第5回 Streamlit基本(1) と状態管理入門

`st.session_state` でインタラクティブなアプリを作ろう

担当: [担当者名]
授業資料: `https://x.gd/NoqkC`
出席認証コード: `1506`

今日の目標

1. Streamlitの基本的な表示ウィジェットを理解し、使用できるようになる
2. `st.session_state` の概念と基本的な使い方を習得する
3. インタラクティブなリスト管理（アイテムの追加など）ができるアプリを作成する
4. Pythonの関数と辞書の基本的な概念を理解する

本日の流れ (90分)

時間	内容
00:00-00:10	前回の復習と本日の課題提示
00:10-00:20	**Python基礎: 辞書 (`dict`) の初歩**
00:20-00:30	**Python基礎: 関数 (`def`) の初歩**
00:30-00:50	**Streamlitの状態管理: `st.session_state` 入門**
00:50-01:15	**演習1: `st.session_state` を使った動的持ち物リスト**
01:15-01:25	**Streamlit基本ウィジェット (表示系)**
01:25-01:30	まとめ、質疑応答、次回予告

前回の復習: 持ち物チェックリスト

```
# 持ち物リスト
items = ["PC", "充電器", "スマートフォン", "財布", "筆記用具", "ノート", "ハンカチ", "ティッシュ"]

st.subheader("必須アイテム:")

# for ループを使って items の各要素を st.checkbox で表示
for item in items:
    st.checkbox(item)
```

課題: このリストに新しいアイテムを追加できるようにするには？

実際に実装を試してみましょう！

通常のリストで試してみる

```
import streamlit as st

# 持ち物リスト
items = ["PC", "充電器", "スマートフォン", "財布"]
# 新しいアイテムの入力
new_item = st.text_input("新しいアイテムを入力:")
# 追加ボタン
if st.button("追加"):
    items.append(new_item)  # リストに追加

# リストの表示
st.subheader("持ち物リスト:")
for item in items:
    st.checkbox(item)
```

問題点:

- ボタンをクリックすると、スクリプトが再実行される
- 再実行時に`items`が初期化される
- 追加したアイテムが保持されない→**保持させる仕組みを作る必要がある。**

Python基礎: 辞書 (`dict`)

辞書とは？

- キー(key)と値(value)のペアでデータを格納するデータ型
- キーを使って値にアクセスできる
- 順序は保持されない（Python 3.7以降は保持される）

基本的な使い方（作成とアクセス）

```
# 辞書の作成
person = {
    "name": "太郎",
    "age": 20,
    "city": "東京"
}

# 値へのアクセス
print(person["name"])  # 出力: 太郎
print(person["age"])   # 出力: 20
```

Python基礎: 辞書 (`dict`) の操作

値の更新と追加

```
# 値の更新
person["age"] = 21
print(person["age"])   # 出力: 21

# 新しいキーと値の追加
person["job"] = "学生"
print(person["job"])   # 出力: 学生

# 辞書の内容を確認
print(person)  # 出力: {'name': '太郎', 'age': 21, 'city': '東京', 'job': '学生'}
```

主な操作

- 値の取得: `dict[key]`
- 値の更新: `dict[key] = new_value`
- 新しいキーと値の追加: `dict[new_key] = new_value`
- キーの存在確認: `key in dict`

Python基礎: 関数 (`def`)

関数とは？

- 同じ処理をまとめて再利用しやすくする
- コードの可読性を向上させる

基本的な使い方

```
# 関数の定義
def greet(name):
    return f"こんにちは、{name}さん！"

# 関数の呼び出し
message = greet("太郎")
print(message)  # 出力: こんにちは、太郎さん！
```

Python基礎: 関数の実践例

素数判定

前回のコードでは`for-else`構文を使用していました：

```
for num in range(2, 21):
    for i in range(2, num):
        if num % i == 0:
            break  # 内側のループを抜ける
    else:
        # breakされずにループが完了した場合に実行
        print(num, end=" ")
```

`for-else`の動作:

- `for`ループが`break`で中断されずに最後まで実行された場合
- `else`ブロックが実行される
- 今回の関数では`return`を使うことで、同じ動作をより簡潔に表現

関数を使った例

```python
# 素数判定の関数
def is_prime(n):
    # 2未満の数は素数ではない
    if n < 2:
        return False

    # 2からn-1までの数で割り切れるかチェック
    for i in range(2, n):
        if n % i == 0:  # 割り切れたら素数ではない
            return False

    # 一度も割り切れなかったら素数
    return True
```

動作の流れ:

1. 2未満の数は素数ではないので、すぐに`False`を返す
2. 2からn-1までの数で順番に割り算を試す
3. 割り切れたら（余りが0なら）素数ではないので`False`を返す
4. 最後まで割り切れなかったら素数なので`True`を返す

