

Computer Vision

# THỊ GIÁC MÁY TÍNH

ThS. Huỳnh Minh Vũ

Khoa Kỹ thuật cơ khí

Trường Đại học Kỹ thuật – Công nghệ Cần Thơ

Email: [hmvu@ctuet.edu.vn](mailto:hmvu@ctuet.edu.vn)



# Tài liệu tham khảo

---

- [1] PGS. TS. Đỗ Toàn Năng, Xử lý ảnh, Viện CNTT, 2013.
- [2] PGS. TS. Nguyễn Quang Hoan, Xử lý ảnh, Học viện bưu chính viễn thông, 2006.
- [3] Nguyễn Văn Long, Ứng dụng xử lí ảnh trong thực tế với OpenCV.
- [4] Alexander Mordvintsev & Abid K, OpenCV-Python Tutorials Documentation, 2017.
- [5] Prateek Joshi, OpenCV with Python By Example, 2015.
- [6] Joe Minichino & Joseph Howse, Learning OpenCV 3 Computer Vision with Python (2nd Edition), 2015.
- [7] Rafael C. Gonzalez & Richard E. Woods, Digital Image Processing (3rd Edition), 2015.

# Tài liệu tham khảo

---

---

[8] Các trang Website:

<https://docs.python.org/3/tutorial/>

<https://www.tutorialspoint.com/python/index.htm>

<https://www.w3schools.com/python/default.asp>

<https://numpy.org/doc/1.19/reference/index.html>

<https://pandas.pydata.org/index.html>

[https://www.tutorialspoint.com/python\\_pandas/python\\_pandas\\_dataframe.htm](https://www.tutorialspoint.com/python_pandas/python_pandas_dataframe.htm)

<https://matplotlib.org/index.html>

# Hình thức đánh giá

---

---

1. Chuyên cần: 5%
2. Báo cáo giữa kì: 35%
3. Báo cáo cuối kì: 60%

# Nội dung môn học

---

---

**Chương 0: Ngôn ngữ lập trình Python và Thư viện**

**Chương 1: Tổng quan về xử lý ảnh và thị giác máy tính**

**Chương 2: Các phép xử lý đơn giản trong OpenCV**

**Chương 3: Xử lý nâng cao chất lượng ảnh**

**Chương 4: Xử lý hình thái ảnh**

**Chương 5: Các phương pháp phát hiện biên**

**Chương 6: Phân lớp ảnh (Image Classification)**

**Chương 7: Phát hiện đối tượng trong ảnh (Object Detection)**

**Chương 8: Phân đoạn ảnh (Image Segmentation)**

# Chương 0: Ngôn ngữ Python và Thư viện

---

---

**0.1 Ngôn ngữ lập trình Python**

**0.2 Thư viện NumPy**

**0.3 Thư viện Pandas**

**0.4 Thư viện Matplotlib**

# 0.1 Ngôn ngữ lập trình Python

---

0.1.1 Giới thiệu

0.1.2 Môi trường lập trình

0.1.3 Biến, kiểu dữ liệu cơ bản và hàm có sẵn

0.1.4 Kiểu dữ liệu Containers: list, tuple, set và dictionary

0.1.5 Xây dựng hàm (Function Construction)

0.1.6 Điều khiển luồng dữ liệu (Điều kiện IF)

0.1.7 Vòng lặp For (For Loop)

0.1.8 Vòng lặp While (While Loop)

0.1.9 Class

## 0.1.1 Giới thiệu ngôn ngữ lập trình Python

- Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991.
- Python là ngôn ngữ mã nguồn mở. Được xếp vào loại “ngôn ngữ kịch bản” (scripting programming language).
- Có khả năng tương tác với các module viết bằng ngôn ngữ lập trình khác. Có thể nhúng vào ứng dụng như một giao tiếp kịch bản (scripting interface)
- Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có cấu trúc rõ ràng, thuận tiện cho người mới học lập trình.

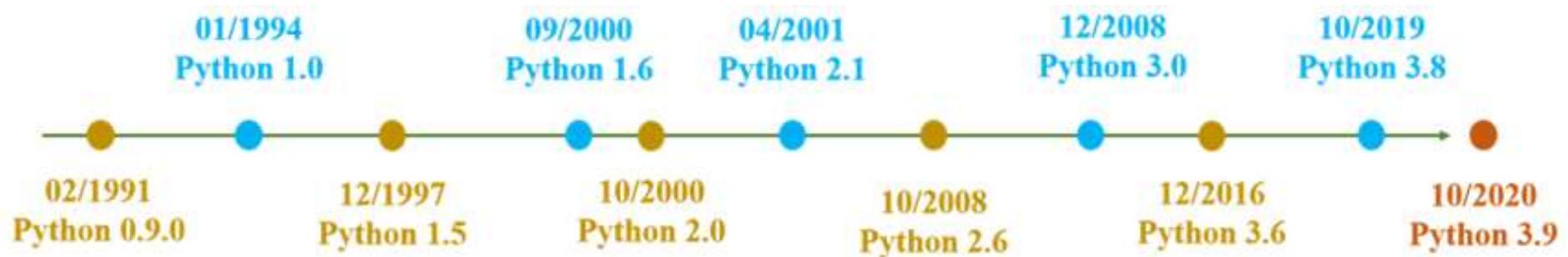
# 0.1.1 Giới thiệu ngôn ngữ lập trình Python

## Language Ranking (2020 – IEEE Spectrum)

Rank	Language	Type	Score
1	Python ▾	🌐💻⚙️	100.0
2	Java ▾	🌐📱💻	95.3
3	C ▾	📱💻⚙️	94.6
4	C++ ▾	📱💻⚙️	87.0
5	JavaScript ▾	🌐	79.5
6	R ▾	💻	78.6
7	Arduino ▾	⚙️	73.2
8	Go ▾	🌐💻	73.1
9	Swift ▾	📱💻	70.5
10	Matlab ▾	💻	68.4

# 0.1.1 Giới thiệu ngôn ngữ lập trình Python

## Lịch sử phát triển



Ý tưởng từ 1980s



Được đặt tên theo  
nhóm hài Monty Python



Bắt đầu cài đặt  
từ 12/1989



Guido van Rossum



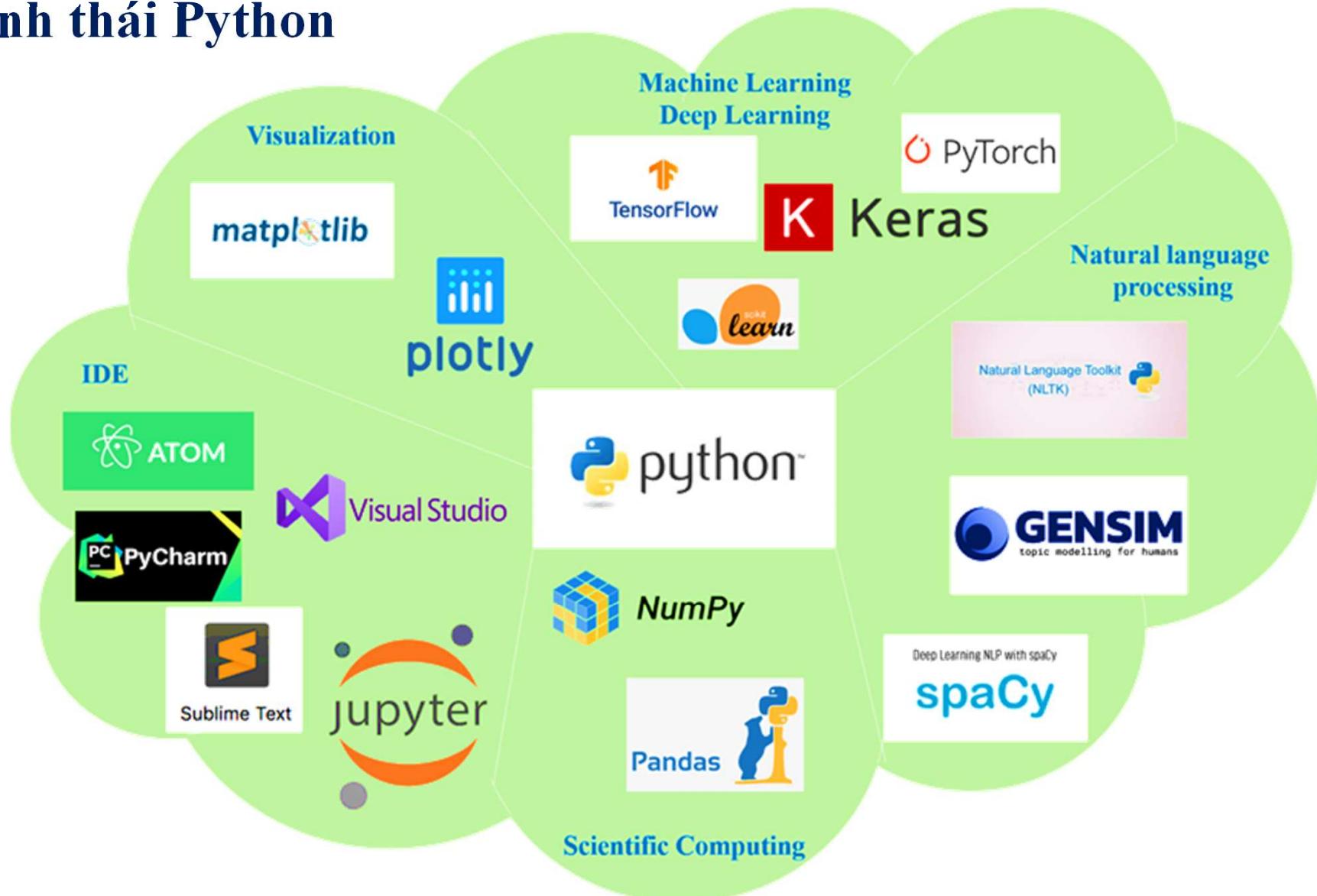
228,855 python packages  
(PyPI)



Hỗ trợ rất mạnh cho  
Data Science và Machine Learning

# 0.1.1 Giới thiệu ngôn ngữ lập trình Python

## Hệ sinh thái Python



## 0.1.2 Môi trường lập trình

Có 2 dạng môi trường để thực hiện ngôn ngữ lập trình Python:

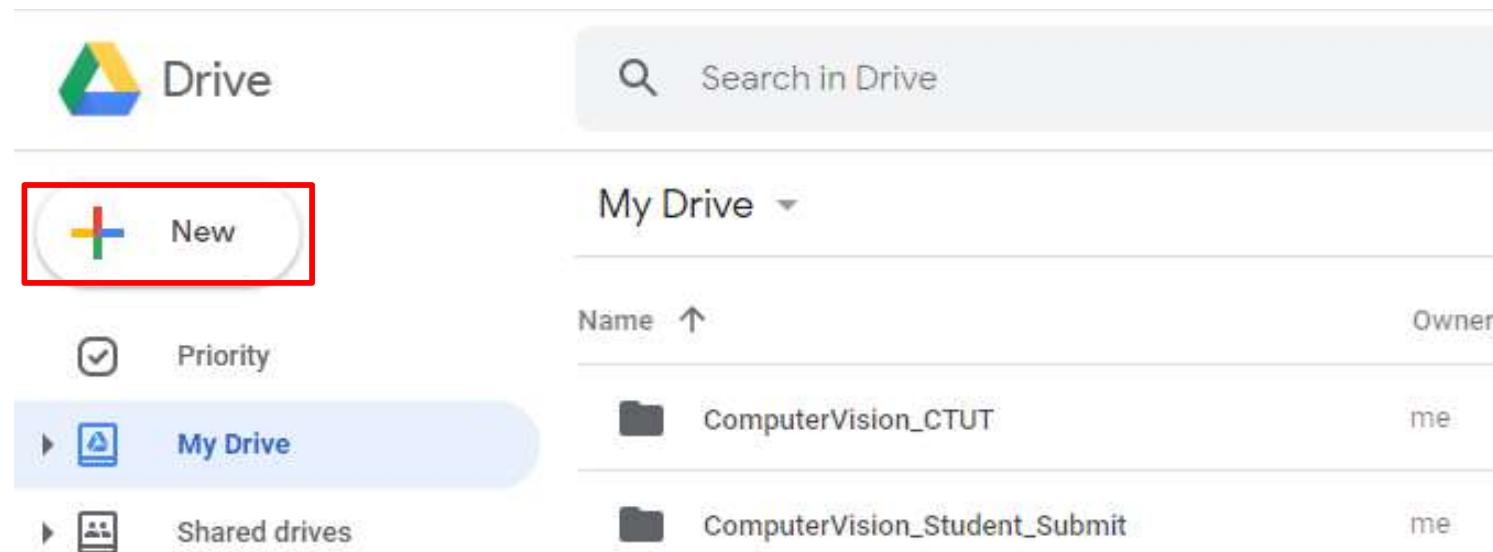
- Thực hiện chạy online bằng Google Colab. Môi trường này cho phép viết code python và chạy online, không cần cài gì trên máy cả nên sẽ đơn giản hơn và không phụ thuộc vào cấu hình máy tính.
- Thực hiện chạy offline trên máy tính local dùng Anaconda hoặc IDE là Spyder, Visual Studio, Jupyter Notebook hoặc Pycharm.

The logo for Google Colab, featuring the word "colab" in a bold, sans-serif font. The letters are primarily orange, with the "c" having a yellow-to-orange gradient. The "o" has a yellow-to-orange gradient. The "l" has a yellow-to-orange gradient. The "a" has a yellow-to-orange gradient. The "b" has a yellow-to-orange gradient.

## 0.1.2 Môi trường lập trình

### Cài đặt môi trường lập trình online Google Colab

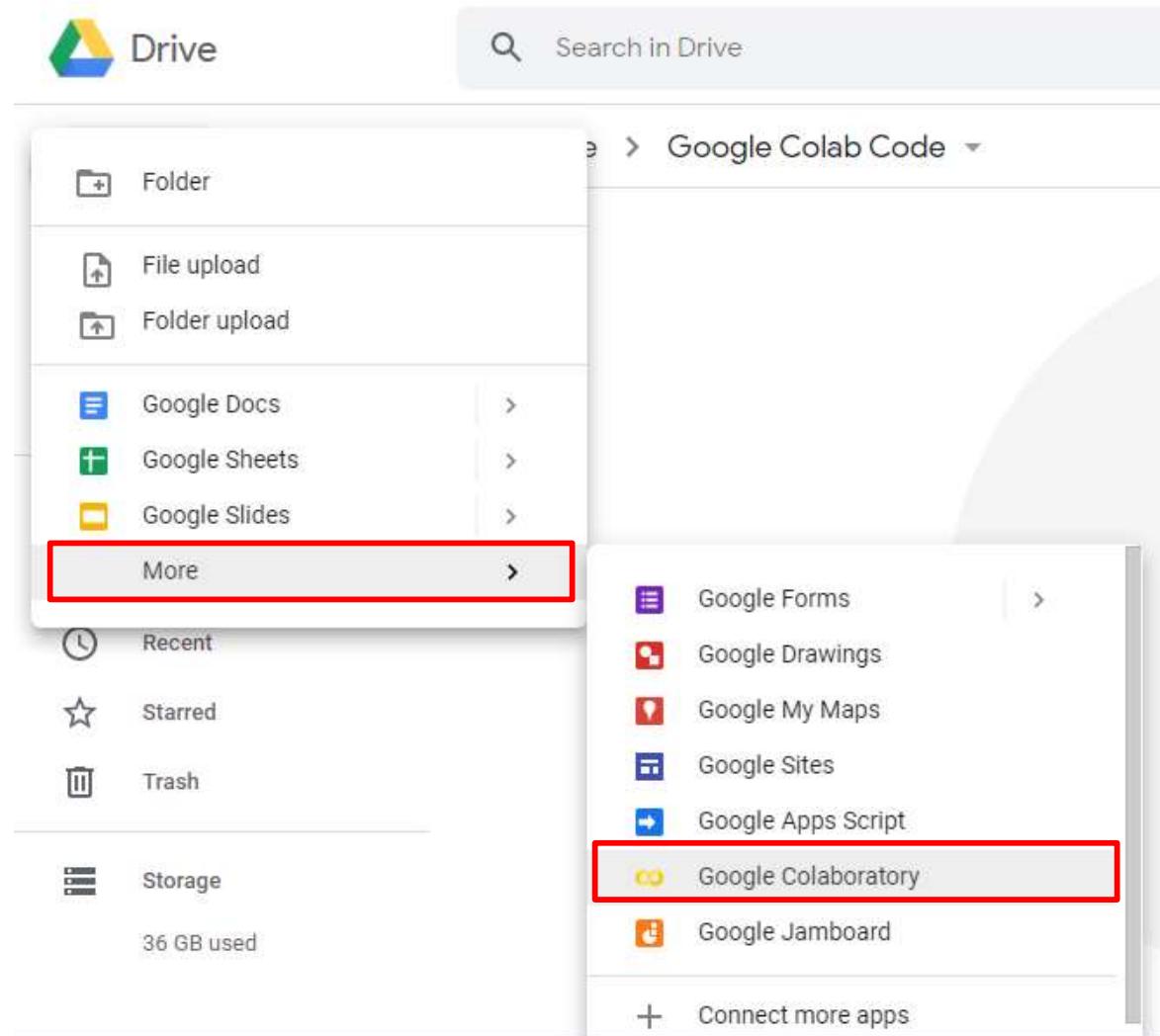
- Đầu tiên, vào Google Drive tạo thư mục để lưu trữ các file Colab



## 0.1.2 Môi trường lập trình

### Cài đặt môi trường lập trình online Google Colab

- Vào thư mục vừa tạo chọn New/More/Google Colaboratory



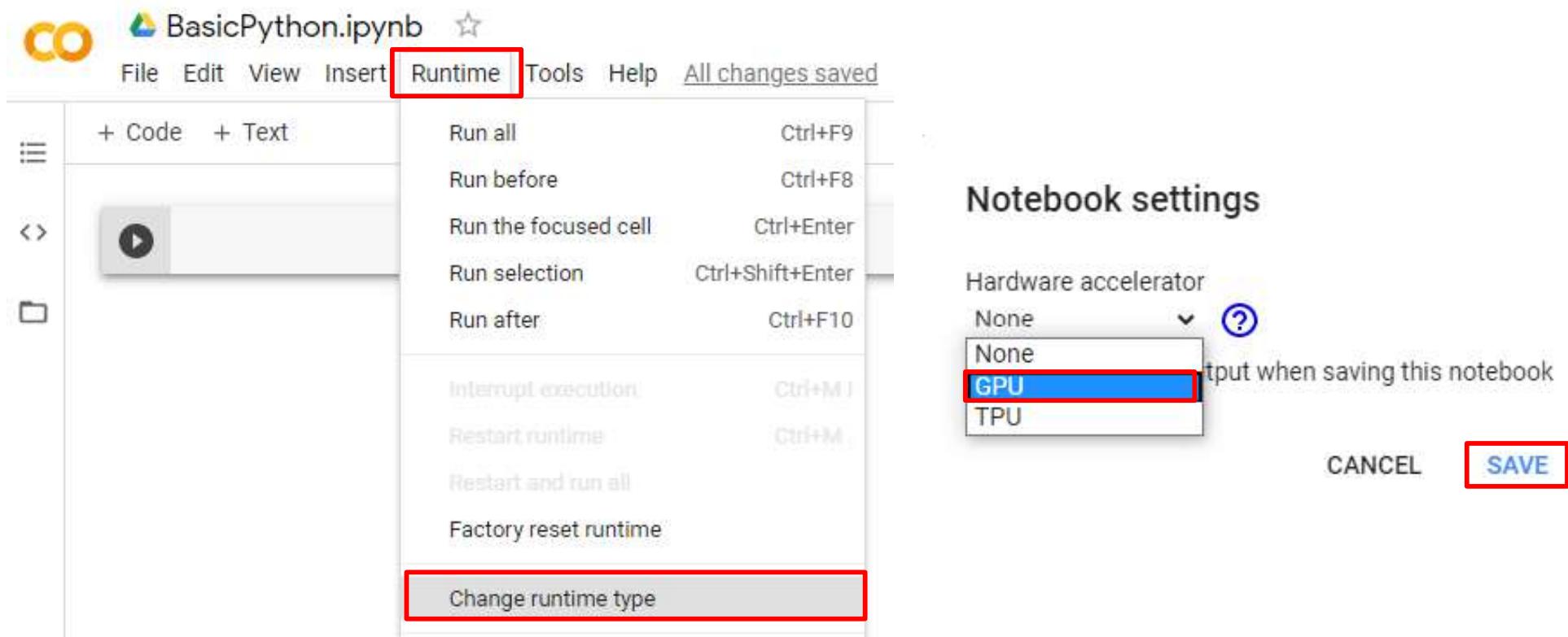
## 0.1.2 Môi trường lập trình

### Cài đặt môi trường lập trình online Google Colab

- Đổi tên file cho hợp lý



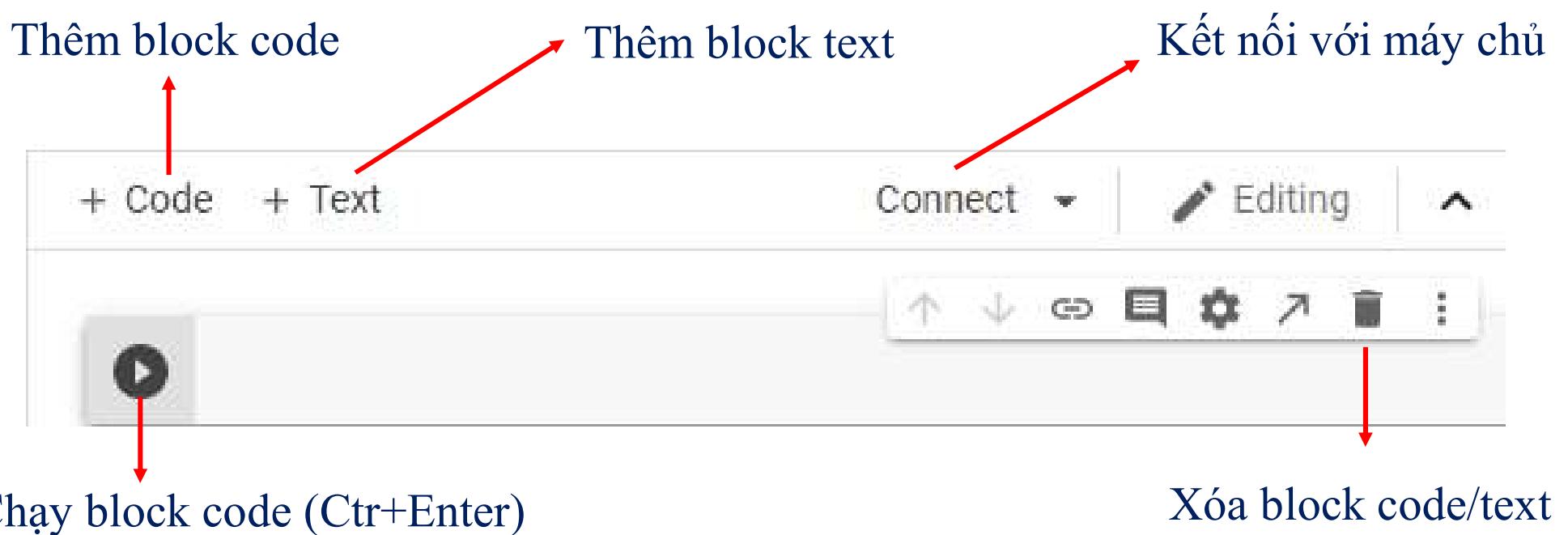
- Chọn GPU cho máy ảo



## 0.1.2 Môi trường lập trình

### Cài đặt môi trường lập trình online Google Colab

- Thao tác cơ bản



## 0.1.2 Môi trường lập trình

### Cài đặt môi trường lập trình online Google Colab

- Liên kết Google Drive với Colab

```
from google.colab import drive  
drive.mount('/content/gdrive')
```

The screenshot shows a Google Colab notebook titled "BasicPython.ipynb". The code cell contains the following Python code:

```
from google.colab import drive  
drive.mount('/content/gdrive')
```

Below the code cell, there is a message: "... Go to this URL in a browser: [https://accounts.google.com/o/oauth2/auth?client\\_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect\\_uri=https://colab.research.google.com/notebooks/api/oauthcallback&response\\_type=code&scope=https://www.googleapis.com/auth/drive](https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=https://colab.research.google.com/notebooks/api/oauthcallback&response_type=code&scope=https://www.googleapis.com/auth/drive)". A red box highlights this URL.

