Pythonとは

Pythonの特徴 (1/2)

- **1. シンプルで読みやすい文法**
  - スタイルの統一をして、誰が書いたコードでも読みやすく理解しやすいようにする思想の言語
  - 他の言語に比べて文法がシンプルで、コードの記述量が少ないのが特徴
  - インデント（字下げ）によってコードのブロックを表現するため、視覚的に構造が分かりやすく、誰が書いても似たようなコードになりやすい
  - プログラミング初心者でも学びやすい言語と言われている

- **2. 豊富なライブラリとフレームワーク**
  - ライブラリが豊富で、広く使われているフレームワークがある
  - AI（人工知能）開発、データ分析、Web開発、自動化など、様々な分野で使える便利なライブラリやフレームワークが豊富に提供されている
  - ゼロからコードを書く必要がなく、効率的に開発を進めること可能
  - AIや機械学習の分野では、TensorFlow、PyTorch、scikit-learnといった強力なライブラリが充実しており、この分野でPythonが広く使われる理由となっている

Pythonの特徴 (2/2)

- **3. 汎用性が高い**
  - 幅広い分野で活用されている
    - WebアプリケーションYouTubeやInstagramなどがPythonで作られている)
    - データ分析、機械学習、AI開発、科学技術計算
  - 業務の自動化（スクレイピング、ファイルの整理など）に使われる
  - API連携が可能
  - 学術研究での利用
    - 多くの学術論文や研究プロジェクトで使われており、最新の技術・アルゴリズムを利用しやすい
- **4. 活発なコミュニティ**
  - コミュニティが充実
  - 世界中にユーザーがおり、情報やチュートリアルが豊富にある
  - 困ったことがあっても、インターネットで検索すれば多くの情報を見つけることが可能

Pythonの不得意分野

- **実行速度が遅い**
  - インタプリタ言語であるため、実行速度はC言語やJavaといったコンパイラ言語に比べて遅い
  - 動的型付け言語であるため、実行時に型が決定され逐次処理するので処理が重くなる

※ インタプリタ言語：プログラミング言語の命令を1つずつ機械語に変換しながら実行する方式

- **スマホアプリ・ゲーム開発**
  - リアルタイム処理や高速な処理が求められる ゲーム開発、金融システムなどには不向き
  - モバイルアプリ開発に特化した言語ではない
  - スマホアプリの開発には他に適した言語やゲームエンジンがあるため、相対的にPythonはあまり使われない傾向にある

- **高速処理を必要とするシステム**
  - 単に「Pythonのコードをそのまま書く」だけでは高速処理に適さない
  - 標準のPythonでは、 GILによりスレッドセーフティを確保し、メモリ管理を簡素化するために複数のスレッドを同時に実行することができない。これにより、CPUをフル活用した並列処理が難しくなる

※ GIL (Global Interpreter Lock): Pythonのインタプリタが一度に1つのスレッドしか実行できないようにするもの

Pythonの活用例

Pythonは計算や機械学習のライブラリが優れていることから、人工知能やデータサイエンスの分野で使われてことが多いことは知られている。
地理空間情報やその他についても広く使われている。

- AI、人工知能、機械学習での活用
- データサイエンス
- Web上の情報収集（Webスクレイピング）
- ブロックチェーンの開発
- Webサイト、Webアプリケーション開発
- 画像処理
- 業務効率化・自動化
- 地理空間情報での活用

