プログラミング演習 2(Python)　第 2 回レポート

21K1016　喜多修也

2022 年 7 月 24 日

1　課題

以下のプログラム課題に取り組みレポートを提出せよ。
自分なりに設計した CGI プログラムを作成せよ。授業で紹介したプログラムを拡張・発展させたものでも構わない。評価ポイントは以下の通りである。
テーマの独自性 (30
データベースの利用 (20
web API の利用 (20
ユーザインタフェースの工夫 (20
結果の HTML ファイルの工夫 (10
前半で学んだデータ分析手法を適切に利用した場合加点する。レベルの目安としては、授業で紹介したプログラムを上記の項目に関して 1 点程度拡張・発展させたものが問題なく動いた場合に C とする。

2　はじめに

2022 年は 7 月に入った。今年は梅雨が早く明けたと話す一方、本レポートを記している時期である 7 月の中旬は毎日雨が降っており、なかなか洗濯物を干すことができない日々が続いた。起床時間では曇っていても、数時間経過したら雨が降っているという日もあった。本レポートは、東京都の一週間の天気に関する web-api を取得し、指定した時間から未来 6 時間分の天気予報のデータを基に、その時間に洗濯物を干す事ができるか判定できる CGI を作成し、その仕様を述べる目的で記した。ここで、東京都を選んだ理由は私が暮らしているからだ。

3　プログラムの内容と手法

3-1　プログラムの内容

本レポートで作成した CGI は、指定した日時に洗濯物を干す事ができるかどうかを判断する物だ。以下より、これを『洗濯判定くん』と記す。具体的には、Open-Meteo が提供する、世界各地域の天気に関する web-api の情報とユーザの入力した時間をもとに、入力された時間帯で洗濯が可能かどうかを判定する CGI だ。この web-API は指定した地域において、閲覧時刻から一週間までのデータを提供している。私が作成した『洗濯判定くん』は、wep ページ上に存在する月、日、時間の 3 つのデータの入力を取得し、入力した時間から 6 時間後分の天気コードのデータを基に判断する。判断する天気コードは 0 の『快晴』、1〜3 の『晴れ〜曇り』、45 以上の雨を指標としている。洗濯できるかできないかの出力は 4 種類あり、『選択可能』、『洗濯微妙』、『洗濯不可』、『洗濯非推奨』である。6 時間の天気予報において、雨が降るタイミングが 1 回でもあれば『洗濯不可』を出力する。これは、雨が洗濯物に濡れてしまうと洗濯物が乾かないからだ。6 時間の内の晴れる回数と曇る回数を比較した時、晴れる回数が曇る回数以上になった時『洗濯可能』を出力し、一方下回った場合は『洗濯微妙』を出力する。曇りであっても洗濯物が濡れる事はなく、乾きやすいからだ。

未来 6 時間分の天気予報のデータから判断する理由は、洗濯物を干す時間を 6 時間としたためだ。

また、入力した時間が 16 時以降、もしくは 0〜3 時になった場合は『洗濯非推奨』を出力する。これは、入

力した値が夕方や夜であり、これは洗濯物を夜に干す事を意味するためだ。ここで、16 時以降を非推奨とした理由は日没の時間を 18 時とすると、干し始めてから干し終わる時間までの間において、日の出る時間より沈んでいる時間の方が長くなるためだ。一方 0 時から 3 時を非推奨とした理由については、日の出の時刻を 6 時としたとき、干し始めてから干し終わる時間までの間において、日の出る時間より沈んでいる時間の方が長くなるためだ。