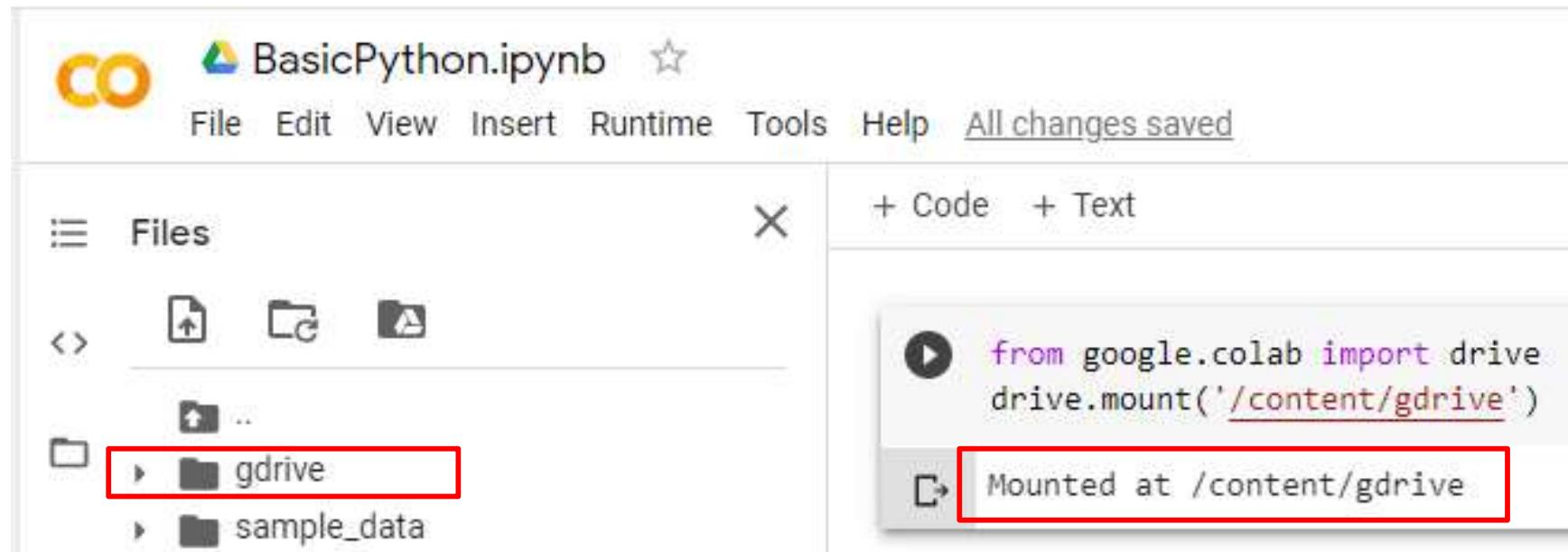
Below the URL, there is a text input field with the placeholder "Enter your authorization code:" and a red arrow pointing to it. A red box highlights this entire area.

At the bottom of the notebook, there is a message: "Please copy this code, switch to your application and paste it there:". Below this message, there is a code snippet: "4/4AFzNgtLFw71DMm2kEofJFNqhk7Dfcj7i0YPz\_M6y99YtM" followed by a red square icon. A red box highlights this entire area.

## 0.1.2 Môi trường lập trình

### Cài đặt môi trường lập trình online Google Colab

- Kiểm tra liên kết



The screenshot shows the Google Colab interface. At the top, there's a toolbar with File, Edit, View, Insert, Runtime, Tools, Help, and a status message "All changes saved". Below the toolbar is a "Files" sidebar with icons for upload, download, and new file. A folder named "gdrive" is highlighted with a red box. Another folder named "sample\_data" is also listed. The main area is a code editor with two code cells. The first cell contains the Python code: 

```
from google.colab import drive  
drive.mount('/content/gdrive')
```

. The second cell shows the output: 

```
Mounted at /content/gdrive
```

, which is also highlighted with a red box.

## 0.1.2 Môi trường lập trình

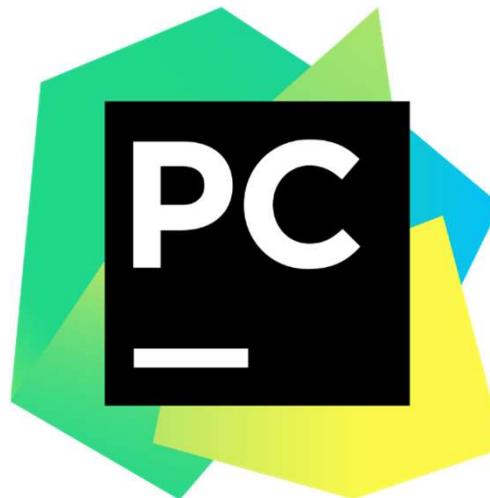
### Cài đặt môi trường lập trình trên máy tính local

Cài đặt Python (Bài giảng sử dụng bản 3.8):

<https://www.python.org/downloads/windows/>

Cài đặt môi trường Pycharm (Bản Comunity):

<https://www.jetbrains.com/pycharm/download/#section=windows>



## 0.1.2 Môi trường lập trình

### Cài đặt ngôn ngữ Python

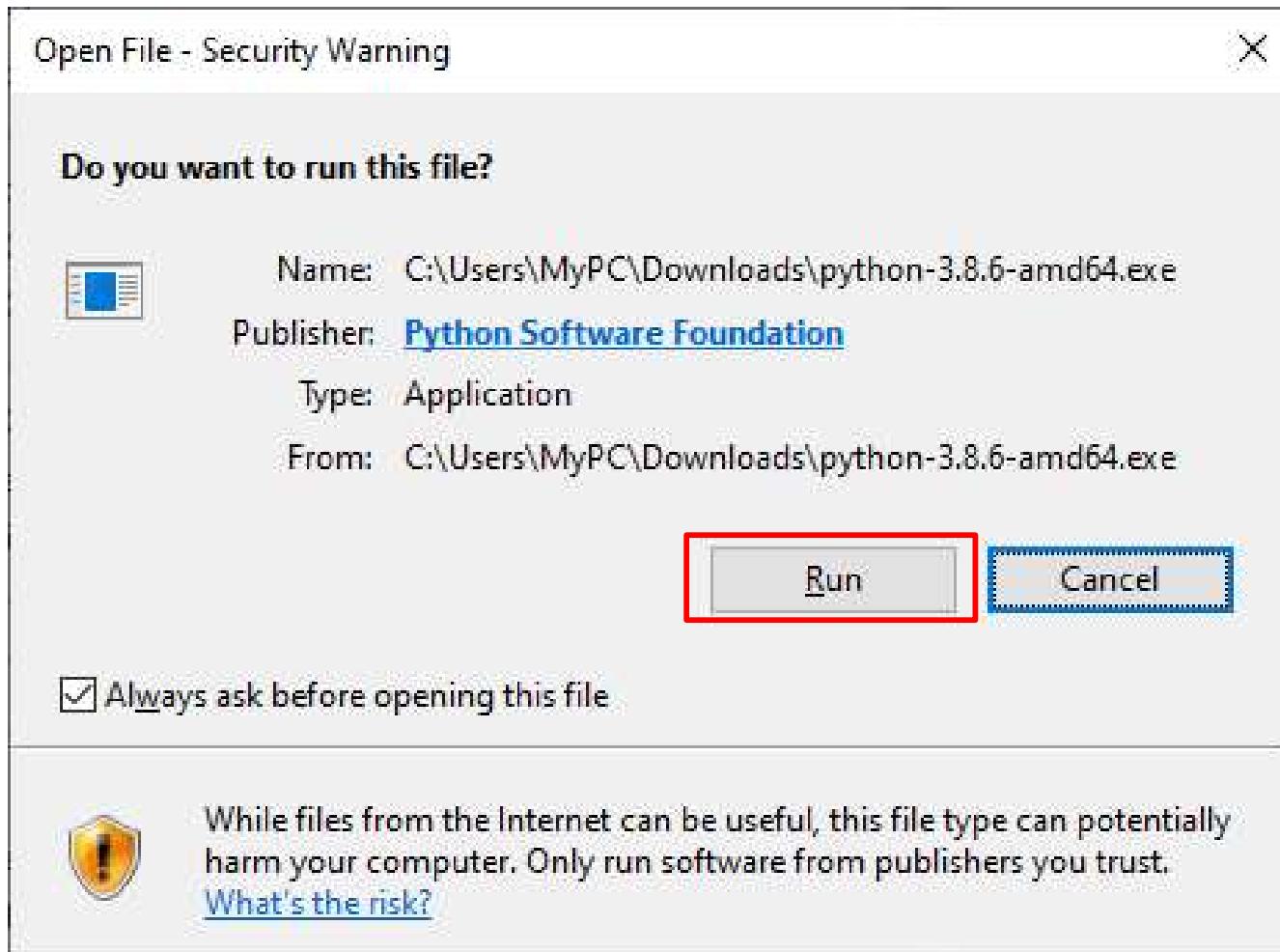
Note that Python 3.8.6 cannot be used on Windows XP or earlier.

- Download [Windows help file](#)
- Download [Windows x86-64 embeddable zip file](#)
- Download [Windows x86-64 executable installer](#) Window 64 bit
- Download [Windows x86-64 web-based installer](#)
- Download [Windows x86 embeddable zip file](#)
- Download [Windows x86 executable installer](#) Window 32 bit
- Download [Windows x86 web-based installer](#)



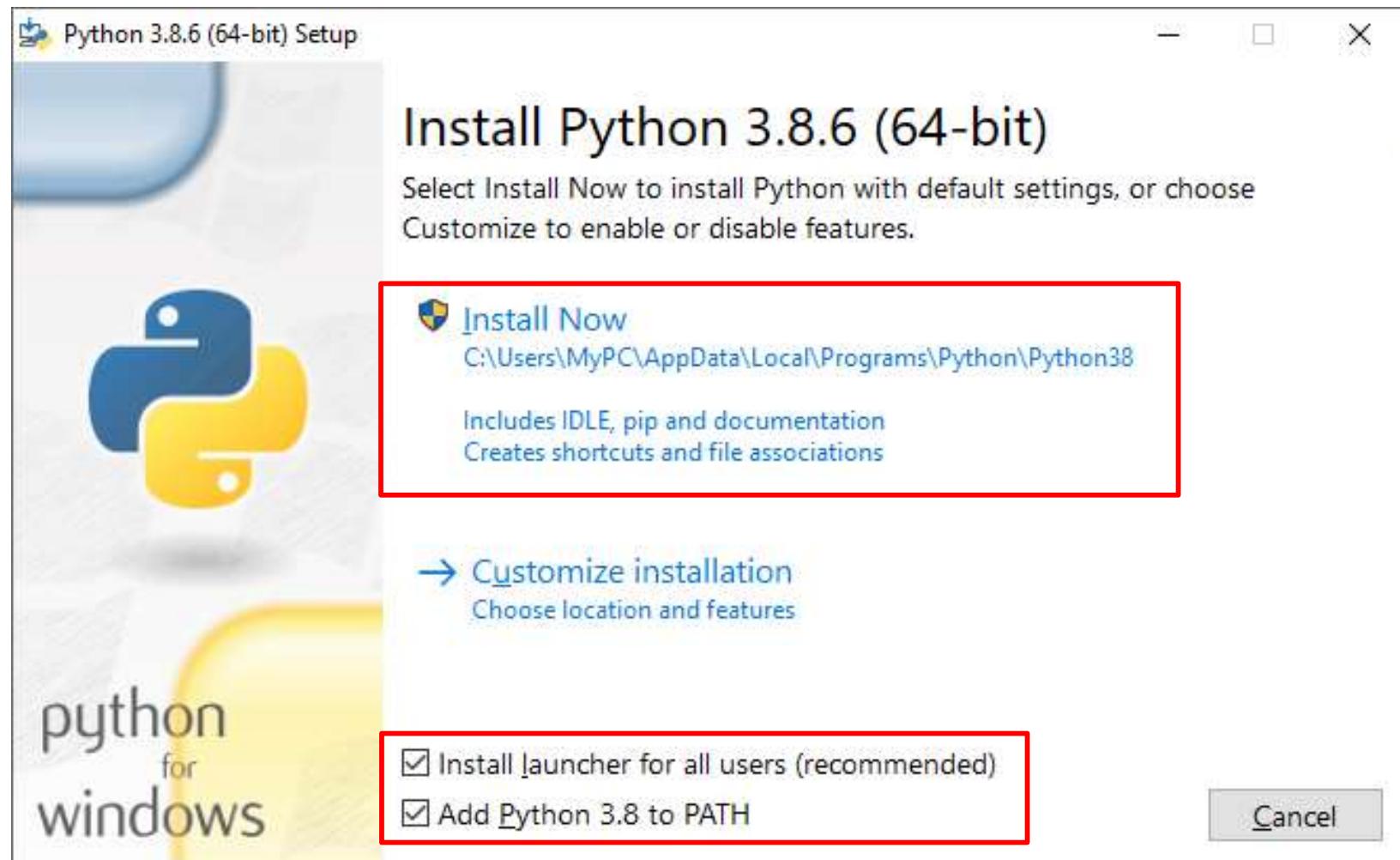
## 0.1.2 Môi trường lập trình

### Cài đặt ngôn ngữ Python



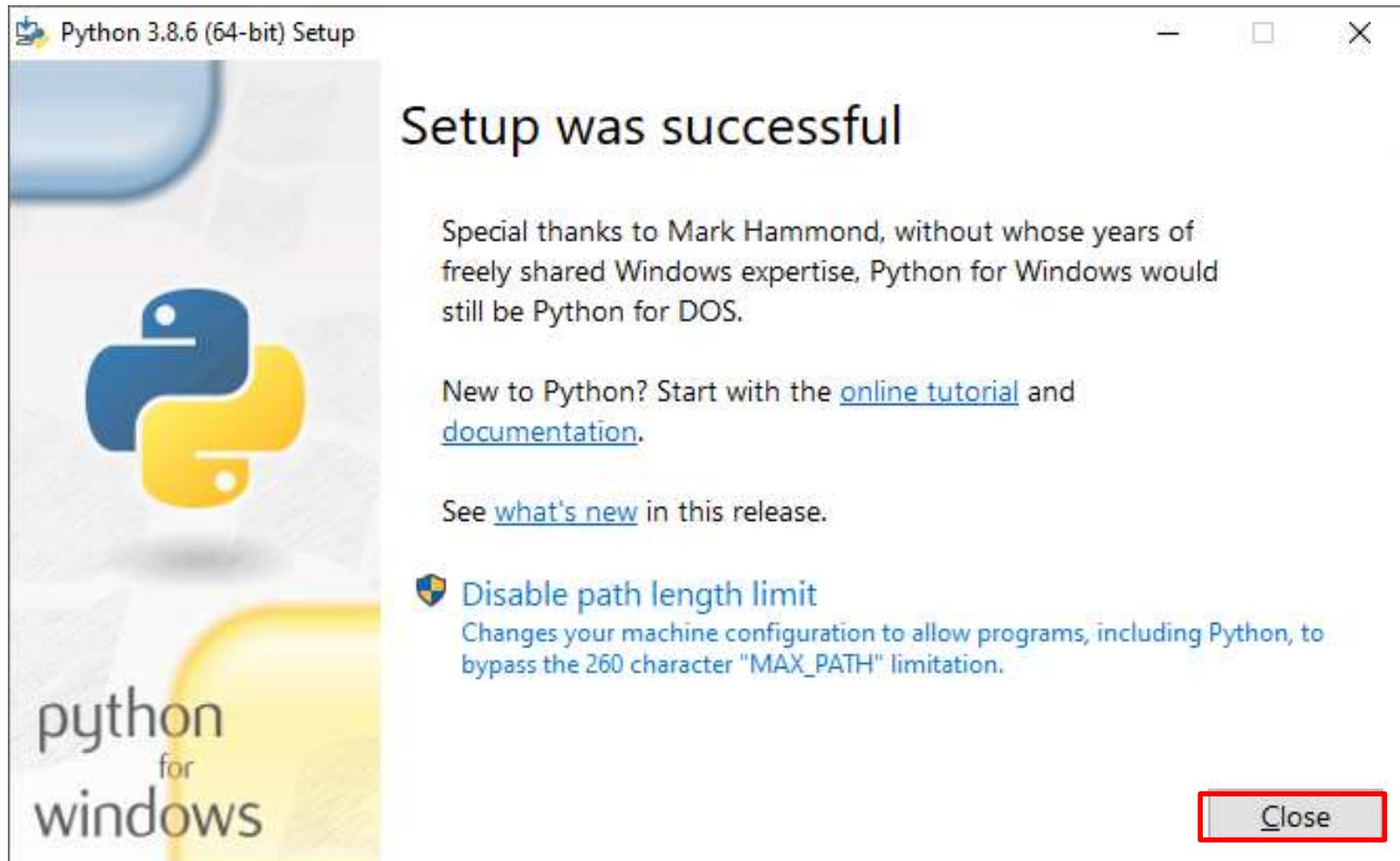
## 0.1.2 Môi trường lập trình

### Cài đặt ngôn ngữ Python



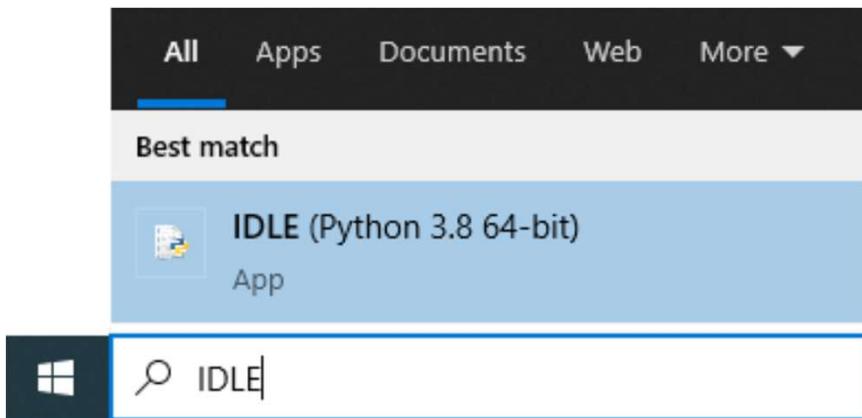
## 0.1.2 Môi trường lập trình

### Cài đặt ngôn ngữ Python



# 0.1.2 Môi trường lập trình

## Cài đặt ngôn ngữ Python



```
>>> print('Hello Python')
Hello Python
>>>
```

A screenshot of the Python 3.8.6 Shell window. The title bar says 'Python 3.8.6 Shell'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area displays the Python version information and a prompt: 'Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AMD64)] on win32'. It also says 'Type "help", "copyright", "credits" or "license()" " for more information.' followed by a '>>>' prompt. At the bottom right, it shows 'Ln: 3 Col: 4'.

# 0.1.2 Môi trường lập trình

## Cài đặt môi trường lập trình PyCharm

### Download PyCharm

Windows

macOS

Linux

#### Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

Download

Free trial

#### Community

For pure Python development.

