

医療とAI・ビッグデータ入門

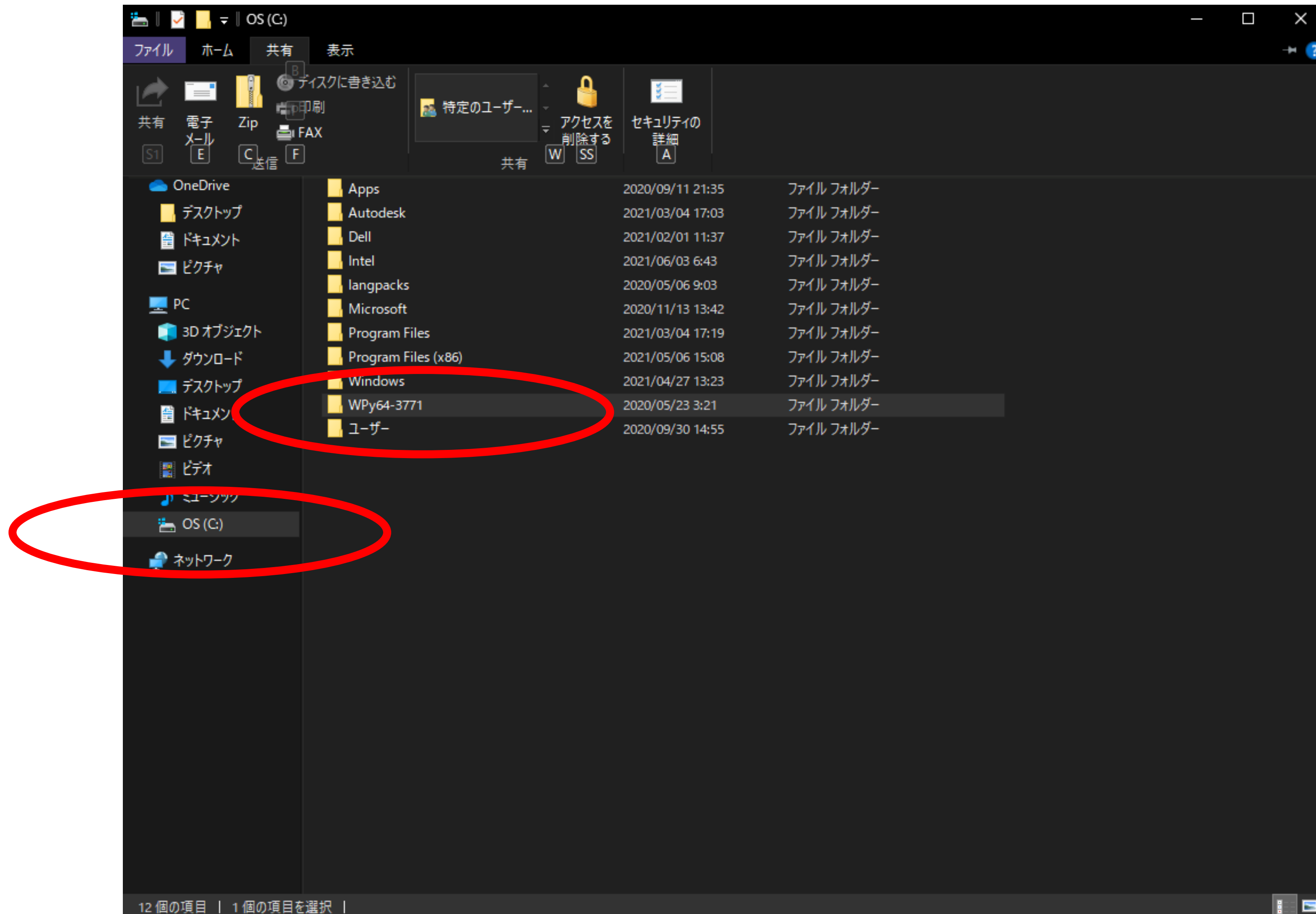
pythonの基礎
~データの扱い、numpy、pandasについて~

統合教育機構
須藤毅顕

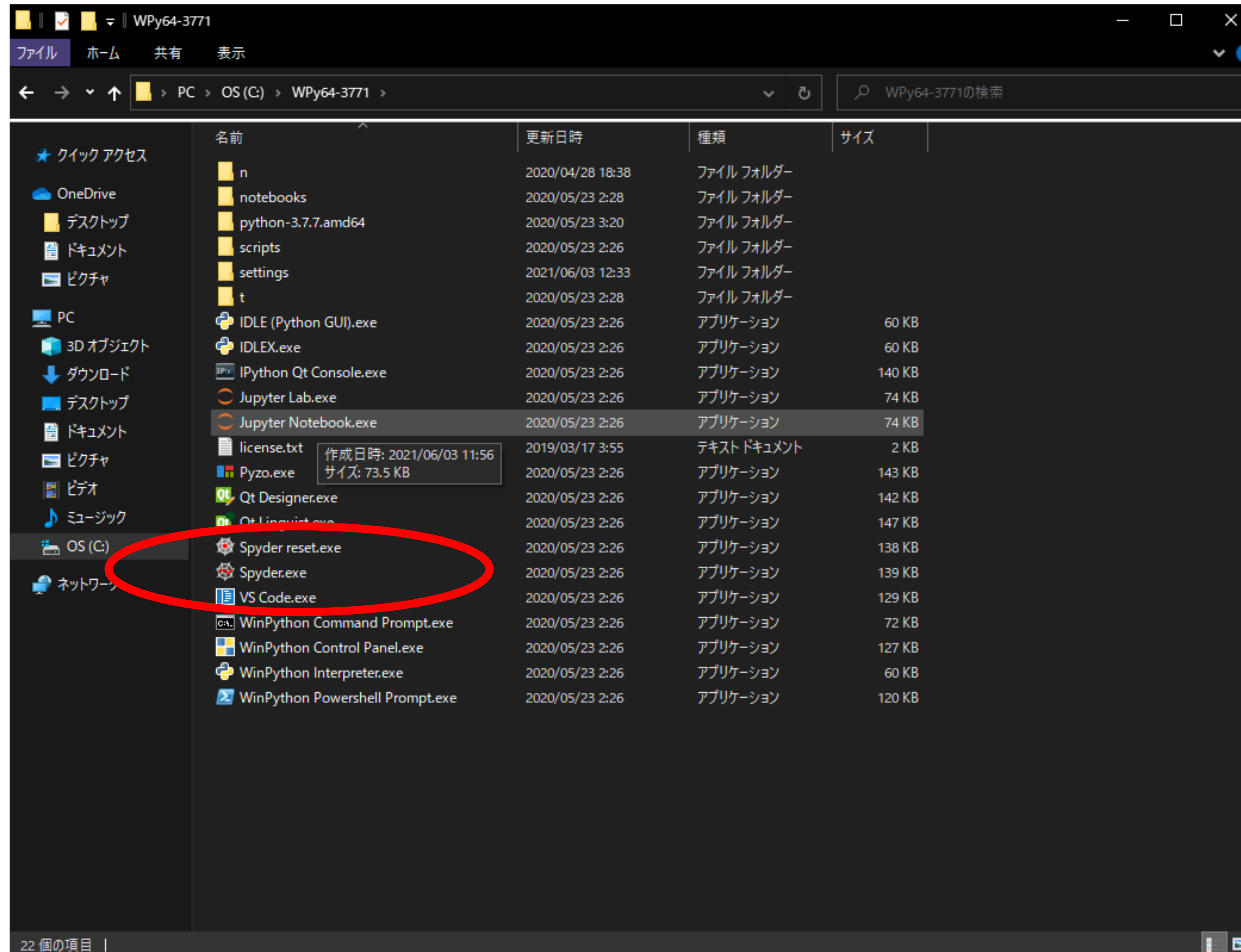
データの扱いとライブラリの使い方

初回では、次回以降の機械学習、深層学習で用いる
pythonの基本知識とデータ取り扱い方について解説します。

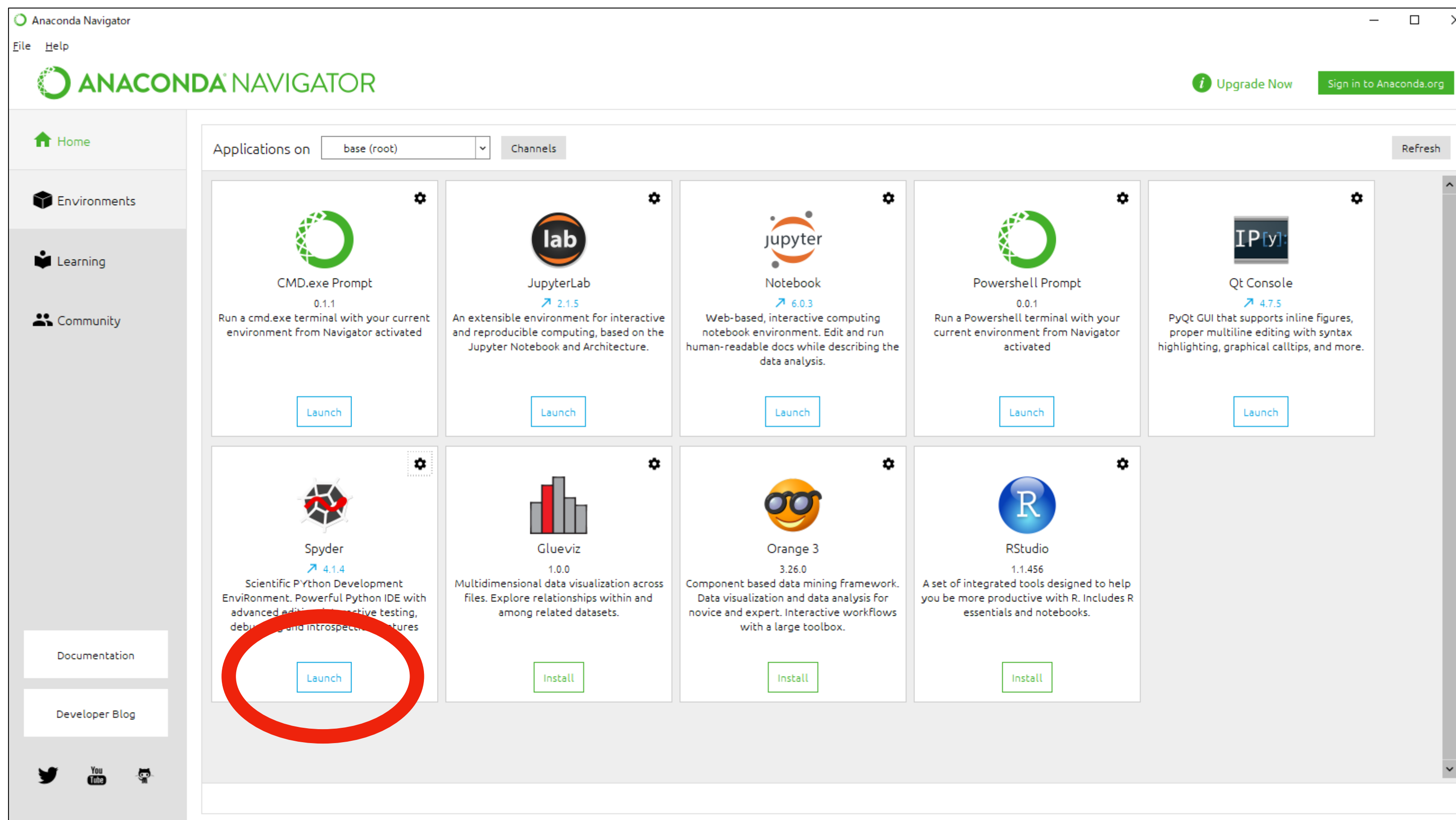
Winpythonを起動しよう (Windows)



Winpythonを起動しよう (Windows)



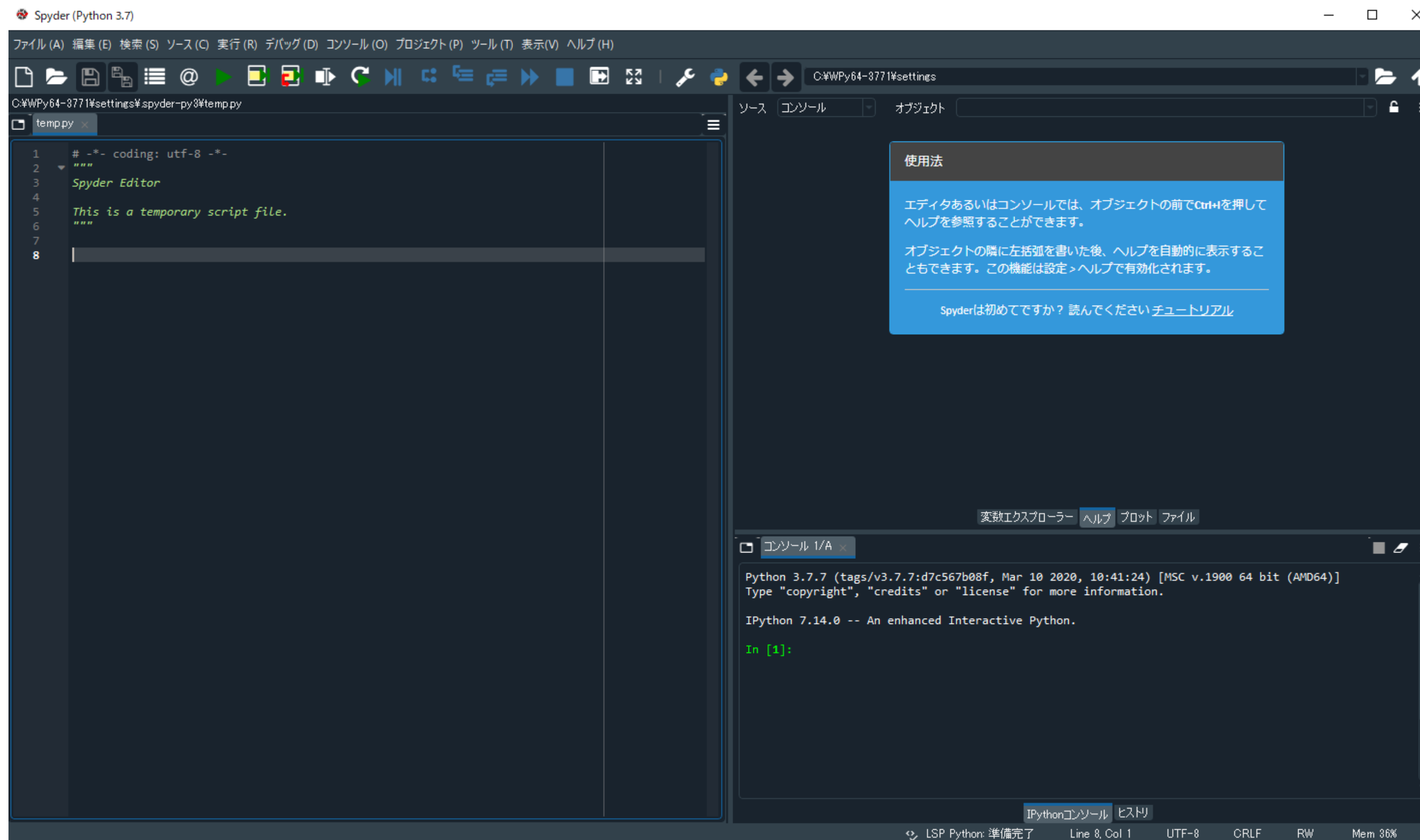
Anaconda NavigatorでSpyderを立ち上げる(Mac)



“Install”となっている人は、クリックしてしばらく待つと
“Launch”になるので、再度クリックします。

Pythonの実行方法

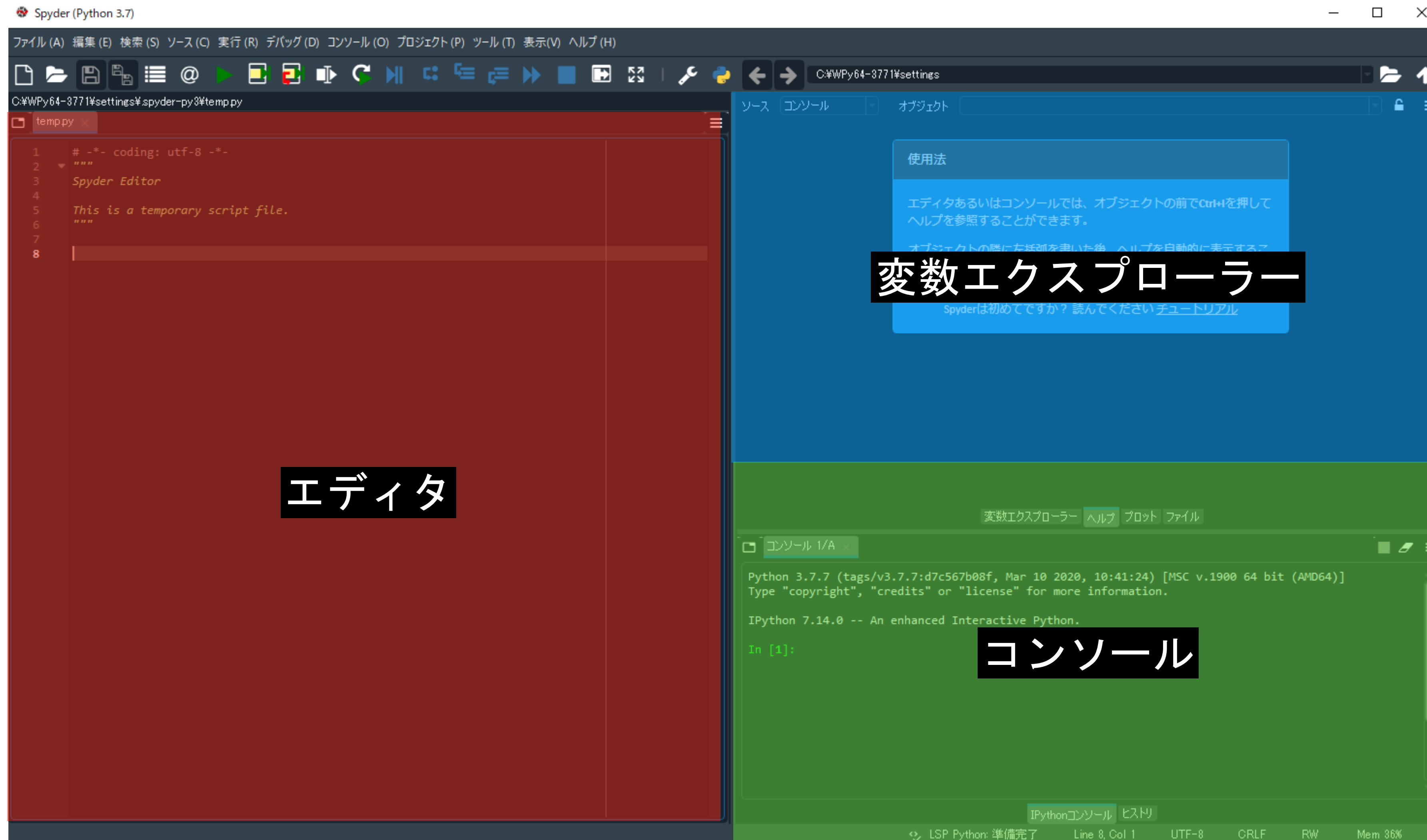
Spyderのホーム画面が表示されます。(spyderのバージョン4)



バージョンが違っていると表示されるアイコンが少し違う可能性があります。そのままでも大丈夫ですし、アップデートしても良いです。

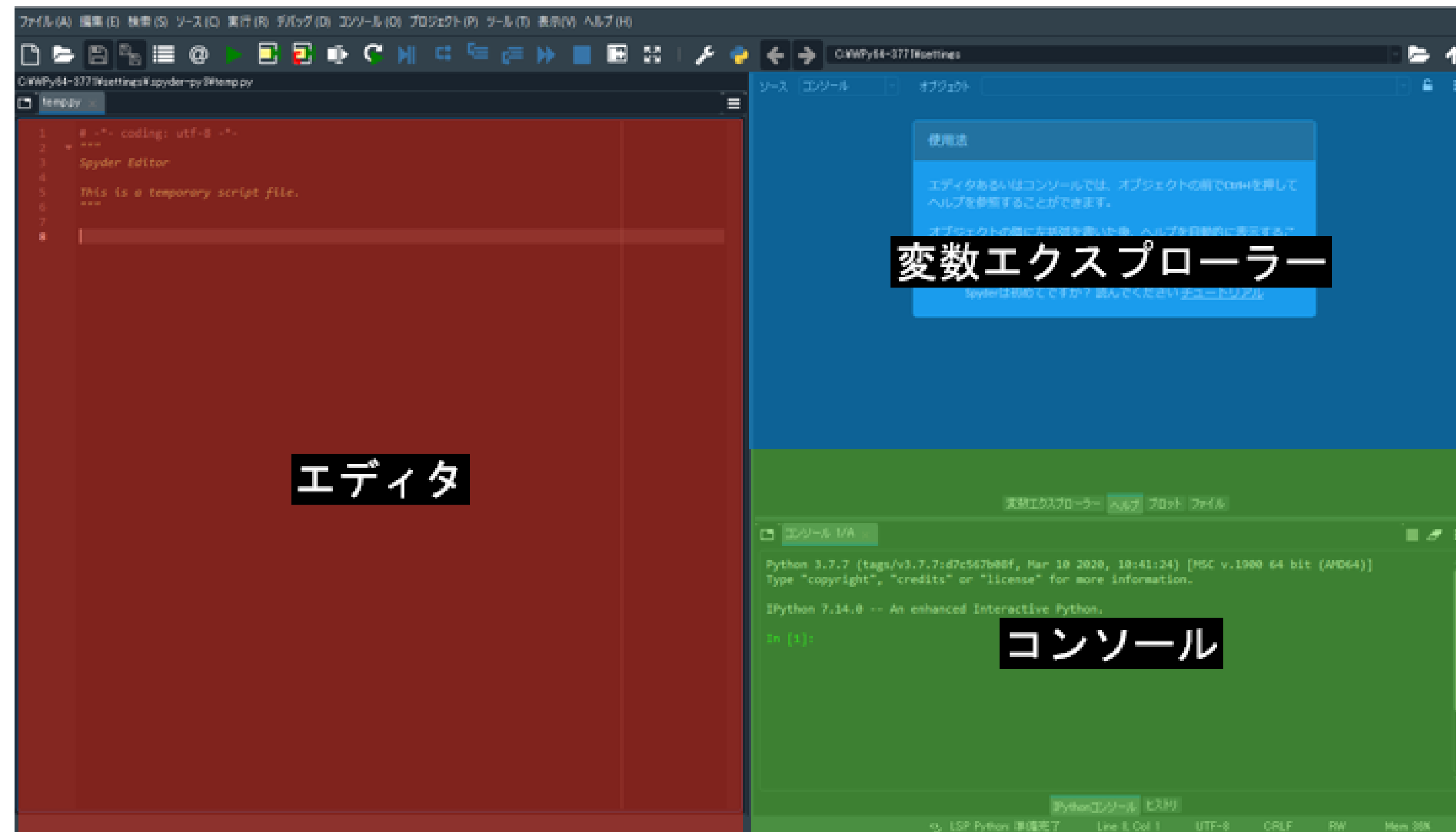
Spyderを起動する

大きく3つの画面に分かれております。左側のエディタ、右上の変数エクスプローラー、右下のコンソールです。



Spyderの実行方法

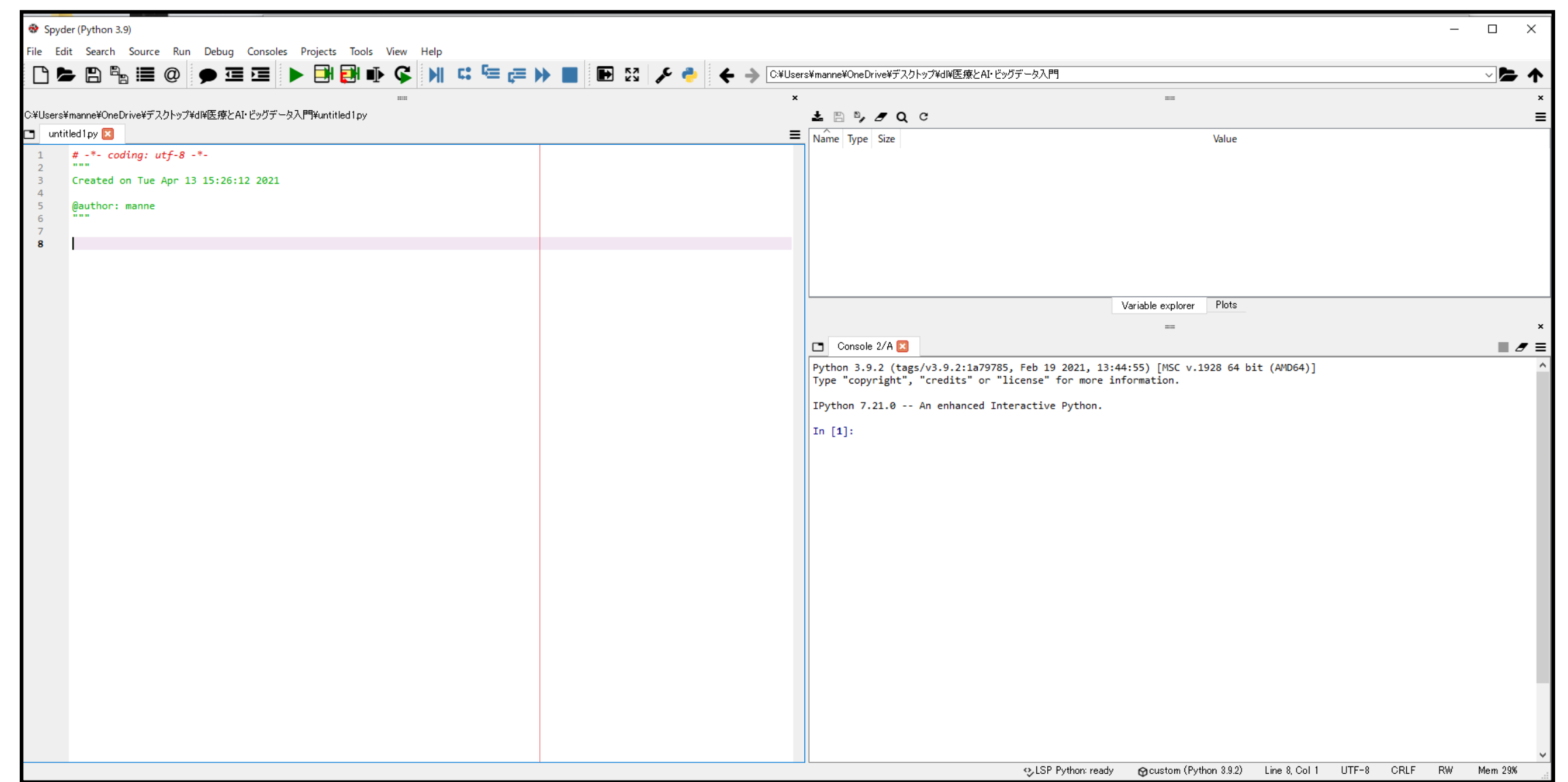
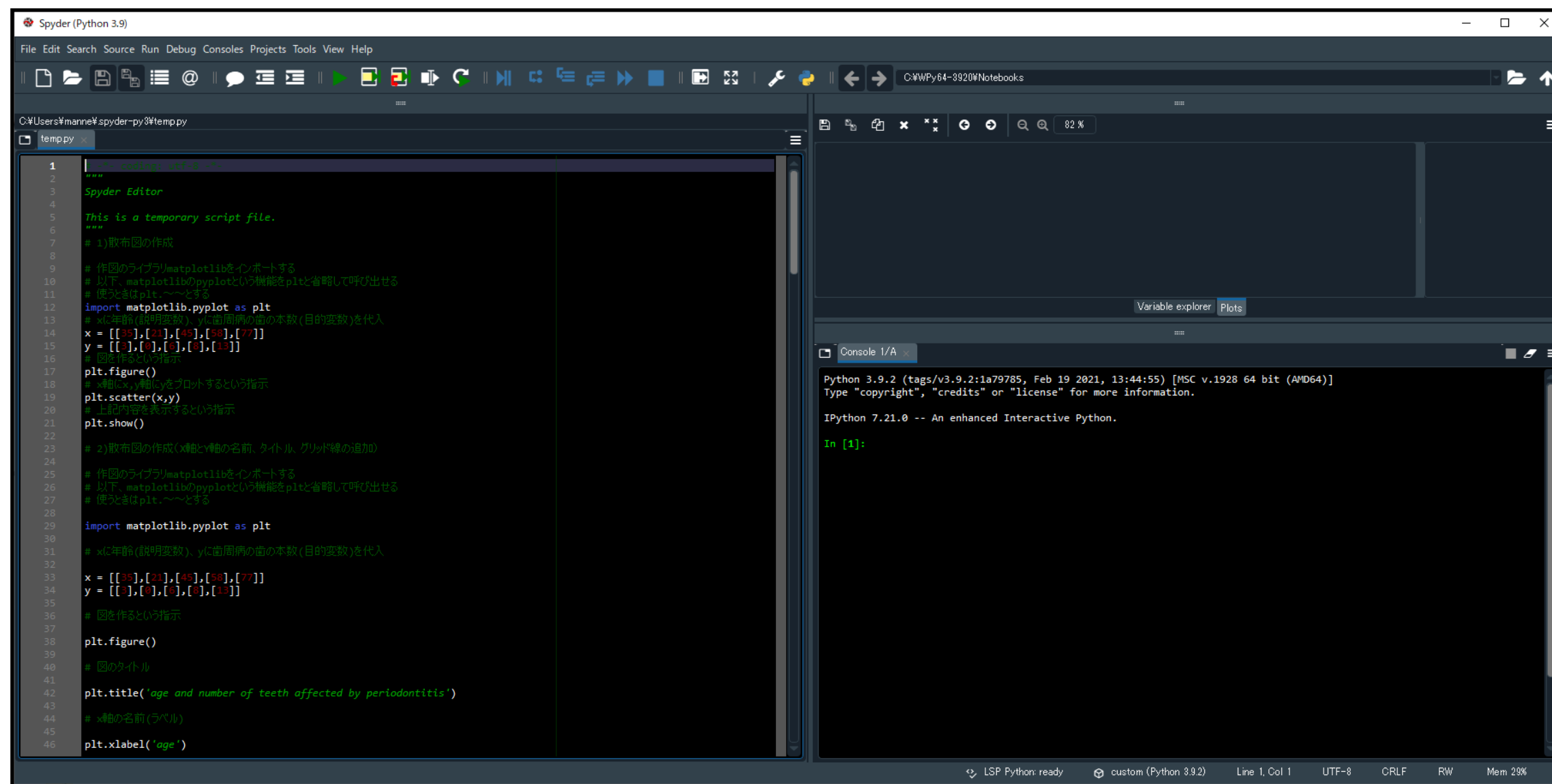
- ① エディタに文を書き、"実行"で(上から順に)全ての行を実行する
- ② エディタに文を書き、"1行ずつ実行"で1行ずつ実行する
- ③ コンソールに直接書き込み、enterで実行する
→結果は全てコンソールに表示される



③は対話的(チャットのように)すぐ結果が表示されるのですが、長い文を書くときや文を修正、保存するにはエディタの使用が適してしますので今回は①②で行います

(補足)

spyderのバージョンによっては背景が黒いかもしれませんが。
そのままでもいいですが、この実習スライドでは背景を白にしています。



メニューを日本語にするには、
[Tools]-[Preferences]-[General]-[Advanced settings]-[Language:]-[日本語]
背景を白にするには、[ツール]-[設定]-[外見]-[構文強調テーマ]を[Spyder]に変更
英語メニューの場合：[Tools]-[Preferences]-[Appearance]-[Syntax highlighting theme]を[Spyder]に変更

Spyderを実行する（確認事項）

新規ファイル作成 ファイルを開く 保存 実行 1行ずつ実行

C:\Users\manne\spyder-py3\temp.py

temp.py

現在開いている
ファイル名

```
1  # -*- coding: utf-8 -*-
2  """
3  Spyder Editor
4
5  This is a temporary script file.
6  """
7
8  
```

これから書き始める
カーソル

Spyderを実行する（確認事項）

新規ファイル作成 ファイルを開く 保存 実行 1行ずつ実行

The screenshot shows the Spyder IDE interface. The top toolbar contains icons for creating a new file, opening a file, saving, running, and running line by line. The file explorer shows the current file path: /Users/takeakisudo/Desktop/タイトル無し1.py. The editor window shows the file name: temp.py and タイトル無し1.py. The code editor displays the following content:

