvad det er

Ruff er et enkelt kommandolinjeprogram, skrevet i Rust, der holder Python-kildekode op mod et katalog af lintregler og omskriver dens layout til én ensartet stil. Bag det står Astral, firmaet bag uv og ty, og to af underkommandoerne klarer det daglige arbejde: `ruff check` er linteren, og `ruff format` er formatteren. Ruff hævder at køre 10 til 100 gange hurtigere end Flake8 og Black, gemmer en cache, så uændrede filer ikke bliver analyseret igen, og kan træde i stedet for Flake8 med snesevis af plugins, Black, isort, pydocstyle, pyupgrade og autoflake.

Linteren har over 900 regler, mange af dem genskrevet fra populære Flake8-plugins. Hver regel har en kode af et kort præfiks og tre cifre, og præfikset fortæller, hvor reglen kommer fra: F401, en ubrugt import, stammer fra Pyflakes, og E501, en linje der er for lang, fra pycodestyle. En konfiguration vælger regler ud fra koderne, én ad gangen eller efter præfiks, så `select = ["E", "F"]` slår hver regel til, hvis kode begynder med E eller F, og den særlige vælger `ALL` slår alt til. Mange regler har en rettelse med, delt op i sikre rettelser, der skal bevare adfærden ved kørsel, og usikre, der kan ændre den; `ruff check --fix` anvender kun den sikre slags, medmindre `--unsafe-fixes` også er angivet.

Formatteren er bygget som en direkte erstatning for Black. På kodebaser som Django og Zulip kommer over 99,9 % af linjerne ud identisk, men dokumentationen siger ligeud, at de to ikke er beregnet til at blive skiftet frem og tilbage på den samme kode. Ligesom Black springer den områder over, der er indrammet af `# fmt: off` og `# fmt: on`.

Begge halvdele læser den samme konfiguration, som ligger under `[tool.ruff]` i `pyproject.toml` eller i en `ruff.toml`- eller `.ruff.toml`-fil. Hver Python-fil tjekkes mod den nærmeste konfigurationsfil over den, og den fil bruges hel: Ruff fletter

intet ind fra overliggende mapper, medmindre den nærmeste fil peger på sin forælder med `extend`. Indstillinger på kommandolinjen går forud for alle filer. Uden indstillinger må en linje være 88 tegn, som i Black, og den Python-version, reglerne går ud fra, hentes fra projektets `requires-python`-felt, når ingen konfiguration angiver en.

Brug begrebet i en beslutning

Brug »Hvad er Ruff?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Ét team, der tager en idé fra skitse til noget, jeres kunder kan logge ind på, i stedet for at kaste den mellem tre leverandører.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har ført en GIS-kortapp til 10x omsætningsvækst — 10x omsætningsvækst, drevet af kortproduktet

Har ført en GIS-kortapp til 10x omsætningsvækst Kontakt os →

Hvorfor det betyder noget

Hastighed afgør, hvor et tjek får lov at køre. En linter, der er så langsom, at den afbryder folk, bliver skubbet ud i continuous integration, hvor den melder sig, efter at forfatteren er gået videre. En, der svarer hurtigt, kan køre i editoren og i en commit-hook, mens ændringen stadig er frisk.

Samling fjerner uenighed mellem værktøjer. Et projekt, der linter med Flake8 og plugins, sorterer imports med isort og formaterer med Black, skal holde flere konfigurationer og versionslåse i takt. Ruff giver én konfiguration, én version at låse og én nummerering af regler, og projektet offentliggør listen over de lintregler, der strides med dets egen formatter, så de to kan bringes til at stemme overens med vilje.

Exitkoderne gør Ruff til en kvalitetsport uden indpakning. `ruff check` afslutter med 0, når der ikke er nogen overtrædelser tilbage, eller alle blev rettet, med 1, når der stadig er overtrædelser, og med 2, når konfigurationen eller kommandolinjen er ugyldig, eller Ruff selv går ned, ligesom ESLint og Prettier gør. `ruff format --check` afslutter med 1, når en fil ville blive omformateret. En regel er enten valgt eller slået fra, og enhver valgt regel, der slår ud, sætter exitkoden til 1.

Prisen er, at regelsættet er en beslutning, som nogen skal eje. Ruffs egne råd er at liste reglerne eksplicit med `select` i stedet for at stable `extend-select`, at starte med E og F og tilføje én gruppe ad gangen og at bruge `ALL` med

omtanke, fordi den slår nye regler til ved hver opgradering.

Sådan ser det ud i praksis

Ruff dækkede Python i en zero-warnings-standard på tværs af seks sprog, hvor hvert sprog havde sin egen linter. Hver diagnostik talte som en fejl, og inline-undertrykkelser var forbudt, hvilket for Ruff udelukker `# noqa`-kommentaren: et fund i Python kunne rettes i koden og på ingen anden måde. Pre-commit- og pre-push-hooks kørte linterne og testene, så tjekket kørte, før en commit fandtes.

