Python 入門書（導入～基本編）

作成日：2020 年 8 月 21 日

目次

本書はプログラミング経験者向けに書いた、Python の入門書である。

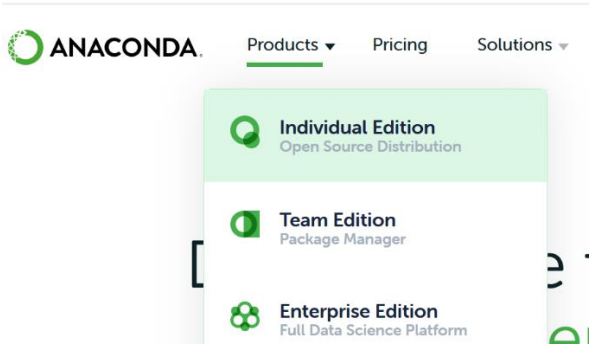
1　開発環境のインストール

Anaconda は Python 本体、ライブラリを最初からまとめたパッケージソフト。
下のように分野ごとに Anaconda のアプリケーションを使い分けることができる。

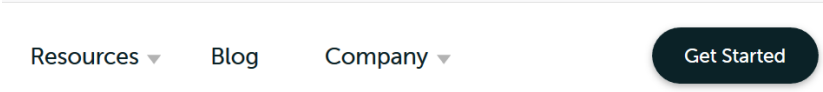
分野	ツール
データサイエンス	Jupyter、JupyterLab、Spyder、RStudio
データ分析	Dask、numpy、pandas、Numba
データ視覚化	Matplotlib、Bokeh、Datashader、Holoviews
機械学習およびディープラーニングモデル作成-	Scikit-learn、Tensorflow、h20、Theano

1.1　Anaconda のダウンロード

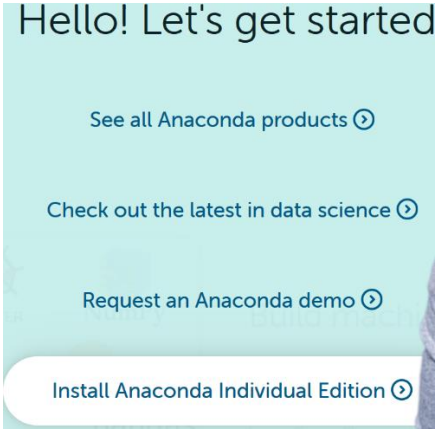
(1) https://www.anaconda.com/へアクセス
(2) 以下のメニューをクリック



