Python 入門

プログラム言語

python（パイソン）は、様々な用途に利用可能なプログラミング言語の１つである。プログラミングとは、「コンピュータに処理させたい事柄をコンピュータが分かる言葉で指示する行為」のことで、プログラミング言語には機械にとって理解しやすい低水準言語（機械語とアセンブリ言語）と人間の言葉に近い形式で記述できる高水準言語（Fortran や C 言語など）がある。python は高水準言語に分類される。また、高水準言語はコンパイル型とインタプリタ型に分けることができる [*] が、python はインタプリタ型の言語に分類される（python の他にインタプリタ言語で最近統計学の解析によく使われる言語として R（アール）がある。R は後期に開講する医学統計学演習で学ぶ）。

**コンパイラ言語はソースプログラムを機械語に変換するプロセスが必要だが、インタプリタ言語は１ステップ（１行）ごとに機械語に解釈しながら実行してくれる。*

python は様々な OS に対してフリーでインストールできるインタプリタ型の高水準言語（スクリプト言語）である。Matlab（マトラボ）などの有償のプログラミング言語とは対照的に、python をマスターしておけば PC さえあればプログラミングを行う環境を無償で構築することができる。また、これまで科学技術計算において中心的なプログラム言語であった C 言語や Fortran に比べ直感的に使うことができ、プログラミング初級者が最初に学ぶ言語として最適なものの１つに挙げられている。科学技術研究分野で発展してきた C 言語や Fortran で書かれたプログラム（ライブラリ）を網羅的に組み込んでおり、C 言語や Fortran を理解することなしにそれらを簡単に利用することができるようになっている点も、初級者にとってはありがたい [**]。近年では Google の研究グループが開発した Tensorflow というディープラーニング系のライブラリが公開され、利用者が爆発的に増えている。量子コンピュータも python を使って簡単に動かすことができるようになっている。

****軽い気持ちでプログラミングを始められるという良い面もあるが、中身を理解していないのにわかった気になるユーザーが出てきてしまう悪い面もある。**

python が使えるようになると高度な実験解析が可能になる。医療の分野でも広く使われており、学部の間に基礎を習得しておくことで就職先や大学院進学後にアドバンテージを得ること間違いないでしょう。Windows, Mac, Linux など様々な OS 用の python 実行環境が提供されているので、自分が所有するパソコンには必ず入れておきましょう。

なお、栄華を極めている python だが、python に代わるプログラム言語が出現することも歴史は否定しておらず、10 年後, 20 年後も同じ状況であるかはわ

からない。今後 50 年は大丈夫、と言える言語は C 言語くらいでしょう。本気でプログラムを勉強したい人は C 言語（と C++）を学ぶことを強く勧めます[***]。C 言語はほとんど全てのコンピュータで使うことができ、コンピュータの最もコアな部分にも使われている言語の 1 つであり、コンピュータがなくならない限りなくならないでしょう。例えば、今後の発展によって世界を大きく変えると想像される量子コンピュータが普及するような時代になったとしても、まずは C 言語との互換性をルール化すると考えられます。C 言語に関しては、みなさんがリタイアするくらいまでなら残っていることでしょう。もっと不変な言語を望む人は物理学を真剣に学びましょう。物理学を押さえておけば、（理論上には）原理から出発してどのような科学技術の発展にも対応できる能力を獲得できます（そして、プログラム言語の多くは物理学の研究の中で必要となったために物理屋が開発に関わってきました）。

****深層学習を一般の人でも使えるように公開された Tensorflow も、大元は C 言語に近い CUDA 言語（Graphycal Processing Unit; GPU で並列化計算を容易にできるようにした C 言語をベースにした言語）で書かれており、それを python で実行できるようにしています。なので、もし自分で Tensorflow の発展バージョンを大元から作ろうと思ったら、CUDA 言語を勉強しないといけません。*

本講義では、学生が自分のノート PC を使って python のコードを走らせます。そのための準備として python のインストールおよびプログラムを書くためのエディタ（Editor）のインストールが終わっていることが前提で進めていきます。python は、windows では anaconda prompt、mac ではターミナルを使って動かせるか確認してください。また、エディタとして Visual Stadio code が起動できるどうかも確認してください（これらは 1 年生のデータ科学入門で既に設定されているはずです）。

Hello world! in python

python はインタプリタ言語です。インタプリタ言語であることをまずは実感してもらいます。Mac の人はターミナルを立ち上げて、Windows の人は Anaconda Prompt を使って、次のように打ちエンター（Enter）キーを押しましょう。

```
python --version
```