Pythonの学習のコツ

- **いきなり、全てを完璧に理解しようとしない**
- **少しずつ、使いながら覚えていく**
- **エラーを体験し、解決する力を養う**
  - プログラミングにおいて、エラーは避けて通れない道。この経験を積むことで、エラーメッセージを読み解く力、問題の原因を特定する力、そしてそれを解決する力が自然と身についていく
- **調べる方法を知る**
  - Webなどの検索で自分なりの解決方法を確立する
  - AI(ChatGPTなど）で「わからない用語を聞いてみる」「問題を出しもらう」「ヒントを出してもらう」「エラーメッセージの意味と原因を聞いてみる」
- **公式サイトなどのマニュアルやドキュメント(チュートリアル)を読む**
  - 例）Python公式　https://docs.python.org/ja/3/
- **「写経」する**
  - 「写経」とは、お手本となるプログラムのコードをそのまま書き写す学習方法である。　一見すると地味で退屈に思えるかもしれないが、実はプログラミングの基礎を身につける上で非常に効果的な方法である
- **Vibe Coding（バイブ・コーディング）を活用**
  - AIを活用する
  - Vibe Codingで、コードのひな型やヒントを作成してもらい、それを元に学習をする

※ バイブコーディング（Vibe Coding）：AIに自然言語で指示を出すことでコード生成を行い、対話と感覚でスピーディにプログラミングをする手法

Pythonの代表的な外部ライブラリ(1/2)

- **NumPy** (https://numpy.org/)
  - 科学技術計算や機械学習などにおいて代表的なライブラリである。
  - 独自のデータ構造「ndarray」
  - 多次元配列を効率的に扱えるため、ベクトルや行列の演算が必要な分野で多用されている
  - 数値計算用のエンジンである

- **Pandas** (https://pandas.pydata.org/)
  - データ解析を支援する機能を持つライブラリである
  - データ操作のための高速で効率的なデータフレーム (DataFrame、Seriesなど) オブジェクトをもつ
  - データの加工処理をする

- **Matplotlib** (https://matplotlib.org/)
  - グラフ描画ライブラリ

- **Scikit-Learn** (https://scikit-learn.org/)
  - オープンソースの機械学習ライブラリで、他のライブラリとの連携を行いやすいのが特徴である

Pythonの代表的な外部ライブラリ(２/2)

- **OpenCV** (https://opencv.org/)
  - 画像解析に用いられるライブラリです。
- **Django** (https://www.djangoproject.com/)
  - Pythonで実装されたのWebフレームワークで、認証や管理画面など豊富な機能を標準搭載し、高速かつ安全に開発ができる
- **Flask** (https://flask.palletsprojects.com/en/stable/)
  - Pythonで簡単に軽量Webフレームワークで、必要最小限の機能を提供し、拡張性が高く柔軟にWebアプリを構築できる
- **FastAPI** (https://fastapi.tiangolo.com/)
  - Python製の高速Webフレームワークで、型ヒントを活用して自動ドキュメント生成や高い開発効率を実現し、非同期処理に強いのが特徴で、Web APIやマイクロサービスの開発に向いている

地理空間情報とPython

- **地理空間情報での活用**
  - 「地理空間情報（GISデータ）」ではPythonで多くの専用ライブラリやツールが整備されている
  - 地理空間情報とは、位置情報とそれに関連付けられた情報を指す。
  - 例えば、ある場所の緯度経度の座標とその場所の地名、人口、建物などの情報が地理空間情報として扱われる。
  - Pythonは地理空間情報の「解析＋可視化＋機械学習応用」を一気通貫でできる強力な言語です。

- **地理空間情報とGIS**
  - 地理空間情報は、GIS（地理情報システム）と呼ばれるシステムで活用されます。GISは、地理空間情報を収集、管理、分析、表示するためのシステムで、地図の作成や空間分析、地理空間データの管理など、様々な機能を提供します。

地理空間情報を扱う代表的なライブラリ(1/2)

- **GeoPandas** (https://geopandas.org/)
  - Pandasを地理空間データ向けに拡張したライブラリ
  - 地理空間データフレームを扱うためのライブラリで、地図データの読み込み、距離や位置の計算、地図の作成と可視化などが可能

- **Shapely** (https://shapely.readthedocs.io/en/stable/)
  - ジオメトリ（点、線、多角形など）を扱うためのライブラリで、幾何学的なオブジェクトの作成、操作、解析に利用される

