令和３年度 授業改善セミナー
～基礎プログラミング ・ データ分析（Python）～

北海道帯広緑陽高等学校　宮川尊充

2021 年 10 月 26 日・11 月 2 日　Version 1.1

目次

0.1 プログラミング環境の準備

プログラムの実行環境は，GoogleColaboratory を利用します。Colaboratory（略称: Colab）とは，Jupyter Notebook（プログラムの作成，実行結果，グラフ，作業メモや関連するドキュメントなどを，ノートブックで一元的に管理できるサービス）のクラウド版のようなアプリケーションです。インターネット環境と Web ブラウザがあれば， Python の記述やプログラムを実行できるサービスです。メリットとしては，環境構築が不要（Google アカウント必要），Google の GPU も無料で利用可能，共有や共同編集もすることも可能などが挙げられます。

ノートブックは，**コードセル**と**テキストセル**で構成されています。**コードセル**は，プログラムを記述して実行することができ，**テキストセル**は，リッチテキスト（画像、HTML、LaTeX なども可）が記述でき，Markdown（文書を記述するための軽量マークアップ言語のひとつ）が利用可能で，手軽に記述することができます。

0.2 プログラミングの環境設定

Colaboratory の「設定（右上の歯車)」 をクリックし，「エディタ」を選択，以下の２点の設定をします。

- インデント幅を「２」から「４」に変更します。
- 行番号の表示を可能にするため，「チェック（ON)」します。

0.3 プログラミングの基本

プログラミングの基本構造は，大きな括りとして「**入力**したものを**処理**して，結果を**出力**する」というものです。

入力：数や文字（列）といった値（データ）を扱うために，変数に代入しておくこと。

処理：変数を使って計算を行うこと。

出力：処理結果をユーザが確認できるように出力すること。

0.4 プログラミングでの処理

プログラムで処理を記述する場合，その処理は大きく捉えると 3 種類あります。

順次処理：プログラムを記述した順から実行する処理のこと。

条件処理：条件が成り立つ場合とそうではない場合に応じて，それぞれの実行が分かれる処理のこと。

反復処理：同じ処理を繰り返すのこと。一定回数だけ繰り返す，ある条件が成り立つまで繰り返す処理のこと。

0.5 コーディング実施についての設定等

0.5.1 コードセル

Colaboratory では，コードセルと呼ばれるセルに，プログラムを記述します。プログラムの実行は，実行ボタンを選択するか，Ctrl + Enter を同時に押します。

0.5.2 コメントアウト

記号＃は，コメントを示します。# がついた行は，プログラムの実行には何も影響を及ぼしません。なお，プログラムの先頭に付けることで，その行を実行させないこともできます。

```
# print('Hello')  ←  # があるので実行しません
```

0.5.3 空白の扱い

半角スペースは無くても実行，判定に問題はありません。 例えば，下記のプログラムはどちらでもプログラムとしては問題なく動きます。半角スペースは，x = 1 や，for in in range(1, 5, 1) のように，等号＝の両側やカンマ，の後に半角の空白が１文字ずつ挿入されています。

a = 1 + 3 （半角スペースあり）　　a=1+3 （半角スペースなし）

0.5.4 処理ブロック

if 文や for 文など内部に別の文を持つコードは複合文と呼ばれます。他の文を内部に持ち，内部の文を実行するかどうかなどの制御をコントロールするもので，インデント（字下げ）によって，一つのブロックを表現します。Python では，複合文のヘッダー行の終わりには，コロン: が必須とされています。よって，コロン: がブロック処理の合図となっています。

0.6 Python で利用する演算子について

Python で利用する演算子は，表１と表２にある通りです。

表１ Python 算術演算子

算術演算子	記述	説明
＋	a ＋ b	足し算
−	a − b	引き算
*	a * b	掛け算
/	a / b	割り算
//	a // b	a を b で割った商の整数値
**	a**n	a を n 回掛けた数（べき乗）

表２ Python 比較演算子

比較演算子	記述	説明
＞	a ＞ b	A は B より大きい
＜	a ＜ b	A は B より小さい
＞=	a ＞= b	A は B 以上
＜=	a ＜= b	A は B 以下
==	a == b	A と B は等しい
!=	a != b	A と B は等しくない

0.7 データの型

データの型は，変数名を付けた後に，変数に対して値を設定する処理をします。Python では，その際に代入した値によって変数のデータ型（数値，文字列など，どのようなタイプのデータか）が決まります。また，関数の処理によって，データの型を変換することも可能です。

