第1章：Streamlitとは何か

Streamlitは、PythonでシンプルにWebアプリケーションを作成できるオープンソースのフレームワークです。特に、データ分析や機械学習の可視化を目的としたアプリを短時間で構築できる点が特徴です。FlaskやDjangoのようにルーティングやHTMLテンプレートを必要とせず、Pythonコードだけで UI部品やグラフ、インタラクティブな操作を表現できます。

Streamlitの魅力は、1ファイルのPythonスクリプトで動作することと、リアルタイムでの更新性です。コードの変更を即座に反映させることができるため、プロトタイピングやチーム内共有にも適しています。本章ではStreamlitの概要と活用イメージについて解説します。

第2章：開発環境の構築

Streamlitを使用するには、まずPythonがインストールされている必要があります。推奨されるバージョンは3.7以上です。開発環境は、仮想環境（venvやcondaなど）を使用することで、依存関係の競合を避けることができます。

インストール方法は非常に簡単で、以下のコマンドを実行するだけでStreamlitを導入できます：

pip install streamlit

さらに、VSCodeなどのPython対応エディタを用いることで、コード補完や実行が効率化されます。開発ディレクトリの構成や初期ファイルの作り方についてもこの章で紹介します。

第3章：基本的なStreamlitの使い方

Streamlitの基本的なアプリは、数行のコードで構成できます。以下は、タイトルを表示し、簡単なメッセージを出力するだけの最小構成の例です。

import streamlit as st

st.title("はじめてのStreamlitアプリ")

st.write("これはサンプルのメッセージです")

このように、関数ベースでUIを定義でき、コードの実行順に画面が構成されていきます。本章では、st.title, st.header, st.write,
st.markdownなど、代表的な出力系コンポーネントの使い方を中心に解説します。

第4章：インタラクティブなUIコンポーネント

Streamlitでは、ユーザーが操作可能なUI部品を簡単に設置できます。代表的なものとしては、ボタン（st.button）、チェックボックス（st.checkbox）、スライダー（st.slider）、セレクトボックス（st.selectbox）などがあります。

たとえば、セレクトボックスの選択肢によって異なる処理を実行するには以下のように記述します：

option = st.selectbox("選択してください", ["A", "B", "C"])

st.write(f"あなたが選んだのは {option} です")

この章では、こうしたUI部品の動的制御や状態管理の方法についても扱います。

第5章：データの可視化

Streamlitはpandasと組み合わせることで、データの読み込みから可視化までを直感的に行うことができます。たとえば、CSVファイルを読み込み、データフレームの表示とともにグラフを表示することも可能です。

st.line_chart(df)

st.bar_chart(df)

さらに、MatplotlibやPlotly、Altairといった外部可視化ライブラリもそのまま使用でき、インタラクティブなチャートもサポートされています。本章では、データ表示とグラフ描画の基本を学びます。

第6章：外部APIとの連携

Streamlitアプリでは、外部のREST APIと連携してリアルタイムなデータを取得・表示することが可能です。Pythonのrequestsライブラリを用いることで、任意のAPIにアクセスできます。

例えば、天気情報APIや社内の業務APIを使って、データを取得して表示するなどの活用例があります。APIキーの管理には、.envファイルなどを使って環境変数から読み込む方法が一般的です。

この章では、APIとの接続・取得・表示までの一連の処理を実装例とともに解説します。

第7章：ChatGPT APIの活用方法

OpenAIが提供するChatGPT APIを使えば、Streamlitアプリに自然言語での対話機能を追加することができます。ユーザーの入力をpromptとして送信し、その応答をリアルタイムに表示するUIを構築できます。

openaiライブラリを用いて、以下のようなシンプルな呼び出しが可能です：

response = openai.ChatCompletion.create(

model="gpt-4",

messages=[{"role": "user", "content": user_input}]

)

この章では、APIキーの安全な管理方法、システムプロンプトの設計、レスポンス表示の工夫などを紹介します。

第8章：ユーザー入力と自然言語処理

Streamlitアプリでは、st.text_inputやst.text_areaなどを用いて自由な日本語入力を受け付けることができます。これを自然言語処理モデルに渡すことで、意味解析や応答生成が可能になります。

入力テキストの前処理（トリミング、文字数制限、NGワードチェック）や、モデルとのインターフェース設計も重要です。LangChainなどのフレームワークを活用すれば、文書検索型のチャット（RAG）も構築可能です。

本章では、ユーザー入力を中心とした自然言語対話アプリの構成を学びます。

第9章：セキュリティと社内展開時の注意点

Streamlitアプリを社内で展開する場合、セキュリティ面での配慮が必要です。IP制限やVPNアクセスのみにすることで、社外からの不正アクセスを防ぐことができます。

また、APIキーや認証情報はコードに直書きせず、.envファイルや環境変数で管理することが推奨されます。さらに、ファイルアップロード機能などを実装する際は、ファイルサイズ制限やファイルタイプの検証も重要です。

この章では、安全な社内利用を実現するためのベストプラクティスを紹介します。

第10章：トラブルシューティングとよくある質問

Streamlitを使っていると、よくあるエラーや不具合に直面することがあります。例えば、「ModuleNotFoundError」や「Port already in use」などのメッセージは、ライブラリのインストール忘れやポート競合によって発生します。

また、外部API連携時の「Invalid API key」や、UIが更新されないといった事象にも対処が必要です。本章では、代表的なトラブルとその解決方法をFAQ形式でまとめています。

Streamlitのログやコンソール出力を確認し、トラブル発生時に落ち着いて原因を特定できる力を養いましょう。