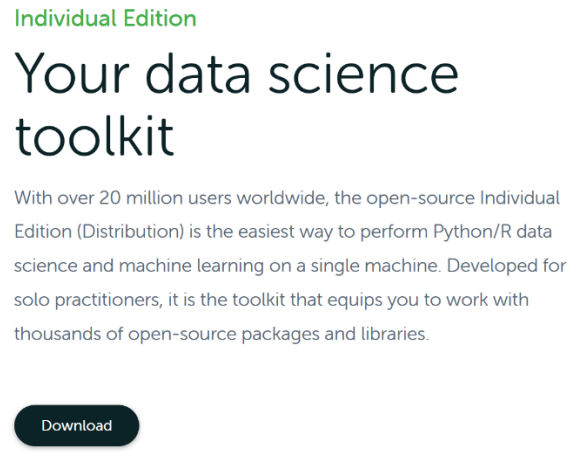
(3) 右上の「GetStarted」をクリック



(4) 「Install Anaconda Individual Edition」をクリック



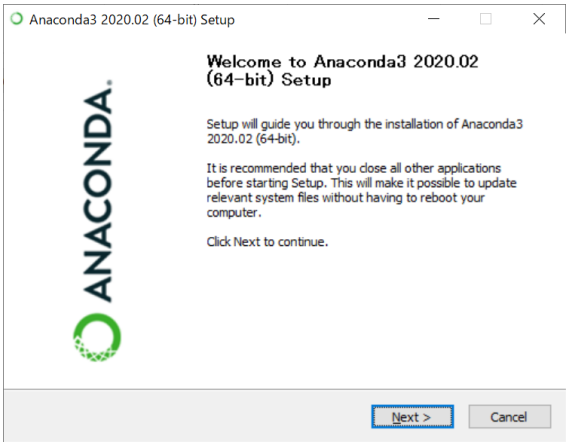
(5) 「Download」をクリック



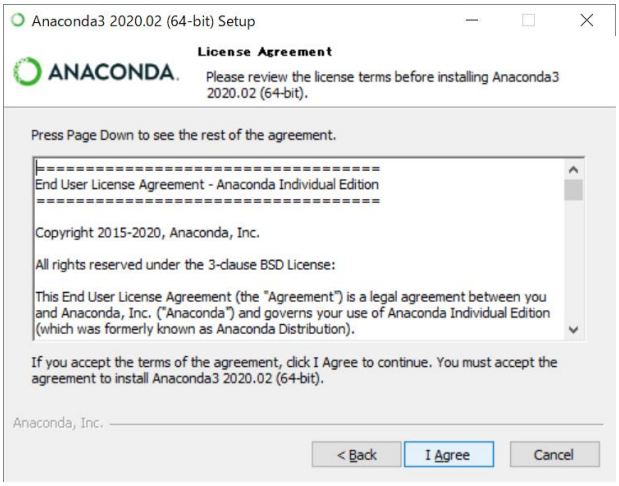
(6) インストーラをダウンロード

1.2 インストール

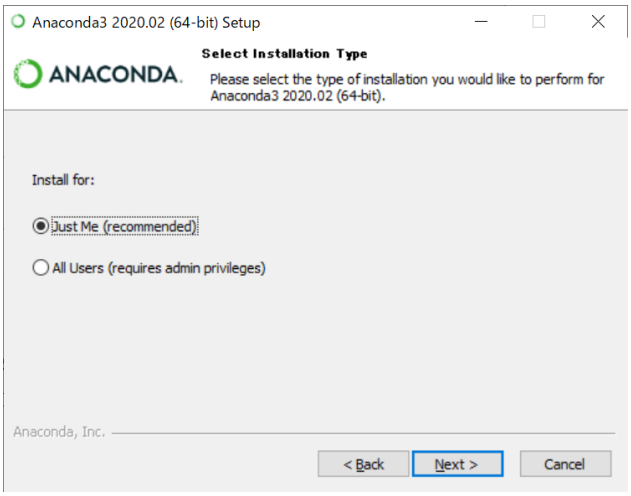
(1) インストーラを実行して「Next」をクリック