- **Pyproj** (pyproj4.github.io/pyproj)
  - 地図投影変換を行うためのライブラリ。
  - ラスターデータを扱うためのライブラリ

- **Fiona** (https://fiona.readthedocs.io/en/stable/)
  - GIS形式のファイルなどのデータの読み書きに特化したシンプルなライブラリ

地理空間情報を扱う代表的なライブラリ(1/2)

- **Folium** (https://python-visualization.github.io/folium/)
  - Pythonでインタラクティブな地図を作成できるライブラリ
  - leaflet.jsというJavascriptでマップデータを使用することのできるライブラリをPythonパッケージ化しにしたもの

- **Cartopy** (https://scitools.org.uk/cartopy/)
  - 地図を描画したりやその他の地理空間データ解析を行うために、地理空間データ処理用のPythonパッケージ
  - 気象データの可視化に強い。

- **leafmap** (https://leafmap.org/)
  - Google Colab/Jupyter notebook環境で最小限のコーディングでインタラクティブなマッピングと地理空間分析を行うためのPythonパッケージ

地理空間情報の活用例 (1/2)

地理空間情報は、位置情報とそれに関連付けられた情報を活用することで、様々な分野で役立つ情報を得ることができる。

- **地図作成**
  - 地図データを作成・編集し、Webサイトやアプリケーションで公開することが可能
- **位置情報分析**
  - GPSデータや位置情報サービスから得られたデータを分析し、人流の把握や混雑状況の予測などに活用できる
- **環境分野**（例: 衛星画像を使った森林変化検出）
  - 衛星画像やセンサーデータを分析し、森林破壊や大気汚染などの環境問題を把握することができる
- **都市計画や交通データ分析**（例: 人口分布と鉄道駅の関係解析）
  - 都市の人口分布や交通状況などの情報を分析し、都市計画や地域開発に役立てることができる
- **防災分野**（例: 洪水浸水予測マップ、避難経路分析）
  - 地図データとハザード情報を組み合わせ、避難経路の計画や災害リスクの評価に活用できる

地理空間情報の活用例 (2/2)

- **ビジネス分野（例: 商圏分析、位置情報マーケティング）**
  - ビジネスにおける地理空間情報の特徴は 「位置」という切り口を加えることでデータの価値を飛躍的に高め、戦略や業務改善に直結できる。
  - 不動産、物流、小売、金融、都市開発など幅広い業界で活用可能である。
  - 例えば、ある地域の人口密度を計算し、地図上に可視化することで、人口が密集している地域や過疎地域を把握することができます。また、複数の地域の人口密度を比較することで、地域間の人口分布の違いを分析することも可能

1. 位置情報に基づく意思決定

- 店舗の出店場所選定（人口動態・交通アクセス・競合位置などを考慮）
- 配送ルート最適化や物流効率化（コスト削減、CO_2排出削減）

2. 顧客理解とマーケティング

- 商圏分析やヒートマップによる顧客分布の把握
- スマホ位置情報やGPSデータを用いたリアルタイム広告配信（ジオターゲティング）

3. リスク管理と予測

- 災害リスクマップ（地震・洪水・土砂崩れ）を活用した保険料設定や不動産評価
- 交通量・気象データを使った需要予測（例: タクシー配車、商品需要）

Python標準の4つの基本的なデータ構造

Python標準では、List（リスト）・Tuple（タプル）・Set（セット）・Dict（辞書）という4つの基本的なデータ構造がある

項目	List	Tuple	Set	Dict（辞書）
読み方	リスト	タプル	セット	ディクト（辞書）
中身の変更	できる	できない	できる	値の変更OK(key, Value)
順番の保証	ある（順番通り）	ある（順番通り）	なし（順不同）	ある（Python3.7以降）
重複の可否	重複OK	重複OK	重複NG	Keyは重複NG
使いどころ	順序が大事で変更もしたいとき	値を固定したいとき、処理速度を上げたいとき	重複を除きたいとき	KeyとValueのペアを扱いたいとき