次に、matplotlib を用いた 6 時間分の天気コードの推移と、気温の推移を示すグラフを作成した。

図 1 は、web ページ『洗濯判定くん』の UI だ。

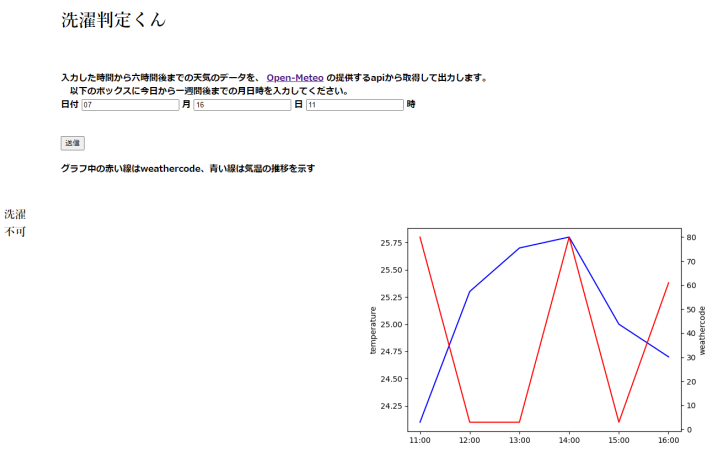

図 1　web ページ『洗濯判定くん』の UI

3-2　プログラムの手法

初めに入力する日付についてだ。これは月、日、時の変数をそれぞれ u_1, u_2, u_3 とし、クエリーを python プログラムに送信する。この変数から、プログラム 54 行目にある変数 search_date に、先ほどの 3 つの変数に加えて先頭には今年の西暦である 2022 を、最後尾には 00 を加え、0 と u_3 との間に" T "を挿入する事で、API に対応する時間指定の文字列を完成させた。

2 番目に、この文字列 search_date と一致する時間から未来 6 時間分のデータを取得する。

プログラム 37 行目に位置する行列『hourly_time』を用いてと一致する時間を探す。この行列は api から取得した時間のリストだ。リストの何番目の要素にあるかを探し、この番地の数を変数 day とした。対応する時間、気温、天気コードの情報の番地は等しいため、この変数 day を用いて 3 つの情報を抜き出す。それぞれ six_hours_time, six_hours_temperature, six_hours_weathercode という 3 つのリストに格納した。

ここで、先月のデータ、来月のデータといった本レポートで予期しないデータが入力された場合について考える。この時、入力されたデータが取得した API の時間のリスト hourly_time に存在するかどうかを判定する。もし存在すれば天気コードをもとにした洗濯判定を行うが、存在しなければ判定結果を示す変数『hantei』に『判定不能』を代入し、web ページ上で表示できるようにした。

3 番目に天気コードを基にした洗濯物を干すか否かの判定を行う。これは 2 番目で得た未来 6 時間分の天気コードの予報を用いる。晴れ、曇り、雨の時間が何回あったかを記録する変数をそれぞれ sunny_count, cloudy_count, rainy_count と置いた。指定した時間に対応する天気コードを読み込み、天気コードに対応し

た変数に 1 を加算する事で各空模様の回数をカウントする事に成功した。このカウントされた変数について、1 回でも雨の時間帯が存在した場合には hantei に『洗濯不可』を、晴れの回数が曇りより多かった場合は『洗濯可能』を、曇りの回数が晴れの回数より多かった場合は『洗濯微妙』を代入し、web 上で表示させた。

図 2 は、判定に用いたプログラムの該当箇所を示す。

```
if rainy_count >= 1:
    hantei = f"<h2>洗濯<br>不可</h2>"
else:
    if sunny_count > cloudy_count:
        hantei = f"<h2>洗濯<br>可能</h2>"
    else:
        hantei = f"<h2>洗濯<br>微妙</h2>"

if 16 <= int(u_3) <= 23 or 0 <= int(u_3) <= 3:
    hantei = f"<h2>洗濯<br>非推奨</h2>"
```

図 2　判定に用いたプログラム

最後に未来 6 時間分の天気予報のデータのグラフを添付した。2 番目で取得した 6 時間分の天気コード、気温、時間のリストについて、時間を x 軸にとり、残り 2 つのパラメータを y 軸にとる。このグラフを、python のモジュール『base64』を用いてエンコードし、それを web ページ上に埋め込む事で実現させた。

