Python初心者勉強会（第2回）

名前：はら　ひろこ

趣味：すいみん

Python歴：1年くらい（Python以外に普段使うのは、Excel、R、Matlab）

0. Python初心者勉強会

つまづくポイント

① 必要な基礎知識が多い、ような気がする。

> 演算 + は使うけれど、 % や // はあまり使わない。
>
> 全て覚える必要はない！

② テキストで学んでも、おもしろくない。

> 基礎を知らなくても、マネから入ればいい。
>
> （例）python グラフ、、、で検索すれば、目当てに近いプログラムが出てくる。

③ 実務に即していない。

> CSVファイルを読み込む時、read(〇〇.csv) とか書いている
>
> けど、ファイルの場所を意識してないでしょ！と言いたい。

0. Python初心者勉強会の予定

3回でやりたいこと

Python の基礎

- Jupyter Notebookの使い方
- リストと辞書型/条件分岐とループ
- ライブラリの読み込み（Numpy/Scipy/Pandas/Matplotlib）

科学計算とデータ加工処理

- NumpyとScipy
- Pandas

  欠損データと異常値、~~時系列データの取り扱い~~

データの可視化

- グラフ
- シミュレーション

機械学習

Scikit-Learn/TensorFlow/…

画像処理

OpenCV/Scikit-Image/…

Webスクレイピング

Requests/Beautiful Soup/…

WEB開発

Django/Flask/Web2Py/…

音声録音・再生

PyAudio/scipy.signal/…

ゲーム開発

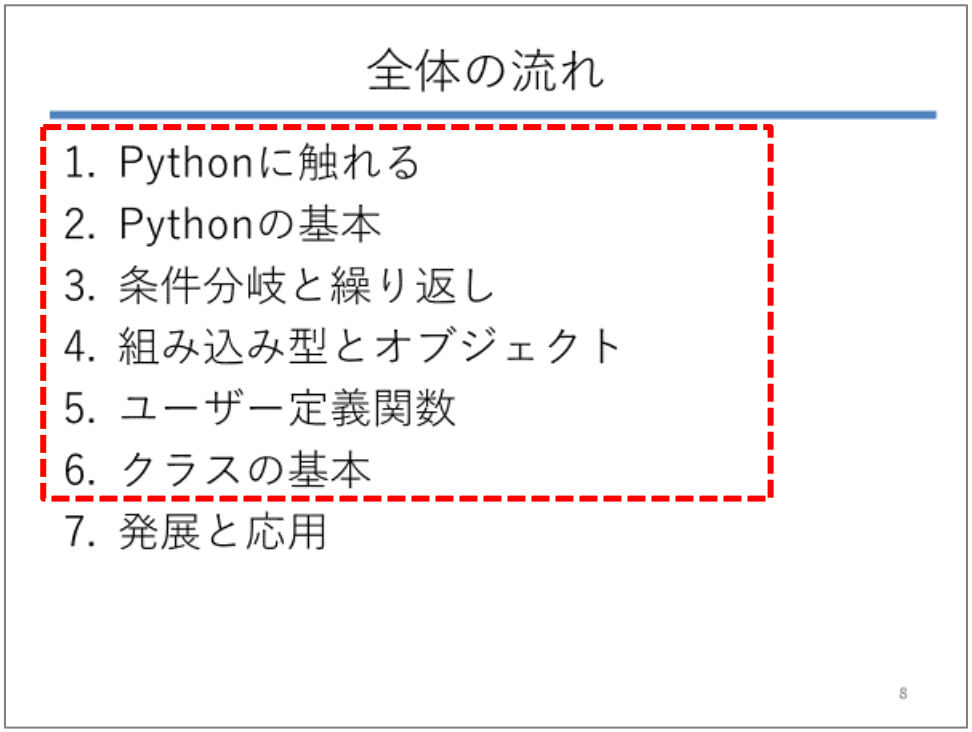
PyGame/Arcade/PyGlet/…

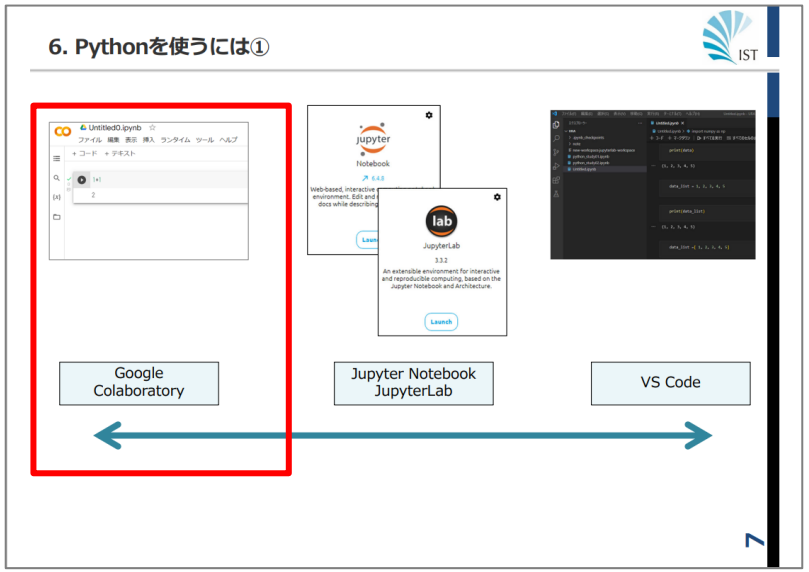
0. 参考テキスト

https://mitani.cs.tsukuba.ac.jp/book_support/python/

WEB上にある無料のテキストを使ったり、Connpasのような勉強会に参加する方法もあります！

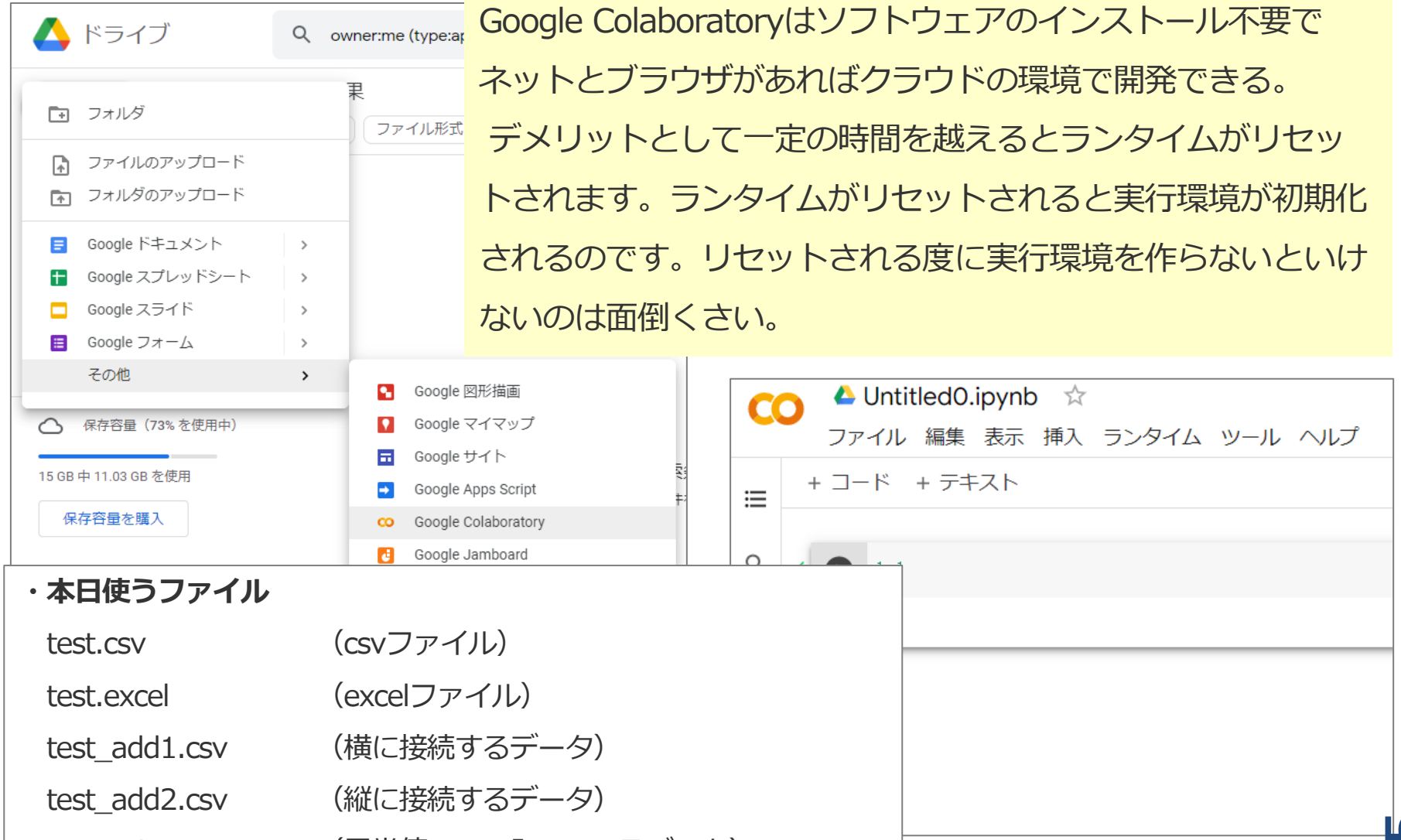
