

BUỔI 1. LÀM QUEN VỚI NGÔN NGỮ LẬP TRÌNH PYTHON

Mục đích:

- Làm quen với cú pháp của ngôn ngữ lập trình python.
- Hiểu về một số cấu trúc dữ liệu lưu trữ thường dùng trong Python.
- Vận dụng các cấu trúc đó trong việc phân tích dữ liệu học máy.

1.1. Giới thiệu Python

1.1.1. Giới thiệu về ngôn ngữ lập trình Python

Python là ngôn ngữ lập trình rất phổ biến hiện nay, được ra mắt chính thức vào năm 1991. Nó thường được sử dụng để tính toán, xây dựng ứng dụng, phát triển website (server-side).

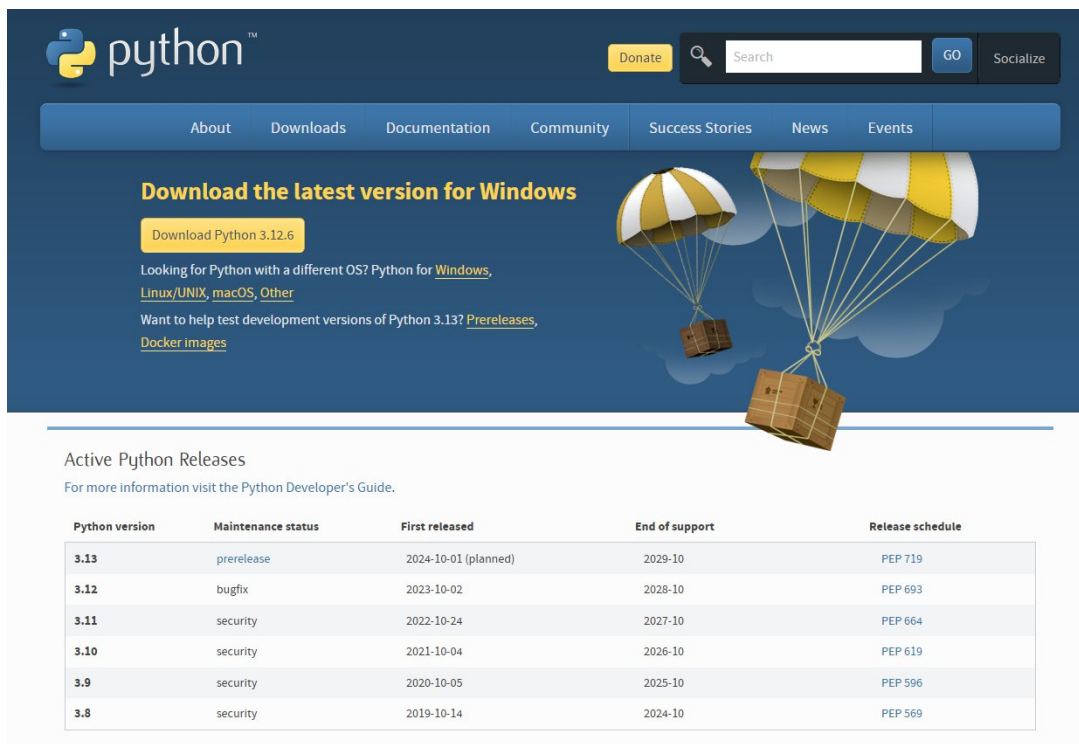
Python có thể được làm việc trên nhiều nền tảng hệ điều hành khác nhau Windows, Mac, Linux,...

Đây là ngôn ngữ lập trình có cú pháp đơn giản và gần giống như ngôn ngữ tiếng Anh, giúp cho lập trình viên có thể viết được các chương trình gọn hơn so với các ngôn ngữ lập trình khác.

Một số công cụ lập trình cho ngôn ngữ Python như là: Pycharm, Netbeans, Eclipse,... Trong bài giảng thực hành của học phần này, chúng ta sẽ sử dụng công cụ Pycharm để minh họa cho các ví dụ cũng như thực hiện các bài tập.

1.1.2. Hướng dẫn cài đặt Python

Download Python : Truy cập vào trang web : <https://www.python.org/> để download Python. Chú ý : Cài đặt phiên bản phù hợp với hệ điều hành đang sử dụng.



Python version	Maintenance status	First released	End of support	Release schedule
3.13	prerelease	2024-10-01 (planned)	2029-10	PEP 719
3.12	bugfix	2023-10-02	2028-10	PEP 693
3.11	security	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569

1.1.3. Thực thi chương trình Python

Sau khi cài đặt, chúng ta có thể kiểm tra phiên bản Python đang sử dụng bằng cách:
Trên Windows, mở bảng lệnh command line lên và gõ vào:

```
python --version
```

Nếu như Python đã có cài đặt sẵn trên máy tính thì kết quả sẽ như hình sau:

```
Command Prompt
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PC>python --version
Python 3.11.7

C:\Users\PC>
```

Chúng ta có thể thực thi câu lệnh Python bằng các cách sau:

- Cách 1: Thực thi trực tiếp trên bảng lệnh command line

Ví dụ: Muốn in ra dòng chữ “Hello world”, chúng ta có thể thực hiện trên command line như sau:

```
Command Prompt - python
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PC>python --version
Python 3.11.7

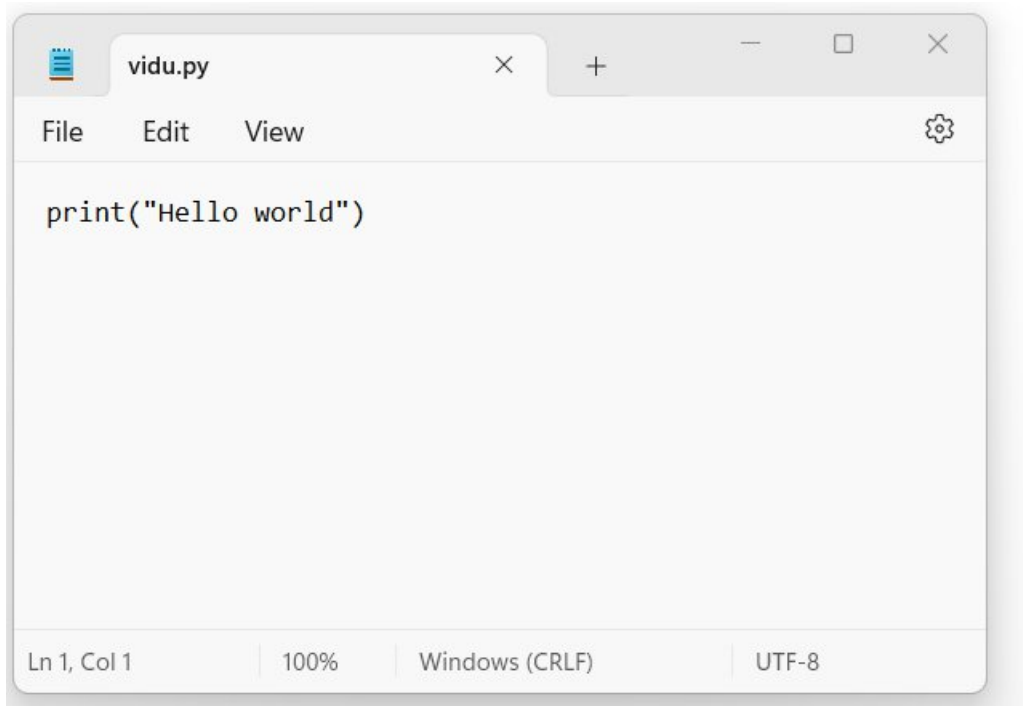
C:\Users\PC>python
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world")
Hello world
>>>
```

- Cách 2: Thực thi trên trình soạn thảo Text Editor

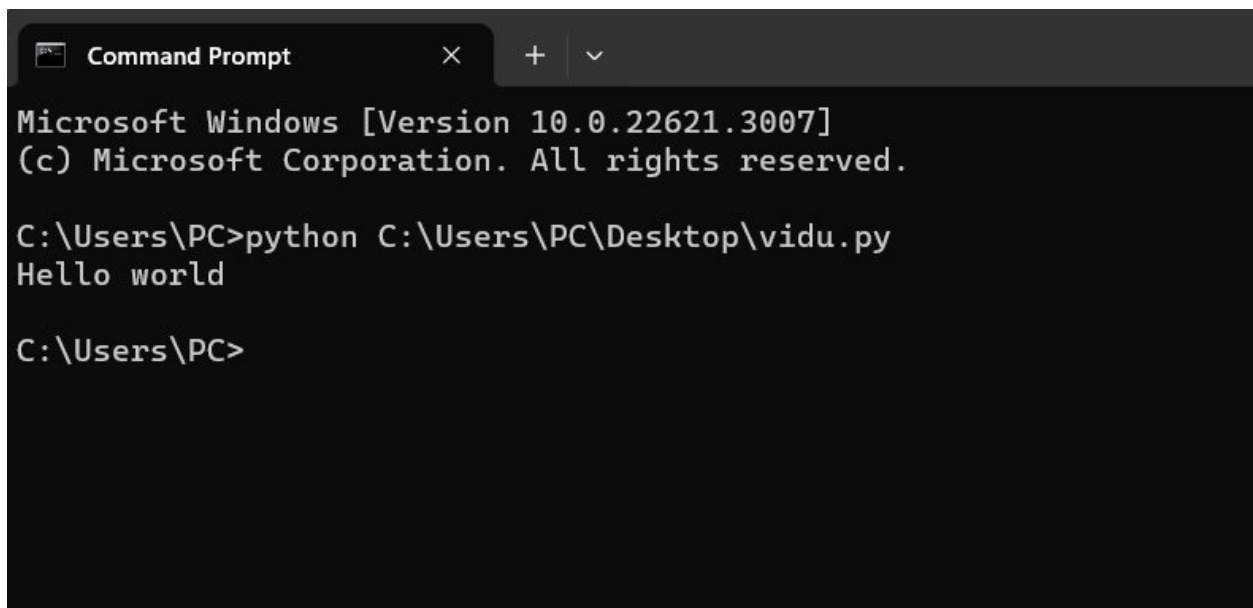
Soạn câu lệnh trên một trình soạn thảo Text Editor nào đó, lưu tập tin lại với đuôi mở rộng (.py). Sau đó thực thi tập tin bằng cách gọi lệnh sau:

```
python <path-to-file>\<file-name>.py
```

Ví dụ: Tương tự khi muốn in ra dòng chữ “Hello world”. Đầu tiên, ta mở trình soạn thảo lên và gõ câu lệnh Python vào như sau:



