第 1 章　　Python プログラミング [1]

データ解析で用いるスクリプト言語とツールを紹介する。

1.1　Python 基礎

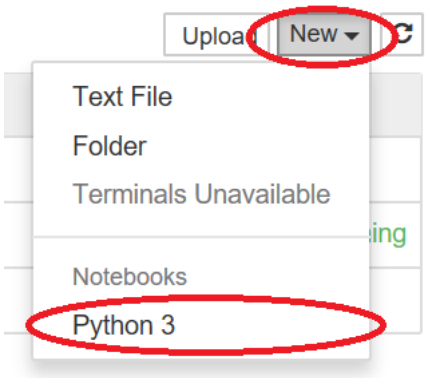
Python は、汎用のプログラミング言語である。コードがシンプルで扱いやすく設計されている。C 言語などに比べて、さまざまなプログラムを分かりやすく、少ないコード行数で書けるといった特徴がある[2]。数値演算や機械学習などのためにデータを処理するライブラリが多く、使いやすいのでデータ解析に適したスクリプト言語である。

1.1.1　プログラムの実行

- コマンドラインに「python」[3]または「ipython」を実行して、スクリプトを入力して、実行する。
- コマンドラインに「python file.py」でプログラムコード file.py を実行する。
- Jupyter Notebook で実行する。今回の演習はこの方法で実施する。

1.1.2　Jupyter Notebook

- コマンドラインに「jupyter notebook」を実行する。ウェブブラウザにタブが自動的に開く(default url: localhost:8888/tree)
- 新しいファイルの作成：「New」→「Notebooks: Python 3」[4]（※注意　ファイルの種類は「Text File」ではない）

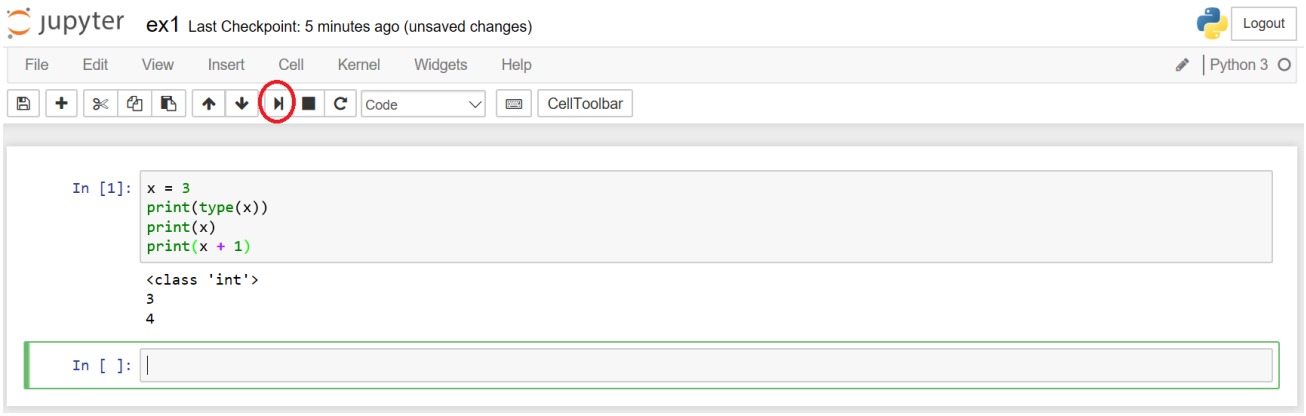


[1] 本資料の 1.1.3, 1.2, 1.3 は下記 URL 先、チュートリアルの一部に基づいて翻訳し作成した。このチュートリアルの作者は Justin Johnson である。このチュートリアルには多くの内容と参考資料がある。時間と興味があれば、参考にしてください。
http://cs231n.github.io/python-numpy-tutorial/#python

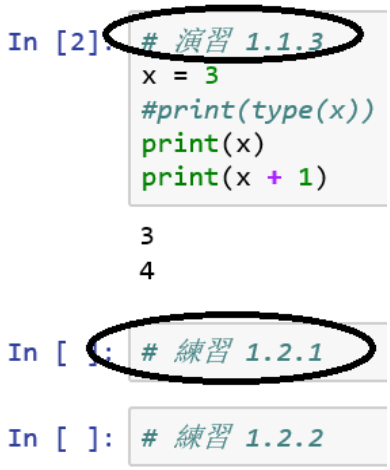
[2] https://ja.wikipedia.org/wiki/Python