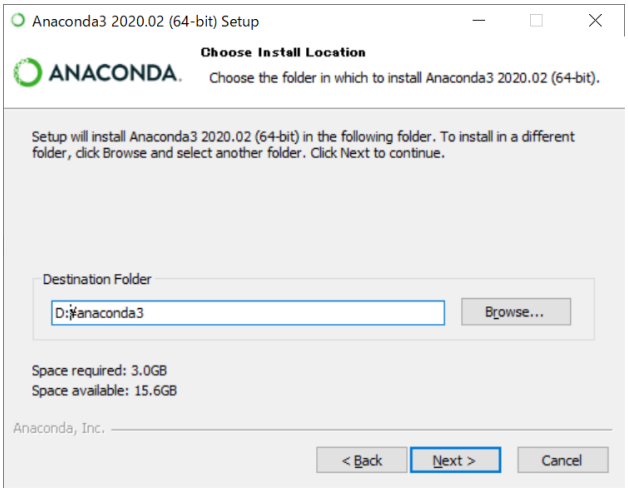
(2) 「I Agree」をクリック



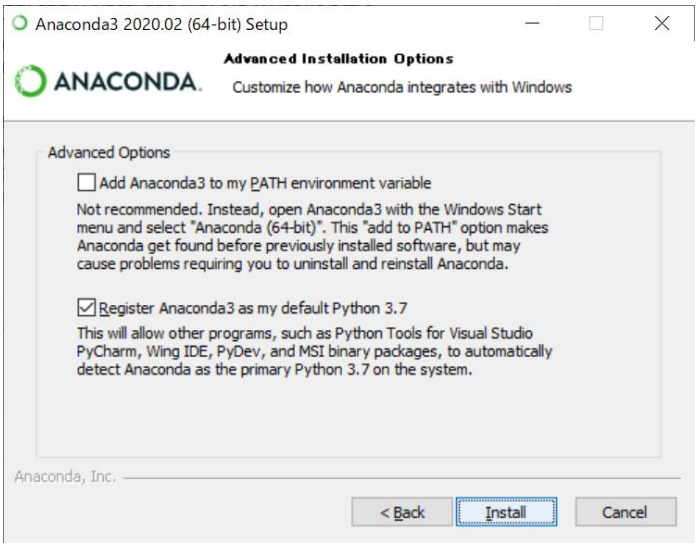
(3) 「Just Me」を選択した状態で「Next」をクリック



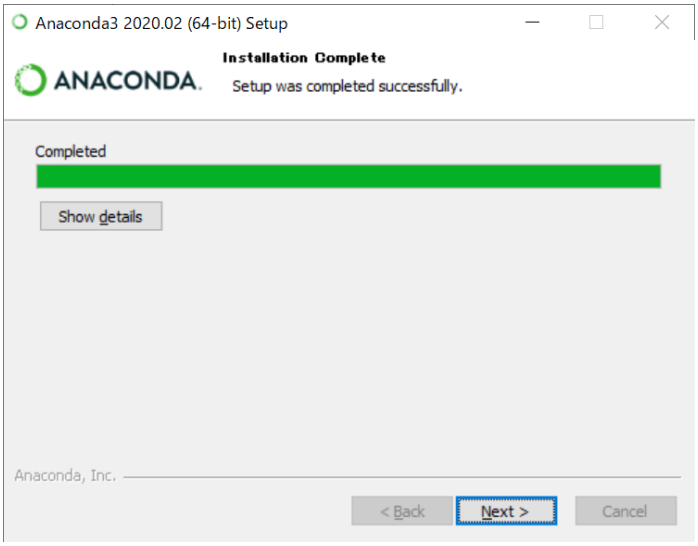
(4) インストール先を設定して「Next」



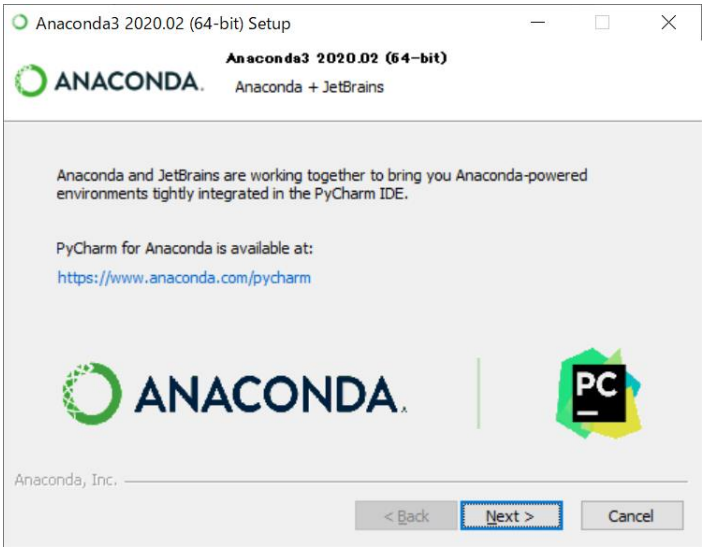
(5) そのまま「Install」をクリック



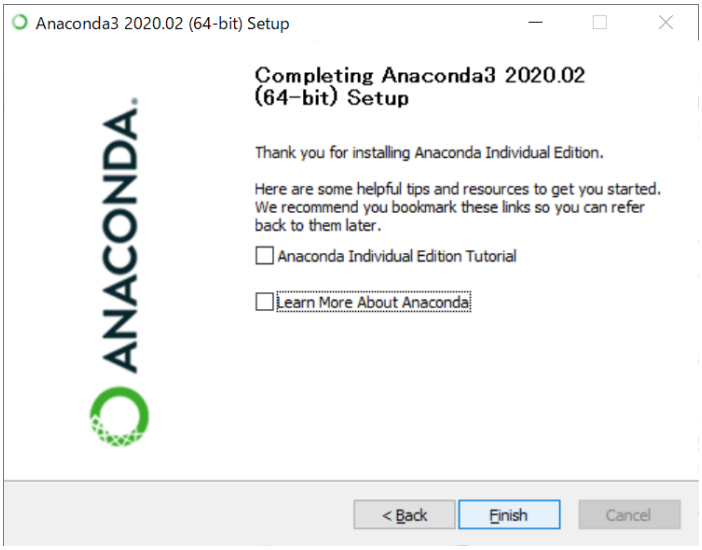