私はPyQ（月額3,040円）のPython独学プラットフォームで3か月間くらい写経しました。





0. Pythonを使うには①

Google Colaboratoryを使う方法



Google Colaboratoryはソフトウェアのインストール不要でネットとブラウザがあればクラウドの環境で開発できる。

デメリットとして一定の時間を越えるとランタイムがリセットされます。ランタイムがリセットされると実行環境が初期化されるのです。リセットされる度に実行環境を作らないといけないのは面倒くさい。

・本日使うファイル

test.csv	（csvファイル）
test.excel	（excelファイル）
test_add1.csv	（横に接続するデータ）
test_add2.csv	（縦に接続するデータ）
test_miss.csv	（異常値Nanの入っているデータ）

1. Numpy、Scipy、Pandasとは

・Numpy（ナムパイ）

基本的な配列処理や数値計算をするライブラリ。

高度で複雑な計算ができるほか、Pythonの通常の計算に比べて速度が速い。

・Scipy（サイパイ）

Numpyをさらに機能強化するライブラリ。

統計や信号計算ができる。

・Pandas（パンダス）

データフレーム形式で様々なデータを加工するためのライブラリ

・MatplotLib（マットプロットリブ）

データをグラフ化するためのライブラリ

・Seaborn（シーボーン）

Matplotlibの機能をより美しく、より簡単に実現するための可視化ライブラリ

1（参考）. Numpy、Scipy、Pandasとは

1990-1991年　1995年　2001年　2008年


Guido van Rossum
1956年生まれ


Travis Oliphant
1971年生まれ

+


Pearu Peterson
1971年生まれ


Eric Jones


Wes McKinney
1985年生まれ

・AQR Capital Managementにて、財務データを定量分析するための高性能で柔軟なツールを開発
・AQRを去る前に上司を説得して、ライブラリー（pandas）の一般公開が可能となった
・AQRを退職後、デューク大で統計学の博士号を取得