- Dict（辞書）：Python 3.7 以降: 挿入順が保持されることが、言語の仕様として保証されている。
  Python 3.6以前は 内部の実装により挿入順が保持されていが、順番は保証されてない

NumPy・Pandas・Python標準の違い

項目	NumPy	Pandas	Python標準
データ構 造	多次元配列（ndarray）	データフレーム（表形式）、シ リーズ（1次元）	可変長のリスト
用途	数値データ. 数値計算・行列計算	データ分析・統計処理	汎用的なデータ保持
速度	高速（C実装、ベクトル 化演算可能）	NumPyに比べると遅め	遅い（逐次処理）
機能	数学演算、線形代数、統 計	データ集計、欠損値処理、ラベ ル付き操作	基本的な追加・削除・走査
利便性	数値処理に特化、ブロー ドキャスト可	行・列ラベル、SQLライクな操作性	汎用的だが数値処理に不向 き
代表的な 使い道	行列計算、画像処理、科 学技術計算	データ解析、表形式データの処理	簡単なデータ管理、順序付 き要素の保持

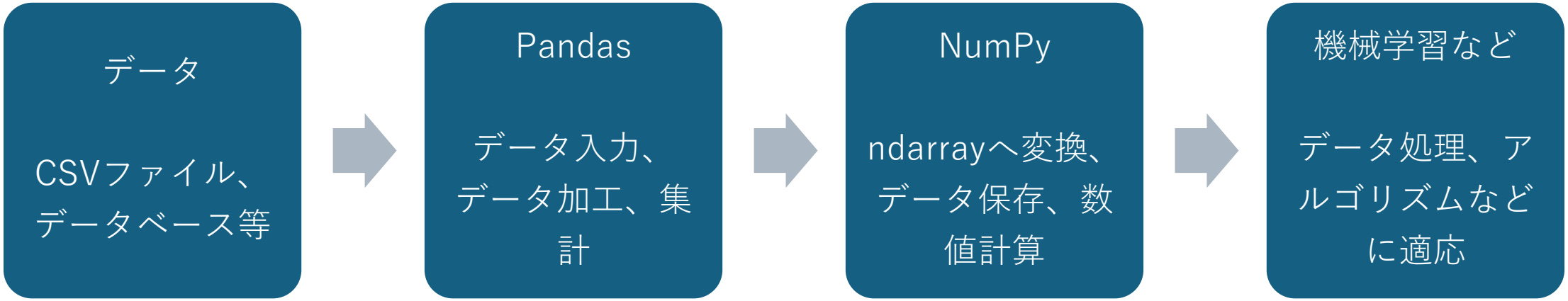
- NumPy → 数値計算・行列演算が得意、格納される要素はすべて同じデータ型（数値型など）で、
  基本的には多次元配列の**数値データ**
- Pandas → 表形式データの処理や分析が得意
- Python標準） → 汎用的だが数値処理には非効率

- つまり、「Python標準」→基礎、「NumPy」→数値計算のエンジン、「Pandas」→データ分析ツール と いう関係

データ分析での流れの例

- 例えば、機械学習をする場合、元データをPandasのDataFrameに落とし込んでデータの前処理と加工をする
- DataFrame内の必要なデータをNumPyのndarrayに変換し、機械学習の演算や高度なアルゴリズムによる処理を行うというの流れが代表的である。
- 実際に触るのが多い型は、DataFrame型が多い