[3]「python --version」で Python のバージョンを確認できる。

[4] KKI の環境には現在 python2 がしかない。本資料は python3 に基づいての資料です。python3 は python2 とは異なる。自分のパソコンで勉強する時は python3 で勉強するのがおすすめ。なお、自分のパソコンに環境構築する時はデータ解析および機械学習関連アプリケーション「Anaconda」がおすすめ。



（注意：提出ファイルには以下のようにコメントアウトでセクションを記入してください）



1.1.3　基本データ型

- 数値：integer, float

```
x = 3
print(type(x))
print(x)
print(x + 1)
print(x - 1)
print(x * 2)
print(x ** 2)
x += 1
print(x)
x *= 2
print(x)
```

```
y = 2.5
print(type(y))
print(y, y + 1, y * 2, y ** 2)
# 「x++」と「x--」がありません。
```

- Booleans

```
t = True
f = False
print(type(t))
print(t and f)
print(t or f)
print(not t)
print(t != f)
```

- Strings

```
hello = 'hello'
world = "world"
print(hello)
print(len(hello))
hw = hello + ' ' + world
print(hw)
hw12 = '%s %s %d' % (hello, world, 12)
print(hw12)
s = "hello"
print(s.capitalize())
print(s.upper())
print(s.rjust(7))
print(s.center(7))
print(s.replace('l', '(ell)'))
print('  world '.strip())
```

1.1.4 コンテナ

- リスト: リストは Python の配列と同じですが、サイズ変更可能で、さまざまな型の要素を含むことができる

```
# Basic
xs = [3, 1, 2]
print(xs, xs[2])
print(xs[-1])
xs[2] = 'foo'
print(xs)
xs.append('bar')
print(xs)
x = xs.pop()
print(x, xs)
```

```
# Slicing
nums = list(range(5))
print(nums)
print(nums[2:4])
print(nums[2:])
print(nums[:2])
print(nums[:])
print(nums[:-1])
nums[2:4] = [8, 9]
print(nums)
```

```
# Loops
animals = ['cat', 'dog', 'monkey']

for animal in animals:
    print(animal)

for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))
```

```
# list comprehension
nums = [0, 1, 2, 3, 4]
squares = []
for x in nums:
    squares.append(x ** 2)
print(squares)

nums = [0, 1, 2, 3, 4]
squares = [x ** 2 for x in nums]
print(squares)

nums = [0, 1, 2, 3, 4]
even_squares = [x ** 2 for x in nums if x % 2 == 0]
print(even_squares)
```

- 辞書: 辞書は（key, value）のペアを格納する。

```
d = {'cat': 'cute', 'dog': 'furry'}
print(d['cat'])
print('cat' in d)
d['fish'] = 'wet'
print(d['fish'])
# print(d['monkey'])  # KeyError
print(d.get('monkey', 'N/A'))
print(d.get('fish', 'N/A'))
del d['fish']
print(d.get('fish', 'N/A'))
```

```
# Loops
animals = {'cat', 'dog', 'fish'}
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))

# set comprehension
```

```
from math import sqrt
nums = {int(sqrt(x)) for x in range(30)}
print(nums)
```

- タプル: （不変の）順序付けられた値のリスト。

```
d = {(x, x + 1): x for x in range(10)}
t = (5, 6)
print(type(t))
print(d[t])
print(d[(1, 2)])
```

- 関数:

```
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'

for x in [-1, 0, 1]:
    print(sign(x))
```

```
# オプションのキーワード引数
def hello(name, loud=False):
    if loud:
        print('HELLO, %s!' % name.upper())
    else:
        print('Hello, %s' % name)

hello('Bob')
hello('Fred', loud=True)
```

- クラス: 本授業にほぼ使わないので、今回紹介しません。

1.2 数値計算

1.2.1 Numpy

Numpy は Python での科学計算のためのコアライブラリである。

- 配列

```
import numpy as np
a = np.array([1, 2, 3])
print(type(a))
print(a.shape)
print(a[0], a[1], a[2])
a[0] = 5
print(a)

b = np.array([[1,2,3],[4,5,6]])
print(b.shape)
print(b[0, 0], b[0, 1], b[1, 0])
```

```
#  さまざまな Array の作成
a = np.zeros((2,2))
print(a)

b = np.ones((1,2))
print(b)

c = np.full((2,2), 7)
print(c)

d = np.eye(2)
print(d)

e = np.random.random((2,2))
```

```
print(e)


f = np.arange(15)
print(f)
```

