全３回で学ぶPython入門講座 3回目

2025/08/05

Kazuma Sekiguchi

前回のAgenda

- 標準ライブラリーの利用
- ユーザーからの入力方法
- リストの利用
- 繰り返し処理の利用
- 辞書、集合の利用
- ファイルを開く、書き込み保存

今回のAgenda

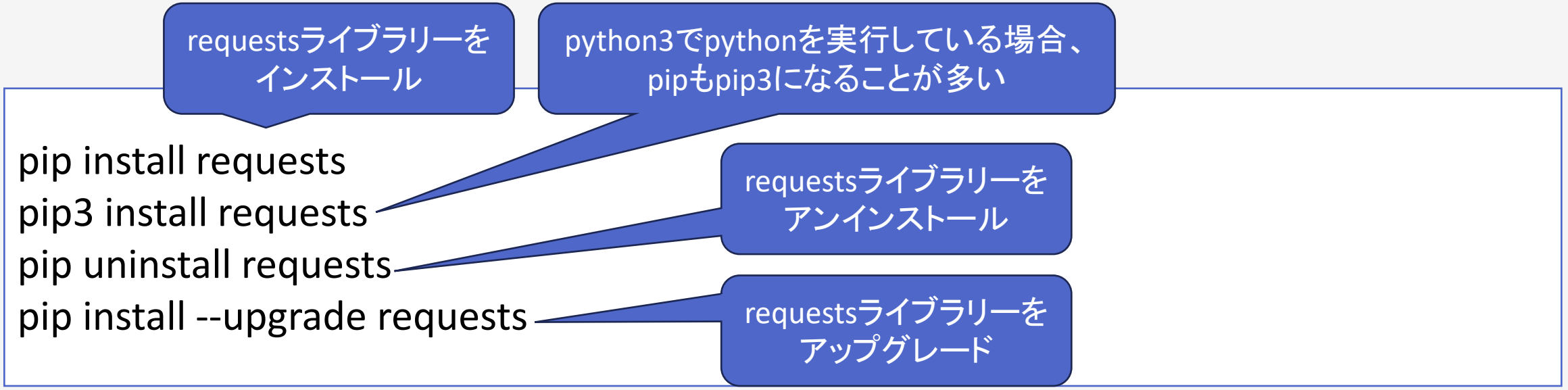
- pipの使い方
- requestsモジュールによる外部Webサイトデータの取得
- JSONデータからの情報抽出
- matplotlibモジュールによるデータ可視化
- グラフの作成

pipの使い方

- Pythonで外部ライブラリーを利用する際に必要なファイルを取得してくれるツール
  - 今はPythonに付属している
- Python Package Indexに登録されていないライブラリーの導入はpipでは不可能
  - ほとんどの主流なライブラリーは登録されているため、実務的に困ることは無いはず
  - あまり新しいものだと登録されていないため、その場合は手動で導入が必要

pipの使い方

- pipの後に取得したいライブラリー名を指定する
  - ネットに繋がっている場合、自動的にライブラリーを取得してくれる
  - 依存関係にあるライブラリーも導入してくれるが、場合によって問題が出ることもある



pipで導入したライブラリーの利用

- 依存関係のあるライブラリーが存在する
  - ライブラリーがほかのライブラリーの機能を利用している場合、依存関係がある、とする
  - この場合、使いたいライブラリーのほかに、機能を利用しているライブラリーを導入する必要がある
- pipで導入したライブラリーはimportを使って読み込めば利用可能
  - 依存関係にあるライブラリーがある場合は、importで依存関係のあるライブラリーも指定する
    - importは必須でない依存パッケージは明示的にimportしなくても動くことも多い

pipでの問題

- 依存関係にあるライブラリーも自動的に導入してくれる
  - バージョンの問題が生じることがある
  - AがVersion2.3を要求、BがVersion2.56を要求
    - pipではどちらかしか満たせないため、インストールに失敗する
  - 例：あるプロジェクトでAというライブラリーのVersion1.0を使っているが、他のプロジェクトではAのVersion1.2を要求する場合
    - 互換性があれば問題無いが、プロジェクトAが動作しなくなることもあり得る
- プロジェクトごとに要求するバージョンが異なる場合、少し面倒なことが起きる可能性がある

プロジェクトごとに分ける

- Poetry,Pipenvなどのプロジェクト単位でライブラリーバージョンを厳密に管理するツールを利用する
  - 独立した仮想環境を作成して、それぞれのプロジェクトごとに分離をすることで、複数バージョンを共存することが可能