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Created on Tue May 25 06:50:31 2021
5
6  @author: takeakisudo
7  """
8
```

Annotations in the image include:

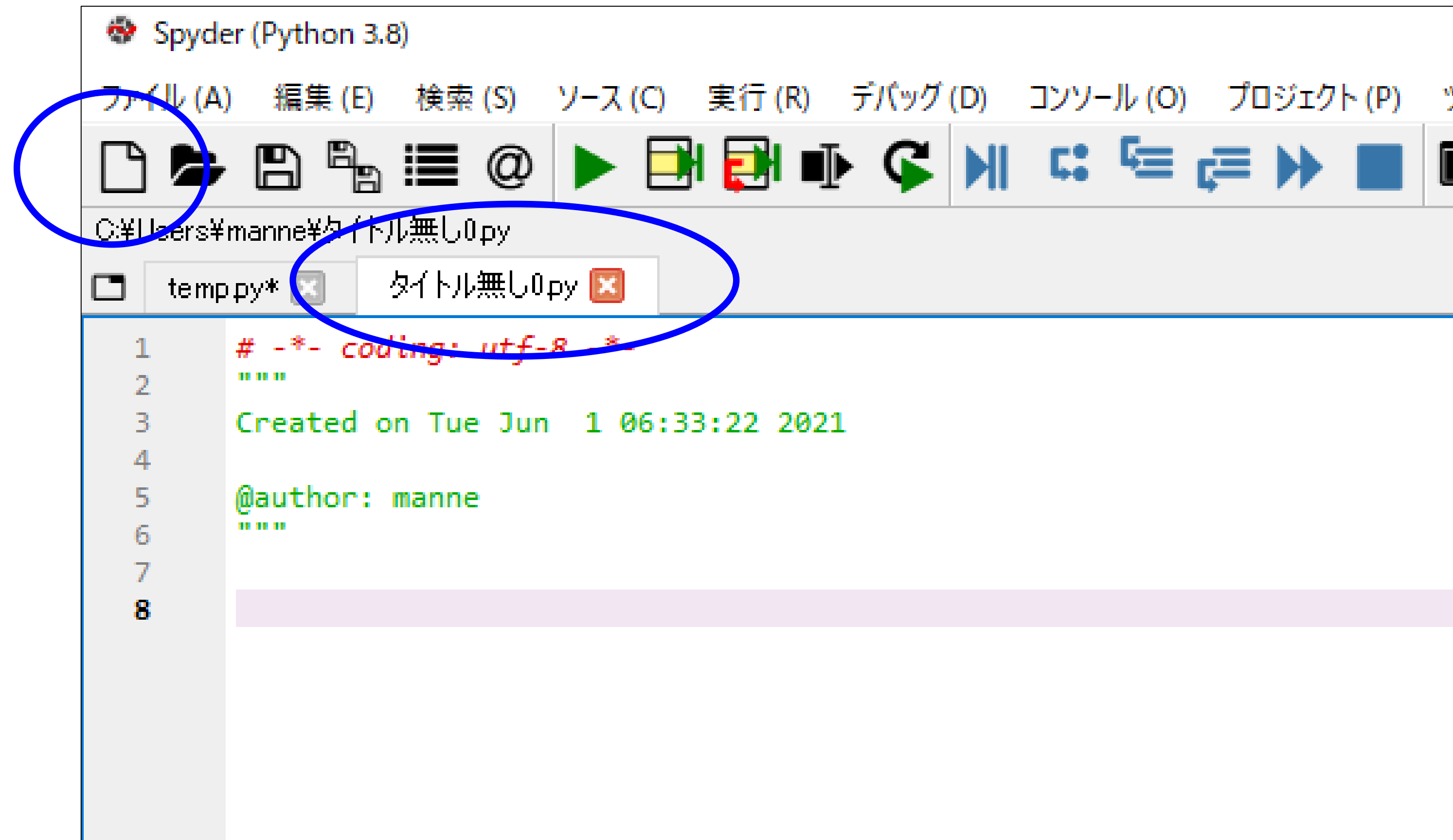
- Red circles around the 'New File', 'Open File', 'Save', 'Run', and 'Run Line by Line' icons in the toolbar.
- A red circle around the 'Run' icon in the toolbar.
- A red circle around the file name 'タイトル無し1.py' in the file explorer.
- A red circle around the line number '8' in the code editor.

Text labels with arrows pointing to the annotations:

- 新規ファイル作成 (New File)
- ファイルを開く (Open File)
- 保存 (Save)
- 実行 (Run)
- 1行ずつ実行 (Run Line by Line)
- 現在開いているファイル名 (Current File Name)
- これから書き始めるカーソル (Cursor to start writing from here)

Spyderを実行する（確認事項）

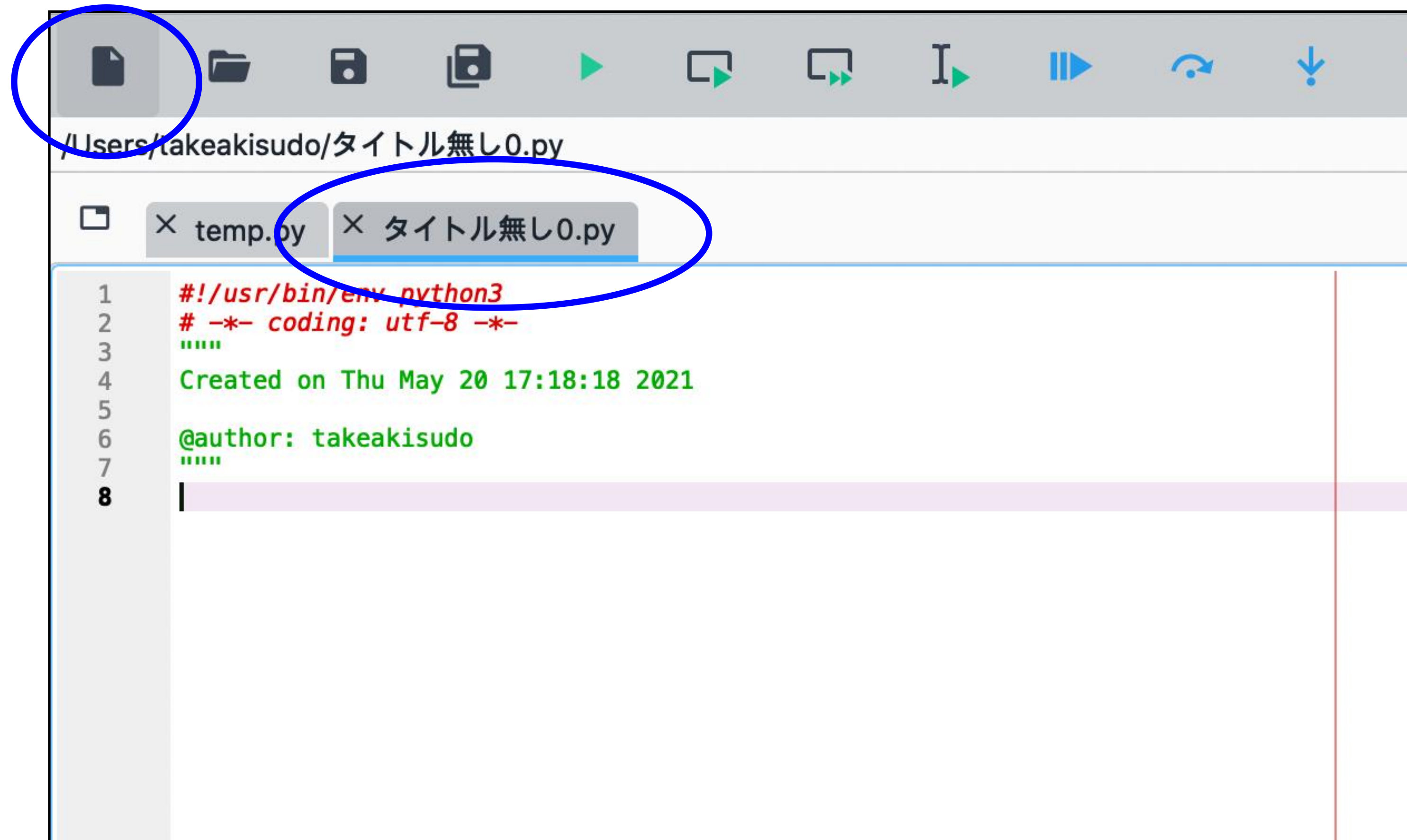
まず新しいファイルを作成してみましょう



左上の新規作成のアイコンをクリックするとエディタに“タイトル無し0.py”のファイルが作成されます。.pyの拡張子なのでpythonのファイルになります。

Spyderを実行する

まず新しいファイルを作成してみましょう



左上の新規作成のアイコンをクリックするとエディタに“タイトル無し0.py”のファイルが作成されます。.pyの拡張子なのでpythonのファイルになります。

データの操作の仕方について

医療とAI・ビッグデータ入門フォルダの1.txtを開いてみましょう

```
演習1のコピー.txt - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
# 1)変数と型
x = 35
data = 35
print(x)
print(data)
data = 45
print(data)
data = "こんにちは"
print(data)
x = 35
print(type(x))
x2 = 3.5
print(type(x2))
data = "こんにちは"
print(type(data))

# 2)リストの作成
x = [1,2,3,4,5]
print(x)
print(type(x))
print(x[0])
print(x[4])

##課題##
#1から10の要素を持つリストをtestという変数に代入しよう
#testを出力しよう
#testの2番目と5番目の要素を出力しよう
#len(test)、max(test)、min(test)を実行してみよう

# 3)numpy
import numpy as np

sample = np.array([1,2,3,4,5])
print(sample)
print(type(sample))
t = sample * 2
print(t)
```

1)変数と型

```
x = 35
data = 35
print(x)
print(data)
data = 45
print(data)
data = "こんにちは"
print(data)
x = 35
print(type(x))
x2 = 3.5
print(type(x2))
data = "こんにちは"
print(type(data))
```


データの操作の仕方について

左側のエディタに”#1) 変数”をコピー＆ペーストをしてみましょう

```
演習1のコピー.txt - ノート
ファイル(E) 編集(E) 書式(O) 表示(V) ヘルプ(H)
# 1)変数と型
x = 35
data = 35
print(x)
print(data)
data = 45
print(data)
data = "こんにちは"
print(data)
x = 35
print(type(x))
x2 = 3.5
print(type(x2))
data = "こんにちは"
print(type(data))

# 2)リストの作成
x = [1,2,3,4,5]
print(x)
print(type(x))
print(x[0])
print(x[4])

##課題##
#1から10の要素を持つリストをtestという変数に代入しよう
#testを出力しよう
#testの2番目と5番目の要素を出力しよう
#len(test)、max(test)、min(test)を実行してみよう

# 3)numpy
import numpy as np

sample = np.array([1,2,3,4,5])
print(sample)
print(type(sample))
t = sample * 2
print(t)
```

```
Spyder (Python 3.7)
エディタ - /Users/takeakisudo/Desktop/2021年度授業スライドまとめ/医療とAI・ビッグデータ入門/第3回/3-1.py
タイトル無し 2.py 3-1.py*
1 #!/usr/bin/env python3
2 #-*- coding: utf-8 -*-
3 """
4 Created on Tue Apr 13 22:51:50 2021
5
6 @author: takeakisudo
7 """
8
9 # 1) 変数とリストの練習
10
11 x = 35
12 data = 35
13 print(x)
14 print(data)
15 data = 45
16 print(data)
17 data = "こんにちは"
18 print(data)
19 x = 35
20 print(type(x))
21 x2 = 3.5
22 print(type(x2))
23 data = "こんにちは"
24 print(type(data))
25 x = [1,2,3,4,5]
26 print(x)
27 print(type(x))
28 print(x[0])
29 print(x[4])

変数エクスプローラー
型 サイズ 値

変数エクスプローラー ファイルエクスプローラー

IPythonコンソール
Python 3.7.9 (default, Aug 31 2020, 07:22:35)
Type "copyright", "credits" or "license" for more information.

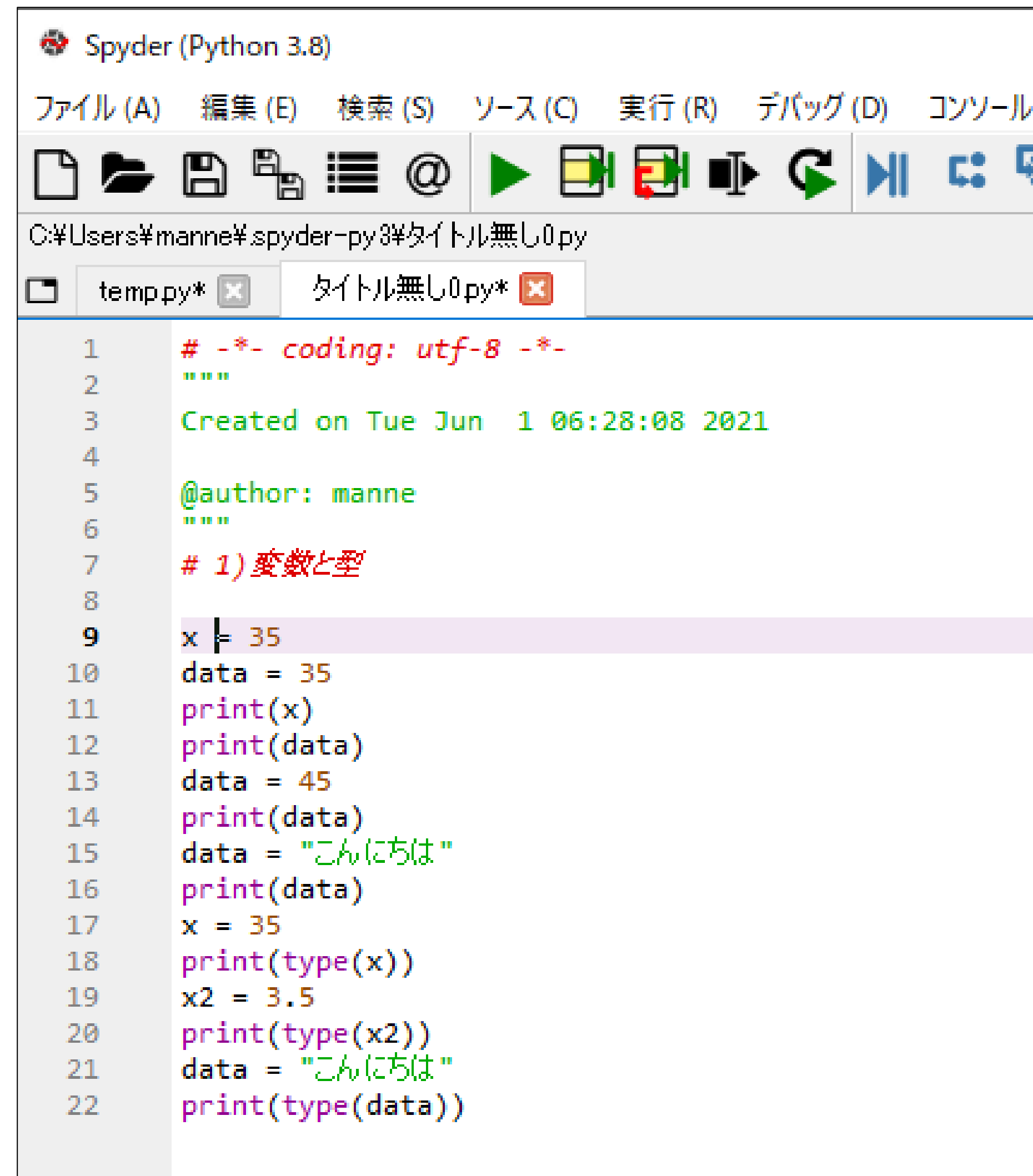
IPython 7.19.0 -- An enhanced Interactive Python.

In [1]:

IPythonコンソール ヒストリログ
権限: RW 改行: LF エンコード: UTF-8 行: 11 列: 7 メモリ: 56 %
```


データの操作の仕方について

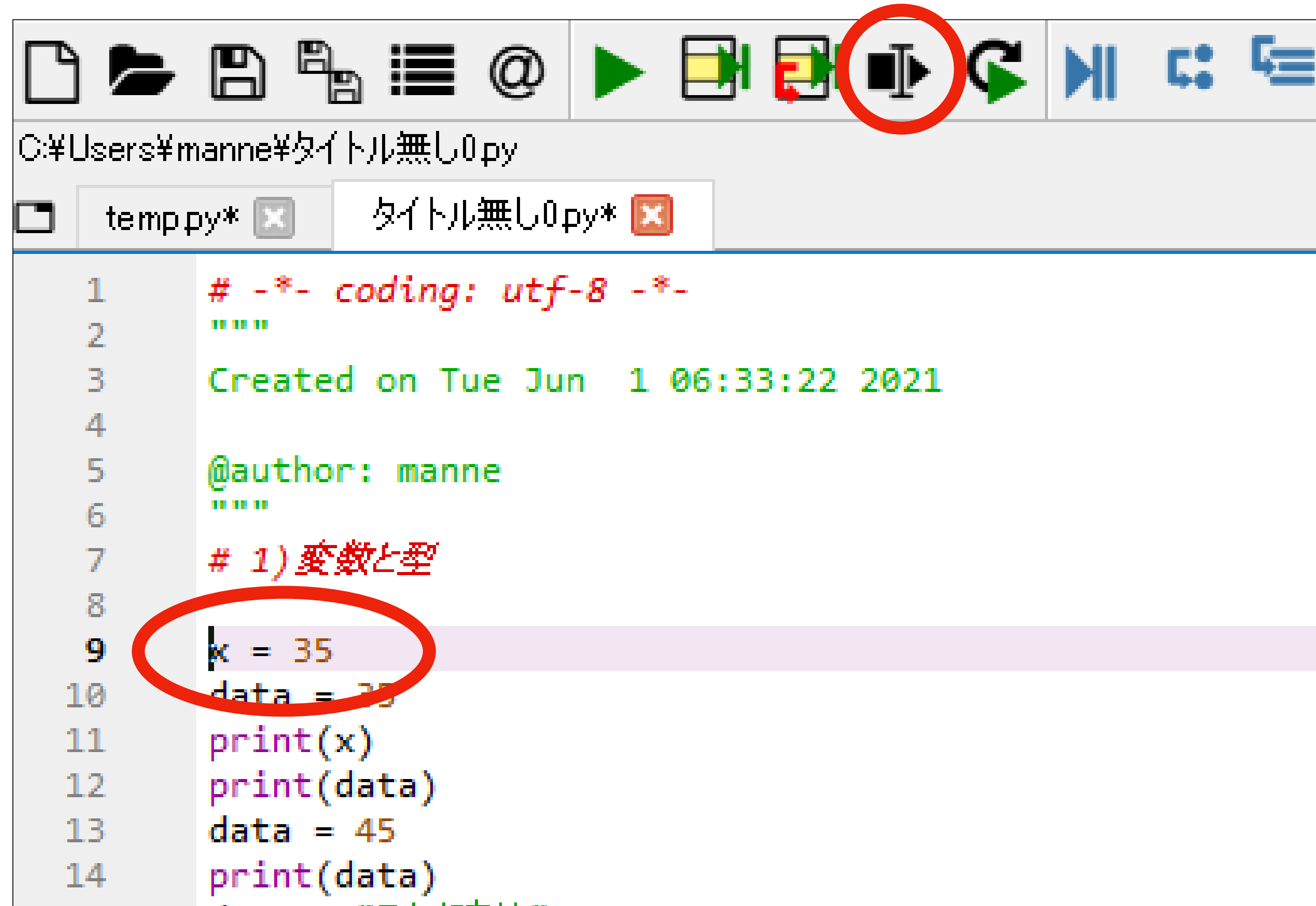
拡大するとこのようになっております。



```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Jun  1 06:28:08 2021
4
5  @author: manne
6  """
7  # 1) 変数と型
8
9  x = 35
10 data = 35
11 print(x)
12 print(data)
13 data = 45
14 print(data)
15 data = "こんにちは"
16 print(data)
17 x = 35
18 print(type(x))
19 x2 = 3.5
20 print(type(x2))
21 data = "こんにちは"
22 print(type(data))
```

データの操作の仕方について

カーソルを” x= 35”の行に合わせて(クリックして)、 1 行ずつ実行を押してみましょう



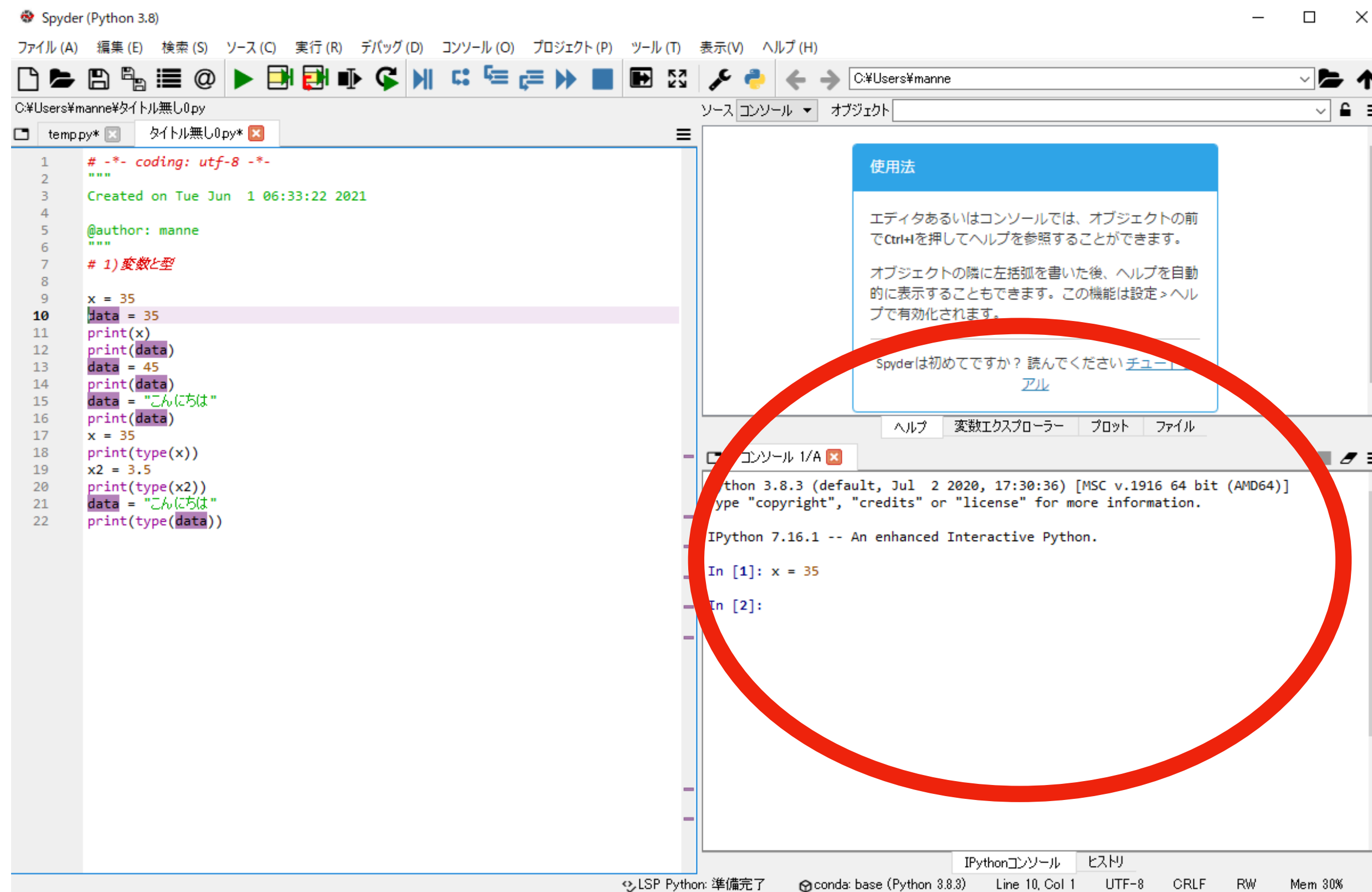
The screenshot shows a Python IDE window with the following content:

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Jun  1 06:33:22 2021
4
5  @author: manne
6  """
7  # 1) 変数と型
8
9  x = 35
10 data = 35
11 print(x)
12 print(data)
13 data = 45
14 print(data)
```

The toolbar at the top contains several icons. The 'Run' button, which is a green play button, is circled in red. The line containing the code `x = 35` is also circled in red.

データの操作の仕方について

右下のコンソール画面に実行結果が表示されます



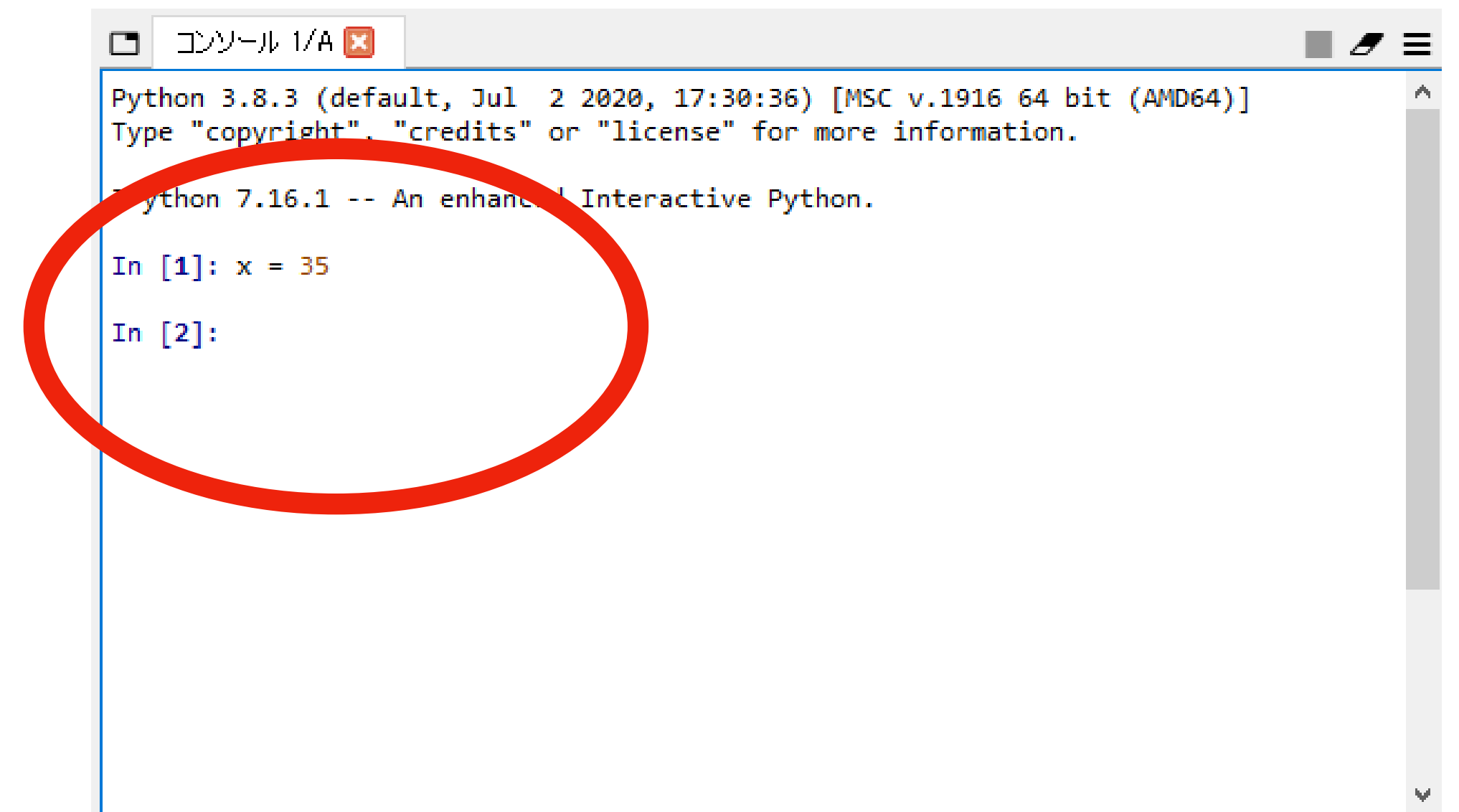
```
1  -*- coding: utf-8 -*-
2  """
3  Created on Tue Jun  1 06:33:22 2021
4
5  @author: manne
6  """
7  # 1) 変数と型
8
9  x = 35
10 data = 35
11 print(x)
12 print(data)
13 data = 45
14 print(data)
15 data = "こんにちは"
16 print(data)
17 x = 35
18 print(type(x))
19 x2 = 3.5
20 print(type(x2))
21 data = "こんにちは"
22 print(type(data))
```

Python 3.8.3 (default, Jul 2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]: x = 35

In [2]:



```
Python 3.8.3 (default, Jul 2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

Python 7.16.1 -- An enhanced Interactive Python.

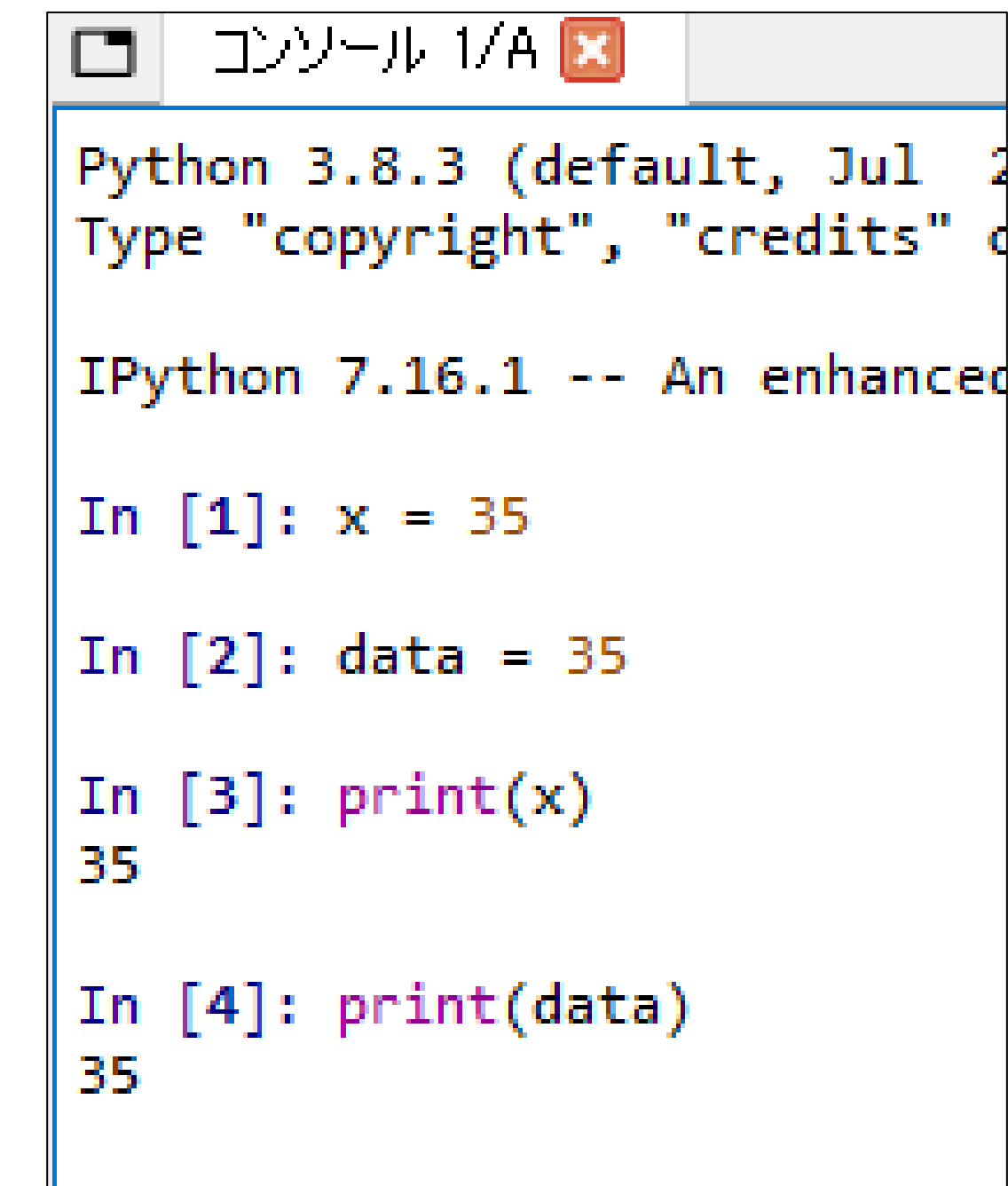
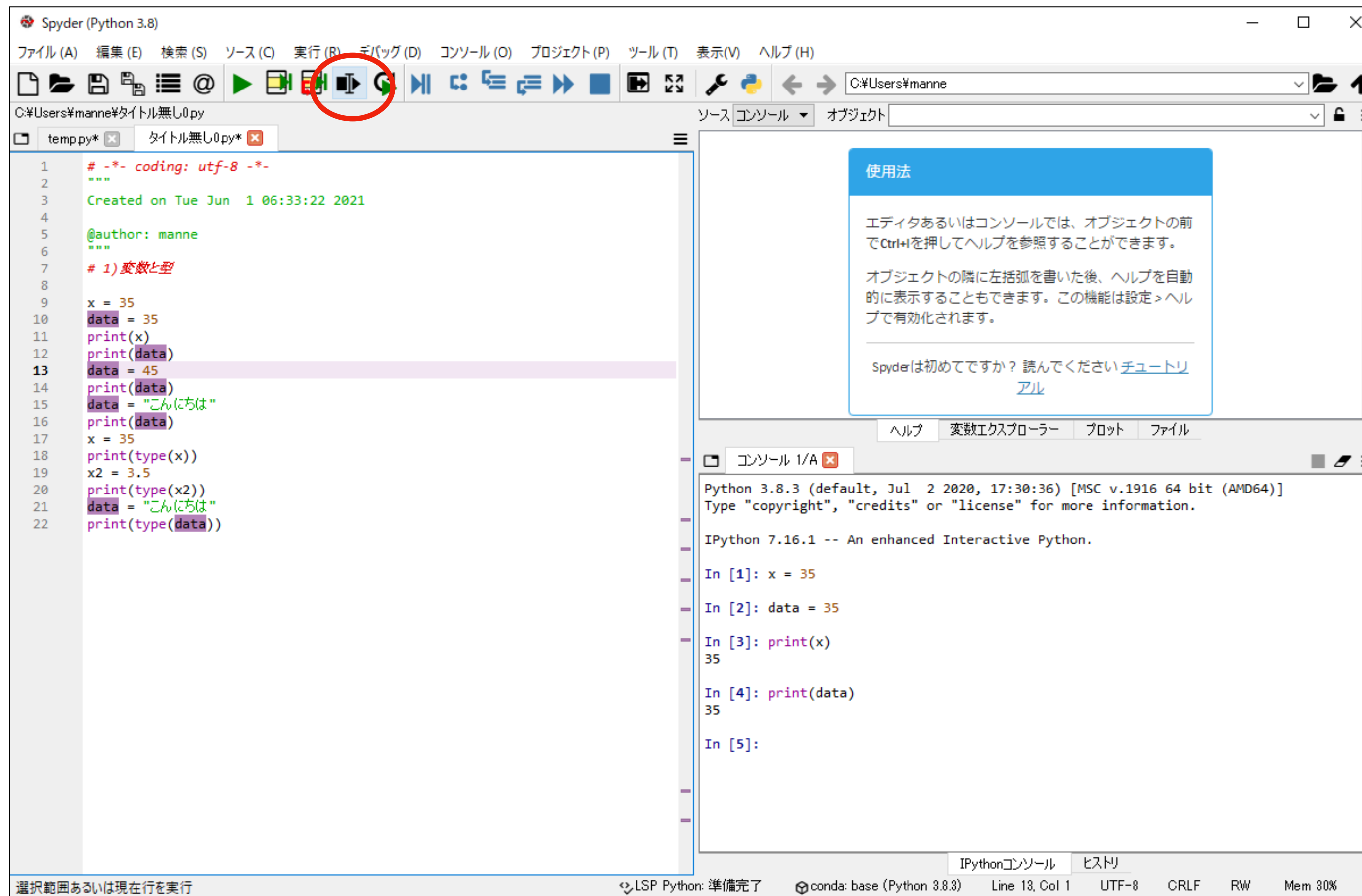
In [1]: x = 35

In [2]:
```

このように、エディタで書いた内容(プログラム)の実行結果が右下のコンソール画面に表示されます

データの操作の仕方について

続けて 1 行ずつ実行を押していきましょう
(この図では 4 行実行しています)



クリックすると自動的に次の行に移動します

1 行ずつ実行結果が表示されます

変数の扱い方

データを保存したものを”変数”と言います

変数名	=	データ
-----	---	-----

独自につける変数名に対して、データを代入します

入力

a という変数名に3という値を代入 a = 3 # aを出力する print(a)
--

“=”は数学的には等しいという意味ですが、多くのプログラミング言語では、左側の変数に右側の値を代入するという意味になります。

出力

3

(#はコメントといい、行の頭に#のある行はコメント行として、何も結果に反映されません)

関数について

プログラミングでは“関数”というものを扱います

関数は、四則演算、繰り返し、比較演算など何かしらの処理を保存したもので、データを関数に与えることで処理を行ったデータを返してくれます。



関数は自作することも可能ですが、pythonではあらかじめ便利な関数が多数用意されています。(組み込み関数といいます)

print関数

print(引数) : ()の中身を出力する

入力

```
# a という変数名に3という値を代入  
a = 3  
# aを出力する  
print(a)
```

出力

3

print()関数は最もよく使う組み込み関数の1つです。関数の()の中を引数(ひきすう)と言い、ここに変数やデータを入れるとコンソールに中身を出力してくれます。

ここではaに3を代入しているので、aをprint関数で出力し、3が表示されます。

1行ずつ実行してみましょう

```
x = 35  
data = 35  
print(x)  
  
print(data)
```

←xという変数に35を代入するという意味
←dataという変数に35を代入するという意味
←変数xを出力しろという意味
→ 35
←変数dataを出力しろという意味
→ 35

```
In [1]: x = 35  
In [2]: data = 35  
In [3]: print(x)  
35  
In [4]: print(data)  
35
```

1行ずつ実行してみましょう

```
x = 35  
data = 35  
print(x)
```

```
print(data)
```

```
data = 45
```

```
print(data)
```

```
data = "こんにちは"
```

```
print(data)
```

←xという変数に35を代入するという意味

←dataという変数に35を代入するという意味

←変数xを出力しろという意味

→ 35

←変数dataを出力しろという意味

→ 35

←dataという変数に45を代入するという意味。
(35から45に上書きされる。)

←変数dataを出力しろという意味

→ 45

←dataという変数に"こんにちは"という文字列を代入するという意味。
(45から"こんにちは"に上書きされる。)

←変数dataを出力しろという意味

→ こんにちは

```
In [1]: x = 35  
In [2]: data = 35  
In [3]: print(x)  
35  
In [4]: print(data)  
35  
In [5]: data = 45  
In [6]: print(data)  
45  
In [7]: data = "こんにちは"  
In [8]: print(data)  
こんにちは
```

Pythonの値には「型」という概念が存在する

変数に入れるデータは、整数なのか、小数なのか、文字のデータ(文字列)なのかを区別する必要があります。これを型といい、型には「文字列(str)」、「整数型(int)」、「小数型(float)」、「リスト型(list)」、「タプル型(tuple)」などがあります。
組み込み関数である`type(引数)`で調べることが出来ます。

```
x = 35  
  
print(type(x))  
  
x2 = 3.5  
  
print(type(x2))  
  
data = “こんにちは”  
  
print(type(data))
```

←`type(x)`を出力しろという意味
→ int

←`type(x2)`を出力しろという意味
→ float

←`data`という変数に“こんにちは”という文字列を代入するという意味。

←`type(data)`を出力しろという意味
→ string

リストについて

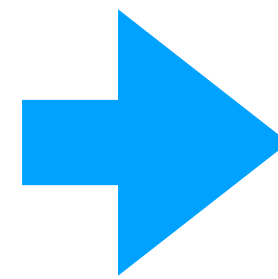
リスト型は、文字列や数値を複数まとめて格納する型です。

データが数多くあった場合、全てを変数に代入して扱うのは大変です。
リストでは、沢山のデータを1つの変数の中に代入することが出来ます。

変数 = [要素1, 要素2, 要素3, ...]