(6) 「Next」をクリック



(7) 「Next」をクリック

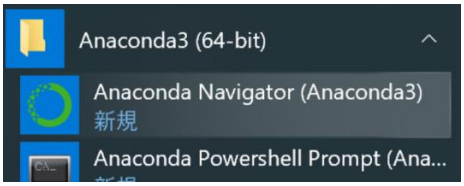


(8) チェックを外して「Finish」をクリック

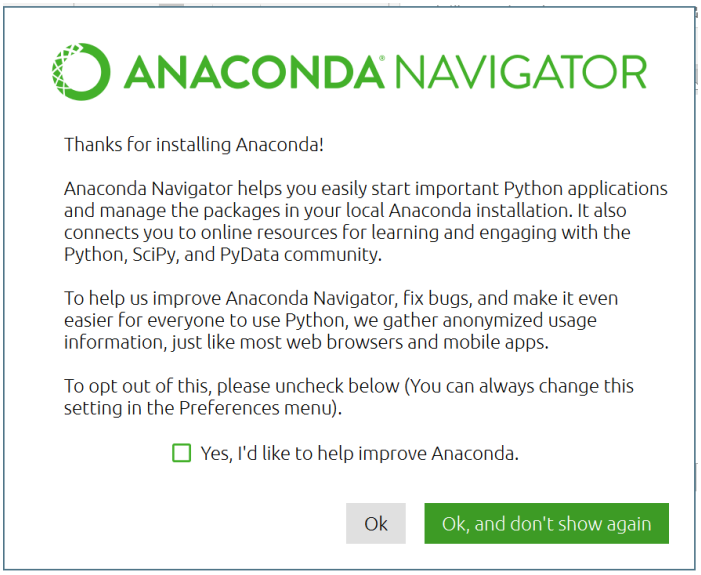


1.3 最初の起動

(1) Anaconda-Navigator(Anaconda3)を起動



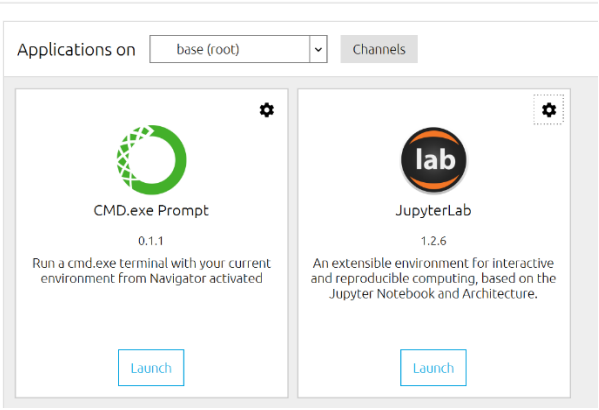
(2) Navigator の改善協力のメッセージが表示されるので、チェックを外して、「OK,and don't・・・」をクリック



