Python 入門: 数値計算・データ・グラフ

スクリプト言語演習

佐藤 正平、草野 翼、伊澤 侑祐、大久保 寛

Scripting Languages / EECS / 05

本日の流れ: 計算して、保存して、グラフにする

段階	できるようになること	主な道具	課題
1. Python に慣れる	計算し、結果を表示する	変数, `print`, `def`	1
2. まとめて計算する	多くの数を一度に計算する	`numpy`	2, 3
3. データを保存する	計算結果をファイルに書き、読み戻す	`np.savetxt`, `np.loadtxt`	4
4. グラフにする	結果を図にして確かめる	`matplotlib`	5

- 今日のゴール: 測定データのような数の並びを、計算・保存・グラフ化できるようになる
- 繰り返し (`for`) と分岐 (`if`) は次回。今日はほとんど使わない
- 各スライドの右側に、コードの読み方を書いてある

Google Colab の使い方

1. Python に慣れる

- ブラウザだけで Python を動かせる。インストールは要らない
- ノートブックは「セル」の並び。セルには、コードと説明文の 2 種類がある
- コードのセルを選んで `Shift + Enter` で実行する。結果はセルのすぐ下に出る
- セルは**上から順に**実行する。途中から始めると「名前が見つからない」と出る
- 最初に ファイル → ドライブにコピーを保存を選ぶ。選ばないと、書いた内容が消える
  困ったら ランタイム → すべてのセルを実行で、最初からやり直せる

値と変数: 計算した結果に名前を付ける


```
v = 5.0          # 電圧 [V]
r = 100          # 抵抗 [Ω]
i = v / r        # 電流 [A]
print(i)         # 0.05

print(2 ** 10)   # 1024
print(7 // 2, 7 % 2)   # 3 1
```

コードの読み方

1. `v = 5.0`: 値 `5.0` に `v` という名前を付ける (変数)
2. `#` から後ろは説明 (コメント)。実行されない
3. `v / r`: 割り算。結果を `i` に入れる
4. `print(...)`: 値を画面に表示する
5. `**` は累乗、`//` は割り算の商、`%` は余り

文字列と f-string: 結果を読みやすく表示する


```
name = "R1"
v, r = 5.0, 100
i = v / r

print("抵抗", name, "の電流")
print(f"{name}: {i} A")          # R1: 0.05 A
print(f"{name}: {i*1000:.1f} mA")  # R1: 50.0 mA
```

コードの読み方

1. `"R1"`: 文字列。`"` で囲む
2. `v, r = 5.0, 100`: 2 つの変数に、まとめて値を入れる
3. `print` に `,` で並べると、空白をはさんで表示する
4. `f"..."` (f-string): `{ }` の中に変数や式を書くと、その値に置き換わる
5. `{i*1000:.1f}`: 小数点以下 1 けたで表示する

リスト: 複数の値を 1 つにまとめる


```
volts = [1.0, 2.0, 3.0]

print(volts[0])      # 1.0 (先頭)
print(volts[-1])     # 3.0 (最後)
print(len(volts))    # 3
volts.append(4.0)
print(volts)         # [1.0, 2.0, 3.0, 4.0]
```

コードの読み方

1. [ ] で囲み , で区切ると、リストになる
2. `volts[0]`: 先頭の要素。番号は 0 から数える
3. `volts[-1]`: 最後の要素
4. `len(...)`: 要素の数
5. `append(...)`: 後ろに 1 つ足す

- 全要素を 2 倍にするような計算は、次に学ぶ numpy の方が簡単

関数: よく使う計算に名前を付ける


```
def current(v, r):
    return v / r


print(current(5.0, 100))    # 0.05
print(current(12, 4))       # 3.0
```

コードの読み方

1. `def current(v, r):`: `current` という関数を作る。`v` と `r` を受け取る
2. 中身は字下げ (空白 4 つ) して書く
3. `return v / r`: 計算した値を、呼んだ所へ返す
4. `current(5.0, 100)`: 関数を呼ぶ。`v` に 5.0、`r` に 100 が入る

print は表示するだけ。値を返すには *return* を書く

numpy: 多くの数をまとめて計算する

2. まとめて計算する

```
import numpy as np

v = np.array([1.0, 2.0, 3.0])
print(v * 2)        # [2. 4. 6.]
print(v + 10)       # [11. 12. 13.]

i = v / 100
print(i)            # [0.01 0.02 0.03]
```

コードの読み方

1. `import numpy as np`: 数値計算の道具 numpy を、`np` という短い名前で使えるようにする
2. `np.array([...])`: リストから numpy の配列を作る
3. `v * 2`: 配列の**すべての要素**を 2 倍にする。`for` は要らない
4. `v / 100`: 全要素を 100 で割る。電圧の列から電流の列が一度に求まる

配列どうしの計算: 同じ位置どうしで計算する

2. まとめて計算する — 課題 3

```
import numpy as np

v = np.array([1.0, 2.0, 3.0])   # 電圧 [V]
i = np.array([0.1, 0.2, 0.5])   # 電流 [A]

p = v * i                       # 電力 [W]
print(p)                        # [0.1 0.4 1.5]
print(v / i)                    # [10. 10.  6.]
```

コードの読み方

1. 同じ長さの配列を 2 つ用意する
2. `v * i`: 1 番目どうし、2 番目どうし…と掛ける。結果も配列
3. `v / i`: 同じ位置どうしで割る。抵抗の列になる
4. 長さが違う配列どうしでは計算できない (エラーになる)

