Engineer

Ruff

🖌 en rask linter og formaterer for Python

Hva det er

Ruff er ett enkelt kommandolinjeprogram, skrevet i Rust, som sjekker Python-kildekode mot en katalog av lintregler og skriver om oppsettet av koden til én konsekvent stil. Det støttes av Astral, selskapet bak uv og ty, og to av underkommandoene gjør det daglige arbeidet: `ruff check` er linteren og `ruff format` er formatereren. Prosjektet hevder at Ruff kjører 10 til 100 ganger raskere enn Flake8 og Black, programmet husker resultatene sine slik at uendrede filer ikke analyseres på nytt, og det kan tre inn i stedet for Flake8 med dusinvis av programtilleggene dens, Black, isort, pydocstyle, pyupgrade og autoflake.

Linteren leveres med mer enn 900 regler, mange av dem egne omskrivninger av populære Flake8-tillegg. Hver regel har en kode laget av et kort prefiks og tre sifre, og prefikset sier hvor regelen kom fra: F401, en import som ikke brukes, kommer fra Pyflakes, og E501, en linje som er for lang, fra pycodestyle. En konfigurasjon velger regler etter disse kodene, én om gangen eller etter prefiks, så `select = ["E", "F"]` slår på hver regel med en kode som begynner på E eller F, og den spesielle velgeren `ALL` slår på alt. Mange regler har en retting, sortert i trygge rettinger, som skal bevare oppførselen under kjøring, og utrygge, som kan endre den; `ruff check --fix` bruker bare den trygge typen med mindre `--unsafe-fixes` også er gitt.

Formatereren er laget for å kunne erstatte Black direkte. På kodebaser som de til Django og Zulip kommer mer enn 99,9 % av linjene ut identiske, selv om dokumentasjonen sier rett ut at de to ikke er ment å byttes fram og tilbake på den samme koden. I likhet med Black hopper den over områder som er avgrenset med `# fmt: off` og `# fmt: on`.

Begge halvdelene leser den samme konfigurasjonen, som ligger under `[tool.ruff]` i `pyproject.toml`, eller i en fil som heter `ruff.toml` eller `.ruff.toml`. Hver Python-fil sjekkes mot den nærmeste konfigurasjonsfilen over den, og den filen brukes i sin helhet: Ruff slår ikke sammen noe fra overordnede kataloger med mindre den nærmeste filen navngir forelderen sin med `extend`. Valg som gis på kommandolinjen, overstyrer hver fil. Står ingenting oppgitt, kan linjer være opptil 88 tegn lange, som i Black, og Python-versjonen reglene forutsetter, hentes fra feltet `requires-python` i prosjektet når ingen konfigurasjon navngir en.

Bruk begrepet i en beslutning

Bruk «Hva er Ruff?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Ett team som tar en idé fra skisse til noe kundene dine kan logge inn på, i stedet for at den sendes videre mellom tre leverandører.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Ledet en GIS-kartapp til 10 ganger mer omsetning — 10x inntektsvekst, ledet fra kartproduktet

Ledet en GIS-kartapp til 10 ganger mer omsetning Ta kontakt →

Hvorfor det er viktig

Hastighet avgjør hvor en sjekk får lov til å kjøre. En linter som er treg nok til å avbryte noen, blir skjøvet ut til kontinuerlig integrasjon, der den rapporterer etter at forfatteren har gått videre. En som svarer raskt, kan kjøre i editoren og i en hook ved commit, mens endringen fortsatt er fersk.

Å samle alt i ett verktøy fjerner krangel mellom verktøy. Et prosjekt som linter med Flake8 og programtillegg, sorterer importer med isort og formaterer med Black, må holde flere konfigurasjoner og versjonslåser i takt. Ruff tilbyr én konfigurasjon, én versjon å låse og én nummerering av regler, og publiserer listen over lintregler som kommer i konflikt med Ruffs egen formaterer, slik at de to kan bringes i samsvar med vilje.

Avslutningskodene gjør Ruff til en kvalitetsport uten noe skript rundt. `ruff check` avslutter med 0 når det ikke er noen brudd igjen, eller når hvert av dem ble rettet, med 1 når brudd gjenstår, og med 2 når konfigurasjonen eller kommandolinjen er ugyldig eller Ruff selv feiler, slik ESLint og Prettier også gjør. `ruff format --check` avslutter med 1 når en fil ville blitt formatert om. En regel er enten valgt eller av, og hver valgt regel som slår ut, setter avslutningskoden til 1.

Kostnaden er at regelsettet er en beslutning noen må eie. Ruff råder selv til å liste regler eksplisitt med `select` i stedet for å stable `extend-select`, til å begynne med E og F og legge til én gruppe om gangen, og til å bruke `ALL` med forsiktighet, fordi den velgeren slår på nye regler ved hver oppgradering.

Slik ser det ut i praksis

Ruff dekket Python i et krav om null advarsler på seks språk, der hvert språk hadde sin egen linter. Hvert funn telte som en feil, og undertrykkelser i selve koden var forbudt, noe som for Ruff utelukker kommentaren `# noqa`: et funn i Python kunne bare rettes i koden og på ingen annen måte. Pre-commit- og pre-push-hooks kjørte linterne og testene, så sjekken kjørte før en commit fantes.