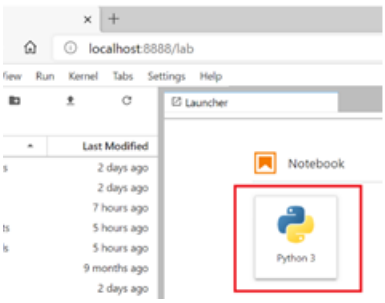
2　Python プログラミングの基礎

2.1　JupyterLab の起動

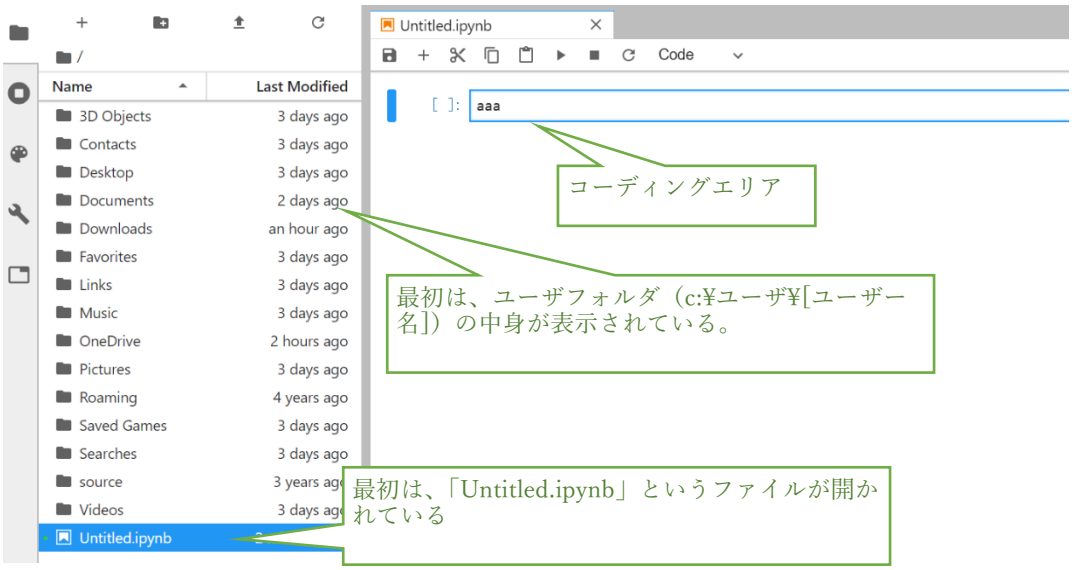
(1) JupyterLab の「Launch」をクリック



(2) 「Notebook」の「Python3」をクリック

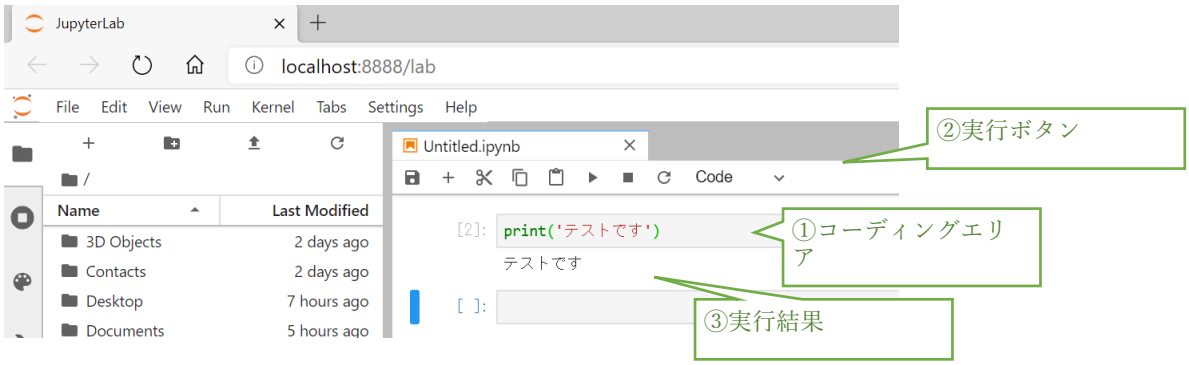


(3) ブラウザが立ち上がり、コーディングや実行が可能となる


Untitled.ipynb
Code
[ ]:
aaa
コーディングエリア
最初は、ユーザフォルダ（c:¥ユーザ¥[ユーザー名]）の中身が表示されている。
最初は、「Untitled.ipynb」というファイルが開かれている
/
Name
Last Modified
3D Objects 3 days ago
Contacts 3 days ago
Desktop 3 days ago
Documents 2 days ago
Downloads an hour ago
Favorites 3 days ago
Links 3 days ago
Music 3 days ago
OneDrive 2 hours ago
Pictures 3 days ago
Roaming 4 years ago
Saved Games 3 days ago
Searches 3 days ago
source 3 years ago
Videos 3 days ago
Untitled.ipynb

2.2　コーディングから実行まで

コーディングエリアにプログラムを記述して、実行結果をクリックすると、実行結果が表示される。



2.3　文字列の操作

項目名	コーディング例	結果
連結（結合）	print('1' + '1')	11
繰り返し	print('1' * 3)	111
エスケープシーケンスは バックスラッシュ	print('¥'') print('¥"')	' "
改行	print('あ¥n い')	あ い

2.4　変数の代入や計算

項目名	コーディング例	結果
まとめて変数へ代入 （アンパック代入）	name,age='山田',18 print(name) print(age)	山田 18
複合代入演算子 += -= *= /=	i = 1 i += 1 print(i)	2 i = i + 1 と同じ。

2.5　データ型変換

項目名	コーディング例	結果
型変換関数 int(x) float(x) str(x) bool(x)	x = '1' x = int(x) print(1 + x)	2

文字型の値と数値型の値を結合する際、数値型を文字型に変換をする。
下の例は、文字型 ＋ 数値型 なのでエラーとなる。

コーディング例
i = 1 i = i + 1 print('答えは' + i)

よって次のように文字型に変換することで正常な結果となる。

コーディング例
i = 1 i = i + 1 print('答えは' + str(i))

2.6　キーボードからの入力を変数へ代入

変数
x

例

コーディング例	結果
name = input('名前を入力してください：') print('ようこそ' + name + 'さん')	名前を入力してください：　山田太郎 ようこそ山田太郎さん

①キーボードからの入力

②キーボードから入力した値を表示

※注意

　input 関数で代入した値は文字型になるので、数値として計算をする場合は、int 関数や float 関数を使用する。

2.7　プレースホルダー

次のように変数の箇所を{}にして、.format()に変数名をしていすることで、コードがスッキリする。

コーディング例
strName = '山田太郎' intAge = 20 print('名前は{}です。年齢は{}です'.format(strName , intAge))

2.8　分岐（if 文）

if 分の基本的な書き方は次のとおり。

基本書式

```
if 条件式 :
    命令
elif 条件式 :
    命令
else :
    命令
```

Python はブロック（if 内の命令処理）はインデントで指定する。
（この後出てくる while なども同様）

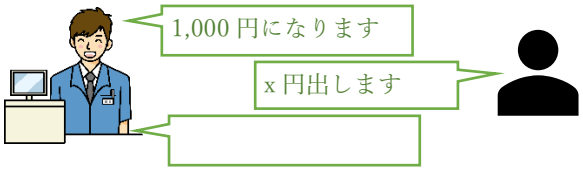
・条件は、「and」、「or」、「not」の論理演算子で組み合わせることが可能。

