Ｗｅｂプログラミング Ⅰ

第2回：Python基礎 (1) - 変数、データ型、演算

2025年度 大妻女子大学 社会情報学部
担当：余野
出席認証コード: 3042
授業資料: https://x.gd/NoqkC (大学のアカウントでgoogleにログインしていることが必要)

本日の内容

1. 前回の復習
2. Pythonの変数とデータ型
3. 演算子と式
4. リスト・タプルの基本操作
5. Streamlitでの簡単な入力処理

前回の復習

Webアプリケーションとは

- ユーザーからの入力に対して**動的に反応**するWebサイト
- サーバー側で処理を行い結果を返す

Streamlitの特徴

- Pythonだけでwebアプリを開発できる
- データ可視化に特化
- シンプルなコードでアプリ作成が可能

Pythonとは？

- 1991年に登場した**汎用プログラミング言語**
- **シンプルで読みやすい**構文が特徴
- 多様な分野で利用されている：
  - データ分析
  - 機械学習・AI
  - Web開発
  - 科学計算
  - 自動化スクリプト
- **大規模なライブラリとフレームワーク**が豊富

Pythonの変数と代入

変数は値を保存するための名前付きの「箱」です。

```
# 変数名 = 値
name = "大妻 花子"
age = 20
is_student = True

# 複数の変数に同時に代入
x, y, z = 1, 2, 3

# 変数の値は上書き可能
age = 21
```

変数の命名規則

- 英数字とアンダースコア(_)を使用
- 数字で始めることはできない
- 大文字小文字を区別（`name` と `Name` は別の変数）
- 予約語は使用できない (`if` , `for` , `while` など)

Pythonの基本データ型

数値型

```
# 整数型 (int)
age = 20
count = -5

# 浮動小数点型 (float)
height = 162.5
pi = 3.14159
```

文字列型 (str)

```
name = "大妻 花子"
message = 'Pythonを学んでいます'
multiline = """複数行の
テキストも
書けます"""
```

Pythonの基本データ型（続き）

ブール型 (bool)

```
is_student = True
has_finished = False
```

None型

```
# 値が未定義であることを表す
result = None
```

型を確認する

```
# type()関数で型を確認できる
print(type(42))        # <class 'int'>
print(type("こんにちは"))  # <class 'str'>
print(type(True))      # <class 'bool'>
```

型変換（キャスト）

Pythonでは型を変換することができます。

```
# 文字列から整数へ
age_str = "20"
age_int = int(age_str)  # 20 (整数)

# 整数から文字列へ
count = 100
count_str = str(count)  # "100" (文字列)

# 文字列から浮動小数点へ
height_str = "165.5"
height_float = float(height_str)  # 165.5 (浮動小数点)

# 注意: 変換できない場合はエラー
# int("abc")  # エラー: 数字ではない文字列
```

算術演算子

Pythonでは様々な数値計算ができます。

```
# 基本的な算術演算子
a = 10
b = 3

print(a + b)   # 13  (加算)
print(a - b)   # 7   (減算)
print(a * b)   # 30  (乗算)
print(a / b)   # 3.3333... (除算 - 結果は常に浮動小数点)
print(a // b)  # 3   (整数除算 - 小数点以下切り捨て)
print(a % b)   # 1   (剰余 - 割り算の余り)
print(a ** b)  # 1000 (べき乗 - aのb乗)
```

代入演算子

計算と代入を同時に行うことができます。

```
x = 10

# 以下の二つは同じ意味
x = x + 5
x += 5

# 他の代入演算子
x -= 3   # x = x - 3 と同じ
x *= 2   # x = x * 2 と同じ
x /= 4   # x = x / 4 と同じ
x //= 2  # x = x // 2 と同じ
x %= 3   # x = x % 3 と同じ
x **= 2  # x = x ** 2 と同じ
```

比較演算子

二つの値を比較し、True/Falseを返します。

```
a = 10
b = 5

print(a == b)  # False (等しい)
print(a != b)  # True  (等しくない)
print(a > b)   # True  (より大きい)
print(a < b)   # False (より小さい)
print(a >= b)  # True  (以上)
print(a <= b)  # False (以下)
```

論理演算子

複数の条件を組み合わせることができます。

```
x = 10
y = 5

# and演算子（両方がTrueならTrue）
print(x > 5 and y < 10)  # True

# or演算子（少なくとも一方がTrueならTrue）
print(x < 5 or y < 10)   # True

# not演算子（真偽を反転）
print(not x == 10)       # False
```