表３ 利用度が多いデータの型種類

データの型	種類	種類
int 型	整数	整数の数値を扱う型
float 型	小数点	小数点を含む数値を扱う型
str 型	文字列	文字、記号、数値など文字を扱う型
bool 型	真偽値	真か偽の値を扱う型
list 型	リスト	インデックス番号により，数字や文字列など複数のデータを扱う型で，データの変更が可能
tuple 型	タプル	インデックス番号により，数字や文字列など複数のデータを扱う型で，データの変更が不可能
dict 型	辞書	キーをもとに値を扱うデータの型
set 型	集合	集合を扱うためのデータの型

0.8 コーディングについて

コードセルの 1 行目には、「# 1.1.1」のように、セクション番号を記載しています。
そのため，例題のプログラミングは、2 行目から記述していきます。 下記掲載のプログラムを参照してください。

```
# １．１．１　←セクション番号
a = １　　←ここからコーディング
b = ２
```

1 Python を活用した基礎プログラミング

1.1 基礎プログラミング

1.1.1 順次処理（変数の利用-数値型）

```
# 1.1.1
x = 3
y = x + 1
print(y)
print(x + y)
```

1.1.2 順次処理（変数の利用-文字型）

```
# 1.1.2
a = 'Hello!'
b = input('name:')
print(a, b)
```

1.1.3 順次処理（変数の利用-リスト型）

```
# 1.1.3
weeks = ['mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun']
print(weeks)
print(weeks[5])
print(weeks[2:5])
print(len(weeks))
```

1.1.4 条件処理

if ... elif ... else :　を利用して条件処理を実行します。
if 文で処理を記述するには，コロン ： とインデントによる処理ブロックが必要になります。(0.5.4 参照)

```
# 1.1.4
n = int(input('1~9 を入力する：' ))

if n < 5:
    print('small')
elif n < 8:
    print('midle')
else:
    print('large')
```

１～９以外が入力された場合のために，エラー回避を記述する。

```
# 1.1.4'
n = int(input('1~9 を入力する：' ))

if 1 <= n <= 9:
    if n < 5:
        print('small')
    elif n < 8:
        print('midle')
    else:
        print('large')
else:
    print('error')
```

if 1 <= n <= 9:　　< − − − − − − − >　　if n >=1 and n <= 9:

1.1.5　反復処理１

for 文により，**反復回数**を指定します。

for 文は，range 関数と組み合わせて利用するのが一般的です。

range 関数は，指定した開始数から終了数までの連続した数値を要素として持つことができます。

```
# 1.1.5
start = 30
stop = 100
step = 10

for i in range(start, stop + 1, step):
    print(i)
```

1.1.6　反復処理２

while 文により，**反復条件**を指定します。

while 文を活用して，1 から 10 までの自然数の合計を表示します。この処理には，数を数える変数や合計を蓄積する変数が必要となります。これらの変数の処理は，初期化（ゼロクリア）を設定した変数を利用します。

初期化を設定する変数：カウントする変数，合計をする変数

```
# 1.1.6
cnt = 0
sum = 0
while cnt < 10:
    cnt = cnt + 1
    print(cnt)
    sum = sum + cnt
print('gokei = ', sum)
```

1.2 データの可視化（グラフ化）

グラフ作成にはモジュールが必要となります。モジュールとは，Python の処理を記述したファイルのことを指します。そして，そのモジュールをいくつかの機能ごとにまとめたものがライブラリと呼ばれます。

「 from matplotlib import pyplot as plt 」,「 import matplotlib.pyplot as plt 」は,

「matplotlib という ライブラリ から，pyplot という モジュールを利用するための準備をする。pyplot という名前は長いので plt として，利用する。」ということになります。

1.2.1 箱ひげ図の作成

```
# 1.2.1
from matplotlib import pyplot as plt
data = [37,42,42,43,54, -------- ,81,70]
plt.boxplot(data, labels = ['camp_trend2020'])
plt.yticks([30,40,50,60,70,80,90,100])
plt.grid('on')
plt.show()
```

