■タイトル

特集

人気の二大Python Web UIフレームワークを使い分けよう

Streamlit ＆ Gradio入門

項目	内容
掲載予定	日経ソフトウエア2025年5月発売号（7月号）
分量	12〜16ページ程度
原稿締め切り	2025年3月中旬

■備考

- 文体は「ですます」調
- Pythonのインデントは「半角スペース2文字分」
- スクリーンショットを撮る際は、フォントを大きめで。ブラウザの表示を適度にズームする。また、ウインドウの四隅の丸の外側が白になるように（スクリーンショットを撮るアプリで設定する。あるいは真っ白なウインドウを背景にしてスクリーンショットを撮る）

■構成案

はじめに

最近、Python界隈では、美しくインタラクティブなWebアプリケーションを作成できる
「Python Web UIフレームワーク」と呼ばれるフレームワークが人気を集めています。
そのなかでも特に注目を集めているのは「Streamlit」と「Gradio」の二つです。
これらはいずれもHTMLやJavaScript、HTTPやCSSの知識をほとんど知らなくても
Pythonのみで開発ができることが特徴です。両者はいずれも高機能で完成度も高く、従
来は大量のコードを書かなければ実現できなかったレベルのWebアプリケーションを
短時間で作ることができます。

ただし、いずれも根幹的なコンセプトや方向性は異なり、開発しようとする
アプリケーションの特性に応じて、もしくは考えかたとして自分に向いているものがどちらか
を理解して使いわけることが重要です。本稿では、両者の特徴と基本的な使い方を、
両者の比較を交えながら解説していきます。

Python Web UIフレームワーク

そもそもPython Web UIフレームワークとは何でしょうか？「Djajgo」や「Flask」、「FastAPI」
などのWebフレームワークとは何が違うのでしょうか？ この説明から始めましょう。
一般には以下がPython Web UIフレームワークの特徴です。

- インタラクティブな処理やグラフ表示など、高度なUIを容易に実現できる。
- HTML/CSS/JavaScript/SPAなどのフロントエンド技術が隠蔽されており、基本的にはこれらを意識せにPythonコードだけでWebアプリを作成できる
- ブラウザとサーバ間の通信処理(HTTP)は隠蔽されていて意識しないでよい。POSTやGETなどのHTTPリクエストの存在を意識する必要はない。
- データベースアクセスなどのバックエンド機能などは含まず、UI構築のみが対象。必要であればバックエンド処理のためのライブラリを併用する。

上記により、見た目も良く操作しやすいWebアプリケーション開発を比較的短期間で行うことができます。
反面、自由度や初期表示速度などは、従来型のチューーニングされたWebアプリケーションの
最高性能と比べると劣ります。データサイエンスや生成AIアプリ、ダッシュボードなど、
比較的負荷や性能要件の高くない用途に向いていると言えます。

StreamlitとGradioの比較

StreamlitやGradioはいずれも現在進行形で活発な開発が続けられているWeb UIフレームワークです。したがって、ある時点では一方にのみある機能が、後になって他方に実装されることも良くあります。なので、以下はあくまで2025年5月時点の比較であることにご注意ください。

- 表1 StreamlitとGradio、およびFlaskの比較

項目	Streamlit	Gradio	
開発言語	Python	Python	Python/CSS/HTI
典型的な用途	柔軟で対話的なダッシュボード	機械学習モデルのデモ	任意のWebアプ
UI構築のコンセプト	UIコンポーネントを呼び出す	UIコンポーネントを配線するI	HTMLそのもの
作成したアプリのシェア	ホスティングあり。	ホスティングあり。 トンネリングでローカル実行しているアプリを全世界に公開	
整備された画面部品のライブラリ	多数	多数	なし
学習コスト	Pythonの延長	Pythonの延長	サーバサイドのF

上記において特に用途は重なりあいが大きいことに注意ください。
作りやすいかどうかの程度問題こそあれ「やろうとおもえばできなくはない」ということで
他方だけでできる機能というのは多くはなく、大差ないという面もあります。
ただ、重要な違いとしては、以下があります。

Streamlitは入力や出力実行の過程がそのまま画面配置を決める

コンソールで実行するコマンドラインプログラムとして以下を考えてみてください

```
# コンソール版
a = int(input("A="))
b = int(input("B="))
if b != 0:
  print("A/B = ", a / b)
else:
  print("error")
```

このコードは「 `python3 divide_console.py` 」で実行できます。このコードは前段の結果を得て、後段の処理の入力とするような実行の過程が処理順序に対応しています。
streamlitではこれを以下のように記述します。

```
# streamlit版
# import streamlit as st

a = st.number_input("A")
b = st.number_input("B")
if b != 0:
  st.write("A/B = ", a / b)
else:
  st.write("error")
```

