計算物理学II

第8回
数値計算プログラミング入門（4）：
データの入出力, 可視化

秋山 進一郎

2025年12月5日

授業日の確認

・全10回

- ・第1回：10月3日（金）
- ・第2回：10月10日（金）
- ・第3回：10月17日（金）
- ・第4回：10月24日（金）★
- ・第5回：10月31日（金）
- ・第6回：11月14日（金）★
- ・第7回：11月21日（金）
- **・第8回：12月5日（金）★**
- ・第9回：12月12日（金）
- ・第10回：12月19日（金）★

・★の付いた授業にてレポート課題を配布予定

今日の授業の目標

- Pythonでのファイル操作に慣れる
- matplotlibによるグラフ作成との組み合わせ
- 終わった人は第3回レポート課題へ

本日の演習内容

- Pythonでのファイル操作に慣れる
  - データをデータファイルとして入手し，プログラムに入力する
  - 入力されたデータに対して様々な処理が行えることを理解する
- matplotlib.pyplotを使ってデータの可視化ができるようになる
  - いくつかの図を書いてみる

- 第2回レポート課題時のアンケート結果はこちら

ファイルの入出力の基本

・データファイル(.txt, .dat, .csv, .jsonファイル等)をPythonで扱う手続き

① open関数を使ってファイルを開く

② ファイル中のデータに対する処理を行う

③ 処理が終わったらclose関数を使ってファイルを閉じる

・よく使うデータファイルの種類

・テキストデータ ⇒ 文字だけで構成されたデータ

・バイナリデータ ⇒ 二進数で表現されたデータ (計算機が直接読むことができる)

ファイルを開く時の前提

- ファイルを開く時は, 読み込み用, 書き込み用, 追記用などの用途を指定する必要がある
  - Pythonのopen関数では, 用途を指定しなければ, 自動的に読み込み用になる
  - オプションから指定する
    - r: 読み込み用(デフォルト), w: 書き込み用, a: 追記用
- 扱うデータの形式（テキストデータ or バイナリデータ)も指定しなければならない
  - Pythonのopen関数では, 形式を指定しなければ, 自動的にテキストデータになる
  - オプションから指定する
    - t: テキストモード(デフォルト), b: バイナリモード

ファイルへの書き込みをやってみよう

・右のようなコードを書き，
“ex_write.py”の名前で保存しよう

```
1    fileRef = open("test.txt","w")
2    fileRef.write("Hello World\n")
3    fileRef.close()
```

・コードの中身は以下の通り

・1行目：text.txtという名前のファイルを開く(“w”: 書き込み用オプション指定)
ファイルオブジェクトとしてfileRefがopenから返され，以降コード上では
fileRefを介してデータの処理ができるようになる

・2行目：文字列を書き込む．最後の＼nは改行を与える

・3行目：ファイルを閉じる

・“ex_write.py”を実行してみよう

・実行後，lsで見ると“test.txt”ができているはずである

・“test.txt”をcatで見てみよう(Hello Worldと書いてあれば成功)

ファイルへ追記してみよう

・下のようなコードを書き，“ex_add.py”の名前で保存しよう

```
1    fileRef = open("test.txt","a")
2    fileRef.write("The second line is added\n")
3    fileRef.close()
```

・openの第二引数の“a”は追記モードの意味

  ・すでに存在するファイル(test.txt)にデータを追記したい場合に使うモード

・“ex_add.py”を実行してみよう

  ・実行後，“test.txt”をcatで見てみよう(以下のようになっていればOK)

```
Hello World
The second line is added
```

書き込みモードを使う時の注意点

- すでに存在するファイルを書き込みモード(w)で開くと，そのファイルの内容は削除される
- 下のようなコードを書き，実行してみよう
  - すでに存在するtest.txtを書き込みモードで開き，その後閉じるだけのコード

```
1    fileRef = open("test.txt","w")
2    fileRef.close()
```

- 実行後，“test.txt”をcatで見てみよう
  - test.txtの内容が削除されていることが確認できればOK
- ファイルを書き込みモードで開くと，開いた瞬間にファイルの内容がリセットされることを忘れないように

ファイルを読み込んでみよう 1/2

- 下のようなコードを書き，“ex_read.py”の名前で保存しよう
- lecture8の中のdataディレクトリに“nobel_prize_2020s.txt”というファイルがある
  - このtxtファイルを“ex_read.py”が保存されいるディレクトリにcpしよう
  - catを使って，“nobel_prize_2020s.txt”の中身を見ておこう
- “ex_read.py”の2行目のfor文では，fileRefを1行ずつlineという変数に読み込み，3行目でlineを出力する．openの第二引数“r”は読み込みモード．読み込みモードがデフォルトなので省略してもよい

```
1    fileRef = open("nobel_prize_2020s.txt","r")
2    for line in fileRef:
3        print(line)
4    fileRef.close()
```