2. 配列

2-1. 配列の種類

List	Array	Series	Dataframe
Python 標準の配列	外部ライブラリの配列		
	numpy で定義された配列	Pandas で定義された一次元配列	Pandas で定義された二次元配列
['A大学', 'B大学', 'C大学'] [11, 12, 13] [123, 'abc', 456.789]	[[11 12 13] [21 22 23] [31 32 33]]	0 10 1 11 2 12	CLM0 CLM1 CLM2 0 100 101 102 1 200 201 202
—	import numpy as np	import pandas as pd	
list1 = ["A大学", "B大学", "C大学"]	ary2 = np.array([[11, 12, 13], [21, 22, 23], [31, 32, 33]])	se1 = pd.Series([10, 11, 12],name='se1')	df1 = pd.DataFrame([[100, 101, 102],[200, 201, 202]], columns=['CLM0', 'CLM1', 'CLM2'])
柔軟性：高い 処理速度：遅い	柔軟性：低い 処理速度：速い		

柔軟性とは、サイズの変更、タイプの混在をさします。

2-2. 配列（list）

```
1 list1 = ["A大学", "B大学", "C大学"]
2 print(list1)
```

```
['A大学', 'B大学', 'C大学']
```

```
1 list2 = [11, 12, 13]
2 print(list2)
```

```
[11, 12, 13]
```

外部モジュールを必要としない

様々な型が混在してもOK

```
1 list3 =  [123, 'abc', 456.789]
2 print(list3)
```

```
[123, 'abc', 456.789]
```

```
1 dict1 = {'a':'ABC', 'b':'BCD', 'c': 'CDE'}
2 print(dict1)
```

```
{'a': 'ABC', 'b': 'BCD', 'c': 'CDE'}
```

```
1 tuple1 = ('AB', 12)
2 print(tuple1)
```

```
('AB', 12)
```

型

Page.44

- データの種類のことを「型」とよぶ

はじめから準備されている型「組み込み型」

型	型名（日本語表記）	値の例
int	整数型	-1, 0, 1, 2, 10, 100
float	浮動小数点数型	小数点を含む数 -0.12, 0.0, 0.5, 2.34
str	文字列型	'hello', 'こんにちは'
bool	真偽値型	True, False

2-3. 配列（array）

```
1 ary1 = np.array([["A大学", "B大学", "C大学"],[21, 22, 23]])
2 print(ary1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:¥Users¥Public¥Documents¥Wondershare¥CreatorTemp/ipykernel_22336/4134498023.py in <module>
----> 1 ary1 = np.array([["A大学", "B大学", "C大学"],[21, 22, 23]])
      2 print(ary1)

NameError: name 'np' is not defined
```

外部モジュールを必要とする

```
1 import numpy as np
2 ary1 = np.array([["A大学", "B大学", "C大学"],[21, 22, 23]])
3 print(ary1)
```

```
[['A大学' 'B大学' 'C大学']
 ['21' '22' '23']]
```

数字もstr（文字列）型になる

```
1 import numpy as np
2 ary2 = np.array([[11, 12, 13], [21, 22, 23], [31, 32, 33]])
3 print(ary2)
```

```
[[11 12 13]
 [21 22 23]
 [31 32 33]]
```

2-4. 配列（series）

```
1 sel = pd.Series([10, 11, 12],name='sel')
2 print(sel)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:¥Users¥Public¥Documents¥Wondershare¥CreatorTemp/ipykernel_22336/3840414658.py in <module>
----> 1 sel = pd.Series([10, 11, 12],name='sel')
      2 print(sel)

NameError: name 'pd' is not defined
```

外部モジュールを必要とする

```
1 import pandas as pd
2 sel = pd.Series([10, 11, 12],name='sel')
3 print(sel)
```

```
0    10
1    11
2    12
Name: sel, dtype: int64
```

DataFrame　　Series

	Unnamed: 0	A大学	B大学	C大学	D大学	E大学	F大学	G大学
0	2018年	1000	2000	3000	4000	5000	6000	7000
1	2019年	1001	2001	3001	4001	5001	6001	7001
2	2020年	1002	2002	3002	4002	5002	6002	7002
3	2021年	1003	2003	3003	4003	5003	6003	7003
4	2022年	1004	2004	3004	4004	5004	6004	7004

2-5. 配列（dataframe）

```
df1 = pd.DataFrame([[100, 101, 102],[200, 201, 202]], columns=['CLM0', 'CLM1', 'CLM2'])
print(df1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:¥Users¥Public¥Documents¥Wondershare¥CreatorTemp/ipykernel_24344/368419745.py in <module>
----> 1 df1 = pd.DataFrame([[100, 101, 102],[200, 201, 202]], columns=['CLM0', 'CLM1', 'CLM2'])
      2 print(df1)

NameError: name 'pd' is not defined
```

```
import pandas as pd
df1 = pd.DataFrame([[100, 101, 102],[200, 201, 202]], columns=['CLM0', 'CLM1', 'CLM2'])
print(df1)
```

```
   CLM0  CLM1  CLM2
0   100   101   102
1   200   201   202
```

```
df1
```