- データ分析では**作業時間の大部分を前処理に割く**ともいわれている。前処理にその後の作業と結果が異なる



Webフレームワークの違い

項目	Django	Flask	FastAPI
特徴	高機能・フルスタック型	軽量・シンプル	高速・型ヒント活用
開発ス タイル	「全部入り」認証・管理画 面・ORMなど標準搭載	必要な機能を拡張で追加	型アノテーション中心で自動ドキュ メント生成
速度	中程度	軽量で比較的速い	非同期処理対応で非常に速い
学習コ スト	やや高い（機能が多い）	低い（シンプル）	中程度（型ヒントと非同期処理の理 解が必要）
利用例	大規模Webアプリ、CMS、 SNS（例: Instagram）	小規模Webアプ リ、プロトタイプ	APIサーバ、マイクロサービス、AI サービスのバックエンド
強み	オールインワンで堅牢	自由度が高く柔軟	高速・自動APIドキュメント生成・ 非同期処理
弱み	重くなりがち、自由度は低め	機能を自前で補う必要	伝統的なWebアプリ機能は少ない

- Django → 大規模・堅牢なWebサービス向け
- Flask → 小規模・柔軟なプロトタイプや実験向け
- FastAPI → 高速なAPI開発やマイクロサービス向け

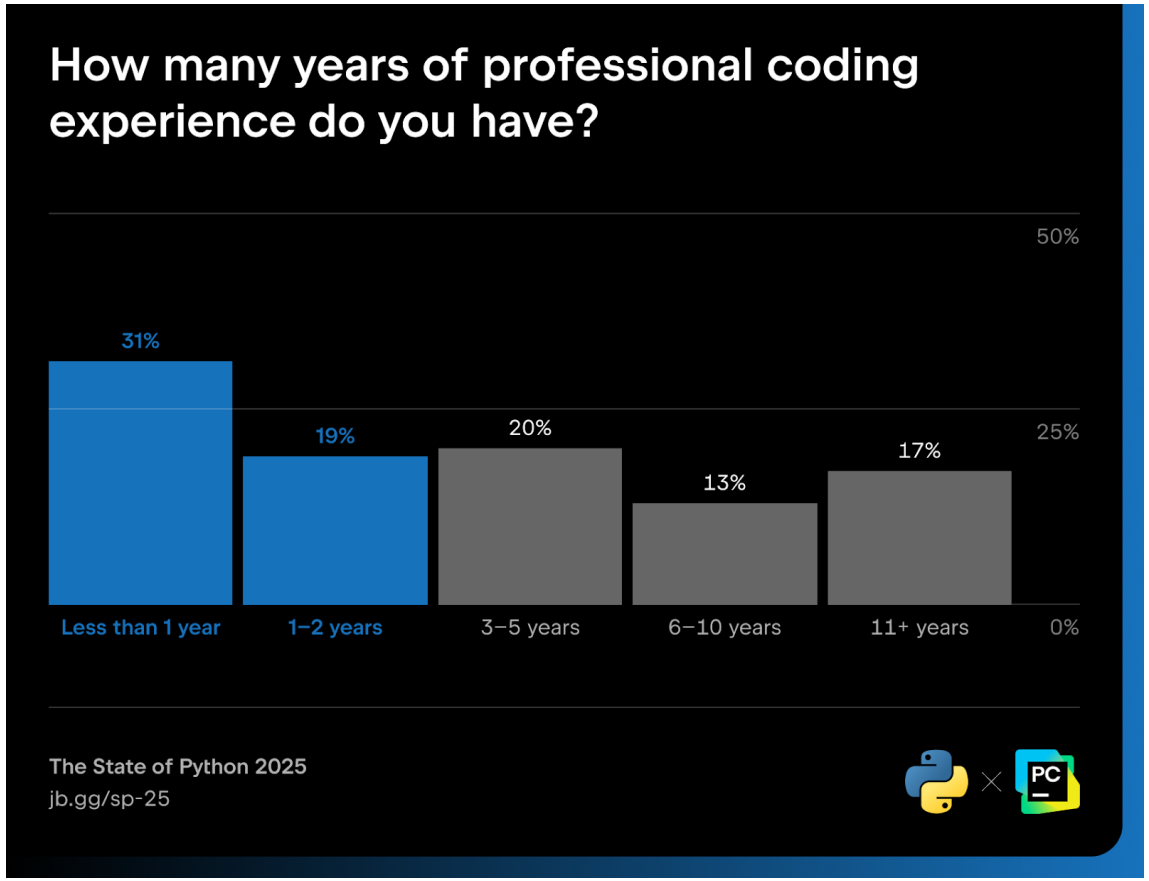
The State of Python 2025

JetBrainsとPython Software Foundationは、3万人以上のPython開発者に対するアンケートを基にしたPythonに関する年次調査結果「The State of Python 2025」を発表