文字列の操作

Pythonでは文字列を様々な方法で操作できます。

```
# 文字列の連結
first_name = "大妻"
last_name = "花子"
full_name = first_name + " " + last_name  # "大妻 花子"

# 文字列の繰り返し
stars = "*" * 5  # "*****"

# 文字列の長さ
length = len(full_name)  # 5

# 文字列のインデックス（0から始まる）
first_char = full_name[0]  # "大"

# 部分文字列（スライス）
substring = full_name[0:2]  # "大妻"
```

文字列のメソッド

文字列には便利なメソッドが多数あります。

```
message = "Python プログラミング"

# 大文字・小文字変換
print(message.upper())       # "PYTHON プログラミング"
print(message.lower())       # "python プログラミング"

# 置換
print(message.replace("Python", "Streamlit"))  # "Streamlit プログラミング"

# 分割
words = message.split()      # ["Python", "プログラミング"]

# 文字列の検索
print("Python" in message)   # True
print(message.find("Python"))  # 0 (見つかった位置のインデックス)
print(message.find("Java"))    # -1 (見つからない場合)
```

リストの基本

リストは複数の値を順序付けて保存するデータ構造です。

```
# リストの作成
fruits = ["りんご", "バナナ", "オレンジ"]
numbers = [1, 2, 3, 4, 5]
mixed = [1, "りんご", True, 3.14]  # 異なる型も混在可能

# 空のリスト
empty_list = []

# リストの要素にアクセス（インデックスは0から始まる）
first_fruit = fruits[0]  # "りんご"
last_fruit = fruits[2]   # "オレンジ"

# 負のインデックスは末尾から数える
last_item = fruits[-1]   # "オレンジ"

# リストの長さ
count = len(fruits)  # 3
```

リストの操作

リストは可変 (mutable) なので、中身を変更できます。

```
fruits = ["りんご", "バナナ", "オレンジ"]

# 要素の変更
fruits[1] = "ぶどう"  # ["りんご", "ぶどう", "オレンジ"]

# 追加
fruits.append("いちご")  # ["りんご", "ぶどう", "オレンジ", "いちご"]

# 挿入
fruits.insert(1, "メロン")  # ["りんご", "メロン", "ぶどう", "オレンジ", "いちご"]

# 削除
fruits.remove("ぶどう")  # ["りんご", "メロン", "オレンジ", "いちご"]

# インデックスで削除
del fruits[2]  # ["りんご", "メロン", "いちご"]

# 末尾の要素を取り出して削除
last = fruits.pop()  # "いちご", fruits = ["りんご", "メロン"]
```

リストのスライス

リストの一部を取り出すことができます。

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

# スライス: [開始:終了] (終了インデックスは含まない)
subset = numbers[2:5]  # [2, 3, 4]

# 開始インデックスを省略すると最初から
first_half = numbers[:5]  # [0, 1, 2, 3, 4]

# 終了インデックスを省略すると最後まで
second_half = numbers[5:]  # [5, 6, 7, 8, 9]

# ステップを指定 [開始:終了:ステップ]
every_second = numbers[::2]  # [0, 2, 4, 6, 8]

# 負のステップで逆順
reversed_list = numbers[::-1]  # [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

タプル

タプルはリストに似ていますが、一度作成すると変更できません（イミュータブル）。

```
# タプルの作成（丸括弧を使用）
coordinates = (3, 4)
person = ("大妻花子", 20, "東京")

# 1要素のタプルは末尾にカンマが必要
single_item = (42,)

# タプルの要素にアクセス（リストと同じ）
name = person[0]  # "大妻花子"
age = person[1]   # 20

# タプルはイミュータブル（変更不可）
# person[1] = 21  # エラー：タプルは変更できない

# タプルのアンパック
name, age, location = person  # 複数の変数に同時に代入
```

演習1: 変数と計算

課題: 身長(m)と体重(kg)を表す変数を作成し、それらを使ってBMI（体重(kg) / 身長(m)の2乗）を計算して表示してください。

条件:

- 身長を1.65m、体重を55kgとして計算してください。

ヒント:

- べき乗の計算には `**` 演算子を使います。

演習2: 文字列操作

課題:

1. 自分の名前（ローマ字）を変数 `my_name` に代入してください。
2. `my_name` をすべて大文字にして表示してください。
3. `my_name` の文字数を表示してください。
4. `my_name` を使って、 `"Hello, [あなたの名前]! Welcome!"` という形式の挨拶文を作成して表示してください。