	CLM0	CLM1	CLM2
0	100	101	102
1	200	201	202

- CSVファイルを読み込む場合

```
df_x = pd.read_csv(test.csv)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
C:¥Users¥Public¥Documents¥Wondershare¥CreatorTemp/ipykernel_24344/173040320.p
----> 1 df_x = pd.read_csv(test.csv)

NameError: name 'test' is not defined
```

```
df_x = pd.read_csv("test.csv")
df_x
```

	Unnamed: 0	A大学	B大学	C大学	D大学	E大学	F大学	G大学
0	2018年	1000	2000	3000	4000	5000	6000	7000
1	2019年	1001	2001	3001	4001	5001	6001	7001
2	2020年	1002	2002	3002	4002	5002	6002	7002
3	2021年	1003	2003	3003	4003	5003	6003	7003
4	2022年	1004	2004	3004	4004	5004	6004	7004

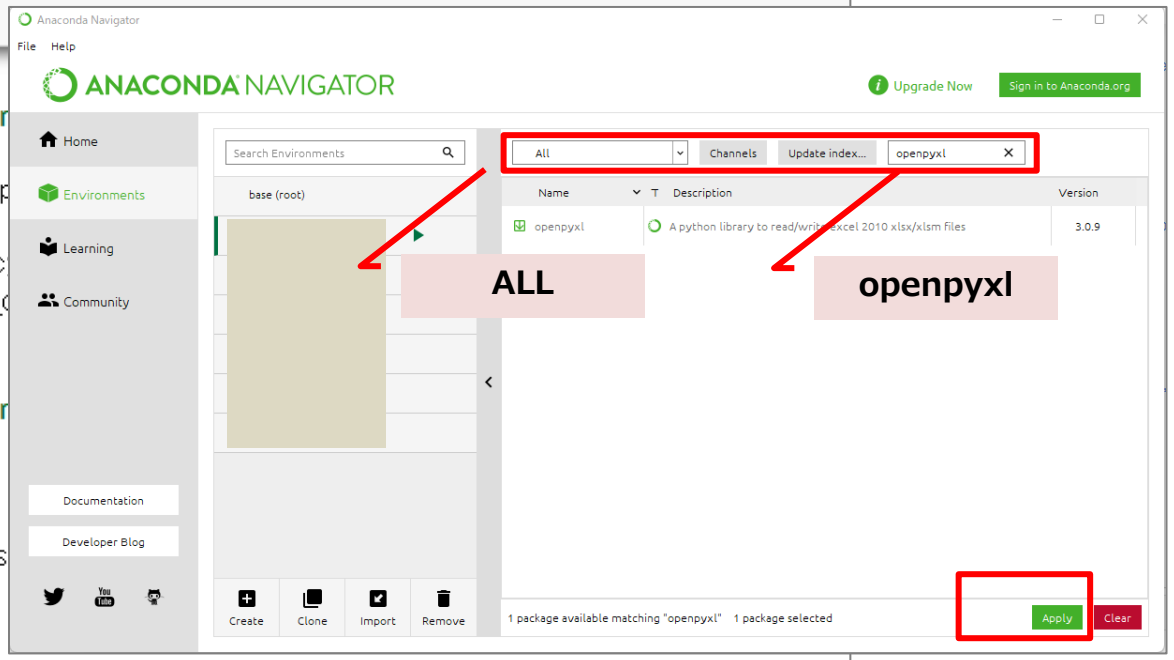
2-6. 配列（dataframe：エクセルファイルの読み込み）

```
1 df_ex1 = pd.read_excel("test_excel.xlsx")
2 df_ex1
```

```
C:¥ProgramData¥Anaconda3¥envs¥ALIVAN_pr
_buffer, storage_options)
    519             passed to fsspec for ap
    520         """
--> 521         import_optional_dependenc
    522         super().__init__(filepath_
    523

C:¥ProgramData¥Anaconda3¥envs¥ALIVAN_pr
e, extra, errors, min_version)
    116     except ImportError:
    117         if errors == "raise":
--> 118             raise ImportError(ms
    119         else:
    120             return None

ImportError: Missing optional dependency 'openpyxl'.  Use pip or conda to install openpyxl.
```




```
1 df_ex2 = pd.read_excel("test_excel.xlsx", sheet_name = "test_sheet2")
2 df_ex2
```

	Unnamed: 0	sheet2_A大学	B大学	C大学	D大学	E大学	F大学	G大学
0	2018年	1000	2000	3000	4000	5000	6000	7000
1	2019年	1001	2001	3001	4001	5001	6001	7001
2	2020年	1002	2002	3002	4002	5002	6002	7002
3	2021年	1003	2003	3003	4003	5003	6003	7003
4	2022年	1004	2004	3004	4004	5004	6004	7004

2-7. 配列（dataframe：colaboratory でのファイル読み込み）

```
1 import pandas as pd
2 df_x = pd.read_csv("test.csv")
3 df_x
```

```
FileNotFoundError                         Traceback (most recent call la
<ipython-input-4-59ec9e75a0c6> in <module>()
      1 import pandas as pd
----> 2 df_x = pd.read_csv("test.csv")
      3 df_x

                                  7 frames
/usr/local/lib/python3.7/dist-packages/pandas/io/common.py in get_handle
storage_options)
    705                 encoding=ioargs.encoding,
    706                 errors=errors,
--> 707                 newline="",
    708             )
    709         else:

FileNotFoundError: [Errno 2] No such file or directory: 'test.csv'
```