Download

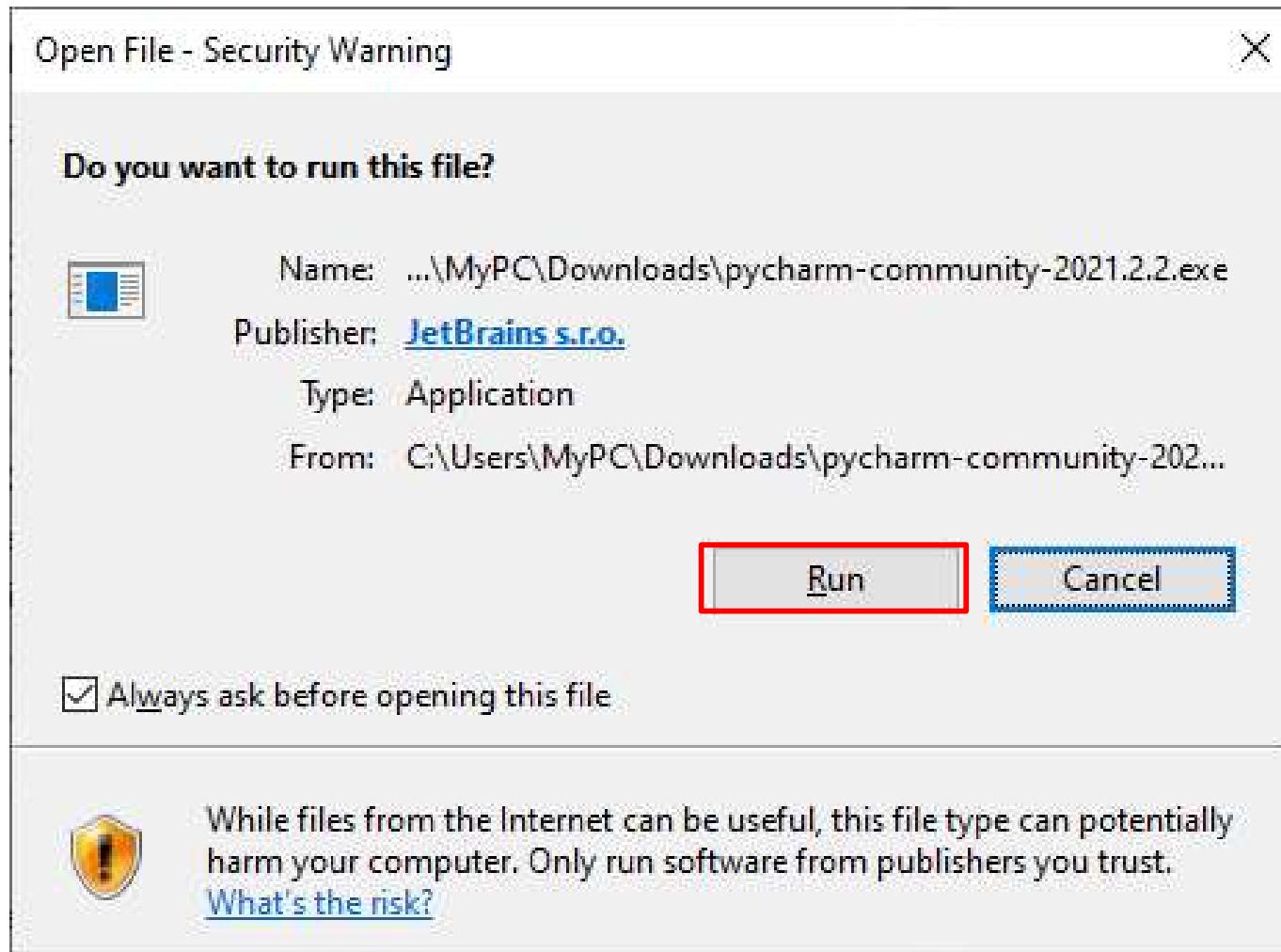
Free, built on open-source



pycharm-community-2021.2.2.exe

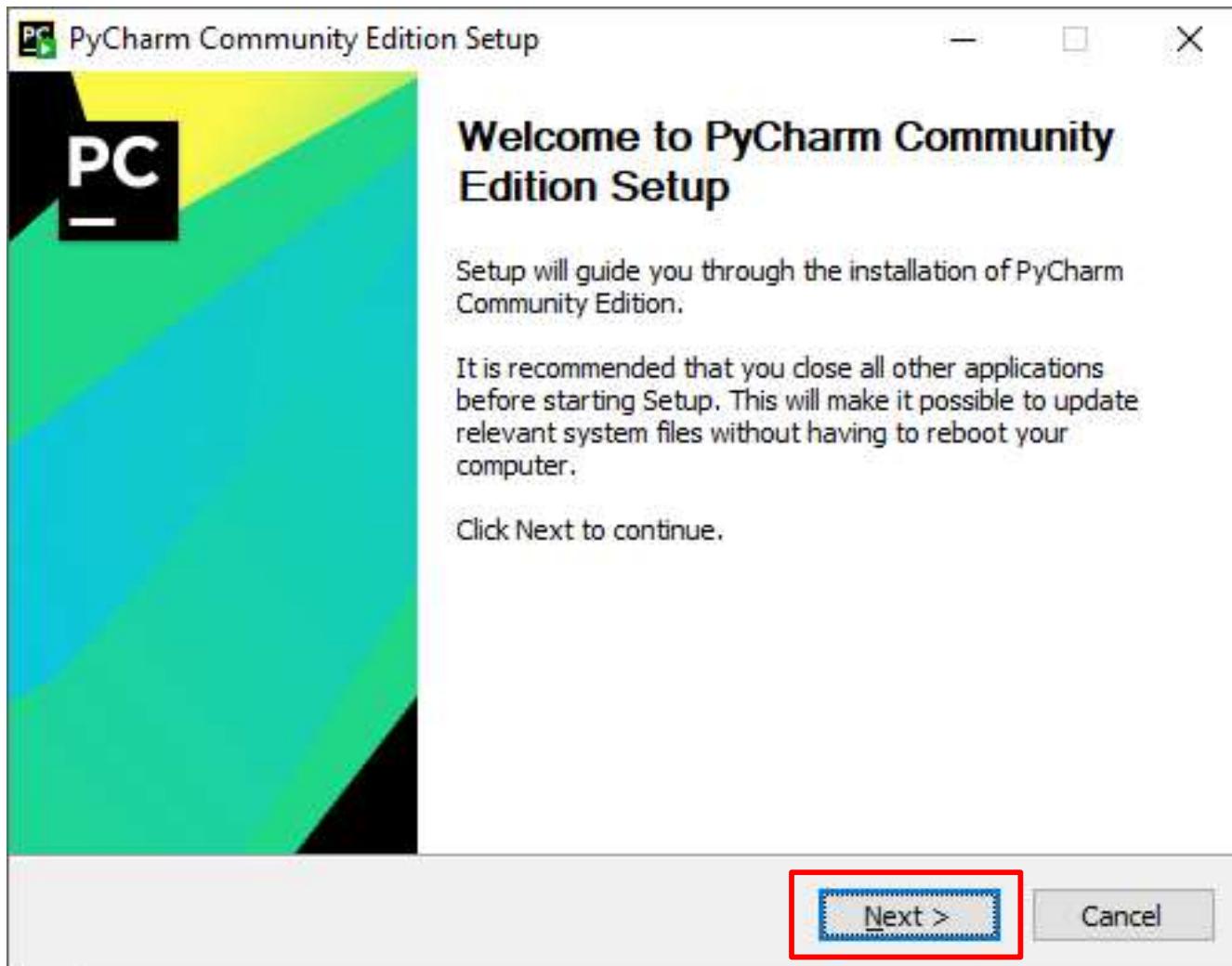
# 0.1.2 Môi trường lập trình

## Cài đặt môi trường lập trình PyCharm



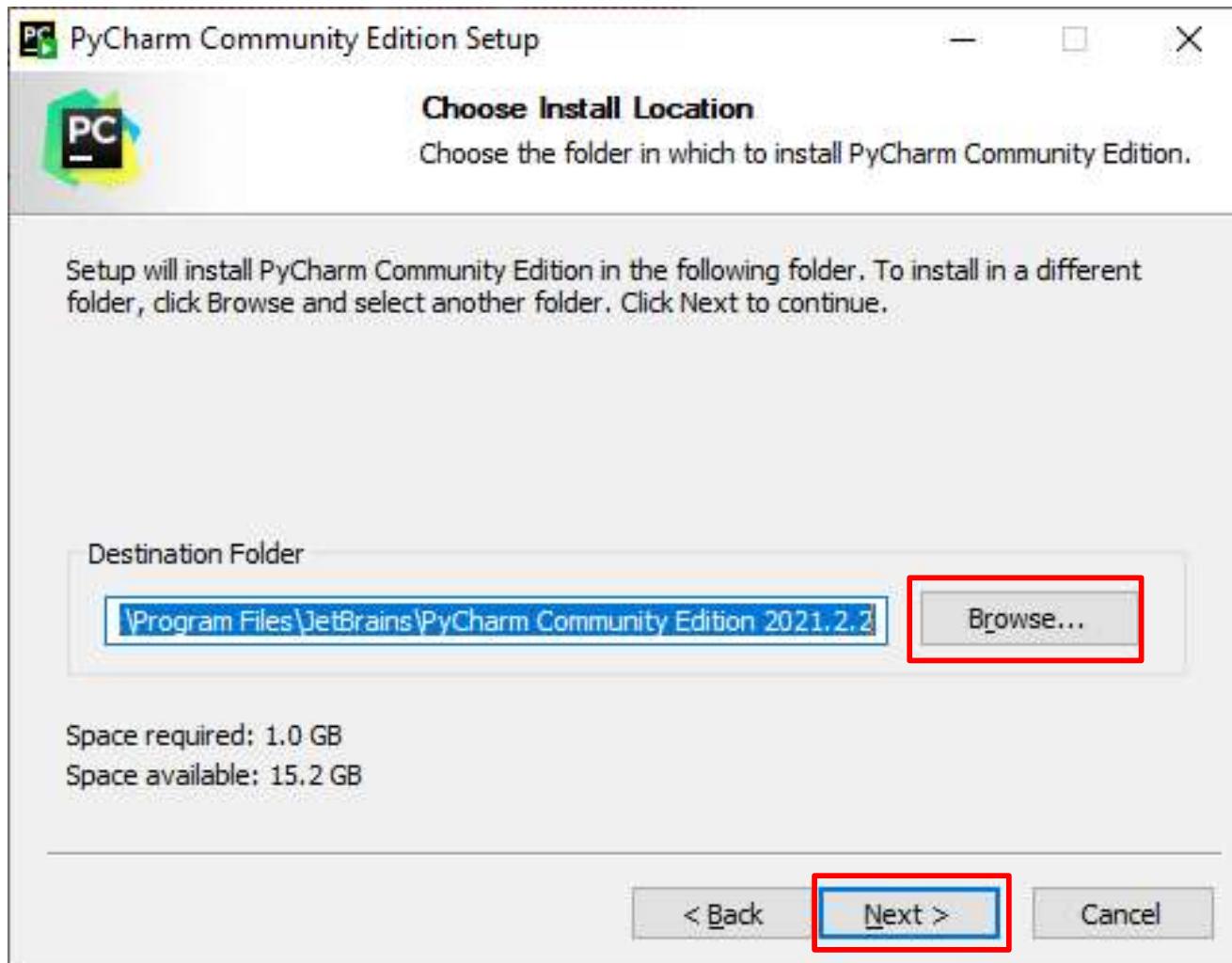
## 0.1.2 Môi trường lập trình

### Cài đặt môi trường lập trình PyCharm



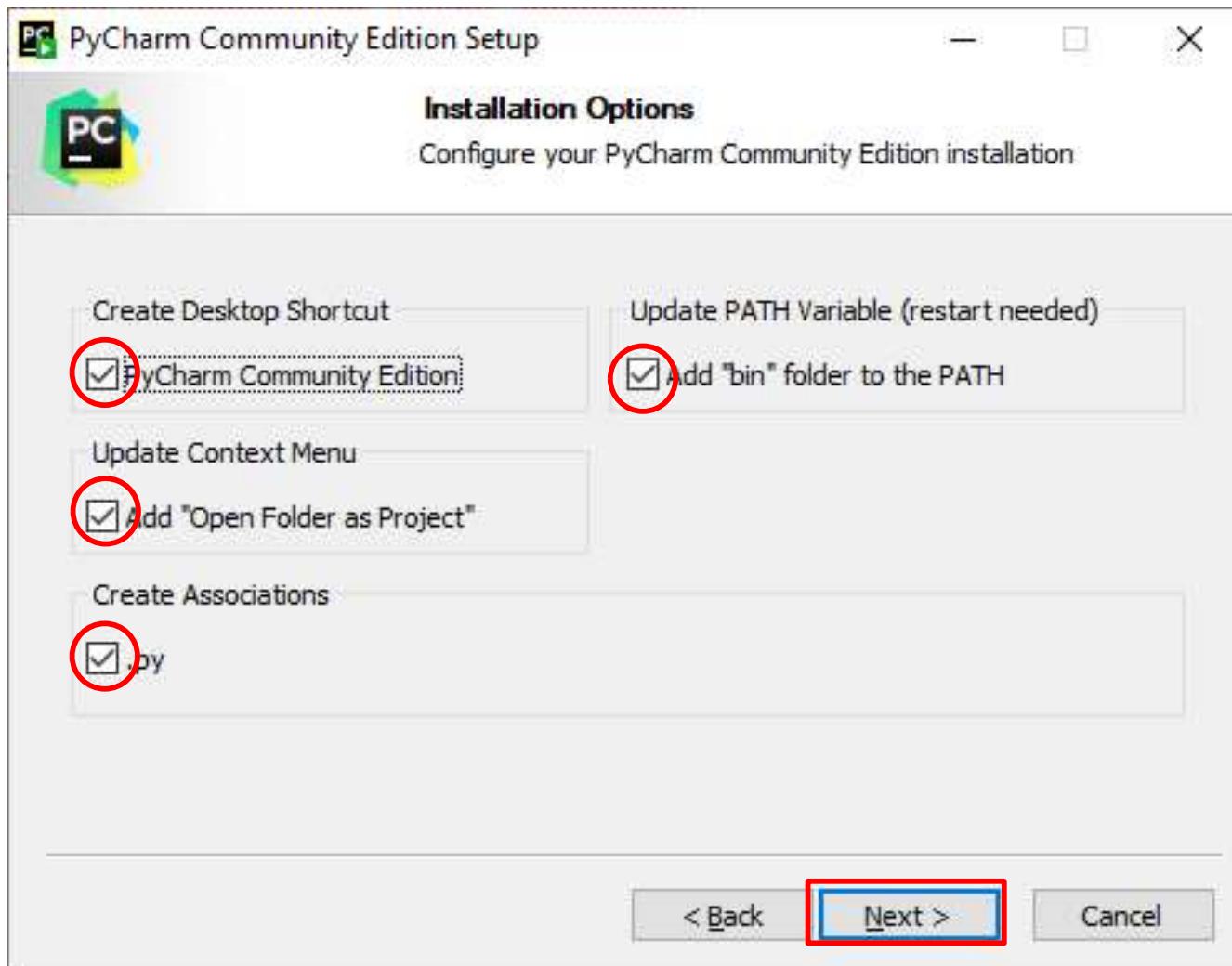
## 0.1.2 Môi trường lập trình

### Cài đặt môi trường lập trình PyCharm



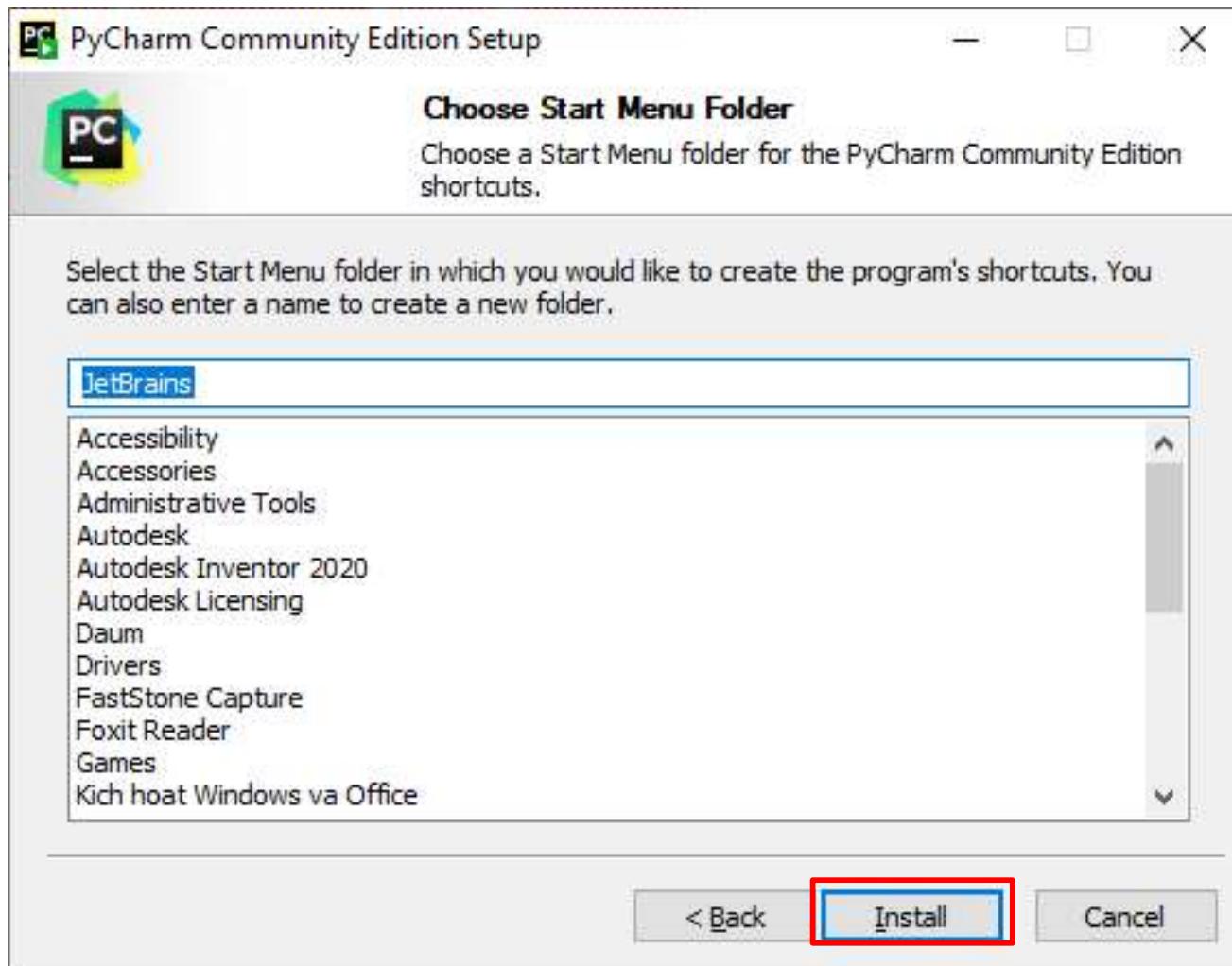
## 0.1.2 Môi trường lập trình

### Cài đặt môi trường lập trình PyCharm



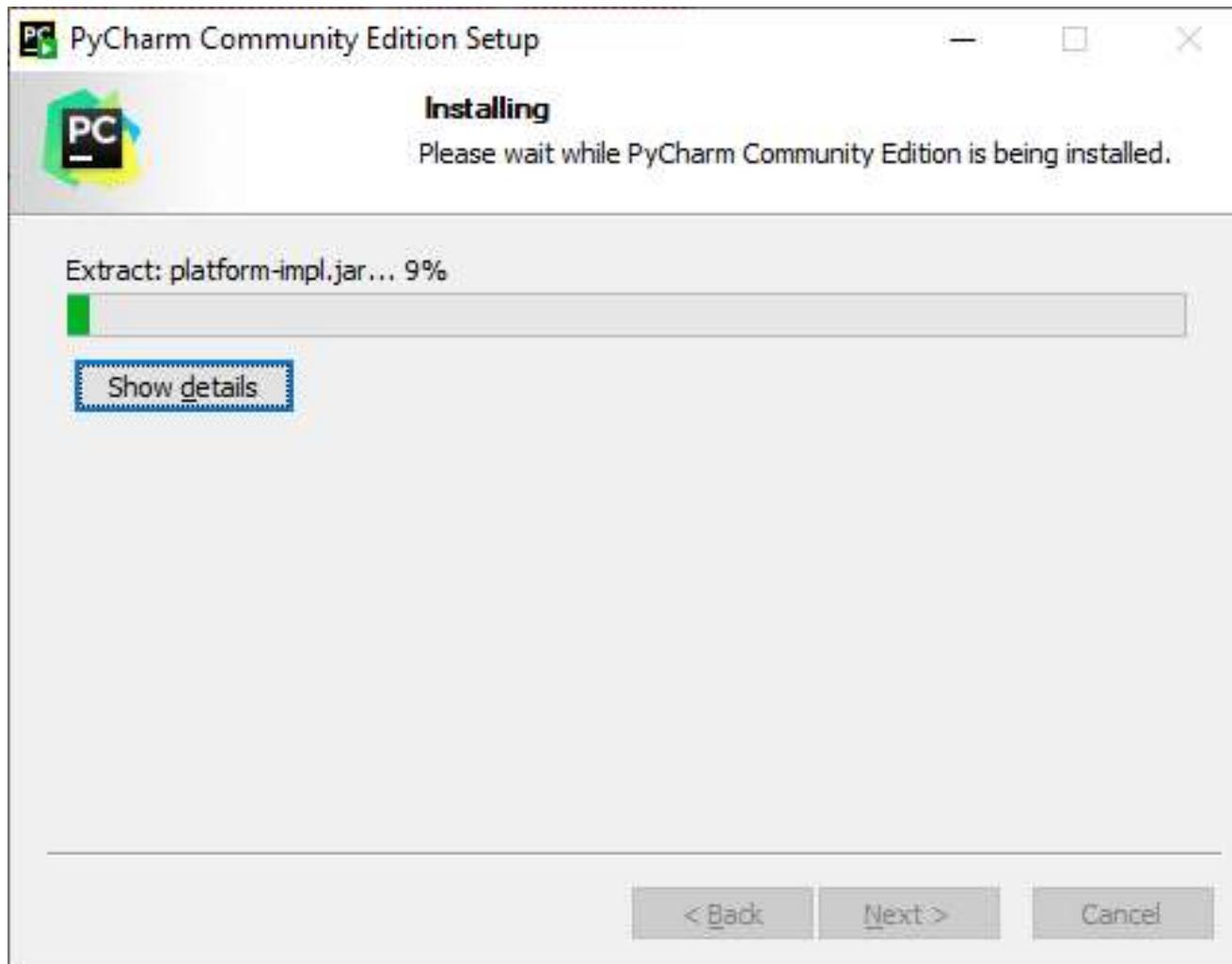
## 0.1.2 Môi trường lập trình

### Cài đặt môi trường lập trình PyCharm



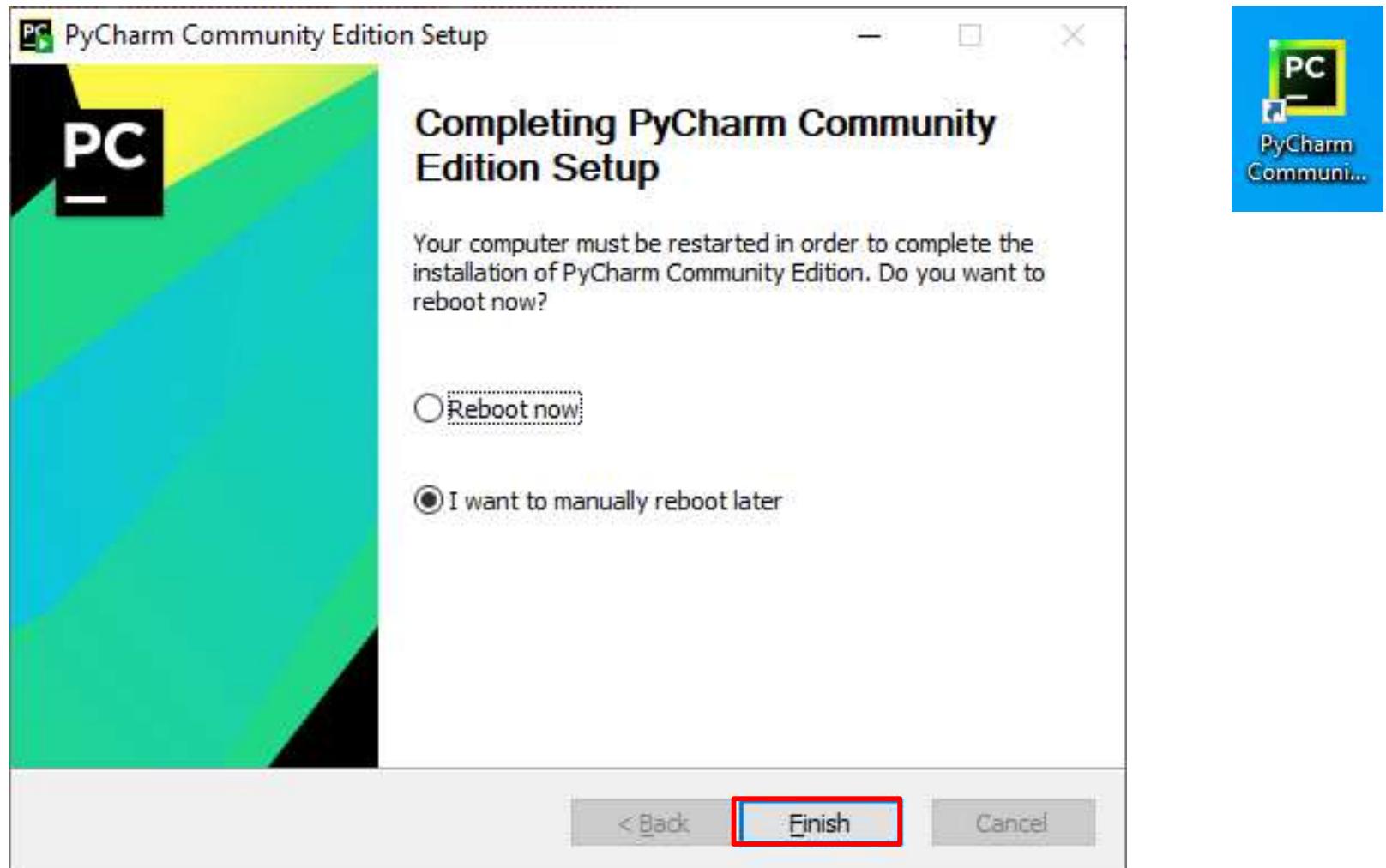
## 0.1.2 Môi trường lập trình

### Cài đặt môi trường lập trình PyCharm



## 0.1.2 Môi trường lập trình

### Cài đặt môi trường lập trình PyCharm



# 0.1.3 Biến, kiểu dữ liệu cơ bản và hàm có sẵn

## Biến (Variable)

variable\_name = variable\_value

### variable\_name

- Nên có ý nghĩa
- Không được bắt đầu với số
- Không sử dụng keywords

and del from not while  
as elif global or with  
assert else if pass yield  
break except import print  
class exec in raise  
continue finally is return  
def for lambda try

### variable\_value

- Integer
- Float
- String
- Boolean...

Kiểu dữ liệu cơ bản	Integer	1, 2, 3, -1, 0
	Float	1.0, 2.5, 3.1, -4.15
	String	“Tom”, “Lena”, ‘Hello’, ‘Xin chao’
	Boolean	True, False

# 0.1.3 Biến, kiểu dữ liệu cơ bản và hàm có sẵn

## Biến (Variable)

variable\_name = variable\_value



```
#Variable  
#Tạo biến với kiểu dữ liệu là Integer  
week = 7  
month = 30  
year = 365  
#Tạo biến với kiểu dữ liệu là Float  
distance = 20.5  
number = -15.3  
temp = 36.5  
#Tạo biến với kiểu dữ liệu là String  
fullname = "Huynh Minh Vũ"  
firstname = 'Vu'  
lastname = "Huynh"  
#Tạo biến với kiểu dữ liệu là Boolean  
techer = True  
student = False
```

# 0.1.3 Biến, kiểu dữ liệu cơ bản và hàm có sẵn

## Hàm có sẵn (Build-in Function)

### print(parameter)

```
▶ week = 7  
print(week)  
  
distance = 20.5  
print(distance)  
  
fullname = "Huynh Minh Vu"  
print(fullname)  
  
student = False  
print(student)
```

```
⇨ 7  
20.5  
Huynh Minh Vu  
False
```

### type(parameter)

```
▶ week = 7  
print(type(week))  
  
distance = 20.5  
print(type(distance))  
  
fullname = "Huynh Minh Vu"  
print(type(fullname))  
  
student = False  
print(type(student))
```

```
⇨ <class 'int'>  
<class 'float'>  
<class 'str'>  
<class 'bool'>
```

# 0.1.3 Biến, kiểu dữ liệu cơ bản và hàm có sẵn

## Hàm có sẵn (Build-in Function)

### input (string)

```
#Nhập thông tin từ bàn phím  
data = input("Vui lòng nhập tuổi của bạn!")  
print(data)  
print(type(data))
```

↪ Vui lòng nhập tuổi của bạn!Hai năm  
Hai năm  
<class 'str'>

```
#Nhập thông tin từ bàn phím  
data = input("Vui lòng nhập tuổi của bạn!")  
print(data)  
print(type(data))
```

↪ Vui lòng nhập tuổi của bạn!25  
25  
<class 'str'>

### int ()

```
#Gán kiểu dữ liệu int  
data = input("Vui lòng nhập tuổi của bạn!")  
data_int = int(data)  
print(data_int)  
print(type(data_int))
```

↪ Vui lòng nhập tuổi của bạn!25  
25  
<class 'int'>

### float ()

```
#Gán kiểu dữ liệu float  
data = input("Vui lòng nhập tuổi của bạn!")  
data_float = float(data)  
print(data_float)  
print(type(data_float))
```

↪ Vui lòng nhập tuổi của bạn!25  
25.0  
<class 'float'>

# 0.1.3 Biến, kiểu dữ liệu cơ bản và hàm có sẵn

## Random Modules

```
▶ #Random số nguyên trong (0, 1]
import random
print(random.random())
print(random.random())
print(random.random())
```

```
⇨ 0.02437489704855278
0.5924683293558537
0.7096817125209349
```

```
▶ #Random choice
```

```
import random
print(random.choice(['red', 'blue', 'green']))
print(random.choice(['red', 'blue', 'green']))
print(random.choice(['red', 'blue', 'green']))
```

```
⇨ red
green
green
```

```
▶ #Random số nguyên trong đoạn [0, 5]
import random
print(random.randint(0, 5))
print(random.randint(0, 5))
print(random.randint(0, 5))
```

```
⇨ 4
3
5
```

```
▶ #Random range(start, stop, step)
```

```
import random
print(random.randrange(0, 101, 5))
print(random.randrange(0, 101, 5))
print(random.randrange(0, 101, 5))
```

```
⇨ 85
35
20
```

# 0.1.3 Biến, kiểu dữ liệu cơ bản và hàm có sẵn

## Math Modules



```
#Tính số pi  
import math  
print(math.pi)
```

↪ 3.141592653589793



```
#Tính sin  
import math  
x = 1  
print(math.sin(x))
```

↪ 0.8414709848078965



```
#Tính căn  
import math  
x = 9  
print(math.sqrt(x))
```

↪ 3.0



```
#Tính số e  
import math  
print(math.e)
```

↪ 2.718281828459045



```
#Tính arctan2  
import math  
x = 1  
y = 2  
print(math.atan2(x , y))
```

↪ 0.4636476090008061



```
#Trị tuyệt đối  
import math  
x = 1  
y = -1  
print(math.fabs(x))  
print(math.fabs(y))
```

↪ 1.0  
1.0

# 0.1.3 Biến, kiểu dữ liệu cơ bản và hàm có sẵn

## Toán tử số học

Toán Tử Số Học		
Toán Tử	Ý nghĩa	Ví dụ
+	Phép cộng	$2 + 3 = 5$
-	Phép trừ	$3 - 2 = 1$
*	Phép nhân	$2 * 2 = 4$
/	Phép chia	$10 / 2 = 5$
%	Phép chia lấy số dư	$10 \% 4 = 2$
//	Phép chia lấy số nguyên	$18 // 5 = 3$
**	Phép lũy thừa	$3**5 = 243$

⇒ 14  
6  
40  
2.5  
2  
2  
10000

#Toán tử số học

x = 10

y = 4

#Giá trị x cộng y và in ra kết quả  
print(x + y)

#Giá trị x trừ y và in ra kết quả  
print(x - y)

#Giá trị x nhân y và in ra kết quả  
print(x \* y)

#Giá trị x chia y và in ra kết quả  
print(x / y)

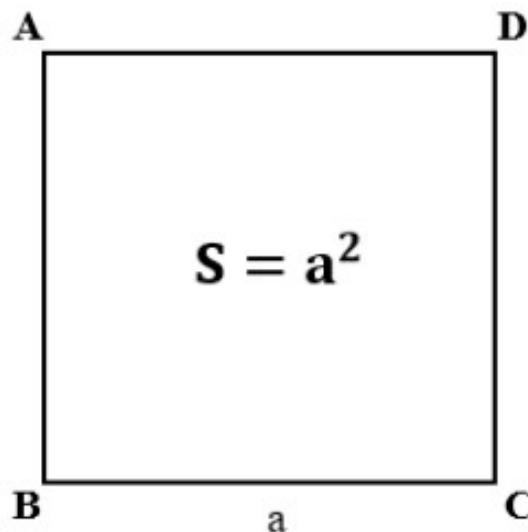
#Giá trị x chia y và in ra kết quả  
#Chỉ lấy phần nguyên  
print(x // y)

#Giá trị x chia y và in ra kết quả  
#Chỉ lấy phần dư  
print(x % y)

#Giá trị x mũ y  
print(x \*\* y)

## 0.1.3 Biến, kiểu dữ liệu cơ bản và hàm có sẵn

Ví dụ: Nhập giá trị cạnh từ bàn phím và tính diện tích hình vuông



```
#Tính diện tích hình vuông  
#input  
a = float(input("Hãy nhập giá trị cạnh!"))  
  
#process  
s = a * a  
  
#output  
print("Diện tích hình vuông là:", s)
```

Why?

