Engineer

Ruff

🖌 Python向けの高速なリンター兼フォーマッター

どのようなものか

Ruffは、Rustで書かれた単一のコマンドラインプログラムです。Pythonのソースコードをリンターのルール集に照らして検査し、書式を一貫したスタイルに整えます。uvとtyを開発する企業Astralが支援しており、日常の作業は二つのサブコマンドが担います。`ruff check`がリンター、`ruff format`がフォーマッターです。Flake8とBlackより10から100倍速いと謳い、変更のないファイルを再解析しないためのキャッシュを保持します。数十のプラグインを含むFlake8、Black、isort、pydocstyle、pyupgrade、autoflakeの代わりにもなれます。

リンターには900を超えるルールが付属し、多くは人気のFlake8プラグインをネイティブ実装で書き直したものです。各ルールには、短い接頭辞と3桁の数字からなるコードがあり、接頭辞は由来を示します。未使用のインポートを表すF401はPyflakes、長すぎる行を表すE501はpycodestyleに由来します。設定では、一つずつか接頭辞単位で、コードからルールを選びます。たとえば`select = ["E", "F"]`はEかFで始まるすべてを有効にし、特別なセレクター`ALL`はすべてを有効にします。修正を備えるルールも多く、実行時の動作を保つための安全な修正と、動作が変わり得る危険な修正に分かれます。`ruff check --fix`は、`--unsafe-fixes`も渡さない限り、安全な修正だけを適用します。

フォーマッターは、Blackをそのまま置き換えられるよう設計されています。DjangoやZulipなどのコードベースでは、99.9%を超える行が同じ結果になります。ただし文書は、同じコードに対して両者を交互に使うことは想定していないと明記しています。Blackと同様、`# fmt: off`と`# fmt: on`で囲んだ部分は処理しません。

二つの機能は同じ設定を読みます。設定は`[tool.ruff]`として`pyproject.toml`内に置くか、`ruff.toml`か`.ruff.toml`に置きます。各Pythonファイルは、それより上にある最も近い設定ファイルを使って検査され、そのファイル全体が採用されます。近い方が`extend`で親を指定しない限り、Ruffは上位ディレクトリーの

設定を何も統合しません。コマンドラインの指定は、すべての設定ファイルより優先されます。設定を省くと、Blackと同じ88文字が行の長さとなり、どの設定にもPythonのバージョンが指定されていなければ、ルールが前提とするバージョンはプロジェクトの `requires-python` から得ます。

概念を判断に生かす

「Ruffとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 3社の間で受け渡す代わりに、1つのチームがアイデアのスケッチから、お客様がログインして使える製品まで担当します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 GIS地図アプリの開発を主導し売上を10倍に — 10x 地図製品の開発を主導して実現した売上成長

GIS地図アプリの開発を主導し売上を10倍に お問い合わせ →

なぜ重要なのか

速度によって、検査を実行できる場所が決まります。作業を中断させるほど遅いリンターは継続的インテグレーションへ追いやられ、作成者の関心が別の作業へ移った後に報告します。すぐに答えるリンターなら、変更がまだ記憶に新しいうちに、エディターやコミットのフックで動かせます。

統合すると、ツール同士の食い違いもなくせます。Flake8とプラグインで検査し、isortでインポートを並べ、Blackで整形するプロジェクトは、複数の設定と固定バージョンを整合させなければなりません。Ruffは、設定を一つ、固定するバージョンを一つ、ルールの番号体系を一つにまとめます。また、自身のフォーマッターと競合するリンターのルールを公開しているため、意図して両方を整合させられます。

終了コードによって、ラッパーなしでも品質ゲートにできます。 `ruff check` は、違反がないか、すべて修正できれば0、違反が残れば1、設定やコマンドラインが不正か、Ruff自体が失敗すれば2で終了します。ESLintやPrettierと同じです。 `ruff format --check` は、整形が必要なファイルが一つでもあれば1で終了します。ルールは、選択されているか、無効かのどちらかです。選択したルールへの違反が残ると、終了コードは1になります。

代償は、誰かが責任を持ってルールを決めなければならないことです。Ruff自身は、 `select` で明示的に列挙して、 `extend-select` を積み重ねるのを避け、EとFから始め、一度に一群ずつ加えるよう勧めています。 `ALL` は更新のたびに新しいルールも有効にするため、慎重に使うよう案内しています。

実際の活用例

六つの言語に共通の警告ゼロの基準を設けた作業では、言語ごとに専用のリンターを置き、RuffがPythonを担当しました。すべての診断をエラーとし、インラインの抑制を禁止したため、Ruffでは `# noqa` コメントも使えません。Pythonの指摘には、コードで直す以外の対処を認めませんでした。コミット前とプッシュ前のフックがリンターとテストを実行し、コミットができる前に検査します。