1.2.2 箱ひげ図 (複数) の作成

おにぎり製造機を購入しようと考えている。おにぎりの量を 120g なるように製造するＡ社，Ｂ社，Ｃ社の機械のうち，どれかの購入を検討している。次のデータは，それぞれの機械が作ったおにぎりの１０個の重さ（単位はｇ）を正確に測定し，値が小さい方から並べたものである。各社の機械の平均値はともに 120 g であった。どの機械を購入すべきか箱ひげ図を活用して説明しよう。

```
# 1.2.2
import matplotlib.pyplot as plt
a = [117,117,118,118,119,120,121,121,124,125]
b = [117,118,118,118,120,121,121,122,122,123]
c = [116,119,120,120,120,120,120,121,122,122]
plotdata = [a, b, c]

plt.boxplot(plotdata, labels= ['A', 'B', 'C'] , widths=[0.5, 0.5, 0.5], vert=
    False)
plt.title('onigiri')
plt.grid('on')
plt.show()
```

plt.boxplot(plotdata) は，plt.boxplot([a, b, c]) でも記述が可能です。boxplot の引数を，[ ] でまとめるか，事前にリストに格納しているかの違いです。

1.2.3 ヒストグラムの作成

ヒストグラムを作成するには，どのような度数分布で作成するかを決めます。特に，変量の数や幅から階級と階級幅を設定する必要があります。

最小値：37，最大値：100　==>　…30…40…50……90…100　==>　階級幅：10　階級：7

```
# 1.2.3
from matplotlib import pyplot as plt
data = [37,42,42,43,54, -------- ,81,70]
bins = 7
plt.hist(data, bins, range=(30, 100), edgecolor='w')
plt.show()
```

1.2.4 散布図の作成

東京都の「キャンプ」というキーワードのトレンド指数の変化に，影響を及ぼす要因を考察する。

利用データ：GoogleTrend　キーワード：キャンプ　，　データ期間：2020 年週毎　，地域：東京

→　その年 (2020 年) の 東京都の週別最高気温との関係を調べてみる。

```
# 1.2.4
from matplotlib import pyplot as plt
temp_data = [12.2,14.9,14.1,18.6,17.5, -------- ,14.1,14]
trend_data = [31,30,30,36,37, -------- ,29,31]
plt.scatter(temp_data, trend_data)
plt.show()
```

1.2.5 回帰直線

回帰直線を表示するには，NumPy ライブラリを活用します。

NumPy は，Numerical Python の略称で、Python の数値計算のためのライブラリです。数値計算のための関数が多数あるのが特徴です。

データの関係性を回帰直線で表すには，データから傾きと切片を算出する必要があります。そのため，データ構造を np.array などの配列やデータフレームに変換する必要があります。

list

- 組み込み型
- 異なる型を格納できる (要素ごとにデータ型を柔軟に変えられる)
- データの集合（順番通りにデータを格納せず，格納するメモリアドレスを記録する）

np.array

- 同じ型しか格納できない（数値型を基本としている）
- 多次元配列を表現できる
- 数値計算のためのメソッドの利用や高速な演算が可能
- 順序付けられたデータ構造（同じ型のデータが規則的に並んでいる）

```
# list ==> array
import numpy as np

# list
a = [1,2,3]
b = [10, 20, 30]
print('list:', a + b)

# np.array
x = np.array([1,2,3])
y = np.array([10, 20, 30])
print('array:', x + y)
```

回帰直線を求めるには，NumPy ライブラリの polyfit 関数を利用し，x と y のデータから傾き（a）と切片を（b）を求めます。polyfit 関数は，y のデータに対して，最適な近似となる n 次の多項式の係数を返します。

```
# 1.2.5
from matplotlib import pyplot as plt
import numpy as np
temp_data = [12.2,14.9,14.1,18.6,17.5, -------- ,14.1,14]
trend_data = [31,30,30,36,37, -------- ,29,31]
plt.scatter(temp_data, trend_data)

# list ==> array
x = np.array(temp_data)
y = np.array(trend_data)

# data check
print(type(temp_data), type(trend_data))
print(type(x), type(y))

# data set
a, b = np.polyfit(x, y, 1)
y = a * x + b
print('y=', a, 'x+', b)

# graph set
plt.plot(x, y, c='g')
plt.grid('on')
plt.show()
```

1.2.6 考察

Google 検索におけるキャンプというキーワードについて，最高気温が 5 ℃上昇したときの，東京都の Google トレンド指数の変化を調べましょう。

1.2.5 で求めた回帰式「 y= 1.959815276250672 x+ 1.9762089669280039 」を活用します。そのため，1.2.5 が実行されていることが前提のプログラムとなります。

```
# 1.2.6
# 20 ℃==> 25 ℃
x1, x2 = 20, 25
y1 = a * x1 + b
y2 = a * x2 + b
print(x1, ' ℃：' , y1)
print(x2, ' ℃：' , y2)
print(y2 - y1)
```