⇒ Hãy nhập giá trị cạnh!5  
Diện tích hình vuông là: 25.0

## 0.1.4 Kiểu dữ liệu containers (List)

### Containers bao gồm:

- List (danh sách).
- Tuple (hàng).
- Dictionary (từ điển).
- Set (tập hợp)

### Giới thiệu List

- Giới hạn bởi các cặp ngoặc vuông [].
- Các phần tử của list cách nhau bằng dấu phẩy.
- List có khả năng chứa mọi kiểu dữ liệu của python, bao gồm cả chính nó.
- Là kiểu dữ liệu container.

## 0.1.4 Kiểu dữ liệu containers (List)

### Khởi tạo

- Tạo list cơ bản

```
list_name = [element-1, element-2,...element-n]
```



```
list1 = [1, 2, 3, 4, 5] # list gồm 5 số nguyên
print(list1)
list2 = ['a', 'b', 'c', 'd'] # list gồm 4 chuỗi
print(list2)
list3 = [[1, 2], [3, 4]] # list gồm 2 list con
print(list3)
list4 = [1, 'one', [2, 'two']] # list hỗn hợp
print(list4)
list5 = [] # list rỗng
print(list5)
```



```
[1, 2, 3, 4, 5]
['a', 'b', 'c', 'd']
[[1, 2], [3, 4]]
[1, 'one', [2, 'two']]
[]
```

## 0.1.4 Kiểu dữ liệu containers (List)

### Khởi tạo

- Tạo list bằng constructor

```
list_name = list(iterable)
```

Iterables
String
Tuple
List
Dictionary
range

```
▶ list1 = list([1, 2, 3, 4, 5]) # list gồm 5 số nguyên  
print(list1)  
  
list2 = list('hello') # list là chuỗi  
print(list2)  
  
list3 = list() # list rỗng  
print(list3)  
  
⇒ [1, 2, 3, 4, 5]  
['h', 'e', 'l', 'l', 'o']  
[]
```

## 0.1.4 Kiểu dữ liệu containers (List)

### Khởi tạo

- Tạo list bằng comprehension

```
list_name = [expression for element in iterable]
```



```
list1 = [n for n in 'Hello']
print(list1)

list2 = [n for n in range(10)]
print(list2)

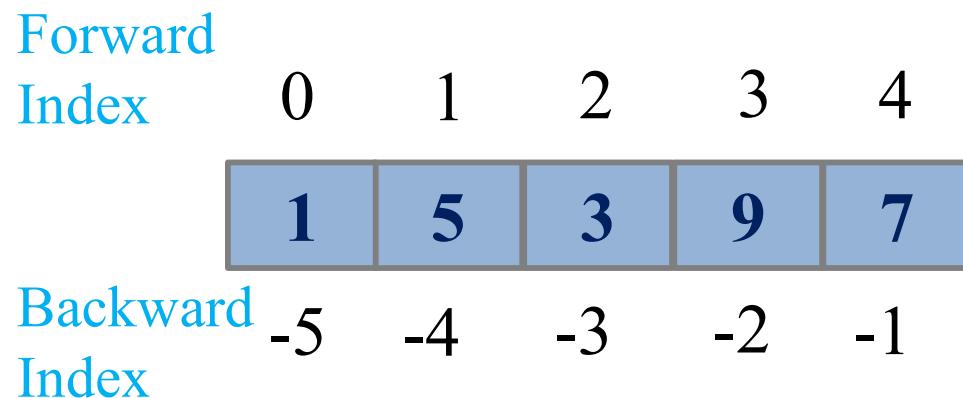
list3 = [n for n in range(10) if n%2 == 0]
print(list3)

list4 = [[n, n*5] for n in range(5)]
print(list4)
```



```
['H', 'e', 'l', 'l', 'o']
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 2, 4, 6, 8]
[[0, 0], [1, 5], [2, 10], [3, 15], [4, 20]]
```

## 0.1.4 Kiểu dữ liệu containers (List)



### Index

```
▶ list_index = [1, 5, 3, 9, 7]
print(list_index)
```

```
print(list_index[1])
print(list_index[3])
```

```
print(list_index[-1])
print(list_index[-3])
```

```
⇨ [1, 5, 3, 9, 7]
5
9
7
3
```

### Slicing

```
[start = 0 : stop : step = 1)
```

```
▶ list_slicing = [1, 5, 3, 9, 7]
print(list_slicing)
```

```
print(list_slicing[:2])
print(list_slicing[1:3])
print(list_slicing[2:])
```

```
⇨ [1, 5, 3, 9, 7]
[1, 5]
[5, 3]
[3, 9, 7]
```

## 0.1.4 Kiểu dữ liệu containers (List)

Index	0	1	2	3	4
	1	5	3	9	7

### Thêm phần tử

```
▶ # Add an element using append()
list_ap = [1, 5, 3, 9, 7]
print(list_ap)
# Thêm 2 vào cuối list
list_ap.append(2)
print(list_ap)      append()
```

```
⇒ [1, 5, 3, 9, 7]
[1, 5, 3, 9, 7, 2]
```

```
▶ # Add an element using insert()
list_in = [1, 5, 3, 9, 7]
print(list_in)
# Thêm 2 vào vị trí index 0
list_in.insert(0, 2)
print(list_in)      insert()
```

```
⇒ [1, 5, 3, 9, 7]
[2, 1, 5, 3, 9, 7]
```

### Cập nhật phần tử

```
▶ # Updating an element
list_up = [1, 5, 3, 9, 7]
print(list_up)

#Thay đổi vị trí 2 thành giá trị 0
list_up[2] = 0
#Thay đổi vị trí 3 thành giá trị 8
list_up[3] = 8
print(list_up)
```

```
⇒ [1, 5, 3, 9, 7]
[1, 5, 0, 8, 7]
```

## 0.1.4 Kiểu dữ liệu containers (List)

### Xóa phần tử

Index	0	1	2	3	4
	1	5	3	9	7



```
# Delete an element
list_del = [1, 5, 3, 9, 7]
print(list_del)
#Xóa phần tử có index = 3
list_del.pop(3)
print(list_del)      pop()
```

```
↳ [1, 5, 3, 9, 7]
[1, 5, 3, 7]
```



```
# Delete an element
list_del = [1, 5, 3, 9, 7]
print(list_del)
#Xóa phần tử thứ 2 và 3
del list_del[2 : 4]
print(list_del) del list()
```

```
↳ [1, 5, 3, 9, 7]
[1, 5, 7]
```



```
# Delete an element
list_del = [1, 5, 3, 9, 7]
print(list_del)
#Xóa phần tử có giá trị là 9 đầu tiên
list_del.remove(9)
print(list_del)      remove()
```

```
↳ [1, 5, 3, 9, 7]
[1, 5, 3, 7]
```



```
# Delete an element
list_del = [1, 5, 3, 9, 7]
print(list_del)
#Xóa tất cả phần tử của list
list_del.clear()
print(list_del) clear()
```

```
↳ [1, 5, 3, 9, 7]
[]
```

## 0.1.4 Kiểu dữ liệu containers (List)

Index	0	1	2	3	4
-------	---	---	---	---	---



▶ list\_s = [1, 5, 3, 9, 7]  
print(list\_s)  
#Sắp xếp phần tử  
list\_s.sort()  
print(list\_s)  
list\_s.sort(reverse = True)  
print(list\_s) **sort()**

⇒ [1, 5, 3, 9, 7]  
[1, 3, 5, 7, 9]  
[9, 7, 5, 3, 1]

▶ list\_s = [1, 5, 3, 9, 7]  
print(list\_s)  
#Đảo ngược vị trí phần tử  
list\_s.reverse()  
print(list\_s) **reverse()**

⇒ [1, 5, 3, 9, 7]  
[7, 9, 3, 5, 1]

▶ list\_ct = [1, 5, 3, 9, 7]  
print(list\_ct)  
#Số lần xuất hiện giá trị 3  
count = list\_ct.count(3)  
print(count) **count()**

⇒ [1, 5, 3, 9, 7]  
1

▶ list\_cy = [1, 5, 3, 9, 7]  
print(list\_cy)  
#Copy một list  
cop = list\_cy.copy()  
print(cop) **copy()**

⇒ [1, 5, 3, 9, 7]  
[1, 5, 3, 9, 7]

## 0.1.4 Kiểu dữ liệu containers (Tuple)

### Giới thiệu

- Giới hạn bởi các cặp ngoặc tròn () .
- Các phần tử của list cách nhau bằng dấu phẩy.
- Tuple có khả năng chứa mọi kiểu dữ liệu của python, bao gồm cả chính nó.
- Tốc độ truy xuất dữ liệu tuple nhanh hơn list.
- Chiếm dung lượng bộ nhớ ít hơn list.
- Là kiểu dữ liệu container.

### Khởi tạo

```
tuple_name = (element-1, element-2,...element-n)
```

```
▶ tup1 = ('Hello', 'Tom', 2020, 2030)
tup2 = (1, 2, 3, 4, 5 )
print(tup1)
print(tup2)
```

```
⇨ ('Hello', 'Tom', 2020, 2030)
(1, 2, 3, 4, 5)
```

# 0.1.4 Kiểu dữ liệu containers (Tuple)

## Một số hàm thường dùng

```
▶ tup = (1, 9, 3, 7, 5)

index1 = tup[2]
index2 = tup[-1]
print(index1)
print(index2)

slicing = tup[1:3]
print(slicing)

sort_tup = sorted(tup)
print(sort_tup)

count_tup = tup.count(3)
print(count_tup)
```

Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1, 2, 3): print x,	1 2 3	Iteration

```
◀ 3
 5
(9, 3)
[1, 3, 5, 7, 9]
1
```

# 0.1.4 Kiểu dữ liệu containers (Set)

## Giới thiệu

- Giới hạn bởi các cặp ngoặc nhọn {}.
- Các phần tử của set cách nhau bằng dấu phẩy.
- Set tập hợp các đối tượng không trùng nhau.
- Set có thể khởi tạo bằng constructor hoặc comprehension tương tự như list.
- Là kiểu dữ liệu container.

## Khởi tạo

```
set_name = {element-1, element-2,...element-n}
```



```
thisset = {1, 2, 3, 4, 5}
print(thisset)
print(type(thisset))
```



```
{1, 2, 3, 4, 5}
<class 'set'>
```

# 0.1.4 Kiểu dữ liệu containers (Set)

## Một số hàm thường dùng



```
thisset = {1, 2, 3, 4, 5}
print(thisset)
#Tạo một bản sao
set1 = thisset.copy()
print(set1)
#Thêm phần tử vào set
thisset.add(6)
print(thisset)
#Xóa tất cả phần tử trong set
set1.clear()
print(set1)
```



```
{1, 2, 3, 4, 5}
{1, 2, 3, 4, 5}
{1, 3, 4, 5}
{1, 2, 3, 4, 5, 6}
set()
```



```
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}

print(set1)
print(set2)
#Xóa phần tử khỏi set
#Nếu không có sẽ bị lỗi
set1.remove(5)
print(set1)
#Xóa phần tử khỏi set
set2.discard(8)
print(set2)
```



```
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{1, 2, 3, 4}
{4, 5, 6, 7}
```

## 0.1.4 Kiểu dữ liệu containers (Dictionary)

### Giới thiệu

- Từ điển là một danh sách các từ (key) và định nghĩa của nó (value).
- Yêu cầu các key không được trùng nhau, như vậy có thể xem từ điển như một loại set.
- Từ điển có thể khai báo theo cú pháp của set.
- Là kiểu dữ liệu container.

### Khởi tạo

element

```
dict_name = {key-1 : value-1, key-2 : value-1,... key-n : value-n}
```



```
dicti = {'Name': 'Tom', 'Age': 7, 'Class': 'First'}  
print(dicti)  
print(type(dicti))
```

```
▶ {'Name': 'Tom', 'Age': 7, 'Class': 'First'}  
<class 'dict'>
```

## 0.1.4 Kiểu dữ liệu containers (Dictionary)

### Một số hàm thường dùng

```
▶ dicti = {'Name': 'Tom', 'Age': 7, 'Class': 'First'}
print(dicti)
#Lấy value của key
print(dicti['Name'])
#Kiểm tra key có thuộc dict
print('Name' in dicti)
#Thêm element
dicti['ID'] = 50
print(dicti)
#Xóa element
del dicti['ID']
print(dicti)
dic1 = dicti.copy()
print(dic1) #Ảnh hưởng dict gốc
dic2 = copy.deepcopy(dicti)
print(dic2) #Không ảnh hưởng dict gốc
```

```
⇨ {'Name': 'Tom', 'Age': 7, 'Class': 'First'}
Tom
True
{'Name': 'Tom', 'Age': 7, 'Class': 'First', 'ID': 50}
{'Name': 'Tom', 'Age': 7, 'Class': 'First'}
{'Name': 'Tom', 'Age': 7, 'Class': 'First'}
{'Name': 'Tom', 'Age': 7, 'Class': 'First'}
```

## 0.1.4 Kiểu dữ liệu containers (Dictionary)

### Một số hàm thường dùng



```
dicti = {'Name': 'Tom', 'Age': 7, 'Class': 'First'}  
#Lấy key  
keys = dicti.keys()  
for key in keys:  
    print(key)  
#Lấy value  
values = dicti.values()  
for value in values:  
    print(value)  
#Lấy key và value  
items = dicti.items()  
for key, value in items:  
    print(key, value)
```



Name  
Age  
Class  
Tom  
7  
First  
Name Tom  
Age 7  
Class First

# 0.1.4 Kiểu dữ liệu containers (Tổng kết)

## List

```
list_name = [element-1, element-2,...element-n]
```

## Tuple

```
tuple_name = (element-1, element-2,...element-n)
```

## Set

```
set_name = {element-1, element-2,...element-n}
```

## Dictionary

```
dict_name = {key-1 : value-1, key-2 : value-1,... key-n : value-n}
```

## 0.1.5 Xây dựng hàm

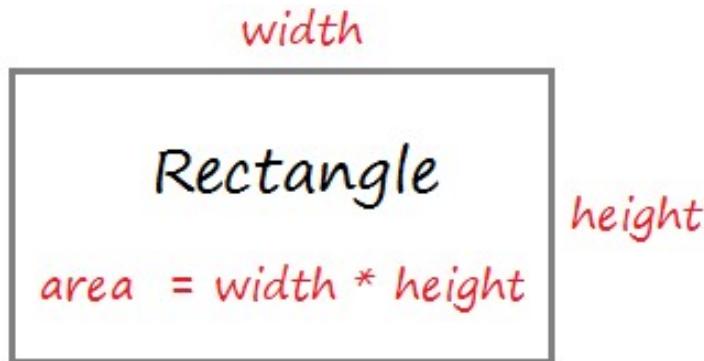
- Function là một khối code python được thực hiện một hoặc một số chức năng nhất định.
- Function trong python được định nghĩa với keyword **def**.

### Cấu trúc:

```
def function_name(parameters):
    """
        do docstring here
    """
    code here
    return result
```

# 0.1.5 Xây dựng hàm

## Ví dụ 1: Viết hàm tính diện tích hình chữ nhật



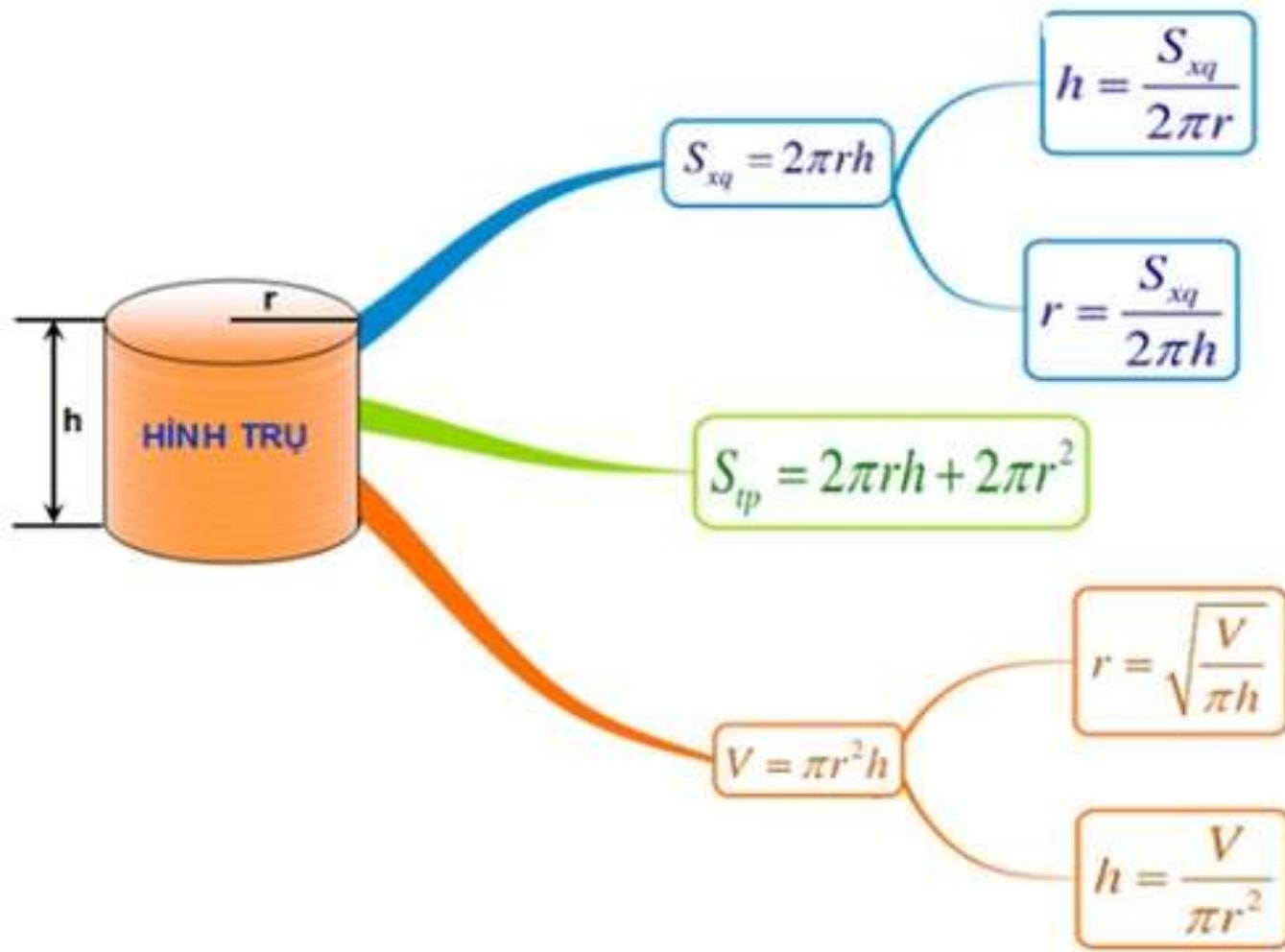
function_name	rectangle_area
parameters	height, width
result	area

```
#Compute rectangle area
def rectangle_area(height, width):
    ...
    This function aim to compute area for a rectangle
    height - the height of the rectangle
    width - the width of the rectangle
    This function returns the area of the rectangle
    ...
    area = height * width
    return area
# test case
# (height = 2,width = 3) -> area = 6
# (height = 20,width = 10) -> area = 200
area1 = rectangle_area(2, 3)
print(area1)
area2 = rectangle_area(height = 20,width = 10)
print(area2)
```

6  
200

## 0.1.5 Xây dựng hàm

**Ví dụ 2:** Viết hàm tính các thông số của hình trụ bên dưới



# 0.1.6 Điều khiển luồng dữ liệu (Điều kiện IF)

## Toán tử so sánh

TOÁN TỬ SO SÁNH	
TOÁN TỬ	Ý NGHĨA
==	Bằng
!=	Khác
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng

False  
True  
False  
True  
False  
True

#Comparison operators

x = 3

y = 6

#Biến x có bằng biến y không?

print(x == y)

#Biến x có khác biến y không?

print(x != y)

#Biến x có lớn hơn biến y không?

print(x > y)

#Biến x có nhỏ hơn biến y không?

print(x < y)

#Biến x có lớn hơn hoặc bằng biến y không?

print(x >= y)

#Biến x có nhỏ hơn hoặc bằng biến y không?

print(x <= y)

