Engineer

Python

このサイトのツール群の大半が使うプログラミング言語

どのようなものか

Pythonは、ソースコードを独立したバイナリへ事前にコンパイルするのではなく、インタープリターが実行する汎用プログラミング言語です。公式文書は、インタープリター方式、対話的、オブジェクト指向と説明し、モジュール、例外、動的型付け、高水準のデータ型を組み込んでいるとしています。ほとんどの人が最初に気づく特徴は、インデントが構文だということです。開始行の下で字下げした部分がブロックなので、読者に見える配置が、そのままインタープリターの実行する構造になります。Guido van Rossumは、アムステルダムのCWIでのABC言語の仕事を踏まえ、1989年のクリスマス休暇に開発を始め、1991年2月に公開しました。名前は、当時読んでいたMonty Python's Flying Circusの台本に由来します。

型付けは動的です。プログラムの実行中に名前とオブジェクトが結び付けられ、型は名前ではなくオブジェクトに属します。そのため、同じ変数がある行では数値を、次の行では文字列を保持できます。PEP 484で標準化された任意のアノテーションを使うと、関数が受け取るものと返すものを宣言できます。ただし、そのPEPは実行時の型検査は行わず、Pythonは動的型付け言語のままであると明言しています。アノテーションを読むのは型チェッカーやエディターなどの別のツールです。インタープリターは保存しますが、その内容を強制しません。

大きな標準ライブラリも付属しています。チュートリアルが「必要なものが初めからそろっている」と表現するように、ファイル、JSON、SQLite、正規表現、子プロセス、引数の解析などを、追加インストールなしで扱えます。それ以外はサードパーティーのパッケージとして提供され、通常はPython Package Indexからpipでインストールします。1つのインタープリターには各パッケージを1バージョンしか置けないため、標準ライブラリには `venv` も含まれます。プロジェクトごとに独立した環境を作り、あるアプリケーションではライブラリの1.0を、別のアプリケーションでは2.0を使えるようにします。

新しいバージョンは予定表に従って登場します。Python 3.9以降、PEP 602により、約17か月の開発を経た機能リリースが毎年10月に公開され、それぞれ5年間保守されます。3.13以降は、最初の2年間におよそ隔月で通常のバグ修正版が出され、その後の3年間はセキュリティ修正だけになります。Python 3.15.0は2026年10月9日に公開されました。Python 3.10のサポートは2026年10月1日に終了し、第2系列の最終版であるPython 2.7は2020年1月1日にサポートを終えました。参照実装のインタープリターであるCPythonは、長い間、同時にPythonコードを実行できるスレッドを1つに制限するグローバルインタープリターロックを持っていました。PEP 703がそれを持たないビルドを追加し、PEP 779は、そのフリースレッド版を公式にサポートしつつ、引き続き任意の選択肢とする段階を3.14の時期に位置付けました。

概念を判断に生かす

「Python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。 データの保存先、処理のルール、他のシステムとの接続方法など、利用者の目に見えない部分をつくります。

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。 ETL/ELTパイプライン6本を設計・運用 — 6 3種類のDBエンジンにまたがるETLパイプライン

ETL/ELTパイプライン6本を設計・運用 お問い合わせ →

なぜ重要なのか

Pythonは、問いから動くプログラムへ進む最短の道になることがよくあります。ファイルを読み、データベースへ問い合わせ、レポートを書くスクリプトを標準ライブラリだけで作り、その場で実行し、ビルド工程なしで変更できます。データパイプライン、ビルド用ツール、テストハーネスの中で処理をつなぐ役として、これほどよく使われる理由です。

読みやすさは偶然の副産物ではなく、意図した設計目標です。PEP 20のThe Zen of Pythonは、読みやすさを重視すると述べています。標準ライブラリ向けスタイルガイドのPEP 8は、命名から1行79文字という規則まで、そうでなければチーム内で議論になる問いに答えています。

柔軟さには代償があり、しかも後になって現れます。属性名の誤記、文字列を期待する場所の `None` 、間違った引数の型などは、その行が実行されて初めてエラーになります。本番環境で初めて通る行かもしれません。そのため、長く使うものをPythonで作るチームは、言語自体にはない検査を加えます。リンター、アノテーションと型チェッカーを品質ゲートに組み込むこと、そしてコードの経路を実際に実行するテストです。

予定表はコードと同じくらい重要です。インタープリターの各バージョンには終了日が決まっているため、誰も触らない環境も、公表された予定に従ってセキュリティサポートの対象外になります。最低バージョンを明示し、各リリースのバグ修正期間である2年間のうちに更新を進めるプロジェクトなら、緊急対応を年1回の計画的なアップグレードに置き換えられます。

