# 第6回: Streamlit基本 (2)

## 多様な入力ウィジェット

担当: 大妻女子大学 社会情報学部

出席認証コード: **1887**

授業資料: https://x.gd/NoqkC

## 前回の復習

- Streamlitの基本表示機能
  - `st.write`：テキストやデータの表示
  - `st.header`, `st.markdown`：見出しやマークダウン表示
- 基本的なインタラクション
  - `st.button`：ボタンクリックによる処理
  - `st.checkbox`：チェックボックスによる条件分岐

# 今回の授業内容

1. Streamlit入力ウィジェットの基本
2. 選択型ウィジェット
   - `st.selectbox`
   - `st.multiselect`
3. 数値入力ウィジェット
   - `st.slider`
   - `st.number_input`
4. テキスト入力ウィジェット
   - `st.text_input`
5. 総合演習

## Streamlit入力ウィジェットの基本

- ユーザーからのデータ入力を受け付ける機能
- 入力値は変数に格納して処理に使用
- 共通パラメータ
  - `label`: ウィジェットのラベル
  - `help`: ヘルプテキスト
  - `key`: ウィジェットの一意の識別子

## 選択型ウィジェット: `st.selectbox`

```python
import streamlit as st

# ドロップダウンリストの作成
option = st.selectbox(
    label="好きな果物を選んでください",
    options=["りんご", "バナナ", "オレンジ", "ぶどう"],
    help="リストから選択してください"
)

# 選択値の表示
st.write(f"あなたが選んだ果物: {option}")
```

## `st.selectbox` の特徴

- ドロップダウンリスト形式
- 単一の選択肢のみ選択可能
- 選択値は変数に格納
- 条件分岐と組み合わせて使用可能

```
# 条件分岐の例
if option == "りんご":
    st.write("りんごは健康に良い果物です")
elif option == "バナナ":
    st.write("バナナはエネルギー補給に最適です")
# 他の条件分岐...
```

## 選択型ウィジェット: `st.multiselect`

```
import streamlit as st

# 複数選択リストの作成
options = st.multiselect(
    label="好きな果物を選んでください（複数選択可）",
    options=["りんご", "バナナ", "オレンジ", "ぶどう", "いちご"],
    default=["りんご", "バナナ"],
    help="複数の果物を選択できます"
)

# 選択値の表示
st.write(f"あなたが選んだ果物: {options}")
```

## `st.multiselect` の特徴

- 複数の選択肢を選択可能
- 選択値はリスト形式で格納
- `default` パラメータで初期選択値を設定可能
- リスト処理と組み合わせて使用可能

```
# リスト処理の例
if options:
    st.write(f"選択した果物の数: {len(options)}")
    for fruit in options:
        st.write(f"- {fruit}")
else:
    st.write("果物が選択されていません")
```

## 数値入力ウィジェット: `st.slider`

```python
import streamlit as st

# 数値スライダーの作成
age = st.slider(
    label="年齢を選択してください",
    min_value=0,
    max_value=100,
    value=20,
    step=1,
    help="スライダーを動かして年齢を選択"
)

# 選択値の表示
st.write(f"あなたの年齢: {age}歳")
```

## `st.slider` の特徴

- スライダー形式の数値入力
- 最小値、最大値、初期値、ステップ値を設定可能
- 日付や時刻の選択にも使用可能
- 範囲選択も可能

```
# 範囲選択の例
age_range = st.slider(
    label="年齢範囲を選択してください",
    min_value=0,
    max_value=100,
    value=(20, 40),  # 範囲の初期値
    step=1
)
st.write(f"選択した年齢範囲: {age_range[0]}歳から{age_range[1]}歳まで")
```

# 日付/時刻スライダー

```python
import streamlit as st
from datetime import datetime, time

# 日付スライダー
date = st.date_input(
    label="日付を選択してください",
    value=datetime.now(),
    help="カレンダーから日付を選択"
)
st.write(f"選択した日付: {date}")

# 時刻スライダー
time_value = st.time_input(
    label="時刻を選択してください",
    value=time(12, 0),
    help="時刻を選択"
)
st.write(f"選択した時刻: {time_value}")
```

## 数値入力ウィジェット: `st.number_input`

```python
import streamlit as st

# 数値入力フィールドの作成
number = st.number_input(
    label="数値を入力してください",
    min_value=0,
    max_value=100,
    value=50,
    step=1,
    help="数値を直接入力"
)

# 入力値の表示
st.write(f"入力された数値: {number}")
```

## `st.number_input` の特徴

- 数値を直接入力するフィールド
- 最小値、最大値、初期値、ステップ値を設定可能
- スライダーと比べて正確な値の入力が可能
- 計算や処理に使用する数値の入力に適している

```
# 計算例
if number > 0:
    st.write(f"{number}の2乗: {number ** 2}")
    st.write(f"{number}の平方根: {number ** 0.5}")
```

## テキスト入力ウィジェット: `st.text_input`

```python
import streamlit as st

# テキスト入力フィールドの作成
name = st.text_input(
    label="お名前を入力してください",
    value="",
    help="あなたの名前を入力してください",
    placeholder="例: 山田 太郎"
)

# 入力値の表示
if name:
    st.write(f"こんにちは、{name}さん！")
else:
    st.write("名前が入力されていません")
```