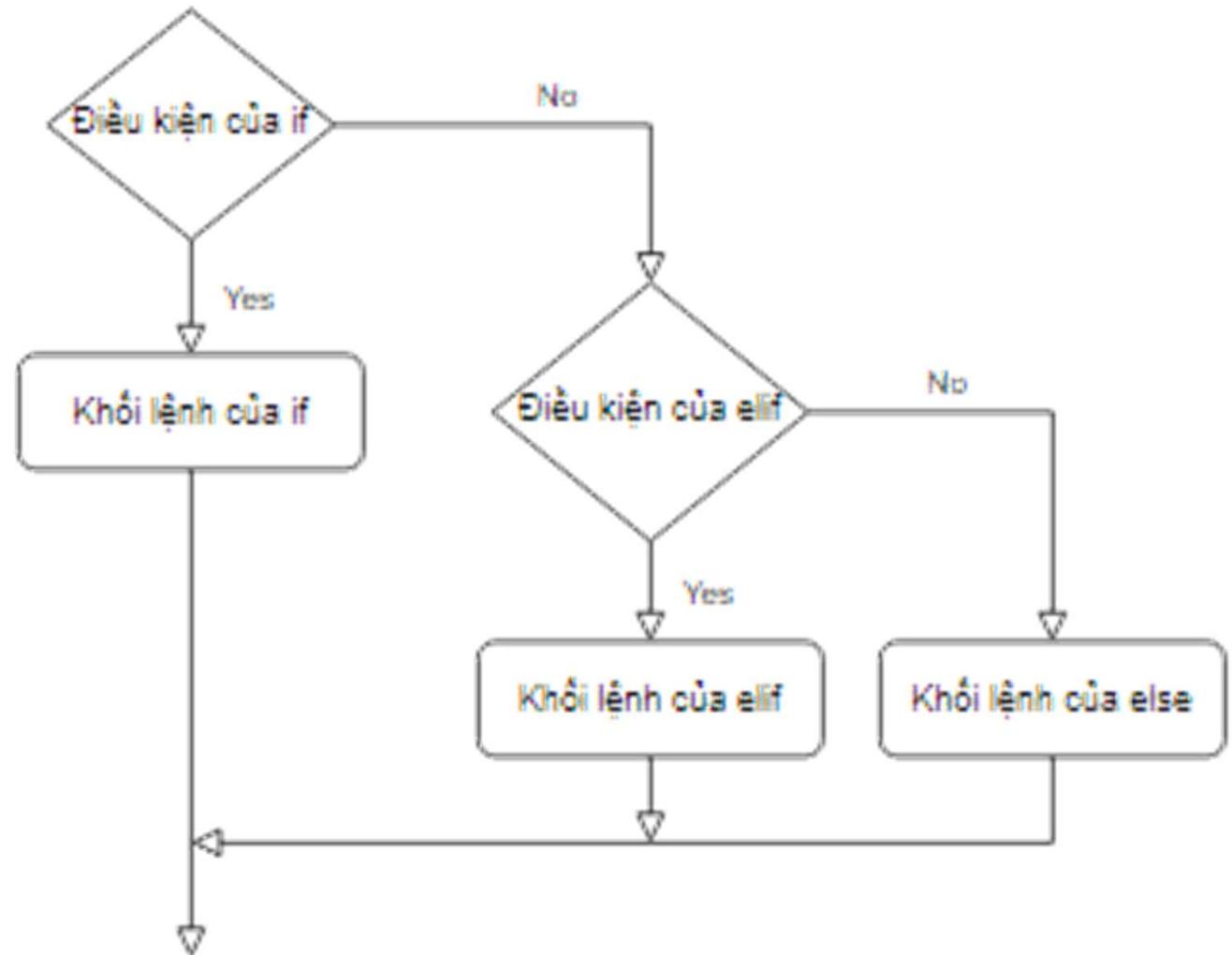
# 0.1.6 Điều khiển luồng dữ liệu (Điều kiện IF)

## Cấu trúc:

if condition : if code

elif condition : elif code

else : else code



# 0.1.6 Điều khiển luồng dữ liệu (Điều kiện IF)

## Ví dụ 1:



```
num = 3
if num >= 0:
    print('Số dương hoặc bằng 0')
else:
    print('Số âm')
```

⇨ Số dương hoặc bằng 0

## Ví dụ 2:



```
x = 10
if x > 0:
    print('Số dương')
elif x == 0:
    print('Số không')
else:
    print('Số âm')
```

⇨ Số dương

## Ví dụ 3:



```
a = 10
if a == 10:
    print('a = 10')
elif a > 5 :
    print('a > 5')
elif a < 0 :
    print('a < 0')
else:
    print('Điều kiện sai')
```

⇨ a = 10

## Ví dụ 4:



```
num = 10
if num >= 0:
    if num == 0:
        print('Số không')
    else:
        print('Số dương')
else:
    print('Số âm')
```

⇨ Số dương

## 0.1.7 Vòng lặp For (For Loop)

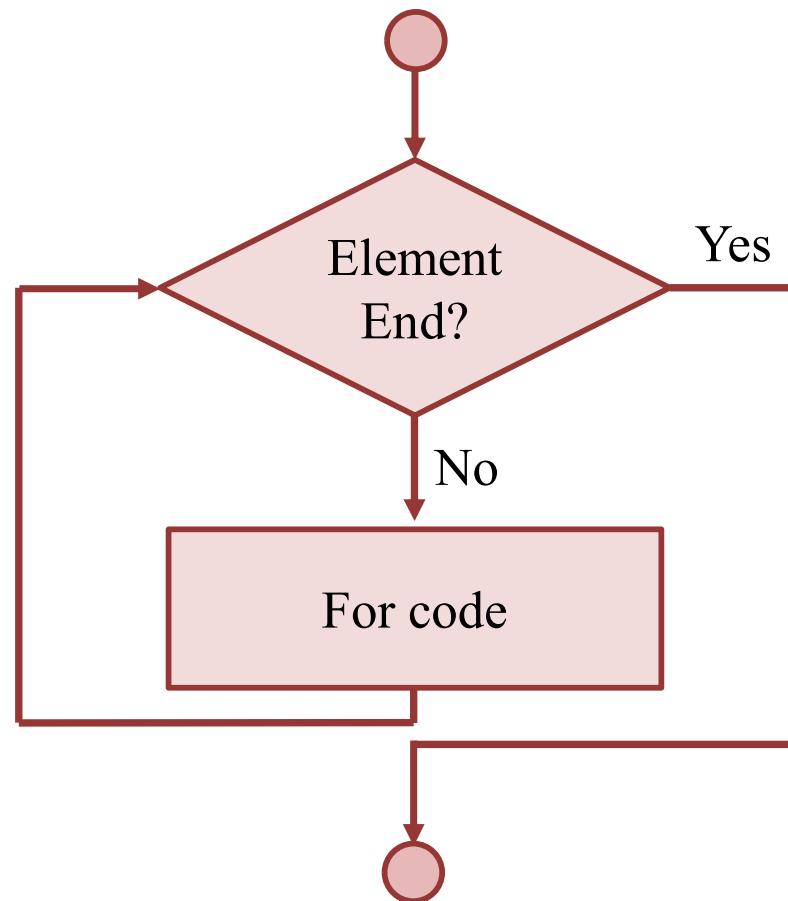
### Cấu trúc:

for element in iterable:



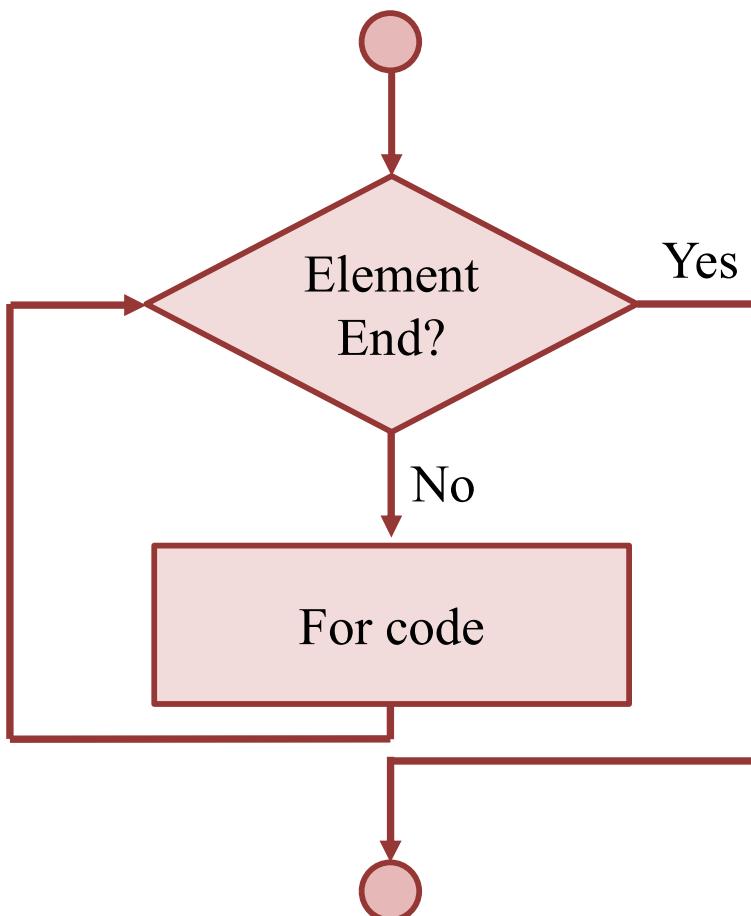
Lấy từng phần tử của iterable gán vào element

Iterables	String
	Tuple
	List
	Dictionary
	range



# 0.1.7 Vòng lặp For (For Loop)

Ví dụ :



```
string1 = "Hello"  
for char in string1:  
    print(char)
```



```
tuple1 = ('A', 'B', 'C')  
for tup in tuple1:  
    print(tup)
```

H  
e  
l  
l  
o

A  
B  
C



```
list1 = ['Tom', 'Lena', 'Thomas']  
for lis in list1:  
    print(lis)
```



```
for i in range(5):  
    print(i)
```

Tom  
Lena  
Thomas

0  
1  
2  
3  
4



```
dictionary1 = {'theta': 0.1, 'alpha': 0.2, 'gama': 'None'}  
for key in dictionary1:  
    print(key, dictionary1.get(key))
```

theta 0.1  
alpha 0.2  
gama None

# 0.1.7 Vòng lặp For (For Loop)

## For...break

Chức năng: thoát khỏi vòng lặp khi điều kiện xảy ra.

### Ví dụ 1:

```
▶ tuple1 = ('A', 'B', 'C', 'D', 'E', 'F')
  for tup in tuple1:
    if tup == 'D':
      break
    print(tup)
```

⇨ A  
B  
C

### Ví dụ 2:

```
▶ for n in range(20):
  if n == 10:
    break
  print('Giá trị n là:', n)
```

⇨ Giá trị n là: 0  
Giá trị n là: 1  
Giá trị n là: 2  
Giá trị n là: 3  
Giá trị n là: 4  
Giá trị n là: 5  
Giá trị n là: 6  
Giá trị n là: 7  
Giá trị n là: 8  
Giá trị n là: 9

# 0.1.7 Vòng lặp For (For Loop)

## For...continue

Chức năng: bỏ qua vòng lặp chứa điều kiện xảy ra.

### Ví dụ 1:



```
tuple1 = ('A', 'B', 'C', 'D', 'E', 'F')
for tup in tuple1:
    if tup == 'D':
        continue
    print(tup)
```

↪ A  
B  
C  
E  
F

### Ví dụ 2:



```
for n in range(10):
    if n == 7:
        continue
    print('Giá trị n là:', n)
```

↪ Giá trị n là: 0  
Giá trị n là: 1  
Giá trị n là: 2  
Giá trị n là: 3  
Giá trị n là: 4  
Giá trị n là: 5  
Giá trị n là: 6  
Giá trị n là: 8  
Giá trị n là: 9

# 0.1.7 Vòng lặp For (For Loop)

## Bài toán tung đồng xu



```
# Bài toán tung đồng xu
import random

tails = 0
flips = 0

for i in range(100):
    n = random.randint(0,1) #tao so nguyen tu 0 den 1
    if n == 0:
        tails = tails + 1
    else: flips = flips + 1

print(tails)
print(flips)
```

⌚ 52  
48

# 0.1.7 Vòng lặp For (For Loop)

## Bài toán tung xúc sắc

```
import random

one = 0
two = 0
three = 0
four = 0
five = 0
six = 0

for i in range(10):
    ran = random.randint(1,6) #tao so nguyen tu 0 den 6
    if ran == 1: one = one + 1
    elif ran == 2: two = two + 1
    elif ran == 3: three = three + 1
    elif ran == 4: four = four + 1
    elif ran == 5: five = five + 1
    else: six = six + 1

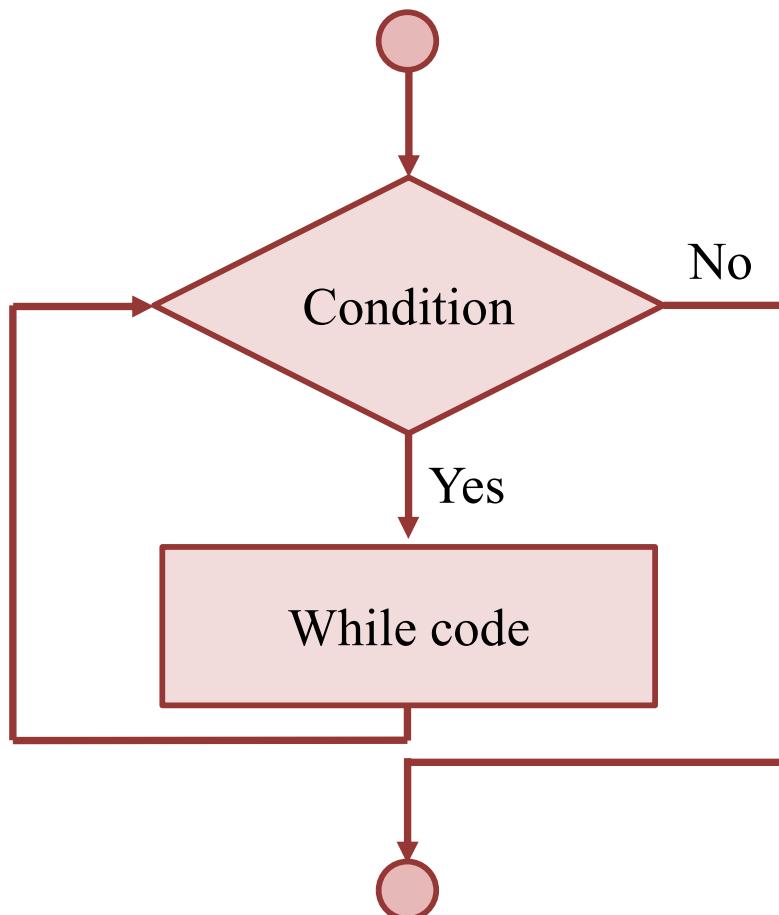
print(one)
print(two)
print(three)
print(four)
print(five)
print(six)
```

3	2
0	3
1	1

## 0.1.8 Vòng lặp While (While Loop)

### Cấu trúc:

while condition: while code



▶ n = 0  
while n < 10:  
 n = n + 1  
 print('Giá trị n là:', n)

⇨ Giá trị n là: 1  
Giá trị n là: 2  
Giá trị n là: 3  
Giá trị n là: 4  
Giá trị n là: 5  
Giá trị n là: 6  
Giá trị n là: 7  
Giá trị n là: 8  
Giá trị n là: 9  
Giá trị n là: 10

## 0.1.8 Vòng lặp While (While Loop)

### Vòng lặp vô tận: while-True-break



```
import random
while True:
    number = random.randint(0, 20)
    print('Số nguyên ngẫu nhiên được sinh ra là:', number)
    if number == 10:
        break
print('Đã ra ngoài vòng lặp!')
```

- ➡ Số nguyên ngẫu nhiên được sinh ra là: 7
- Số nguyên ngẫu nhiên được sinh ra là: 11
- Số nguyên ngẫu nhiên được sinh ra là: 13
- Số nguyên ngẫu nhiên được sinh ra là: 7
- Số nguyên ngẫu nhiên được sinh ra là: 0
- Số nguyên ngẫu nhiên được sinh ra là: 9
- Số nguyên ngẫu nhiên được sinh ra là: 4
- Số nguyên ngẫu nhiên được sinh ra là: 13
- Số nguyên ngẫu nhiên được sinh ra là: 17
- Số nguyên ngẫu nhiên được sinh ra là: 20
- Số nguyên ngẫu nhiên được sinh ra là: 0
- Số nguyên ngẫu nhiên được sinh ra là: 10
- Đã ra ngoài vòng lặp!

## 0.1.9 Class

- Class trong python được định nghĩa với keyword **class**.

### Cấu trúc:

```
#Class
class class_name:
    ...
    do docstring here
    ...
    def __init__(self, Attribute_1,...Attribute_n)
        self.Attribute_1 = Attribute_1
        .
        .
        .
        self.Attribute_n = Attribute_n
    code here
```

- Phương thức khởi tạo luôn có tên là **\_\_init\_\_**
- Thuộc tính đầu tiên luôn là **self**

# 0.1.9 Class

Ví dụ 1: Viết một class biểu diễn hình chữ nhật

width

Rectangle

height

area = width \* height

class_name	rectangle
Attribute_1	width
Attribute_2	height

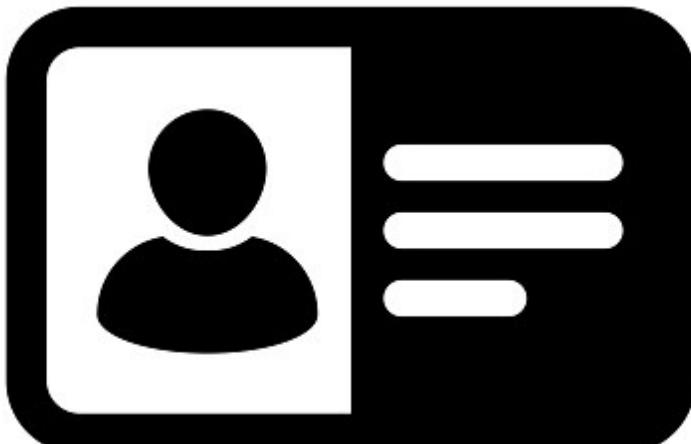
```
class Rectangle :  
    '''This is Rectangle class'''  
    def __init__(self, width, height):  
        self.width= width  
        self.height = height  
  
    def getWidth(self):  
        return self.width  
  
    def getHeight(self):  
        return self.height  
  
    def getArea(self):  
        return self.width * self.height
```

```
r = Rectangle(3,10)  
print ("r.width = ", r.width)  
print ("r.height = ", r.height)  
print ("r.getArea() = ", r.getArea())
```

⇒ r.width = 3  
r.height = 10  
r.getArea() = 30

# 0.1.9 Class

**Ví dụ 2:** Viết một class  
biểu diễn thông tin 1 người



class_name	Person
Attribute_1	name
Attribute_2	age
Attribute_2	gender

```
class Person:  
    def __init__(self, name, age, gender):  
        self.name = name  
        self.age = age  
        self.gender= gender  
  
    def showInfo(self):  
        print ("Name: ", self.name)  
        print ("Age: ", self.age)  
        print ("Gender: ", self.gender)  
  
aimee = Person("Lena", 18, "Female")  
aimee.showInfo()
```

⇨ Name: Lena  
Age: 18  
Gender: Female

# 0.2 Thư viện NumPy

---

---

## 0.2.1 Introduction

## 0.2.2 Create NumPy Array

## 0.2.3 NumPy Array Indexing

## 0.2.4 NumPy Array Operations

## 0.2.5 Broadcasting

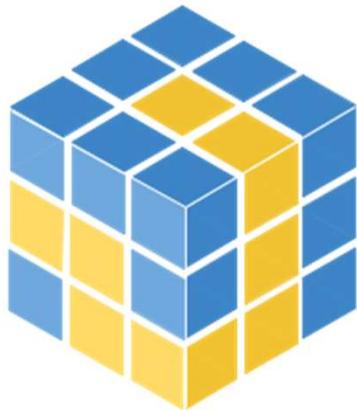
## 0.2.6 Exercises

## 0.2.1 Introduction

---

NumPy là một cấu trúc dữ liệu dùng để lưu trữ và truy cập hiệu quả các mảng đa chiều (còn được gọi là tensors), đồng thời cho phép nhiều phép tính khoa học.

NumPy là thư viện lập trình mảng chính cho ngôn ngữ Python.



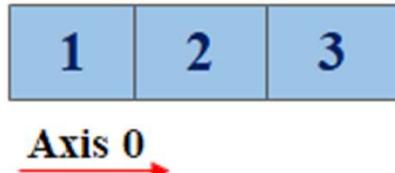
*NumPy*

<https://numpy.org/doc/1.19/reference/index.html>

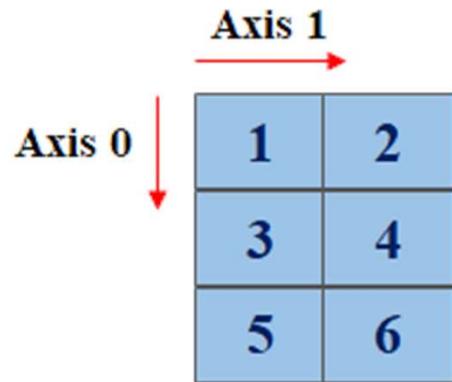
## 0.2.1 Introduction

NumPy là mảng đa chiều

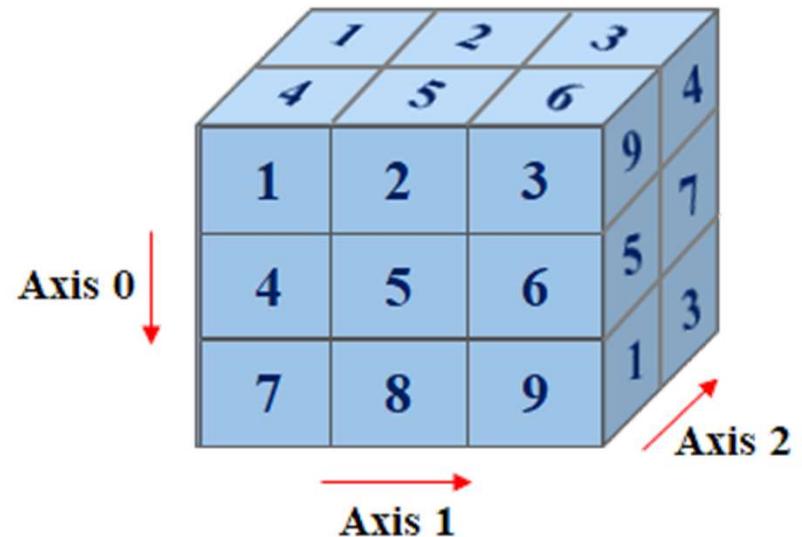
**1D array**



**2D array**



**3D array**



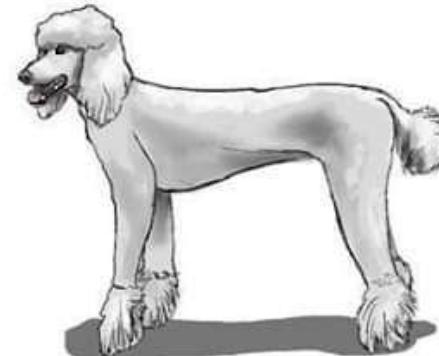
Scalar



Vector



Matrix



Tensor



## 0.2.2 Create NumPy Array

### Create array

1	2	3
---	---	---

1	2
3	4
5	6

```
▶ # Create 1D array  
import numpy as np  
  
arr1D = np.array([1, 2, 3])  
print(arr1D)  
print(arr1D.shape)  
print(type(arr1D))
```

```
⇨ [1 2 3]  
(3,)  
<class 'numpy.ndarray'>
```

```
▶ # Create 2D array  
import numpy as np  
  
arr2D = np.array([[1, 2], [3, 4], [5, 6]])  
print(arr2D)  
print(arr2D.shape)  
print(type(arr2D))
```

```
⇨ [[1 2]  
 [3 4]  
 [5 6]]  
(3, 2)  
<class 'numpy.ndarray'>
```

## 0.2.2 Create NumPy Array

### Default array

zero() function

0	0	0
0	0	0

one() function

1	1	1
1	1	1

full() function

2	2	2
2	2	2

▶ # Zero array

```
import numpy as np  
a = np.zeros((2,3))  
print(a)
```

⇨ [[0. 0. 0.]  
 [0. 0. 0.]]

▶ # One array

```
import numpy as np  
b = np.ones((2,3))  
print(b)
```

⇨ [[1. 1. 1.]  
 [1. 1. 1.]]