(リストではそれぞれのデータのことを要素と言います)

a = 1
b = 1
c = 2
d = 4
e = 6
f = 8



a = [1, 1, 2, 4, 6, 8]

#2) リストの作成

1.txtから #2)リストの作成をコピーしましょう

```
# 2)リストの作成
```

```
x = [1,2,3,4,5]
```

```
print(x)
```

```
print(type(x))
```

```
print(x[0])
```

```
print(x[4])
```

リストについて

```
x = [1, 2, 3, 4, 5]
```

```
print(x)
```

```
print(type(x))
```

```
print(x[0])
```

```
print(x[4])
```

← 変数xに要素1に1、要素2に2、要素3に3、
要素4に4、要素5に5をリストとして代入

→ [1, 2, 3, 4, 5]

→ list

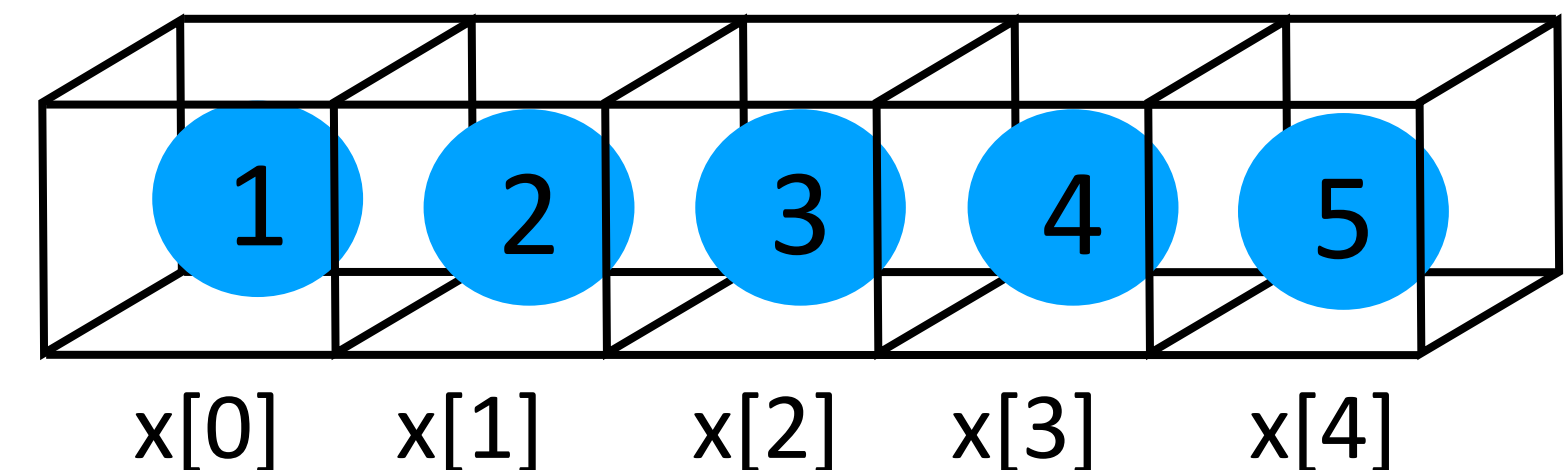
← xの1番目の要素を表示する

→ 1

← xの5番目の要素を表示する

→ 5

このリストのように、複数の値を変数に入れて操作できる
形のものを配列と言います。



#2) リストの作成

1.txtから #2)リストの作成をコピーして、実行してみましょう

入力

```
# 2)リストの作成
```

```
x = [1,2,3,4,5]
```

```
print(x)
```

```
print(type(x))
```

```
print(x[0])
```

```
print(x[4])
```

出力

```
[1,2,3,4,5]
```

```
list
```

```
1
```

```
5
```


リストについて

結果が共有出来ていますでしょうか

The screenshot displays a Jupyter Notebook environment. The left pane contains a Python script with the following code:

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Jun  1 06:33:22 2021
4
5  @author: manne
6  """
7  # 1) 変数と型
8
9  x = 35
10 data = 35
11 print(x)
12 print(data)
13 data = 45
14 print(data)
15 data = "こんにちは"
16 print(data)
17 x = 35
18 print(type(x))
19 x2 = 3.5
20 print(type(x2))
21 data = "こんにちは"
22 print(type(data))
23
24 # 2) リストの作成
25
26 x = [1,2,3,4,5]
27 print(x)
28 print(type(x))
29 print(x[0])
30 print(x[4])
31
```

The right pane shows the Variable Explorer, which is circled in red. It displays the following variables:

名前	型	サイズ	値
data	str	1	こんにちは
x	list	5	[1, 2, 3, 4, 5]
x2	float	1	3.5

Below the Variable Explorer, the IPython Console is also circled in red. It shows the execution history of the code, including the creation and manipulation of the variables.