https://blog.jetbrains.com/pycharm/2025/08/the-state-of-python-2025/

The State of Python 2025

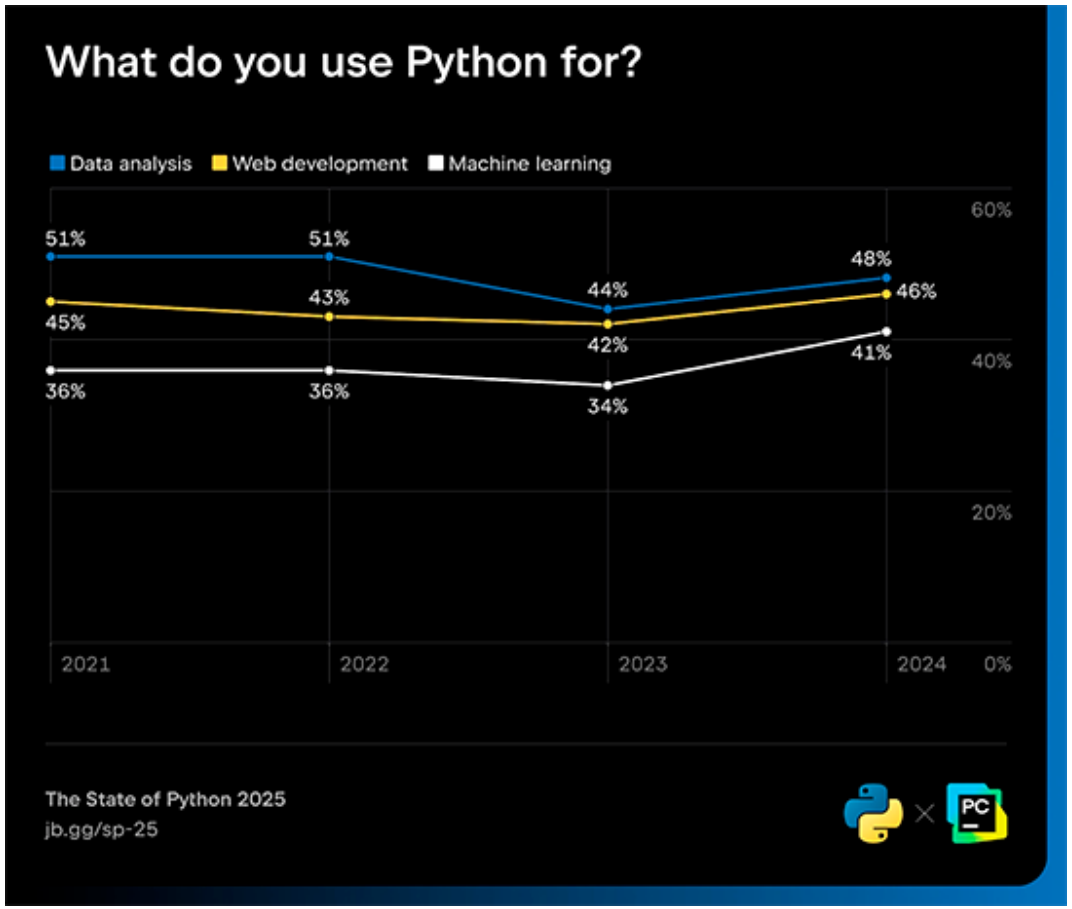
- 回答者の半数が仕事としてのPython経験が2年未満
- この2年で急速にPythonプログラマが増えている？