I det dokumenterte verktøyoppsettet fra oktober 2026 hadde Ruff en mindre jobb på dette nettstedet: de 22 hjelpeskriptene ved siden av Hugo-bygget. Kolofonen nevner Ruff blant verktøyene som holder kildekoden ærlig. Regelsettet i `pyproject.toml` er målt og ikke valgt. Den 15. oktober 2026 ble hver av de 54 regelfamiliene i Ruff kjørt mot disse skriptene, og 39 av dem rapporterte ingenting. De tause familiene som faktisk handler om denne typen kode, ble lagt til de opprinnelige elleve, og det gir et utvalg på 33. Sikkerhetsfamilien, S, står på, med fem regler ignorert etter kode, hver ved siden av en kommentar som forklarer hvorfor mønsteret som flagges her, er den trygge tvillingen til et farlig mønster, slik at de andre reglene i familien fortsatt er en del av kvalitetsporten. Tre Ruff-spesifikke regler om tvetydige Unicode-tegn er ignorert, fordi teksten på nettstedet bruker ekte lange og korte tankestreker og danske bokstaver med vilje.

Pre-commit-konfigurasjonen kjører `ruff check` og `ruff format --check` på hver Python-fil under `scripts/` som en commit rører, og oppgaven i prosjektet som retter alt som kan rettes automatisk, kjører de samme to kommandoene i den formen som retter. Ved siden av Ruff står basedpyright, en typesjekker, med en kommentar som markerer grensen mellom dem: en linter følger ikke en verdi over en funksjonsgrense, og ved første kjøring fant typesjekkeren to reelle programvarefeil som Ruff hadde sluppet gjennom.

Hvor det går galt

Det meste av trøbbelet med Ruff kommer av en konfigurasjon som er mer bokstavelig enn folkene som skrev den.

En oppgradering feiler en gren ingen har rørt. Kode som besto i går, rapporterer nye brudd etter at Ruff er oppdatert. Den vanlige årsaken er `select = ["ALL"]`, som tar inn hver regel en utgivelse legger til, sammen med en versjon som ikke er låst. Lås Ruff-versjonen slik at en oppgradering blir en commit noen har sett gjennom, og list opp familiene prosjektet vil ha i stedet for `ALL`, så en ny familie kommer først når noen legger den til.

Én mappe slutter å følge prosjektets regler. Brudd som rotkonfigurasjonen forbyr, slipper gjennom i en underkatalog, eller en regel teamet har slått av, slår ut der. En `ruff.toml`, eller en `pyproject.toml` med en `[tool.ruff]`-tabell, ligger i den mappen, og fordi Ruff bruker den nærmeste filen i sin helhet, slutter hver innstilling fra roten å gjelde under den. `ruff check --show-settings` på en fil der skriver ut stien til innstillingene Ruff kom fram til. Slett den nestede filen, eller la den arve med `extend = "../pyproject.toml"`.

En undertrykkelse lever lenger enn grunnen til den. En `# noqa`-kommentar ble lagt inn av en god grunn, koden endret seg, og kommentaren skjuler nå ingenting, eller det neste problemet på den linjen. En naken `# noqa` er verre: den demper hver regel på linjen. Velg RUF100, som rapporterer `noqa`-kommentarer som ikke lenger undertrykker noe, og PGH004, som krever en bestemt kode, og la `--fix` fjerne de døde. Her har importen siden blitt tatt i bruk, så kommentaren beskytter ingenting, og RUF100 rapporterer den:

```
import json  # noqa: F401

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

Formatereren og linteren er uenige. `ruff format` skriver om en fil, og `ruff check` flagger deretter resultatet. Noen lintregler passer på ting formatereren bestemmer, som avsluttende kommaer (COM812), innrykk (W191, E111) og stil på anførselstegn (Q-reglene), og formatereren gjør bare et forsøk etter beste evne på linjelengden, så E501 kan fortsatt slå ut på formatert kode. Les advarslene `ruff format` skriver ut, siden en kjøring uten dem er dokumentasjonens egen test på at de to er enige, ignorer hver regel en advarsel navngir, og behandle E501 som et signal om å bygge om linjen.

En masseretting endrer oppførsel. Etter `ruff check --fix --unsafe-fixes` over hele treet feiler tester, eller en feilhåndterer slutter å fange det den pleide å fange. Utrygge rettinger har lov til å endre oppførselen under kjøring eller fjerne

kommentarer, og dokumentasjonens eget eksempel endrer hvilket unntak et stykke kode kaster. Hold `--unsafe-fixes` ute av hooks og skript, bruk utrygge rettinger én regel om gangen med `--select` og les diffen, og flytt en regel over med `extend-safe-fixes` først når teamet har sjekket den.

Les mer

- Ruff — hva Ruff erstatter, hvor raskt prosjektet hevder at det er, og hvem som vedlikeholder det.
- The Ruff Linter — valg av regler og rekkefølgen de gjelder i, trygge og utrygge rettinger, `noqa` -kommentarer og avslutningskoder.
- The Ruff Formatter — samsvaret med Black, og lintreglene som kommer i konflikt med formateringen.
- Rules — hver regel etter kode, hvor den kommer fra, og om den kan rettes automatisk.
- Configuring Ruff — hvordan konfigurasjonsfiler blir funnet, regelen om den nærmeste filen, `extend` og valgene på kommandolinjen.

Ordliste

Velg ut fra deres faktiske rammer

Beskriv arbeidsflyten, eksisterende grensesnitt og brukerne som trenger endringen. Vi kan definere en testbar leveranse og en overlevering som gjør teamet i stand til å vedlikeholde løsningen.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er Ruff?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/ruff/