・比較演算子は以下のとおり。

比較演算子一覧

x == y	x と y が等しい
x != y	x と y が等しくない
x > y	x は y よりも大きい
x < y	x は y よりも小さい
x >= y	x は y と等しいか大きい
x <= y	x は y と等しいか小さい
x in y	**x という要素 が y に存在する**
x not in y	**x という要素 が y に存在しない**

下の例は、1,000 円の買い物に対し、お客さんから預かった金額を input 関数で入力し、応答を返すプログラム。



コーディング例

```
x = input('合計は 1,000 円です。いくら出しますか？ :')
if not x.isdecimal():
    print('数値を入力してください。')
elif int(x) == 1000 :
    print('ちょうどお預かりします。ありがとうございました。')
```

```
elif int(x) > 1000:
    print('{}円のお返しです。ありがとうございました。'.format(int(x)-1000))
elif int(x) < 1000:
    print('お金が足りません。')
```

2.9　繰り返し（while、for）

- while を使った繰り返し処理

基本書式
while 条件式 : 　　条件式が真だった場合の命令

- for を使った繰り返し処理

基本書式
for 変数名 in 配列などのオブジェクト :　（次章で説明） 　　命令

- break や continue

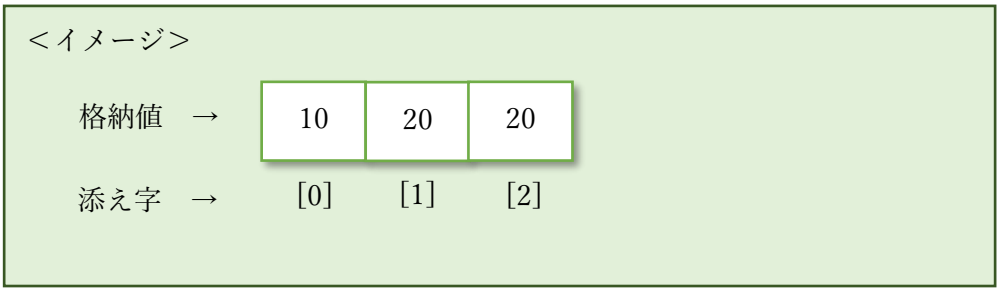
基本書式
while 条件式 : 　　: 　　if 条件式 1 :　（条件式 1 を満たす場合は while を抜ける） 　　　break 　　elif 条件式 2 :　（条件式 2 を満たす場合は「処理」を実行せず、while 文頭へ） 　　　continue 　　処理

3　配列（リスト、ディクショナリ）

python では配列（コレクション）は、リスト、ディクショナリ、タプル、セットの４つがある。
タプル、セットは使う機会が少ないので、ここではリストとディクショナリについて説明する。

3.1　リスト

リストは一般的な配列と同様、添え字を使ってデータを格納、抽出をする。

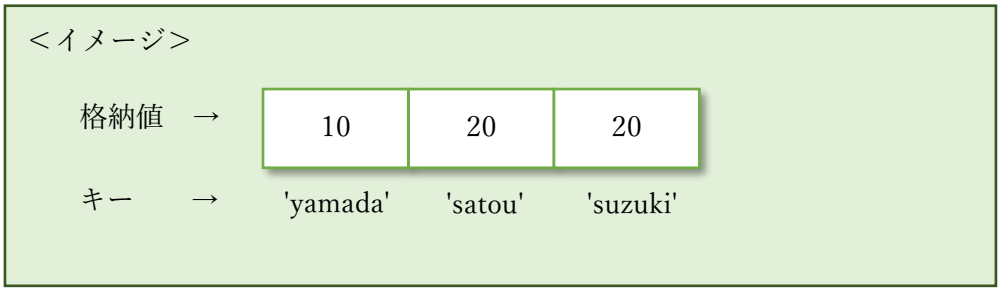


項目名	コーディング例	実行結果
リストの定義	scorelist = [10 , 20 , 20]	
全抽出	scorelist = [10 , 20 , 20] print(scorelist)	[10 , 20 , 20]
１つ抽出	scorelist = [10 , 20 , 20] print(scorelist[0])	10
範囲指定して抽出（スライス）	scorelist = [10 , 20 , 20] print(scorelist[0:1])	[10] 添え字が 0 以上 1 未満
合計を抽出	scorelist = [10 , 20 , 20] sum(scorelist)	50
要素の追加	scorelist = [10 , 20 , 20] scorelist.append(1) print(scorelist)	[10, 20, 20, 1] 末尾に追加される
位置を指定して削除	scorelist = [10 , 20 , 30] del scorelist [1] print(scorelist)	[10, 30]
値を指定して削除	scorelist = [10 , 20 , 10 , 20 , 5] scorelist.remove(20) print(scorelist)	[10, 10, 20, 5] 同じ値がある場合、最初の方が削除

