# 14. 佐賀の気候

プログラミング・データサイエンス I

2022/7/14

## 1　今日の目的

今日の目的

- Web ページ内の表を取得
- データをきれいにする
- 作図する

Web ページの中に表がある場合があります。気象庁のページの中には、いくつかの観測地点での、毎月の気温などをまとめた表があります。今回は、以下の URL にある、佐賀市の最高気温の変化を使います。この表を取り出して、毎月の最高気温の変化を表すグラフを作成しましょう。

`http://www.data.jma.go.jp/obd/stats/etrn/view/monthly_s3.php?prec_no=85&block_no=47813&year=&month=&day=&elm=monthly&view=a2`

これまでの講義で使った佐賀県の人口推移の Excel などでは、不要な行の削除などの前処理が必要でした。Web 上の表も前処理が必要となります。

まずは、Web ブラウザを使って上記のページを見てください。一番右に「年の値」という列があります。毎月の変化を図示するには、不要な部分です。

また、2011 年 8 月のデータには、”)”が入っています。2022 年のデータには、更に”]”が入っています。”)”は、観測データ数が完全では無いという意味で使われています。数字でないものは、データとして使えません。何か、対処が必要になります。なお、2022 年は、始まったばかりなので、今回は使わないことにします。

このような前処理を順番に進めていきましょう。今回は、作図できる値になるように一歩一歩進めていきます。それでは、サンプルプログラムを取得してください。

`https://github.com/first-programming-saga/withURL`

# 2　データをきれいにする

データをきれいにする

- HTML から対象となる表を取り出す
- 不要な列を削除する
- 行の内容を確認する
- データをきれいにする

ソースコード 2.1　SagaTenki.ipynb

```
import numpy
import pandas
import matplotlib.pyplot as plt
import japanize_matplotlib
import re

url = 'http://www.data.jma.go.jp/obd/stats/etrn/view/monthly_s3.php?'
url +=
↪  'prec_no=85&block_no=47813&year=&month=&day=&elm=monthly&view=a2'

dataFrames = pandas.read_html(url)
print('表の数 ', len(dataFrames))
for d in dataFrames:
    print(d.head())
```

それでは、SagaTenki.ipynb を開いてください。実行すると、この URL には、二つの表があり、最初が気温の変化、二番目が URL の下のほうにあるリンクであることが解ります。dataFrames[0] が、対象となる DataFrame であることが分かりました。なお、DataFrame の head() というメソッドは、表の最初の部分を表示します。

ソースコード 2.2　「年の値」の列を削除

```
dataFrames = pandas.read_html(url)
df = dataFrames[0].drop('年の値', axis = 1)
df.set_index('年', inplace = True)
df
```

最初に、表の「年の値」の列を削除しましょう (ソースコード 2.2)。drop() メソッドは、指定した行や列を削除した DataFrame を返します。axis='columns'が、列を削除することを表しています。また、「年」の列をインデクスに指定しましょう。実行し、「年の値」の列が無くなっていることを確かめてください。

前述のように、2011 年のデータには、")"が入っていました。他の年はどうでしょうか。例として 2010 年のデータを見ましょう (ソースコード 2.3)。出力結果 (出力例 2.1) をみると、object という型のデータと分かります。

ソースコード 2.3　2010 年の行

```
dataFrames = pandas.read_html(url)
df = dataFrames[0].drop('年の値', axis = 1)
df.set_index('年', inplace = True)
ser = df.loc[2010]
ser
```

```
1 月      9.8
2 月     13.2
3 月     14.6
4 月     19.1
5 月     24.8
6 月     28.3
7 月     31.4
8 月     34.7
9 月     30.3
10 月    24.1
11 月    17.6
12 月    12.0
Name: 2010, dtype: object
```

**出力例 2.1:** 2010 年のデータ

ソースコード 2.4　Series の要素の型を調べる

```
    for k in ser.index:
        v = ser[k]
        print(v,type(v))
```

更に、Series 内の要素の型を確かめましょう (ソースコード 2.4)。出力例 2.2 のようになりました。numpy.float64 は、Pandas が表の中の小数に使う型です。問題なのは、