- Pythonで複数プロジェクトを作成するときには採用を検討した方が無難

requestsライブラリーの利用

- Webページを取得することのできるライブラリーがrequestsライブラリー
  - pipで導入する必要がある
  - Pythonだけでも取得は可能だが、HTTPクライアントとして動作してくれるため、取得が容易になる
- 基本的にはHTMLや画像などWeb上で展開されているものは取得可能
  - 取得した後にどう加工するかなどは別途記述が必要

requestsライブラリーの利用

```
import requests #requestライブラリーの読み込み
url = “http://example.com” #取得するURL
res = requests.get(url) #指定したページの内容の取得
print(res.text) #取得したページの表示
print(res.status_code) #ステータスコードの表示
print(res.headers) #ヘッダーの表示

payload = {'key1': 10, 'key2': 'string'}
res = requests.get(url, params=payload)
#getでパラメータはparams=で指定

res = requests.post(url, data=payload) #POSTでデータを送信
#postでパラメータはdata=などで指定
```

requestsライブラリーの利用

- 取得したデータをファイルに保存することももちろん可能
  - 順番にいろいろなWebサイトを巡回させて情報を取得することも可能＝スクレイピング
  - 頻度などに充分注意すること

- POSTとGETで渡すパラメータは辞書型として与える
  - POST時にJSONとして渡したい場合は、dataの代わりにjsonを利用する

```
res = requests.post(url, json=payload) #POSTでJSONデータを送信
```

JSONデータの取得

- いわゆるAPIなどでは、返って来るデータはJSONが一般的になっている
  - JSON=JavaScript Object Notation
  - データを記述するための軽量なデータ形式
  - もともとはJSのオブジェクトに似た構造でJSで扱うことを前提として作成されていたが、今はWebのAPIなどで頻繁に使われるため、どの言語でも容易に扱うことが可能
  - そのままJSONの形式で扱うのはほとんどの場合扱いづらいため、形式を変換して利用することが多い

PythonでのJSON扱い

```
import requests


response = requests.get('https://api.example.com/data')
data = response.json()  # ←ここでJSONをPythonのdict/listに変換
```

ここでJSONデータを
受け取る

- 外部からJSONデータを受け取るときは、requestsライブラリーを使い、json()を利用することでPythonの辞書やリストに変換する
  - 万が一正しいJSONを返してこないときはエラーになる
- 受け取ったデータを格納したdata変数の構造は確認するべき
  - それによって、どのようにデータを取り扱うかが変わってくる

PythonでのJSONの扱い

- print()ではざっくりとした構造しか把握しにくい
  - 細かい構造を知るためにはjsonライブラリーを読み込んで

JSONデータを変換したデータの入っている変数

```
json.dumps(data, indent=2, ensure_ascii=False)
```

のようにして詳細な構造を確認した方が良い

- JSONの作りによっては極めて複雑な構造をしていることがある
  - パースしてデータを取り出すために工夫が必要なことも多い

json.dump()

- json.dump()
  - Pythonのデータ（辞書やリスト）をJSON形式でファイルに書き込む

```
import json

data = {"name": "田中", "age": 20}
with open("sample.json", "w", encoding="utf-8") as f:
    json.dump(data, f, ensure_ascii=False, indent=2)
```

data辞書をJSON形式でsample.jsonに書き込む

- json.dumps()は文字列でやりとりを行う

json.load()

- ファイルに保存されたJSON形式のデータをPythonのデータとして読み出す

```
import json

with open("sample.json", "r", encoding="utf-8") as f:
    data = json.load(f)
print(data)  # {'name': '田中', 'age': 20}
```

sample.jsonからJSON形式のデータを取り出して、dataに辞書データとして格納する

- json.loads()は文字でやりとりを行う

zip()

- 複数のリストなどの「同じ順番の要素」をペア（タプル）にして並べる関数

```
a = [1, 2, 3]
b = ['a', 'b', 'c']

for x, y in zip(a, b):
    print(x, y)
```

```
1 a
2 b
3 c
```

- 2つのリストをペアにできる
  - 更にこれをリストに格納することも可能
  - リストの長さが異なると短い方に合わせて止まる点には注意

zip()を使いリスト化

4番目　3番目　2番目　1番目

```
daily_weather = [{"time": t, "temp": temp} for t, temp in zip(times, temps)]
```

- 時間のリストと気温のリストが存在する
  - zip()で時間と気温でペアを作成する
  - 時刻と気温をそれぞれ取り出し、{"time":t,"temp":temp}の辞書形式に格納し、それをリスト化する

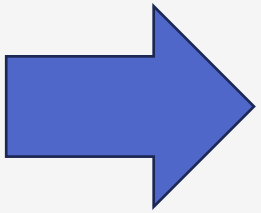
```
daily_weather = []
for i in range(len(times)):
    daily_weather.append({
        "time": times[i],
        "temp": temps[i]
    })
```