記録された2026年10月のツール構成では、このサイトはRuffに、Hugoのビルドを補助する22本のスクリプトという、より小さな仕事を任せていました。クレジットのページも、ソースの正しさを保つ道具にRuffを挙げています。`pyproject.toml` のルール群は、好みではなく測定で決めています。2026-10-15に、Ruffの54のルール群をすべてそのスクリプトに実行したところ、39群は何も報告しませんでした。そのうち、対象がこのコードに関係する群を、最初の11群に加え、選択は33群になりました。セキュリティーのS群は有効のまま、5ルールをルールコードで指定して除外しています。それぞれの横には、ここで指摘される形が危険な形に似ていても安全である理由をコメントに書き、同じ群の他のルールは引き続きゲートにしています。曖昧なUnicode文字を扱うRuff独自の三つのルールも除外しています。このサイトは、本物のemダッシュ、enダッシュ、デンマーク語の文字を意図して使うためです。

pre-commitの設定は、`ruff check` と `ruff format --check` を、コミットで変更する `scripts/` 以下のPythonファイルに実行します。プロジェクトの修正用タスクも、同じ二つのコマンドを、修正を行う形で実行します。Ruffの隣には型検査器basedpyrightがあり、コメントで役割の境界を示しています。リンターは、関数の境界を越えて値を追跡しません。型検査器は初回の実行で、Ruffが問題なしとした実際の不具合を二つ見つけました。

問題が起きるところ

Ruffの問題の大半は、Ruffが、書いた人の意図よりも設定を文字どおりに解釈することで生じます。

触っていないブランチが、更新後に失敗する。 昨日通ったコードが、Ruffの更新で新しい違反を報告します。よくある原因は、リリースごとに加わるルールをすべて取り込む `select = ["ALL"]` と、固定していないバージョンの組み合わせです。Ruffのバージョンを固定し、更新をレビューするコミットにします。また、`ALL` ではなく、プロジェクトが必要な群を列挙し、誰かが追加したときにだけ新しい群が入るようにします。

一つのフォルダーが、プロジェクトの規則に従わなくなる。 ルート設定で禁止した違反がサブディレクトリーで通ったり、チームが無効にしたルールがそこで発火したりします。そのフォルダーに `ruff.toml` 、または `pyproject.toml` の `[tool.ruff]` テーブルがあり、Ruffが最も近いファイルを丸ごと使うため、その下ではルートの設定がすべて適用されなくなります。そこのファイルに `ruff check --show-settings` を実行すると、解決した設定のパスを表示します。入れ子のファイルを削除するか、 `extend = "../pyproject.toml"` で継承させます。

抑制だけが、理由を失っても残る。 正当な理由で付けた `# noqa` コメントが、コードの変更後には何も隠していなかったり、同じ行の次の問題を隠したりします。ルールコードを指定しない `# noqa` は、その行のすべてのルールを黙らせるため、さらに危険です。もう何も抑制していない `noqa` コメントを報告するRUF100と、具体的なルールコードを求めるPGH004を選び、 `--fix` で不要なものを削除します。次の例では、後からインポートが使われるようになり、コメントは何も保護していないため、RUF100が報告します。

```
import json  # noqa: F401

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

フォーマッターとリンターが食い違う。 `ruff format` でファイルを書き換えた後、 `ruff check` がその結果を指摘します。末尾のカンマ（COM812）、インデント（W191、E111）、引用符のスタイル（Q群）など、フォーマッターが決めることを取り締まるルールがあるためです。また、フォーマッターは、可能な範囲でしか指定の行長に折り返さないため、整形後にもE501が発火し得ます。 `ruff format` が出す警告を読みます。文書自身が、警告のない実行を両者が整合しているかの検査としているためです。警告が挙げるルールは除外し、E501は行を組み直すきっかけとして扱います。

一括修正が動作を変える。 ツリー全体に `ruff check --fix --unsafe-fixes` を実行した後、テストが失敗したり、エラーハンドラーが以前は捕捉したものを捕捉しなくなったりします。危険な修正では、実行時の動作の変更やコメントの削除が許容されており、文書自身の例も、コードが送出する例外の種類を変えます。フックやスクリプトでは `--unsafe-fixes` を使わず、 `--select` で一度に一つのルールの危険な修正を適用して差分を読み、チームが検査してから、そのルールを `extend-safe-fixes` で安全な修正に昇格させます。

参考資料

- Ruff — 置き換える道具、主張する速度、維持する組織を説明します。
- Ruffのリンター — ルールの選択と優先順位、安全な修正と危険な修正、`noqa` コメント、終了コードを説明します。
- Ruffのフォーマッター — Blackとの互換性と、整形に競合するリンターのルールを説明します。
- ルール — 各ルールのコード、由来、修正できるかを確認できます。
- Ruffの設定 — ファイルの探索、最も近い設定を使う規則、`extend`、コマンドラインの指定を説明します。

用語集

実際の制約に合わせて選ぶ

業務の流れ、既存のインターフェース、変更を必要とする利用者をお知らせください。検証可能な納品内容と、お客様のチームが保守を続けられる引き継ぎ内容を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「Ruffとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/ruff/