二つ str 型、つまり文字列が含まれていました。これでは、気温をプロットすることはできません。

```
9.8 <class 'numpy.float64'>
13.2 <class 'str'>
14.6 <class 'numpy.float64'>
19.1 <class 'numpy.float64'>
24.8 <class 'numpy.float64'>
28.3 <class 'numpy.float64'>
31.4 <class 'numpy.float64'>
34.7 <class 'str'>
30.3 <class 'numpy.float64'>
24.1 <class 'numpy.float64'>
17.6 <class 'numpy.float64'>
12.0 <class 'numpy.float64'>
```

**出力例 2.2:** 2010 年のデータの型

Series の値を一括して小数型に変換するには、astype() メソッドがあります。しかし、今回は、2011 年のデータのことも考えて、Series の要素を一つ一つ確かめて、必要に応じて変更することを考えましょう。ソースコード 2.5 を見てください。この部分は、配布ファイルに含んでいます。

ソースコード 2.5　データを数値に

```
1  pat = re.compile(r'(\S*)\s')
2  def cleanSeries(ser):
3      for k in ser.index:
4          v = ser[k]
5          if type(v) is str:
6              if ' ' in ser[k]:
7                  m = pat.match(v)
8                  v = m.group(0)
9              ser[k]=numpy.float64(v)
```

気温のデータの後ろに、スペースが入り、その後に) や] が、現れています。そこで、スペースが現れる前の部分だけを切り出して、数値の型に変換後、上書きすることにします。ここでは、具体的な文字列ではなく、文字列のパターンを探す正規表現というものを使っています。正規表現を理解して使うことができると、テキストを扱うプログラムを書く際の強力な武器になります。興味のある人は、調べてみてください。

1 行目の (\S*)\s が、空白以外の文字の後に空白が続くパターンを表しています。for ループで、Series の要素を一つ一つ取り出し、それが文字列である場合 (5 行目) に処理

します。文字列中に空白があれば、空白の前の部分を取り出します。9 行目で、文字列を numpy.float64 に変換します。このようにすれば、数値が文字列として表現されている場合も含めて、小数の型に変更することができます。ソースコード 2.6 を実行すると、全てが numpy.float64 になっていることが解ります。これで、一番古い 1890 年と最新の 2021 年以外のデータは作図できそうです。

ソースコード 2.6　データの変換

```
1  dataFrames = pandas.read_html(url)
2  df = dataFrames[0].drop('年の値', axis = 'columns')
3  df.set_index('年', inplace = True)
4  ser = df.loc[2010]
5  cleanSeries(ser)
6  for k in ser.index:
7      v = ser[k]
8      print(v,type(v))
```

# 3　作図

作図

- 1 年だけ作図する
- 複数年作図する

最後に作図をしましょう。始めに、1 年だけを作図しましょう。ソースコード 3.1 を見てください。もう、詳しい説明は不要でしょう。9 行目は、横軸に毎月の名前を出すための処理です。これがないと、隔月に名前が出てしまいます。ソースコード 3.1 まで作業したものは、SagaTenkiFinal.ipynb として配布しています。作図結果を図 1 に示します。

**課題 1**　ソースコード 3.1 を参考に、2000 年、2005 年、2010 年、2015 年の最高気温の毎月の変化を作図するプログラムを作成しなさい。また、その動作を確認しなさい。

ソースコード 3.1　2010 年の最高気温変化

```
dataFrames = pandas.read_html(url)
df = dataFrames[0].drop('年の値',axis='columns')
df.set_index('年', inplace = True)

plt.figure(figsize = (15, 10))
plt.rcParams["font.size"] = 32
plt.title('佐賀市の最高気温')
plt.xlim(0, 11)
plt.ylim(0, 40)
plt.xticks(ticks = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])

year = 2010
ser = df.loc[year]
cleanSeries(ser)
ser.plot(label = str(year), linewidth = 3)

plt.legend(loc = 'best')
plt.savefig('SagaTemperature.pdf')
plt.show()
```

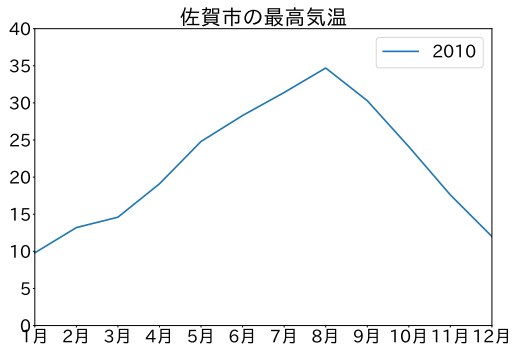


図 1　2010 年の佐賀市における最高気温変化