上記と同じことを
実現する別の書き方

文字の抜き出し

- 変数に格納されている文字列から一部を抜き出したい場合

```
times = [
    "2025-08-04T00:00",
    "2025-08-04T01:00",
    "2025-08-04T02:00",
]
```



```
times = [
    "00:00",
    "01:00",
    "02:00",
]
```

- x_labels ＝ [t[-5:] for t in times]のようにして取り出せる
- t[-5:]で後から5文字分取り出す、として機能させることが可能

matplotlibモジュール

- Pythonでグラフや図を描画するための代表的な可視化ライブラリ
  - データを与えることで、さまざまなグラフを描画することが可能
  - 折れ線グラフ・棒グラフ・円グラフ・散布図などが作成できる
- もともとPythonはデータ解析に良く使われるため、グラフ化する需要が高い

```
import matplotlib.pyplot as plt
plt.plot([1,2,3], [4,5,6])
plt.show()
```

matplotlibをpltという名前で利用する
asで別名を付けている

X軸で1,2,3、Y軸で4,5,6を通る折れ線
グラフを描画（実際には直線になる）

matplotlibモジュールの利用

- サイズを指定する

```
plt.figure(figsize=(10, 5))
```

- 軸（X,Y）を指定し、折れ線グラフを描画

```
plt.plot(x_labels, temps, marker="o")
```

  - X軸名、Y軸名、点の形状
  - 点の形状はいろいろと指定することができる
  - 丸：o、四角：s、三角（上）：^、三角（下）:v、三角（左）：<
    三角（右）：>、菱形：D、星形：*

matplotlibモジュールの利用

- グラフのタイトルを指定する

```
plt.title("グラフタイトル")
```

- X軸のラベルを指定する

```
plt.xlabel("X軸ラベル")
```

- Y軸のラベルを指定する

```
plt.ylabel("Y軸ラベル")
```

- X軸ラベルを傾ける

```
plt.xticks(rotation=45)
```

matplotlibモジュールの利用

- グリッド線表示

```
plt.grid()
```

- レイアウト調整

```
plt.tight_layout()
```

- グラフを表示する

```
plt.show()
```

matplotlibモジュールの利用

- 少しずつ与えるデータは異なるが、基本的に最初に
  指定したメソッドで描画されるグラフが変わる
  - data、sizes、x,yはリスト、labelsは項目名、binsは分割数
  - 棒グラフ：plt.bar(x, y)
  - 円グラフ：plt.pie(sizes, labels=labels,
    autopct="%1.1f%%")
  - 散布図：plt.scatter(x, y, marker="o", color="red")
  - ヒストグラム：plt.hist(data, bins=10)
  - 箱ひげ図：plt.boxplot(data)

ラベルの文字

- デフォルトのままだと文字化けするので日本語フォントに変更しておく

```
plt.rcParams["font.family"] = "MS Gothic"
```

mac環境であれば、YuGothic などを指定する

ファイル名指定だけで関数を実行する

```
if __name__ == "__main__":
    main()
```

- pythonでファイル名を指定して実行したときには__name__の名前が__main__となる
  - ゆえに、main()が呼び出されて実行される
- importされた場合には、__name__の名前が異なるため、main()は実行されない
  - ファイル名だけで任意の関数を実行することが可能になる

チャレンジ

- 以下のプログラムを作成してみてください
  - 答えの例は格納してあります

- テキストファイルを指定したら、そのファイルを読み込んで、中に書いてある文字数を数えて表示するプログラムを作成してください。
- pipは使用しません

チャレンジ

- 以下のプログラムを作成してみてください
  - 答えの例は格納してあります

- タスク管理アプリケーションを作成してください。タスクを追加、表示、完了にする、削除するというメニューを持ち、追加を選択した場合はタスクの追加が可能で、表示は保存されているタスクを表示します。
  完了にするを選択したら、完了にするタスクを選択し、完了として扱うようにします。
- pipは使用しません

チャレンジ

- 以下のプログラムを作成してみてください
  - 答えの例は格納してあります

- 住所録を保存したり参照できるようなアプリケーションを作成してください。テキストファイルに住所録は保存します。名前、住所、電話番号を登録できるようにし、表示もできるようにします。保存を選択することで、ファイルに書き込みをします。
- pipは使用しません

ありがとうございました。

Pythonはまだ他にもいろいろなことができます

プログラムを考えて実践してみてください

今後の参考書など

- クジラ飛行机『実践力を身につける Pythonの教科書 第2版』,2024,マイナビ出版
- 国本 大悟（著), 須藤 秋良（著）『スッキリわかるPython 入門 第2版』,2023,インプレス
- wat（著） 『いきなりプログラミング Python』,2024, 翔泳社