1.2.7 関数の利用

入力した気温に対するトレンド指数を表示します。(1.2.5 が実行されていることが前提)

```
# 1.2.7
def henka(x):
    y = a * x + b
    print(x, ' ℃：' , y)
kion = int(input('kion:'))
henka(kion)
```

2　データの可視化

2.1　データの準備

2.1.1　オープンデータの利用

今回は，Google トレンドのトレンド指数を１つの csv データにまとめて利用します。

2.1.2　ライブラリの活用

Python には，処理を記述したファイル（モジュール）があります。そのモジュールをいくつか機能ごとにまとめたものがライブラリとよばれます。

今回の Python によるデータの可視化では，以下のライブラリを利用します。

・ Pandas

Pandas ライブラリにはデータ分析作業を支援するためのモジュールが含まれています。
Pandas モジュールは, リストや配列, 辞書などのデータをシリーズ (Series) あるいはデータフレーム (DataFrame) のオブジェクトとして保持します。シリーズは列, データフレームは複数の列で構成されています。よって, Pandas ライブラリは，複数列の csv データをデータフレームとして読み込むことができます。

・ matplotlib

matplotlib ライブラリには，グラフ描画するモジュールです。

・ seaborn

seaborn ライブラリは、matplotlib の内包プログラム（matplotlib が内部で動いいる）ライブラリで，データを可視化します。データフレームを扱うことができます。

2.1.3　Pyhton による データの扱い方について

Colaboratory で csv データを扱うには，プログラムの記述によりデータをアップロードして読み込む方法と，サイドバーからのデータをアップロードして読み込む方法との大きく２つの方法があります。

１．プログラムにより，GoogleDrive をマウント
２．プログラムにより，ファイル（ローカルに保存されているファイル）のアップロード
３．サイドバーから，GoogleDrive マウント（サイドバーにあるフォルダから、GoogleDrive にアクセス）
４．サイドバーから，ファイル（ローカルに保存されているファイル）のアップロード

2.2　csv データの読み込み・表示

2.2.1　プログラムによるファイル（ローカルに保存されているファイル）のアップロード

```
# 2.2.1
from google.colab import files
uploaded = files.upload()
```

2.2.2　データの読み込み

```
# 2.2.2
import pandas as pd
df_trend = pd.read_csv('---filepass-paste---', header = 0)
df_trend = df_trend.dropna()
df_trend.head()
```

2.3 基本統計量・相関係数の表示

2.3.1 基本統計量（要約統計量）

describe() を利用すると，各列ごとに平均や標準偏差，最大値，最小値，最頻値などの基本統計量を取得することができます。

```
# 2.3.1
df_trend.describe()
```

2.3.2 相関係数の表示

データフレームにある相関（correlation）係数の算出には，corr() メソッドを活用します。
corr() メソッドでは，データ型が object（文字列）の列は除外され，数値（int, float）型および bool 型の列の間の相関係数が算出されます。

```
# 2.3.2
df_corr = df_trend.corr()
print(df_corr)
```

DataFrame のデータを総当たりで比較していることがわかります。

2.4 csv データの可視化

seaborn ライブラリの pairplot (行列散布図) を利用して，データを可視化します。

```
# 2.4
import matplotlib.pyplot as plt
import seaborn as sns
sns.pairplot(df_trend)
plt.show()
```

2.5 相関係数の可視化

seaborn ライブラリの heatmap(ヒートマップ) を利用し，2.3.2 で算出した相関係数を可視化します。

なお，seaborn.heatmap で利用する引数は，以下の通りです。
annot　=>　True に設定すると，セルに値を出力します。
fmt　=>　出力フォーマットを指定します。(.2f　小数点第 2 位まで表示します)
cmap　=>　カラーマップの色の指定します。
square　=>　True に設定すると，X 軸，Y 軸を正方形になるように調整します。

fig = plt.figure(figsize=(横サイズ, 縦サイズ))　=>　グラフを描画するキャンバスの準備します。
context(' 文字サイズの指定')　=>　文字サイズ：paper < notebook < talk < poster の順に大きくります。

```
# 2.5
import matplotlib.pyplot as plt
import seaborn as sns
fig = plt.subplots(figsize=(10, 10))
sns.set_context('talk')
sns.heatmap(df_corr, annot=True, fmt='.2f', cmap='Blues', square=True)
plt.show()
```