集計: 平均・最大・最小を一度に求める

2. まとめて計算する — 課題 2

```
import numpy as np

t = np.array([20.5, 22.0, 19.5, 23.0])

print(np.mean(t))    # 21.25  平均
print(np.max(t))     # 23.0   最大
print(np.min(t))     # 19.5   最小
print(np.std(t))     # 1.35…  ばらつき
```

コードの読み方

1. `t`: 4 回測った温度の配列
2. `np.mean(t)`: 平均 (全部足して個数で割る)
3. `np.max(t)` / `np.min(t)`: いちばん大きい値 / 小さい値
4. `np.std(t)`: 標準偏差。値のばらつきの大きさ

等間隔の数を作り、関数の値を求める

2. まとめて計算する

```
import numpy as np

t = np.linspace(0, 1, 5)
print(t)          # [0.   0.25 0.5  0.75 1.  ]

f = 2             # 周波数 [Hz]
v = np.sin(2 * np.pi * f * t)
print(v.round(2))
```

コードの読み方

1. `np.linspace(0, 1, 5)`: 0 から 1 までを、等間隔に 5 個の数で区切る
2. `np.pi`: 円周率
3. `np.sin(...)`: 配列の各要素の sin をまとめて求める
4. `v` は、時刻 `t` での正弦波の値の列になる
5. `round(2)`: 小数点以下 2 けたに丸める (表示を見やすくするため)

ファイルに書く・読む

3. データを保存する

```
with open("memo.txt", "w") as f:
    f.write("5.0 V\n")
    f.write("100 ohm\n")

with open("memo.txt") as f:
    text = f.read()
print(text)
```

コードの読み方

1. `open("memo.txt", "w")`: ファイルを書き込み用 (`"w"`) で開く。無ければ作る
2. `with ... as f:` の中で `f.write(...)` すると、文字が書き込まれる
3. `with` を抜けると、ファイルは自動で閉じる
4. `"w"` を付けないと、読み込み用で開く。`f.read()` で中身を全部読む
5. `\n` は改行

- Colab では、左の「ファイル」(フォルダの絵) から、できたファイルを見られる

数の表を CSV で保存し、読み戻す

3. データを保存する — 課題 4

```
import numpy as np

t = np.array([0.0, 0.1, 0.2])
v = np.array([0.0, 0.6, 0.9])

np.savetxt("wave.csv", np.column_stack([t, v]),
           delimiter=",")

t2, v2 = np.loadtxt("wave.csv", delimiter=",",
                    unpack=True)
print(t2, v2)
```

コードの読み方

1. `np.column_stack([t, v])`: 2 つの配列を、2 列の表にする
2. `np.savetxt(...)`: 表を `wave.csv` に書く。`delimiter=","` で列を , で区切る (CSV)
3. `np.loadtxt(...)`: CSV を読み戻す
4. `unpack=True`: 列ごとに分けて返す。1 列目が `t2`、2 列目が `v2` に入る

- CSV は、数を , で区切って並べた文字のファイル。Excel でも開ける

グラフを描く

4. グラフにする — 課題 5

```
import numpy as np
import matplotlib.pyplot as plt

t = np.linspace(0, 1, 100)
v = np.sin(2 * np.pi * 2 * t)

fig, ax = plt.subplots()
ax.plot(t, v)
ax.set_xlabel("time [s]")
ax.set_ylabel("voltage [V]")
ax.set_title("2 Hz sine wave")
plt.show()
```

コードの読み方

1. `matplotlib.pyplot` を `plt` という名前で使う
2. `plt.subplots()`: 図 (`fig`) と、描く場所 (`ax`) を作る
3. `ax.plot(t, v)`: 横軸 `t`、縦軸 `v` の折れ線を描く
4. `set_xlabel` / `set_ylabel` / `set_title`: 軸の名前と題を付ける
5. `plt.show()`: 図を表示する

軸の名前は英語で書く。日本語は Colab では文字化けする (□ になる)

線を重ねて凡例を付け、画像に保存する

4. グラフにする — 発展

```
fig, ax = plt.subplots()
ax.plot(t, np.sin(2 * np.pi * 2 * t), label="2 Hz")
ax.plot(t, np.sin(2 * np.pi * 4 * t), label="4 Hz")
ax.set_xlabel("time [s]")
ax.legend()
fig.savefig("waves.png")
```

コードの読み方

1. 同じ `ax` に `plot` を 2 回書くと、線が 2 本重なる
2. `label="2 Hz"`: その線の名前
3. `ax.legend()`: 線の名前の一覧 (凡例) を図に出す
4. `fig.savefig("waves.png")`: 図を画像ファイルに保存する。レポートに貼れる

まとめと課題

まとめ

やりたいこと	使う道具	課題
計算に名前を付ける	`def`, `return`	1
平均・最大・最小	`np.mean`, `np.max`, `np.min`	2
多くの数をまとめて計算	`np.array`, 配列どうしの計算	3
CSV に保存・読み戻す	`np.savetxt`, `np.loadtxt`	4
グラフにする	`plt.subplots`, `ax.plot`, `set_xlabel`	5
線を重ねる・移動平均でならす	`label`, `legend`, `np.convolve`	発展 1, 2

- 「Colab で開く」から課題を開き、ファイル → ドライブにコピーを保存
- 各課題: 例題セルを実行 → `TODO` を埋める → 直下の確認セルが `OK` になれば完了
- 基本課題 1〜5 で 90 点。発展は、余裕があれば取り組む