```
<class 'float'>
In [13]: data = "こんにちは"
In [14]: print(type(data))
<class 'str'>
In [15]: x = [1,2,3,4,5]
In [16]: print(x)
[1, 2, 3, 4, 5]
In [17]: print(type(x))
<class 'list'>
In [18]: print(x[0])
1
In [19]: print(x[4])
5
In [20]:
```

The bottom status bar indicates the environment: LSP Python: 準備完了, conda: base (Python 3.8.3), Line 31, Col 1, UTF-8, CRLF, RW, Mem 30%.

変数エクスペローラー
をクリックするとこれ
まで作成した変数の情
報が表示される

課題に取り組んでみよう

(Windowsの人)

#1から10の要素を持つリストをtestという変数に代入しよう

#testを出力しよう

#testの3番目と5番目の要素を出力しよう

#len(test)、max(test)、min(test)を出力しよう

ライブラリについて

Pythonの機能を拡張する機能の1つ

ライブラリを読み込むと、ライブラリに含まれる多くの関数などの機能を追加できる

標準ライブラリと外部ライブラリがある

標準ライブラリ

インストールすればすぐ使える

外部ライブラリ

非常に多くの種類があるので、必要なものをダウンロードしてから使用する

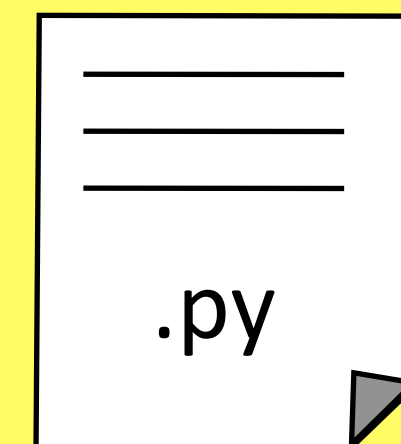
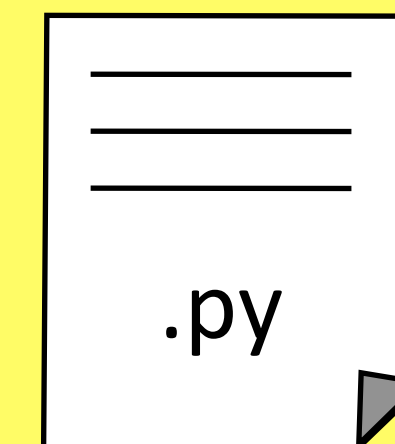
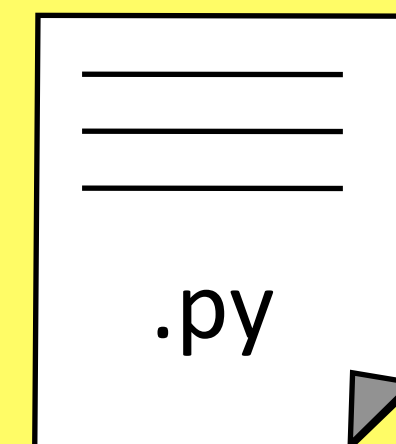
ここでは次回から多用する
numpy、pandas、matplotlib
の3つライブラリを使用してみます

ライブラリ 複数のパッケージをまとめたもの

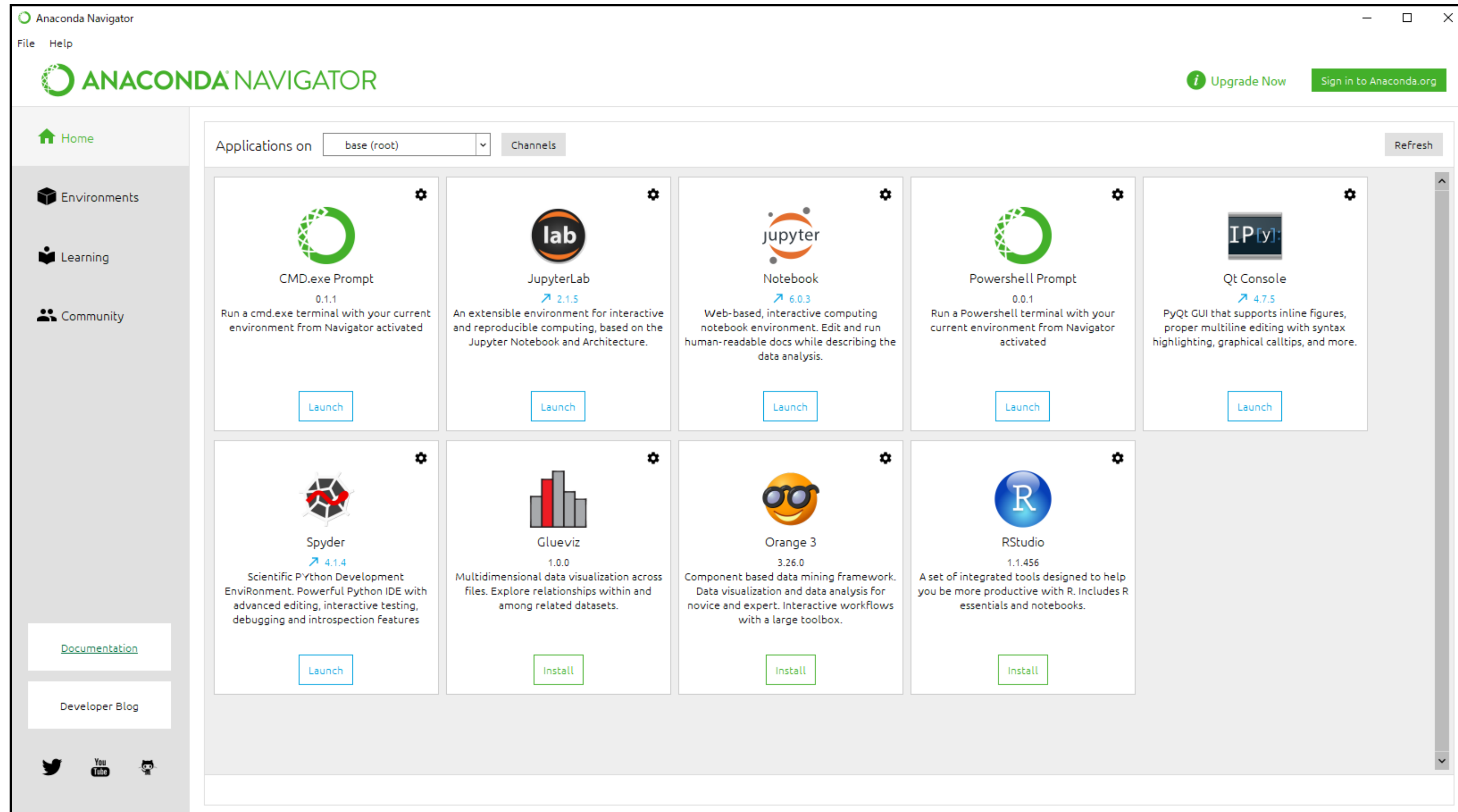
パッケージ 複数のモジュールをまとめたもの

モジュール

複数の.pyファイルをまとめたもの

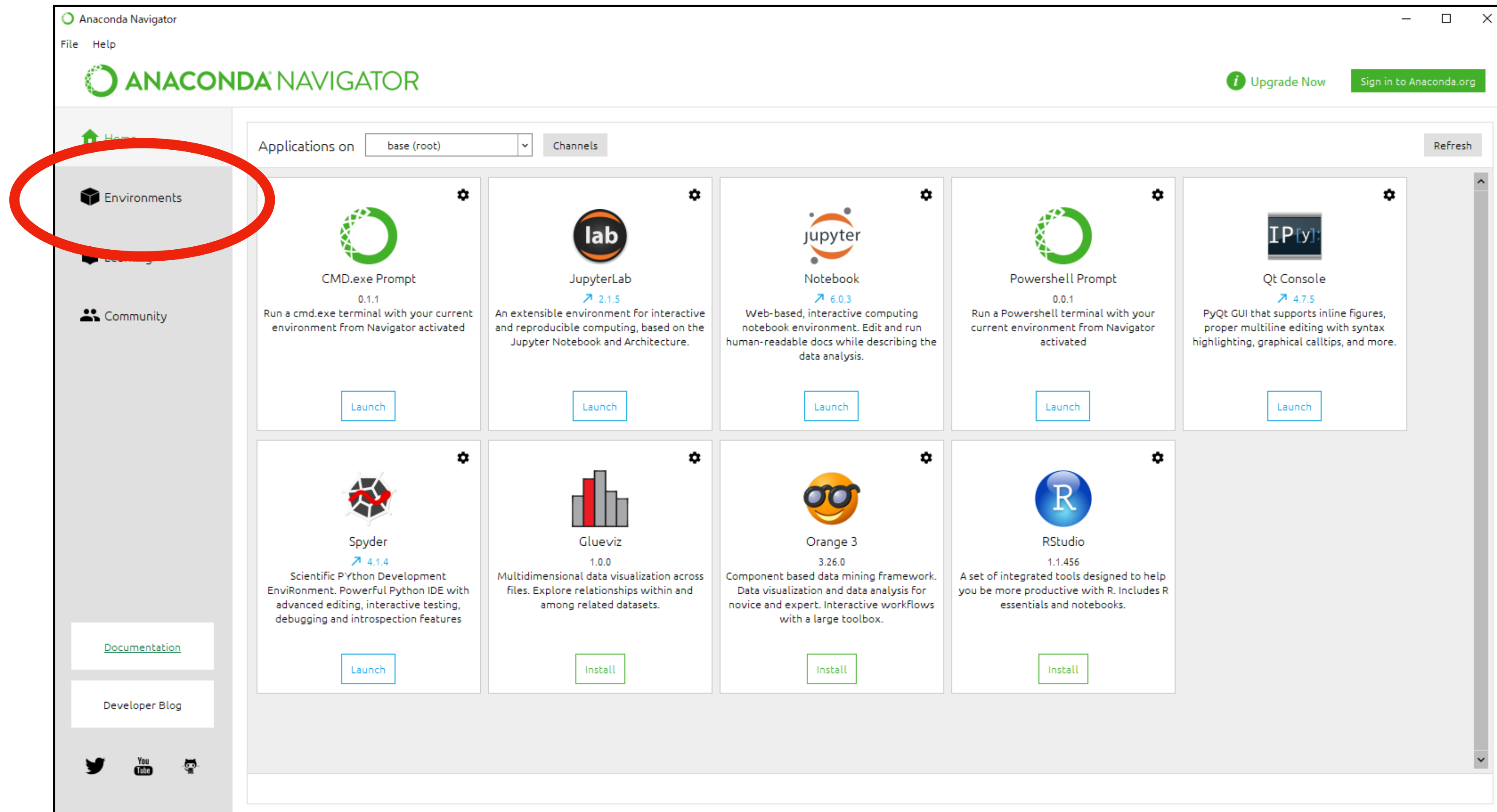


医療とAI・ビッグデータ入門のための仮想環境を作ろう



Anaconda Navigatorのホーム画面を開く

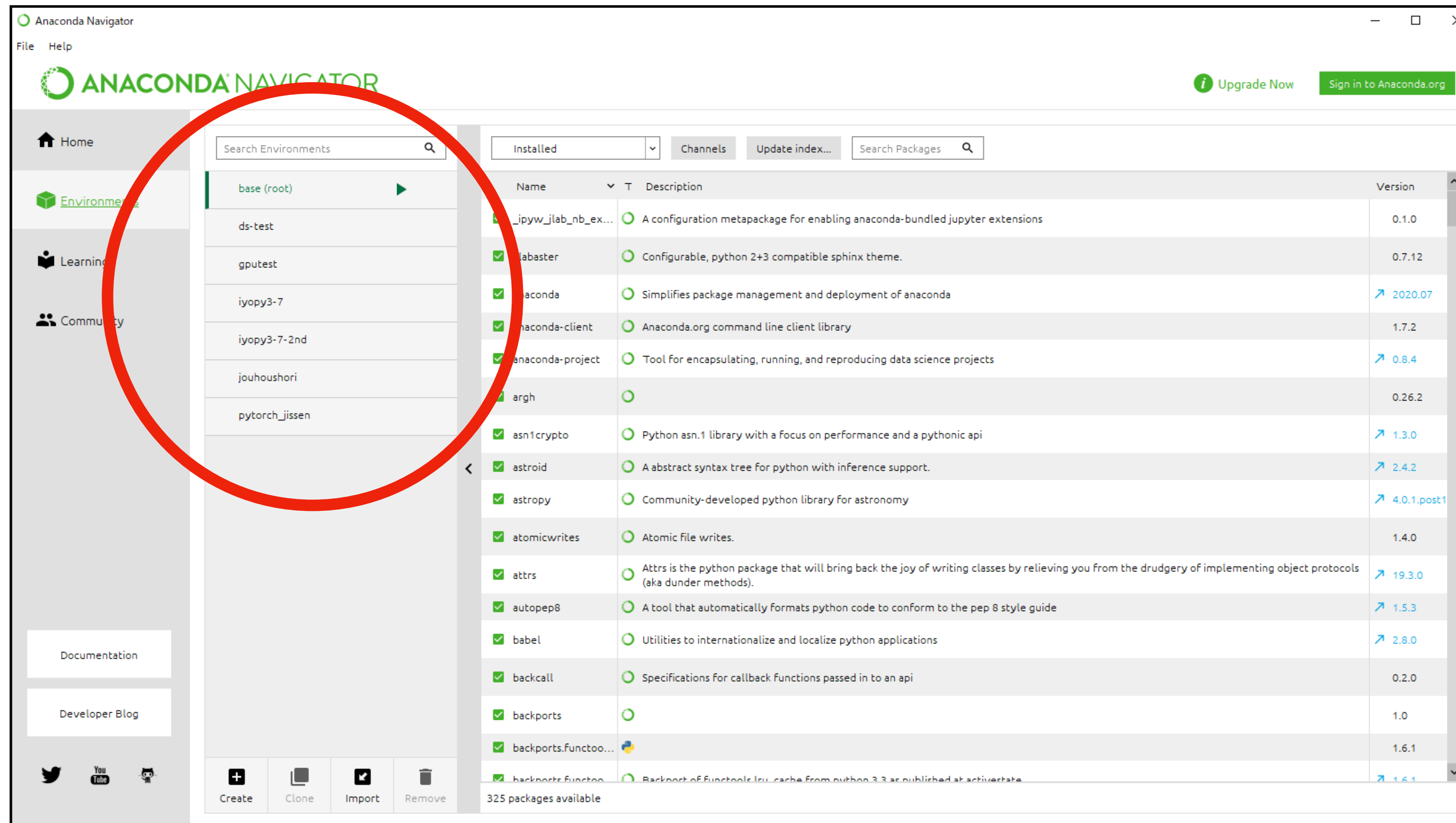
医療とAI・ビッグデータ入門のための仮想環境を作ろう



Environmentをクリック

医療とAI・ビッグデータ入門のための仮想環境を作ろう

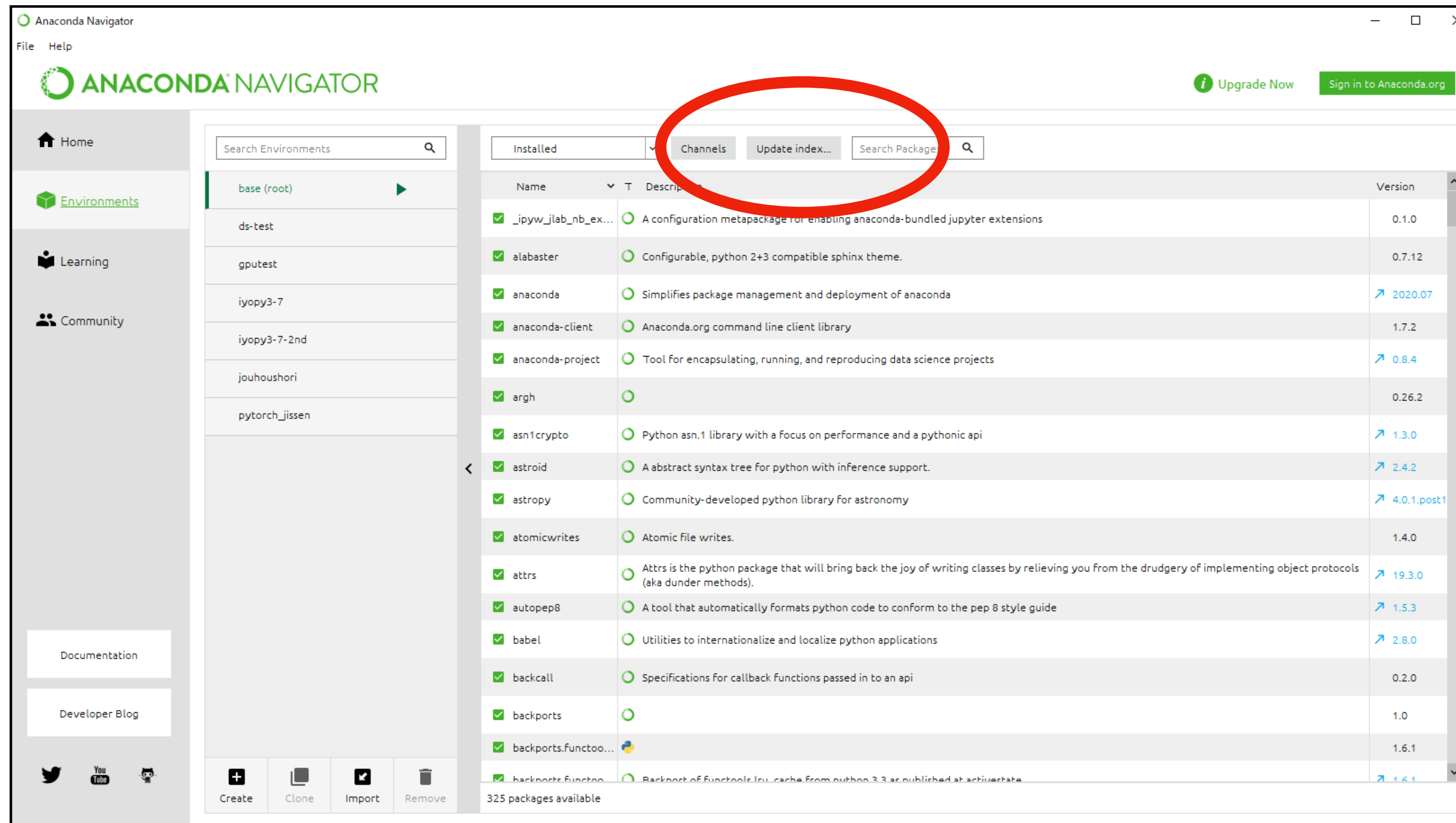
仮想環境の一覧



anacondaでは自分の好きな数だけ色々なバージョンのpythonやpythonのツールを作ることができます。

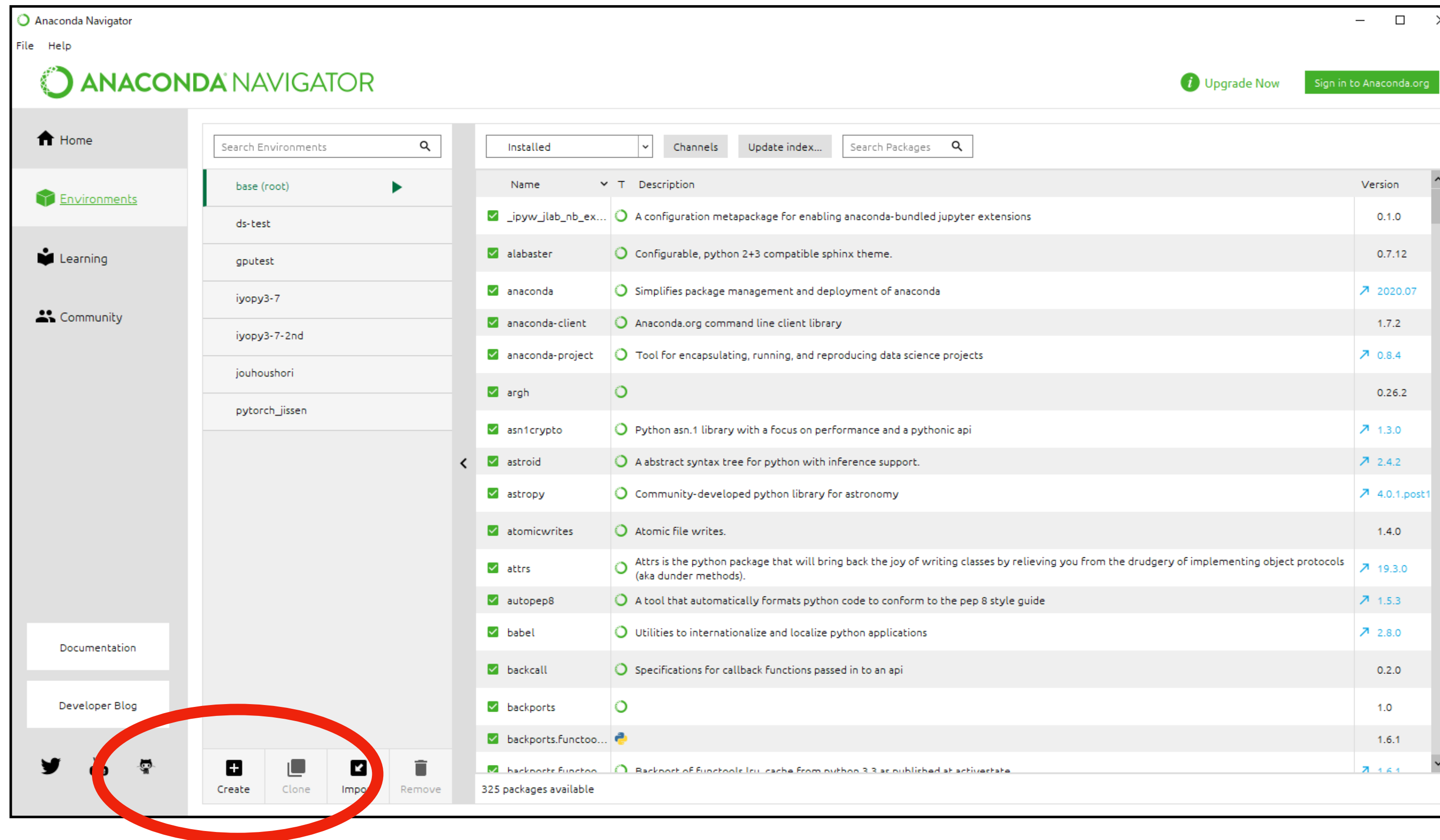
医療とAI・ビッグデータ入門のための仮想環境を作ろう

最初にupdateをクリックして更新しておきます



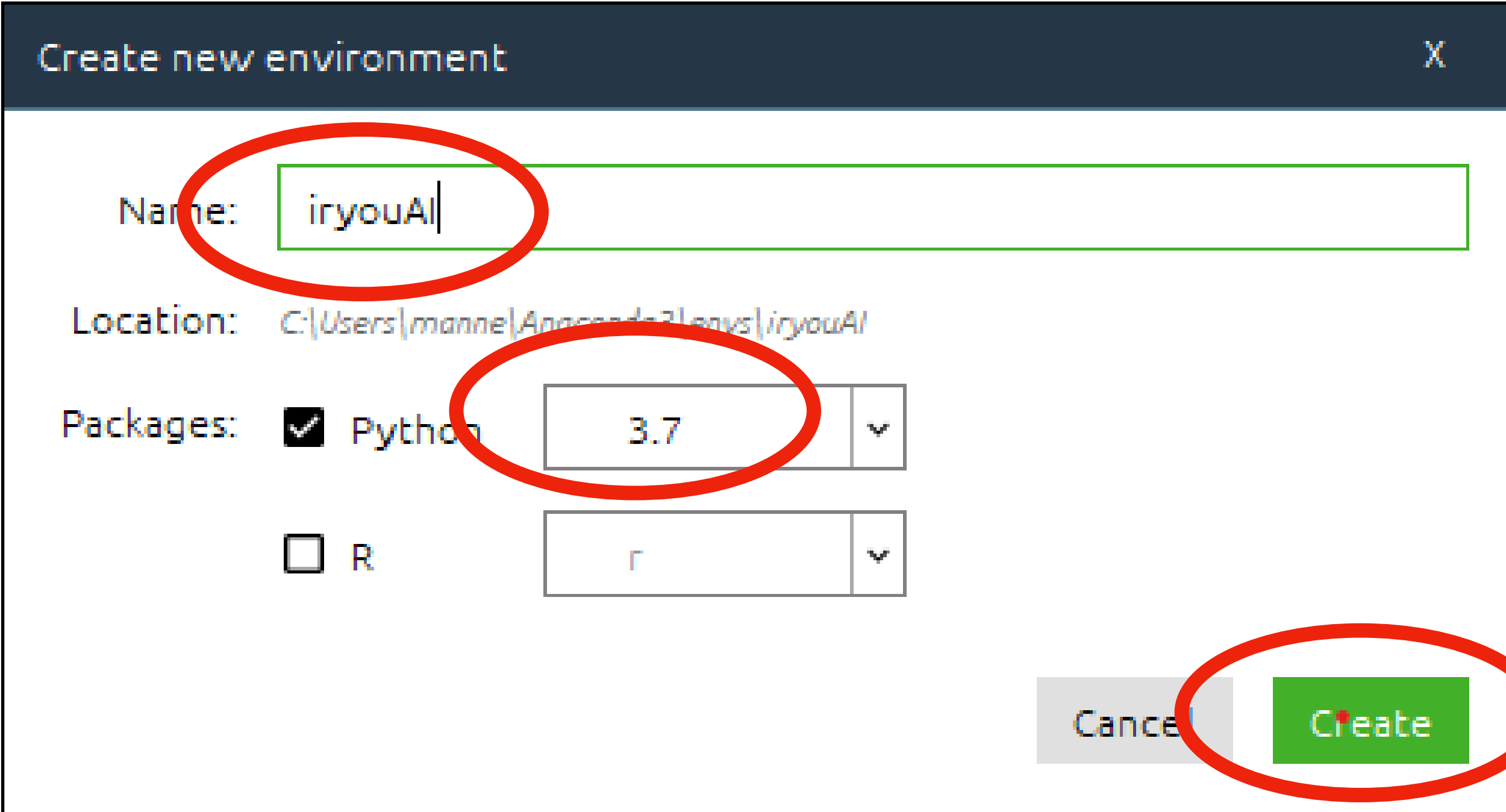
医療とAI・ビッグデータ入門のための仮想環境を作ろう

Createをクリックします



医療とAI・ビッグデータ入門のための仮想環境を作ろう

NameをiryouAI、pythonを3.7を選んで、createをクリックします

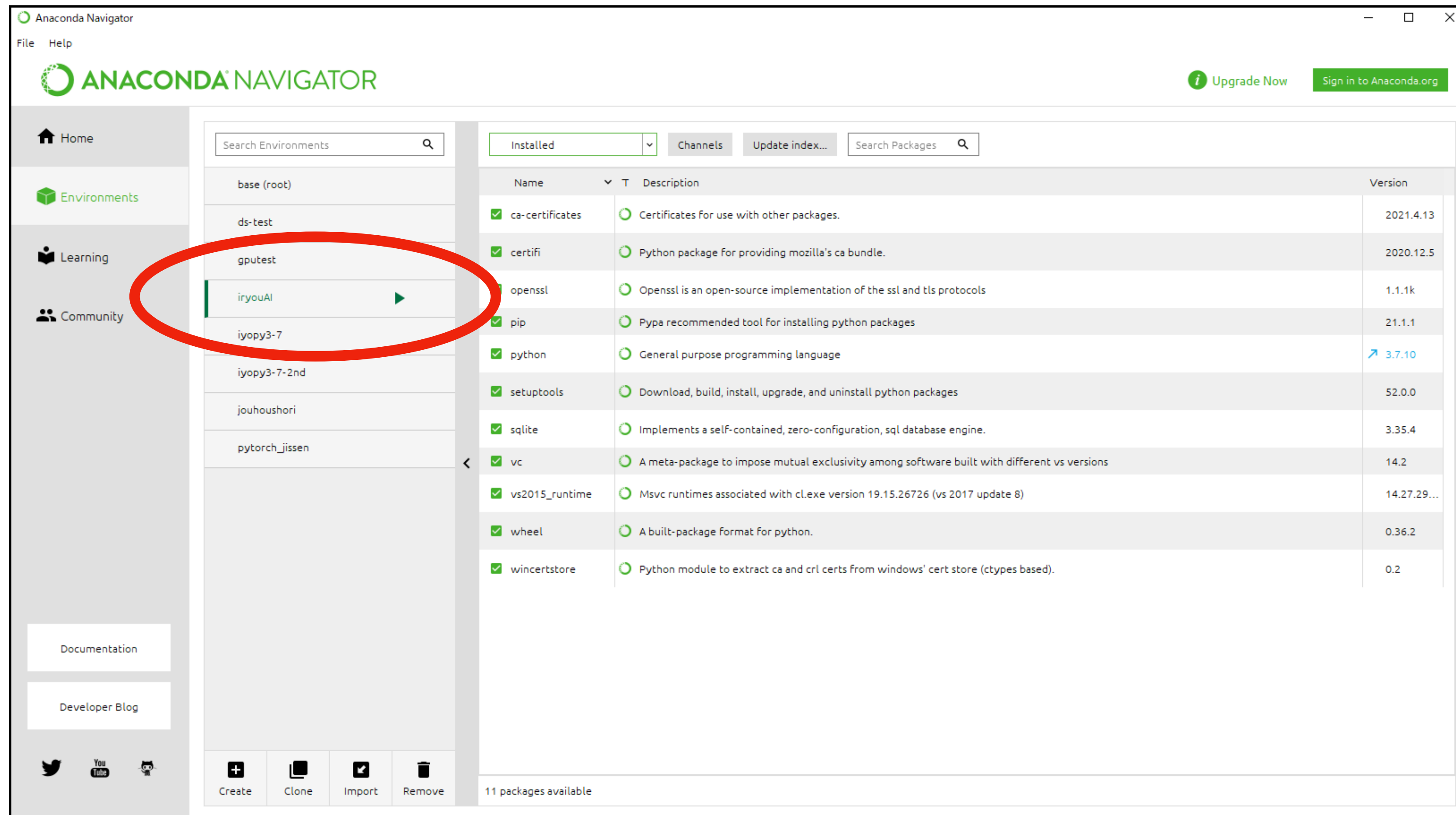


The screenshot shows a 'Create new environment' dialog box. The 'Name' field is set to 'iryouAI'. The 'Location' field shows the path 'C:\Users\manne\AppData\Local\Programs\Python\Python37\envs\iryouAI'. Under 'Packages', the 'Python' checkbox is checked and the version '3.7' is selected. The 'R' checkbox is unchecked. The 'Create' button is highlighted in green. Red circles are drawn around the 'Name' field, the 'Python 3.7' selection, and the 'Create' button.

Field	Value
Name	iryouAI
Location	C:\Users\manne\AppData\Local\Programs\Python\Python37\envs\iryouAI
Packages	<input checked="" type="checkbox"/> Python 3.7 <input type="checkbox"/> R r
Buttons	Cancel Create

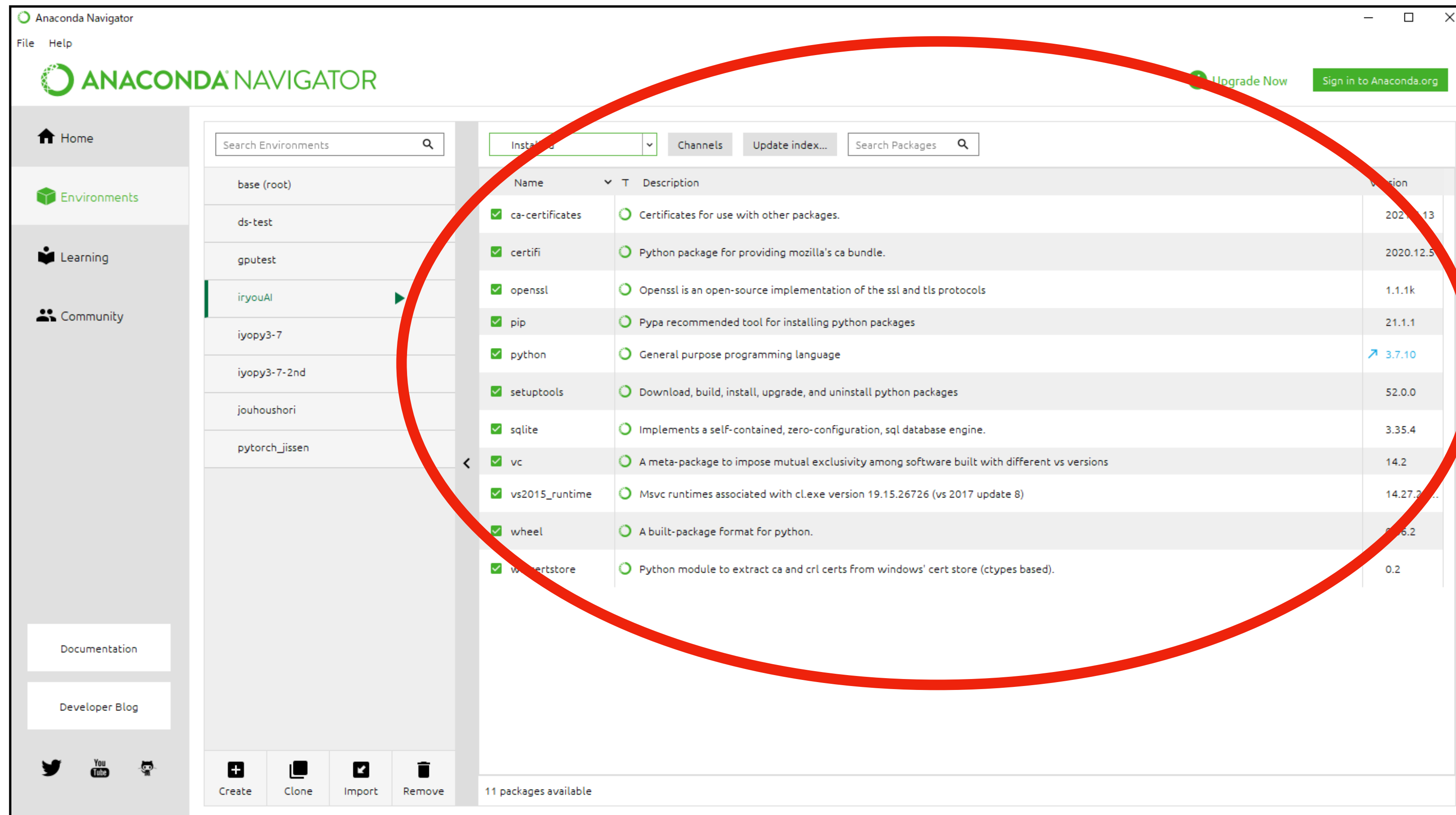
医療とAI・ビッグデータ入門のための仮想環境を作ろう

iryouAIという仮想環境が作れました



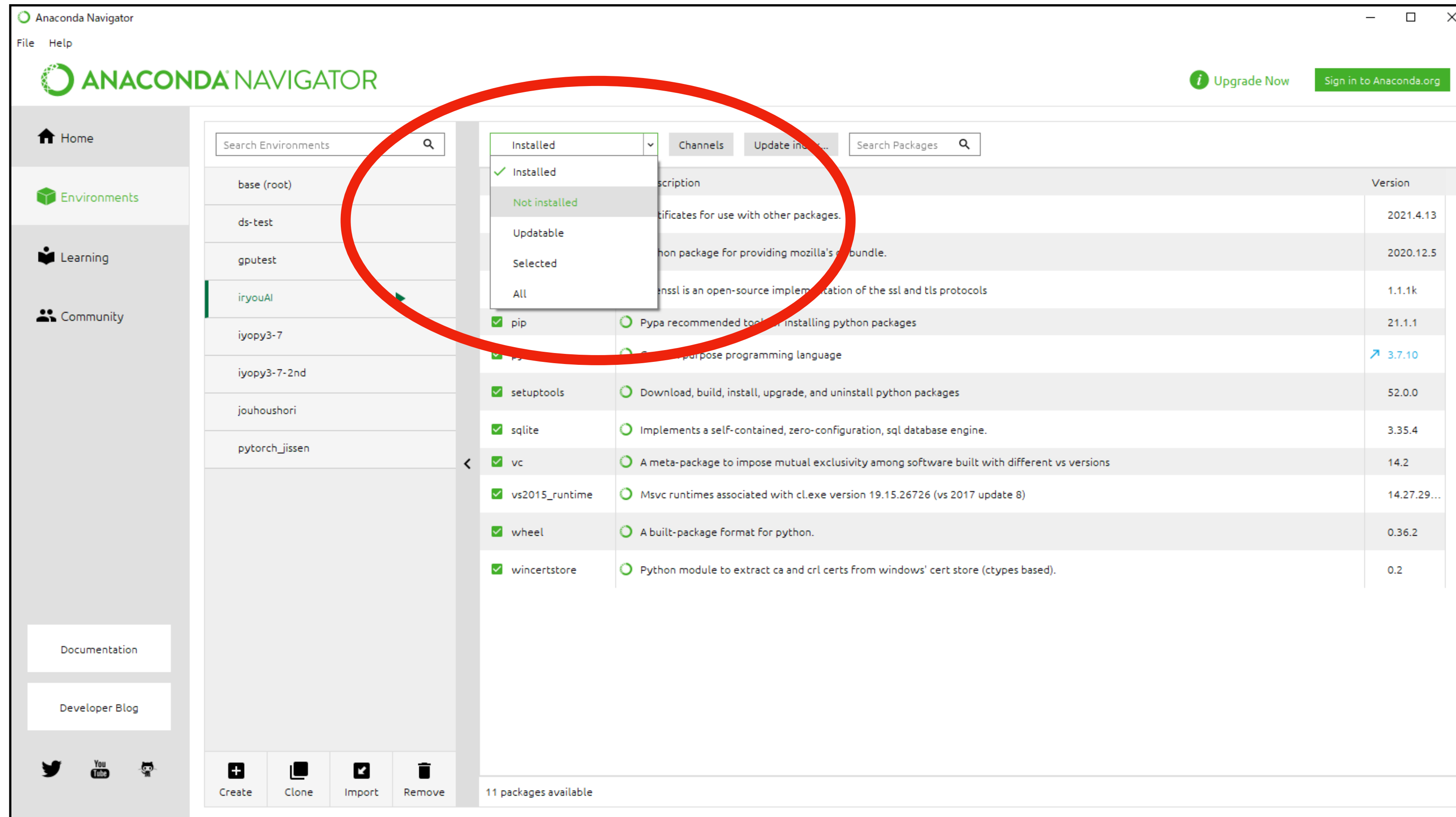
医療とAI・ビッグデータ入門のための仮想環境を作ろう

iryouAIに入っているライブラリ



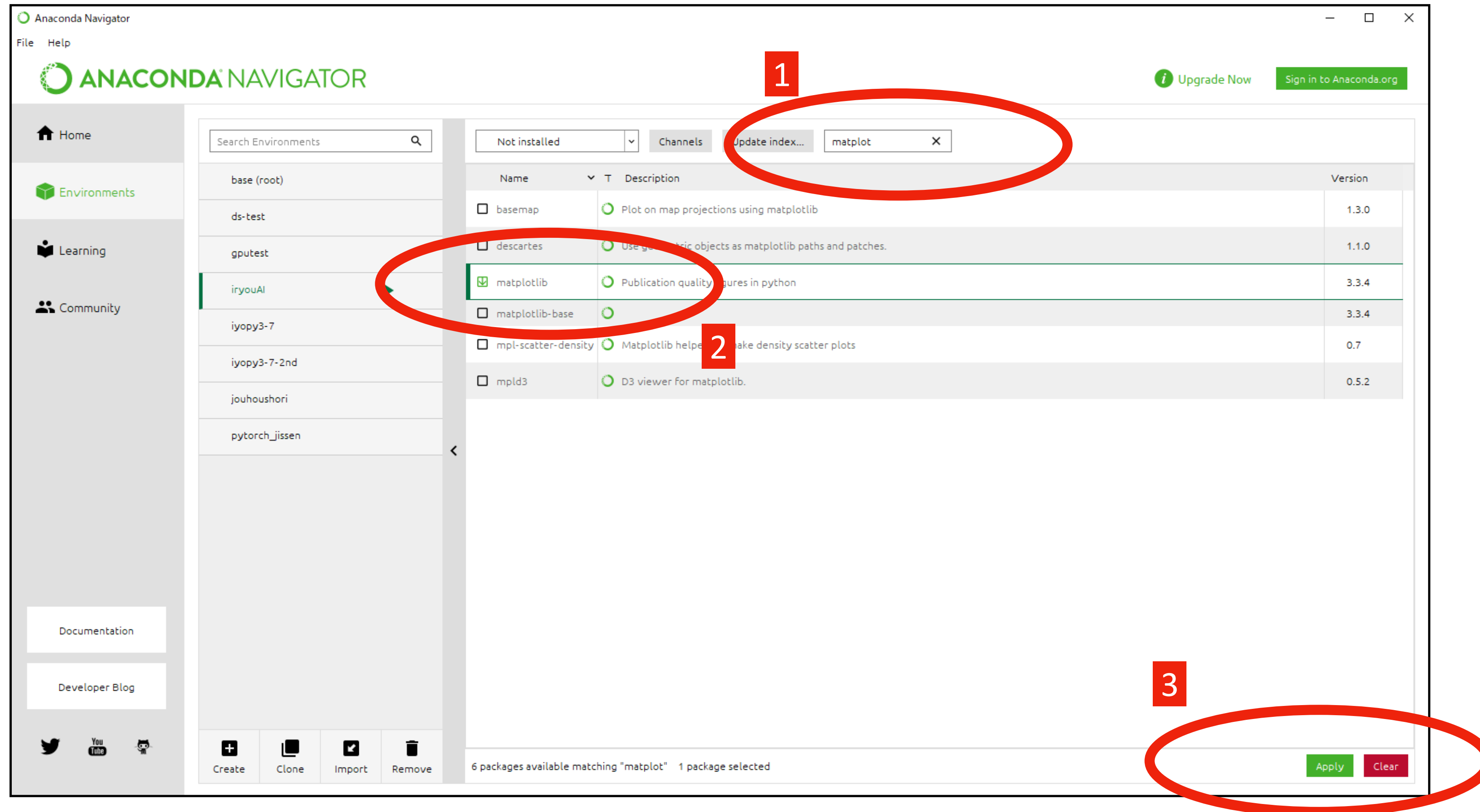
医療とAI・ビッグデータ入門のための仮想環境を作ろう

Not installedをクリックします



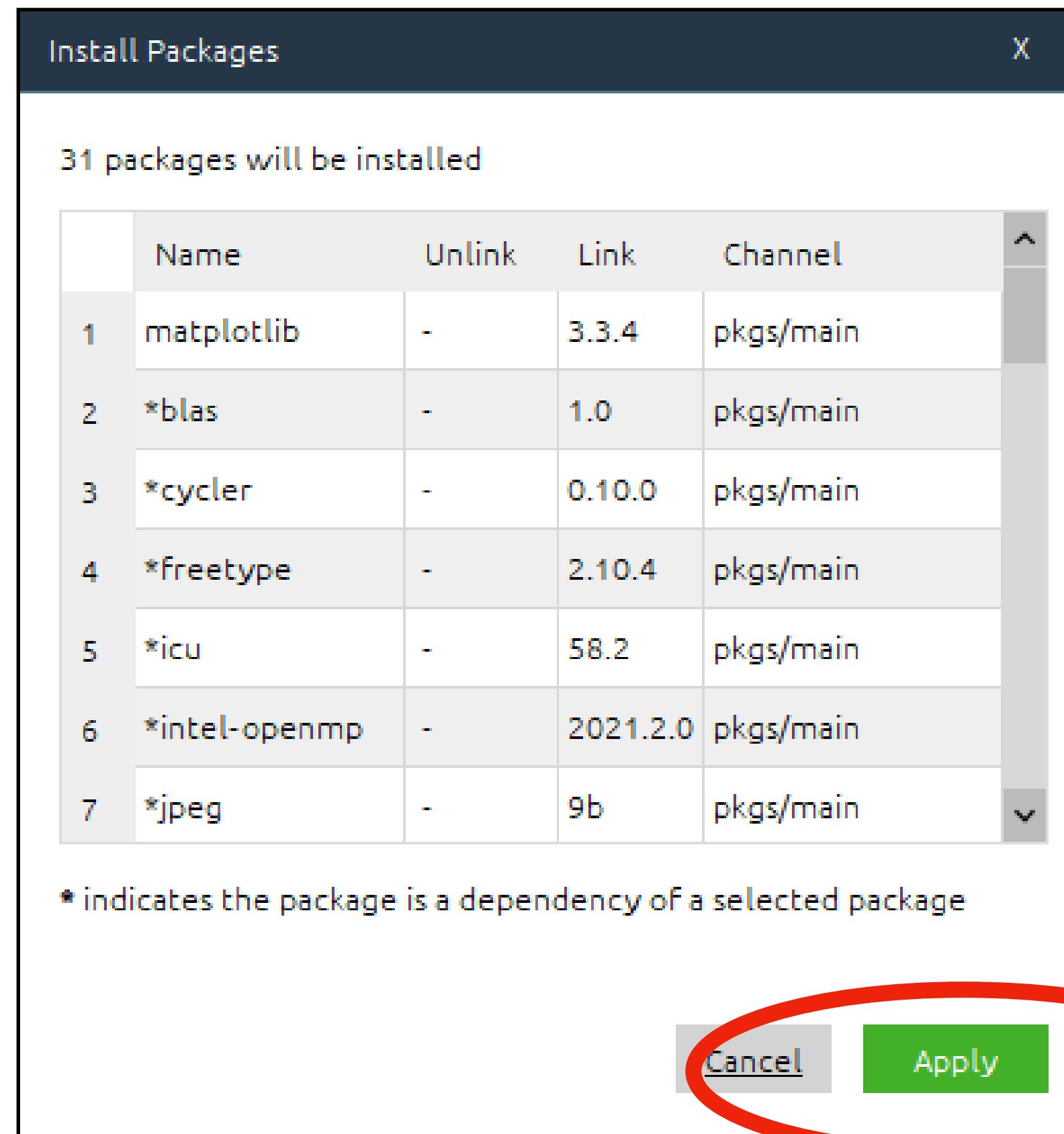
医療とAI・ビッグデータ入門のための仮想環境を作ろう

検索でmatplotlibと打ち込むと関連するライブラリが表示されるので、matplotlibにチェックを入れてapplyをクリックします



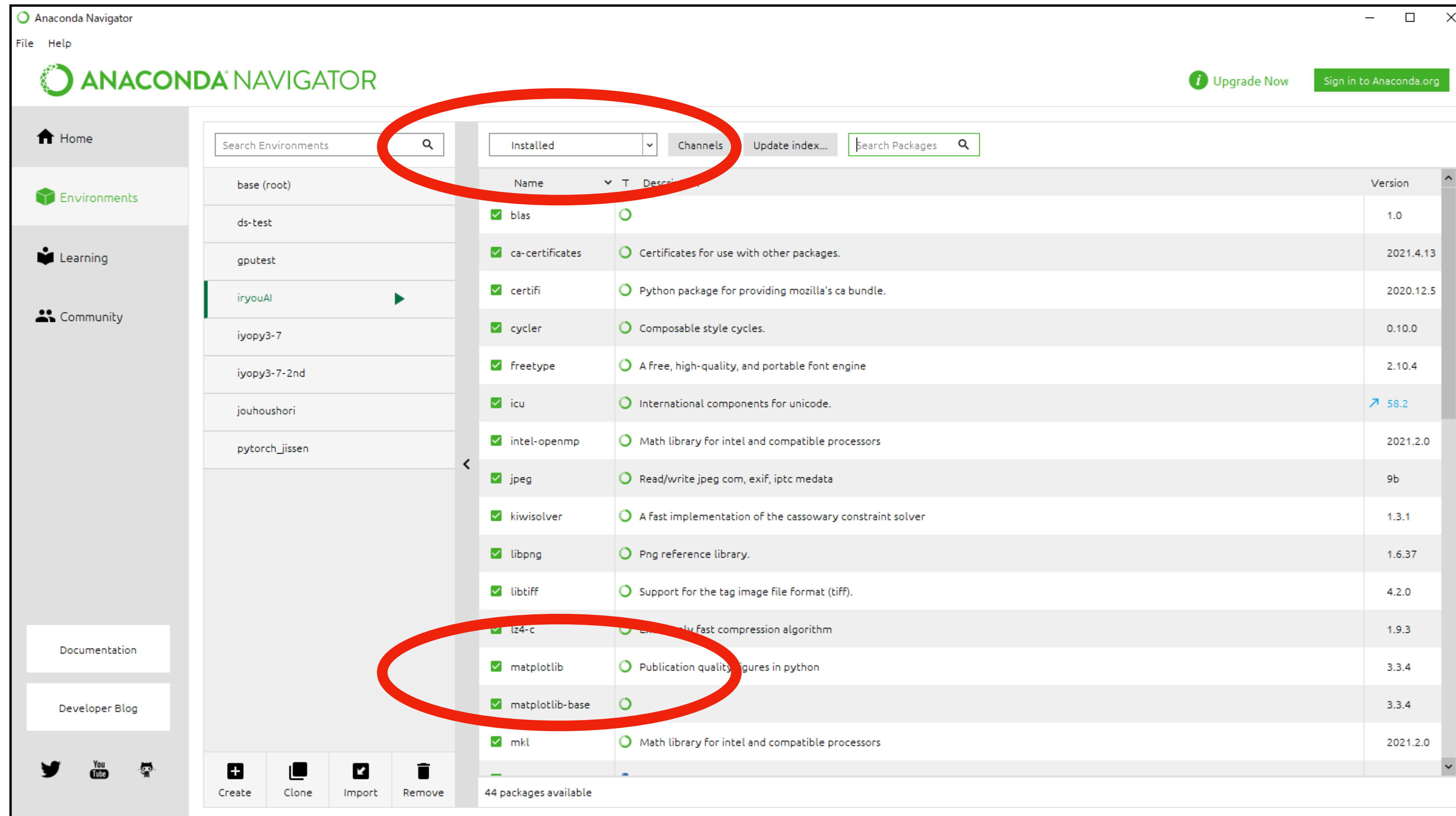
医療とAI・ビッグデータ入門のための仮想環境を作ろう

Applyをクリック



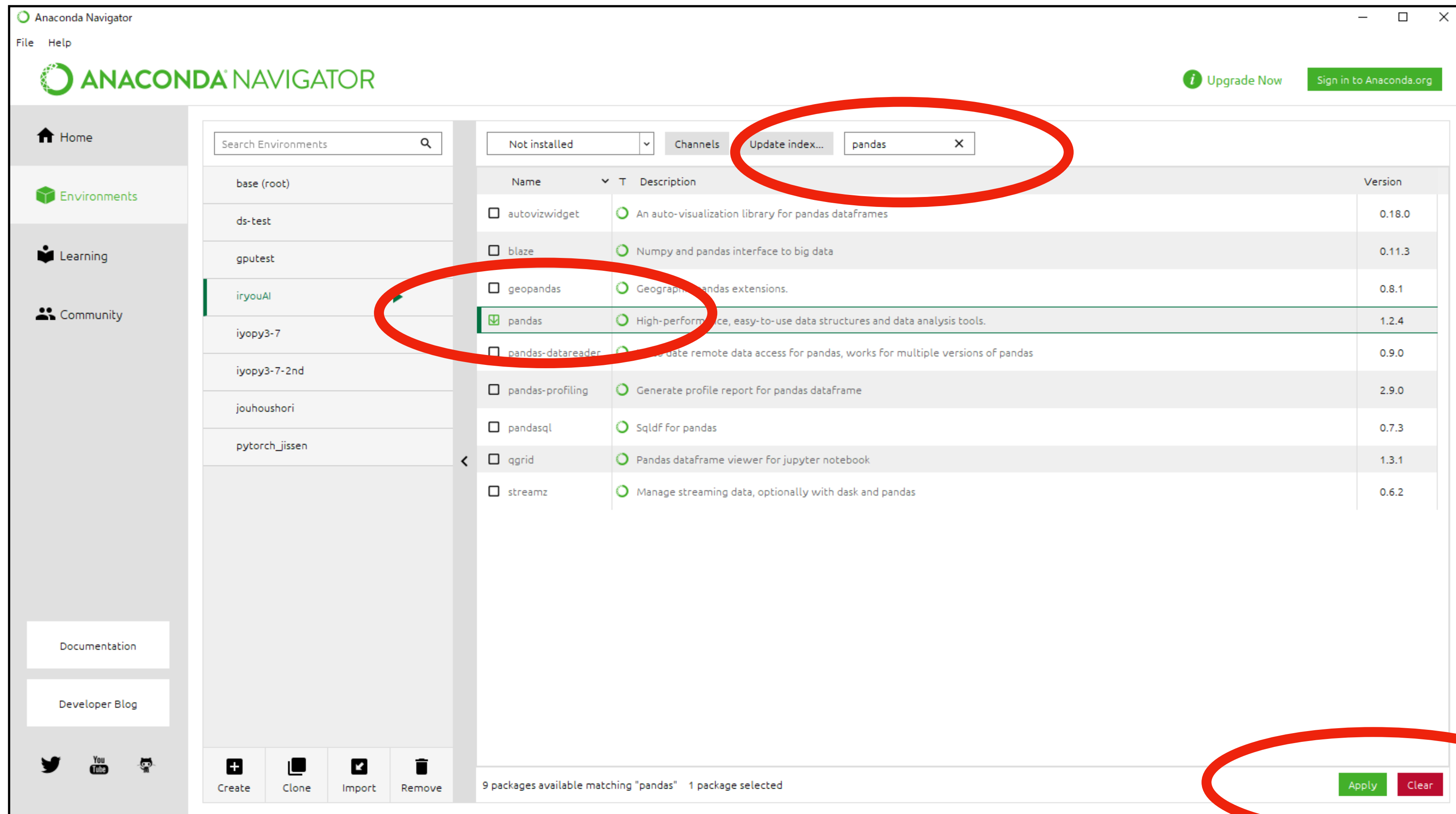
医療とAI・ビッグデータ入門のための仮想環境を作ろう

installedの一覧にmatplotlibが入っていることが確認できます



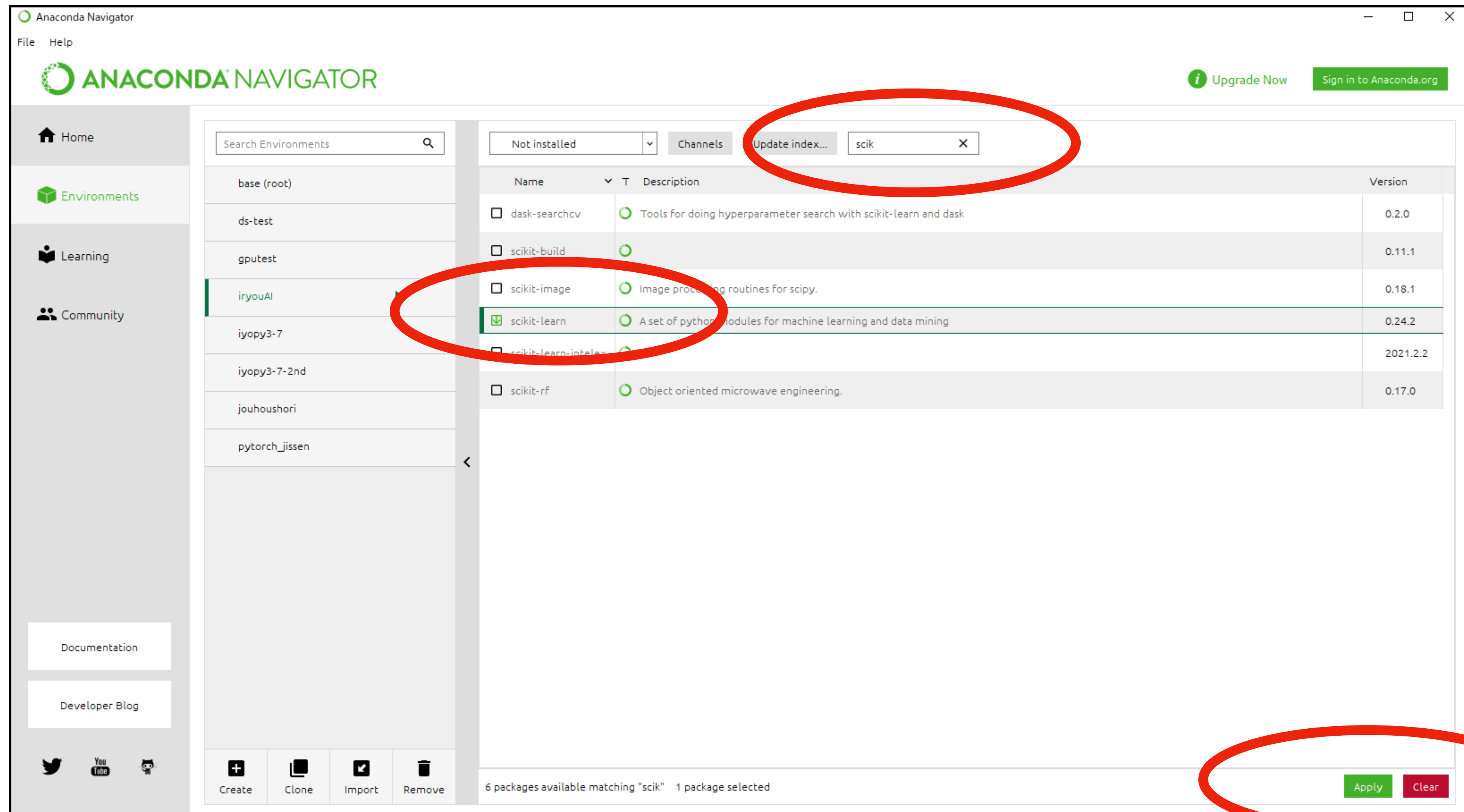
医療とAI・ビッグデータ入門のための仮想環境を作ろう

同様にpandasをインストールする



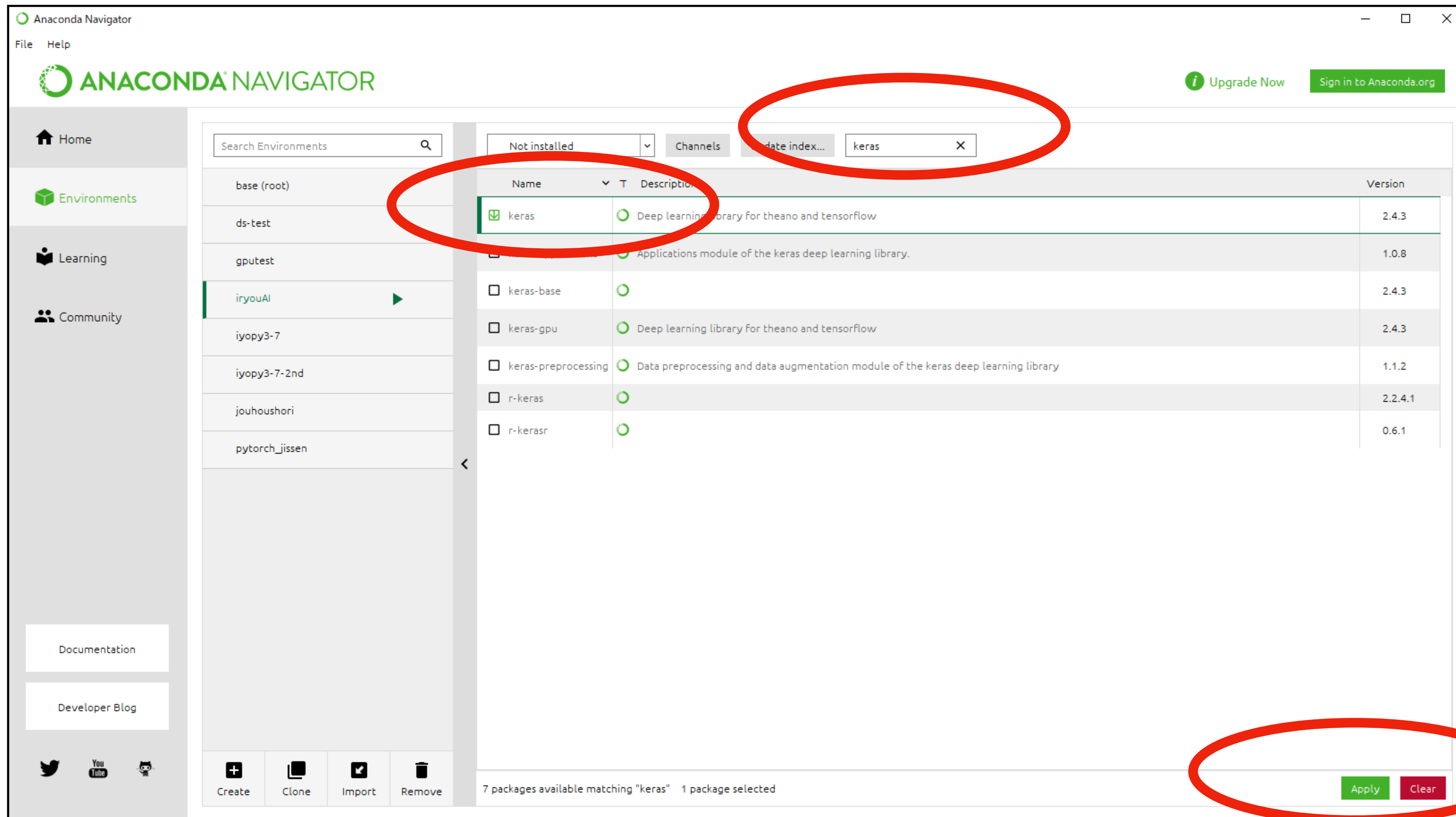
医療とAI・ビッグデータ入門のための仮想環境を作ろう

同様にscikit-learnをインストールする



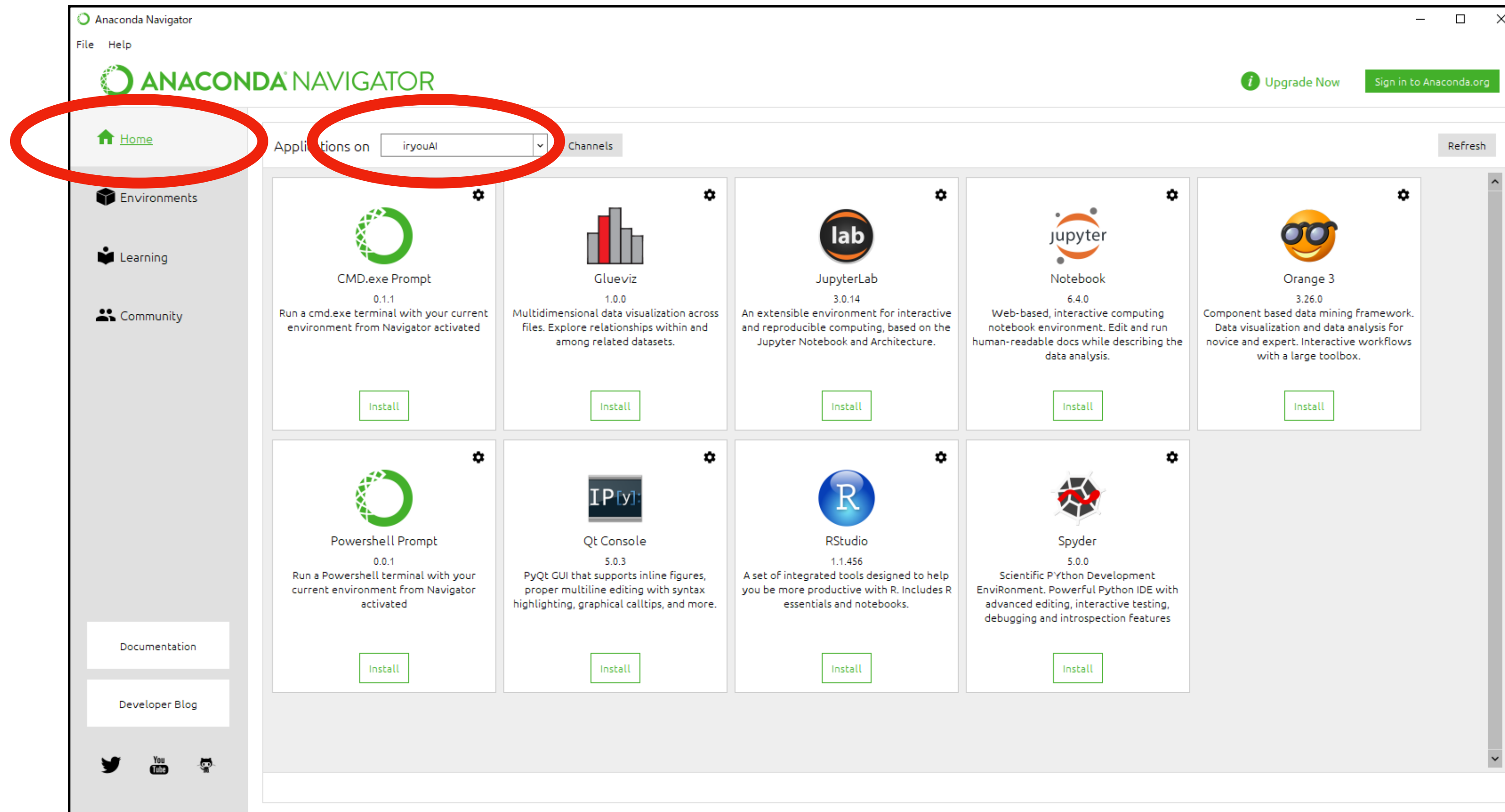
医療とAI・ビッグデータ入門のための仮想環境を作ろう

同様にkerasをインストールする



医療とAI・ビッグデータ入門のための仮想環境を作ろう

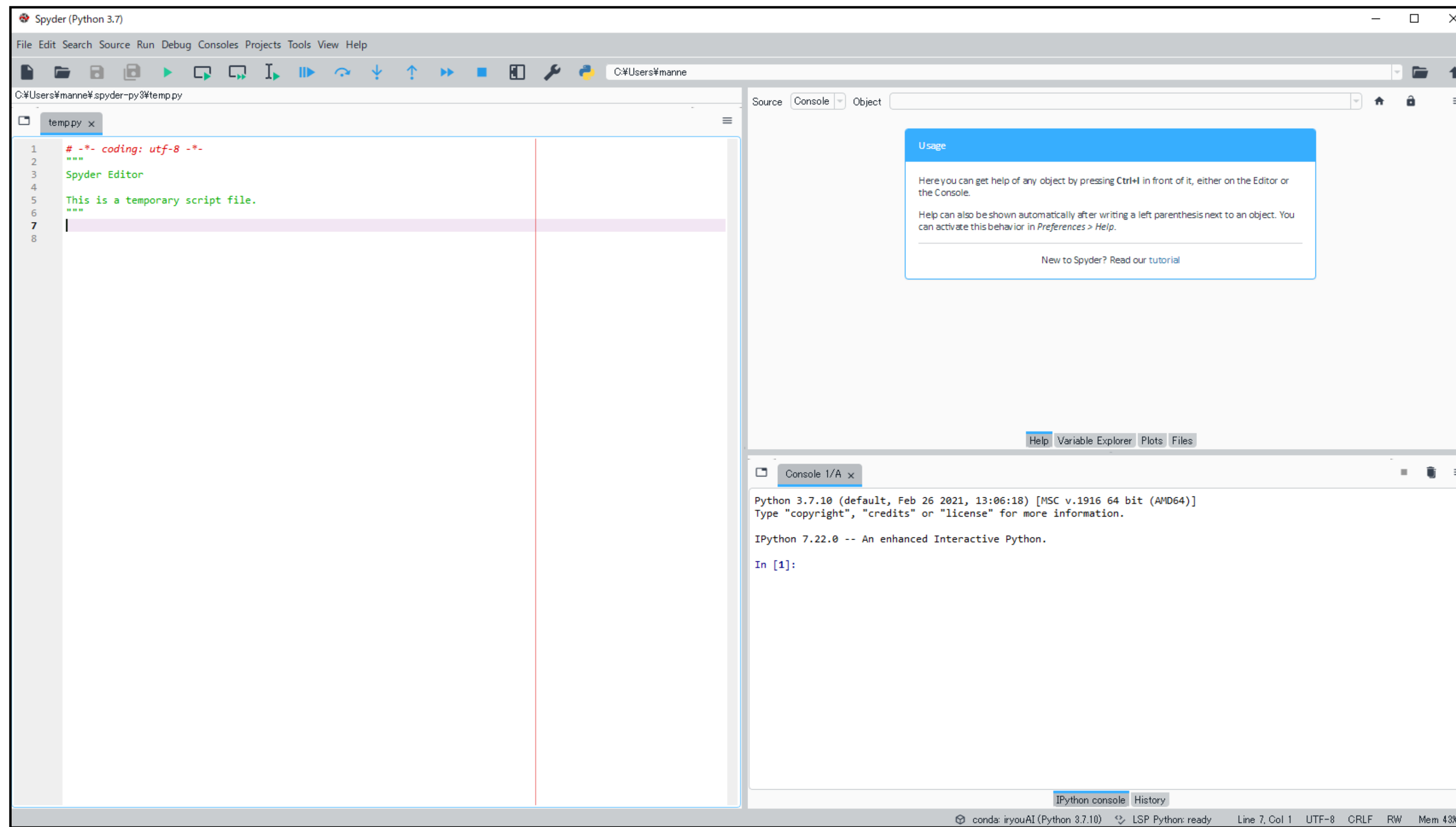
そのままHomeを押すと、Application on “iryouAI”になっていることが確認できる



これでSpyderを起動するとiryouAIの環境で作業することが出来ます。

医療とAI・ビッグデータ入門のための仮想環境を作ろう

Spyderを起動させましょう



課題に取り組んでみよう

(Windowsの人)

#1から10の要素を持つリストをtestという変数に代入しよう

#testを出力しよう

#testの3番目と5番目の要素を出力しよう

#len(test)、max(test)、min(test)を出力しよう

前半終了

インストールするライブラリ

matplotlib

pandas

scikit-learn

keras

後半はnumpy、pandas、matplotlibを使用してみます

(numpyはmatplotlibをインストールすると一緒にインストールされます)

課題に取り組んでみよう