- 配列インデックス

```
# Slicing
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
b = a[:2, 1:3]
print(a[0, 1])
b[0, 0] = 77
print(a[0, 1])


row_r1 = a[1, :]
row_r2 = a[1:2, :]
print(row_r1, row_r1.shape)
print(row_r2, row_r2.shape)
```

- ブール配列インデックス

```
a = np.array([[1,2], [3, 4], [5, 6]])
bool_idx = (a > 2)


print(bool_idx)


print(a[bool_idx])
print(a[a > 2])
```

- データ型

```
x = np.array([1, 2])
print(x.dtype)


x = np.array([1.0, 2.0])
```

```
print(x.dtype)

x = np.array([1, 2], dtype=np.int64)
print(x.dtype)
```

- 配列演算

```
x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# 要素ごとの和
print(x + y)
print(np.add(x, y))

# 要素ごとの差
print(x - y)
print(np.subtract(x, y))

# 要素ごとの積
print(x * y)
print(np.multiply(x, y))

# 内積
print(x.dot(y))
print(np.dot(x, y))

# 要素ごとの除
print(x / y)
print(np.divide(x, y))

# 要素ごとの平方根
print(np.sqrt(x))
```

- 行列の和

```
x = np.array([[1,2],[3,4]])


print(np.sum(x))
print(np.sum(x, axis=0))  # 各列の和
print(np.sum(x, axis=1))  # 各行の和
```

- 転置

```
x = np.array([[1,2], [3,4]])
print(x)
print(x.T)
```

[練習 1.2]：numpy を利用して、実装してください

1. 値が 10 から 49 まで順に並んだ配列を作る
2. 1 次元配列で、3 以上 8 以下の値は負値にする
3. 10×10 の乱整数行列を作って、最大値と最小値を見つける
4. 2 つの配列の中から同じ値を見つける
5. サイズ 10 のランダム配列を作って sort する
6. 行列の各行に対し、その行の平均を引いていく

1.2.2 Scipy

Numpy は、高性能の多次元配列と配列を計算し操作するための基本的なツールである。 SciPy はこの上に構築されており、numpy 配列で動作する多数の関数を提供し、様々なタイプの科学的および工学的アプリケーションに役立つ。
演習時間を考えて、本資料に詳細な紹介がしません。必要の場合、勉強してください。

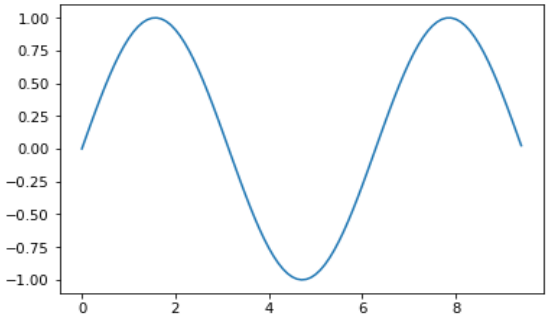
1.3 可視化

Matplotlib は可視化用ライブラリである。matplotlib.pyplot モジュールについて簡単に紹介する。このモジュールは MATLAB と同様の機能を提供する。

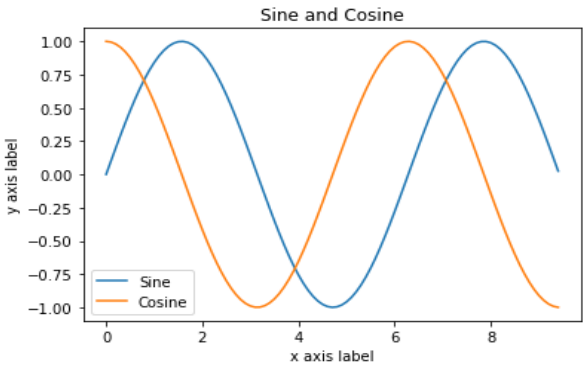
```
import numpy as np
import matplotlib.pyplot as plt


x = np.arange(0, 3 * np.pi, 0.1)
```

```
y = np.sin(x)
plt.plot(x, y)
plt.show()
```




```
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
plt.show()
```



[練習 1.3]：下記の設定を変更する

1. x 軸と y 軸ラベルのフォントサイズ
2. 線幅と線型

1.4　その他

他のライブラリの例

- pandas: データ処理
- scikit-learn: 機械学習
- seaborn: 可視化

1.5 Python 練習

問題を解決するプログラムを Python で作成して、理解を深める。

1.5.1 [練習 1.5.1]