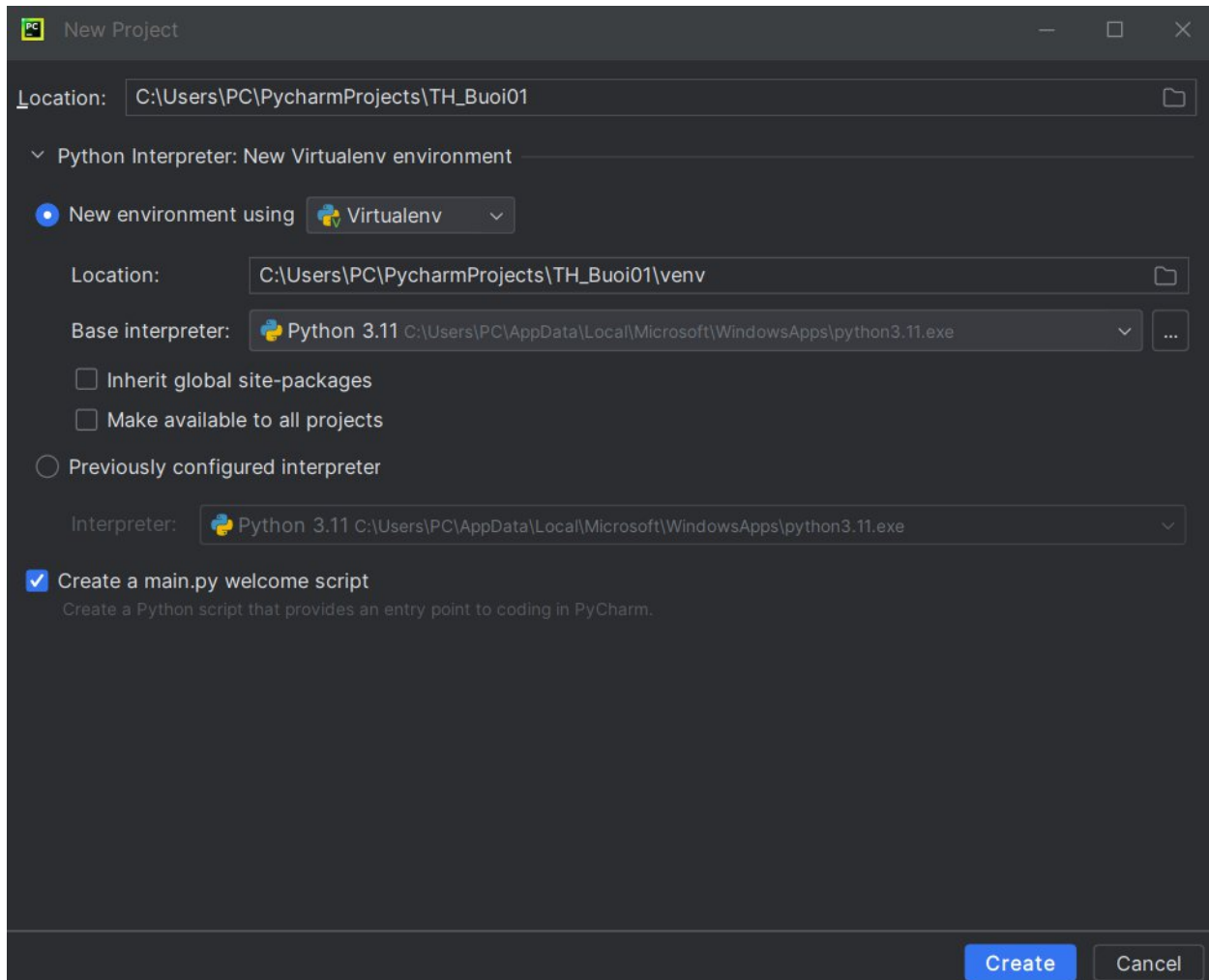
Sau đó, chúng ta thực thi tập tin này trên bảng lệnh command line như sau:



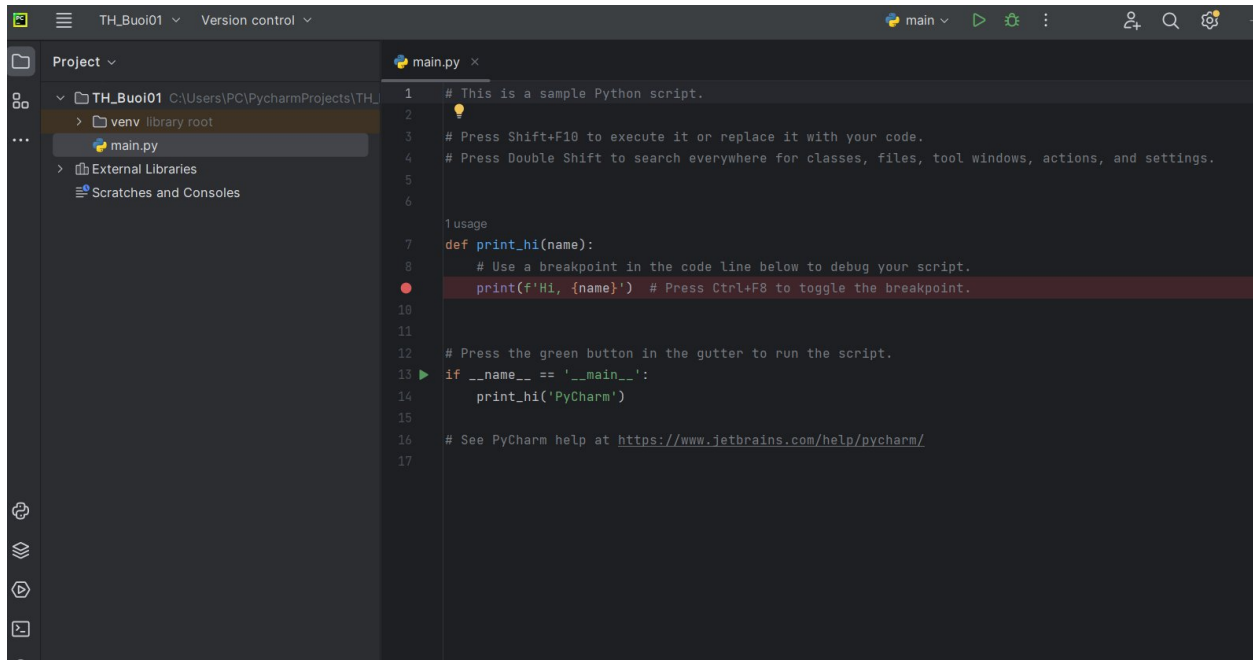
- Cách 3: Thực thi trên công cụ lập trình Pycharm

Về cách cài đặt Pycharm thì tương tự như cài đặt các phần mềm khác, không quá phức tạp để thực hiện. Lưu ý, phiên bản Pycharm chúng ta lựa chọn sẽ là Pycharm Community.

Sau khi cài đặt xong, mở Pycharm lên và tạo project để lập trình như sau:

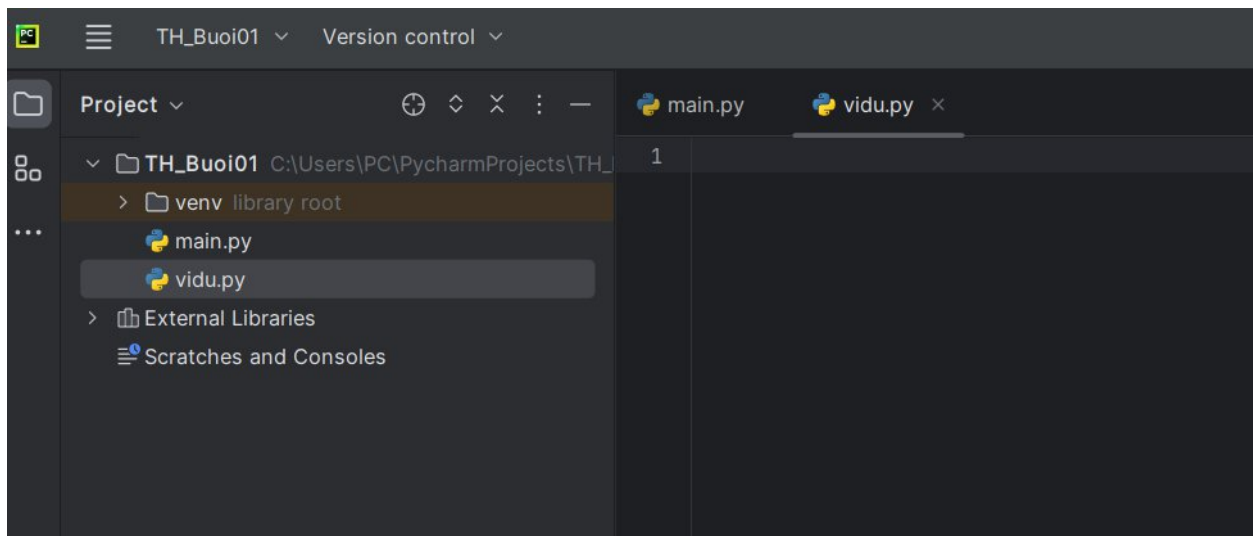


Tại giao diện “New Project”, chúng ta có thể đặt tên cho project của mình. Ngoài ra, chúng ta còn có thể chọn phiên bản Python để thực thi nếu như trong máy tính có nhiều phiên bản khác nhau. Sau khi cấu hình xong, chúng ta chọn “Create” để tạo project mới. Giao diện khi tạo xong sẽ như sau:

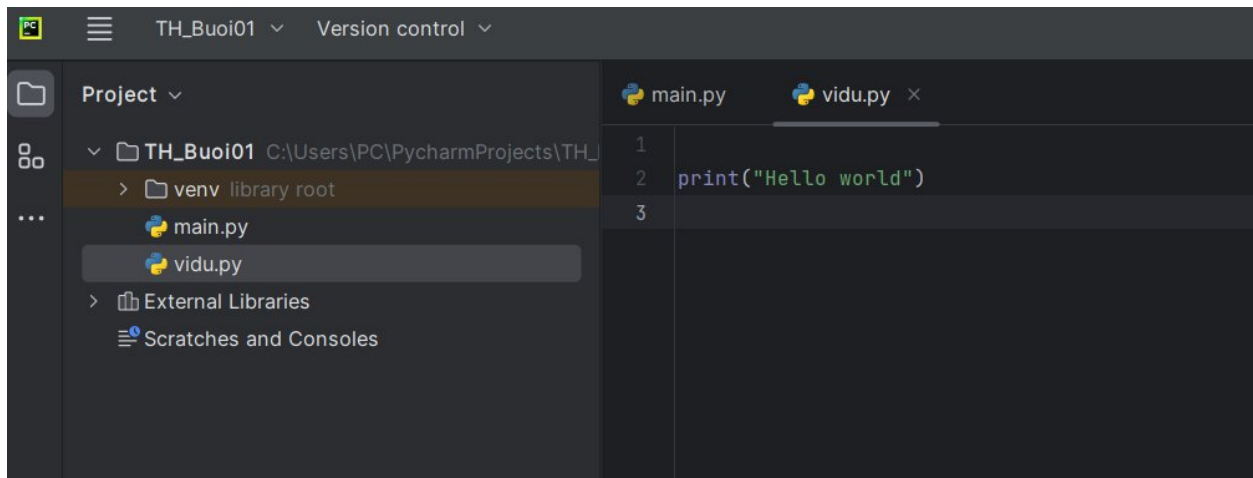


Thông thường, ở bước cấu hình bên trên, chúng ta thường chọn vào tạo sẵn tập tin “main.py” nên công cụ sẽ tạo sẵn cho chúng ta để thực hiện lập trình. Chúng ta có thể tạo thêm file khác bằng cách chuột phải vào tên project và chọn New -> Python File.