ファイルを読み込んでみよう 2/2

- “ex_read.py”を実行してみよう
  - 右のように1行毎に改行が入った出力が得られればOK
    - “ex_read.py”のような形で各行毎の読み込みを行うと，変数lineには改行コードが含まれるため
- line末尾の改行を削除するには，rstrip()を使うとよい
  - “ex_read.py”の3行目を以下のように修正して実行してみよう(改行が消えたことを確認しましょう)

```
1    fileRef = open("nobel_prize_2020s.txt","r")
2    for line in fileRef:
3        print(line.rstrip())
4    fileRef.close()
```

```
2020 Roger Penrose

2020 Reinhard Genzel

2020 Andrea Ghez

2021 Syukuro Manabe

2021 Klaus Hasselmann

2021 Giorgio Parisi

2022 Alain Aspect

2022 John Clauser

2022 Anton Zeilinger

2023 Anne L'Huillier

2023 Ferenc Krausz

2023 Pierre Agostini

2024 John Hopfield

2024 Geoffrey Hinton
```

with構文を使う

- with構文を使うことで，Pythonは自動でファイルを閉じてくれる
  - 明示的にclose()を書く必要がなくなる
- 下のようなコードを書き，実行してみよう
  - 先ほどの“ex_read.py”をwith構文を使って書き直したもの
  - コードブロックが終わった時に，ファイルが自動的にcloseされる
  - ファイルの入出力に際しては，with構文の使用が好ましい

```
1    with open("nobel_prize_2020s.txt","r") as fileRef:
2        for line in fileRef:
3            print(line.rstrip())
```

csvファイル

- csvファイルとは，コンマ(，)でデータが区切られたテキストファイルの総称
  - csv = comma-separated values
- 様々なオープンデータがcsvファイルで提供されている

過去の大学入試共通テストのデータを見てみよう

- lecture8/dataの中にscore.csvというファイルが格納してある
  - 大学入試共通テスト(旧センター試験も含む)のいくつかの科目について，受験者数と平均点が年度毎に収録されている
  - csvファイルの1行目はメタデータで，データに関する情報が記述されている
  - catでscore.csvを見て，ファイル内のデータ構造を確認しよう
    - 各行毎に年度が異なる
    - 各列に異なる項目のデータが記述されている
    - 収録科目は国語，数学IA，数学IIB，物理，化学，生物，地学，英語(reading)，英語(listening)．受験者数に続いて平均点が100点満点換算で記載されている

score.csvからデータを取得するコードを書こう 1/2

・右のようなコードを書き，“score.py”という名前で保存しよう

  ・1~3行目は後で説明, 使用する

  ・5行目以下のコードブロックでscore.csvからデータを取得している

  ・6~9行目の空のリストは後で使う

  ・11行目では，score.csvから取得した各行について，split(“,”)によりカンマで分離したリストfileListを作っている

```
1   import matplotlib
2   matplotlib.use("Agg")
3   import matplotlib.pyplot as plt
4
5   with open("score.csv","r") as fileRef:
6       year = []
7       jpn = []
8       math_ia = []
9       math_iib = []
10      for line in fileRef:
11          fileList = line.split(",")
12          print(fileList)
```

・“score.py”を実行すると12行目により，score.csvの各行毎の要素から構成されたリストが出力されることを確認せよ

・type()を使い，fileListがstr型になっていることも確認しておこう

score.csvからデータを取得するコードを書こう 2/2

- 続いて“score.py”に右のようなコードを書き足そう
  - 13行目は“score.csv”の1行目にある文字列を省くため
  - appendする時にfileListの要素の型をintやfloatに変換しておく
    - fileListは文字型になっていることに注意
- これで，score.csvの特定の列(0, 4, 6, 8列目)の要素のみからなるリストができる
- 以降では，これらのリストを使って図を描いていく

```
1   import matplotlib
2   matplotlib.use("Agg")
3   import matplotlib.pyplot as plt
4
5   with open("score.csv","r") as fileRef:
6       year = []
7       jpn = []
8       math_ia = []
9       math_iib = []
10      for line in fileRef:
11          fileList = line.split(",")
12          print(fileList)
13          if fileList[0] != "Year":
14              year.append(int(fileList[0]))
15              jpn.append(float(fileList[4]))
16              math_ia.append(float(fileList[6]))
17              math_iib.append(float(fileList[8]))
```