▶ # Full array

```
import numpy as np  
c = np.full((2,3), 2)  
print(c)
```

⇨ [[2 2 2]  
 [2 2 2]]

## 0.2.2 Create NumPy Array

### Default array

eye() function

1	0	0
0	1	0
0	0	1

arange() function

0	1	2	3	4
---	---	---	---	---

1	3	5	7	9
---	---	---	---	---



# Eye array

```
import numpy as np  
d = np.eye(3)  
print(d)
```



```
[[1. 0. 0.]  
 [0. 1. 0.]  
 [0. 0. 1.]]
```



# Arange array

```
import numpy as np  
# np.arange(start = 0, stop, step = 1)  
# [start : stop)  
e1 = np.arange(5)  
e2 = np.arange(1, 10, 2)  
print(e1)  
print(e2)
```



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

## 0.2.2 Create NumPy Array

### Default array

random() function

0.9	0.6	0.8
0.7	0.7	0.7
0.2	0.9	0.9



# Random array

```
import numpy as np  
f = np.random.random((3, 3))  
print(f)
```



```
[[0.94890075 0.63842827 0.86592682]  
 [0.78797348 0.77225518 0.77438132]  
 [0.21043683 0.93176845 0.92767475]]
```

### Reshape

array

1	2	3
4	5	6



array\_rs

1	2
3	4
5	6

#Reshape

```
import numpy as np  
array = np.array([[1, 2, 3], [4, 5, 6]])  
array_rs = np.reshape(array, (3, 2))  
print(array)  
print(array.shape)  
print(array_rs)  
print(array_rs.shape)
```



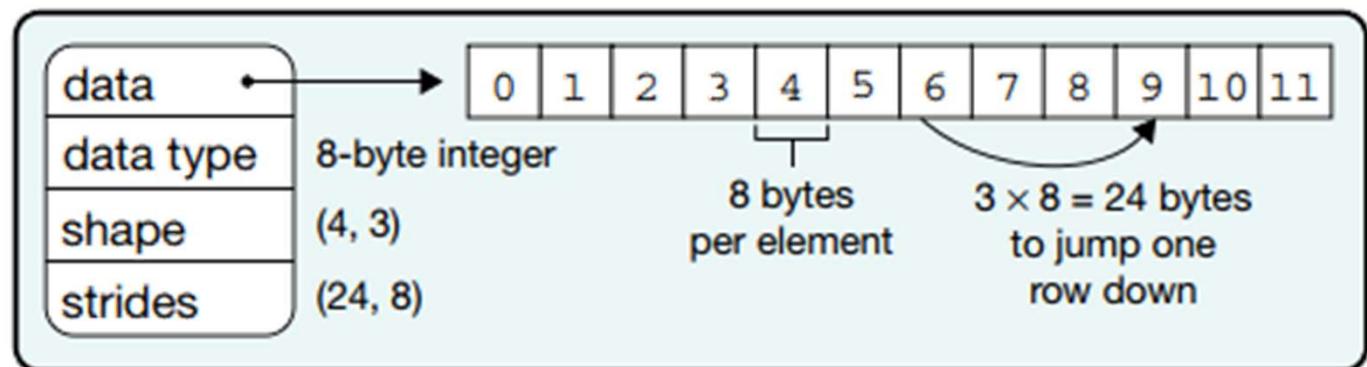
```
[[1 2 3]  
 [4 5 6]]  
(2, 3)  
[[1 2]  
 [3 4]  
 [5 6]]  
(3, 2)
```

## 0.2.2 Create NumPy Array

### Data structure

0	1	2
3	4	5
6	7	8
9	10	11

x =



```
# Data type
import numpy as np

arr1 = np.array([[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10, 11]])
arr2 = np.array([[0., 1., 2.], [3., 4., 5.], [6., 7., 8.], [9., 10., 11.]])
print(arr1)
print(arr1.dtype)
print(arr2)
print(arr2.dtype)
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
int64
[[ 0.  1.  2.]
 [ 3.  4.  5.]
 [ 6.  7.  8.]
 [ 9. 10. 11.]]
float64
```

## 0.2.2 Create NumPy Array

### Update element

Index 0 1 2

1	0	3
---	---	---

Index 0 1 2

1	2	3
4	5	6
7	0	9

```
# Update element
import numpy as np
arr1 = np.array([1, 2, 3])
print(arr1)
print(arr1[1])
arr1[1] = 0
print(arr1)
```

```
[1 2 3]
2
[1 0 3]
```

```
# Update element
import numpy as np
arr2 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(arr2)
print(arr2[2, 1])
print(arr2[1, 2])
arr2[2, 1] = 0
print(arr2)
```

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

## 0.2.3 NumPy Array Indexing

### Array Indexing

array[axis 0, axis 1,...]

array2D[axis 0 (row), axis1 (column)]

“:” lấy tất cả phần tử [start = 0 : stop : step = 1)

array	0	1
0	2	3
1	5	6
2	8	9

array [1, 1]	0	1
0	2	3
1	5	6
2	8	9

array [0: 2, 0]	0	1
0	2	3
1	5	6
2	8	9

array [0: 2]	0	1
0	2	3
1	5	6
2	8	9

array[:, :, ]	0	1
0	2	3
1	5	6
2	8	9

## 0.2.3 NumPy Array Indexing

### Slicing

array



array [1:3]    array [:−4]    array [3:]



array [:]



array [::−1]



# Indexing array

```
import numpy as np
array = np.array([1, 3, 5, 7, 9])
print(array[1: 3])
print(array[ : -4])
print(array[3: ])
print(array[ : ])
print(array[ :: -1])
```



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

# 0.2.3 NumPy Array Indexing

## Slicing

array

1	2	3
4	5	6
7	8	9

array[:, 0:2]

1	2
4	5
7	8

array [0: 2, :]

1	2	3
4	5	6

array [1:3, 1:]

5	6
8	9



```
# # Indexing array (slicing)
import numpy as np

array = np.array([[1, 2, 3],
                 [4, 5, 6],
                 [7, 8, 9]])

print(array)
print(array[:, 0:2])
print(array[0:2, :])
print(array[0:2, :])
print(array[1:3, 1:])
```



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

## 0.2.3 NumPy Array Indexing

### Get a row

array =

1	2	3
4	5	6
7	8	9

array\_row1 =

4	5	6
shape(3,)		

array\_row2 =

4	5	6
shape(1, 3)		



```
# Indexing array (get a row)
import numpy as np

array = np.array([[1, 2, 3],
                 [4, 5, 6],
                 [7, 8, 9]])

array_row1 = array[1, :]
array_row2 = array[1:2, :]

print(array_row1, array_row1.shape)
print(array_row2, array_row2.shape)
```

⇒ [4 5 6] (3,)
[[4 5 6]] (1, 3)

## 0.2.3 NumPy Array Indexing

### Get a column

array =

1	2	3
4	5	6
7	8	9

array\_col1 =

2	5	8
---	---	---

shape(3,)

array\_col2 =

2
5
8

shape(3, 1)



```
# Indexing array (get a row)
import numpy as np
```

```
array = np.array([[1, 2, 3],
                  [4, 5, 6],
                  [7, 8, 9]])
```

```
array_col1 = array[:, 1]
array_col2 = array[:, 1:2]
```

```
print(array_col1, array_col1.shape)
print(array_col2, array_col2.shape)
```

```
⇒ [2 5 8] (3,)
[[2]
 [5]
 [8]] (3, 1)
```

# 0.2.4 NumPy Array Operations

## Addition

$$\begin{array}{c} \text{x} \\ \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array} \end{array} + \begin{array}{c} \text{y} \\ \begin{array}{|c|c|} \hline 5 & 6 \\ \hline 7 & 8 \\ \hline \end{array} \end{array} = \begin{array}{c} \text{result} \\ \begin{array}{|c|c|} \hline 6 & 8 \\ \hline 10 & 12 \\ \hline \end{array} \end{array}$$

```
# Addition
import numpy as np
x = np.array([[1, 2], [3, 4]])
y = np.array([[5, 6], [7, 8]])
print("Array 1 \n", x)
print("Array 2 \n", y)

print("Method 1 \n", x + y)
print("Method 2 \n", np.add(x, y))
```

Array 1  
[[1 2]  
[3 4]]  
Array 2  
[[5 6]  
[7 8]]  
Method 1  
[[ 6 8]  
[10 12]]  
Method 2  
[[ 6 8]  
[10 12]]

# 0.2.4 NumPy Array Operations

## Subtraction

$$\begin{array}{c} \text{x} \\ \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array} \end{array} - \begin{array}{c} \text{y} \\ \begin{array}{|c|c|} \hline 5 & 6 \\ \hline 7 & 8 \\ \hline \end{array} \end{array} = \begin{array}{c} \text{result} \\ \begin{array}{|c|c|} \hline 6 & 8 \\ \hline 10 & 12 \\ \hline \end{array} \end{array}$$

```
# Subtraction
import numpy as np
x = np.array([[1, 2], [3, 4]])
y = np.array([[5, 6], [7, 8]])
print("Array 1 \n", x)
print("Array 2 \n", y)

print("Method 1 \n", x - y)
print("Method 2 \n", np.subtract(x, y))
```

▶ Array 1  
[[1 2]  
[3 4]]  
Array 2  
[[5 6]  
[7 8]]  
Method 1  
[[-4 -4]  
[-4 -4]]  
Method 2  
[[-4 -4]  
[-4 -4]]

# 0.2.4 NumPy Array Operations

## Multiplication (element-wise)

$$\begin{array}{c} \text{x} \\ \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array} \end{array} * \begin{array}{c} \text{y} \\ \begin{array}{|c|c|} \hline 5 & 6 \\ \hline 7 & 8 \\ \hline \end{array} \end{array} = \begin{array}{c} \text{result} \\ \begin{array}{|c|c|} \hline 6 & 8 \\ \hline 10 & 12 \\ \hline \end{array} \end{array}$$



```
# Multiplication
import numpy as np
x = np.array([[1, 2], [3, 4]])
y = np.array([[5, 6], [7, 8]])
print("Array 1 \n", x)
print("Array 2 \n", y)

print("Method 1 \n", x * y)
print("Method 2 \n", np.multiply(x, y))
```



```
Array 1
[[1 2]
 [3 4]]
Array 2
[[5 6]
 [7 8]]
Method 1
[[ 5 12]
 [21 32]]
Method 2
[[ 5 12]
 [21 32]]
```

# 0.2.4 NumPy Array Operations

## Divition

$$\begin{array}{c} \text{x} \\ \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array} \end{array} \quad / \quad \begin{array}{c} \text{y} \\ \begin{array}{|c|c|} \hline 5 & 6 \\ \hline 7 & 8 \\ \hline \end{array} \end{array} \quad = \quad \begin{array}{c} \text{result} \\ \begin{array}{|c|c|} \hline 6 & 8 \\ \hline 10 & 12 \\ \hline \end{array} \end{array}$$



```
# Divition
import numpy as np
x = np.array([[1, 2], [3, 4]])
y = np.array([[5, 6], [7, 8]])
print("Array 1 \n", x)
print("Array 2 \n", y)

print("Method 1 \n", x / y)
print("Method 2 \n", np.divide(x, y))
```

→ Array 1  
[[1 2]  
[3 4]]  
Array 2  
[[5 6]  
[7 8]]  
Method 1  
[[0.2 0.33333333]  
[0.42857143 0.5 ]]  
Method 2  
[[0.2 0.33333333]  
[0.42857143 0.5 ]]

# 0.2.4 NumPy Array Operations

## Square root

X

1	2
3	4

-----

sqrt(x) =

1	1.4
1.7	2

result

# Square root

```
import numpy as np
x = np.array([[1, 2], [3, 4]])
print("Array \n", x)

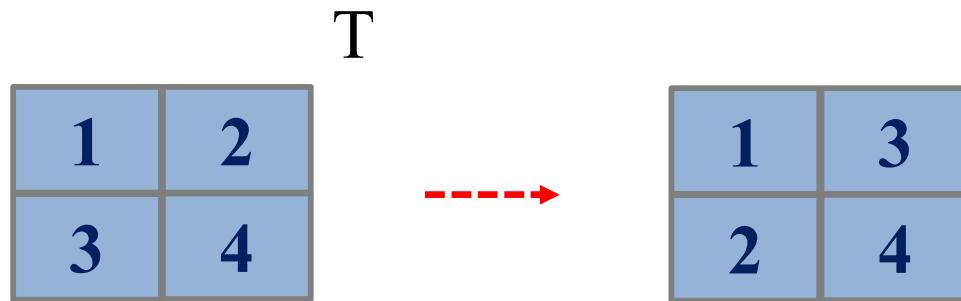
print("Sqrt \n", np.sqrt(x))
```

Array  
[[1 2]  
[3 4]]

Sqrt  
[[1. 1.41421356]  
[1.73205081 2. ]]

# 0.2.4 NumPy Array Operations

## Transpose



```
# Transpose
import numpy as np
x = np.array([[1, 2], [3, 4]])
print("Array \n", x)

print("Transpose \n", x.T)
```

Array  
[[1 2]  
[3 4]]  
Transpose  
[[1 3]  
[2 4]]

# 0.2.4 NumPy Array Operations

## Matrix (Vector) Multiplication

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 4 \end{bmatrix} = 11$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$



```
# Matrix Multiplication
import numpy as np
a = np.array([1, 2])
b = np.array([3, 4])
c = np.array([[1, 2], [3, 4]])
d = np.array([[5, 6], [7, 8]])
print("Method 1 a.b \n", a.dot(b))
print("Method 2 a.b \n", np.dot(a, b))

print("Method 1 c.d \n", c.dot(d))
print("Method 2 c.d\n", np.dot(c, d))
```



```
Method 1 a.b
11
Method 2 a.b
11
Method 1 c.d
[[19 22]
 [43 50]]
Method 2 c.d
[[19 22]
 [43 50]]
```

## 0.2.4 NumPy Array Operations

### Summation

array =

1	2	3
4	5	6
7	8	9

sum(array) =

45

sum(array, axis=0) =

12	15	18
----	----	----

sum(array, axis=1) =

6	15	24
---	----	----



# Summation

import numpy as np

```
array = np.array([[1, 2, 3],  
                 [4, 5, 6],  
                 [7, 8, 9]])
```

```
print(np.sum(array))
```

```
print(np.sum(array, axis=0))
```

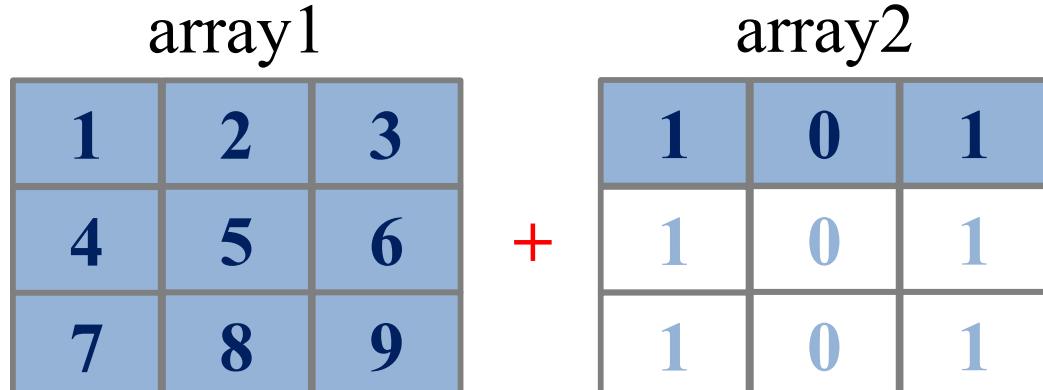
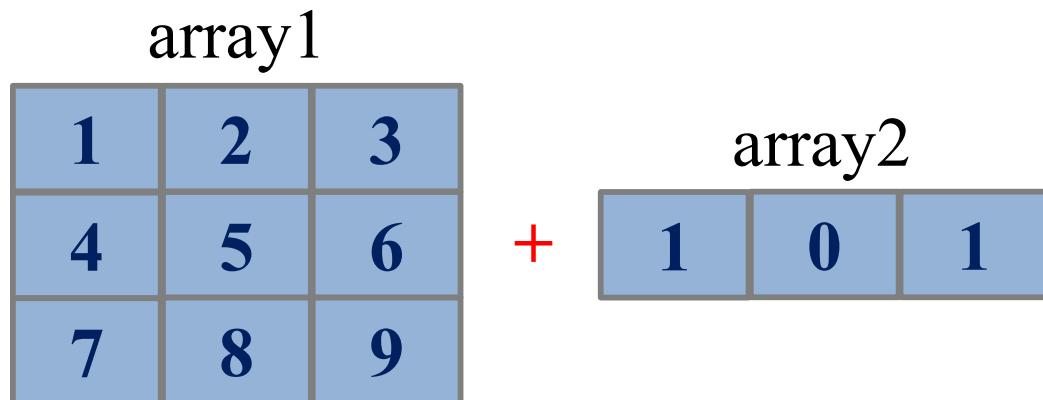
```
print(np.sum(array, axis=1))
```



```
45  
[12 15 18]  
[ 6 15 24]
```

## 0.2.5 Broadcasting

**Broadcasting** là một kỹ thuật cho phép numpy làm việc với các array có kích thước (shape) khác nhau khi thực hiện các phép toán.



```
# Broadcasting
import numpy as np

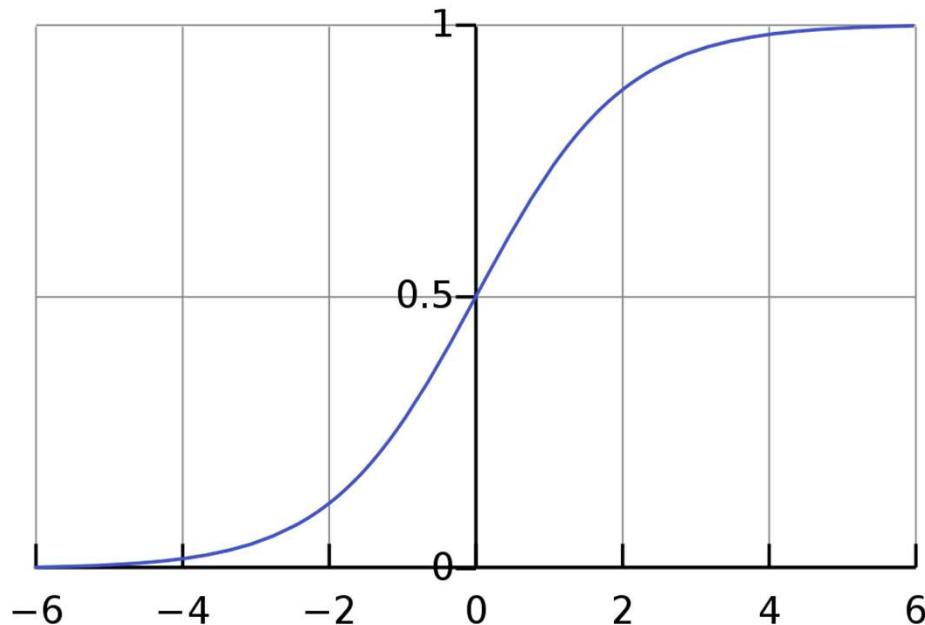
array1 = np.array([[1, 2, 3],
                  [4, 5, 6],
                  [7, 8, 9]])
array2 = np.array([1, 0, 1])
print('array1 \n', array1)
print('array2 \n', array2)
print('Sum \n', array1 + array2)
```

```
array1
[[1 2 3]
 [4 5 6]
 [7 8 9]]
array2
[1 0 1]
Sum
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]]
```

## 0.2.6 Exercises

### Exercises1: Viết hàm tính giá trị hàm sigmoid

$$f(x) = \frac{1}{1 + e^{-x}}$$



```
# GRADED FUNCTION: sigmoid
import numpy as np

def sigmoid(x):
    """
    Compute sigmoid of x.

    Arguments:
    x -- A scalar or numpy array

    Return:
    s -- sigmoid(x)
    """

    s = 1/(1+np.exp(-x))
    return s

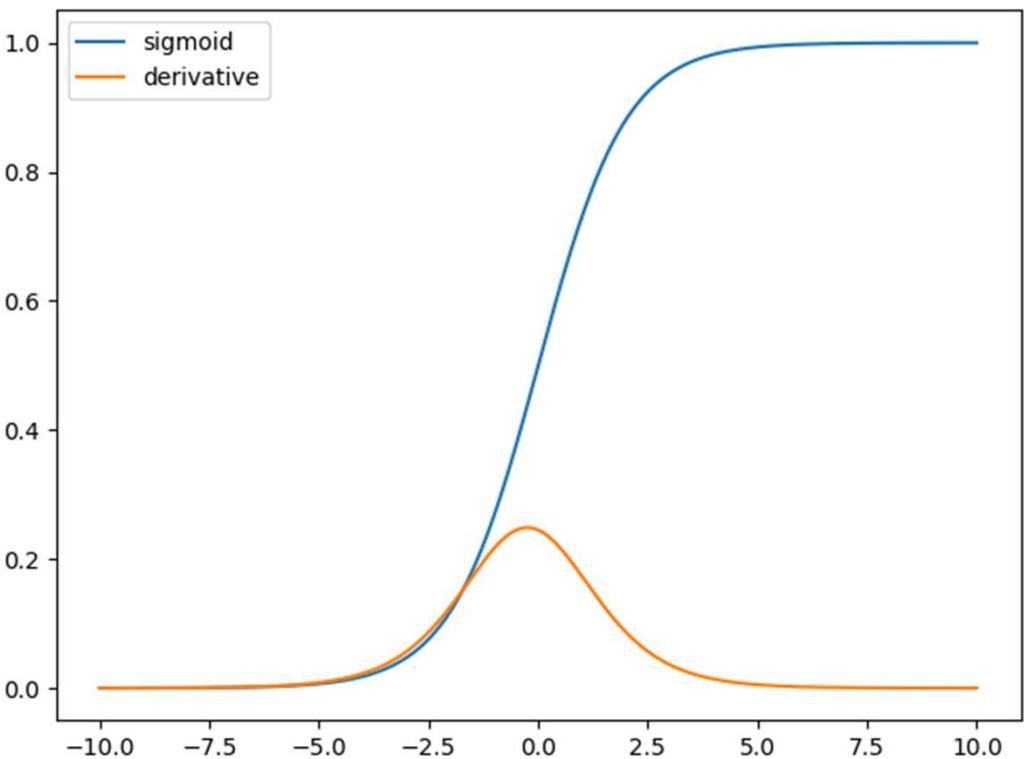
print(sigmoid(3))

x = np.array([1, 2, 3])
sigmoid(x)
```

⇒ 0.9525741268224334  
array([0.73105858, 0.88079708, 0.95257413])

## 0.2.6 Exercises

### Exercises2: Viết hàm tính giá trị đạo hàm của hàm sigmoid



```
# GRADED FUNCTION: sigmoid_derivative  
import numpy as np
```

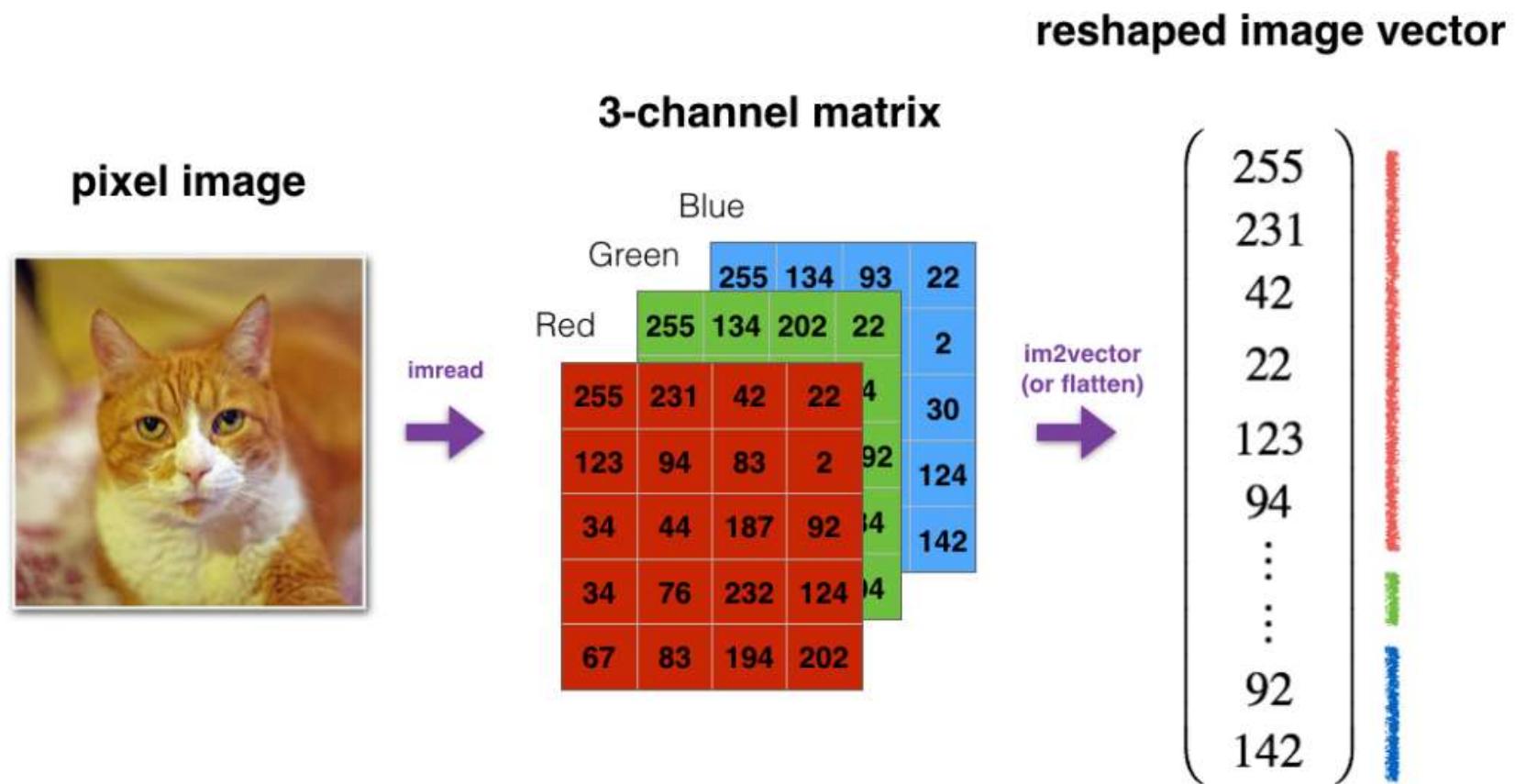
```
def sigmoid_derivative(x):  
    """  
    Compute sigmoid derivative of x.  
    Arguments:  
    x -- A scalar or numpy array  
    Return:  
    s -- sigmoid(x)  
    """  
    s = 1/(1+np.exp(-x))  
    ds = s*(1-s)  
    return ds
```

```
x = np.array([1, 2, 3])  
sigmoid_derivative(x)
```

```
array([0.19661193, 0.10499359, 0.04517666])
```

## 0.2.6 Exercises

**Exercises3:** Viết hàm tính chuyển hình ảnh từ 3D sang vector 1D



## 0.2.6 Exercises

**Exercises3:** Viết hàm tính chuyển hình ảnh từ 3D sang vector 1D



```
import numpy as np
def image2vector(image):
    """
    Argument:
    image -- a numpy array of shape (length, height, depth)

    Returns:
    v -- a vector of shape (length*height*depth, 1)
    """
    v = image.reshape(image.shape[0]*image.shape[1]*image.shape[2], 1)
    return v
```

## 0.2.6 Exercises

### Exercises3: Viết hàm tính chuyển hình ảnh từ 3D sang vector 1D

```
▶ image = np.array([[ [ 0.67826139,  0.29380381],
    [ 0.90714982,  0.52835647],
    [ 0.4215251 ,  0.45017551]],

    [[ 0.92814219,  0.96677647],
    [ 0.85304703,  0.52351845],
    [ 0.19981397,  0.27417313]],

    [[ 0.60659855,  0.00533165],
    [ 0.10820313,  0.49978937],
    [ 0.34144279,  0.94630077]]])

print ("image2vector(image) = " + str(image2vector(image)))
```

```
▶ image2vector(image) = [[0.67826139]
[0.29380381]
[0.90714982]
[0.52835647]
[0.4215251 ]
[0.45017551]
[0.92814219]
[0.96677647]
[0.85304703]
[0.52351845]
[0.19981397]
[0.27417313]
[0.60659855]
[0.00533165]
[0.10820313]
[0.49978937]
[0.34144279]
[0.94630077]]
```

# 0.3 Thư viện Pandas

---

---

## 0.3.1 Giới thiệu

## 0.3.2 Cấu trúc dữ liệu trong Pandas

## 0.3.3 Series trong Pandas

## 0.3.4 DataFrame trong Pandas

## 0.3.1 Giới thiệu

- Thư viện pandas trong python là một thư viện mã nguồn mở.
- Pandas là thư viện mở rộng từ numpy, chuyên để xử lý dữ liệu cấu trúc dạng bảng.
- Tên pandas là dạng số nhiều của “**panel data**”.