I den dokumenterede værktøjsopsætning fra oktober 2026 havde Ruff en mindre opgave på dette site: de 22 hjælpescripts ved siden af Hugo-buildet. Kolofonen nævner Ruff blandt de værktøjer, der holder kilden ærlig. Regelsættet i `pyproject.toml` er målt frem for valgt. Hver af Ruffs 54 regelfamilier blev den 15. oktober 2026 kørt mod scriptene, 39 af dem meldte intet, og de tavse familier, hvis emne er netop denne kode, blev føjet til de oprindelige elleve, så udvalget nu tæller 33. Sikkerhedsfamilien, S, er slået til med fem regler ignoreret efter kode, hver med en kommentar om, hvorfor det mønster, der slår ud her, er den sikre tvilling til et farligt, så familiens øvrige regler stadig kan stoppe en commit. Tre Ruff-specifikke regler om tvetydige Unicode-tegn er ignoreret, fordi sitets tekst med vilje bruger rigtige tankestreger og danske bogstaver.

Pre-commit-konfigurationen kører `ruff check` og `ruff format --check` på hver Python-fil under `scripts/`, som en commit rører, og projektets fix-task kører de samme to kommandoer i deres rettende form. Ved siden af Ruff står basedpyright, en typechecker, med en kommentar, der trækker grænsen mellem dem: en linter følger ikke en værdi på tværs af en funktionsgrænse, og ved sin første kørsel fandt typecheckeren to reelle fejl, som Ruff havde godkendt.

Hvor det går galt

Det meste besvær med Ruff kommer af en konfiguration, der er mere bogstavelig end de mennesker, der skrev den.

En opgradering får en urørt branch til at fejle. Kode, der bestod i går, melder nye overtrædelser, efter at Ruff er blevet opdateret. Den sædvanlige årsag er `select = ["ALL"]`, der tager hver regel med, som en udgivelse tilføjer, sammen med en version, der flyder. Lås Ruff-versionen, så en opgradering er en commit, som nogen reviewer, og list de familier, projektet vil have, i stedet for `ALL`, så en ny familie først kommer med, når nogen tilføjer den.

Én mappe holder op med at følge projektets regler. Overtrædelser, som rodkonfigurationen forbyder, slipper igennem i en undermappe, eller en regel, teamet slog fra, slår ud dér. En `ruff.toml` eller en `pyproject.toml` med en `[tool.ruff]`-tabel ligger i mappen, og fordi Ruff bruger den nærmeste fil hel, holder alle rodindstillinger op med at gælde under den. `ruff check --show-settings` på en fil dér udskriver stien til de indstillinger, den endte med. Slet den indlejrede fil, eller lad den arve med `extend = "../pyproject.toml"`.

En undertrykkelse overlever sin grund. En `# noqa`-kommentar kom ind af en god grund, koden ændrede sig, og nu skjuler kommentaren ingenting eller det næste problem på linjen. En bar `# noqa` er værre: den gør alle regler på linjen tavse. Vælg RUF100, der melder `noqa`-kommentarer, som ikke længere undertrykker noget, og PGH004, der kræver en bestemt kode, og lad `--fix` fjerne de døde. Her er importen siden kommet i brug, så kommentaren beskytter ingenting, og RUF100 melder den:

```
import json  # noqa: F401

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

Formatteren og linteren er uenige. `ruff format` omskriver en fil, og `ruff check` markerer bagefter resultatet. Nogle lintregler kontrollerer det, formatteren bestemmer, som afsluttende kommaer (COM812), indrykning (W191, E111) og anførselstegn (Q-reglerne), og formatteren gør kun et bedste forsøg på linjelængden, så E501 kan stadig slå ud på formateret kode. Læs de advarsler, `ruff format` skriver ud, for en kørsel uden dem er dokumentationens egen prøve på, at de to er enige, ignorér enhver regel, en advarsel nævner, og tag E501 som et signal til at bygge linjen om.

En samlet rettelse ændrer adfærd. Efter `ruff check --fix --unsafe-fixes` over hele træet fejler testene, eller en fejlhåndtering holder op med at fange det, den plejede. Usikre rettelser må gerne ændre adfærden ved kørsel eller fjerne kommentarer, og dokumentationens eget eksempel ændrer, hvilken undtagelse et stykke kode kaster. Hold `--unsafe-fixes` ude af hooks og scripts, anvend usikre rettelser én regel ad gangen med `--select` og læs diffen, og forfrem kun en regel med `extend-safe-fixes`, når teamet har tjekket den.

Læs mere

- Ruff — hvad Ruff erstatter, hvor hurtigt det hævder at være, og hvem der vedligeholder det.
- The Ruff Linter — regelvalg og forrang, sikre og usikre rettelser, `noqa`-kommentarer og exitkoder.
- The Ruff Formatter — kompatibilitet med Black og de lintregler, der strides med formateringen.
- Rules — hver regel efter kode, dens oprindelse, og om den kan rettes automatisk.
- Configuring Ruff — hvordan filerne findes, reglen om den nærmeste fil, `extend` og kommandolinjens indstillinger.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er Ruff?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/ruff/