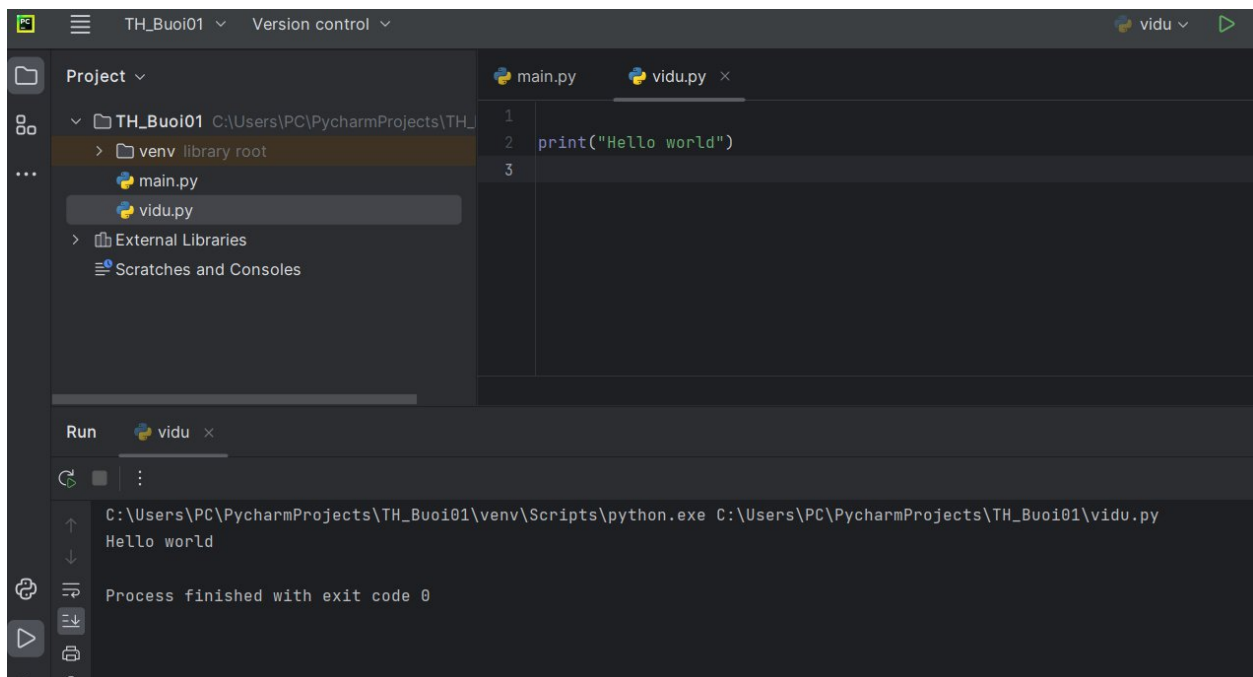
Ví dụ, chúng ta tạo thêm tập tin tên vidu.py như hình sau:



Tiến hành viết lệnh trên tập tin mới này để in ra dòng chữ “Hello world” như sau:



Để thực thi câu lệnh này, chúng ta có thể click chuột phải vào tập tin và chọn “Run”.
Kết quả như sau:



- Cách 4: Thực thi trên Google Colaboratory

Colaboratory, hay viết tắt là "Colab", cho phép bạn viết và thực thi Python trong trình duyệt của mình, với:

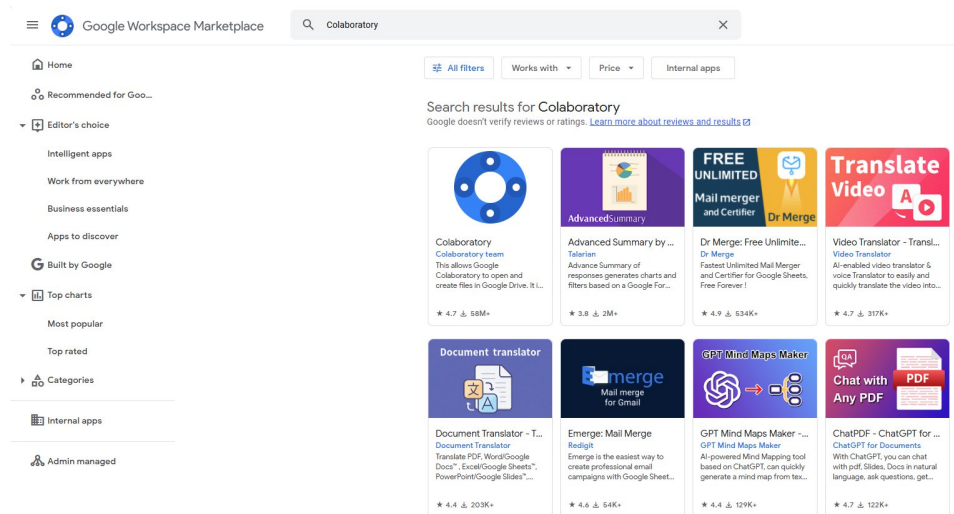
- Không cần cấu hình

- Truy cập miễn phí vào GPU
- Chia sẻ dễ dàng

❖ Hướng dẫn cài đặt

Truy cập vào đường dẫn sau: <https://workspace.google.com/u/1/marketplace>

➤ Tìm kiếm từ khóa “Colaboratory”



➤ Nhấn “Install”




Colaboratory

This allows Google Colaboratory to open and create files in Google Drive. It is automatically installed on first use; uninstalling this will not prevent access to Colaboratory.

By: [Colaboratory team](#)

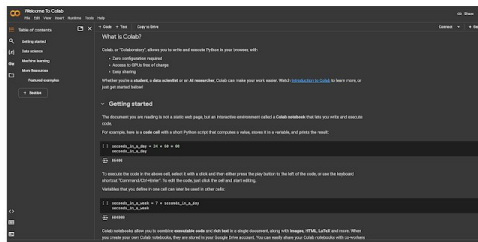
Listing updated: May 25, 2024

[Install](#)

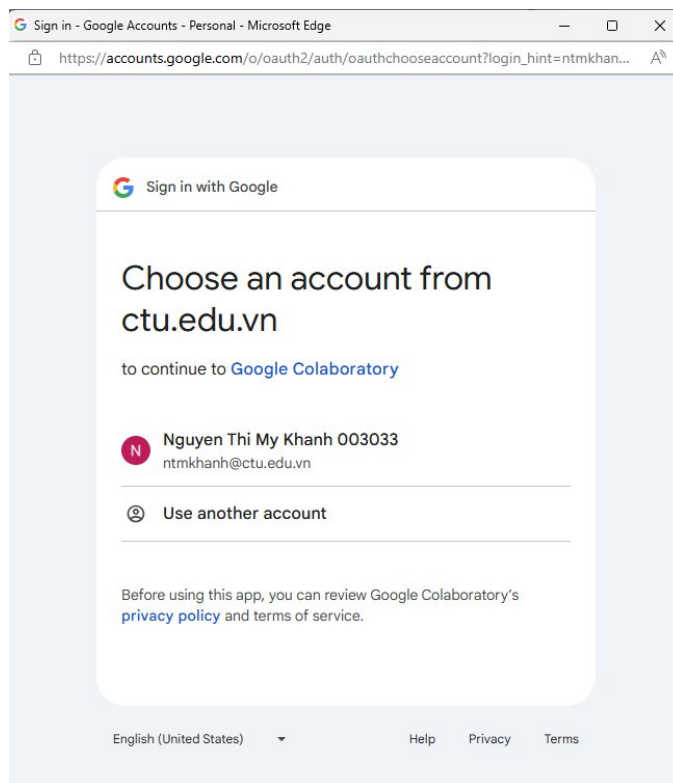
Works with: 

★★★★★ 3,609 ⓘ ⬇ 58M+

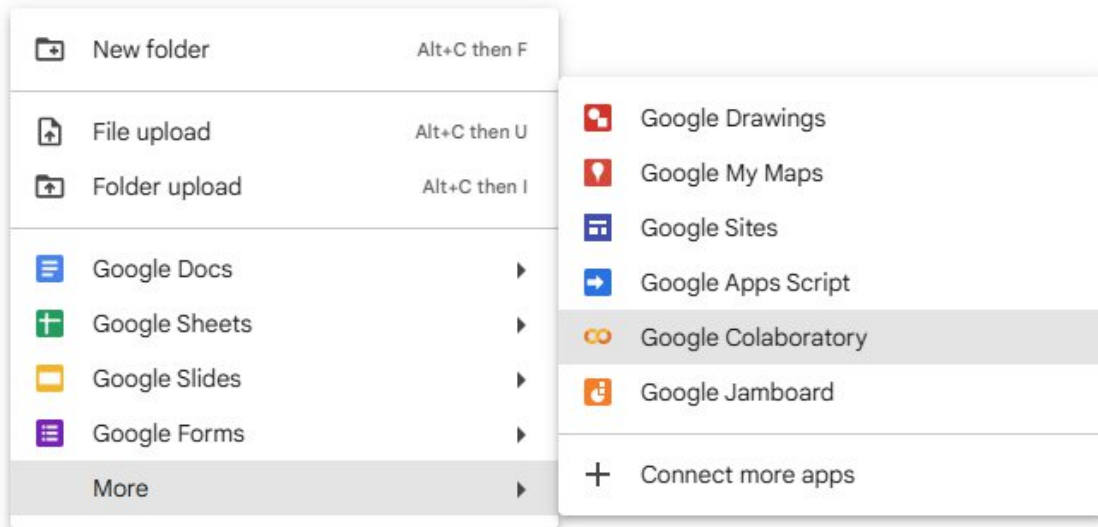
[Overview](#)
[Permissions](#)
[Reviews](#)