Pandas



<https://pandas.pydata.org/index.html>

[https://www.tutorialspoint.com/python\\_pandas/python\\_pandas\\_dataframe.htm](https://www.tutorialspoint.com/python_pandas/python_pandas_dataframe.htm)

## 0.3.2 Cấu trúc dữ liệu trong Pandas

Dữ liệu của pandas có 3 cấu trúc chính:

- Series: cấu trúc 1 chiều, mảng dữ liệu đồng nhất, series gần giống kiểu array trong numpy, chấp nhận dữ liệu thiêú (NaN – không xác định).
- Dataframe: cấu trúc 2 chiều, dữ liệu trên các cột là đồng nhất, các cột có tên, các dòng có thể có tên, chấp nhận dữ liệu thiêú (NaN – không xác định).
- Panel: cấu trúc 3 chiều, có thể xem như một tập các dataframe với thông tin bổ sung, đã bị gỡ bỏ từ phiên bản 0.25.

The diagram illustrates the structure of a DataFrame. On the left, two blue arrows point from the words "rows" and "columns" to a 3x3 grid representing a DataFrame. The grid has columns labeled "Regd. No", "Name", and "Marks%", and rows labeled 1000, 1001, and 1002. The data entries are Steve, 86.29; Mathew, 91.63; and Jose, 72.90 respectively.

Regd. No	Name	Marks%
1000	Steve	86.29
1001	Mathew	91.63
1002	Jose	72.90

## 0.3.3 Series trong Pandas

pandas.Series(data, index, dtype, copy)

- **data** sẽ nhận giá trị từ các kiểu khác nhau như ndarray, list, hằng số...
- **index** là nhãn chỉ vị trí, tương tự chiều dài dữ liệu.
- **dtype** là kiểu dữ liệu.
- **copy** nhận giá trị True/False để chỉ rõ dữ liệu có được copy sang vùng nhớ mới không, mặc định là False

10	23	56	17	52	61	73	90	26	72
----	----	----	----	----	----	----	----	----	----

## 0.3.3 Series trong Pandas

### Tạo Series từ ndarray

#### Ví dụ 1:

```
#Create a Series from ndarray  
import pandas as pd  
import numpy as np  
data = np.array(['a','b','c','d'])  
s = pd.Series(data)  
print(s)
```

```
0    a  
1    b  
2    c  
3    d  
dtype: object
```

#### Ví dụ 2:

```
#Create a Series from ndarray  
import pandas as pd  
import numpy as np  
data = np.array(['a','b','c','d'])  
s = pd.Series(data,index=[100,101,102,103])  
print(s)
```

```
100    a  
101    b  
102    c  
103    d  
dtype: object
```

## 0.3.3 Series trong Pandas

### Tạo Series từ Dictionary

Ví dụ 1:

```
#Create a Series from dict  
import pandas as pd  
import numpy as np  
data = {'a' : 0., 'b' : 1., 'c' : 2.}  
s = pd.Series(data)  
print(s)
```

```
→ a    0.0  
   b    1.0  
   c    2.0  
dtype: float64
```

Ví dụ 2:

```
#Create a Series from dict  
import pandas as pd  
import numpy as np  
data = {'a' : 0., 'b' : 1., 'c' : 2.}  
s = pd.Series(data,index=['b','c','d','a'])  
print(s)
```

```
→ b    1.0  
   c    2.0  
   d    NaN  
   a    0.0  
dtype: float64
```

## 0.3.3 Series trong Pandas

### Truy cập dữ liệu Series với vị trí

#### Ví dụ 1:

```
#Accessing Data from Series with Position
import pandas as pd
s = pd.Series([1,2,3,4,5],
              index = ['a','b','c','d','e'])
#retrieve the first element
print(s[0])
```

#### Ví dụ 2:

```
#Accessing Data from Series with Position
import pandas as pd
s = pd.Series([1,2,3,4,5],
              index = ['a','b','c','d','e'])
#retrieve the last three element
print(s[-3:])
```

```
1
```

```
#Accessing Data from Series with Position
import pandas as pd
s = pd.Series([1,2,3,4,5],
              index = ['a','b','c','d','e'])
#retrieve the first three element
print(s[:3])
```

```
c    3
d    4
e    5
dtype: int64
```

```
a    1
b    2
c    3
dtype: int64
```

## 0.3.3 Series trong Pandas

### Truy cập dữ liệu Series với index

#### Ví dụ 1:

```
▶ #Retrieve Data Using Label (Index)
import pandas as pd
s = pd.Series([1,2,3,4,5],
              index = ['a','b','c','d','e'])
#retrieve a single element
print(s['a'])
```

⇨ 1

#### Ví dụ 2:

```
▶ #Retrieve Data Using Label (Index)
import pandas as pd
s = pd.Series([1,2,3,4,5],
              index = ['a','b','c','d','e'])
#retrieve multiple elements
print(s[['a','c','d']])
```

⇨ a 1  
c 3  
d 4  
dtype: int64

## 0.3.4 DataFrame trong Pandas

`pandas.DataFrame(data, index, columns, dtype, copy)`

- **data** sẽ nhận giá trị từ nhiều kiểu khác nhau như list, dictionary, ndarray, series,... và cả các DataFrame khác.
- **index** là nhãn chỉ mục hàng của dataframe.
- **columns** là nhãn chỉ mục cột của dataframe.
- **dtype** là kiểu dữ liệu cho mỗi cột.
- **copy** nhận giá trị True/False để chỉ rõ dữ liệu có được copy sang vùng nhớ mới không, mặc định là False

Name	Age	Gender	Rating
Steve	32	Male	3.45
Lia	28	Female	4.6
Vin	45	Male	3.9
Katie	38	Female	2.78

## 0.3.4 DataFrame trong Pandas

### Tạo Dataframe từ List

#### Ví dụ 1:

```
#Create a DataFrame from Lists  
import pandas as pd  
data = [1,2,3,4,5]  
df = pd.DataFrame(data)  
print(df)
```

```
0  
0 1  
1 2  
2 3  
3 4  
4 5
```

#### Ví dụ 2:

```
#Create a DataFrame from Lists  
import pandas as pd  
data = [['Alex',10],['Bob',12],['Clarke',13]]  
df = pd.DataFrame(data,columns=['Name','Age'])  
print(df)
```

```
Name  Age  
0    Alex  10  
1     Bob  12  
2  Clarke  13
```

## 0.3.4 DataFrame trong Pandas

### Tạo DataFrame từ Dict các ndarray / List

Ví dụ 1:

```
#Create a DataFrame from Dict of ndarrays / Lists
import pandas as pd
data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'], 'Age':[28,34,29,42]}
df = pd.DataFrame(data)
print(df)
```

```
→      Name  Age
0      Tom   28
1     Jack   34
2    Steve   29
3    Ricky   42
```

Ví dụ 2:

```
#Create a DataFrame from Dict of ndarrays / Lists
import pandas as pd
data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'], 'Age':[28,34,29,42]}
df = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])
print(df)
```

```
→      Name  Age
rank1    Tom   28
rank2   Jack   34
rank3  Steve   29
rank4  Ricky   42
```

## 0.3.4 DataFrame trong Pandas

### Tạo DataFrame từ List các Dict

Ví dụ 1:



```
#Create a DataFrame from List of Dicts
import pandas as pd
data = [{ 'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data)
print(df)
```

```
→      a    b    c
0     1    2   NaN
1     5   10  20.0
```

Ví dụ 2:



```
#Create a DataFrame from List of Dicts
import pandas as pd
data = [{ 'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data, index=['first', 'second'])
print(df)
```

```
→          a    b    c
first     1    2   NaN
second    5   10  20.0
```

## 0.3.4 DataFrame trong Pandas

### Tạo Dataframe từ Dict các Series

Ví dụ 1:



```
#Create a DataFrame from Dict of Series
import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
      'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)
print(df)
```

⇨

	one	two
a	1.0	1
b	2.0	2
c	3.0	3
d	NaN	4

## 0.3.4 DataFrame trong Pandas

### Đọc dữ liệu từ file.csv



,country,population,area,capital  
BR,Brazil,200,8515767,Brasilia  
RU,Russia,144,17098242,Moscow  
IN,India,1252,3287590,New Delhi  
CH,China,1357,9596961,Beijing  
SA,South Africa,55,1221037,Pretoria

Nội dung của file Brics.csv:

- Số liệu về các quốc gia thuộc khối BRICS bao gồm: tên quốc gia, dân số, diện tích, thủ đô.
- Sử dụng dấu phẩy để ngăn giữa các dữ liệu.
- Mỗi dữ liệu trên 1 dòng.
- Dòng đầu tiên là tên các cột

## 0.3.4 DataFrame trong Pandas

### Đọc dữ liệu từ file.csv

```
▶ import pandas as pd  
df = pd.read_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Brics.csv")  
print(df)
```

```
↪   Unnamed: 0      country  population      area      capital  
0        BR        Brazil          200    8515767    Brasilia  
1        RU       Russia          144   17098242     Moscow  
2        IN        India         1252   3287590  New Delhi  
3        CH        China         1357   9596961    Beijing  
4        SA  South Africa          55   1221037  Pretoria
```

# Đọc dữ liệu và quy định cột 0 dùng làm chỉ số dòng

```
import pandas as pd  
df = pd.read_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Brics.csv", index_col = 0)  
print(df)
```

```
↪      country  population      area      capital  
BR        Brazil          200    8515767    Brasilia  
RU       Russia          144   17098242     Moscow  
IN        India         1252   3287590  New Delhi  
CH        China         1357   9596961    Beijing  
SA  South Africa          55   1221037  Pretoria
```

## 0.3.4 DataFrame trong Pandas

### Đọc dữ liệu từ file.csv với head / tail

```
# Head/Tail in Pandas
import pandas as pd
df = pd.read_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Brics.csv", index_col = 0)
print(df)
print(df.head(3))
print(df.tail(3))
```

```
country population area capital
BR Brazil 200 8515767 Brasilia
RU Russia 144 17098242 Moscow
IN India 1252 3287590 New Delhi
CH China 1357 9596961 Beijing
SA South Africa 55 1221037 Pretoria
country population area capital
BR Brazil 200 8515767 Brasilia
RU Russia 144 17098242 Moscow
IN India 1252 3287590 New Delhi
country population area capital
IN India 1252 3287590 New Delhi
CH China 1357 9596961 Beijing
SA South Africa 55 1221037 Pretoria
```

## 0.3.4 DataFrame trong Pandas

### Truy cập cột theo tên (Column Selection)

```
# Column Selection
import pandas as pd
df = pd.read_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Brics.csv", index_col = 0)
print(df)
print("Column Selection \n", df["capital"])
```

```
country  population      area   capital
BR        Brazil          200    8515767  Brasilia
RU        Russia         144    17098242  Moscow
IN        India          1252   3287590  New Delhi
CH        China          1357   9596961  Beijing
SA  South Africa       55    1221037  Pretoria
Column Selection
BR        Brasilia
RU        Moscow
IN        New Delhi
CH        Beijing
SA        Pretoria
Name: capital, dtype: object
```

## 0.3.4 DataFrame trong Pandas

### Truy cập cột sử dụng iloc (Column Selection)

```
# Column Selection
import pandas as pd
df = pd.read_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Brics.csv", index_col = 0)
print(df)
print("Column Selection \n", df.iloc[:, 1])
```

```
country  population      area   capital
BR        Brazil          200    8515767  Brasilia
RU        Russia          144    17098242  Moscow
IN        India           1252   3287590   New Delhi
CH        China           1357   9596961   Beijing
SA  South Africa         55    1221037   Pretoria
Column Selection
BR        200
RU        144
IN        1252
CH        1357
SA        55
Name: population, dtype: int64
```

## 0.3.4 DataFrame trong Pandas

### Thêm cột (Column Addition)

```
#Column Addition
import pandas as pd
df = pd.read_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Brics.csv", index_col = 0)

df["On_Asia"] = [False, True, True, True, False]

df["density"] = df["population"] / df["area"] * 1000000

print(df)
```

```
BR      Brazil        200   8515767  Brasilia    False  23.485847
RU      Russia       144   17098242  Moscow     True   8.421918
IN      India        1252  3287590  New Delhi  True  380.826076
CH      China        1357  9596961  Beijing    True  141.398928
SA  South Africa     55   1221037  Pretoria  False  45.043680
```

## 0.3.4 DataFrame trong Pandas

### Xóa cột (Column Selection)

#Column Deletion

```
import pandas as pd
df = pd.read_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Brics.csv", index_col = 0)
df_new = df.drop("population", axis=1)
print(df_new)
```

```
country      area    capital
BR        Brazil  8515767  Brasilia
RU        Russia 17098242   Moscow
IN         India  3287590 New Delhi
CH         China  9596961   Beijing
SA South Africa 1221037  Pretoria
```

#Column Deletion

```
import pandas as pd
df = pd.read_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Brics.csv", index_col = 0)
df_new = df.drop(["area", "population"], axis=1)
print(df_new)
```

```
country    capital
BR        Brasilia
RU        Moscow
IN        New Delhi
CH        Beijing
SA South Africa  Pretoria
```

## 0.3.4 DataFrame trong Pandas

### Truy cập dòng (Row Selection)

```
#Row Selection
import pandas as pd
df = pd.read_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Brics.csv", index_col = 0)

df_new = df.loc['RU']
```

```
print(df)
print("Row Selection \n", df_new)
```

```
BR      country      population      area      capital
RU      Brazil          200    8515767    Brasilia
RU      Russia         144    17098242    Moscow
IN      India          1252   3287590  New Delhi
CH      China          1357   9596961    Beijing
SA  South Africa       55    1221037    Pretoria
```

```
Row Selection
country      Russia
population     144
area          17098242
capital        Moscow
Name: RU, dtype: object
```

## 0.3.4 DataFrame trong Pandas

### Truy cập dòng (Row Selection using operator)

```
#Row Selection
import pandas as pd
df = pd.read_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Brics.csv", index_col = 0)

df_new = df[2:4]

print(df)
print("Row Selection using operator \n", df_new)
```

```
BR      country  population      area    capital
RU      Brazil        200  8515767  Brasilia
IN      Russia       144  17098242    Moscow
IN      India         1252  3287590  New Delhi
CH      China         1357  9596961   Beijing
SA  South Africa      55  1221037  Pretoria
Row Selection using operator
      country  population      area    capital
IN      India         1252  3287590  New Delhi
CH      China         1357  9596961   Beijing
```

## 0.3.4 DataFrame trong Pandas

### Thêm dòng (Row Addition)

```
#Row Addition
import pandas as pd
df = pd.read_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Brics.csv", index_col = 0)
df1 = df.drop(["BR", "RU", "IN", "CH"], axis=0)
df_new = df.append(df1)
print(df)
print("df1 \n", df1)
print("df_new \n", df_new)
```

```
BR      country  population      area    capital
RU      Russia       144  17098242    Moscow
IN      India        1252  3287590  New Delhi
CH      China        1357  9596961   Beijing
SA  South Africa       55  1221037  Pretoria
df1
```

```
SA  South Africa       55  1221037  Pretoria
df_new
```

```
BR      country  population      area    capital
RU      Russia       144  17098242    Moscow
IN      India        1252  3287590  New Delhi
CH      China        1357  9596961   Beijing
SA  South Africa       55  1221037  Pretoria
SA  South Africa       55  1221037  Pretoria
```

## 0.3.4 DataFrame trong Pandas

### Xóa dòng (Row Deletion)

```
#Row Deletion
import pandas as pd
df = pd.read_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Brics.csv", index_col = 0)

df_new = df.drop("RU", axis=0)

print(df)
print("Row Deletion \n", df_new)
```

```
BR      country  population      area    capital
RU      Russia       144  17098242     Moscow
IN      India        1252  3287590  New Delhi
CH      China        1357  9596961    Beijing
SA  South Africa       55  1221037   Pretoria
```

```
Row Deletion
BR      country  population      area    capital
RU      Russia       144  17098242     Moscow
IN      India        1252  3287590  New Delhi
CH      China        1357  9596961    Beijing
SA  South Africa       55  1221037   Pretoria
```

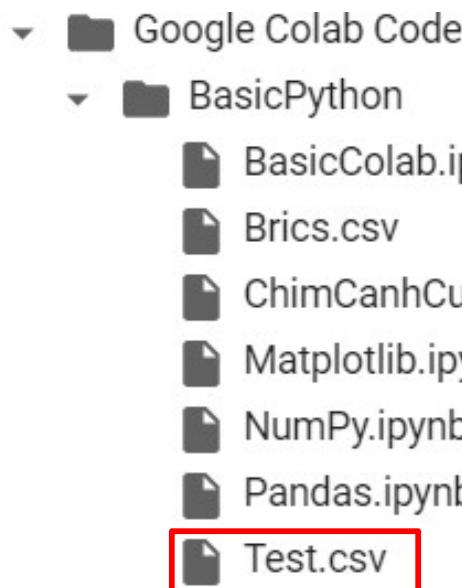
## 0.3.4 DataFrame trong Pandas

### Chuyển từ dataframe sang file.csv

```
#Convert DF to CSV  
import pandas as pd  
df = pd.read_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Brics.csv", index_col = 0)
```

```
df_new = df.drop("RU", axis=0)
```

```
df_new.to_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Test.csv")
```



Địa chỉ lưu file

Tên file

# 0.4 Thư viện Matplotlib

---

---

0.4.1 Giới thiệu

0.4.2 Vẽ biểu đồ các hàm số đơn giản

0.4.3 Một số loại biểu đồ thông dụng

0.4.4 Hiển thị ảnh sử dụng Matplotlib

0.4.5 Kết hợp Pandas với Matplotlib

## 0.4.1 Giới thiệu

- Matplotlib là thư viện chuyên về vẽ biểu đồ, mở rộng từ numpy.
- Có mục tiêu đơn giản hóa tối đa công việc vẽ biểu đồ một cách đơn giản chỉ cần vài dòng lệnh.
- Hỗ trợ rất nhiều loại biểu đồ, đặc biệt là các loại được sử dụng trong nghiên cứu hoặc kinh tế như biểu đồ dòng, đường, tần suất (histograms), phô, tương quan, errorcharts, scatterplots,...
- Cấu trúc của matplotlib gồm nhiều phần, phục vụ cho các mục đích sử dụng khác nhau.



<https://matplotlib.org/index.html>

## 0.4.1 Giới thiệu

- plot(x-axis values, y-axis values) - đồ thị tương quan giữa x và y
- show() - hiển thị biểu đồ
- title("string") - đặt tiêu đề
- xlabel("string") - đặt nhãn cho trục x
- ylabel("string") - đặt nhãn cho trục y
- legend("string") - dùng để tạo chú thích của đồ thị
- figure() - dùng để control các thuộc tính của mức hình
- subplot(nrows, ncols, index) - thêm một subplot vào figure hiện tại
- suptitle("string") - thêm một tiêu đề chung vào hình
- bar(categorical variables, values, color) - tạo đồ thị thanh dọc
- barh(categorical variables, values, color) - tạo biểu đồ thanh ngang
- pie(value, categorical variables) - tạo biểu đồ dạng bánh
- xticks(index, categorical variables) – gán nhãn cho vị trí được đánh dấu hiện tại.