```
test = [1,2,3,4,5,6,7,8,9,10]
print(test)
print(test[2])
print(test[4])
print(len(test))
print(max(test))
print(min(test))
```

```
In [20]: test = [1,2,3,4,5,6,7,8,9,10]
```

```
In [21]: print(test)
```

```
Output from spyder call 'get_namespace_view':
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
In [22]: print(test[2])
```

```
3
```

```
In [23]: print(test[4])
```

```
5
```

```
In [24]: print(len(test))
```

```
10
```

```
In [25]: print(max(test))
```

```
Output from spyder call 'get_namespace_view':
10
```

```
In [26]: print(min(test))
```

```
1
```

```
In [27]:
```


ライブラリについて

ライブラリはimport文でプログラムに取り込む

```
import matplotlib
```

matplotlibをインポートする

```
from matplotlib import pyplot
```

matplotlibの中のpyplotというモジュールのみをインポートする

```
import matplotlib.pyplot as plt
```

matplotlibの中のpyplotというモジュールをpltと省略してインポートする

numpyとpandas

numpyは数値計算に優れたライブラリ
pandasはデータ操作に優れたライブラリ

```
import numpy as np
```

```
import pandas as pd
```

この1文で、“np(pd)という別名をつけてnumpy(pandas)を読み込む”という意味になります。

使うときは“np.処理の名前(クラスと言います)”のように、頭にnp.(or pd.)をつけます
これでそれぞれのライブラリが持つクラスという機能を使用することが出来ます。

numpyのarrayについて

numpyのarrayというクラスを使ってみましょう
リストを用いてarrayを作ります。

変数 = np.array(リスト)

入力

```
import numpy as np

sample = np.array([1,2,3,4,5])
print(sample)
print(type(sample))
```

numpyのarrayについて

numpyのarrayというクラスを使ってみましょう
リストを用いてarrayを作ります。

変数 = np.array(リスト)

入力

```
import numpy as np  
  
sample = np.array([1,2,3,4,5])  
print(sample)  
print(type(sample))
```

出力

```
[1,2,3,4,5]  
numpy.ndarray
```

numpyが作るデータの型をnumpy配列
(ndarray)と言います

numpyのarrayについて

numpy配列に足し算やかけ算を行うと、配列の全ての要素に対して演算が行われます。

```
sample = np.array([1,2,3,4,5])
```

入力

```
t = sample * 2  
print(t)  
tt = sample + 2  
print(tt)
```

numpyのarrayについて

numpy配列に足し算やかけ算を行うと、配列の全ての要素に対して演算が行われます。

```
sample = np.array([1,2,3,4,5])
```

入力

```
t = sample * 2  
print(t)  
tt = sample + 2  
print(tt)
```

出力

```
[2 4 6 8 10]  
  
[3 4 5 6 7]
```

numpyのarrayについて

numpy配列に足し算やかけ算を行うと、配列の全ての要素に対して演算が行われます。

```
sample = np.array([1,2,3,4,5])
```

入力

```
t = sample * 2  
print(t)  
tt = sample + 2  
print(tt)
```

出力

```
[2 4 6 8 10]  
  
[3 4 5 6 7]
```

これはnumpyのarrayの機能で、リストではこのような結果になりません。

```
list = [1,2,3,4,5]  
u = list * 2  
print(u)
```

```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```


numpyの他の機能で色々な配列を作る

変数 = np.arange(数字)

0から引数個分の連続した値を作ります。

入力

```
a = np.arange(10)  
print(a)
```

出力

```
[0 1 2 3 4 5 6 7 8 9]
```

numpyの他の機能で色々な配列を作る

変数 = np.arange(数字)

0から引数個分の連続した値を作ります。

入力

```
a = np.arange(10)
print(a)
```

出力

```
[0 1 2 3 4 5 6 7 8 9]
```

変数 = np.arange(始点、終点、間隔)

始点から終点までを決めた間隔おきに配列を作ります

入力

```
b = np.arange(1, 15, 2)
print(b)
```

出力

```
[ 1  3  5  7  9 11 13]
```

numpyの他の機能で色々な配列を作る

2次元配列は次のようにして作成します。

入力

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

numpyの他の機能で色々な配列を作る

2次元配列は次のようにして作成します。

入力

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

出力

```
[[1 2 3]  
 [4 5 6]]  
(2, 3)
```

この2次元配列の形状は行列と一緒に2 **×** 3となり、np.arrayの配列(変数)に.shapeをつけることで確認することができます。

$$c = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

numpyの他の機能で色々な配列を作る

`reshape`という機能を使って配列を変えることも出来ます。

入力

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

```
d = c.reshape(3,2)  
print(d)  
print(d.shape)
```

```
e = c.reshape(6,1)  
print(e)  
print(e.shape)
```

出力

```
[[1 2 3]  
 [4 5 6]]  
(2, 3)
```

numpyの他の機能で色々な配列を作る

`reshape`という機能を使って配列を変えることも出来ます。

入力

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

```
d = c.reshape(3,2)  
print(d)  
print(d.shape)
```

```
e = c.reshape(6,1)  
print(e)  
print(e.shape)
```

出力

```
[[1 2 3]  
 [4 5 6]]  
(2, 3)
```

```
[[1 2]  
 [3 4]  
 [5 6]]  
(3, 2)
```

```
[[1]  
 [2]  
 [3]  
 [4]  
 [5]  
 [6]]  
(6, 1)
```

`np.array([])`のリストの中にリストを重ねていけば3次元以上の配列も可能です

課題に取り組んでみよう

#1から 8 の要素を小さい順(昇順)に持つ(4,2)のnumpy配列を
test2という変数に代入しよう
#test2を出力しよう
#test2のshapeを確認しよう
#test2のshapeを(2,4)に変えてみよう
#test2のshapeを(2,2,2)に変えてみよう

課題に取り組んでみよう

#1から8の要素を持つ(4,2)のnumpy配列をtest2という変数に代入しよう
#test2を出力しよう
#test2のshapeを確認しよう
#test2のshapeを(2,4)に変えてtest3に代入しよう
#test2のshapeを(2,2,2)に変えてtest4に代入しよう

```
test2 = np.array([[1,2],[3,4],[5,6],[7,8]])  
print(test2)  
print(test2.shape)  
test3 = test2.reshape(2,4)  
print(test3)  
test4 = test2.reshape(2,2,2)  
print(test4)
```

```
print(test2)  
[[1 2]  
 [3 4]  
 [5 6]  
 [7 8]]
```

```
print(test2.shape)  
(4, 2)
```

```
print(test3)  
[[1 2 3 4]  
 [5 6 7 8]]
```

```
print(test4)  
[[[1 2]  
  [3 4]]  
 [[5 6]  
  [7 8]]]
```

(2, 2)が2つあるイメージ

pandas

pandasではデータフレームを使用してみます

```
変数 = pd.DataFrame(データ)
```

データをデータフレームという型（形式）で読み込む

データは自分で作ることも出来ますが、外から読み込むことも出来ます。

行列

データの扱いにおいて行列の概念が重要になります
行が縦、列が横になり、 3×4 の行列は下の様になります。

	1 列目	2 列目	3 列目	4 列目
1行目				
2 行目				
3 行目				

行列

実際の分析に用いるデータは扱いやすい様にするために、

- ・ 個々の値が1つのセル
- ・ 個々の変数が1つの列(プログラミングの変数ではない)
- ・ 個々のサンプルが1つの行

となる様に整形するのが一般的です。

	体重	身長	年齢
Aさん	40	160	20
Bさん	55	170	45
Cさん	62	175	38

このデータであれば3 **×** 3の行列で考えることができます

pandas

pandasではデータフレームを使用してみます

```
変数 = pd.DataFrame(データ)
```

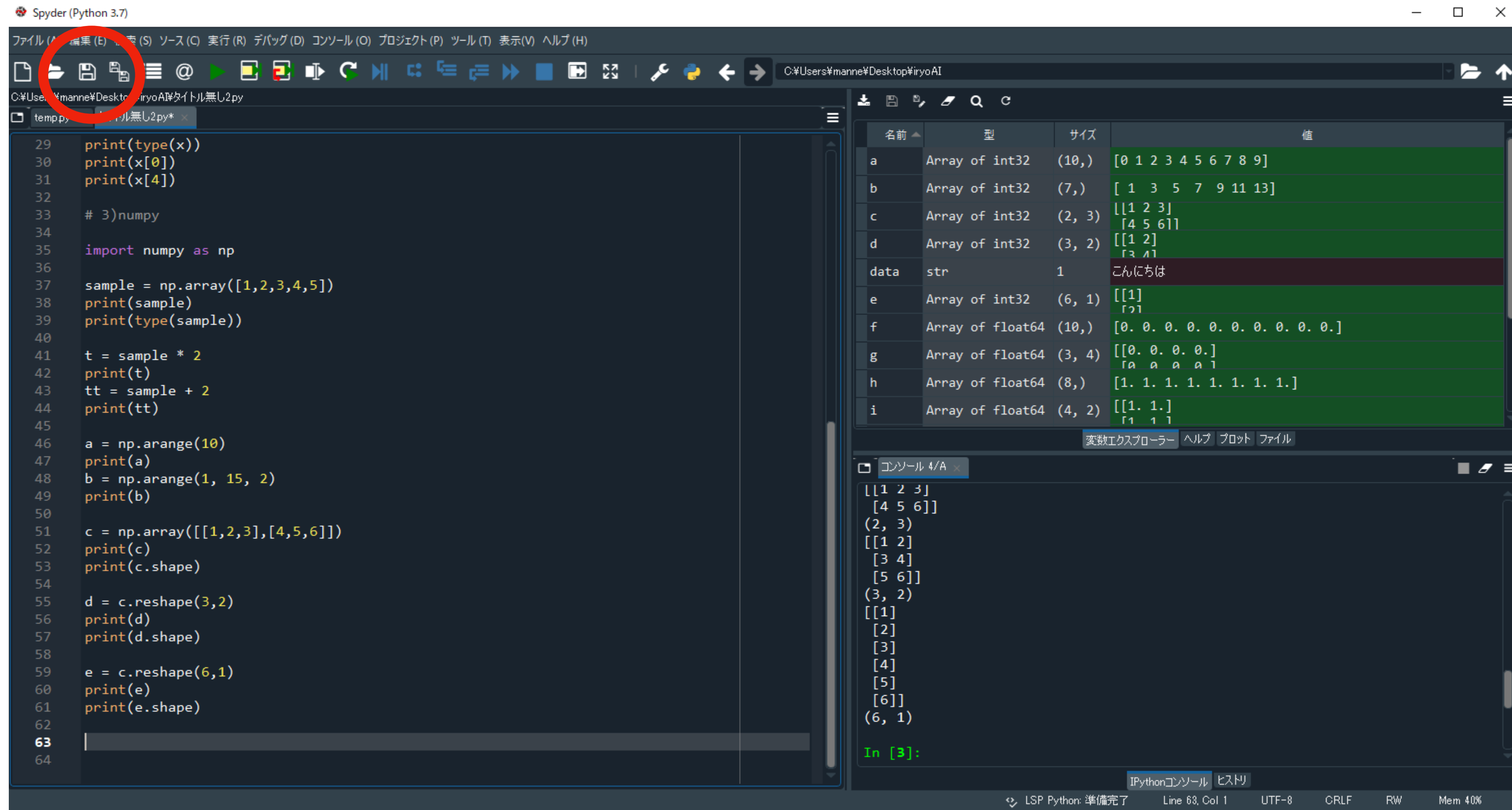
データは自分で作ることも出来ますが、外から読み込むことも出来ます。

	体重	身長	年齢
Aさん	40	160	20
Bさん	55	170	45
Cさん	62	175	38

データフレームはこの形状でデータを扱うことが出来る

外部のデータを読み込んでみましょう

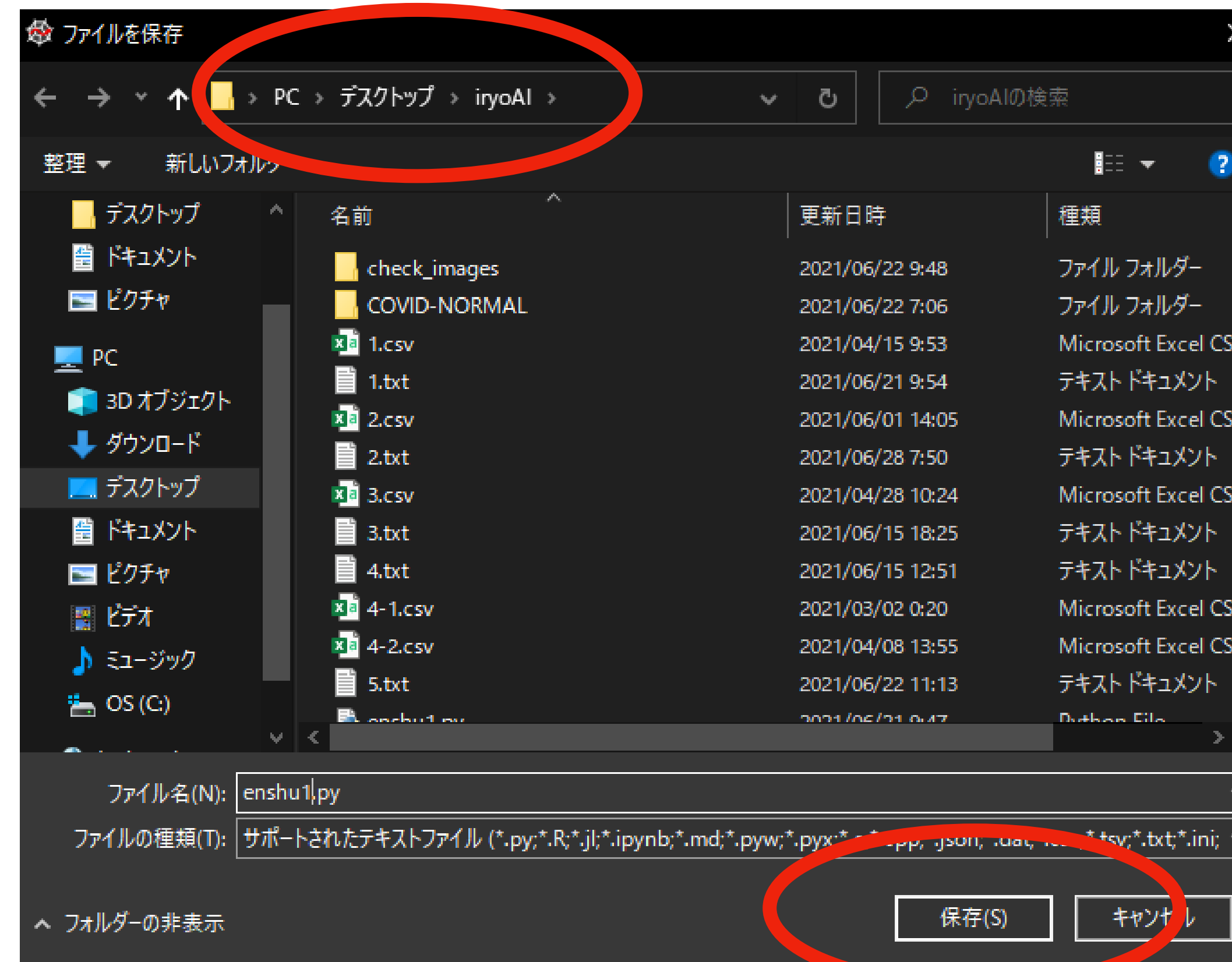
まず今使っているエディタ内のPythonプログラムを保存しましょう



左上の保存アイコンをクリック(or 「ファイル」 → 「形式を指定して保存」)

外部のデータを読み込んでみましょう

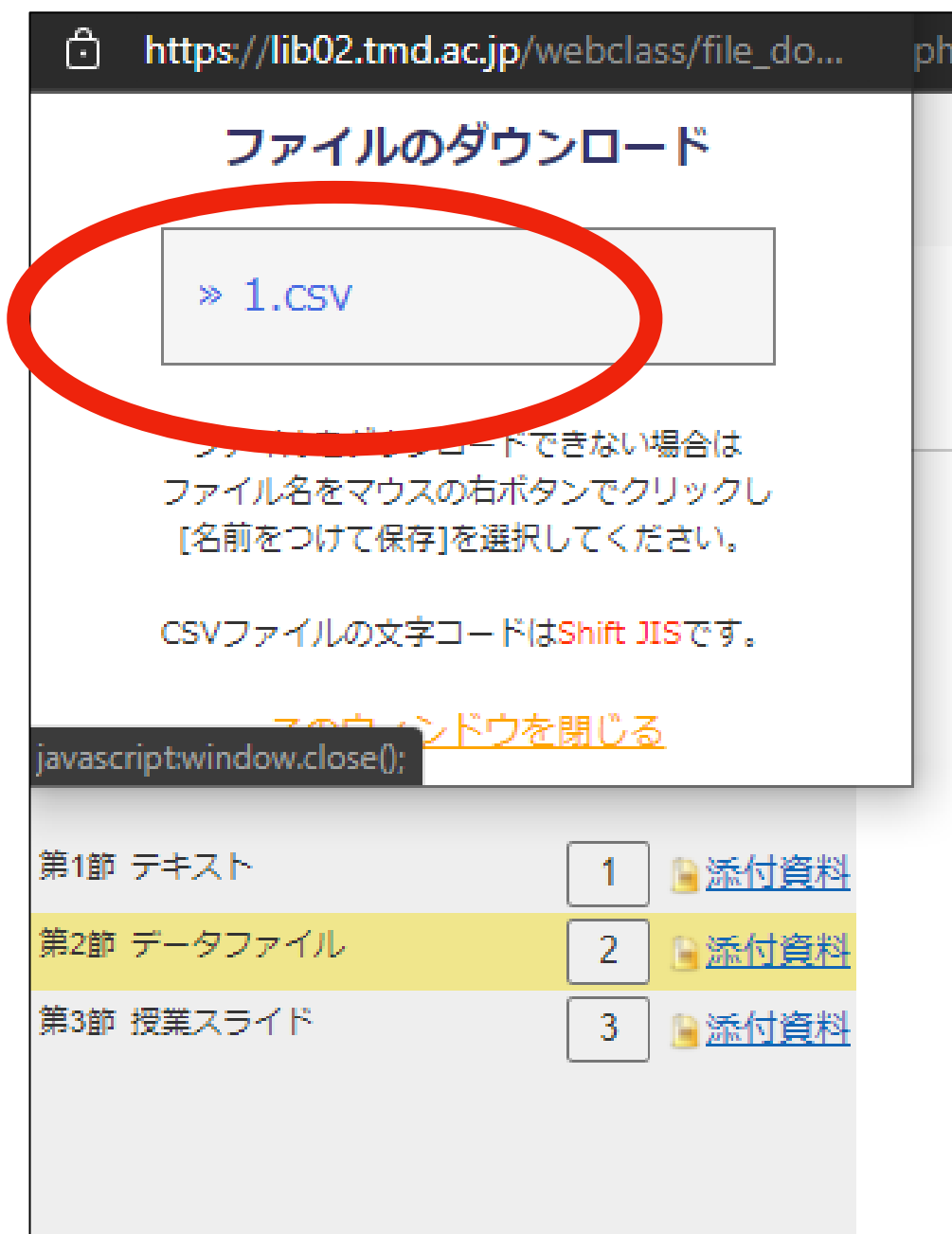
保存場所をデータをダウンロードしている場所(iryōAI)に設定する



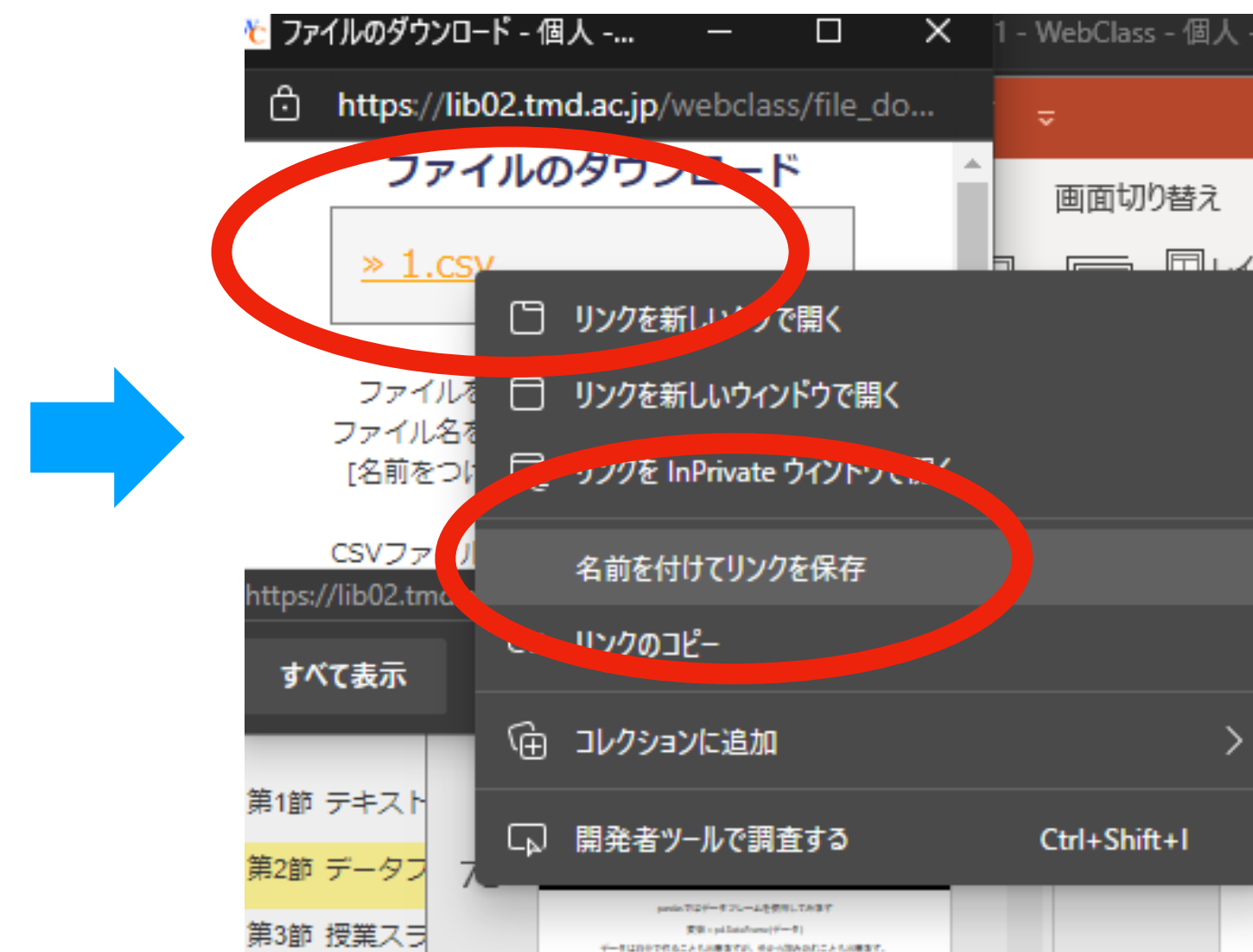
事前にwebclass上で配布している1.csvを同じ場所に移動します
(今回はファイル名は「enshu1.py」としてデスクトップに保存してください)

外部のデータを読み込んでみましょう

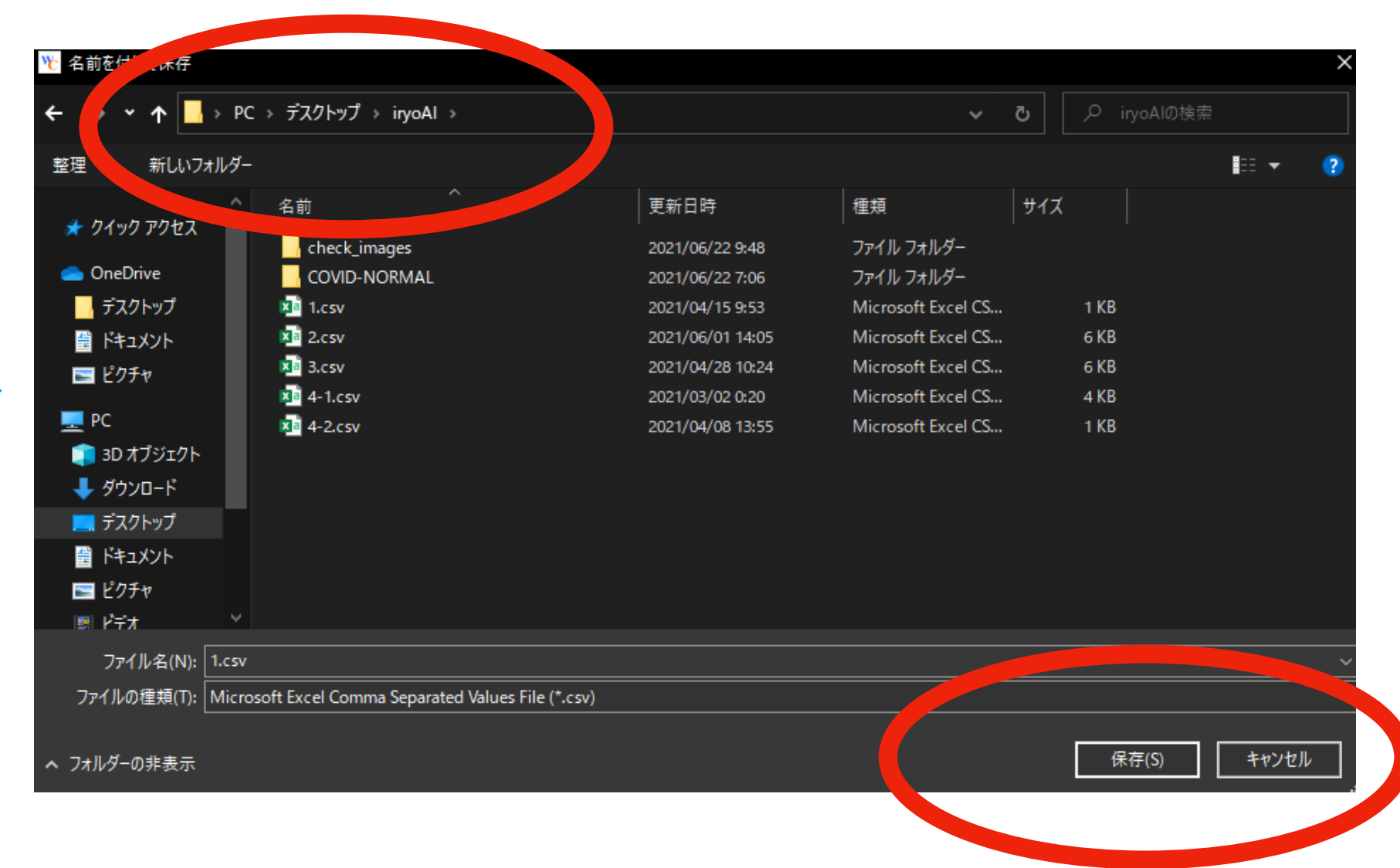
webclass 演習教材(演習1回目)



右クリックで名前を付けて保存



iryoAIを選んで保存



事前にwebclass上で配布している1.csvを同じ場所に移動します
(今回はiryoAI)

外部のデータを読み込んでみましょう

作業する場所(ディレクトリ)をiryoiAIに指定する

The screenshot shows the Spyder Python IDE interface. The main editor on the left contains a Python script using NumPy. The right-hand pane is divided into two sections: the top section displays a table of variables and their values, and the bottom section shows the IPython console output.

Script Content:

```
29 print(type(x))
30 print(x[0])
31 print(x[4])
32
33 # 3)numpy
34
35 import numpy as np
36
37 sample = np.array([1,2,3,4,5])
38 print(sample)
39 print(type(sample))
40
41 t = sample * 2
42 print(t)
43 tt = sample + 2
44 print(tt)
45
46 a = np.arange(10)
47 print(a)
48 b = np.arange(1, 15, 2)
49 print(b)
50
51 c = np.array([[1,2,3],[4,5,6]])
52 print(c)
53 print(c.shape)
54
55 d = c.reshape(3,2)
56 print(d)
57 print(d.shape)
58
59 e = c.reshape(6,1)
60 print(e)
61 print(e.shape)
62
63
64
```

Variable Explorer Table:

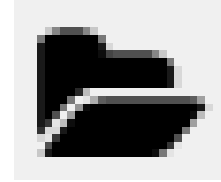
名前	型	サイズ	値
a	Array of int32	(10,)	[0 1 2 3 4 5 6 7 8 9]
b	Array of int32	(7,)	[1 3 5 7 9 11 13]
c	Array of int32	(2, 3)	[[1 2 3] [4 5 6]]
d	Array of int32	(3, 2)	[[1 2] [3 4]]
data	str	1	こんにちは
e	Array of int32	(6, 1)	[[1] [2] [3] [4] [5] [6]]
f	Array of float64	(10,)	[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
g	Array of float64	(3, 4)	[[0. 0. 0. 0.] [0. 0. 0. 0.] [0. 0. 0. 0.]]
h	Array of float64	(8,)	[1. 1. 1. 1. 1. 1. 1. 1.]
i	Array of float64	(4, 2)	[[1. 1.] [1. 1.] [1. 1.] [1. 1.]]

IPython Console Output:

```
[[1 2 3]
 [4 5 6]]
(2, 3)
[[1 2]
 [3 4]
 [5 6]]
(3, 2)
[[1]
 [2]
 [3]
 [4]
 [5]
 [6]]
(6, 1)
In [3]:
```

現在のディレクトリ

外部のデータを読み込んでみましょう



アイコンをクリック

Spyder (Python 3.7)

ファイル (A) 編集 (E) 検索 (S) ソース (C) 実行 (R) デバッグ (D) コンソール (O) プロジェクト (P) ツール (T) 表示 (V) ヘルプ (H)

C:\Users\manne\Desktop\iryoAI\enshu1.py

```
29 print(type(x))
30 print(x[0])
31 print(x[4])
32
33 # 3)numpy
34
35 import numpy as np
36
37 sample = np.array([1,2,3,4,5])
38 print(sample)
39 print(type(sample))
40
41 t = sample * 2
42 print(t)
43 tt = sample + 2
44 print(tt)
45
46 a = np.arange(10)
47 print(a)
48 b = np.arange(1, 15, 2)
49 print(b)
50
51 c = np.array([[1,2,3],[4,5,6]])
52 print(c)
53 print(c.shape)
54
55 d = c.reshape(3,2)
56 print(d)
57 print(d.shape)
58
59 e = c.reshape(6,1)
60 print(e)
61 print(e.shape)
62
63
64
```

名前	型	サイズ	値
a	Array of int32	(10,)	[0 1 2 3 4 5 6 7 8 9]
b	Array of int32	(7,)	[1 3 5 7 9 11 13]
c	Array of int32	(2, 3)	[[1 2 3] [4 5 6]]
d	Array of int32	(3, 2)	[[1 2] [3 4]]
data	str	1	こんにちは
e	Array of int32	(6, 1)	[[1] [2]]
f	Array of float64	(10,)	[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
g	Array of float64	(3, 4)	[[0. 0. 0. 0.] [a a a a]]
h	Array of float64	(8,)	[1. 1. 1. 1. 1. 1. 1. 1.]
i	Array of float64	(4, 2)	[[1. 1.] [1 1 1]]

変数エクスプローラー ヘルプ プロット ファイル

コンソール 4/A

```
[[1 2 3]
 [4 5 6]]
(2, 3)
[[1 2]
 [3 4]
 [5 6]]
(3, 2)
[[1]
 [2]
 [3]
 [4]
 [5]
 [6]]
(6, 1)

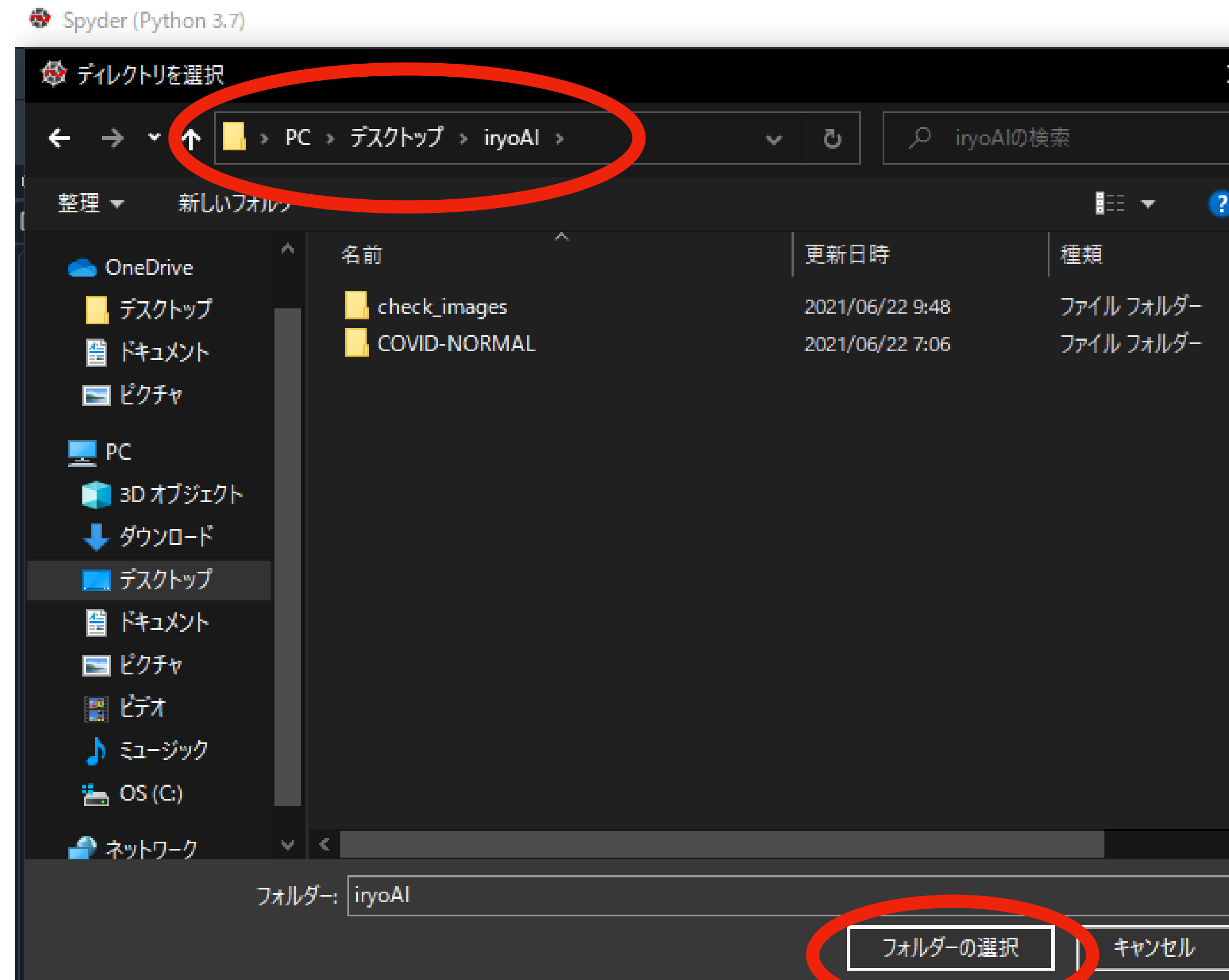
In [3]:
```

IPythonコンソール ヒストリ

LSP Python: 準備完了 Line 63, Col 1 UTF-8 CRLF RW Mem 40%

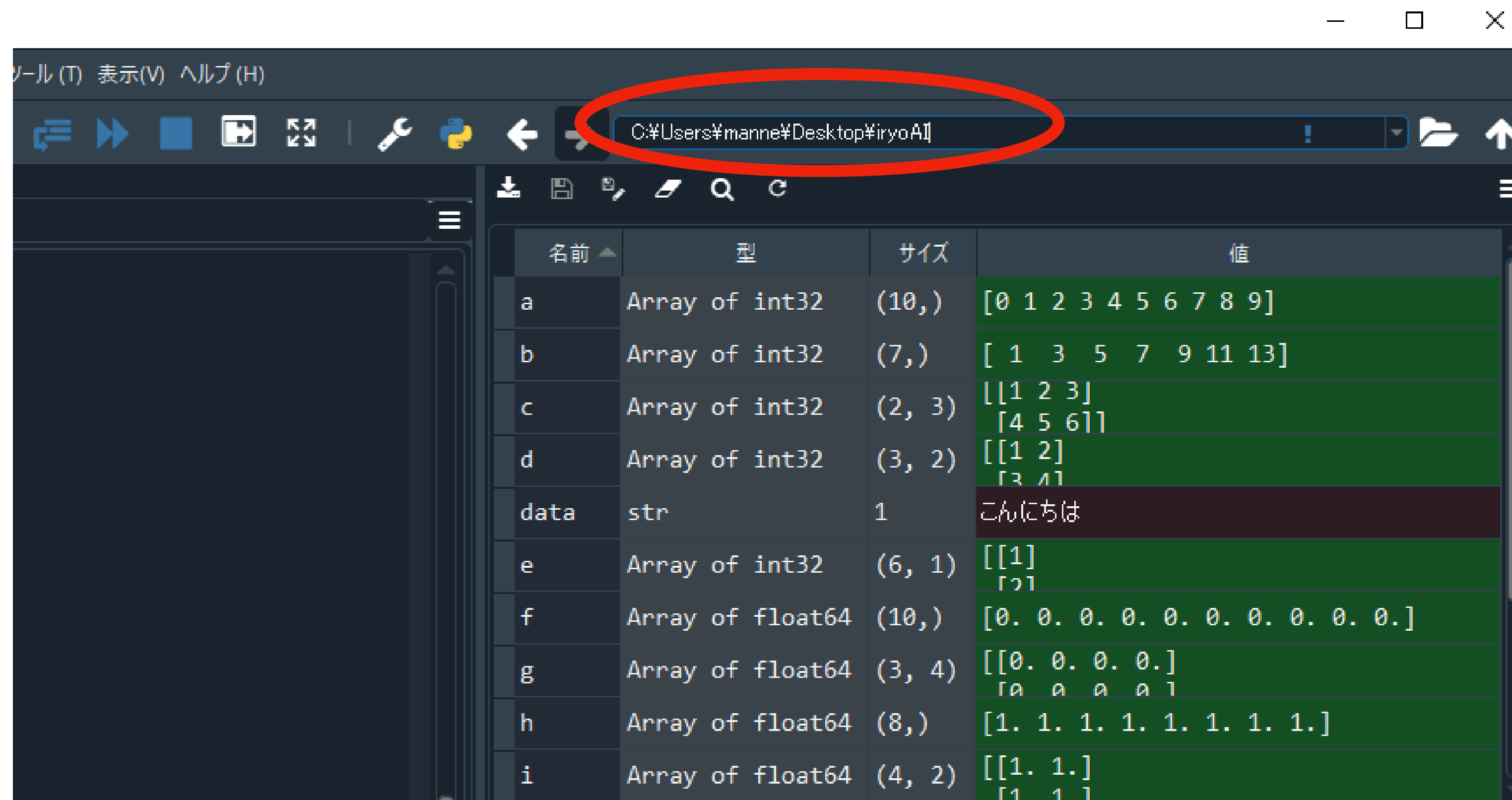
外部のデータを読み込んでみましょう

iryoAIを選んで、フォルダの選択をクリック



外部のデータを読み込んでみましょう

C:\¥### ¥iryoAI に変更されていることを確認



The screenshot shows a Jupyter Notebook window. The file path in the top bar is `C:\Users\manne\Desktop\iryoAI`, which is circled in red. The main area displays a table with 10 rows of data.