matplotlibを使ったPythonによるグラフ作成

- matplotlibはPythonでグラフ等を書く場合に便利なライブラリ
  - もちろんgnuplotを使っても良いが，Pythonでコードを書いているなら図もPythonで作れた方が便利でしょう
  - matplotlibの公式ドキュメントはここ
- score.pyの1~3行目でmatplotlibをimportしていた
  - 3行目でmatplotlib.pyplotにpltという名前をつけてimportしている
  - 1, 2行目は対話型バックエンドが使えない場合に必要
    - 例えば，コード実行時に以下のようなErrorが出たら1,2行目を書けば良い

```
1    import matplotlib
2    matplotlib.use("Agg")
3    import matplotlib.pyplot as plt
```

```
qt.qpa.plugin: Could not find the Qt platform plugin "xcb" in ""
This application failed to start because no Qt platform plugin could be initialized. Reinstalling the application may fix this problem.
```

matplotlibを使ってみよう 1/2

- score.pyにさらに書き足しましょう
  - lecture8/src/score.pyをコピペしても良い
- 各行の意味は以下の通り
  - 19~21行目：x軸にyear, y軸にjpn, math_ia, math_iibを使って折れ線グラフを作る．marker="o"でデータ点に丸シンボルを付す．labelで凡例の名前を設定
  - 22行目：凡例の表示

```
1   import matplotlib
2   matplotlib.use("Agg")
3   import matplotlib.pyplot as plt
4
5   with open("score.csv","r") as fileRef:
6       year = []
7       jpn = []
8       math_ia = []
9       math_iib = []
10      for line in fileRef:
11          fileList = line.split(",")
12          print(fileList)
13          if fileList[0] != "Year":
14              year.append(int(fileList[0]))
15              jpn.append(float(fileList[4]))
16              math_ia.append(float(fileList[6]))
17              math_iib.append(float(fileList[8]))
18
19  plt.plot(year, jpn, marker="o", label="Japanese")
20  plt.plot(year, math_ia, marker="o", label="Math IA")
21  plt.plot(year, math_iib, marker="o", label="Math IIB")
22  plt.legend()
23  plt.xlim(2006, 2025)
24  plt.xticks(range(2006,2025,3))
25  plt.ylim(0, 100)
26  plt.xlabel("Year")
27  plt.ylabel("Average Score")
28  plt.savefig("score.pdf")
```