Streamlitの状態管理: `st.session_state` 入門

なぜ `st.session_state` が必要か？

- Streamlitのスクリプトは、ユーザーの操作（ボタンクリックなど）があるたびに**最初から最後まで再実行**される
- 通常の変数は再実行時に**リセット**される
- ユーザーの操作やデータの変更を**保持**するには、特別な仕組みが必要

→ `st.session_state` の出番！

`st.session_state` とは？

- アプリケーションのセッション間で**持続する**辞書型のオブジェクト
- キーと値のペアでデータを保存
- ブラウザを閉じるまで値が保持される

```
# 基本的な使い方（2つの書き方）
# 1. ドット記法
if 'count' not in st.session_state:
    st.session_state.count = 0
st.session_state.count += 1

# 2. 角括弧記法
if 'count' not in st.session_state:
    st.session_state['count'] = 0
st.session_state['count'] += 1

# どちらの書き方でも同じ結果
st.write(f"カウント: {st.session_state.count}")  # または st.session_state['count']
```

簡単なデモ: カウンターアプリ

```
import streamlit as st

# カウンターの初期化
if 'count' not in st.session_state:
    st.session_state.count = 0

# カウントアップボタン
if st.button('カウントアップ'):
    st.session_state.count += 1

# カウンターの表示
st.write(f"現在のカウント: {st.session_state.count}")

# リセットボタン
if st.button('リセット'):
    st.session_state.count = 0
```

演習1: 動的持ち物リスト

目標: 第4回の持ち物リストを改造し、ユーザーが新しいアイテムを追加できるようにする

ステップ:

1. 持ち物リストを `st.session_state` で初期化
2. `st.text_input` で新しいアイテム名を入力
3. `st.button` でアイテムをリストに追加
4. `for` ループでリストを表示

演習1: コードのひな形

```
import streamlit as st

# 持ち物リストの初期化
if 'items' not in st.session_state:
    st.session_state.items = ["PC", "充電器", "スマートフォン", "財布"]

# 新しいアイテムの入力
new_item = st.text_input("新しいアイテムを入力:")

# 追加ボタン
if st.button("追加"):
    # ここに追加のロジックを書く
    pass

# リストの表示
st.subheader("持ち物リスト:")
for item in st.session_state.items:
    st.checkbox(item)
```

Streamlit基本ウィジェット (表示系)

`st.write()`

- 最も汎用的な表示コマンド
- 文字列、数値、データフレームなど、様々なデータを表示

```
st.write("Hello, World!")  # 文字列
st.write(42)  # 数値
st.write(["apple", "banana", "orange"])  # リスト
```

Streamlit基本ウィジェット (表示系)

テキストフォーマット

```
st.header("大見出し")
st.subheader("中見出し")
st.caption("キャプション（小さな説明文）")
```

Markdown

```
st.markdown("""
# Markdown見出し
これは **太字** で、これは *イタリック* です。
- リスト項目1
- リスト項目2
""")
```

Streamlit基本ウィジェット (表示系)

`st.button()`

- クリック可能なボタンを表示
- クリックイベントで条件分岐を実行

```
if st.button("クリックしてください"):
    st.write("ボタンがクリックされました！")
```

まとめ

1. **Python基礎:**
    - 辞書 (`dict`): キーと値のペアでデータを管理
    - 関数 (`def`): 処理のまとめ、再利用
    - `for-else` 構文: ループの完了条件に応じた処理
2. **Streamlitの状態管理:**
    - `st.session_state` でユーザーの操作やデータの変更を保持
    - 辞書型のオブジェクトとして扱う
    - ドット記法と角括弧記法の両方でアクセス可能
3. **Streamlit基本ウィジェット:**
    - `st.write()` : 汎用的な表示
    - `st.header()` , `st.subheader()` , `st.caption()` : テキストフォーマット
    - `st.markdown()` : Markdown記法での表示
    - `st.button()` : クリックイベントの処理
    - `st.text_input()` : テキスト入力

次回予告

第6回: Streamlit基本(2) 多様な入力ウィジェット

- `st.selectbox`: ドロップダウンリスト
- `st.multiselect`: 複数選択
- `st.slider`: スライダー
- `st.number_input`: 数値入力
- `st.text_input`: テキスト入力（より詳しく）

より複雑なユーザー入力を受け付ける方法を学びます！

付録: 演習1の解答例

```python
import streamlit as st

# 持ち物リストの初期化
if 'items' not in st.session_state:
    st.session_state.items = ["PC", "充電器", "スマートフォン", "財布"]

# 新しいアイテムの入力
new_item = st.text_input("新しいアイテムを入力:")

# 追加ボタン
if st.button("追加") and new_item:  # new_itemが空でない場合のみ追加
    # 新しいリストを作成して代入
    st.session_state.items = st.session_state.items + [new_item]

# リストの表示
st.subheader("持ち物リスト:")
for item in st.session_state.items:
    st.checkbox(item)
```