名前	型	サイズ	値
a	Array of int32	(10,)	[0 1 2 3 4 5 6 7 8 9]
b	Array of int32	(7,)	[1 3 5 7 9 11 13]
c	Array of int32	(2, 3)	[[1 2 3] [4 5 6]]
d	Array of int32	(3, 2)	[[1 2] [3 4]]
data	str	1	こんにちは
e	Array of int32	(6, 1)	[[1] [2]]
f	Array of float64	(10,)	[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
g	Array of float64	(3, 4)	[[0. 0. 0. 0.] [0 0 0 0]]
h	Array of float64	(8,)	[1. 1. 1. 1. 1. 1. 1. 1.]
i	Array of float64	(4, 2)	[[1. 1.] [1 1 1]]

外部のデータを読み込んでみましょう

事前に準備しているcsvファイル(“1.csv”)を読み込んでみます。

```
df = pd.read_csv(“ファイル名”)
```

入力

```
import pandas as pd  
df = pd.read_csv("1.csv")  
print(df)
```

外部のデータを読み込んでみましょう

事前に準備しているcsvファイル("1.csv")を読み込んでみます。

```
df = pd.read_csv("ファイル名")
```

入力

```
import pandas as pd  
df = pd.read_csv("1.csv")  
print(df)
```

出力

	体重	身長	年齢
0	40	160	20
1	55	170	45
2	62	175	38
3	66	166	40
4	55	162	52
5	80	181	30

pd.read_csv()で代入した変数(この場合df)はデータフレームになっています。
このデータフレームは行、列に対して様々な操作が出来ます。

列に対するデータフレームの操作

列名を指定して取得する場合は[]を用います。

データフレーム['列名']

複数の列を取得したい場合は、[]をもう1つ入れます

入力

```
print(df['体重'])
```

```
print(df[['体重','年齢']])
```

列に対するデータフレームの操作

列名を指定して取得する場合は[]を用います。

データフレーム['列名']

複数の列を取得したい場合は、[]をもう1つ入れます

入力

```
print(df['体重'])
```

```
print(df[['体重','年齢']])
```

出力

```
0    40
1    55
2    62
3    66
4    55
5    80
Name: 体重, dtype: int64
```

```
   体重  年齢
0  40  20
1  55  45
2  62  38
3  66  40
4  55  52
5  80  30
```


行に対するデータフレームの操作

データフレーム[開始:終了]で、データセットの開始行(0行から)から終了行(-1)までを抜き出せます

抜き出したデータフレームを別の変数に代入することも出来ます

入力

```
print(df[0:2])
```

```
a = df[1:4]  
print(a)
```

行に対するデータフレームの操作

データフレーム[開始:終了]で、データセットの開始行(0行から)から終了行(-1)までを抜き出せます

抜き出したデータフレームを別の変数に代入することも出来ます

入力 0から(2-1)行目まで
`print(df[0:2])`

1から(4-1)行目まで
`a = df[1:4]`
`print(a)`

行に対するデータフレームの操作

データフレーム[開始:終了]で、データセットの開始行(0行から)から終了行(-1)までを抜き出せます

抜き出したデータフレームを別の変数に代入することも出来ます

入力 0から(2-1)行目まで
`print(df[0:2])`

出力

	体重	身長	年齢
0	40	160	20
1	55	170	45

1から(4-1)行目まで
`a = df[1:4]`
`print(a)`

	体重	身長	年齢
1	55	170	45
2	62	175	38
3	66	166	40

matplotlib

matplotlibは図を書く機能を持ったライブラリです

```
import matplotlib.pyplot as plt
```

matplotlibの中のpyplotというモジュールをpltと省略してインポートします

被験者	年齢	歯周病の歯の本数
1	35	3
2	21	0
3	45	6
4	58	8
5	77	13

このようなデータを使って作図をしてみます

#5) をコピーして一度に実行する

```
# matplotlib
# 5) 散布図の作成

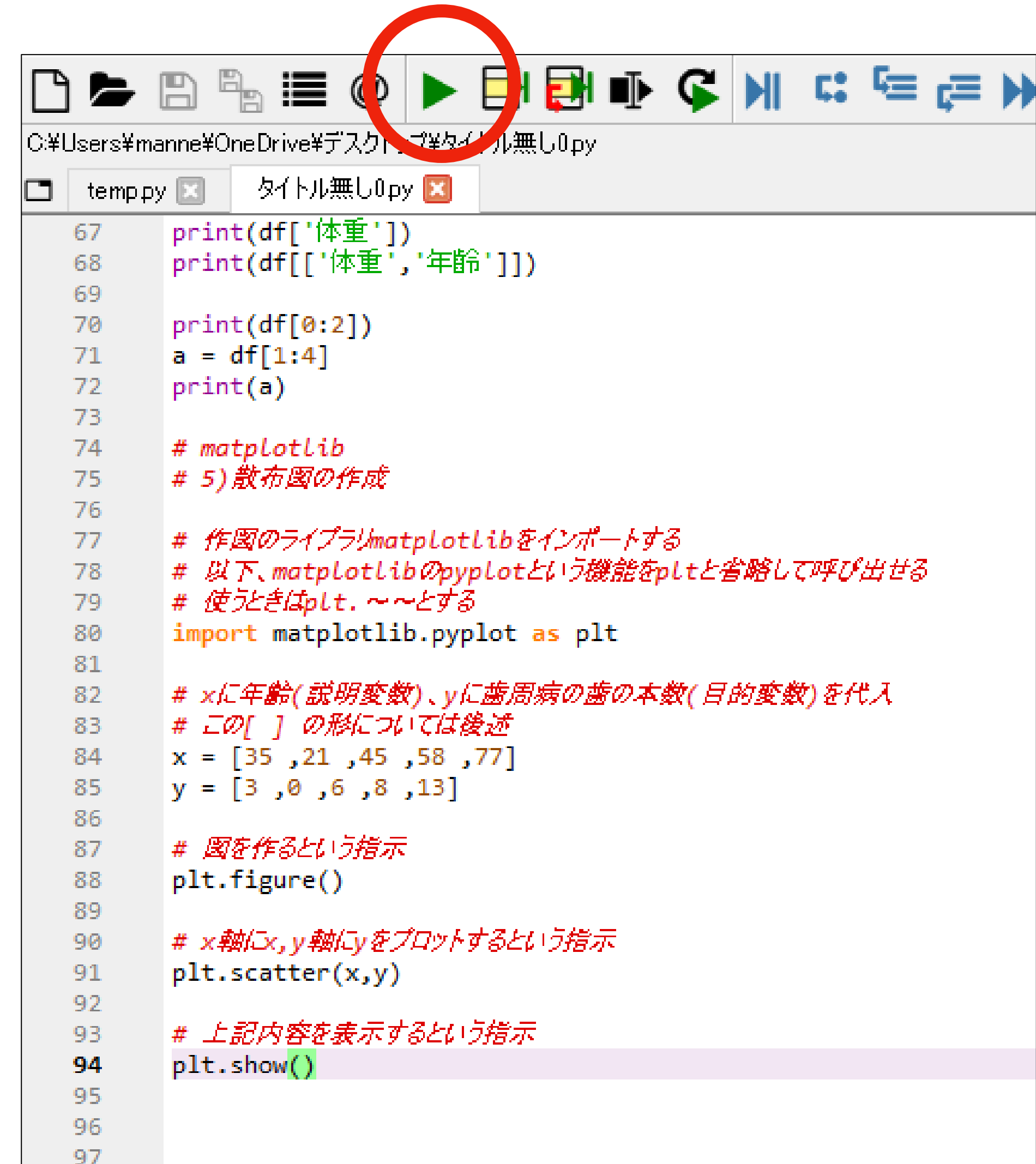
# 作図のライブラリmatplotlibをインポートする
# 以下、matplotlibのpyplotという機能をpltと省略して呼び出せる
# 使うときはplt.~~とする
import matplotlib.pyplot as plt

# xに年齢、yに歯周病の歯の本数を代入
x = [35, 21, 45, 58, 77]
y = [3, 0, 6, 8, 13]

# 図を作るという指示
plt.figure()

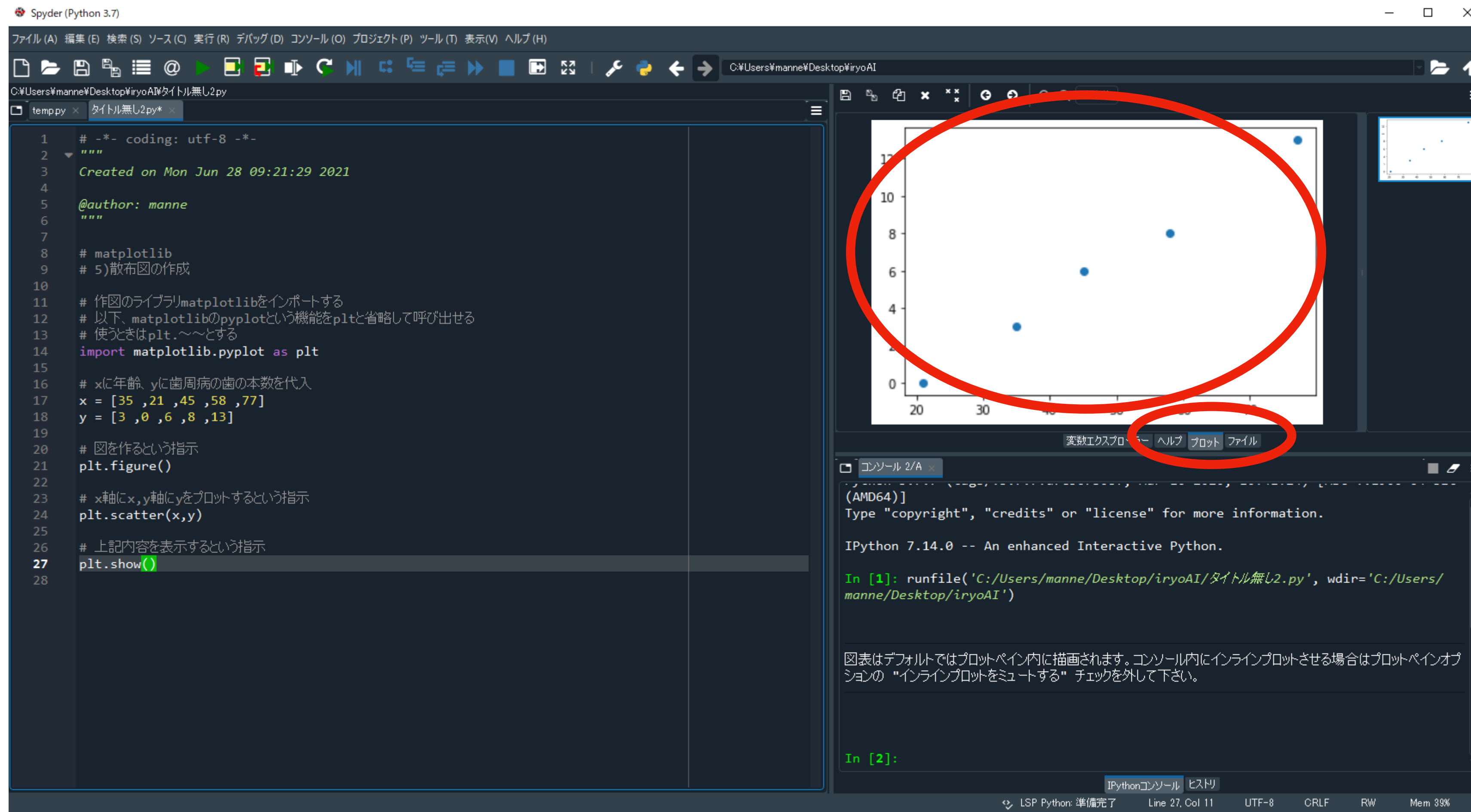
# x軸にx, y軸にyをプロットするという指示
plt.scatter(x, y)

# 上記内容を表示するという指示
plt.show()
```



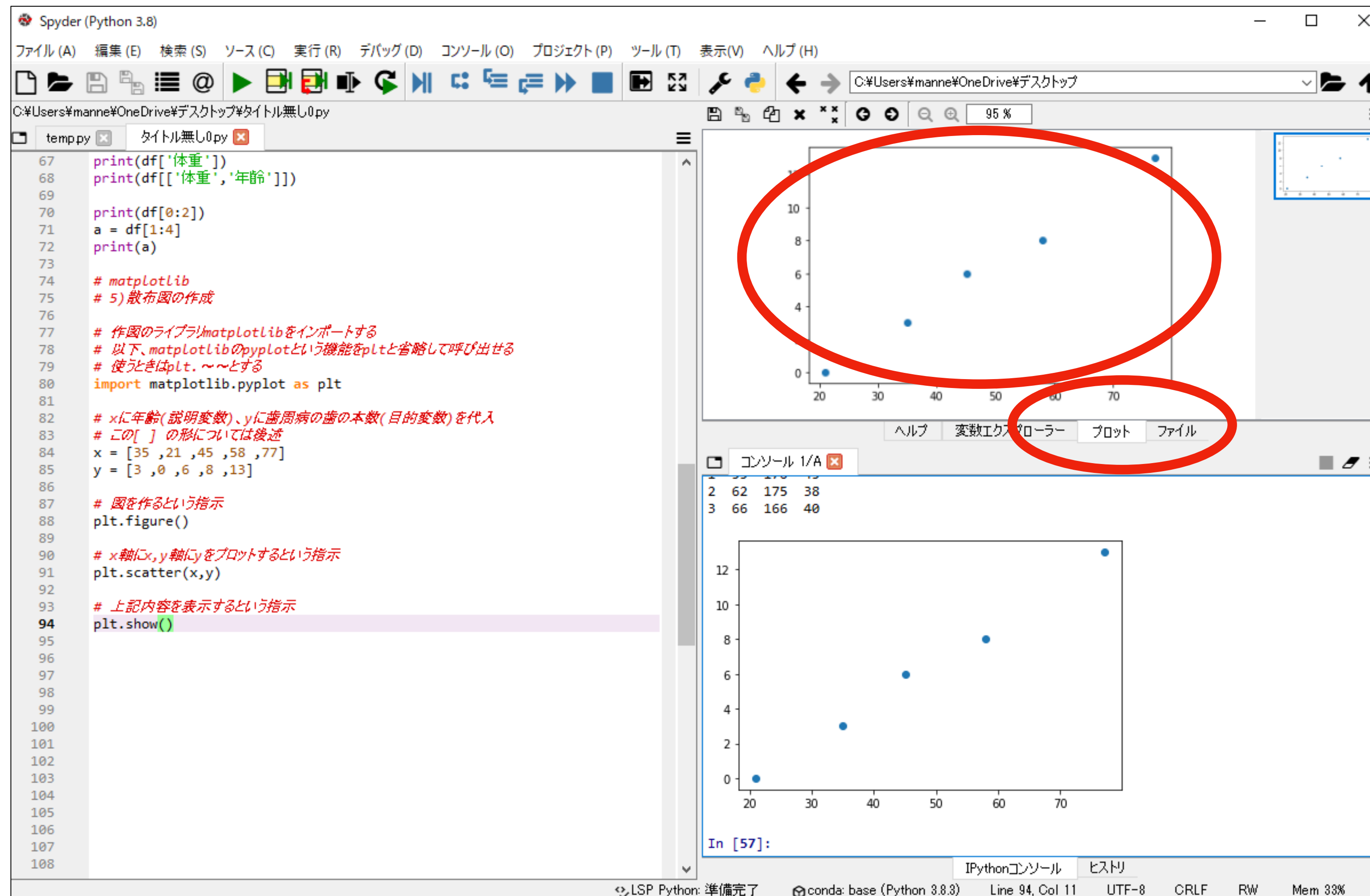
```
67 print(df['体重'])
68 print(df[['体重', '年齢']])
69
70 print(df[0:2])
71 a = df[1:4]
72 print(a)
73
74 # matplotlib
75 # 5) 散布図の作成
76
77 # 作図のライブラリmatplotlibをインポートする
78 # 以下、matplotlibのpyplotという機能をpltと省略して呼び出せる
79 # 使うときはplt.~~とする
80 import matplotlib.pyplot as plt
81
82 # xに年齢(説明変数)、yに歯周病の歯の本数(目的変数)を代入
83 # この[ ] の形については後述
84 x = [35, 21, 45, 58, 77]
85 y = [3, 0, 6, 8, 13]
86
87 # 図を作るという指示
88 plt.figure()
89
90 # x軸にx, y軸にyをプロットするという指示
91 plt.scatter(x, y)
92
93 # 上記内容を表示するという指示
94 plt.show()
95
96
97
```

実行してみる



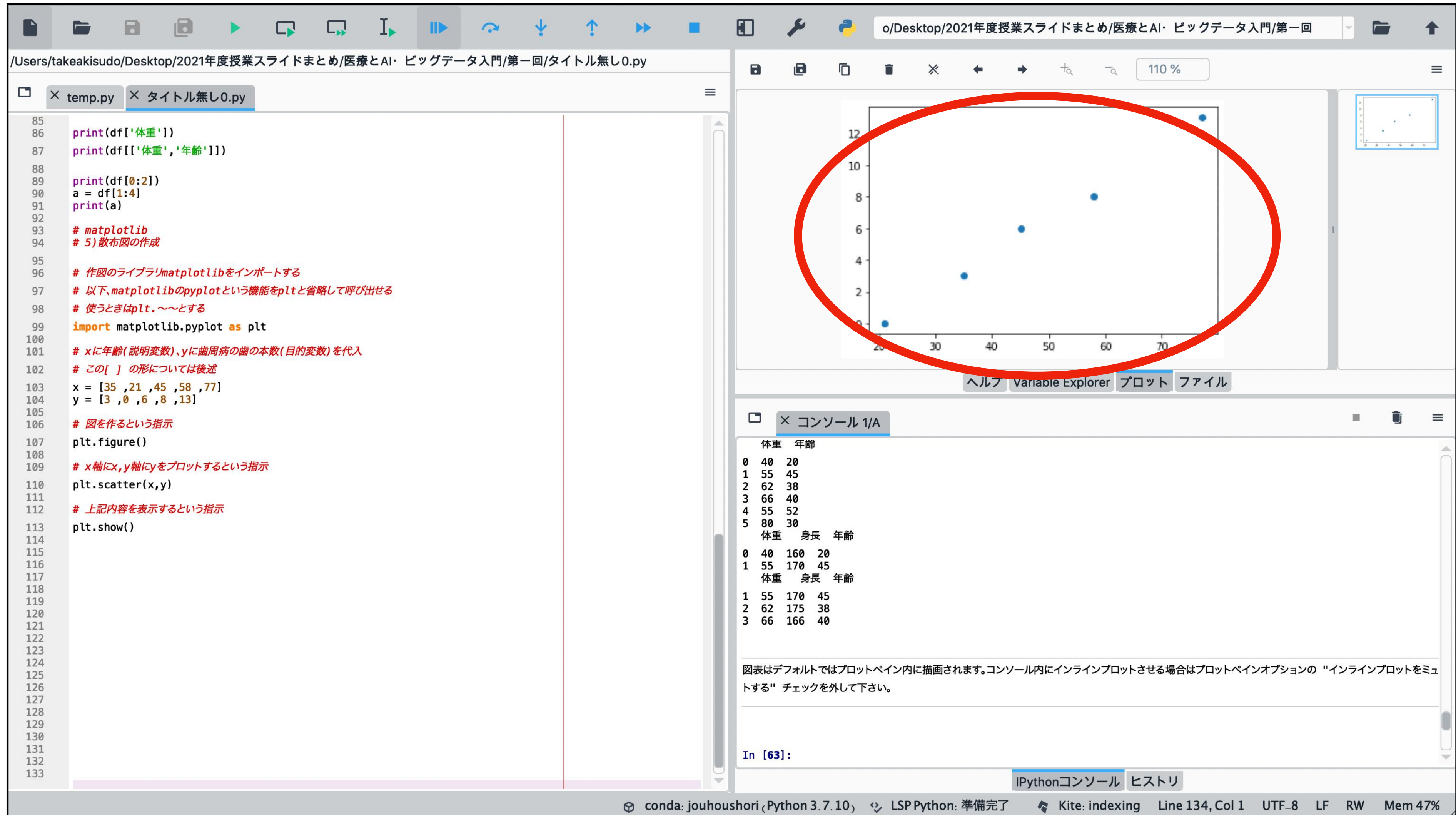
右上のプロットをクリックすると図が表示されます。
(コンソールにも図が表示されてますが、基本プロットを見るようにします)

実行してみる



右上のプロットをクリックすると図が表示されます。
(コンソールにも図が表示されてますが、基本プロットを見るようにします)

実行してみる



これはバージョン5ですが、コンソールに表示されていなくても
プロットに表示されていれば大丈夫です。

点をプロットしてみる(散布図)

プログラムの中身

1) 散布図の作成

作図のライブラリmatplotlibをインポートする

以下、matplotlibのpyplotという機能をpltと省略して呼び出せる

使うときはplt.~~とする

```
import matplotlib.pyplot as plt
```

xに年齢、yに歯周病の歯の本数を代入

```
x = [35, 21, 45, 58, 77]
```

```
y = [3, 0, 6, 8, 13]
```

図を作るという指示(図の枠を作成)

```
plt.figure()
```

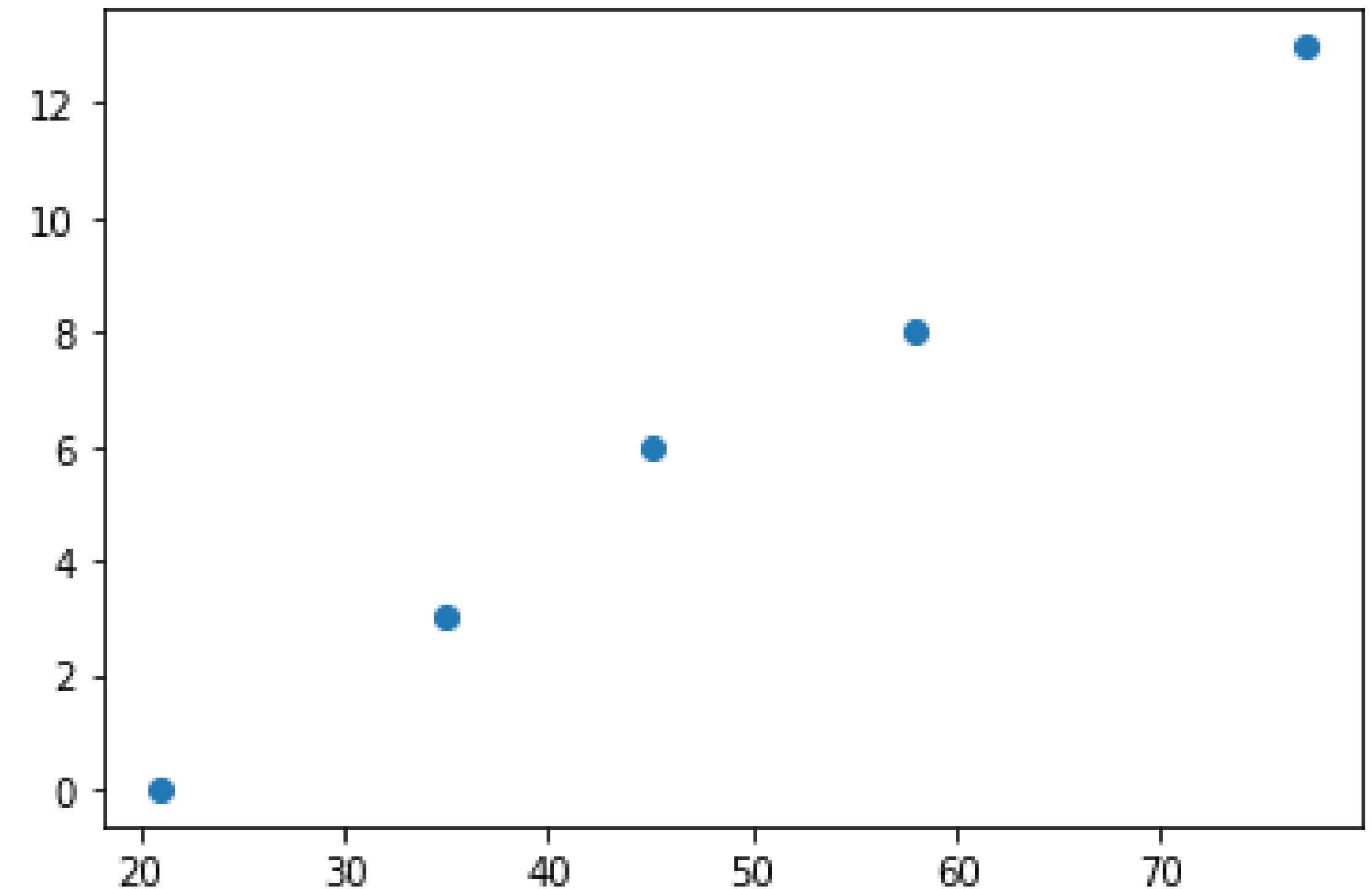
x軸にx, y軸にyをプロットするという指示

```
plt.scatter(x, y)
```

上記内容を表示するという指示

```
plt.show()
```

作図の結果



実際に指示しているプログラムは6行
(他はコメント)

点をプロットしてみる(散布図)

プログラムの中身

1) 散布図の作成

作図のライブラリmatplotlibをインポートする

以下、matplotlibのpyplotという機能をpltと省略して呼び出せる

使うときはplt.~~とする

import matplotlib.pyplot as plt

xに年齢、yに歯周病の歯の本数を代入

x = [35,21,45,58,77]

y = [3,0,6,8,13]

点をプロットしてみる(散布図)

プログラムの中身

1) 散布図の作成

作図のライブラリmatplotlibをインポートする

以下、matplotlibのpyplotという機能をpltと省略して呼び出せる

使うときはplt.~~とする

`import matplotlib.pyplot as plt`

xに年齢、yに歯周病の歯の本数を代入

`x = [35, 21, 45, 58, 77]`

`y = [3, 0, 6, 8, 13]`

被験者	年齢	歯周病の歯の本数
1	35	3
2	21	0
3	45	6
4	58	8
5	77	13

リストとして各変数の値をxとyに代入している

x[0]は35、y[0]は3になる

点をプロットしてみる(散布図)

matplotlib(.pyplot)の機能

作図の結果

図を作るという指示(図の枠を作成)

```
plt.figure()
```

x軸にx,y軸にyをプロットするという指示

```
plt.scatter(x,y)
```

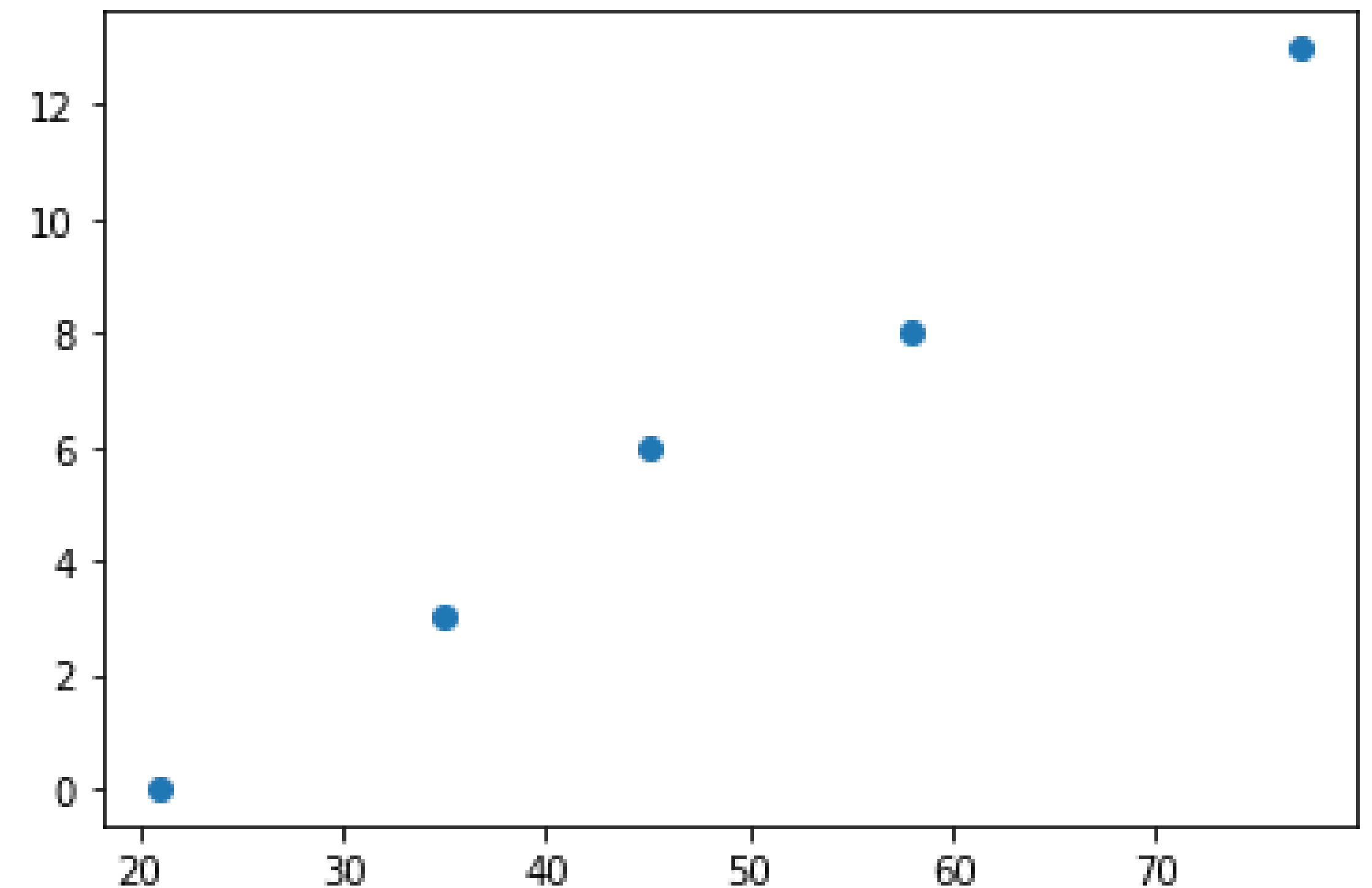
上記内容を表示するという指示

```
plt.show()
```

仮に、x=ではなく、

x2 = [35,21,45,58,77]としていたら

plt.scatter(x2, y)とすれば良い



点をプロットしてみる(散布図)

6) をコピーしてそのまま次に貼り付けてみよう

(上のコマンドは消してもそのままでもいいです)

```
*演習1のコピー.txt - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
plt.figure()

# x軸にx,y軸にyをプロットするという指示
plt.scatter(x,y)

# 上記内容を表示するという指示
plt.show()

# 6) 散布図の作成 (X軸とY軸の名前、タイトル、グリッド線の追加)

import matplotlib.pyplot as plt
x = [35, 21, 45, 58, 77]
y = [3, 0, 6, 8, 13]
plt.figure()

# 図のタイトル
plt.title('age and number of teeth affected by periodontitis')

# x軸の名前(ラベル)
plt.xlabel('age')

# y軸の名前(ラベル)
plt.ylabel('number of teeth affected by periodontitis')

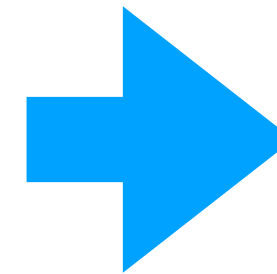
# グリッド線の追加
plt.grid(True)

plt.scatter(x,y)
plt.show()

# 7) 散布図の作成 (日本語を反映させるライブラリをインストール)

import matplotlib.pyplot as plt

# rcParamsという関数をインポートして日本語を反映させる
from matplotlib import rcParams
rcParams['font.family'] = 'sans-serif'
rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro', 'Yu G
```