- 所得格差[5]

統計データを処理する際、真っ先に平均値を計算することが多い。 確かに、平均値は、データの傾向を把握するためのよい指標だが、常に最善とは限らない。場合によっては、平均値がデータの理解を妨げることもある。

たとえば、国民所得を考えてみよう。所得格差という言葉が示すように、少数の人が国民所得の大部分を得ている国が多い。 このような場合、所得の平均値は、大多数の国民の所得よりはるかに高くなってしまう。 平均値を典型的な国民の所得と見なすのは間違いである。

以上のような事情を具体的なデータで検証してみよう。 n 人の人の所得 a1, … , an が与えられる。 その中で、平均値 (a1 + … + an) / n 以下の所得の人の人数を答えるプログラムを書いてほしい。

- 入力: n 人の国民それぞれの所得、配列：[a1 a2 … an]

n は整数であり、$2 \leq n \leq 10\,000$ が成り立つ。
ai $(1 \leq i \leq n)$ が i 人目の国民の所得である。 この値は整数であり、1 以上 100 000 以下である。

- 実行例：

==========
入力 1：
15 15 15 15 15 15 15
出力 1：
7
==========
入力 2：
10 20 30 60
出力 2：
3

[5] 2018 年 ACM-ICPC 国内インターネット予選問題 A: http://icpc.iisf.or.jp/past-icpc/domestic2018/contest/all_ja.html

==========
入力 3：
1 1 1 1 1 1 1 1 1 100
出力 3：
9
==========
入力 4：
90 90 90 90 90 90 10
出力 4：
1
==========
入力 5：
2 7 1 8 2 8 4
出力 5：
4
==========

練習（任意）：

- Python ファイル入出力の方法を調査して、入力ファイルを使ってください。

1.5.2 [練習 1.5.2]

- 太郎君の買物[6]

お母さんは太郎に初めてのお買物経験をさせてみることにした。 お母さんは商品カタログから好きなものを二つ選んでいいのよと言うのだが、欲しいものばかりなので太郎は決められなくて困った。 そこで、お母さんが許してくれる金額の範囲で、 合わせた値段が一番高くなる二つの品物を買うことにした。 同じものが二つあってもつまらないから、別々の二つが欲しい。

太郎が二つの品物を選ぶのを助けてほしい。 全商品の価格表が与えられる。 この中の品物二つの組のうち、価格の合計が許容額の範囲で最も高いものを見つけ、その価格の合計を答えよ。 太郎が買う品物の数は二つに決まっていて、一つでも、三つ以上でもいけない。 二つ以上の品物が同じ価格のこともあることに注意せよ。

- 入力: 最大の金額: m; n 個の品物の価格、配列[a1 a2 … an]

n は整数であり、$2 \le n \le 1000$ が成り立つ。
m は整数であり、$2 \le m \le 2{,}000{,}000$ が成り立つ。
ai ($1 \le i \le n$) が i 番目の品物の価格である。この値は整数であり、1 以上 1,000,000 以下である。

- 実行例：

[6] 2017 年 ACM-ICPC 国内インターネット予選問題 A: http://icpc.iisf.or.jp/past-icpc/domestic2017/contest/all_ja.html

==========
入力 1：
45
10 20 30
出力 1：
40
==========
入力 2：
10
1 2 5 8 9 11
出力 2：
10
==========
入力 3：
100
11 34 83 47 59 29 70
出力 3：
99
==========
入力 4：
100
80 70 60 50
出力 4：
NONE
==========
入力 5：
20
10 5 10 16
出力 5：
20
==========

1.6　まとめ

- Python プログラミングについて演習を行った。
    1. Python プログラミング基礎
    2. 科学計算用 numpy
    3. 可視化用 matplotlib
    4. その他ライブラリ
    5. プログラミング練習

- 参考資料
  1. http://cs231n.github.io/python-numpy-tutorial/#python
  2. https://www.kaggle.com/learn/python
  3. http://icpc.iisf.or.jp/past-icpc/domestic2018/contest/all_ja.html

- 宿題
  1. 1.4 のライブラリを調査
  2. Python プログラミングを練習

- Special Thanks
  TA: 三島　大進
  TA: 小宮山　亮太

- 誤字・脱字などを見つけたら
  口頭でもメールでも構いませんので、李 (jyli@yamanashi.ac.jp)まで連絡して下さい。