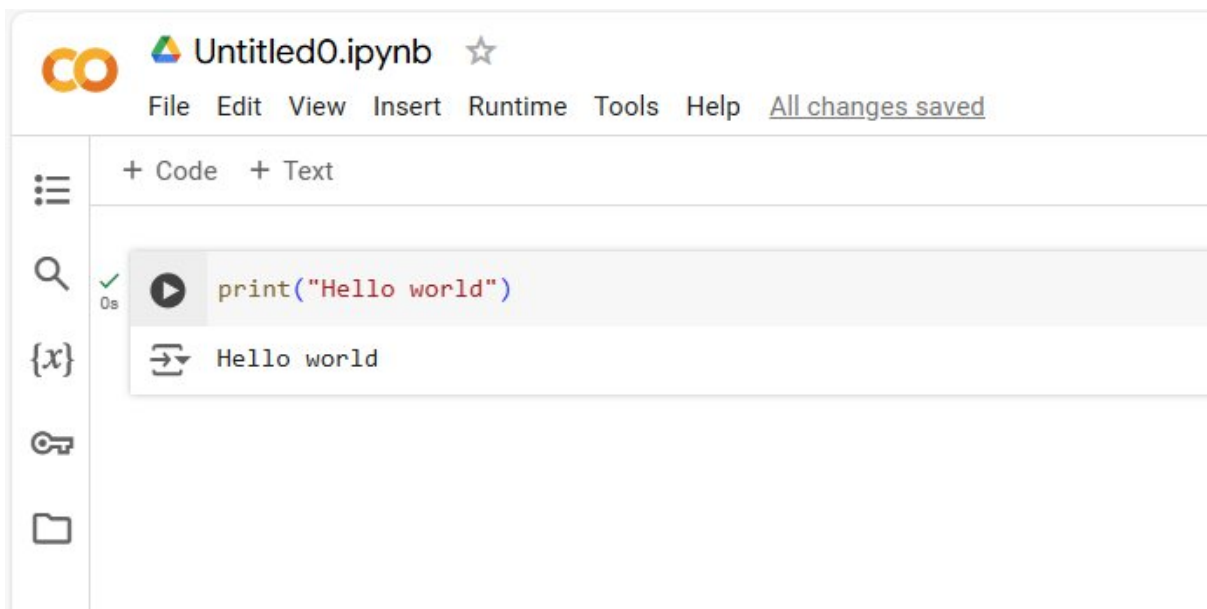
➤ Xác thực tài khoản để hoàn thành cài đặt.



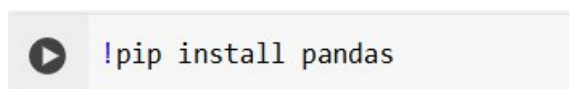
- Sau đó, truy cập vào Google Drive. Nhấn chuột phải -> More -> Google Colaboratory



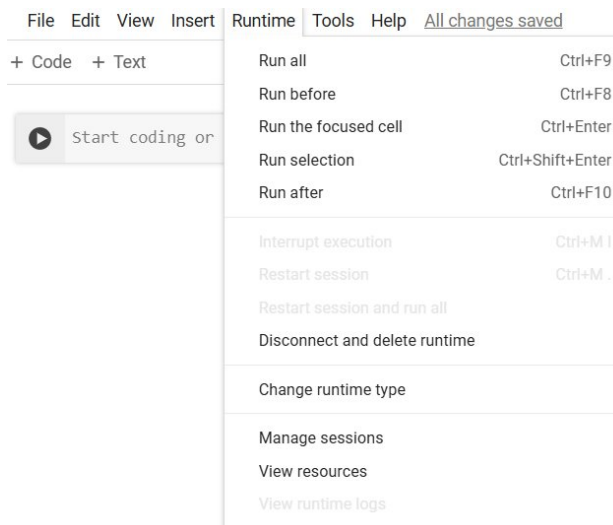
Sau khi mở Google Colab. Gõ code Python vào khối lệnh. Ví dụ:



hoặc sử dụng dấu chấm than (“ ! ”) đầu hàng để thay đổi khối code thành môi trường bash thực thi mã Linux. Ví dụ:



Ngoài ra, để sử dụng GPU miễn phí của Google cung cấp, click “Runtime” -> “Change Runtime Type”.



Có 3 tùy chọn trong “Hardware Accelerator”, chọn “T4 GPU” --> Save

Change runtime type

Runtime type

Python 3 ▼

Hardware accelerator ?

☐ CPU ☒ T4 GPU ☐ A100 GPU ☐ L4 GPU
☐ TPU v2-8

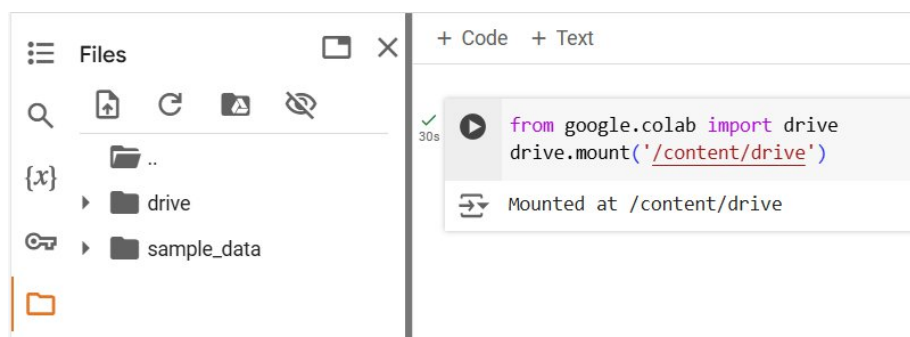
Want access to premium GPUs? [Purchase additional compute units](#)
Your organization can purchase Colab Pro and Colab Pro+ licenses in bulk.
[Learn more](#)

Cancel Save

Một lợi thế của việc sử dụng Google Colab là kết nối với các dịch vụ khác của Google như Google Drive rất đơn giản. Bằng cách mounting google drive, các tệp làm việc có thể được lưu trữ vĩnh viễn. Sau khi thực hiện khối code sau, hãy đăng nhập vào tài khoản Google và xác thực để kết thúc quá trình.

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

Sau khi mount, nội dung của Google Drive sẽ nằm trong thư mục  MyDrive và tất cả các thay đổi sẽ được đồng bộ hóa.

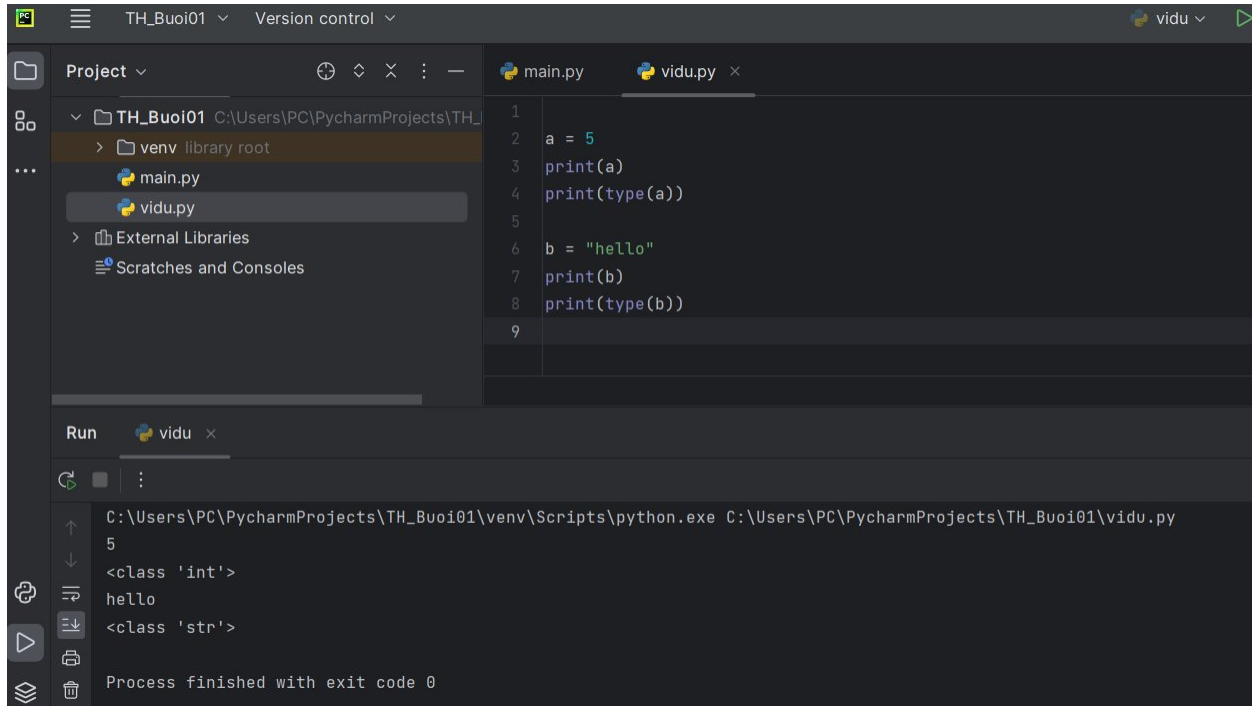


1.2. Giới thiệu ngôn ngữ lập trình Python

1.2.1. Biến và kiểu dữ liệu

Biến trong python không cần khai báo kiểu dữ liệu, kiểu dữ liệu sẽ được “ngầm hiểu” sau khi gán giá trị cho biến. Chúng ta có thể kiểm tra kiểu dữ liệu của biến bằng câu lệnh **type()**.

Ví dụ như sau:



The screenshot shows the PyCharm IDE interface. The top toolbar includes icons for file operations and a 'Run' button. The left sidebar shows the project structure for 'TH_Buoi01', including a 'venv' directory and files 'main.py' and 'vidu.py'. The main editor window displays the code in 'vidu.py':

```
1  
2 a = 5  
3 print(a)  
4 print(type(a))  
5  
6 b = "hello"  
7 print(b)  
8 print(type(b))  
9
```

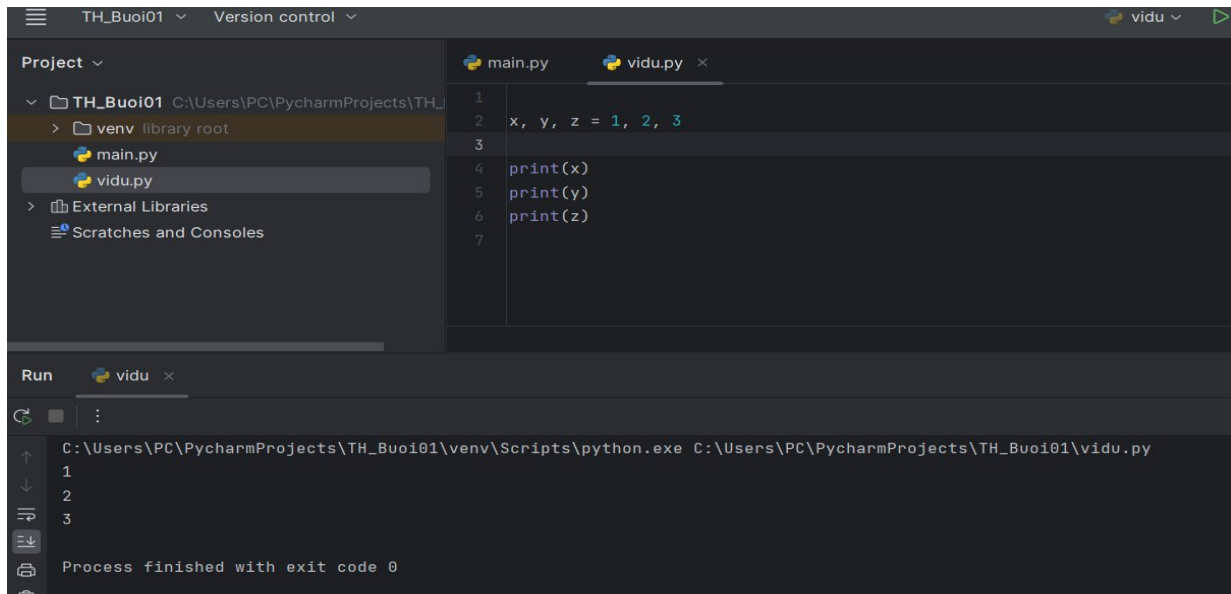
Below the editor, the 'Run' tab shows the execution command and output:

```
C:\Users\PC\PycharmProjects\TH_Buoi01\venv\Scripts\python.exe C:\Users\PC\PycharmProjects\TH_Buoi01\vidu.py  
5  
<class 'int'>  
hello  
<class 'str'>
```

The status bar at the bottom indicates 'Process finished with exit code 0'.