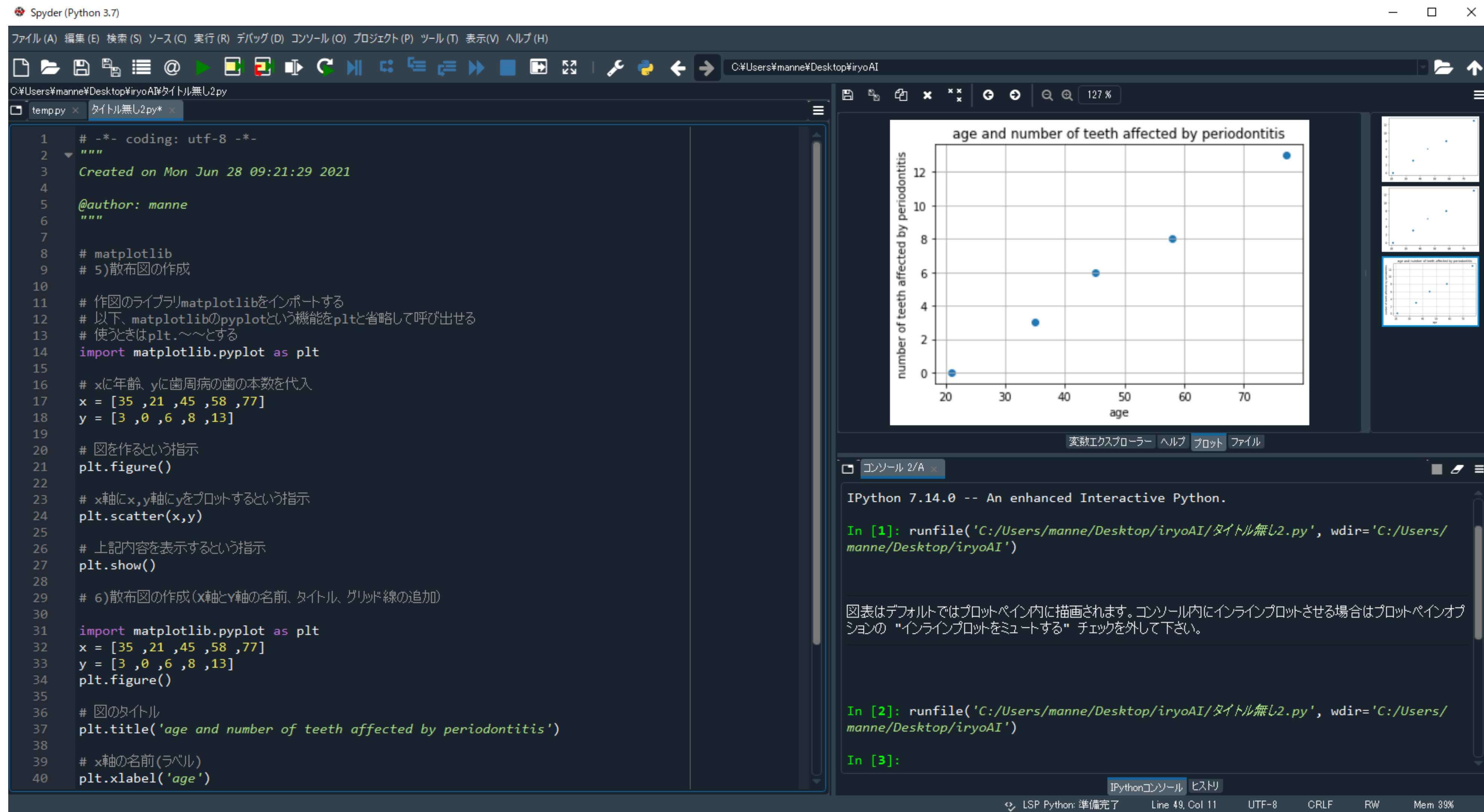


```
temp.py x タイトル無し0.py
C:\Users\manne\OneDrive\デスクトップ\タイトル無し0.py

82 # xに年齢、yに歯周病の歯の本数を代入
83 x = [35, 21, 45, 58, 77]
84 y = [3, 0, 6, 8, 13]
85
86 # 図を作るという指示
87 plt.figure()
88
89 # x軸にx,y軸にyをプロットするという指示
90 plt.scatter(x,y)
91
92 # 上記内容を表示するという指示
93 plt.show()
94
95 # 6) 散布図の作成 (X軸とY軸の名前、タイトル、グリッド線の追加)
96
97 import matplotlib.pyplot as plt
98 x = [35, 21, 45, 58, 77]
99 y = [3, 0, 6, 8, 13]
100 plt.figure()
101
102 # 図のタイトル
103 plt.title('age and number of teeth affected by periodontitis')
104
105 # x軸の名前(ラベル)
106 plt.xlabel('age')
107
108 # y軸の名前(ラベル)
109 plt.ylabel('number of teeth affected by periodontitis')
110
111 # グリッド線の追加
112 plt.grid(True)
113
114 plt.scatter(x,y)
115 plt.show()
116
117
118
119
120
```

点をプロットしてみる(散布図)

実行すると少し違った図が出てきます



点をプロットしてみる(散布図)

4行追加

3) 散布図の作成 (X軸とY軸の名前、タイトル、グリッド線の追加)

```
import matplotlib.pyplot as plt
x = [35, 21, 45, 58, 77]
y = [3, 0, 6, 8, 13]
plt.figure()
```

図のタイトル

```
plt.title('age and number of teeth affected by periodontitis')
```

x軸の名前(ラベル)

```
plt.xlabel('age')
```

y軸の名前(ラベル)

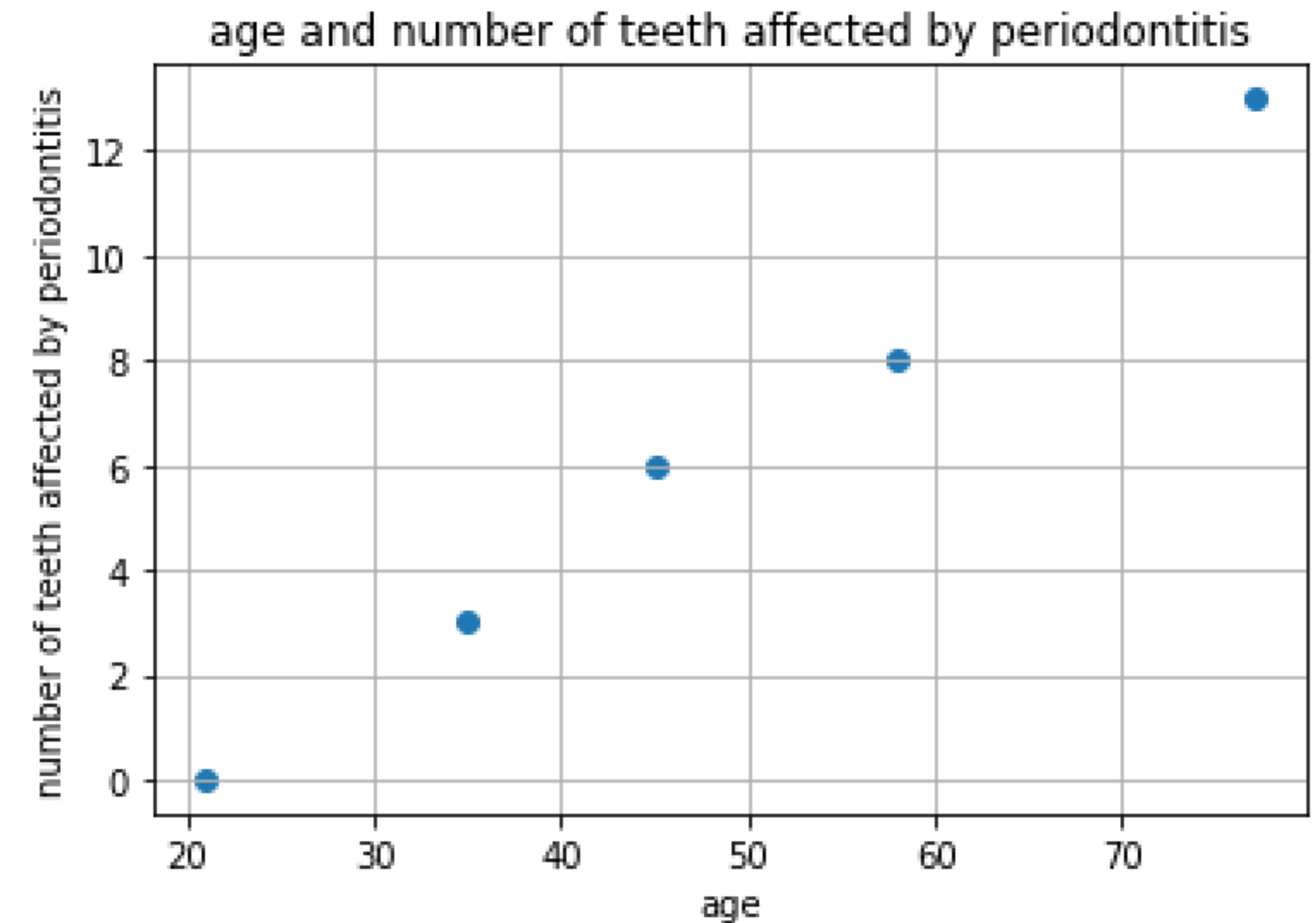
```
plt.ylabel('number of teeth affected by periodontitis')
```

グリッド線の追加

```
plt.grid(True)
```

```
plt.scatter(x, y)
```

```
plt.show()
```



```
plt.title("タイトル名")
```

```
plt.xlabel("x軸の名前")
```

```
plt.ylabel("y軸の名前")
```

```
plt.grid(True)
```


#7) をコピーして実行する

日本語が表示されるようにする

4) 散布図の作成（日本語を反映させるライブラリをインストール）

```
import matplotlib.pyplot as plt
```

rcParamsという関数をインポートして日本語を反映させる

```
from matplotlib import rcParams
```

```
rcParams['font.family']='sans-serif'
```

```
rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro', 'Yu Gothic', 'Meirio']
```

```
x = [35,21,45,58,77]
```

```
y = [3,0,6,8,13]
```

```
plt.figure()
```

```
plt.title('年齢と歯周病の歯の本数')
```

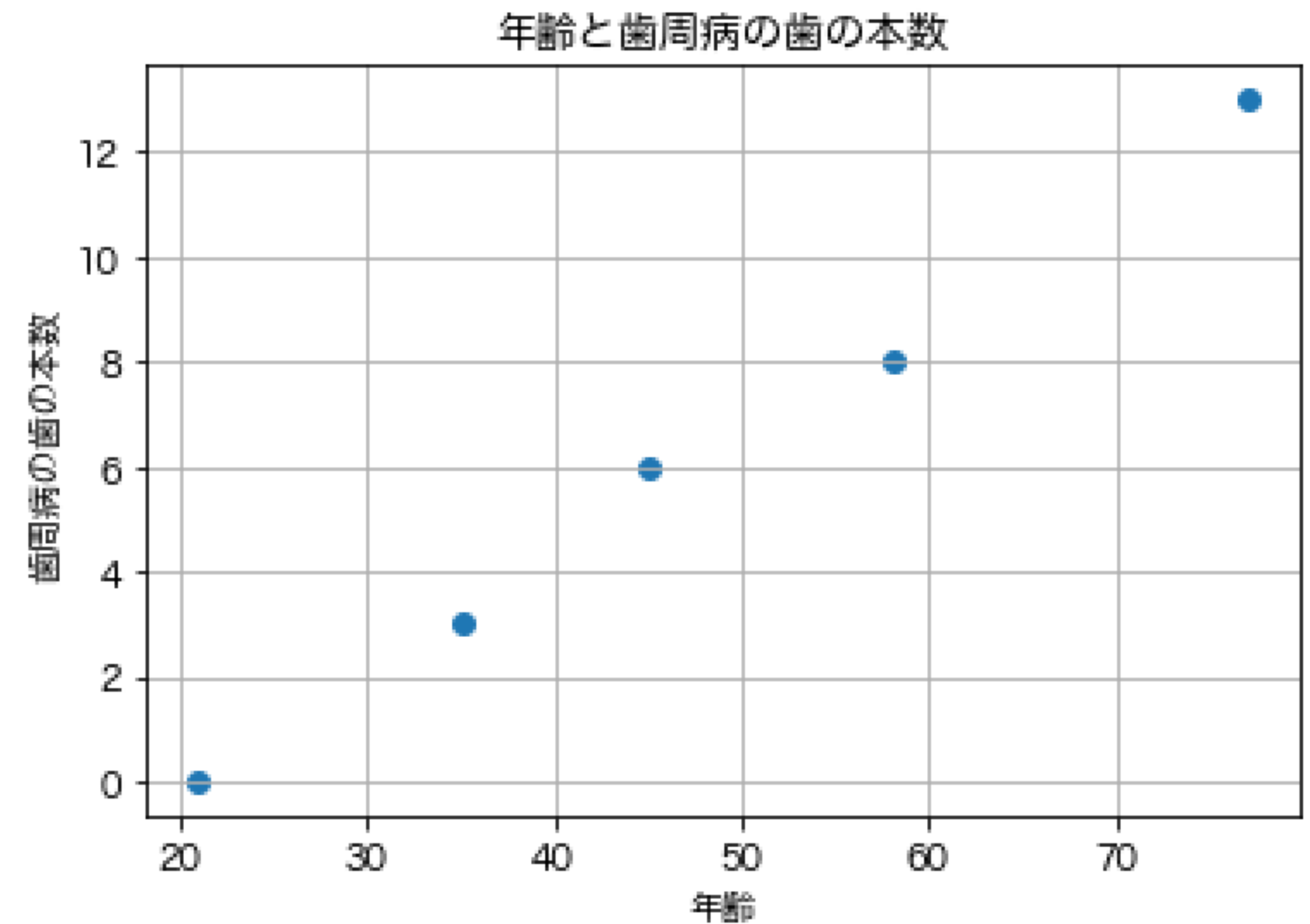
```
plt.xlabel('年齢')
```

```
plt.ylabel('歯周病の歯の本数')
```

```
plt.grid(True)
```

```
plt.scatter(x,y)
```

```
plt.show()
```



日本語が正しく表記された

#8) を実行する

plt.scatter()はxとyの点でしたが、plt.plotとすると点を順に結びます

4)折れ線の作成

```
import matplotlib.pyplot as plt
```

```
# rcParamsという関数をインポートして日本語を反映させる
```

```
from matplotlib import rcParams
```

```
rcParams['font.family']='sans-serif'
```

```
rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro', 'Yu Gothic', 'Meirio']
```

```
x = [35,21,45,58,77]
```

```
y = [3,0,6,8,13]
```

```
plt.figure()
```

```
plt.title('年齢と歯周病の歯の本数')
```

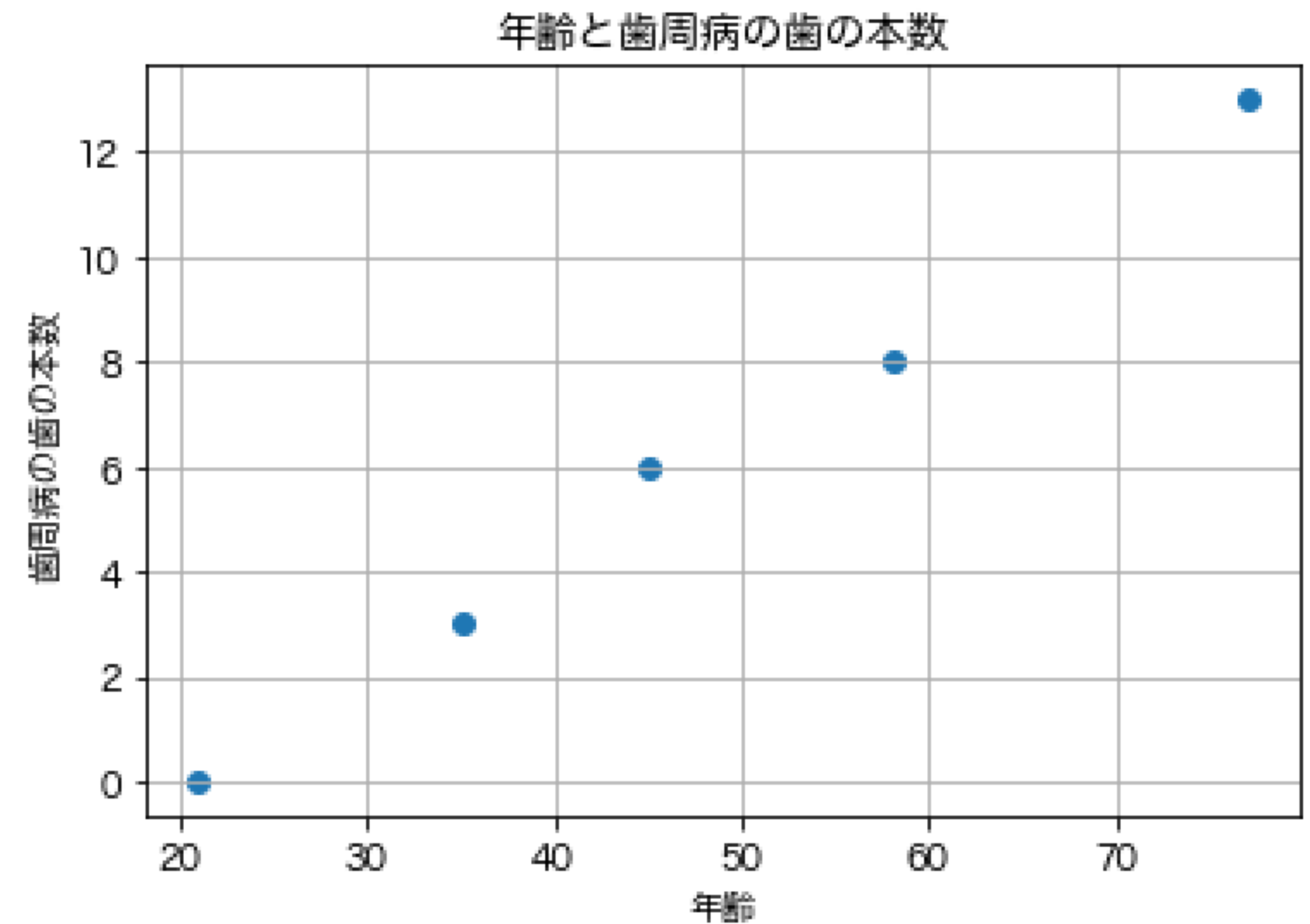
```
plt.xlabel('年齢')
```

```
plt.ylabel('歯周病の歯の本数')
```

```
plt.grid(True)
```

```
plt.plot(x,y)
```

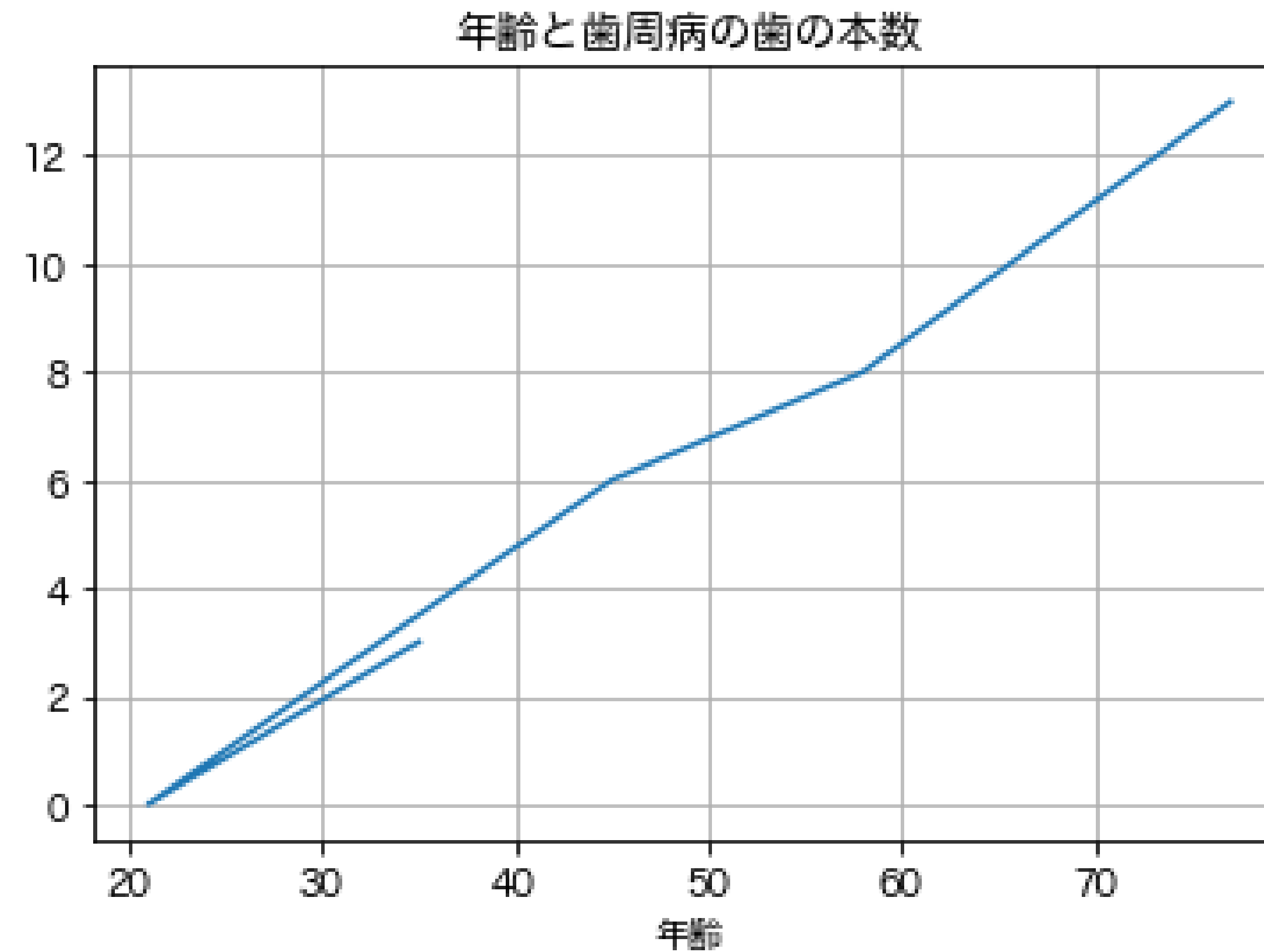
```
plt.show()
```



日本語が正しく表記された

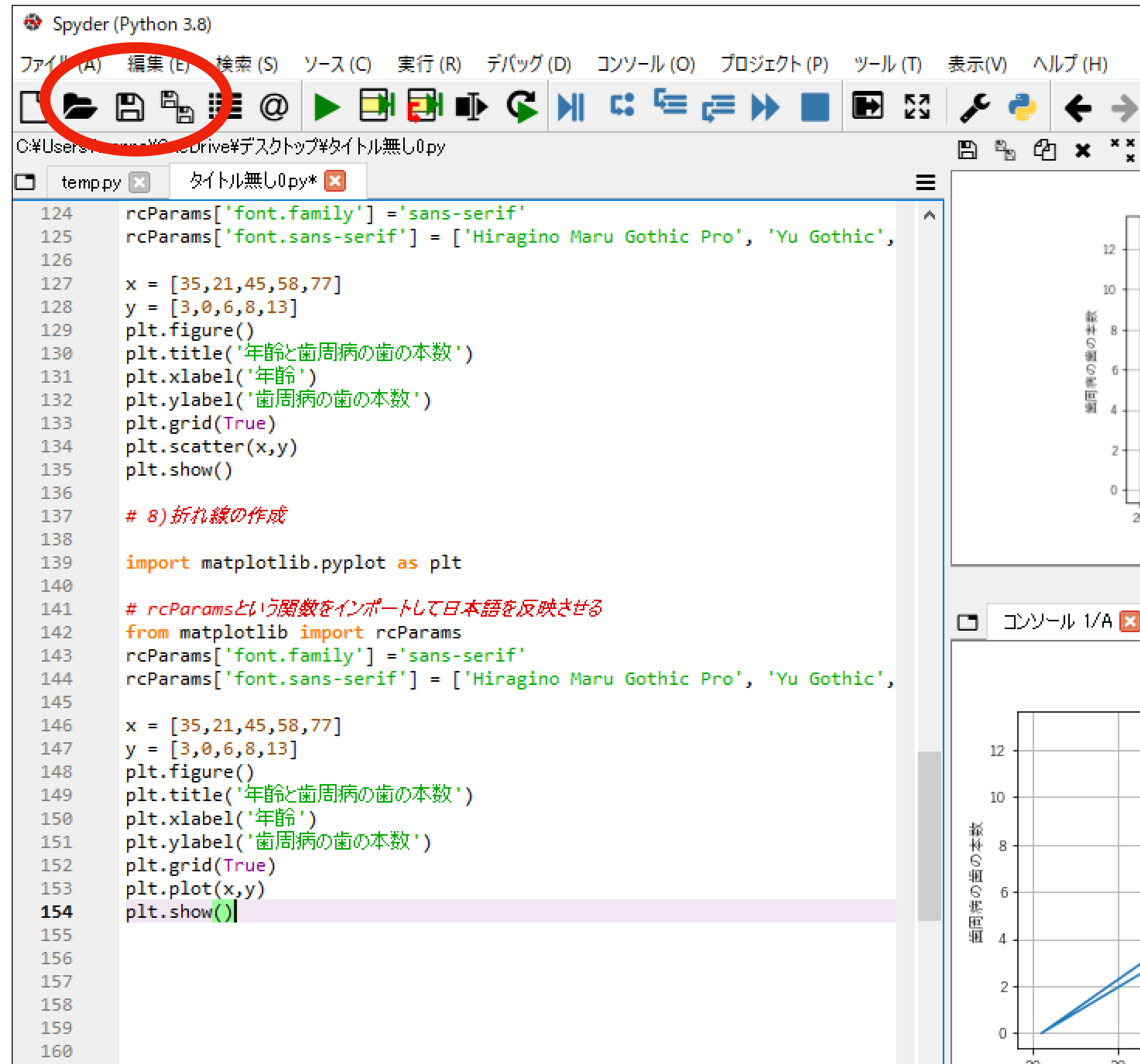
#8) を実行する

plt.scatter()はxとyの点でしたが、plt.plotとすると点を順に結びます

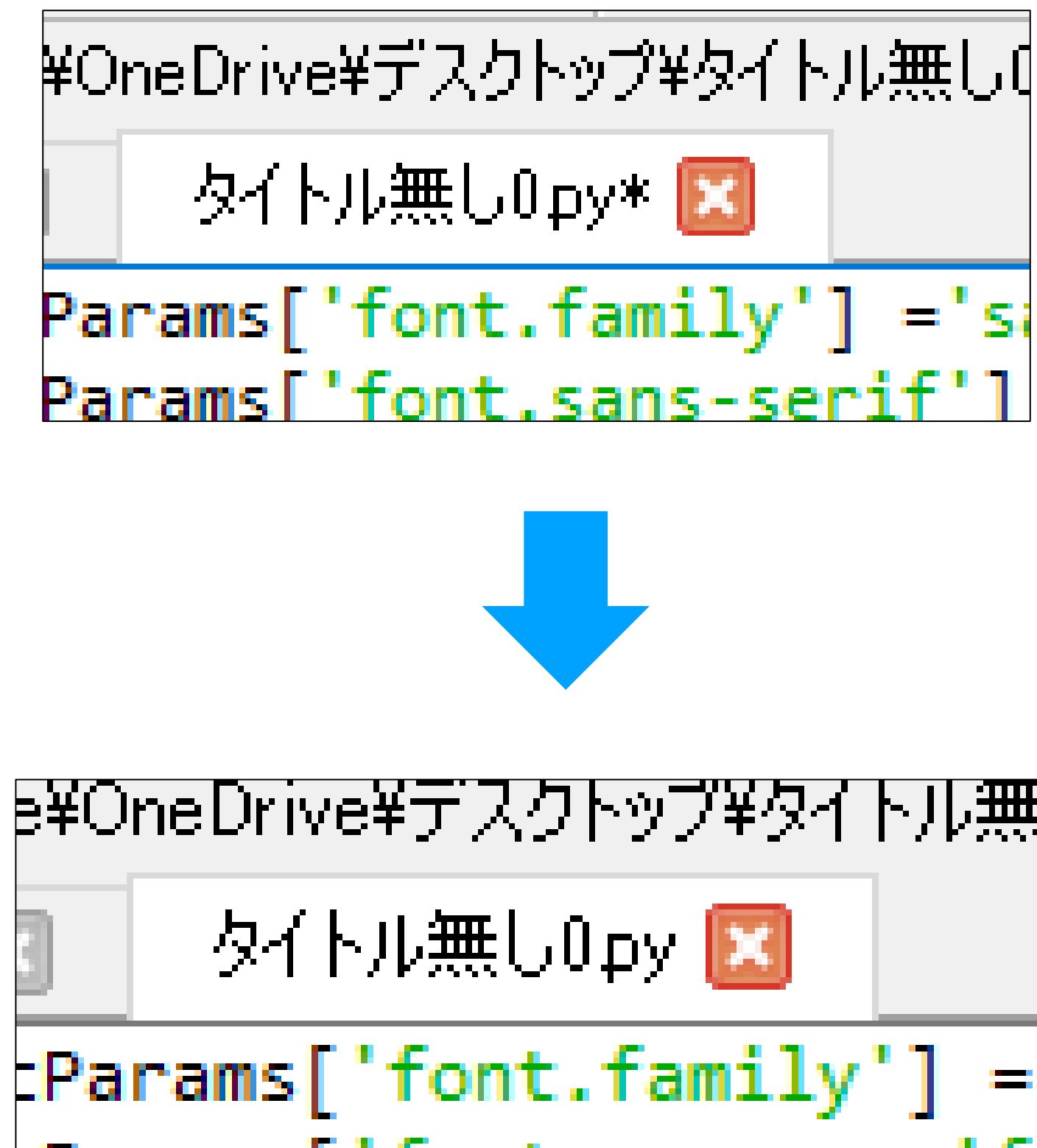


最後にデータを保存しておく

保存するとファイル名の“*”が消えます。（何か変更すると*が付きます）

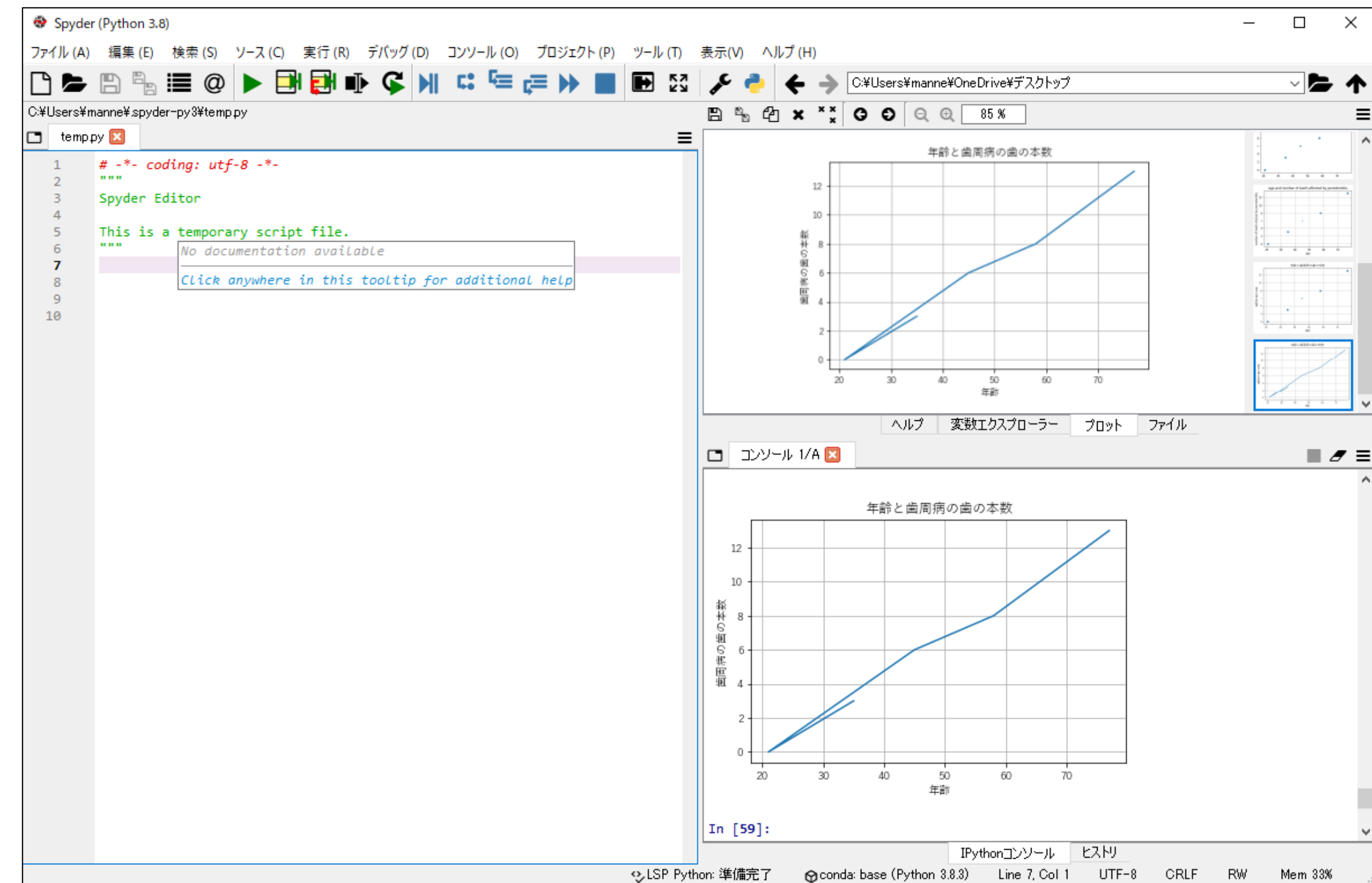
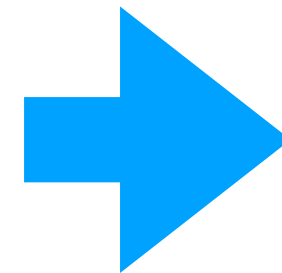
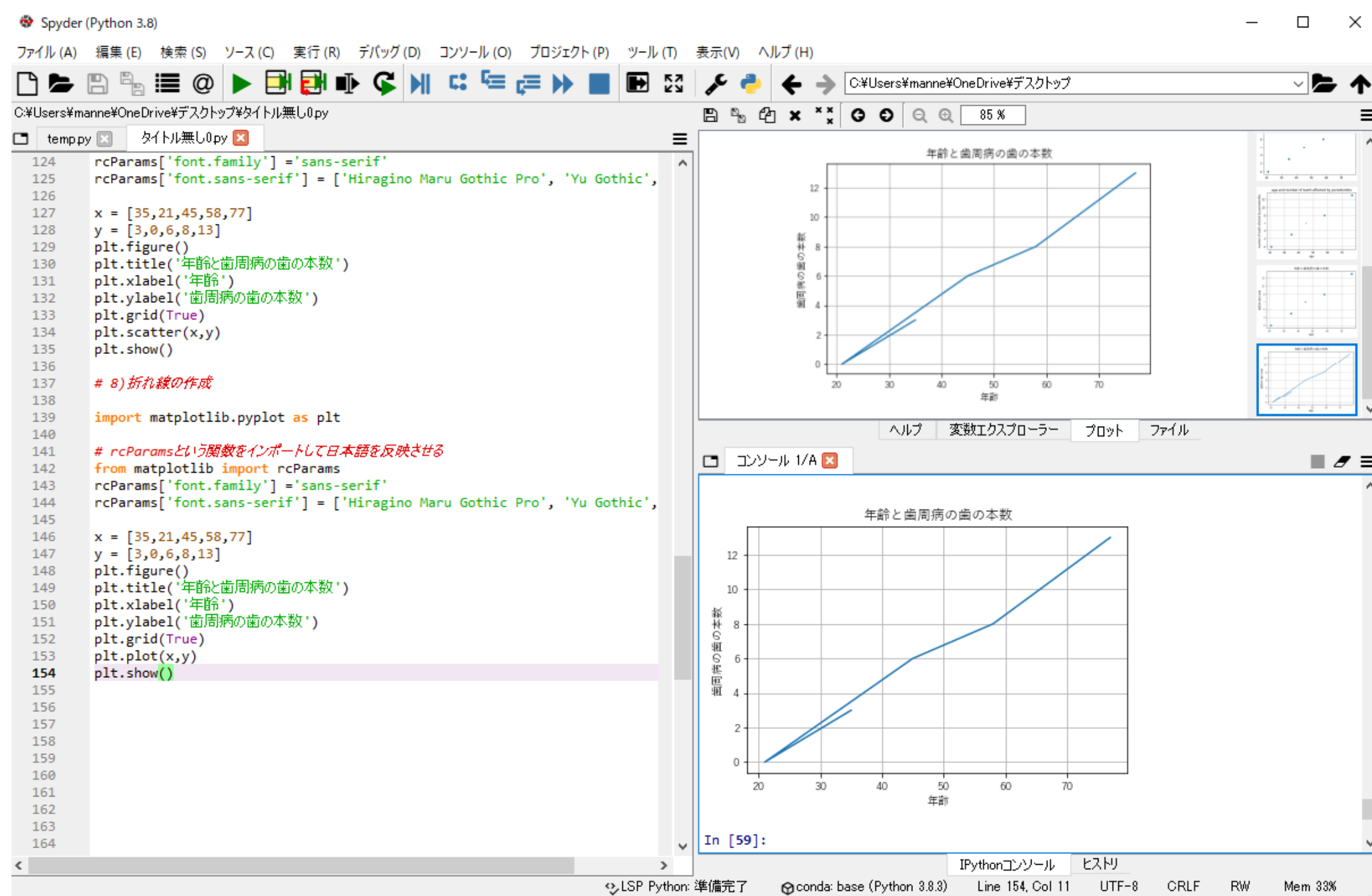


```
124 rcParams['font.family'] = 'sans-serif'
125 rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro', 'Yu Gothic',
126
127 x = [35, 21, 45, 58, 77]
128 y = [3, 0, 6, 8, 13]
129 plt.figure()
130 plt.title('年齢と歯周病の歯の本数')
131 plt.xlabel('年齢')
132 plt.ylabel('歯周病の歯の本数')
133 plt.grid(True)
134 plt.scatter(x, y)
135 plt.show()
136
137 # 8) 折れ線の作成
138
139 import matplotlib.pyplot as plt
140
141 # rcParamsという関数をインポートして日本語を反映させる
142 from matplotlib import rcParams
143 rcParams['font.family'] = 'sans-serif'
144 rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro', 'Yu Gothic',
145
146 x = [35, 21, 45, 58, 77]
147 y = [3, 0, 6, 8, 13]
148 plt.figure()
149 plt.title('年齢と歯周病の歯の本数')
150 plt.xlabel('年齢')
151 plt.ylabel('歯周病の歯の本数')
152 plt.grid(True)
153 plt.plot(x, y)
154 plt.show()
```