コンソール版にほぼ対応していることがわかるでしょうか。これをstreamlitでは「 `streamlit run divide.py` 」で以下のようにWebアプリとして実行してWebブラウザで特定ポート番号を開くと以下のようにWEbアプリとして実行することができます。

ここで `st.number_input` や `st.witer` などはStreamlitが用意しているUIコンポーネントですが、これを呼び出すことでその順番で画面に配置されてUIが構築されます。

このコードは前段の結果を得て、後段の処理の入力とするような実行の過程が処理順序に対応しています。つまり実行過程とUIの構築が一体です。

この特徴により、Google CoabやJupyter Labs/Notebookなどで検討した結果をstreamlitのアプリに変換するのは容易です。なぜなら、Google Collabなどでも前段のセルの結果をもとにして次の計算を行うため、そのように過程を書き下していけば良いからです。

streamlitdeのUI記述は処理が進むにつれて下に結果が続けられていく処理と相性がよいのです。

GradioでのUI構築は配線である

GradioでのUIコードには大きくわけて2つの方法がある。高レベルと低レベル。

Gradioの低レベルな記述では、イベントハンドラを画面部品に設定していく伝統的なGUIライブラリと同様だが、それを大きくまとめる高水準コンポーネントがあり、「配線」のメタファーでコードを簡易に保つ(ここは実は説明図が必要)

GradioのBlockはwith句を用いたビルダーパターンであるが、これも比較的伝統的なものである。

Streamlit入門

Streamlitのインストール

「Hello, world!」と表示するプログラム

リスト1●「hello_streamlit.py」。Streamlitで「Hello, world!」を表示するプログラム
図1●リスト1の実行結果

BMI計算機のプログラム

リスト2●「bmi_streamlit.py」。Streamlitで作ったBMI計算機のプログラム
図2●リスト2の実行例

リサージュ図形のグラフを描画するプログラム

（ユーザーが係数a、b、cの値を入力できる）
前段を引きついて「BMIの値が健康範囲に入っているかどうか」をプロットするようなコードでも良いか

リスト3●「qf_streamlit.py」。Streamlitで作った2次関数のグラフを描画するプログラム
図3●リスト3の実行例

(カラム)
Streamlitにおけるリアクティブな画面更新

Streamlitは画面が最更新される。一見効率がわるいが、Reactで実装されており描画がモサモサすることはない。またキャッシュや状態の使用が重要である。

Gradio入門

※できればStreamlitと比較しながら書く

Gradioのインストール

Gradioで 「Hello, world!」と表示するプログラム

リスト4●「hello_gradio.py」。Gradioで「Hello, world!」を表示するプログラム
図4●リスト4の実行結果

BMI計算機のプログラム

リスト5●「bmi_gradio.py」。Gradioで作ったBMI計算機のプログラム

図5●リスト5の実行例

Gradioでリサージュ図形のグラフを描画するプログラム

（ユーザーが係数a、b、cの値を入力できる）
前段を引きついて「BMIの値が健康範囲に入っているかどうか」をプロットするようなコードでも良いか
リスト6●「qf_gradio.py」。Gradioで作った2次関数のグラフを描画するプログラム
図6●リスト6の実行例

チャットAIを作ってみよう

（※弊社では今のところ、「AIチャット」ではなく「チャットAI」と表記している）

OllamaとLLM（※gemma2あたりがおすすめ）を導入（※この部分の説明は簡潔に。多少、略していてもOK）

StreamlitでチャットAIを作る

リスト7●「chatai_streamlit.py」。Streamlitで作ったチャットAIのプログラム
図7●リスト7の実行例

GradioでチャットAIを作る

リスト8●「chatai_gradio.py」。Gradioで作ったチャットAIのプログラム
図8●リスト8の実行例

（※おそらくここまでで12ページくらいは埋まると思いますが、足りなそうならor余裕があれば以下を追加。画像の著作権に注意。横浜の風景写真を撮影して利用する）

画像を説明するWebアプリを作ってみよう

マルチモーダル対応LLM（llama3.2-vision:11b）を導入

Streamlitで画像を説明するWebアプリを作る

（画像ファイルのアップロード機能が必要。別のLLMを使って、日本語に翻訳する機能も付ける？）
リスト9●「image_streamlit.py」。Streamlitで作った画像を説明するWebアプリのプログラム
図9●リスト9の実行例

Gradioで画像を説明するWebアプリを作る

（画像ファイルのアップロード機能が必要。別のLLMを使って、日本語に翻訳する機能も付ける？）
リスト10●「image_gradio.py」。Gradioで作った画像を説明するWebアプリのプログラム
図10●リスト10の実行例

(カラム)
マルチページアプリの開発方法、StreamlitとGradioそれぞれで
ページ見合い

まとめ