Quy ước đặt tên biến trong python tương tự như các ngôn ngữ lập trình khác (trong đó có ngôn ngữ lập trình C).

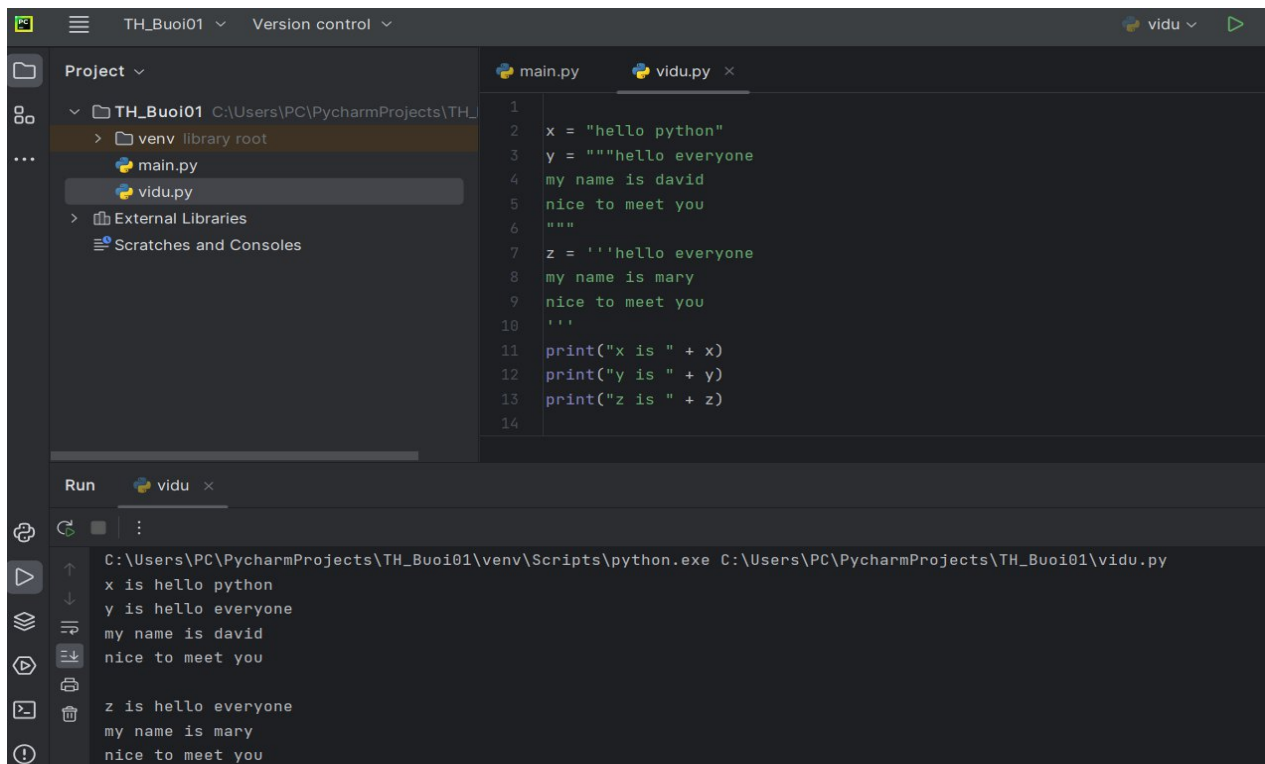
Chúng ta có thể khai báo và gán giá trị cùng lúc nhiều biến trên cùng 1 dòng như sau:



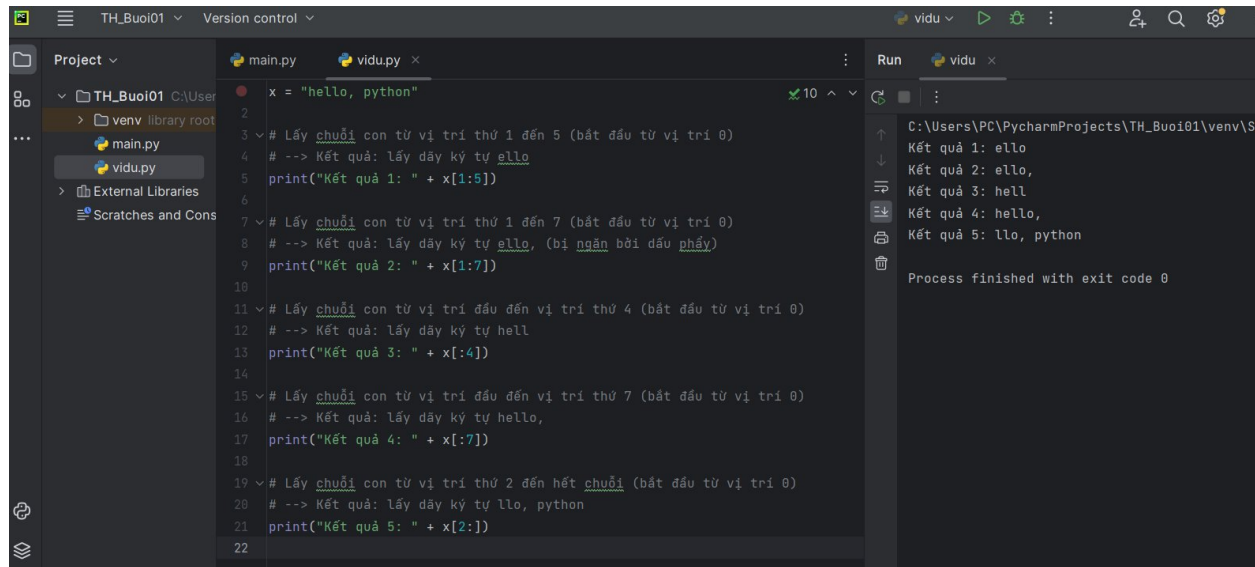
Một số kiểu dữ liệu của python như sau:

- Kiểu dữ liệu text: str

Chúng ta có thể khai báo biến lưu trữ dữ liệu string trên 1 dòng hoặc nhiều dòng như sau:



Chúng ta có thể trả về một dãy ký tự trong chuỗi (có ngăn cách bởi dấu phẩy) bằng việc cắt chuỗi (slice string) theo ví dụ sau:



```
x = "hello, python"

# Lấy chuỗi con từ vị trí thứ 1 đến 5 (bắt đầu từ vị trí 0)
# --> Kết quả: lấy dãy ký tự 'ello'
print("Kết quả 1: " + x[1:5])

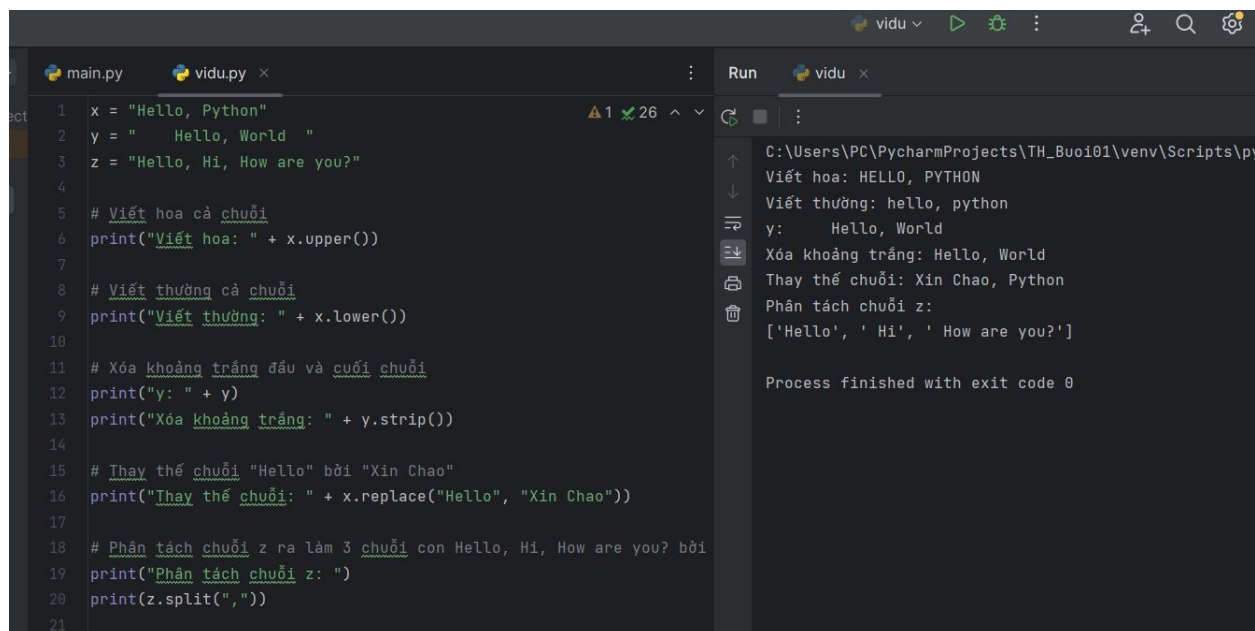
# Lấy chuỗi con từ vị trí thứ 1 đến 7 (bắt đầu từ vị trí 0)
# --> Kết quả: lấy dãy ký tự 'ello,', (bị ngăn bởi dấu phẩy)
print("Kết quả 2: " + x[1:7])

# Lấy chuỗi con từ vị trí đầu đến vị trí thứ 4 (bắt đầu từ vị trí 0)
# --> Kết quả: lấy dãy ký tự 'hell'
print("Kết quả 3: " + x[:4])

# Lấy chuỗi con từ vị trí đầu đến vị trí thứ 7 (bắt đầu từ vị trí 0)
# --> Kết quả: lấy dãy ký tự 'hello,'
print("Kết quả 4: " + x[:7])

# Lấy chuỗi con từ vị trí thứ 2 đến hết chuỗi (bắt đầu từ vị trí 0)
# --> Kết quả: lấy dãy ký tự 'llo, python'
print("Kết quả 5: " + x[2:])
```