図 3 は、作成したグラフを挿入するプログラムだ。

```
tmpfile = BytesIO()
fig.savefig(tmpfile, format='png')
encoded = base64.b64encode(tmpfile.getvalue()).decode('utf-8')

html = '<img src=\'{}\'>'.format(encoded)
```

図 3　作成したグラフを挿入するプログラム

4　プログラムの動作結果

4-1　洗濯可能か否かの判定の出力

web ページ上で 3 つのナンバーボックスに 7 月、16 日、11 時を入力すると、その時刻から 6 時間分の天気データを取得した。そこから判明した事は、この時刻においては、11 時から 16 時までで得られた天気コードのデータは 80, 3, 3, 80, 3, 3 を得た。80 は Open-Meteo を引用すると『Rain showers: Slight, moderate, and violent』、つまり『激しい雨』なため、洗濯は不可能であると判断し、『洗濯不可』の出力が行われた。

図 4 は、7 月、16 日、11 時を入力して得られた洗濯の判定と、その時の天気コードの推移を数値化した出力だ。

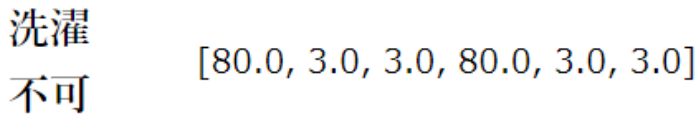

図 4　7 月、16 日、11 時を入力して得られた洗濯の判定と、その時の天気コードの推移を数値化した出力

一方で、ナンバーボックスに 8 月、25 日、9 時を入力すると、その時刻のデータを API はまだ保持していない。つまり API 上の時刻のリスト内に入力した時刻のデータが存在しないため、hantei に『判定不能』が代入され、それを web 上に出力された。

図 5 は、8 月、25 日、9 時を入力して得られた洗濯の判定と、その時の天気コードの推移を数値化した出力だ。

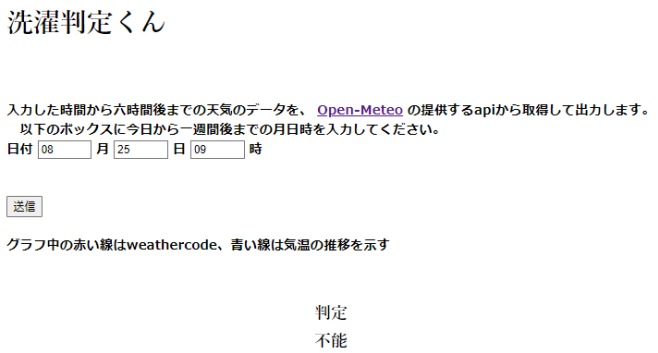

図 5　8 月、25 日、9 時を入力して得られた洗濯の判定、およびその時の洗濯判定くんのページ状況

4-2　天気コードと気温の推移を表すグラフの出力

4-1 と同様の日付データを入れると、得られた天気コードのデータは 80, 3, 3, 80, 3, 3 を、気温のデータは 24.1, 25.3, 25.7, 25.8, 25.0, 24.7 を得た。それと当時に、x 軸を時間の推移、y 軸に気温、天気コードとするグラフも出力された。

図 6 は、7 月、16 日、11 時を入力して得た天気コード、気温の推移を示したグラフだ。

5　課題

動作の結果から、月、日、時を入力したことで洗濯ができるかどうかの出力、及び入力した時間から 6 時間後までの天気コード、気温のデータの推移を表すグラフを出力する事に成功した。しかし、本レポートでは主に 2 つの機能を実装するに至らなかった。

第一に、入力した日時からの一週間の予報のデータから最も洗濯物を干すべき日時の候補を出力する事ができなかった点だ。これは、日中かつ晴れの時間が長い日程から順番に出力すれば実装が可能であると考えた。