ヒント:

- 大文字変換には `.upper()` メソッドを使います。
- 文字数の取得には `len()` 関数を使います。
- 文字列の連結には `+` 演算子を使います。

演習3: リスト操作

課題:

1. `["apple", "banana", "cherry"]` というリストを作成し、変数 `fruits` に代入してください。
2. リスト `fruits` に `"orange"` を追加してください。
3. リスト `fruits` の2番目の要素（インデックス1）を `"grape"` に変更してください。
4. リスト `fruits` から `"apple"` を削除してください。
5. 最終的なリスト `fruits` とその要素数を表示してください。

ヒント:

- 要素の追加には `.append()` メソッドを使います。
- 要素の変更はインデックスを指定して代入します: `fruits[1] = ...`
- 要素の削除には `.remove()` メソッドを使います。
- リストの要素数は `len()` 関数で取得できます。
- リストを表示するには `print("Final list:", fruits)` のように表示します。

Streamlitの入力ウィジェット: st.text_input

テキスト入力フィールドを作成し、ユーザー入力を受け取ります。

```
import streamlit as st

st.title("テキスト入力の例")

# シンプルなテキスト入力
name = st.text_input("お名前を入力してください")

# 入力があれば挨拶を表示
if name:
    st.write("こんにちは、"+ name + "さん！")
```

演習4: Streamlit テキスト入力

課題: GitHub Codespaces上で、ユーザーの名前を入力させ、挨拶を表示する簡単なStreamlitアプリを作成してください。

手順:

1. Codespacesでファイル (`src/lecture02/app.py`) を編集します。
2. `streamlit` ライブラリをインポートします。
3. `st.title()` でアプリのタイトルを設定します（例: "挨拶アプリ"）。
4. `st.text_input()` を使って、ユーザーに名前を入力してもらうフィールドを作成し、結果を変数に格納します。
5. 入力された名前が存在する場合 (`if name:` などでチェック)、 `st.write()` を使って `"こんにちは、[名前]さん！"` のような挨拶を表示します。
6. Codespacesのターミナルを開き、 `streamlit run src/lecture02/app.py` コマンドでアプリを実行します。
7. 表示されるURLをクリックして、アプリが動作することを確認します。

ヒント:

- 前のスライドの `st.text_input` の例を参考にしてください。

本日のまとめ

- **変数**は値を保存するための名前付きの箱
- Pythonの**基本データ型**：
  - 数値型（int, float）
  - 文字列型（str）
  - ブール型（bool）
- **演算子**を使って計算や比較ができる
- **リスト**は複数の値を順序付けて保存する可変なデータ構造
- **タプル**はリストに似た不変のデータ構造
- Streamlitの**text_input**で簡単なユーザー入力が実装できる

次回予告

「Python基礎 (2): 制御構造と関数」

- 条件分岐（if, elif, else）
- 繰り返し処理（for, while）
- 関数の定義と呼び出し
- スコープとライフタイム

Google Colaboratoryの使い方（前回資料）

- **Pythonの学習用**として活用できます。
- ブラウザ上で**すぐに始められ**、環境構築は不要です。

手順

1. 大学のIDとPWでGoogleアカウントでアクセス: https://colab.research.google.com/
2. 「ファイル」メニューから「ノートブックを新規作成」を選択。
3. セルにPythonコードを記述し、「Shift + Enter」で実行。

例: Hello Worldを表示

```
print("Hello, World!")
```

GitHubアカウントの作成（前回資料）

今後の演習で作成したコードやアプリを保存・共有するために、GitHubアカウントの作成します。

1. https://github.com/ にアクセス。
2. 右上の「Sign up」ボタンをクリック。（**既に作成済みの場合は「Sign in」**）
3. 画面の指示に従って、ユーザー名、メールアドレス、パスワードを設定。
   ※ ユーザー名は OWU 学籍番号　例）OWU131300000
   ※ メールアドレスは大学のもの

GitHub Codespacesの使い方（前回資料）

授業用のテンプレートから、自分専用の開発環境をブラウザ上で起動します。

1. 以下のテンプレートリポジトリURLにアクセス:
   `https://github.com/kyouto-yono-ac/web_programming_2025`
2. 緑色の **`Use this template`** ボタンをクリックし、「Open in codespace」を選択。
3. 少し待つと、ブラウザにVS Codeのようなエディタが表示されます。
4. エディタでコードを編集し、ターミナルで実行します。