3.2　ディクショナリ

リストは添え字を指定してデータを抽出するが、ディクショナリはキーを割り当てて、キーでデータを指定できる。

＜イメージ＞



項目名	コーディング例	実行結果
ディクショナリの定義	scorelist = {'yamada':10 , 'suzuki':20 , 'satou':20}	
全抽出	scorelist = {'yamada':10 , 'suzuki':20 , 'satou':20} print(scorelist)	{'yamada': 10, 'suzuki': 20, 'satou': 20}
１つ抽出	scorelist = {'yamada':10 , 'suzuki':20 , 'satou':20} print(scorelist['yamada'])	10
要素の追加	scorelist = {'yamada':10} scorelist['suzuki'] = 20　キーを指定して代入 print(scorelist)	{'yamada': 10, 'suzuki': 20}
キーを指定して削除	scorelist = {'yamada':10 , 'suzuki':20 , 'satou':20} del scorelist['suzuki'] print(scorelist)	{'yamada': 10, 'satou': 20}

3.3　ディクショナリからリストへの変換

コーディング例	実行結果
scorelist1 = {'yamada':10 , 'suzuki':20 , 'satou':20} scorelist2 = list(scorelist1) print(scorelist2)	['yamada', 'suzuki', 'satou']

3.4　リストのコピー（代入）は参照コピー

「リスト=リスト」のような書き方でリストをコピーすると、参照コピーとなるので注意すること。

次の例では、「list2 = list1」とした後に list1 の値を書き換えると、list2 の値も連動して変わっている。

コーディング例	実行結果
list1 = [1,2] list2 = [3,4] list2 = list1 list1[0],list1[1] = 9,8 print(list2[0]) print(list2[1])	9 8

4　関数

4.1　関数の定義

構文は次のとおり。

基本構文
def 関数名(引数１, 引数２, ・・・): 　　処理 　　return 戻り値

関数の中の処理はインデント

4.2　関数での変数のスコープ

関数の外で定義された変数は、グローバル変数なので、関数内でも使用することができる。

以下の例では、変数「name」は関数「proc_test」内でも使用できている。

コーディング例	実行結果
name = 'AA' proc_test() #関数「proc_test」 def proc_test(): 　　print(name)	AA

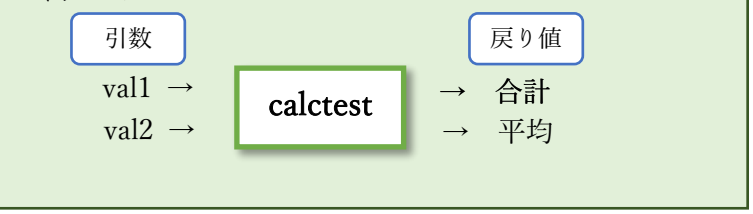
なお、関数内で定義された変数は、ローカル変数なので、関数の外では使用できない。

4.3　戻り値を配列（タプル）に返す

戻り値をタプルにすることで、複数の値を返すことができる。

下の例は、引数２つの値の合計と平均を戻り値としている。

＜イメージ＞



コーディング例	実行結果
(res_sum , res_avg) = calctest(10 , 20) print('合計は{} 平均は{}'.format(res_sum , res_avg)) #-------------------- #関数「calctest」 #-------------------- def calctest(val1 , val2): 　　return val1 + val2 , (val1 + val2) / 2	合計は 30 平均は 15.0

5　クラス

python でオリジナルクラスを定義する場合の注意事項などを説明する。

5.1　クラス定義

オリジナルクラスを定義する場合の構文は以下のとおり。

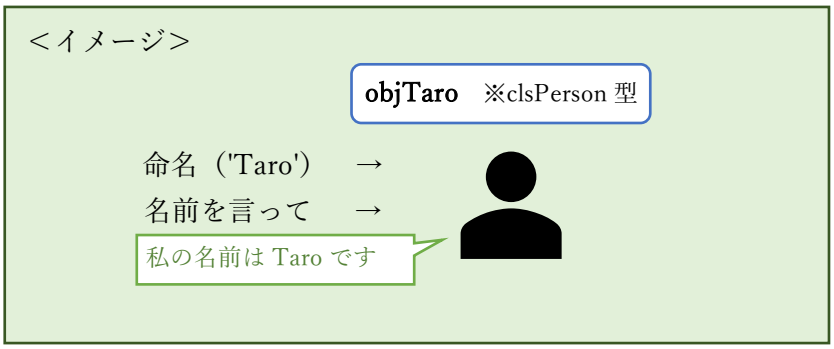
例

```
class クラス名
    プロパティ
    def メソッド:
        処理
```