バージョンが 3 以上であればこのまま以下を続けてください。

```
python
```

すると、“»>” というようにコンソール画面に変わるかと思います。今、python の世界にいるということを意味します。そこで 1+1 と打ちましょう。

```
>>> 1+1
```

2 という答えが返ってきましたか？ このように、１行ずつ解釈して実行してくれるプログラム言語がインタプリタ言語です。C 言語ではコンパイルという作業が必要で、これは人間が理解できる内容（プログラム）を一旦機械語に翻訳する作業です（機械語で書かれているのが実行ファイルが生成される）。python は翻訳作業も一気にやってくれるため、プログラム初級者にはすごくわかりやすく扱いやすい言語になっています。Hello world!という文字列を返してくれる操作は、

```
>>> print("Hello world!")
```

でいけてしまいます。しかし、１行１行打って作業をするのはしんどいです。ファイルにまとめて書いたものを一気に実行してくれるとありがたい時がありますし、編集や改変も簡単で便利です。「実行内容をファイルにまとめて書いたもの」をスクリプトと呼びます。次に、それを作成し動かしてみます。そのために一旦 python から抜けます。

```
>>> exit()
```

スクリプト（script）

では、Hello.py という名前のファイルをダウンロードしましょう。エディタ（Visual Stadio code をインストールしていればそれを使いましょう。他のエディタ、例えば Emax や Mac の場合は Xcode などでも OK）を開いて中身を確認してください。

（*Hello.py* ファイルの中身）

```
print("Hello world!")
```

確認したら、ターミナルで次のように打ちましょう。

```
python Hello.py
```

Hello world!が表示されましたか？「python Hello.py」として Enter を打つことを「(python プログラムもしくはスクリプトの）実行 (execute)」と言います。python **** と打つことで****に書かれたファイルの内容（スクリプト）が上から順に実行されます。スクリプトを使うことでいろいろなことができるようになります。例えば Hello.py を次のように編集してみましょう。

（*Hello.py* ファイルの中身）

```
x = 10
y = 20
```

```
x = x + y
print("Hello world!: x = ", x)
```

保存したら、また次を実行しましょう。

```
python Hello.py
```

何が出力されたでしょう？ どうしてそのように出力されたのか、説明できますか？ 同じ結果は以下でも得ることができます。

（*Hello.py* ファイルの中身）

```
x = 10
y = 20
x += y
print("Hello world!: x = ", x)
```

このように、インタプリタ言語でファイルに実行する手順を書いて実行することをスクリプト処理という時があります。そのファイルをスクリプトと呼んでいます。python ではスクリプトを作成することがプログラミングするということです。

高度なスクリプトを作成するためにはいくつかのルールを覚えなければなりません。このルールを本講義で少しずつ学ぶことになりますが、まずは for 文, while 文、if 文を取り上げたいと思います。

```
for i in range(10):
    x += i
    if i % 2 == 0: # i//2*2 == i でも良い
    y += i
print("(x, y) = (", x, ",", y, ")")
```

何が出力されたでしょうか？ なぜそのような出力になったのかを考えましょう（for の後や if の後はインデント（タブ）を必ず入れる。#はコメントアウト、その行の以降に書かれている文は実行時に無視される）.

また、

```
for i in range(10):
```

を

```
i = 0
while i < 10:
    i += 1
```

と変えてみてください。これは i が 10 未満である限りループを続けることになり、for 文のように繰り返し処理に使うことができます。

データの入出力

データの入出力ができるようになると、やれることが拡がります。1. データを入れて、2. 学習させて、3. 予測結果を返す、という AI の基本構造の最初 (1) と最後 (3) の部分です。出力をさらに解析にかける、ということも頻繁に行うため、出力データの設計もよく考えてやらなければなりません。この講義では、csv ファイルの入出力について学びます。任意のテキストファイルや画像の入出力についてはのちに必要に応じて学ぶこととします。

output.csv をダウンロードして自分のいるディレクトリにおいてください。この csv ファイルを読み込み、２列目と３列目の和を 4 列目に追記してその結果を output2.csv という名前で出力してみます。（カンマ区切りで書かれた）csv データの入出力や編集には pandas というライブラリを使うと便利です。この機能を使うために import pandas as pd と最初に宣言します。次のようにファイル（input_output.py）を作成しましょう。