## `st.text_input` の特徴

- テキストを直接入力するフィールド
- `value` パラメータで初期値を設定可能
- `placeholder` パラメータでプレースホルダーテキストを表示可能
- 入力値の検証や処理に使用可能

```
# 入力値の検証例
if name:
    if len(name) < 2:
        st.error("名前は2文字以上で入力してください")
    elif len(name) > 20:
        st.warning("名前が長すぎます")
    else:
        st.success(f"こんにちは、{name}さん！")
```

## 総合演習: 簡単なアンケートアプリ (例)

`src/lecture06/app_comprehensive_example.py` を確認してみましょう。
このファイルには、これまで学んだ複数の入力ウィジェット（`st.text_input`, `st.slider`, `st.selectbox`, `st.multiselect`）と、フォーム (`st.form`) を組み合わせたアンケートアプリの例が記述されています。
入力値の検証や条件分岐による表示の変化も含まれています。
（旧 `app_input_widgets.py` です。）

## 演習課題

以下の2つのStreamlitアプリケーションを作成してみましょう。
それぞれ、課題用のテンプレートファイル ( `_template.py` ) と解答例ファイル ( `_solution.py` ) を `src/lecture06/` フォルダ内に用意します。

**取り組み方:**
ペアワークで取り組みましょう。どちらか1つの課題を選び、まずは各自で作成してみてください。
完成したら、ペアの人にどのように作成したか、工夫した点などを説明し合いましょう。
もし行き詰まった場合は、ペアで相談しながら進めてみてください。

## 課題1: 大学生活充実度チェッカー

**目的:** 自分の大学生活についていくつかの質問に答えることで、簡単なフィードバックやアドバイスが得られるアプリを作成します。

**テンプレートファイル:** `src/lecture06/app_life_checker_template.py`
**解答例ファイル:** `src/lecture06/app_life_checker_solution.py`

**作成するアプリの仕様:**

1. **タイトル:** 「大学生活充実度チェッカー」
2. **入力項目:**
   - 学部: `st.selectbox` (例: ["文学部", "経済学部", "理工学部", "社会情報学部", "その他"])
   - 週の平均勉強時間: `st.slider` (0〜50時間)
   - 所属しているサークル・部活動: `st.multiselect` (例: ["運動系サークル", "文化系サークル", "部活動", "特になし", "その他"])
   - 友人関係の満足度: `st.radio` (例: ["大変満足", "満足", "普通", "少し不満", "不満"])
   - 睡眠時間: `st.number_input` (0.0〜12.0時間、ステップ0.5)
3. **処理:** 「診断する」ボタン ( `st.button` ) を押すと、入力内容に基づいて簡単なフィードバックを表示します。
   - 例: 勉強時間が短い場合は「もう少し勉強時間を確保しましょう！」、サークル活動がない場合は「何か新しい活動を始めてみては？」など。
   - 複数の条件を組み合わせたユニークなアドバイスを考えてみましょう。

## 課題2: 理想のバイトプランナー

**目的:** 自分の希望するアルバイトの条件を入力し、それに基づいた（架空の）おすすめ情報やアドバイスを表示するアプリを作成します。

**テンプレートファイル:** `src/lecture06/app_job_planner_template.py`
**解答例ファイル:** `src/lecture06/app_job_planner_solution.py`

**作成するアプリの仕様:**

1. **タイトル:** 「理想のバイトプランナー」
2. **入力項目:**
   - 希望職種（キーワード）: `st.text_input` (例: "カフェ", "塾講師", "データ入力")
   - 希望時給: `st.number_input` (最低800円〜、ステップ50円)
   - 週の希望勤務時間: `st.slider` (0〜40時間)
   - 希望勤務曜日: `st.multiselect` (例: ["月", "火", "水", "木", "金", "土", "日"])
   - 重視する点: `st.selectbox` (例: ["給与", "楽しさ", "スキルアップ", "通いやすさ", "シフトの柔軟性"])
3. **処理:** 「プランニング開始」ボタン ( `st.button` ) を押すと、入力内容をまとめた上で、いくつかの（架空の）おすすめバイト情報や、探し方のアドバイスを表示します。
   - 例: 希望時給が高い場合は「高時給のバイトは競争率が高いので、スキルをアピールしましょう！」、特定の曜日しか働けない場合は「シフトの融通が利きやすいバイトを探してみましょう」など。

## まとめ

- Streamlitの多様な入力ウィジェット
  - 選択型: `st.selectbox`, `st.multiselect`
  - 数値入力: `st.slider`, `st.number_input`
  - テキスト入力: `st.text_input`
- 各ウィジェットの特徴と適切な使用場面
- 入力値の取得と処理方法
- 条件分岐やリスト処理との組み合わせ

## 次回予告

次回は「Streamlit基本 (3): レイアウト、状態管理、ファイル入力」について学びます。

- レイアウト機能 (`st.columns`, `st.expander`, `st.sidebar`)
- 状態管理 (`st.session_state`)
- フォーム (`st.form`, `st.form_submit_button`)
- ファイル入力 (`st.file_uploader`)