Một số câu lệnh dùng để chỉnh sửa chuỗi thường dùng như upper(), lower(), strip(), replace(), split(). Có thể hình dung trong ví dụ sau:



```
x = "Hello, Python"
y = "  Hello, World  "
z = "Hello, Hi, How are you?"

# Viết hoa cả chuỗi
print("Viết hoa: " + x.upper())

# Viết thường cả chuỗi
print("Viết thường: " + x.lower())

# Xóa khoảng trắng đầu và cuối chuỗi
print("y: " + y)
print("Xóa khoảng trắng: " + y.strip())

# Thay thế chuỗi "Hello" bởi "Xin Chao"
print("Thay thế chuỗi: " + x.replace("Hello", "Xin Chao"))

# Phân tách chuỗi z ra làm 3 chuỗi con Hello, Hi, How are you? bởi
print("Phân tách chuỗi z: ")
print(z.split(","))
```

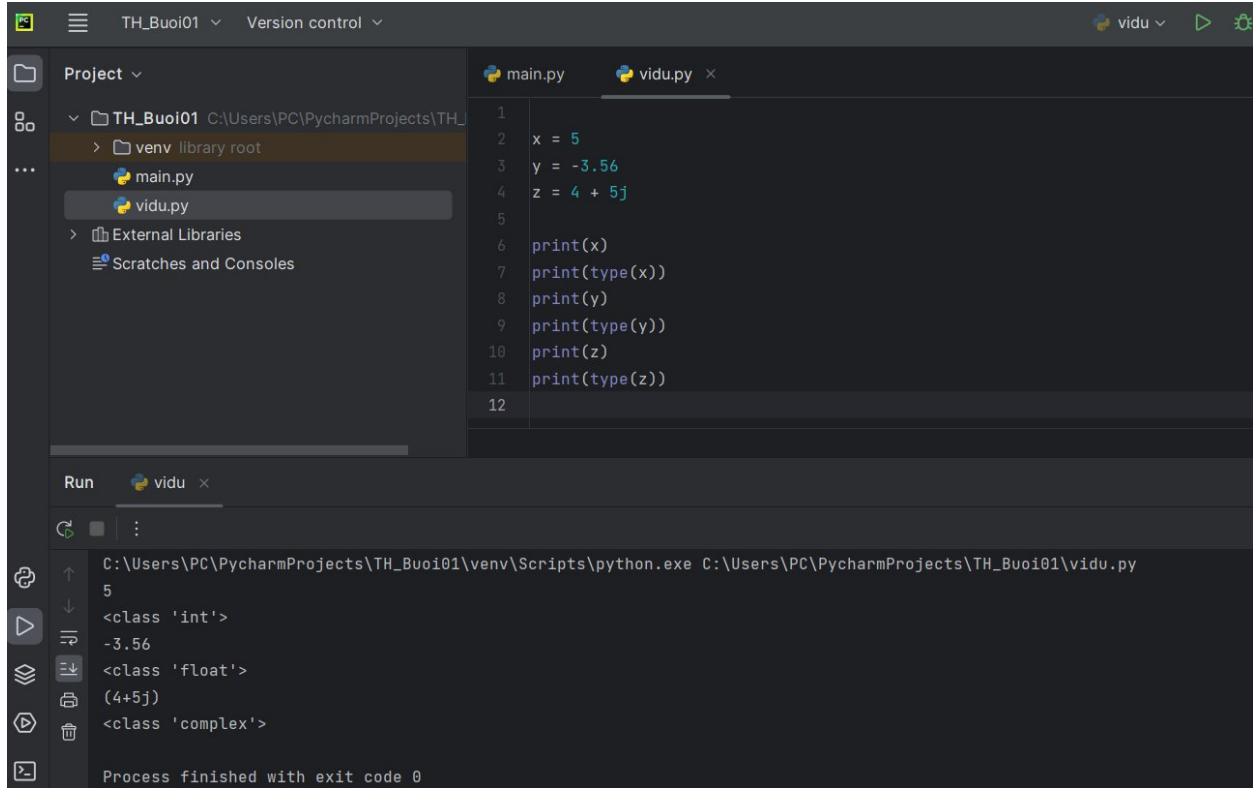
Các ký tự điều khiển như \n \r \t \\ \' cơ bản giống với ngôn ngữ lập trình C.

Ngoài ra, chúng ta còn có nhiều hàm xử lý chuỗi khác, các bạn có thể tìm hiểu thêm tại link sau:

https://www.w3schools.com/python/python_strings_methods.asp

- Kiểu dữ liệu số: int (số nguyên), float (số thực), complex (số phức)

Chúng ta có thể khai báo các biến lưu trữ dữ liệu số qua ví dụ sau:



The screenshot shows the PyCharm IDE interface. On the left, the 'Project' view displays the file structure of 'TH_Buoi01', including a 'venv' directory and files 'main.py' and 'vidu.py'. The 'vidu.py' file is open in the editor, showing the following code:

```
1  
2 x = 5  
3 y = -3.56  
4 z = 4 + 5j  
5  
6 print(x)  
7 print(type(x))  
8 print(y)  
9 print(type(y))  
10 print(z)  
11 print(type(z))  
12
```

Below the editor, the 'Run' tab shows the execution command: `C:\Users\PC\PycharmProjects\TH_Buoi01\venv\Scripts\python.exe C:\Users\PC\PycharmProjects\TH_Buoi01\vidu.py`. The output of the script is displayed in the console:

```
5  
<class 'int'>  
-3.56  
<class 'float'>  
(4+5j)  
<class 'complex'>
```

The console also indicates 'Process finished with exit code 0'.

- Kiểu Boolean: bool

- Các kiểu dữ liệu lưu trữ khác như: list, tuple, dict, set (sẽ được trình bày ở phần sau)

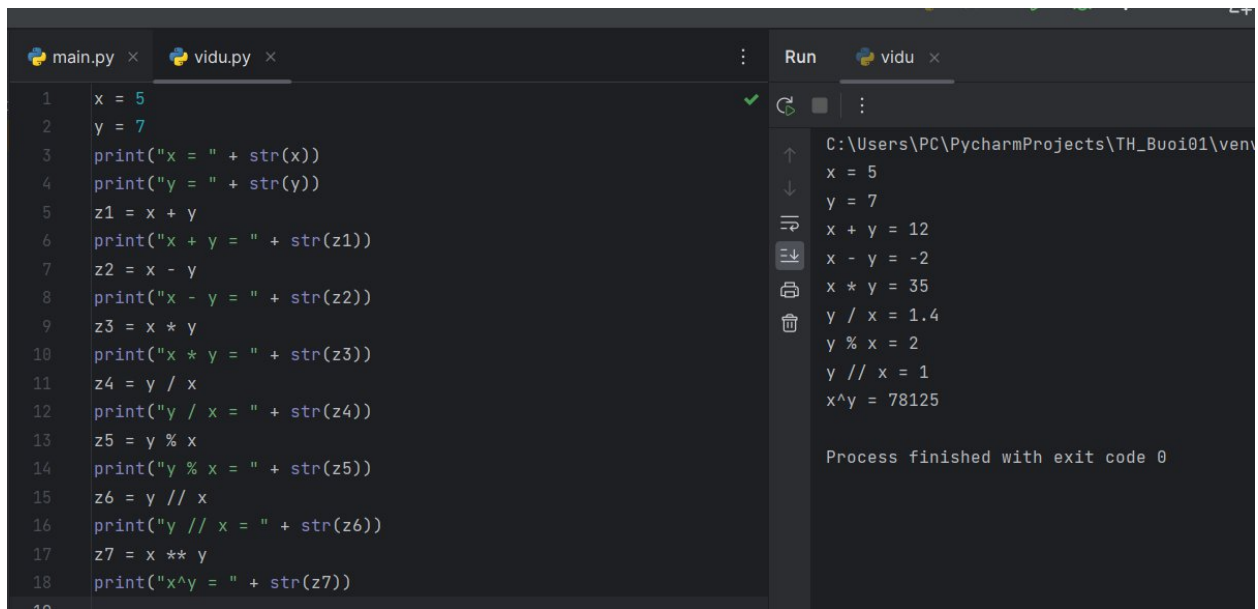
- Kiểu dữ liệu NoneType

1.2.2. Các phép toán thông dụng

Các phép toán trong python cũng tương tự như các phép toán trong các ngôn ngữ lập trình khác, chúng có thể chia thành những nhóm như sau:

- Toán tử tính toán số học: +, -, *, /, %, **, //

Ví dụ cho các toán tử trên như sau:



The screenshot shows a PyCharm IDE with two tabs: 'main.py' and 'vidu.py'. The 'vidu.py' tab is active, displaying a Python script that calculates various arithmetic operations on variables x=5 and y=7. The script includes assignments, addition, subtraction, multiplication, division, modulus, and exponentiation, each followed by a print statement. To the right, the 'Run' window shows the output of the script, which matches the print statements in the code. The output includes the values of x, y, and the results of the operations: x+y=12, x-y=-2, x*y=35, y/x=1.4, y%x=2, y//x=1, and x^y=78125. The process finished with exit code 0.

```
1 x = 5
2 y = 7
3 print("x = " + str(x))
4 print("y = " + str(y))
5 z1 = x + y
6 print("x + y = " + str(z1))
7 z2 = x - y
8 print("x - y = " + str(z2))
9 z3 = x * y
10 print("x * y = " + str(z3))
11 z4 = y / x
12 print("y / x = " + str(z4))
13 z5 = y % x
14 print("y % x = " + str(z5))
15 z6 = y // x
16 print("y // x = " + str(z6))
17 z7 = x ** y
18 print("x^y = " + str(z7))
```

Run vidu x

C:\Users\PC\PycharmProjects\TH_Buoi01\venv

x = 5
y = 7
x + y = 12
x - y = -2
x * y = 35
y / x = 1.4
y % x = 2
y // x = 1
x^y = 78125

Process finished with exit code 0

- Các toán tử dùng cho phép gán: =, +=, -=, *=, /=, %=, ^=,...

- Các toán tử so sánh: ==, !=, >, <, >=, <=

- Các toán tử logic: and, not, or

Các ví dụ về các toán tử này các bạn có thể tham khảo và thực tập thêm tại đường link sau: https://www.w3schools.com/python/python_operators.asp

1.2.3. Cấu trúc điều khiển

- Cấu trúc lựa chọn

Nguyên tắc hoạt động của câu lệnh if...else... hoàn toàn giống với các ngôn ngữ lập trình khác, chỉ khác ở cú pháp.