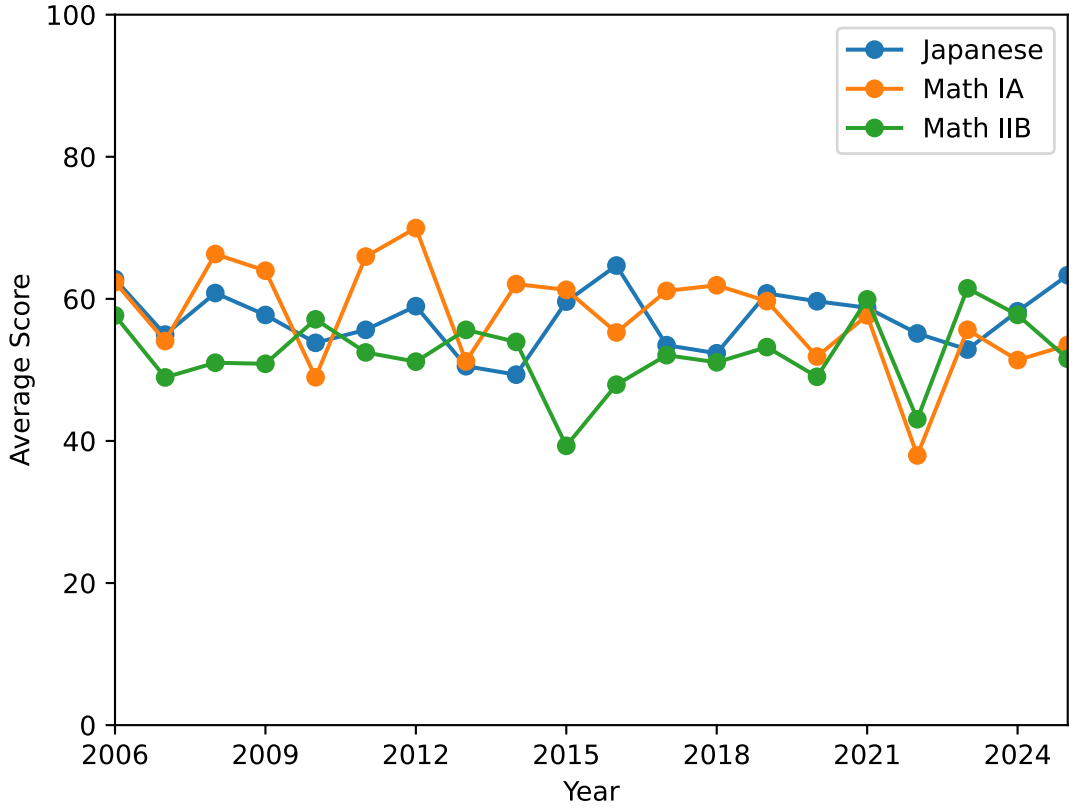
matplotlibを使ってみよう 2/2

- score.pyにさらに書き足しましょう
  - lecture8/src/score.pyをコピペしても良い
- 各行の意味は以下の通り
  - 23~25行目：軸の範囲などの設定
    - 24行目で, x軸の目盛りを設定
    - range(2006, 2025, 3)は開始値2006, 終了値2025, 増分3の整数値のリスト
  - 26~27行目：軸のラベルの設定
  - 28行目：グラフをscore.pdfの名前で保存

```
1   import matplotlib
2   matplotlib.use("Agg")
3   import matplotlib.pyplot as plt
4
5   with open("score.csv","r") as fileRef:
6       year = []
7       jpn = []
8       math_ia = []
9       math_iib = []
10      for line in fileRef:
11          fileList = line.split(",")
12          print(fileList)
13          if fileList[0] != "Year":
14              year.append(int(fileList[0]))
15              jpn.append(float(fileList[4]))
16              math_ia.append(float(fileList[6]))
17              math_iib.append(float(fileList[8]))
18
19  plt.plot(year, jpn, marker="o", label="Japanese")
20  plt.plot(year, math_ia, marker="o", label="Math IA")
21  plt.plot(year, math_iib, marker="o", label="Math IIB")
22  plt.legend()
23  plt.xlim(2006, 2025)
24  plt.xticks(range(2006,2025,3))
25  plt.ylim(0, 100)
26  plt.xlabel("Year")
27  plt.ylabel("Average Score")
28  plt.savefig("score.pdf")
```

完成したscore.pyを実行してみましょう

- 実行後, lsでscore.pdfが出力されたことを確かめ，xdg-openで開いてみましょう
  - 下のような図が描けていればOK



練習問題

- score.pyを修正することで，以下のようなグラフを描いてみよう
  - 国語の受験者数(csvの4行目)の年度毎の推移をグラフに描いてみよう
  - 英語(readingとlistening)の受験者数(csvの18, 20行目)の年度毎の推移をグラフに描いてみよう
  - その他，csvファイルの項目を自由に選んで，その項目の年度推移をグラフに描いてみよう

Collatz問題の履歴をdatファイルに保存してみよう

- 第5回の演習で作成したCollatz問題を解くコードをlecture8にcpし, 右のように修正しよう
  - 入力値が1になるまでの履歴を.datファイルとして書き出すように修正しています
  - 書き出しをしやすくするために, 関数collatzを「nの履歴の格納したリストを返す」ように修正しています
  - 17行目で出力ファイル名を設定
    ⇒ f“…_{init_val}.dat”と書くことで, init_valに代入された値を文字列に変換している
  - 18行目のopen関数はテキストモードになっていることに注意
    ⇒ 21行目でも“i”や“val”を文字列に変換している

```
1   def collatz(n):
2       history = [n]
3       print(n)
4       while n != 1:
5           if n%2 == 0:
6               n = n // 2
7           else:
8               n = 3 * n + 1
9           print(n)
10          history.append(n)
11      return history
12
13  if __name__ == '__main__':
14      init_val = 100
15      history = collatz(init_val)
16
17      filename = f"collatz_{init_val}.dat"
18      with open(filename, "w") as f:
19          f.write("# step value\n")
20          for i, val in enumerate(history):
21              f.write(f"{i} {val}\n")
```

書き出したdatファイルを読み込んで, 可視化しよう

- 先ほど書き出したファイルを読み込み, 横軸をCollatz問題のステップ数(1列目), 縦軸をそのステップ数での値(2列目)とする図を作りましょう
- 各行の意味を確認しながら, 右のようなスクリプトを書いてみましょう
  - 14行目のifブロックは, 出力されたファイルの1行目(#から始まるヘッダー)を無視するため
  - 19行目では, y=1の直線を引いている
    - 収束値が1であることを確認するため
- 書き終わったら実行し, 出来上がったpdfファイルを見てみましょう

```
1   import matplotlib.pyplot as plt
2
3   init_val = 100
4   filename = f"collatz_{init_val}.dat"
5
6   steps = []
7   history = []
8
9   with open(filename, "r") as f:
10      for line in f:
11
12          fileList = line.split()
13
14          if fileList[0] != "#":
15              steps.append(int(fileList[0]))
16              history.append(int(fileList[1]))
17
18  plt.plot(steps, history, marker="o")
19  plt.axhline(y=1, linestyle="--",color="red")
20  plt.xlabel("Step")
21  plt.ylabel("History of the Collatz problem")
22  plt.grid(True)
23  plt.savefig(f"collatz_{init_val}.pdf")
```