第二に、得た api のデータを基にした一週間の天気の変遷を表示できなかった点だ。これは昼の 12 時を基

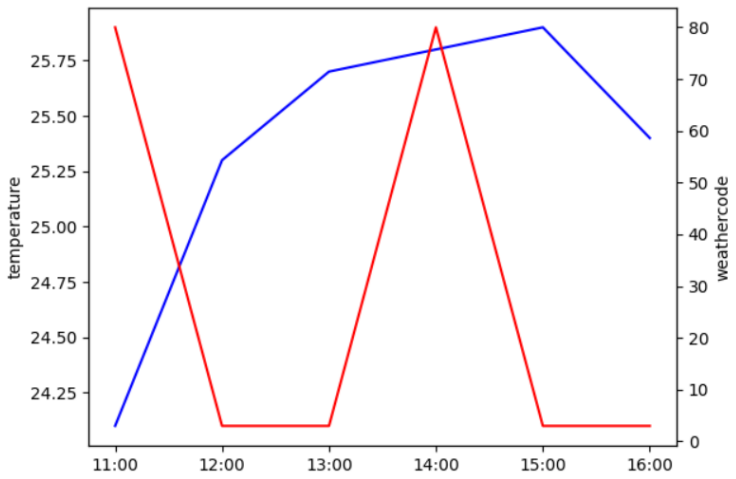

図 6　7 月、16 日、11 時を入力して得た天気コード、気温の推移を示したグラフ

準に、一週間分のデータをグラフ化させる事で実現すると考えた。

他にも、洗濯物を干すか否かの判定を、降水量等の天気コード以外の指標を用いて厳密に算出する事を行いたかったが至らなかった。

6　結論

本レポートは、東京都の一週間の天気に関する web-api を取得し、指定した時間から未来 6 時間分の天気予報のデータを基に、その時間に洗濯物を干す事ができるか判定できる CGI を作成し、その仕様を述べる目的で記した。

Open-Meteo が提供する世界各地域の web-api の情報を使用し、晴れの時間帯、曇りの時間帯、雨の時間帯の数を基に洗濯物が干せるかを判断し、また入力した時間から 6 時間後分の天気コード、気温のデータの推移を示すグラフを出力し、動作結果から正しく動いたことを確認した。

一方で、洗濯物を干すべき時間帯の表示、一週間分のグラフの表示を達成できなかった。これらはそれぞれ daytime による今の時間の検索とそれを基にした入力の阻害、晴れの時間の長い日程を上から表示する事、昼の 12 時を基準としたグラフの作成をする事で解決すると考えた。

結論として、外部 API から天気の情報を取得し、その天気コードを基に洗濯できるか否かを判定する CGI『洗濯判定くん』の妥当性は確かだとし、本レポートの締めとする。