実際の活用例

ポートフォリオのデータベースを基にした履歴書・ポートフォリオ生成システムは、文書に記録されたある実装段階でPythonを使っていました。41モジュール、10,580行のコードが、23テーブルを持つ1つのSQLiteデータベースを基に、HTMLやPDFからMarkdownやJSONまで6種類の出力形式を生成していました。データベースを組み立てるのは19段階のパイプラインで、各段階を安全に再実行できました。実行時の依存関係は、テンプレートエンジンとPDFレンダラーの2つだけに抑えました。5年後にインストールできない文書生成システムでは、何も保存したことにならないという考えからです。その代わり、当時はすべての文書がそのPythonコードに依存し、ほかに生成方法は残っていませんでした。これらの数値が表すのは、文書に記録されたその段階の実装です。

Pythonは、電力消費を時間単位で集計するパイプラインでの直接比較でも最速でした。不規則な計測値から時間・地域ごとに1行を作り、発電量に輸入を足して輸出を引いた正味の消費量を求める処理を、Python、JQ、PostgreSQL上のSQLという3通りで実装しました。純粋なPython関数で書いた版が最も速く、1日分を34 ms、30日分のデータセットを180 msで処理しました。

このサイトのビルドにはHugoを使っています。2026年10月に記録されたツール群の状態では、22本、合計8,408行の補助スクリプトがPythonで書かれていました。図を取り込むスクリプトが切り抜きのためにPillowを必要としたほかは、すべて標準ライブラリだけを使っていました。`.tool-versions` にはPython 3.14.7を指定し、それより新しくてもよいが古くてはならない方針を取りました。`task doctor` は、そのファイルを信用するだけでなく、インストール済みの実行ファイルに対して条件を確認していました。リンターはRuffで、basedpyrightの型検査は `standard` レベルで実行していました。既定のレベルでは358件が報告され、そのほぼすべてが意図的に単純な書き方にしたコードのアノテーション不足だったためです。型チェッカーは初回実行で、`None` になり得る値を辞書のキーとして使う箇所を見つけました。Ruffでは問題なしとして通っていた不具合です。

問題が起きるところ

Pythonの問題の多くは、インタープリターが何も指摘せず受け入れるのに、読んだとおりには動かないコードから生じます。

既定の引数が以前の呼び出しを覚えている。 関数が返すリストに前回までの要素が残り、呼び出し順によって結果が変わります。既定値は `def` 文の実行時に1回だけ評価されます。そのため、リストや辞書のような変更可能な既定値は、引数を省いたすべての呼び出しで共有する1つのオブジェクトになります。既定値は `None` にし、関数の本体でリストを作ります。RuffのB006ルールは、このパターンを報告します。

```
def add_tag(tag, tags=None):
    if tags is None:
        tags = []  # 呼び出すたびに新しいリストを作る
    tags.append(tag)
    return tags
```

プロジェクト同士が依存関係を壊し合う。 あるプロジェクトのためにライブラリを更新すると別のプロジェクトが壊れたり、 `pip install` が外部管理環境についてのエラーで止まったりします。どちらも、共有インタープリターへパッケージを直接入れていることが原因です。各プロジェクトがライブラリの同じ1コピーを取り合います。また、PEP 668ではOSの配布元が、自らのPython環境へのpipによる変更を禁止する印を付けられます。 `python -m venv .venv` でプロジェクトごとに環境を作り、その中へインストールし、依存関係をファイルに記録して環境を再構築できるようにします。

型ヒントを検査そのもののように信頼する。 `int` を受け取ると型注釈を付けた関数にJSONファイルから文字列が渡り、3回先の呼び出しで初めて問題が表面化します。アノテーションが示すのは意図であり、インタープリターが強制するわけではありません。型チェッカーを品質ゲートに組み込み、型注釈付きコード同士の不一致でビルドを失敗させます。また、入力を保証できるアノテーションはないため、プログラムへデータが入る場所で検証します。

インタープリターが古くなりサポート対象から外れる。 サーバーが導入時のバージョンを使い続け、そのセキュリティリリースが終わり、パッケージも対応バージョンから外していきます。各機能バージョンのサポートは5年間で、3.10の5年は2026年10月1日に終わりました。検査が読み取るファイルに最低バージョンを明記し、各リリースのバグ修正期間である2年のうちに更新し、切り替え前に新しいバージョンでテストスイートを実行します。

参考資料

- Python公式文書 — 現行リリースのチュートリアル、ライブラリリファレンス、言語リファレンスです。
- Pythonチュートリアル — 変更可能な既定引数への警告を含め、言語を基礎から説明します。
- PEP 8：Pythonコードのスタイルガイド — Pythonコードの多くが従う配置と命名の慣習です。
- PEP 602：Pythonの年次リリースサイクル — 毎年10月に機能リリースを出す理由と、各版のサポート期間を説明します。
- PEP 484：型ヒント — アノテーションの構文と、インタープリターが検査をほかのツールに任せる理由です。
- 仮想環境とパッケージ — `venv` でプロジェクトごとの環境を作り、pipでその中へインストールする方法です。

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

「Python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/python/