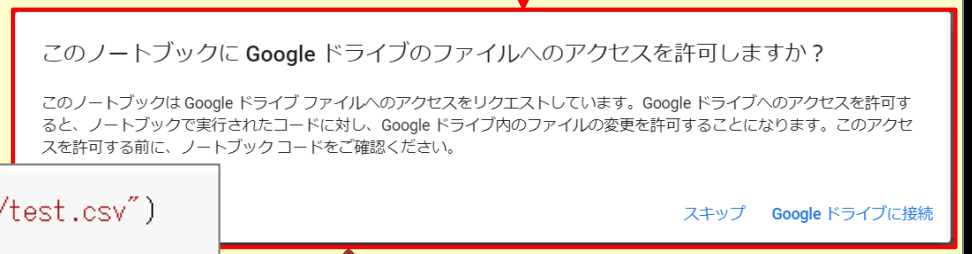
google colaboratoryの場合は同じフォルダ内にファイルを置いても、そのデータは見つからないので、パスを設定します。

```
[5]  1 from google.colab import drive

[6]  1 drive.mount('/content/drive')

     Mounted at /content/drive
```

初めてマウントする時は、アクセス許可の確認がある




```
1 df_x = pd.read_csv("drive/My Drive/Colaboratory Notebook/test.csv")
2 df_x
```

	Unnamed: 0	A大学	B大学	C大学	D大学	E大学	F大学	G大学
0	2018年	1000	2000	3000	4000	5000	6000	7000
1	2019年	1001	2001	3001	4001	5001	6001	7001
2	2020年	1002	2002	3002	4002	5002	6002	7002
3	2021年	1003	2003	3003	4003	5003	6003	7003
4	2022年	1004	2004	3004	4004	5004	6004	7004

2-8. 配列（dataframe：ファイル読み込み）

- 様々なフォルダにあるファイルを読み込む場合のパスの設定方法は、
  第1回の資料を参照ください。

11. CSVファイルの表示

デスクトップ上の[data_current]フォルダのtest_python_studyファイル（.csv）を開く

```
conda install pandas
```

```
Collecting package metadata (current_repodata.json)
Note: you may need to restart the kernel to use upd

Solving environment: ...working... done
```

```
import pandas as pd
```

```
df = pd.read_csv("C:\\Users\\hirop\\Desktop\\data_current\\test_python_study.csv")
```

```
df
```

	TimeStamp	Delta_TP9	Delta_AF7	Delta_AF8	Delta_TP10	Theta_TP9	Theta_AF7	Theta_AF8	Th
0	2022-01-17 18:13:37.921	1.759517	0.0	-0.151326	0.377344	1.079512	0.0	-0.230209	
1	2022-01-17	1.759517	0.0	-0.151326	0.377344	1.079512	0.0	-0.230209	

```
folder = "C:\\Users\\hirop\\Desktop\\data_current\\"
in_file01 = folder + "test.csv"
```

```
print(in_file01)
```

```
C:\Users\hirop\Desktop\data_current\test.csv
```

20

3. 配列の変換

3-1. 配列の相互変換

		To（変更後）			
		list	Array	Series	DataFrame
From（変更前）	llist (ist1)	—	np.array(list1)	pd.Series(list1)	pd.DataFrame(list1)
	Array (ary1)	ary1.tolist()	—	pd.Series(ary2) ※arrayが1次元の時のみ	pd.DataFrame(ary1)
	Series (se1)	se1.values.tolist()	se1.values	—	pd.DataFrame(se1)
	DataFrame (df1)	df1.values.tolist()	df1.to_numpy()	df1.iloc[:,0] ※0列目を指定	—

3-2. List から変換する場合

```
list1 = [11, 12, 13]
```

```
# arrayに変換
list_array = np.array(list1)
list_array
```

```
array([11, 12, 13])
```

```
# seriesに変換
list_series = pd.Series(list1)
list_series
```

```
0    11
1    12
2    13
dtype: int64
```

```
# dataframeに変換
list_dataframe = pd.DataFrame(list1)
list_dataframe
```

	0
0	11
1	12
2	13

3-3. Array から変換する場合

```
1 ary1 = np.array([[11, 12, 13], [21, 22, 23], [31, 32, 33]])
```

```
1 # listに変換
2 array_list = ary1.tolist()
3 array_list
```

```
[[11, 12, 13], [21, 22, 23], [31, 32, 33]]
```

```
1 # seriesに変換 --->次元エラー
2 array_series = pd.Series(ary1)
3 array_series
```

```
C:¥ProgramData¥Anaconda3¥envs¥ALIVAN_practice¥lib¥site-packages¥pandas¥core¥construction.py in sanitize_array(data, index,
dtype, copy, raise_cast_failure, allow_2d)
    574                 subarr = maybe_infer_to_datetimelike(subarr)
    575
--> 576     subarr = _sanitize_ndim(subarr, data, dtype, index, allow_2d=allow_2d)
    577
    578     if isinstance(subarr, np.ndarray):

C:¥ProgramData¥Anaconda3¥envs¥ALIVAN_practice¥lib¥site-packages¥pandas¥core¥construction.py in _sanitize_ndim(result, data,
dtype, index, allow_2d)
    625             if allow_2d:
    626                 return result
--> 627             raise ValueError("Data must be 1-dimensional")
    628         if is_object_dtype(dtype) and isinstance(dty
    629             # i.e. PandasDtype("O")

ValueError: Data must be 1-dimensional
```