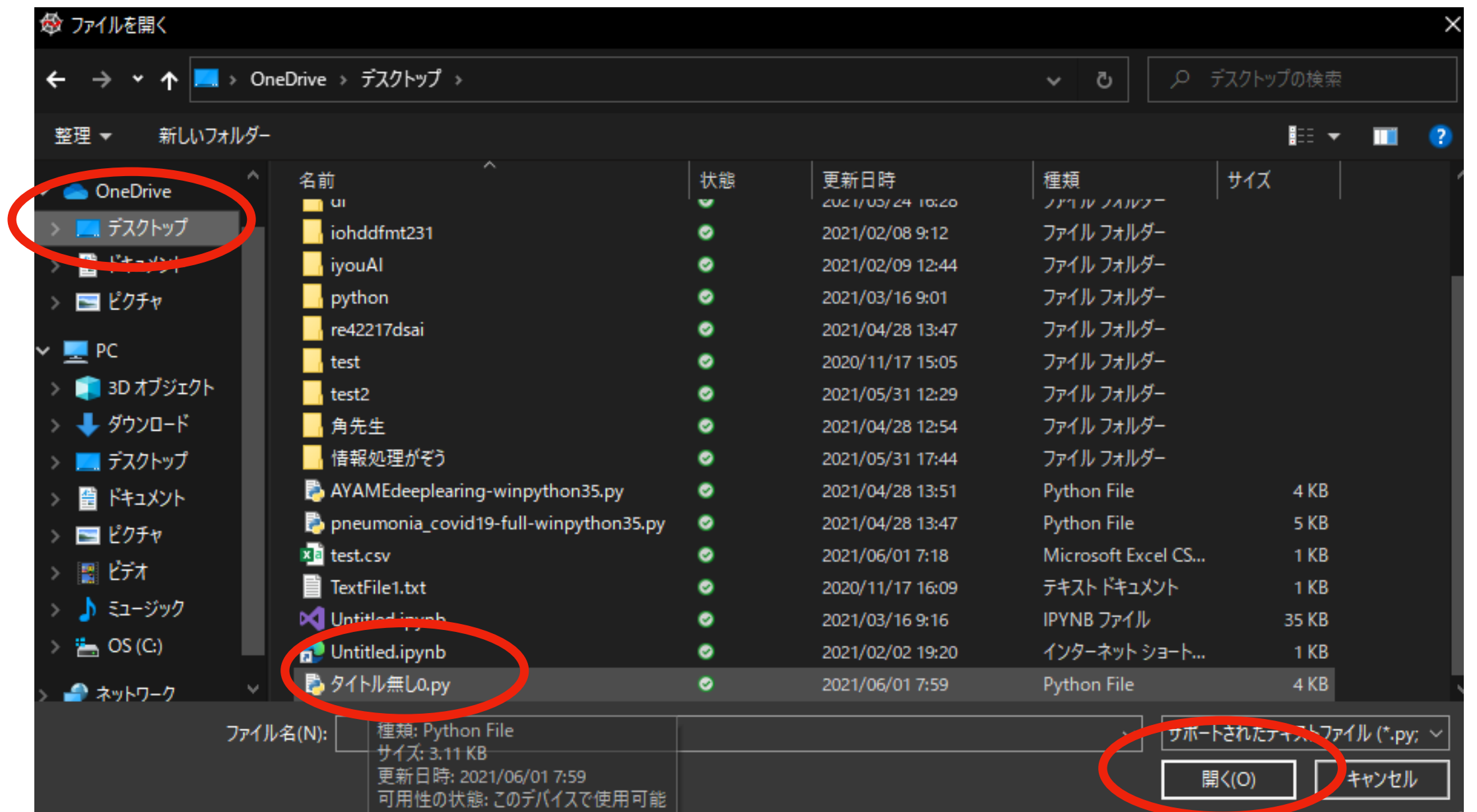
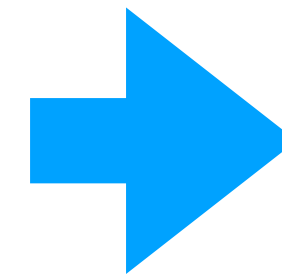
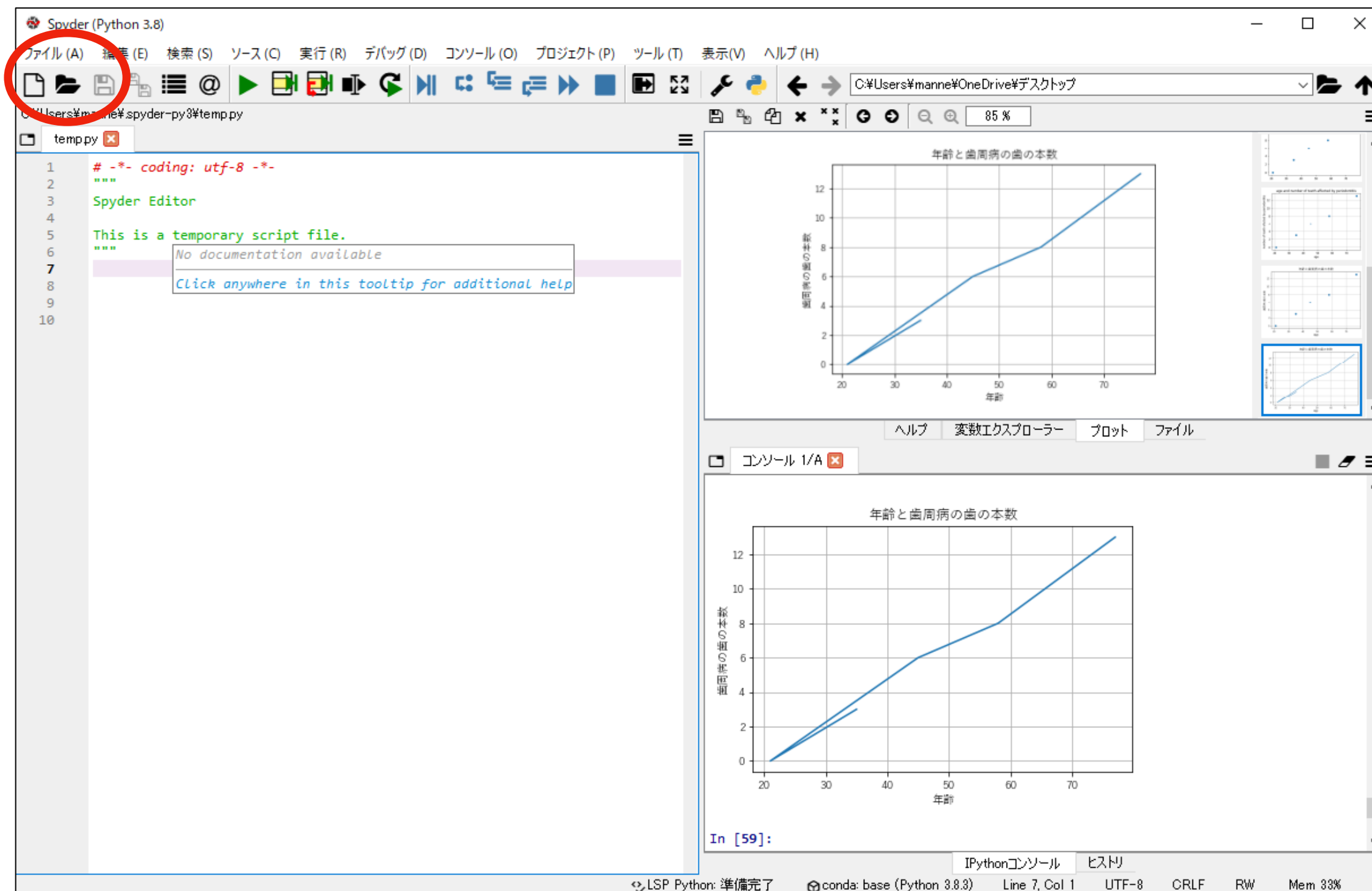
最後にデータを保存しておく

エディタ上部のファイル名の右にある✕をクリックして、
エディタ内のpythonプログラムを消してみる



保存したファイルを開いてみる

「ファイル」→「開く」で、先ほど保存したpythonプログラム「enshu1.py」を開く



最後にデータを保存しておく

再度続きからエディタを修正することが出来ます。

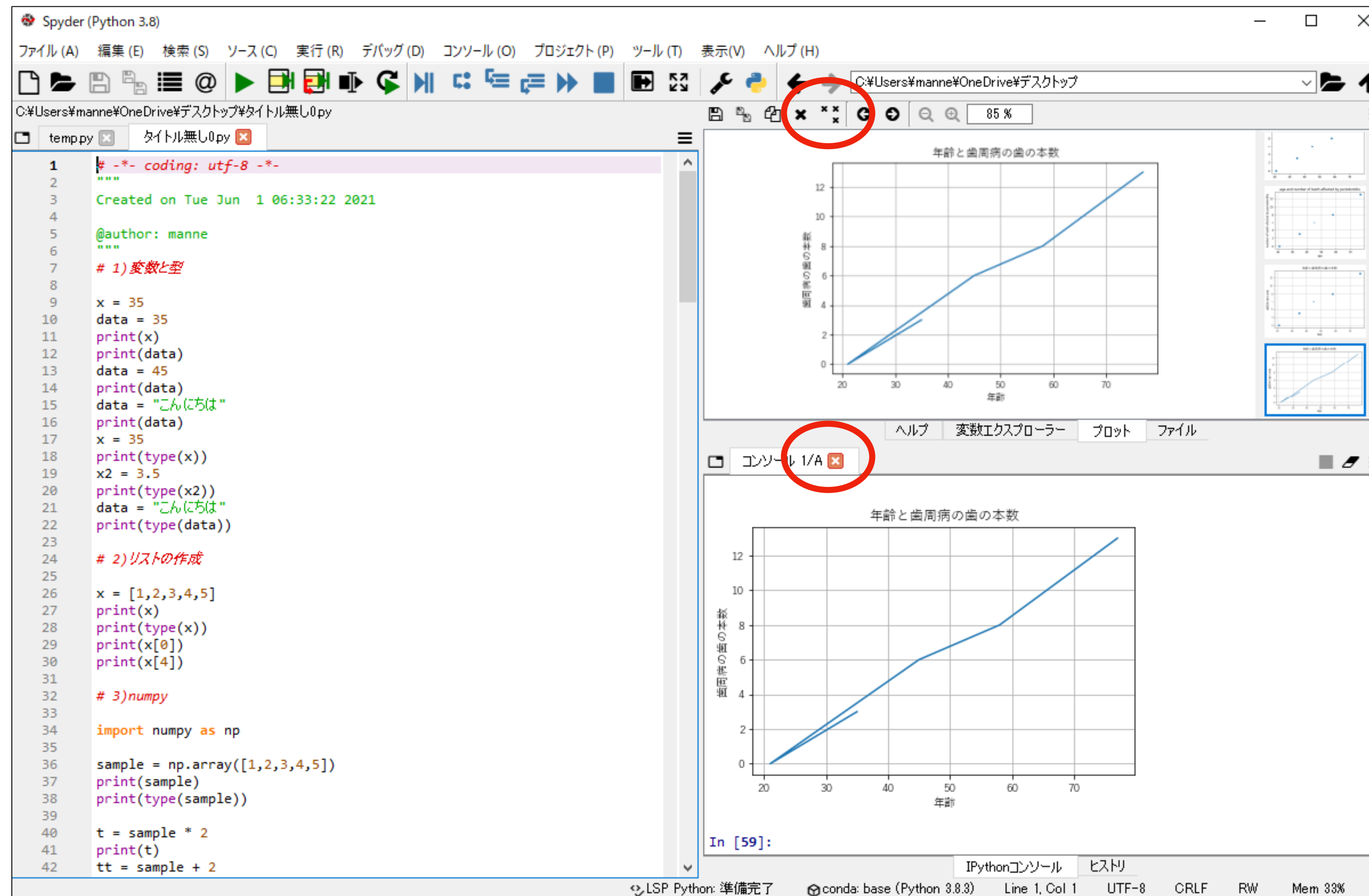
The screenshot displays the Spyder Python IDE interface. The left pane shows a Python script with the following code:

```
1  -*- coding: utf-8 -*-
2
3  Created on Tue Jun  1 06:33:22 2021
4
5  @author: manne
6
7  # 1) 変数と型
8
9  x = 35
10 data = 35
11 print(x)
12 print(data)
13 data = 45
14 print(data)
15 data = "こんにちは"
16 print(data)
17 x = 35
18 print(type(x))
19 x2 = 3.5
20 print(type(x2))
21 data = "こんにちは"
22 print(type(data))
23
24 # 2) リストの作成
25
26 x = [1,2,3,4,5]
27 print(x)
28 print(type(x))
29 print(x[0])
30 print(x[4])
31
32 # 3) numpy
33
34 import numpy as np
35
36 sample = np.array([1,2,3,4,5])
37 print(sample)
38 print(type(sample))
39
40 t = sample * 2
41 print(t)
42 tt = sample + 2
```

The right pane features a plot window titled "年齢と歯周病の歯の本数" (Age and Number of Teeth Affected by Periodontitis). The plot shows a line graph with age on the x-axis (ranging from 20 to 70) and the number of teeth affected on the y-axis (ranging from 0 to 12). The data points are connected by a blue line, showing a positive correlation. Below the plot is a console window titled "コンソール 1/A" showing the output of the script, including the IPython prompt "In [59]:".

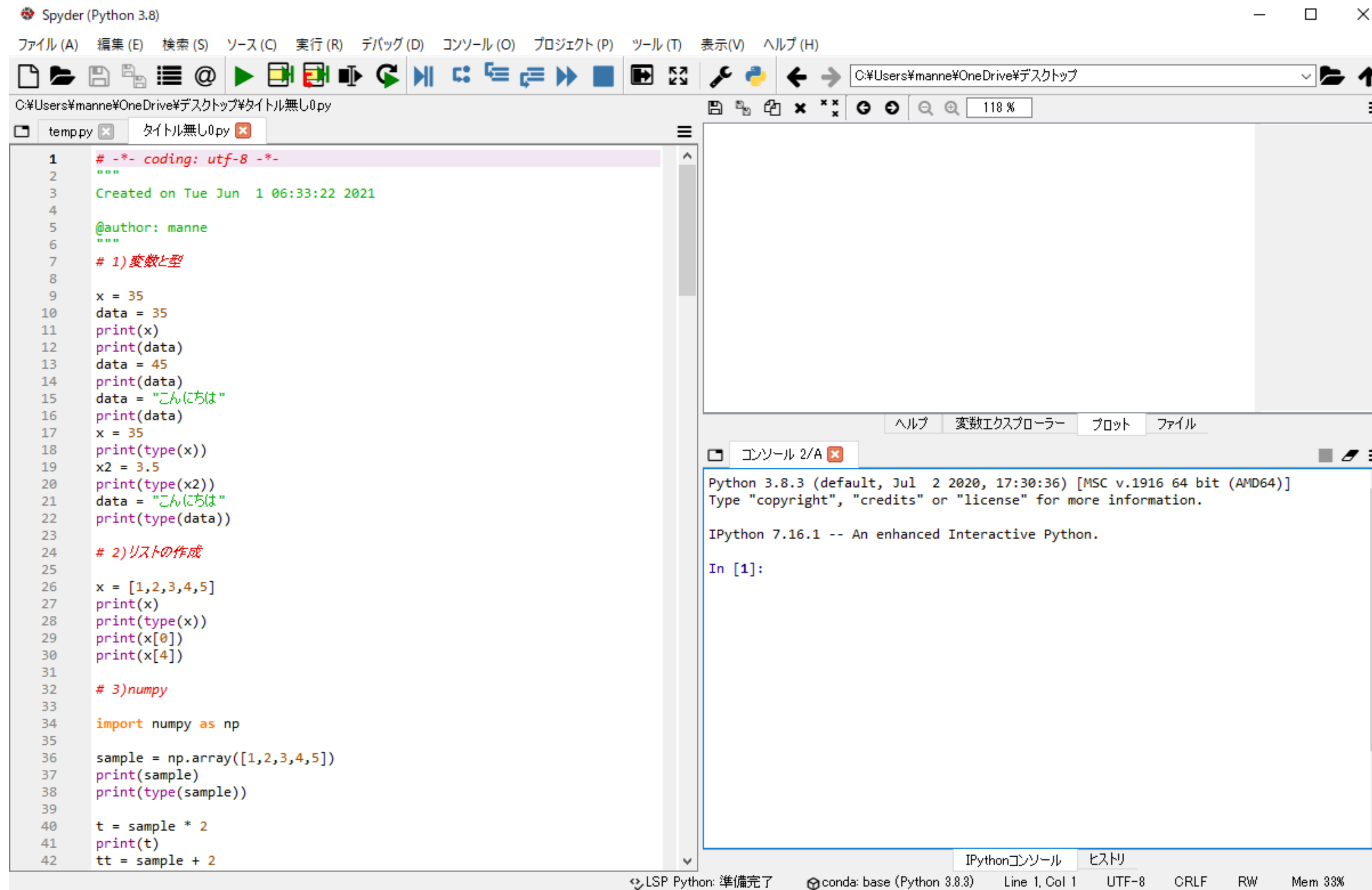
おまけ

プロットとコンソールも下のアイコンを押すとリセットすることができます。



おまけ

プロットとコンソールも下のアイコンを押すとリセットすることができます。



作図を試みよう

xに2, 5, 8, 11, 14, 16, 20

yに1, 4, 8, 11, 15, 30, 55

タイトルを”名前_学籍番号”

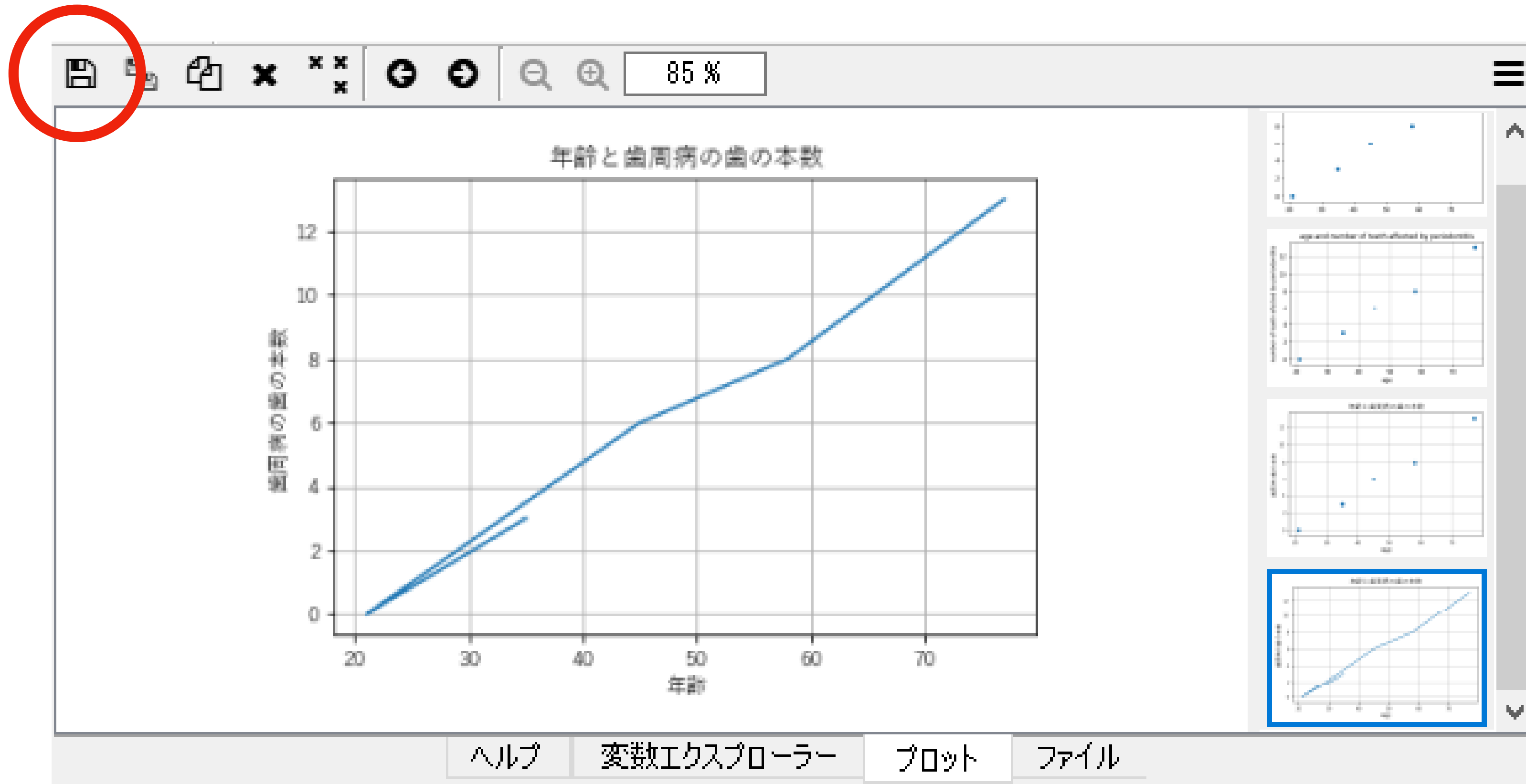
x軸の名前を”X軸”

y軸の名前を”Y軸”

として、散布図と折れ線グラフをそれぞれ作成してみよう

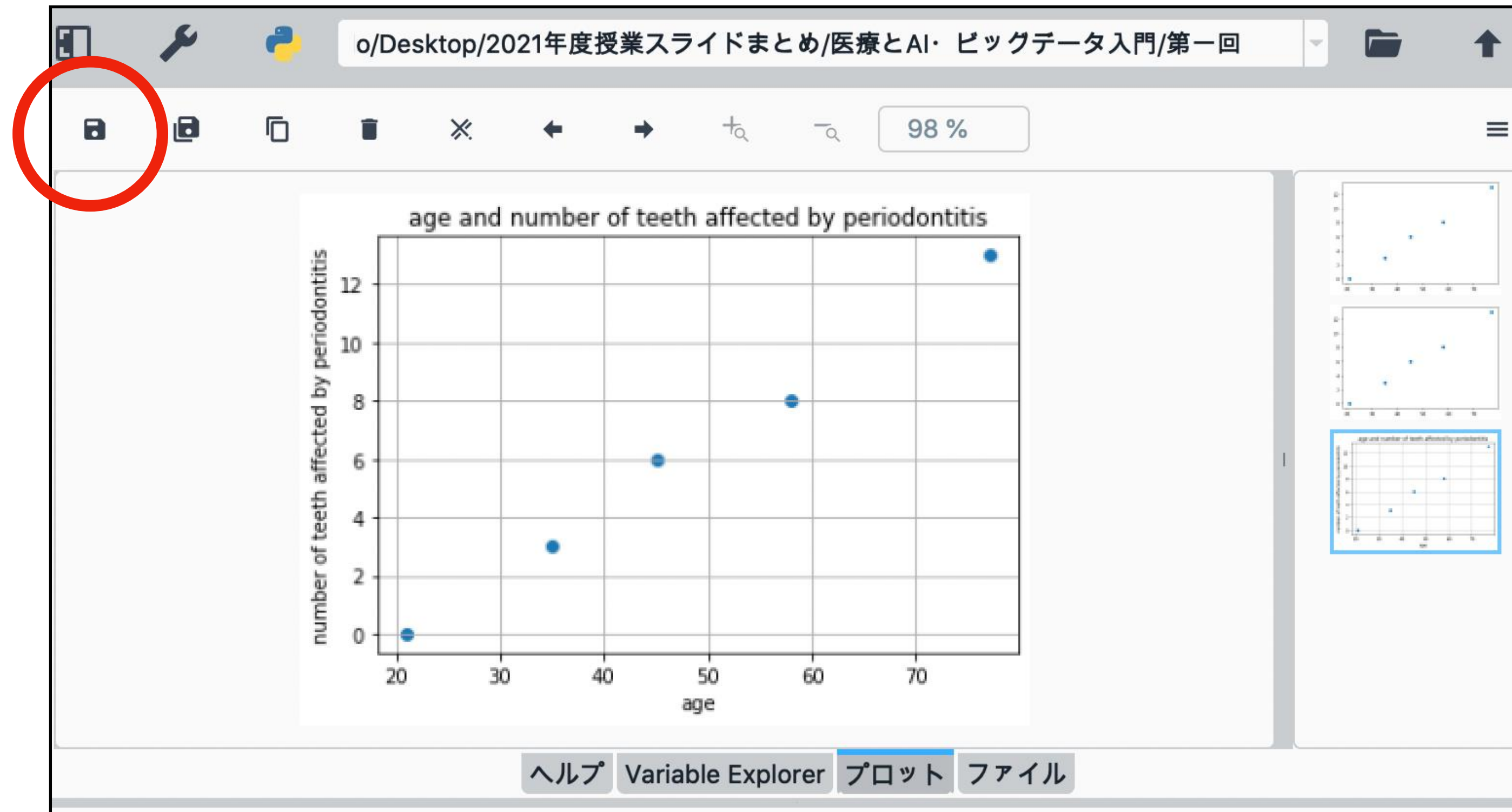
画面を保存して、それぞれ「学籍番号_名前_散布図」、
「学籍番号_名前_折れ線」で提出して下さい

画像の保存方法



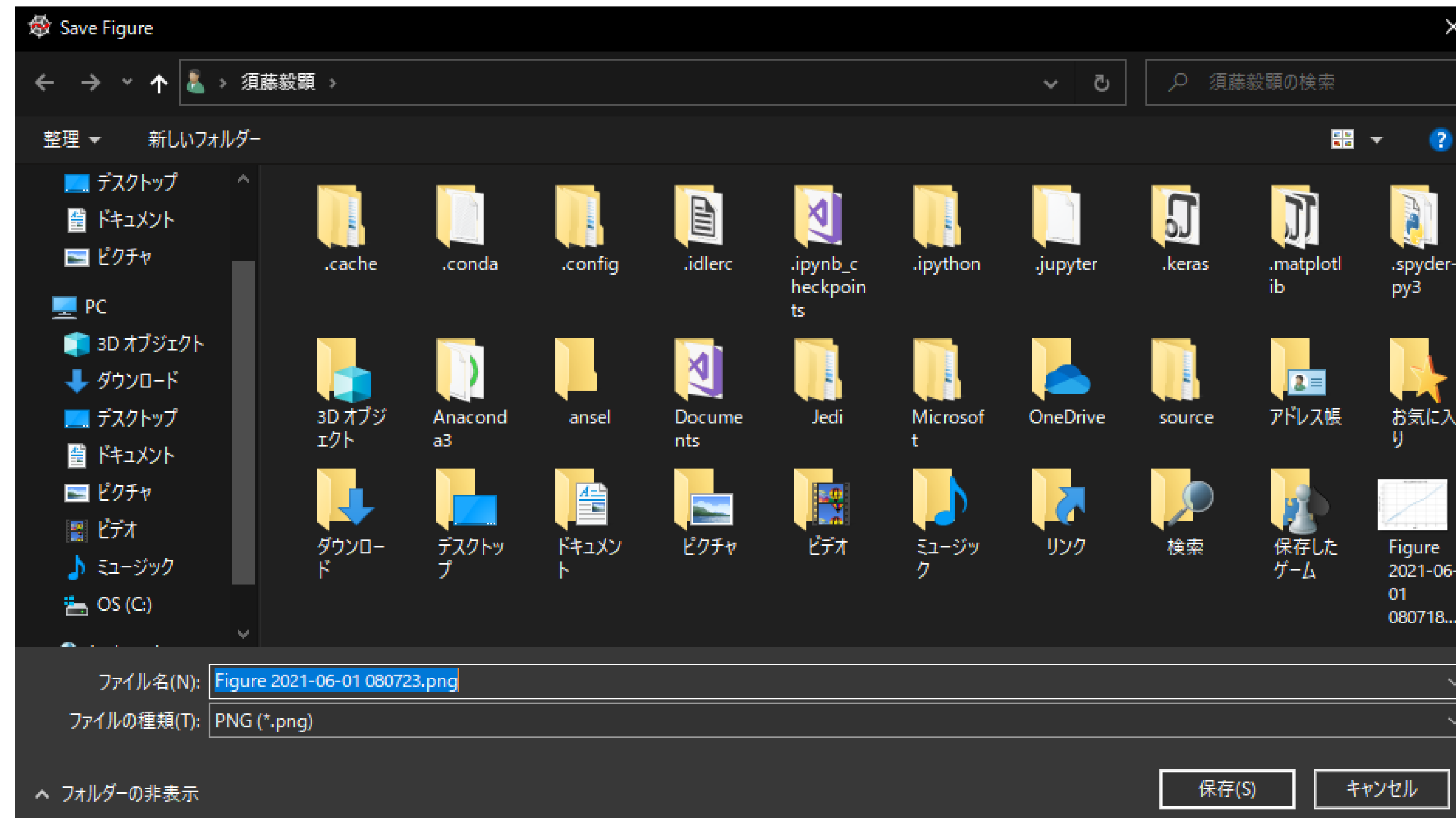
左上の保存アイコンをクリック

画像の保存方法



バージョン5も同様

画像の保存方法

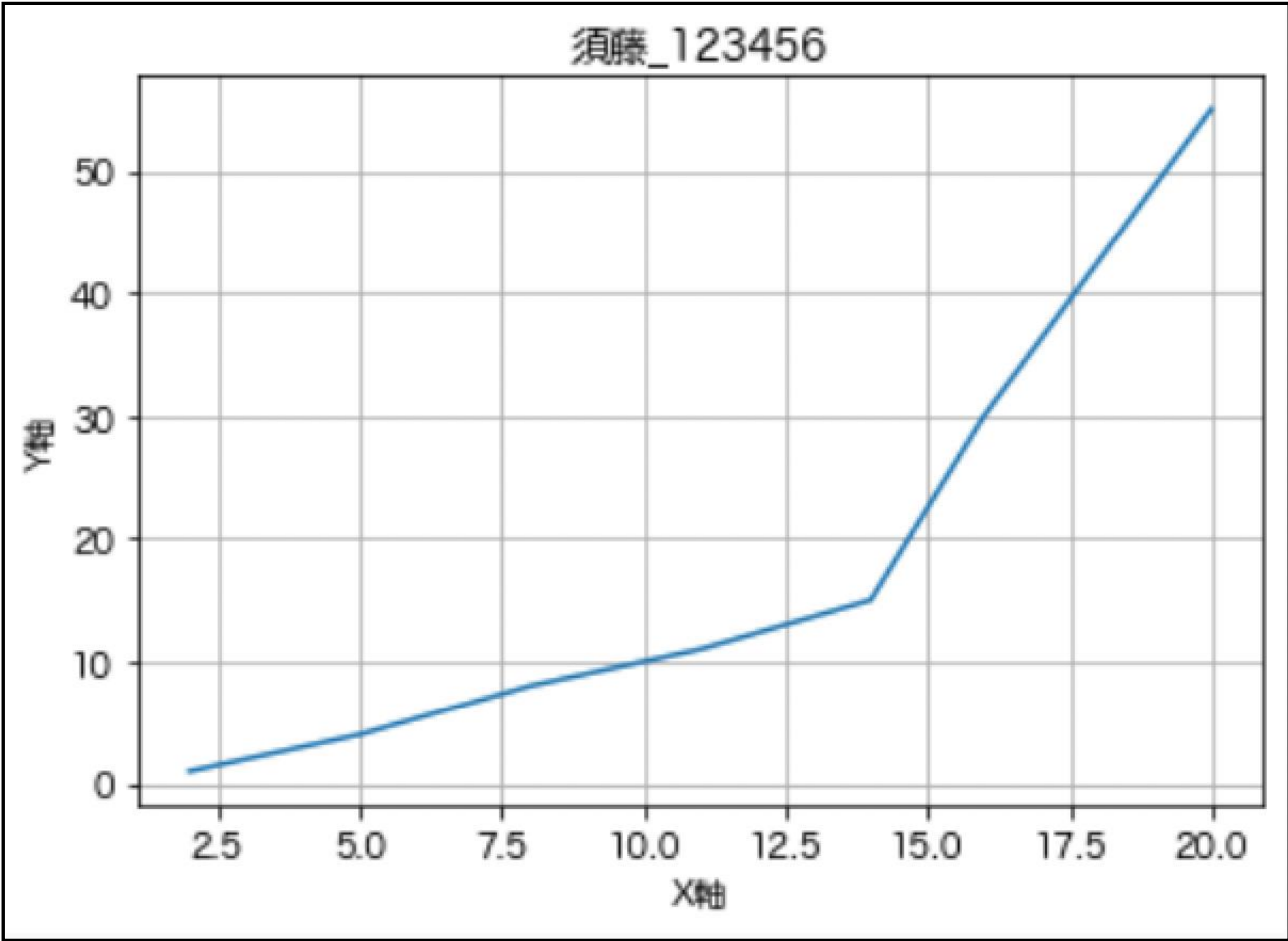
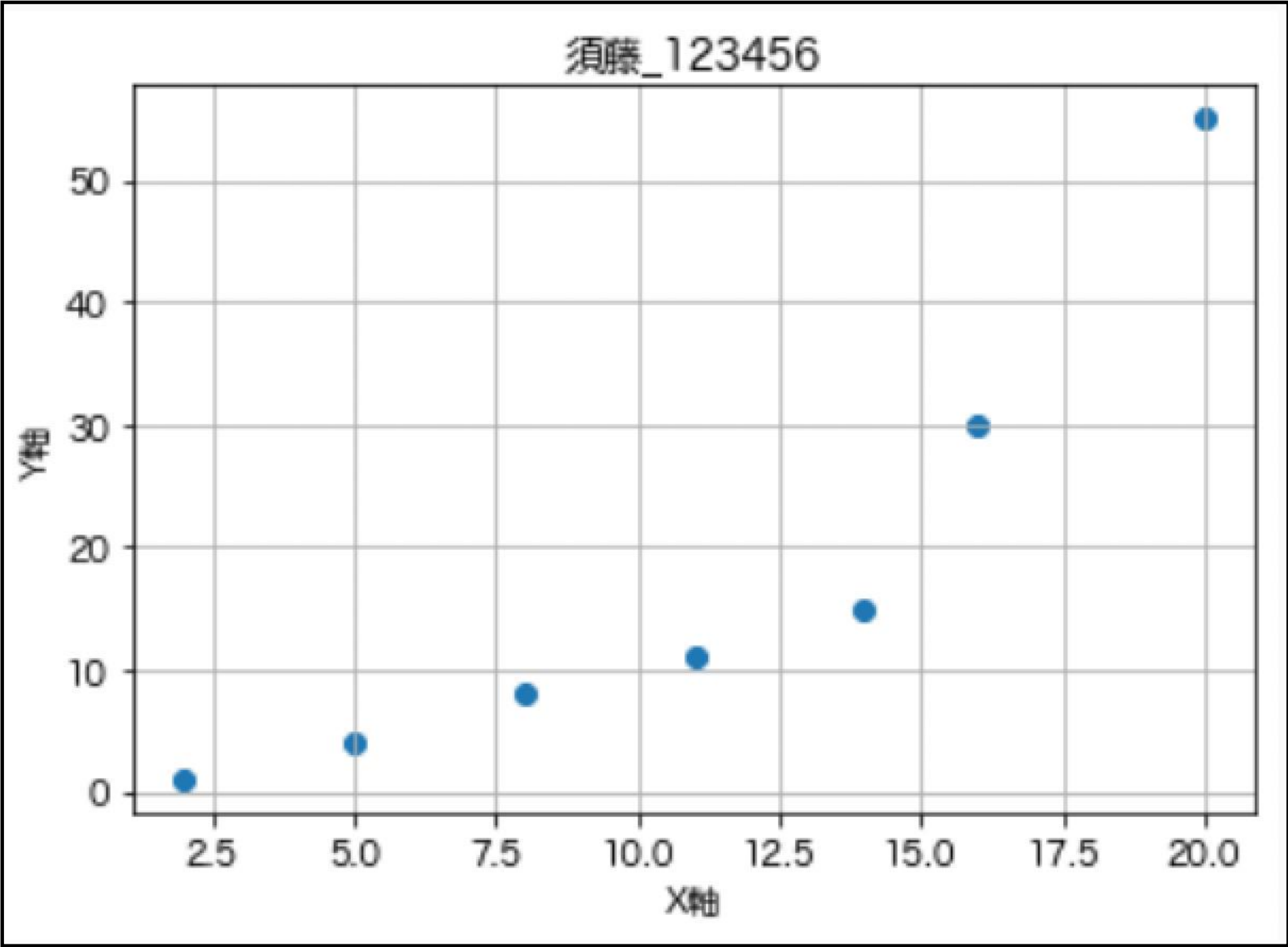


場所を指定して保存

終わり

初回はこれで終わりになります。

次回はこれらの知識を使いながら機械学習に取り組んでみましょう



休憩10分で考える