(input_output.py **のファイルの中身***)*

```
import pandas as pd

df = pd.read_csv("output.csv", encoding="SHIFT-JIS")
print(df)
df["total"] = df["sin_2px"]+df["noise"]
print(df)
df.to_csv("output2.csv", index=None, encoding="SHIFT-JIS")
```

出力結果はどうでしたか？ output2.csv が新たに生成されたでしょうか？（実行時に pandas のエラーが出た人は, python ではなく python3 で実行してみること）

pandas で事が足りればこれだけで済みますが、pandas ではないデータ形式を出力したくなる時があります。よくあるのは、数値計算で使われる Numpy 形式で書かれたデータを出力することです。その場合も本演習では常に pandas のデータフレームに直してから pandas を使って csv ファイルで保存しましょう（これらの相互変換については、必要性が生じたときに説明します）。

グラフの書き方

まずはインタラクティブに output.csv を使ってグラフを生成してみたいと思います。python を起動します。

```
python
```

続いて、ライブラリをインポートします。pandas のほか、グラフ作成用に matplotlib.pyplot というライブラリを使います。

```
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
```

では、データを読み込みましょう。

```
>>> df = pd.read_csv("output.csv",encoding="SHIFT-JIS")
>>> df
```

output.csv のファイルの中身が出力されましたね？ では、横軸を項目 x、縦軸を項目 sin_2px として散布図を描いてみます。

```
>>> plt.scatter(df["x"], df["sin_2px"], color="blue",label="sin_2px")
>>> plt.show()  # 図の出力
```

グラフが出力されましたか？ plt.scatter の代わりに plt.plot を使うとデータ間を直線でつなぐことができます。

```
>>> plt.plot(df["x"], df["sin_2px"], color="blue",label="sin_2px")
>>> plt.show()  # 図の出力
```

オプションで marker を使うと、データ点を表示させることができます。

```
>>> plt.plot(df["x"], df["sin_2px"], marker="o", color="blue",label="sin_2px")
>>> plt.show()  # 図の出力
```

ラインの幅や種類はオプションの lw や ls を使う。

```
>>> plt.plot(df["x"], df["sin_2px"], marker="o", ls=":", lw=4, color="blue",label="sin_2p
>>> plt.show()  # 図の出力
```

プロットの範囲は plt.xlim と plt.ylim を使う。凡例は plt.legend、レイアウトの自動調整 plt.tight_layout() を入れてプロットする。

```
>>> plt.plot(df["x"], df["sin_2px"], marker="o", ls=":", lw=4, color="blue",label="sin_2p
>>> plt.xlim(-2,2) # x の範囲
>>> plt.ylim(-2,2) # y の範囲
>>> plt.legend() # レジェンドを追加
```

```
>>> plt.tight_layout() # レイアウトの自動調整（常に書いておくと良い）
>>> plt.show()  # 図の出力
```

練習１：項目 sin_2px の代わりに noise を y 軸にとり、$x = [-1, 1]$, $y = [-1, 1]$ の範囲で散布図を作成せよ。

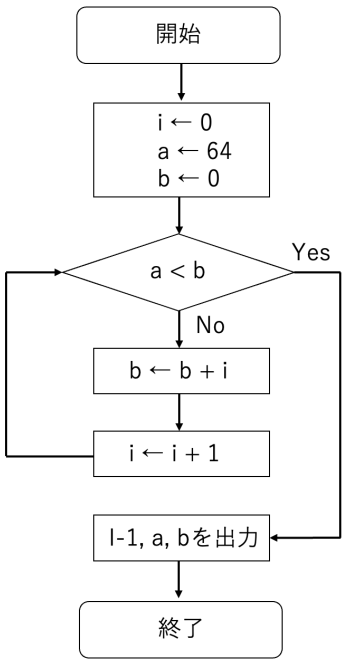
複数のグラフを書くには、plt を続けて入力し、最後に plt.show() で出力させれば良い。\

```
>>> plt.plot(df["x"], df["sin_2px"], marker="o", ls=":", lw=4, color="blue",label="sin_2p
>>> plt.scatter(df["x"], df["noise"], color="red",label="noise")
>>> plt.show()  # 図の出力
```

練習２：上記のグラフを出力するスクリプトを作成せよ。

演習レポート

- 図に示されるフローチャートをプログラムで作成してみよう。manaba のレポートに、作成したスクリプトを貼り付けて解説をしてください。



演習問題：このチャートをプログラム化せよ！