次元エラー：seriesは1次元のみ対応

```
1 # seriesに変換 ※arrayが1次元の時のみ可能
2 ary2 = np.array([11, 12, 13])
3 array_series = pd.Series(ary2)
4 array_series
```

```
0    11
1    12
2    13
dtype: int32
```

3-4. Array から変換する場合

```
# dataframeに変換
array_dataframe = pd.DataFrame(ary1)
array_dataframe
```

	0	1	2
0	11	12	13
1	21	22	23
2	31	32	33

```
# dataframeに変換(カラム名の追加)
array_dataframe = pd.DataFrame(ary1, columns=['clm0', 'clm1', 'clm2'])
array_dataframe
```

	clm0	clm1	clm2
0	11	12	13
1	21	22	23
2	31	32	33

dataframeでは、カラム名やindex名を指定することができます。
オプションを指定する場合は、columns = [], index =[]を用います。

3-5. Series から変換する場合

```
1  se1 = pd.Series([10, 11, 12],name='se1')
```

```
1  # listに変換
2  series_list = se1.values.tolist()
3  series_list
```

```
[10, 11, 12]
```

```
1  # arrayに変換
2  series_array = se1.values
3  series_array
```

```
array([10, 11, 12], dtype=int64)
```

```
1  # dataframeに変換
2  series_dataframe = pd.DataFrame(se1)
3  series_dataframe
```

	se1
0	10
1	11
2	12

3-6. DataFrame から変換する場合

```
df1 = pd.DataFrame([[100, 101, 102],[200, 201, 202]], columns=['CLM0', 'CLM1', 'CLM2'])
```

```
# listに変換
dataframe_list = df1.values.tolist()
dataframe_list
```

```
[[100, 101, 102], [200, 201, 202]]
```

```
# arrayに変換
dataframe_array = df1.to_numpy()
dataframe_array
```

```
array([[100, 101, 102],
       [200, 201, 202]], dtype=int64)
```

```
# seriesに変換
dataframe_series = df1.iloc[:,0]  # 0列目を指定して抽出
dataframe_series
```

```
0    101
1    201
Name: CLM1, dtype: int64
```

4. DataFrame の操作

4-1. DataFrameの操作（行・列の抽出）

```
df_x = pd.read_csv("test.csv")
df_x
```

	Unnamed: 0	A大学	B大学	C大学	D大学	E大学	F大学	G大学
0	2018年	1000	2000	3000	4000	5000	6000	7000
1	2019年	1001	2001	3001	4001	5001	6001	7001
2	2020年	1002	2002	3002	4002	5002	6002	7002
3	2021年	1003	2003	3003	4003	5003	6003	7003
4	2022年	1004	2004	3004	4004	5004	6004	7004

インデックスを指定

```
df_a = pd.read_csv("test.csv", index_col=0)
df_a
```

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000	2000	3000	4000	5000	6000	7000
2019年	1001	2001	3001	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002
2021年	1003	2003	3003	4003	5003	6003	7003
2022年	1004	2004	3004	4004	5004	6004	7004

```
# 転置
df_a.T
```

	2018年	2019年	2020年	2021年	2022年
A大学	1000	1001	1002	1003	1004
B大学	2000	2001	2002	2003	2004
C大学	3000	3001	3002	3003	3004
D大学	4000	4001	4002	4003	4004
E大学	5000	5001	5002	5003	5004
F大学	6000	6001	6002	6003	6004
G大学	7000	7001	7002	7003	7004

★追加. DataFrameの操作（カラム名の変更）

df_a

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000	2000	3000	4000	5000	6000	7000
2019年	1001	2001	3001	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002
2021年	1003	2003	3003	4003	5003	6003	7003
2022年	1004	2004	3004	4004	5004	6004	7004

-- 追加（カラム名、インデックス名の変更）

```
1 df_new = df_a.rename(columns={'B大学': '列名の変更'}, index={'2020年': '行名の変更'})
2 df_new
```

	A大学	列名の変更	C大学	D大学	E大学	F大学	G大学
2018年	1000	2000	3000	4000	5000	6000	7000
2019年	1001	2001	3001	4001	5001	6001	7001
行名の変更	1002	2002	3002	4002	5002	6002	7002
2021年	1003	2003	3003	4003	5003	6003	7003
2022年	1004	2004	3004	4004	5004	6004	7004

4-2. DataFrameの操作（行・列の抽出）

df_a

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000	2000	3000	4000	5000	6000	7000
2019年	1001	2001	3001	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002
2021年	1003	2003	3003	4003	5003	6003	7003
2022年	1004	2004	3004	4004	5004	6004	7004

```
# 列の抽出①
df_a["A大学"]
```

```
2018年    1000
2019年    1001
2020年    1002
2021年    1003
2022年    1004
Name: A大学, dtype: int64
```

```
# 列の抽出②
df_a.A大学
```

```
2018年    1000
2019年    1001
2020年    1002
2021年    1003
2022年    1004
Name: A大学, dtype: int64
```

```
# 行の抽出①
df_a[1:3]
```

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2019年	1001	2001	3001	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002

```
# 行の抽出②
df_a[:3]
```