Ví dụ sau đây sẽ minh họa cách sử dụng câu lệnh if...else...

```
1 x = 13
2 print("Giá trị của x = " + str(x))
3
4
5 if x % 2 == 0:
6     print("x chia hết cho 2")
7 else:
8     print("x không chia hết cho 2")
9
10
```

Run output:

```
C:\Users\PC\PycharmProjects\TH_Buoi01\venv\Scr
Giá trị của x = 13
x không chia hết cho 2
Process finished with exit code 0
```

- Cấu trúc lặp

Cấu trúc lặp trong python gồm có vòng lặp while và vòng lặp for.

Về cơ bản, nguyên tắc hoạt động của các vòng lặp này giống với vòng lặp của các ngôn ngữ lập trình khác, chỉ khác ở cú pháp sử dụng.

Ví dụ sau đây minh họa cho việc sử dụng vòng lặp while trong python:

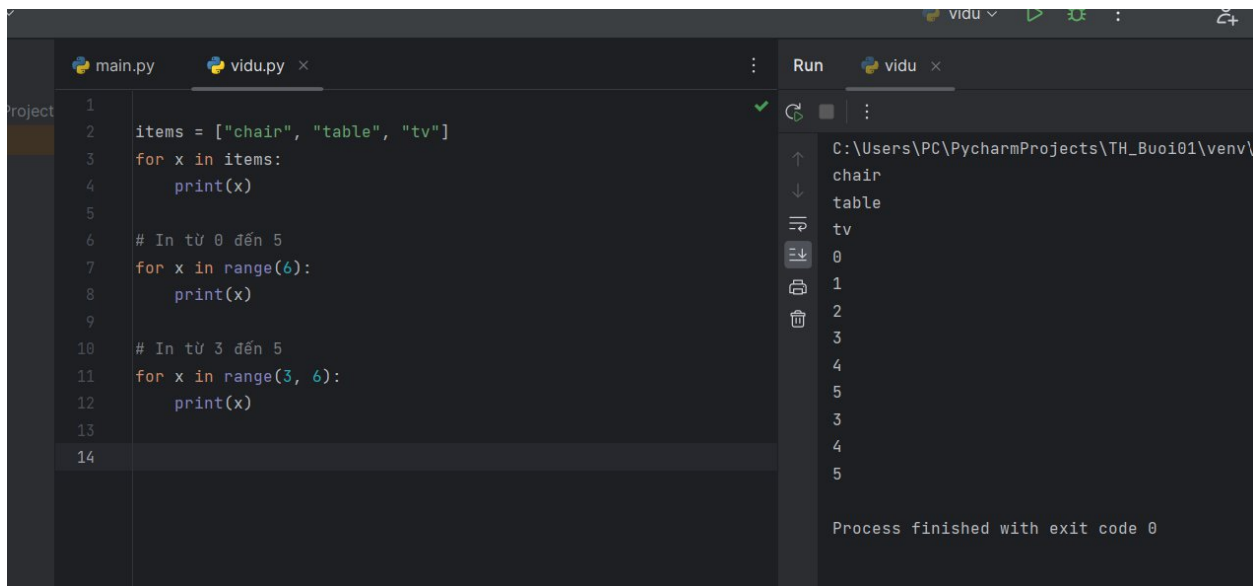
```
1 i = 0
2 sum = 0
3 while i <= 10:
4     sum += i
5     i += 1
6 print("Tổng từ 1 đến 10 bằng: " + str(sum))
7
```

Run output:

```
C:\Users\PC\PycharmProjects\TH_Buoi01\venv\Scr
Tổng từ 1 đến 10 bằng: 55
Process finished with exit code 0
```

Thông thường, vòng lặp for sẽ được dùng nhiều trong việc duyệt qua các phần tử của các cấu trúc list, tuple, dictionary, string, set.

Ví dụ về vòng lặp for như sau:



The screenshot shows the PyCharm IDE with a file named `main.py` open. The code in the editor is as follows:

```
1 items = ["chair", "table", "tv"]
2 for x in items:
3     print(x)
4
5 # In từ 0 đến 5
6 for x in range(6):
7     print(x)
8
9 # In từ 3 đến 5
10 for x in range(3, 6):
11     print(x)
12
13
14
```

The Run window on the right shows the output of the script:

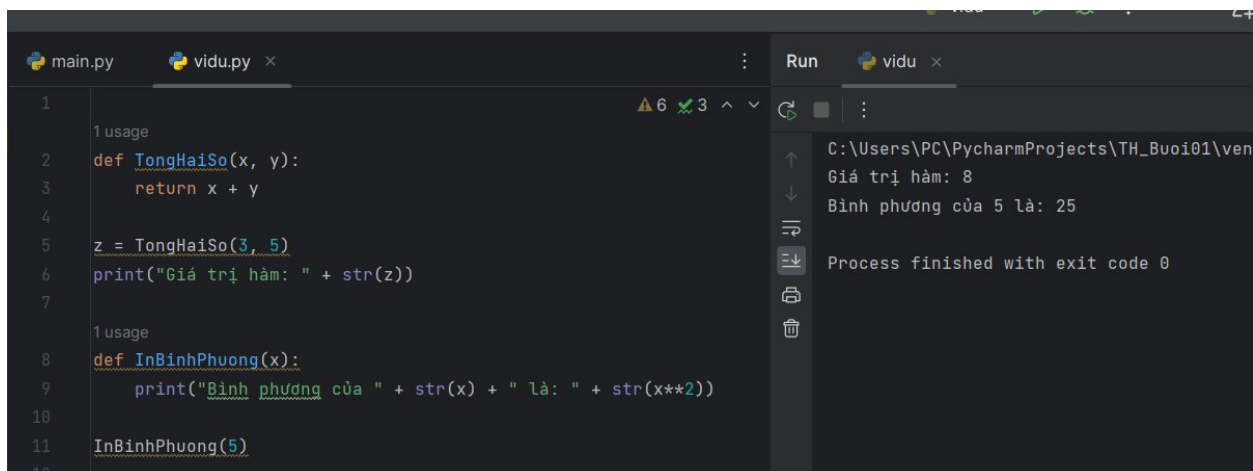
```
C:\Users\PC\PycharmProjects\TH_Buoi01\venv\
chair
table
tv
0
1
2
3
4
5
3
4
5

Process finished with exit code 0
```

1.2.4. Hàm trong python

Hàm trong python là một khối chứa các câu lệnh python mà chúng chỉ thực thi khi ta gọi hàm đó. Mỗi hàm thì có thể có hoặc không chứa tham số, có thể trả về kết quả hoặc không trả về kết quả.

Ví dụ sau đây minh họa cách định nghĩa và sử dụng hàm trong python



The screenshot shows the PyCharm IDE with a file named `main.py` open. The code in the editor is as follows:

```
1 1 usage
2 def TongHaiSo(x, y):
3     return x + y
4
5 z = TongHaiSo(3, 5)
6 print("Giá trị hàm: " + str(z))
7
8 1 usage
9 def InBinhPhuong(x):
10     print("Bình phương của " + str(x) + " là: " + str(x**2))
11
12 InBinhPhuong(5)
```

The Run window on the right shows the output of the script:

```
C:\Users\PC\PycharmProjects\TH_Buoi01\venv\
Giá trị hàm: 8
Bình phương của 5 là: 25

Process finished with exit code 0
```

1.3. Một số cấu trúc dữ liệu lưu trữ thường dùng trong python

1.3.1. Kiểu dữ liệu danh sách (List)

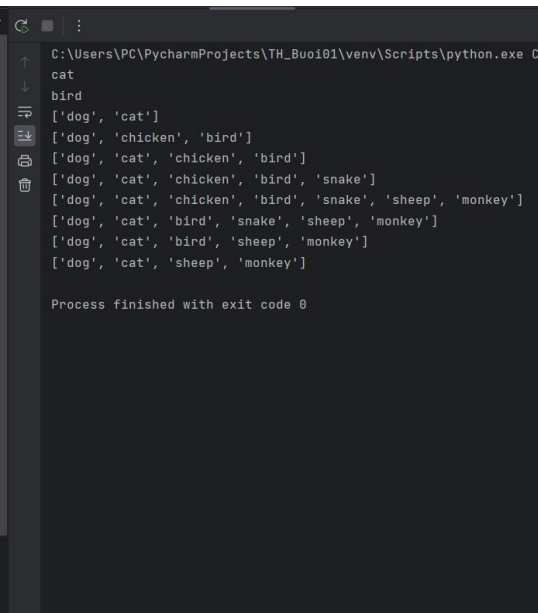
Danh sách có thể lưu trữ nhiều phần tử có thứ tự, có thể thay đổi được và các phần tử này có thể có các giá trị giống nhau. Phần tử đầu tiên được lưu ở chỉ số 0, phần tử thứ n sẽ được lưu ở chỉ số $n - 1$.

Chúng ta có thể định nghĩa một danh sách chứa các phần tử cùng kiểu dữ liệu hoặc khác kiểu dữ liệu với nhau.

Một số thao tác có thể thực hiện trên danh sách như: truy xuất đến các phần tử danh sách thông qua chỉ số, thêm phần tử vào danh sách, sửa phần tử trong danh sách, xóa phần tử khỏi danh sách, duyệt qua các phần tử trong danh sách, sắp xếp danh sách, nối các danh sách lại với nhau và một số cú pháp thu gọn liên quan đến danh sách.