付録

1 ソースコード

```
import datetime
import io
import json
import sys
import requests
import cgi
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import base64
from io import BytesIO

sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')

form = cgi.FieldStorage()

hantei = ""

u_1 = form.getvalue('mon', 0)
u_2 = form.getvalue('day', 0)
u_3 = form.getvalue('hour', 0)

html = ""

if u_1 != 0 and u_2 != 0 and u_3 != 0:
    response = requests.get(
        'https://api.open-meteo.com/v1/forecast',
        params={
            "latitude": 35.6785,
            "longitude": 139.6823,
            "timezone": "Asia/Tokyo",
            "hourly": ["temperature_2m", "relativehumidity_2m", "precipitation", "weathercode"]})
```

```
result = json.loads(response.text)

hourly_time = result["hourly"]["time"]
temperatures = result["hourly"]["temperature_2m"]
kosuiryo = result["hourly"]["precipitation"]
weather = result["hourly"]["weathercode"]

six_hours_time = []
six_hours_weathercode = []
six_hours_temperature = []
six_hours_precipitation = []

if len(u_1) == 1:
    u_1 = f"0{u_1}"
if len(u_2) == 1:
    u_2 = f"0{u_2}"
if len(u_3) == 1:
    u_3 = f"0{u_3}"

search_date = f"2022-{u_1}-{u_2}T{u_3}:00"
if  search_date in hourly_time:
    day = hourly_time.index(search_date)
    for x in range(day, day+6):
        six_hours_time.append(hourly_time[x])
        six_hours_weathercode.append(weather[x])
        six_hours_temperature.append(temperatures[x])

    x_time = []

    for x in six_hours_time:
        temp = x.split("T")
        x_time.append(temp[1])

    sunny_count = 0
    cloudy_count = 0
    rainy_count = 0

    for x in six_hours_weathercode:
        weather_num = int(x)
```

```
            if 0 <= weather_num <= 1:
                sunny_count += 1
            elif 2 <= weather_num <= 3:
                cloudy_count += 1
            else:
                rainy_count += 1

        if rainy_count >= 1:
            hantei = f"<h2>洗濯<br>不可</h2>"
        else:
            if sunny_count > cloudy_count:
                hantei = f"<h2>洗濯<br>可能</h2>"
            else:
                hantei = f"<h2>洗濯<br>微妙</h2>"

        if 16 <= int(u_3) <= 23 or 0 <= int(u_3) <= 3:
            hantei = f"<h2>洗濯<br>非推奨</h2>"

        fig, ax1 = plt.subplots( )
        ax1.plot(x_time, six_hours_temperature ,"b-")
        ax1.set_ylabel("temperature")

        ax2 = ax1.twinx()

        ax2.plot(x_time, six_hours_weathercode ,"r-")
        ax2.set_ylabel("weathercode")

        tmpfile = BytesIO()
        fig.savefig(tmpfile, format='png')
        encoded = base64.b64encode(tmpfile.getvalue()).decode('utf-8')

        html = '<img src=\'{}\'>'.format(encoded)
    else:
        hantei = f"<h2>判定<br>不能</h2>"

style = """
    <style>
    .main{
        display:flex;
```

```
        align-items: center;
        justify-content: center;
        }
    .container{
        display: flex;
        justify-content: space-around;
    }
    h1 { font-family: serif }
    h4 { font-family: sans-serif}
    h2 {font-size: 20px;}
    h2 { font-family: serif}
</style>
"""

template = f"""
<html><head>
    <meta charset = "utf-8">
    <title>この日は東京都で洗濯できるのか</title>
    {style}
</head>
<body>
<div class="main">
    <form action="/cgi-bin/repo/R02.py" method="post">
    <h1>洗濯判定くん</h1>
          <h4>入力した時間から六時間後までの天気のデータを、
        <a href="https://open-meteo.com/en">Open-Meteo</a>
        の提供する api から取得して出力します。
        <br>
          以下のボックスに今日から一週間後までの月日時を入力してください。
        <br>
        <label for="date">日付</label>
        <input type="number" name="mon" value="{u_1}" min="1" max="12" required>
        月
        <input type="number" name="day" value="{u_2}" min="1" max="31" required>
        日
        <input type="number" name="hour" value="{u_3}" min="0" max="23" required>
        時</h4>
        <br>
        <input type="submit" value="送信">
```

```
        <h4>グラフ中の赤い線は weathercode、青い線は気温の推移を示す</h4>
    </form>
    </div>
    <div class="container">
    <p>{hantei}</p>
    {html}
    </div>
</body>
</html>
"""

print("Content-type: text/html\n")
print(template)
```

2 引用

0.0.1 API

・Open-Meteo, Weather Forecast API,

https://api.open-meteo.com/v1/forecast

0.0.2 web ページ

・Open-Meteo, Weather Forecast API,

https://open-meteo.com/en/docs#latitude=35.6785&longitude=139.6823&hourly=temperature_2m