https://blog.jetbrains.com/pycharm/2025/08/the-state-of-python-2025/

The State of Python 2025

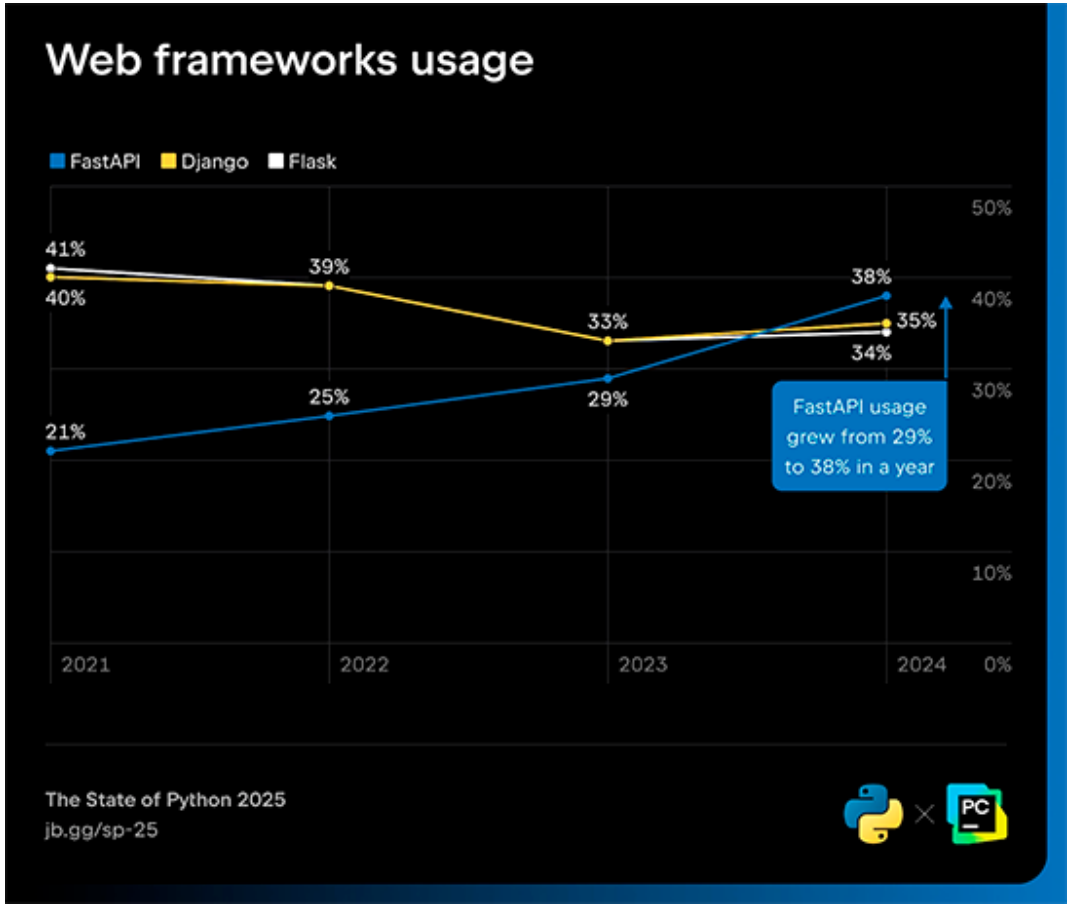
- Pythonを何に使っているのか？
  - データ分析48%、Webアプリケーション開発46%、機械学習41%



https://blog.jetbrains.com/pycharm/2025/08/the-state-of-python-2025/

The State of Python 2025

- WebフレームワークとしてFastAPIが急増
  → 新規参入者が多いことが原因と推測されている



https://blog.jetbrains.com/pycharm/2025/08/the-state-of-python-2025/