5.2　クラスの注意事項や暗黙のルール

- ✓　メソッド内でプロパティを使用する場合は「self.変数名」とする。
- ✓　メソッドの第 1 引数は self を指定する必要がある。
- ✓　コンストラクタは「def __init__(self):」
- ✓　デストラクタは「def __del__(self):」

次の例は人をイメージしたクラスで、命名する「NamingName」メソッドと、名前を言う（print する）「SayName」メソッドが用意されている。



例

```
class clsPerson:
    myname = str()
    def __init__(self):
        self.myname = ''
    def NamingName(self , NameVal):
        self.myname = NameVal
    def SayName(self):
        if self.myname == '':
            print('名前はまだありません。')
        else:
            print('私の名前は{}です。'.format(self.myname))

objTaro = clsPerson()
objTaro.SayName()
objTaro.NamingName('Taro')
objTaro.SayName()
```

self で指定すること

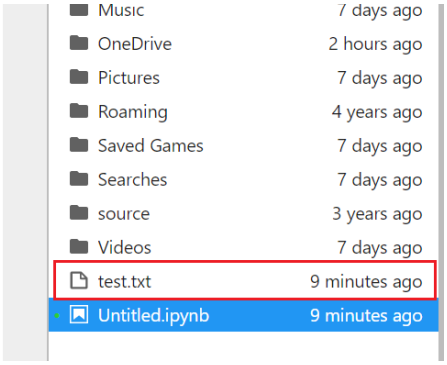
6　ファイル出力

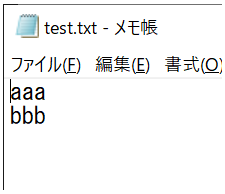
例

```
myfile = open('test.txt' , 'a')
myfile.write('aaa¥n')
myfile.write('bbb¥n')
myfile.close()
```

'a'は追記モード

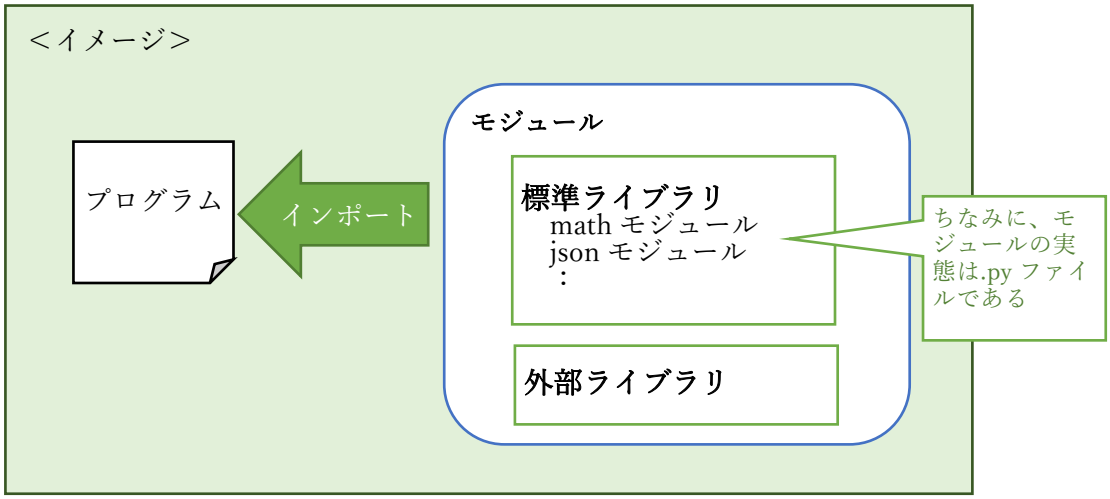
＜結果＞





7　モジュールを import

モジュール（Python 標準ライブラリや外部で作成されたライブラリ）を利用することができる。



次の例は、数学計算に関するモジュール「math」をインポートして、math モジュールの小数点以下切り捨て関数：floor を利用している。

コーディング例	実行結果
import math print(math.floor(1.5))	1

次のようにインポートしたモジュール名の変更も可能。

コーディング例	実行結果
import math as mdlMath print(mdlMath.floor(1.5))	1

特定の関数のみを利用する場合は以下のようにする。

コーディング例	実行結果
from math import floor print(floor(1.5))	1

以上