## 0.4.2 Vẽ biểu đồ các hàm số đơn giản

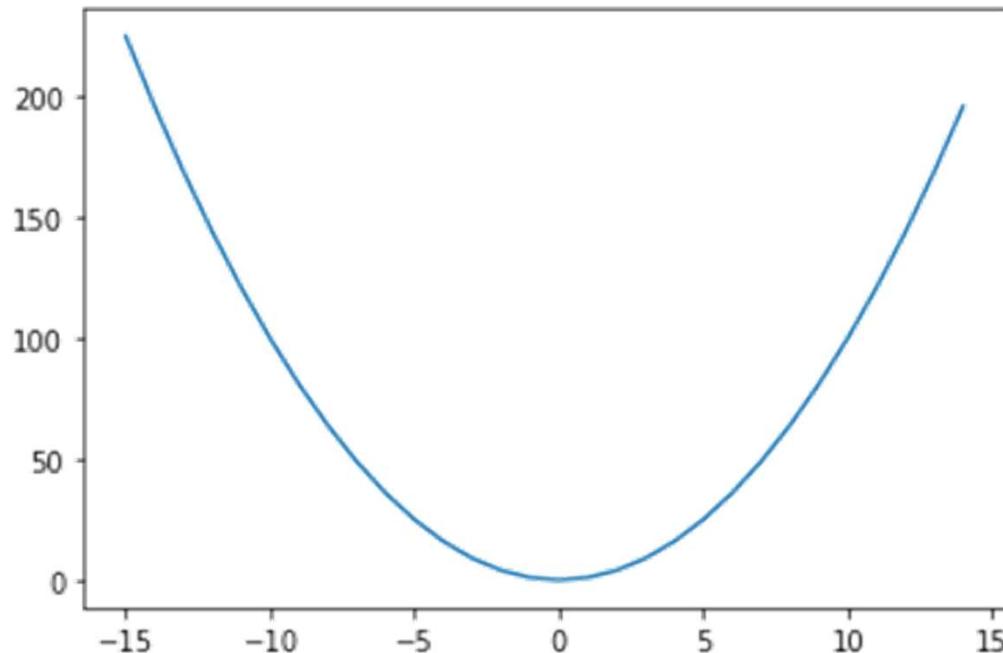
### Biểu đồ $y = x^2$



```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(-15, 15)
y = x * x

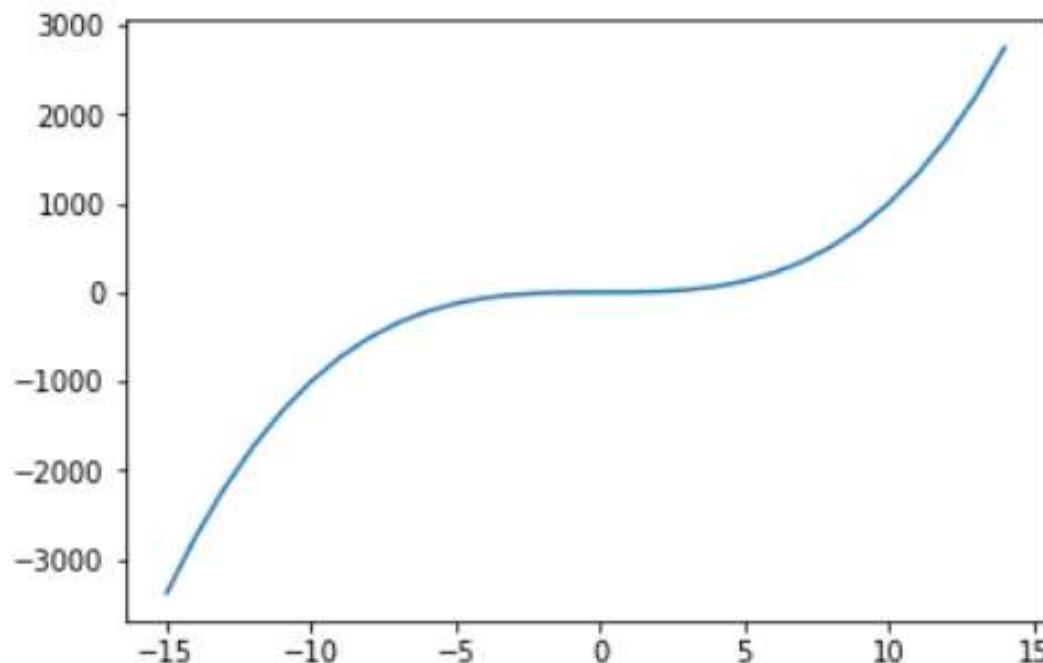
plt.plot(x, y) # vẽ biểu đồ tương quan giữa x và y
plt.show() # hiển thị biểu đồ
```



## 0.4.2 Vẽ biểu đồ các hàm số đơn giản

### Biểu đồ $y = x^3$

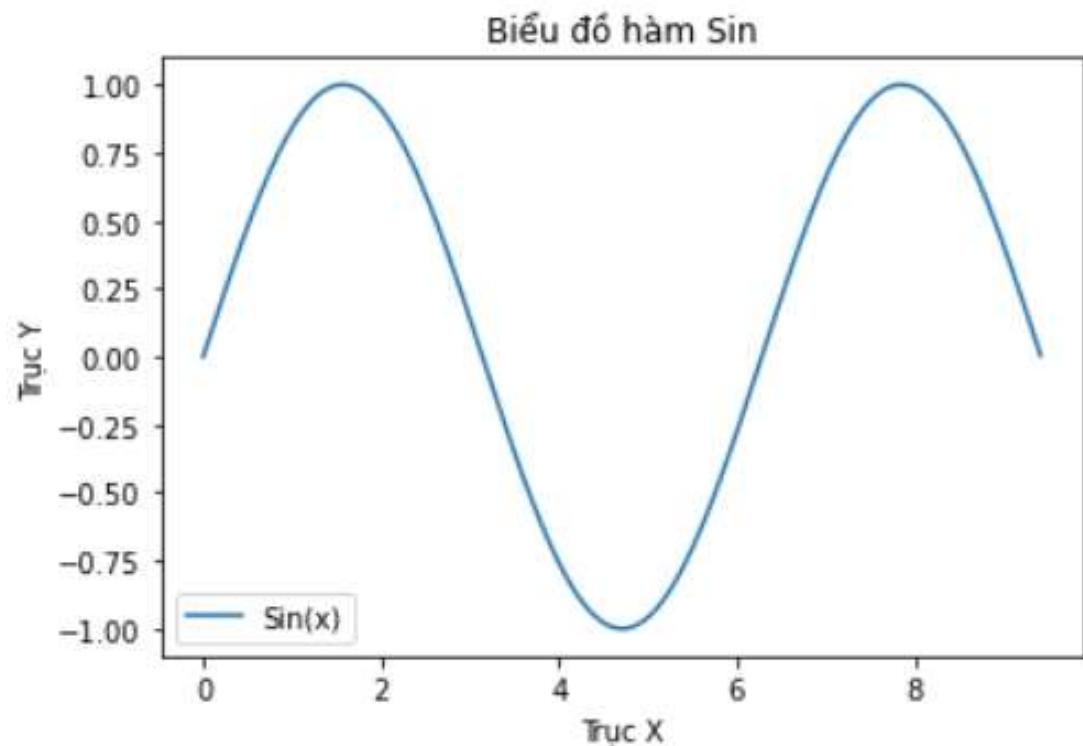
```
▶ import numpy as np  
import matplotlib.pyplot as plt  
  
x = np.arange(-15, 15)  
y = x * x * x  
  
plt.plot(x, y) # vẽ biểu đồ tương quan giữa x và y  
plt.show() # hiển thị biểu đồ
```



## 0.4.2 Vẽ biểu đồ các hàm số đơn giản

### Biểu đồ hàm $\sin$

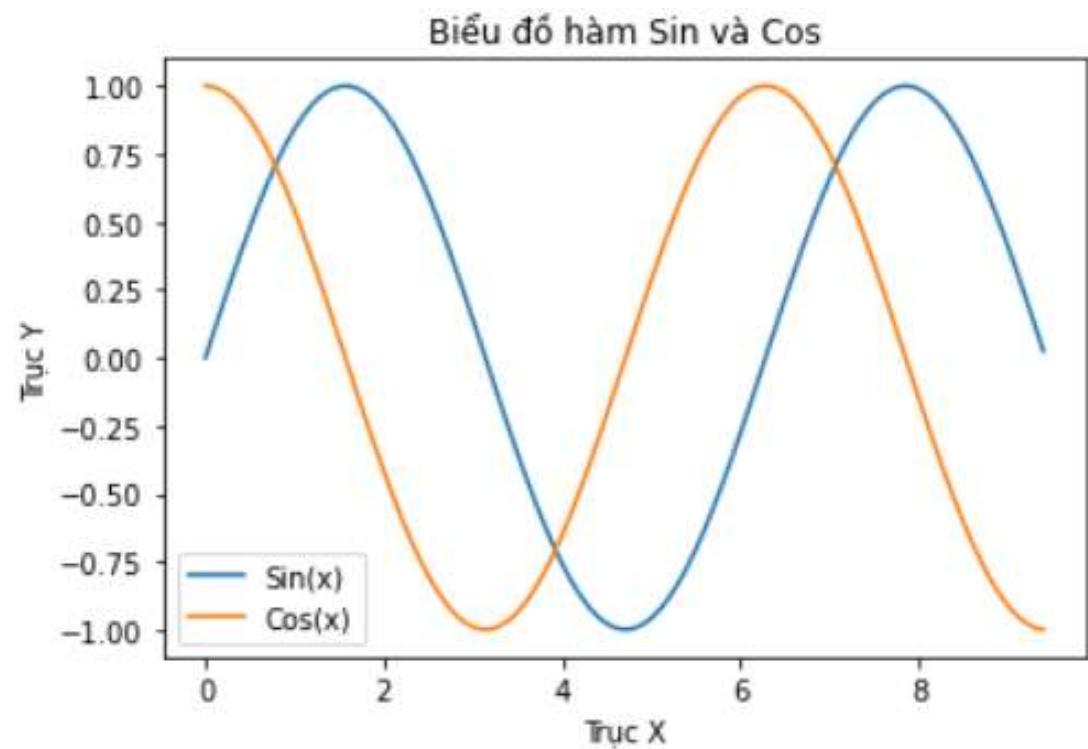
```
import numpy as np  
import matplotlib.pyplot as plt  
  
x = np.arange(0, 3 * np.pi, 0.01)  
y = np.sin(x)  
  
plt.plot(x, y)  
  
plt.xlabel('Trục X')  
plt.ylabel('Trục Y')  
  
plt.title('Biểu đồ hàm Sin')  
  
plt.legend(['Sin(x)'])  
  
plt.show()
```



## 0.4.2 Vẽ biểu đồ các hàm số đơn giản

### Biểu đồ hàm $\sin$ và $\cos$

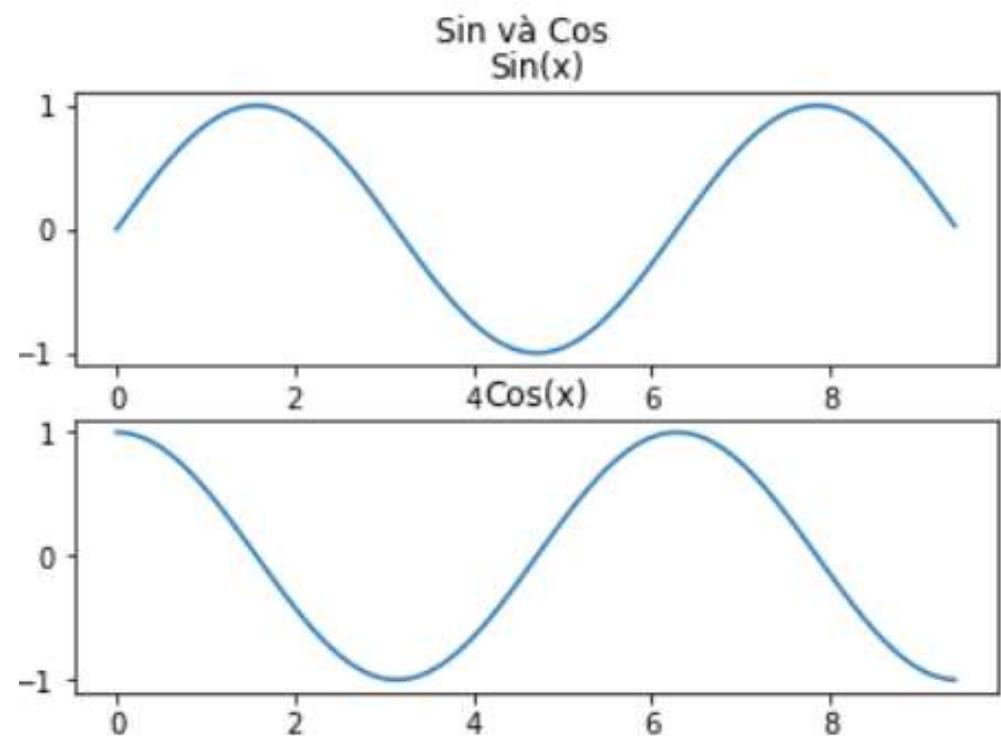
```
▶ import numpy as np  
import matplotlib.pyplot as plt  
  
x = np.arange(0, 3 * np.pi, 0.1)  
y_sin = np.sin(x)  
y_cos = np.cos(x)  
  
plt.plot(x, y_sin)  
plt.plot(x, y_cos)  
  
plt.xlabel('Trục X')  
plt.ylabel('Trục Y')  
  
plt.title('Biểu đồ hàm Sin và Cos')  
  
plt.legend(['Sin(x)', 'Cos(x)'])  
  
plt.show()
```



## 0.4.2 Vẽ biểu đồ các hàm số đơn giản

### Biểu đồ hàm $\sin$ và $\cos$ (subplot)

```
▶ import numpy as np  
import matplotlib.pyplot as plt  
  
x = np.arange(0, 3 * np.pi, 0.1)  
y_sin = np.sin(x)  
y_cos = np.cos(x)  
  
plt.subplot(2, 1, 1)  
plt.plot(x, y_sin)  
plt.title('Sin(x)')  
  
plt.subplot(2, 1, 2)  
plt.plot(x, y_cos)  
plt.title('Cos(x)')  
  
plt.suptitle('Sin và Cos')  
  
plt.show()
```



## 0.4.3 Một số loại biểu đồ thông dụng

### Biểu đồ dạng đường

plot(x, y, color, marker, line...)

#### Color

'b' – blue  
'g' – green  
'r' – red  
'c' – cyan  
'm' – magenta  
'y' – yellow  
'b' – black  
'w' – white

#### Marker

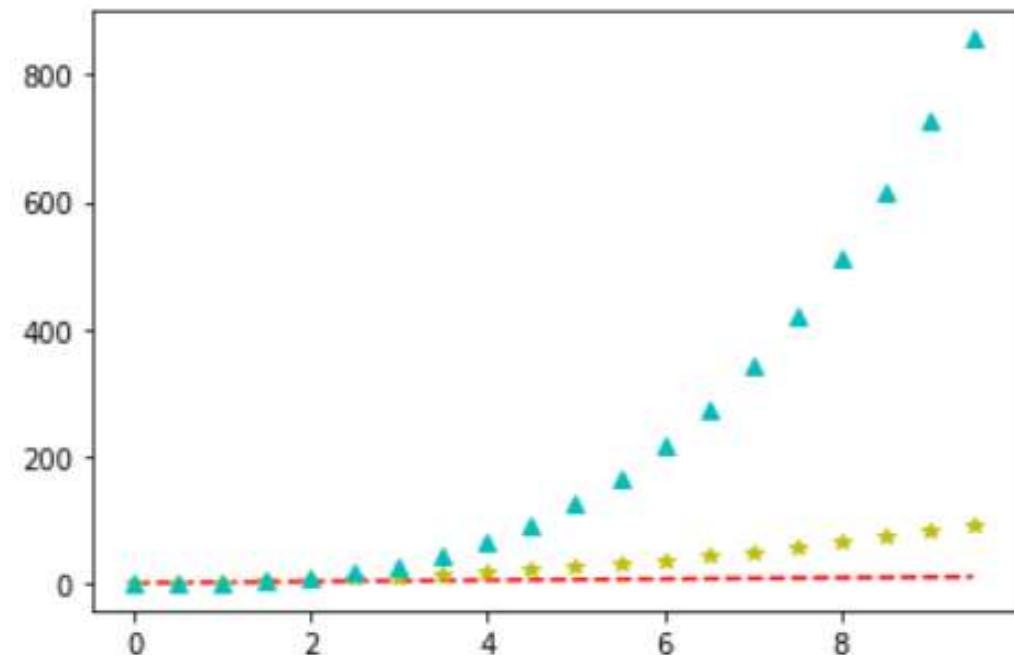
'o' – hình tròn  
'v' – tam giác xuống  
'\*' – ngôi sao  
'.' – chấm  
'p' – ngũ giác  
...

#### Line

'-' – nét liền  
'--' – nét đứt  
'-.-' – gạch chấm  
':' – đường chấm



```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0, 10, 0.5)
y_1 = x
y_2 = x**2
y_3 = x**3
plt.plot(x, y_1, 'r--')
plt.plot(x, y_2, 'y*')
plt.plot(t, y_3, 'c^')
plt.show()
```



## 0.4.3 Một số loại biểu đồ thông dụng

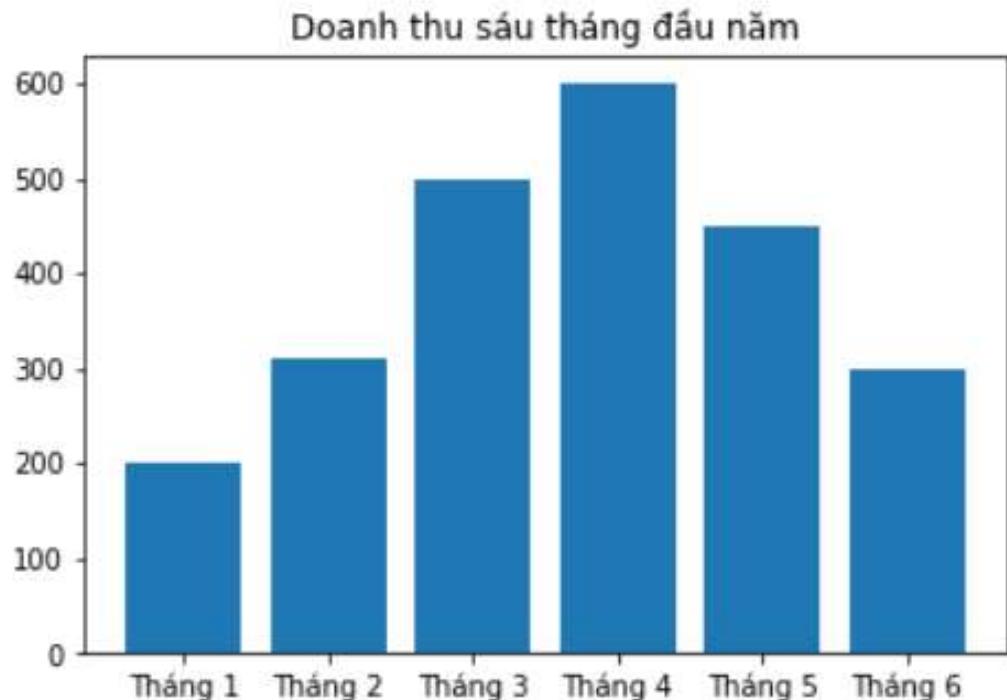
### Biểu đồ dạng cột đứng



```
import numpy as np
import matplotlib.pyplot as plt

x = {'Tháng 1': 200,
      'Tháng 2': 310,
      'Tháng 3': 500,
      'Tháng 4': 600,
      'Tháng 5': 450,
      'Tháng 6': 300 }

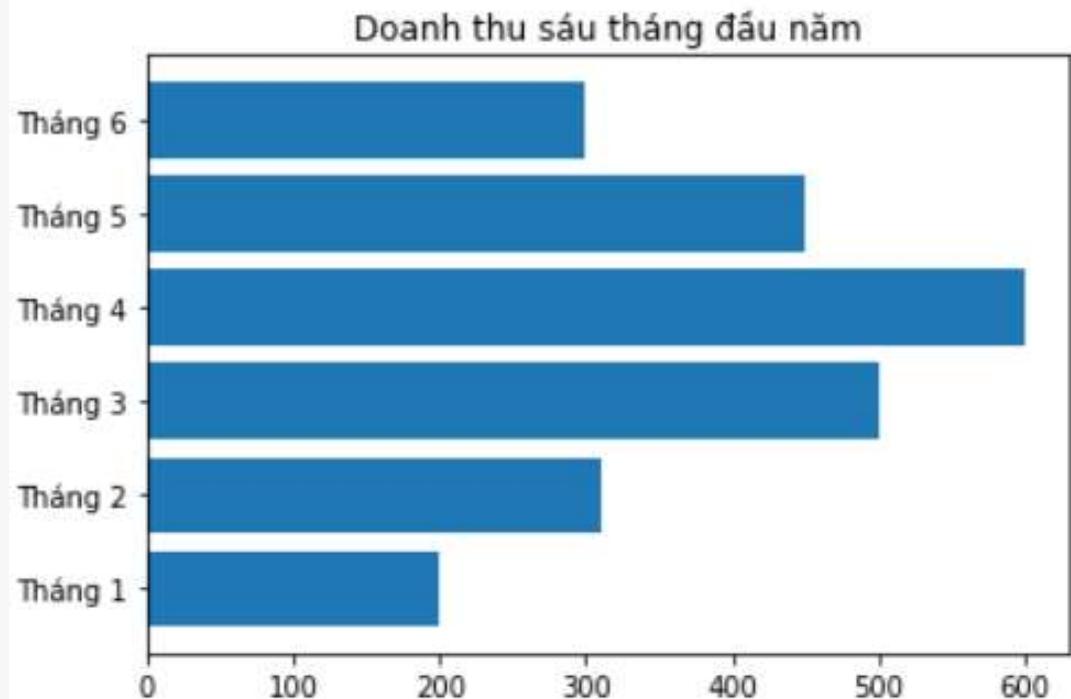
plt.bar(range(len(x)), x.values())
plt.xticks(range(len(x)), x.keys())
plt.title('Doanh thu sáu tháng đầu năm')
plt.show()
```



## 0.4.3 Một số loại biểu đồ thông dụng

### Biểu đồ dạng cột ngang

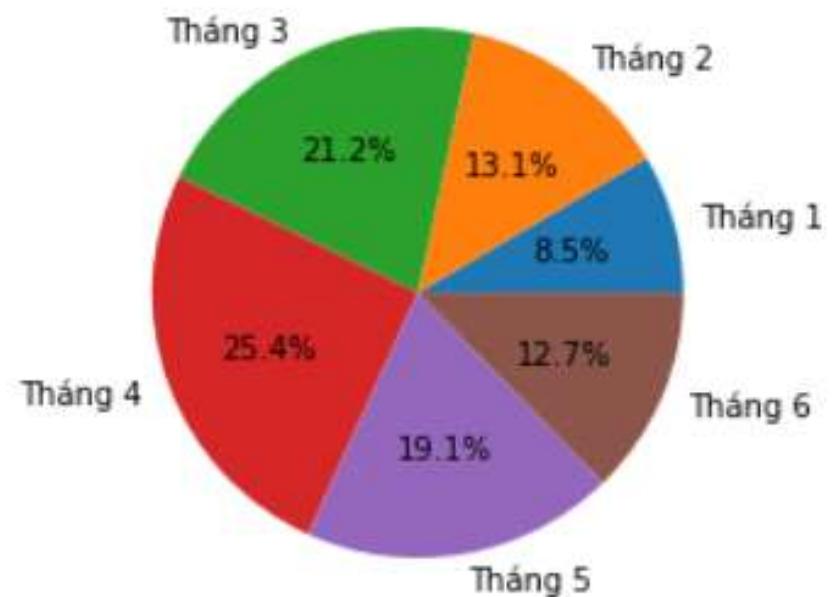
```
import numpy as np  
import matplotlib.pyplot as plt  
  
X = {'Tháng 1': 200,  
      'Tháng 2': 310,  
      'Tháng 3': 500,  
      'Tháng 4': 600,  
      'Tháng 5': 450,  
      'Tháng 6': 300 }  
  
plt.barh(range(len(X)), list(X.values()))  
plt.yticks(range(len(X)), X.keys())  
  
plt.title('Doanh thu sáu tháng đầu năm')  
  
plt.show()
```



## 0.4.3 Một số loại biểu đồ thông dụng

### Biểu đồ dạng bánh

```
▶ import numpy as np  
import matplotlib.pyplot as plt  
  
x = {'Tháng 1': 200,  
     'Tháng 2': 310,  
     'Tháng 3': 500,  
     'Tháng 4': 600,  
     'Tháng 5': 450,  
     'Tháng 6': 300 }  
  
plt.pie(x.values(), labels=x.keys(),  
        autopct='%1.1f%%')  
  
plt.show()
```



## 0.4.4 Hiển thị ảnh sử dụng Matplotlib



```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
img = mpimg.imread("/content/.../ChimCanhCut.jpg")

plt.subplot(221), plt.imshow(img)
plt.title('Hình 1')
plt.xticks([]), plt.yticks([])

plt.subplot(222), plt.imshow(img)
plt.title('Hình 2')
plt.xticks([]), plt.yticks([])

plt.subplot(223), plt.imshow(img)
plt.title('Hình 3')
plt.xticks([]), plt.yticks([])

plt.subplot(224), plt.imshow(img)
plt.title('Hình 4')
plt.xticks([]), plt.yticks([])
plt.show()
```

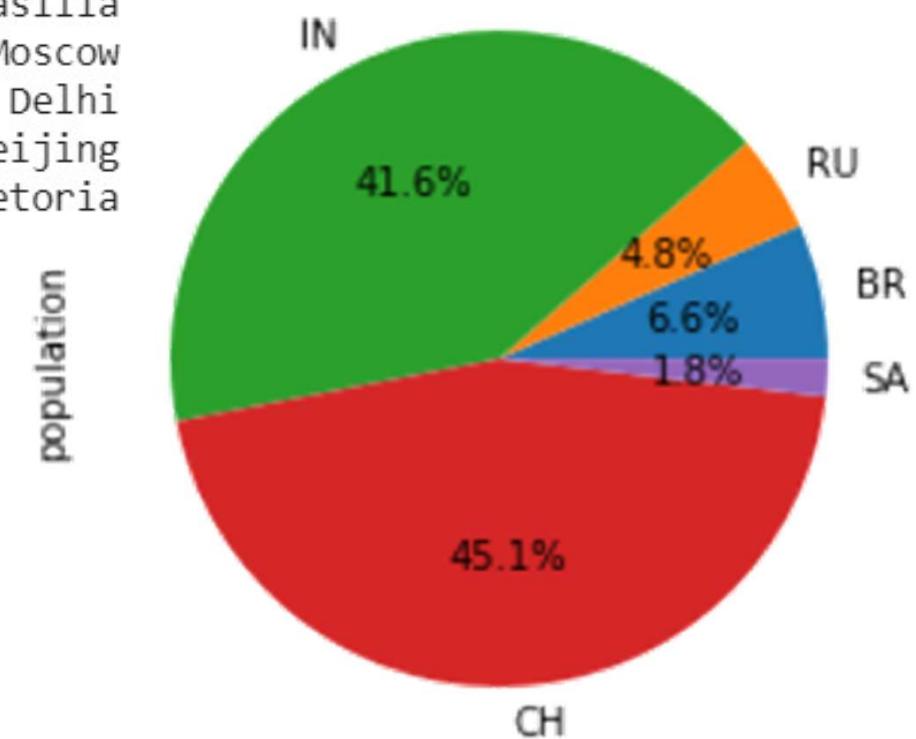


## 0.4.5 Kết hợp Pandas với Matplotlib

```
▶ import pandas as pd  
import matplotlib.pyplot as plt  
df = pd.read_csv("/content/gdrive/My Drive/Google Colab Code/BasicPython/Brics.csv", index_col = 0)  
df.population.plot(kind='pie', autopct='%1.1f%%')  
plt.show()
```

→

	country	population	area	capital
BR	Brazil	200	8515767	Brasilia
RU	Russia	144	17098242	Moscow
IN	India	1252	3287590	New Delhi
CH	China	1357	9596961	Beijing
SA	South Africa	55	1221037	Pretoria



**Thank you !!!**