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000	2000	3000	4000	5000	6000	7000
2019年	1001	2001	3001	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002

```
# 行の抽出③
df_a['2019年':'2021年']
```

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2019年	1001	2001	3001	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002
2021年	1003	2003	3003	4003	5003	6003	7003

4-3. DataFrameの操作（行・列の抽出）

df_a

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000	2000	3000	4000	5000	6000	7000
2019年	1001	2001	3001	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002
2021年	1003	2003	3003	4003	5003	6003	7003
2022年	1004	2004	3004	4004	5004	6004	7004

```
1 df_a.iloc[[1,2,4],[0,2]]
```

	A大学	C大学
2019年	1001	3001
2020年	1002	3002
2022年	1004	3004

```
1 df_a.loc[:,['B大学','C大学']]
```

	B大学	C大学
2018年	2000	3000
2019年	2001	3001
2020年	2002	3002
2021年	2003	3003
2022年	2004	3004

```
1 df_a.loc['2019年':'2021年',['B大学','C大学']]
```

	B大学	C大学
2019年	2001	3001
2020年	2002	3002
2021年	2003	3003

- locは行名もしくは列名を指定することで特定の値を抽出できます。
- ilocはindexを指定することで特定の値を抽出できます。

4-4. DataFrameの操作（条件）

df_a

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000	2000	3000	4000	5000	6000	7000
2019年	1001	2001	3001	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002
2021年	1003	2003	3003	4003	5003	6003	7003
2022年	1004	2004	3004	4004	5004	6004	7004

```
1  # 条件を指定して行・列を取得します。
2  df_a[df_a > 3001]
```

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	NaN	NaN	NaN	4000	5000	6000	7000
2019年	NaN	NaN	NaN	4001	5001	6001	7001
2020年	NaN	NaN	3002.0	4002	5002	6002	7002
2021年	NaN	NaN	3003.0	4003	5003	6003	7003
2022年	NaN	NaN	3004.0	4004	5004	6004	7004

4-5. DataFrameの操作（行・列の削除）

df_a

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000	2000	3000	4000	5000	6000	7000
2019年	1001	2001	3001	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002
2021年	1003	2003	3003	4003	5003	6003	7003
2022年	1004	2004	3004	4004	5004	6004	7004

```
1  # 行の削除
2  df_a.drop('2021年', axis=0)
```

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000	2000	3000	4000	5000	6000	7000
2019年	1001	2001	3001	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002
2022年	1004	2004	3004	4004	5004	6004	7004

```
1  # 列の削除
2  df_a.drop('D大学', axis=1)
```

	A大学	B大学	C大学	E大学	F大学	G大学
2018年	1000	2000	3000	5000	6000	7000
2019年	1001	2001	3001	5001	6001	7001
2020年	1002	2002	3002	5002	6002	7002
2021年	1003	2003	3003	5003	6003	7003
2022年	1004	2004	3004	5004	6004	7004

4-6. DataFrameの操作（利用するcsvファイルの説明）

test_add2.csv → df_c

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2015年	997	1997	2997	3997	4997	5997	6997
2016年	998	1998	2998	3998	4998	5998	6998
2017年	999	1999	2999	3999	4999	5999	6999

test.csv → df_a

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000	2000	3000	4000	5000	6000	7000
2019年	1001	2001	3001	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002
2021年	1003	2003	3003	4003	5003	6003	7003
2022年	1004	2004	3004	4004	5004	6004	7004

test_add1.csv → df_b

	H大学	I大学	J大学
2018年	100	200	300
2019年	101	201	301
2020年	102	202	302
2021年	103	203	303
2022年	104	204	304

4-7. DataFrameの操作（結合：横）

test_add2.csv → df_c

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2015年	997	1997	2997	3997	4997	5997	6997
2016年	998	1998	2998	3998	4998	5998	6998
2017年	999	1999	2999	3999	4999	5999	6999

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000	2000	3000	4000	5000	6000	7000
2019年	1001	2001	3001	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002
2021年	1003	2003	3003	4003	5003	6003	7003
2022年	1004	2004	3004	4004	5004	6004	7004

test.csv → df_a

	H大学	I大学	J大学
2018年	100	200	300
2019年	101	201	301
2020年	102	202	302
2021年	103	203	303
2022年	104	204	304

test_add1.csv → df_b

```
1 # 横結合
2 # インデックス（行ラベル）をキーに指定する場合は、引数left_index, right_indexをTrueとする。
3 pd.merge(df_a, df_b, left_index=True, right_index=True)
```

	A大学	B大学	C大学	D大学	E大学	F大学	G大学	H大学	I大学	J大学
2018年	1000	2000	3000	4000	5000	6000	7000	100	200	300
2019年	1001	2001	3001	4001	5001	6001	7001	101	201	301
2020年	1002	2002	3002	4002	5002	6002	7002	102	202	302
2021年	1003	2003	3003	4003	5003	6003	7003	103	203	303
2022年	1004	2004	3004	4004	5004	6004	7004	104	204	304

4-8. DataFrameの操作（結合：縦）

test_add2.csv → df_c

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2015年	997	1997	2997	3997	4997	5997	6997
2016年	998	1998	2998	3998	4998	5998	6998
2017年	999	1999	2999	3999	4999	5999	6999

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000	2000	3000	4000	5000	6000	7000
2019年	1001	2001	3001	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002
2021年	1003	2003	3003	4003	5003	6003	7003
2022年	1004	2004	3004	4004	5004	6004	7004

test.csv → df_a

	H大学	I大学	J大学
2018年	100	200	300
2019年	101	201	301
2020年	102	202	302
2021年	103	203	303
2022年	104	204	304

test_

```
1 # 縦結合
2 pd.concat([df_a, df_c])
```

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000	2000	3000	4000	5000	6000	7000
2019年	1001	2001	3001	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002
2021年	1003	2003	3003	4003	5003	6003	7003
2022年	1004	2004	3004	4004	5004	6004	7004
2015年	997	1997	2997	3997	4997	5997	6997
2016年	998	1998	2998	3998	4998	5998	6998
2017年	999	1999	2999	3999	4999	5999	6999

5. 欠損値の補完

5-1. 欠損値の補完

Test_miss.csv → df_miss

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000	2000	3000	4000	5000	6000	7000
2019年		2001	NaN	4001	5001	6001	7001
2020年	1002	2002	3002	4002	5002	6002	7002
2021年	1003	2003	3003	4003	5003	6003	7003
2022年	1004	2004	3004	4004	5004	6004	7004

```
1  import numpy as np
2  from numpy import nan as NA
3  import pandas as pd
4
5  df_miss = pd.read_csv("test_miss.csv", index_col=0)
6  df_miss
```

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000.0	2000	3000.0	4000	5000	6000	7000
2019年	NaN	2001	NaN	4001	5001	6001	7001
2020年	1002.0	2002	3002.0	4002	5002	6002	7002
2021年	1003.0	2003	3003.0	4003	5003	6003	7003
2022年	1004.0	2004	3004.0	4004	5004	6004	7004

```
1  # 欠損値有無の確認
2  df_miss.isnull().sum()
```

```
A大学    1
B大学    0
C大学    1
D大学    0
E大学    0
F大学    0
G大学    0
dtype: int64
```

5-2. 欠損値の補完

df_miss

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000.0	2000	3000.0	4000	5000	6000	7000
2019年	NaN	2001	NaN	4001	5001	6001	7001
2020年	1002.0	2002	3002.0	4002	5002	6002	7002
2021年	1003.0	2003	3003.0	4003	5003	6003	7003
2022年	1004.0	2004	3004.0	4004	5004	6004	7004

```
# 前の値で埋める
df_miss.fillna(method = 'ffill')
# 後ろの値で埋める
# df_miss.fillna(method = 'bfill')
```

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000.0	2000	3000.0	4000	5000	6000	7000
2019年	1000.0	2001	3000.0	4001	5001	6001	7001
2020年	1002.0	2002	3002.0	4002	5002	6002	7002
2021年	1003.0	2003	3003.0	4003	5003	6003	7003
2022年	1004.0	2004	3004.0	4004	5004	6004	7004

```
# リストワイズ削除
# NaNのある行を全て削除
df_miss.dropna()
```

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000.0	2000	3000.0	4000	5000	6000	7000
2020年	1002.0	2002	3002.0	4002	5002	6002	7002
2021年	1003.0	2003	3003.0	4003	5003	6003	7003
2022年	1004.0	2004	3004.0	4004	5004	6004	7004

```
# fillna(値)で埋める
df_miss.fillna(0)
```

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000.0	2000	3000.0	4000	5000	6000	7000
2019年	0.0	2001	0.0	4001	5001	6001	7001
2020年	1002.0	2002	3002.0	4002	5002	6002	7002
2021年	1003.0	2003	3003.0	4003	5003	6003	7003
2022年	1004.0	2004	3004.0	4004	5004	6004	7004

5-3. 欠損値の補完

df_miss

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000.0	2000	3000.0	4000	5000	6000	7000
2019年	NaN	2001	NaN	4001	5001	6001	7001
2020年	1002.0	2002	3002.0	4002	5002	6002	7002
2021年	1003.0	2003	3003.0	4003	5003	6003	7003
2022年	1004.0	2004	3004.0	4004	5004	6004	7004

```
1 # 前後の値から予測する
2 df_miss.interpolate()
```

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000.0	2000	3000.0	4000	5000	6000	7000
2019年	1001.0	2001	3001.0	4001	5001	6001	7001
2020年	1002.0	2002	3002.0	4002	5002	6002	7002
2021年	1003.0	2003	3003.0	4003	5003	6003	7003
2022年	1004.0	2004	3004.0	4004	5004	6004	7004

```
1 # 平均で埋める
2 # 平均値
3 df_miss.mean()
```

```
A大学    1002.25
B大学    2002.00
C大学    3002.25
D大学    4002.00
E大学    5002.00
F大学    6002.00
G大学    7002.00
dtype: float64
```

```
1 # 平均値で埋める
2 df_miss.fillna(df_miss.mean())
```

	A大学	B大学	C大学	D大学	E大学	F大学	G大学
2018年	1000.00	2000	3000.00	4000	5000	6000	7000
2019年	1002.25	2001	3002.25	4001	5001	6001	7001
2020年	1002.00	2002	3002.00	4002	5002	6002	7002
2021年	1003.00	2003	3003.00	4003	5003	6003	7003
2022年	1004.00	2004	3004.00	4004	5004	6004	7004

おわりに。

目次

1. Pythonに触れる
2. Pythonの基本
3. 条件分岐と繰り返し
4. 組み込み型とオブジェクト
5. ユーザー定義関数
6. クラスの基本

モジュールのインストール