Các ví dụ dưới đây lần lượt sẽ minh họa cho chúng ta lần lượt các thao tác bên trên:

```
1 myList = ["dog", "cat", "bird"]
2 print(myList[1])
3 # --> return "cat"
4 print(myList[-1])
5 # --> return "bird"
6 print(myList[0:2])
7 # --> return "dog", "cat"
8 myList[1] = "chicken"
9 print(myList)
10 # --> return ['dog', 'chicken', 'bird']
11 myList.insert(1, "cat")
12 print(myList)
13 # --> return ['dog', 'cat', 'chicken', 'bird']
14 myList.append("snake")
15 print(myList)
16 # --> return ['dog', 'cat', 'chicken', 'bird', 'snake']
17 newList = ['sheep', 'monkey']
18 myList.extend(newList)
19 print(myList)
20 # --> return ['dog', 'cat', 'chicken', 'bird', 'snake', 'sheep', 'monkey']
21 myList.remove("chicken")
22 print(myList)
23 # --> return ['dog', 'cat', 'bird', 'snake', 'sheep', 'monkey']
24 myList.pop(3)
25 print(myList)
26 # --> return ['dog', 'cat', 'bird', 'sheep', 'monkey']
27 del myList[2]
28 print(myList)
29 # --> return ['dog', 'cat', 'sheep', 'monkey']
```



The output window shows the following sequence of lists:

- cat
- bird
- ['dog', 'cat']
- ['dog', 'chicken', 'bird']
- ['dog', 'cat', 'chicken', 'bird']
- ['dog', 'cat', 'chicken', 'bird', 'snake']
- ['dog', 'cat', 'chicken', 'bird', 'snake', 'sheep', 'monkey']
- ['dog', 'cat', 'bird', 'snake', 'sheep', 'monkey']
- ['dog', 'cat', 'bird', 'sheep', 'monkey']
- ['dog', 'cat', 'sheep', 'monkey']

Process finished with exit code 0

```

print(myList)
# --> return ['dog', 'cat', 'sheep', 'monkey']

# some ways for loop list
# 1st
for x in myList:
    print(x)
# 2nd
for i in range(len(myList)):
    print(myList[i])
# 3rd
[print(x) for x in myList]

```

Cú pháp để tạo nhanh một danh sách từ giá trị của một danh sách có sẵn. Chúng ta có thể dùng cú pháp theo dạng chung như sau:

newlist = [expression for item in iterable if condition == True]

```

list_01 = ['a', 'b', 'c', 'd', 'e', 'f']
list_02 = [x for x in list_01]
print(list_02)
# --> return ['a', 'b', 'c', 'd', 'e', 'f']
list_03 = [x for x in list_01 if x != 'd']
print(list_03)
# --> return ['a', 'b', 'c', 'e', 'f']
list_04 = [x for x in range(10)]
print(list_04)
# --> return [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
list_05 = [1] * 10
print(list_05)
# --> return [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

```

Ngoài ra còn một số hàm xử lý liên quan đến danh sách thì các bạn có thể xem thêm tại link sau: https://www.w3schools.com/python/python_lists_methods.asp

1.3.2. Kiểu dữ liệu *Tuples*

Đây cũng là một kiểu dữ liệu tương tự như danh sách, tuy nhiên chúng ta không thể thay đổi giá trị trên tuple được như chúng ta đã từng làm với danh sách.

Một số thao tác trên tuple có thể là truy xuất phần tử, cập nhật lại tuple, duyệt qua các phần tử của tuple và nối các tuple lại với nhau.

Các ví dụ dưới đây lần lượt sẽ minh họa cho chúng ta lần lượt các thao tác bên trên:

```

myTuple = ('cat', 'dog', 'bird')
print(myTuple)
# --> return ('cat', 'dog', 'bird')
print(myTuple[1])
# --> return dog
print(myTuple[-1])
# --> return bird

# Change from Tuple to List
list_of_Tuple = list(myTuple)
list_of_Tuple[1] = 'chicken'
myTuple = tuple(list_of_Tuple)
print(myTuple)
# --> return ('cat', 'chicken', 'bird')

# Loop Tuple
for x in myTuple:
    print(x)

for i in range(len(myTuple)):
    print(myTuple[i])

[print(x) for x in myTuple]

newTuple = ('dog', 'sheep')
joinTuple = myTuple + newTuple
print(joinTuple)
# --> return ('cat', 'chicken', 'bird', 'dog', 'sheep')

```

1.3.3. Kiểu dữ liệu tập hợp (Set)

Đây là kiểu dữ liệu lưu trữ các giá trị không cần theo thứ tự, không thể thay đổi giá trị và không thể đánh chỉ mục cho dữ liệu dùng trong tìm kiếm.

Một số thao tác trên tập hợp có thể được là: truy xuất phần tử của tập hợp (không qua index), thêm phần tử, xóa phần tử, duyệt qua phần tử, nối các tập hợp với nhau.

Các ví dụ dưới đây lần lượt sẽ minh họa cho chúng ta lần lượt các thao tác bên trên:

```
mySet = {'dog', 'cat', 'bird'}
print(mySet)
# --> return {'dog', 'cat', 'bird'}

# Loop and access elements in set
for x in mySet:
    print(x)

mySet.add('sheep')
print(mySet)
# --> return {'bird', 'sheep', 'cat', 'dog'}
mySet.remove('bird')
print(mySet)
# --> return {'sheep', 'dog', 'cat'}

mySet_01 = {1, 2, 3, 'dog'}
newSet = mySet.union(mySet_01)
print(newSet)
# --> return {1, 2, 3, 'sheep', 'cat', 'dog'}
```

Ngoài ra còn một số hàm xử lý liên quan đến tập hợp thì các bạn có thể xem thêm tại link sau: https://www.w3schools.com/python/python_sets_methods.asp

1.3.4. Kiểu dữ liệu từ điển (Dictionary)

Từ điển trong python thì được lưu trữ dưới dạng một cặp key – value, đây là kiểu dữ liệu lưu trữ có thứ tự, có thể thay đổi giá trị và không có lưu giá trị trùng lặp.

Một số thao tác trên từ điển có thể là truy xuất các phần tử trong từ điển thông qua khóa, thay đổi giá trị từ điển, thêm, xóa, duyệt qua phần tử của từ điển, từ điển lồng nhau.

Các ví dụ dưới đây lần lượt sẽ minh họa cho chúng ta lần lượt các thao tác bên trên:


```

myDict_01 = {
    "key_01": "dog",
    "key_02": "cat",
    "key_03": "bird"
}
print(myDict_01)
# --> return {'key_01': 'dog', 'key_02': 'cat', 'key_03': 'bird'}
myDict_02 = {
    "ID": 1290,
    "Name": "Steven",
    "Hobbies": ['Travel', 'Shopping']
}
print(myDict_02)
# --> return {'ID': 1290, 'Name': 'Steven', 'Hobbies': ['Travel', 'Shopping']}

val_dog = myDict_01["key_01"]
print(val_dog)
# --> return dog
val_name = myDict_02.get("Name")
print(val_name)
# --> return Steven
print(myDict_02.keys())
# --> return dict_keys(['ID', 'Name', 'Hobbies'])

myDict_02["ID"] = 1390
print(myDict_02)
# --> return {'ID': 1390, 'Name': 'Steven', 'Hobbies': ['Travel', 'Shopping']}
myDict_01.update({"key_03": "chicken"})
print(myDict_01)
# --> return {'key_01': 'dog', 'key_02': 'cat', 'key_03': 'chicken'}

```

```

myDict_01["key_04"] = "sheep"
print(myDict_01)
# --> return {'key_01': 'dog', 'key_02': 'cat', 'key_03': 'chicken', 'key_04': 'sheep'}
myDict_02.update({"Address": "CT"})
print(myDict_02)
# --> return {'ID': 1390, 'Name': 'Steven', 'Hobbies': ['Travel', 'Shopping'], 'Address': 'CT'}

myDict_01.pop("key_04")
print(myDict_01)
# --> return {'key_01': 'dog', 'key_02': 'cat', 'key_03': 'chicken'}
del myDict_02["Address"]
print(myDict_02)
# --> return {'ID': 1390, 'Name': 'Steven', 'Hobbies': ['Travel', 'Shopping']}

for x in myDict_01:
    print(x)
# --> return keys
for x in myDict_01:
    print(myDict_01[x])
# --> return values
for x in myDict_01.values():
    print(x)
# --> return values
for x, y in myDict_01.items():
    print(x, y)
# --> return key, value

```

Ngoài ra còn một số hàm xử lý liên quan đến từ điển thì các bạn có thể xem thêm tại link sau: https://www.w3schools.com/python/python_dictionaries_methods.asp

1.3.5. Kiểu dữ liệu Dataframe trong thư viện Pandas

Đây là kiểu dữ liệu 2 chiều, hoàn toàn tương tự như ma trận hoặc có thể hình dung như 1 cái bảng có dòng và cột.

Sở dĩ trình bày kiểu dữ liệu này bởi vì nó có cấu trúc lưu trữ rất giống với các tập dữ liệu mà chúng ta thường huấn luyện.

Các ví dụ dưới đây lần lượt sẽ minh họa cho chúng ta lần lượt các thao tác liên quan đến kiểu dữ liệu Dataframe:

```

import pandas as pd
data = {
    "Att1": [410, 280, 400],

```

```

    "Att2": [20, 30, 35]
}
df = pd.DataFrame(data)
print(df)
""" --> return
    Att1 Att2
0  410   20
1  280   30
2  400   35
"""

# Return one or more specified row(s)
print(df.loc[0])
""" --> return
Att1   410
Att2    20
Name: 0, dtype: int64
"""

print(df.loc[1:2])
""" --> return
    Att1 Att2
1  280   30
2  400   35
"""

print(df.loc[:, 'Att1'])
""" --> return
0    410
1    280
2    400
Name: Att1, dtype: int64
"""

print(df.loc[1:2, 'Att2'])
""" --> return
1    30
2    35
Name: Att2, dtype: int64
"""

```

1.3.6. Kiểu dữ liệu Array trong thư viện Numpy

Để có thể thao tác trên kiểu dữ liệu mảng, thường người ta sẽ sử dụng thư viện Numpy. Cách cài đặt thư viện này khá đơn giản thông qua dòng lệnh sau:

```
pip install numpy
```

Sau khi cài thì chúng ta cứ import vào và sử dụng để thao tác với mảng.

- Sử dụng numpy để tạo mảng dữ liệu:

Mảng thì có n chiều (n có thể nhận giá trị 0, 1, 2, 3,...). Thông thường, chúng ta sẽ gặp nhiều nhất là mảng 1 chiều và mảng 2 chiều (ma trận). Để tạo mảng, chúng ta có thể thực hiện theo minh họa sau đây:

```
import numpy as np

a = np.array(3)
print(a)
# --> return 3 (0-dim array)
b = np.array([2, 4])
print(b)
# --> return [2, 4] (1-dim array)
c = np.array([[1, 3], [2, 5]])
print(c)
'''--> return (2-dim array - matrix)
[[1 3]
 [2 5]]
'''
d = np.array([[[1, 2], [3, 4]], [[2, 5], [2, 1]]])
print(d)
'''--> return (3-dim array)
[[[1 2]
   [3 4]]
 [[2 5]
   [2 1]]]
'''
```

- Truy xuất phần tử trên mảng

Việc truy xuất phần tử trên mảng sẽ thực hiện thông qua chỉ số index. Cụ thể qua ví dụ minh họa sau:

```
import numpy as np

a = np.array([1, 4, 3, 2, 6])
print(a[2])
# --> return 3

b = np.array([[1, 3, 5, 6], [2, 4, 8, 7]])
print(b)
print(b[1, 2])
# --> return 8
print(b[1, 1:3])
# --> return [4 8]
print(b[0, :])
# --> return [1 3 5 6]
```

- Nối mảng với nhau:

```

import numpy as np

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

c = np.concatenate((a, b), axis=1)
print(c)
''' --> return
[[1 3 6 8]
 [2 4 5 7]]
'''

c = np.concatenate((a, b), axis=0)
print(c)
''' --> return
[[1 3]
 [2 4]
 [6 8]
 [5 7]]
'''

c = np.concatenate((a, b), axis=-1)
print(c)
''' --> return
[[1 3 6 8]
 [2 4 5 7]]
'''

```

1.3.7. Vẽ đồ thị trong python với Matplotlib

Để vẽ đồ thị trong python, trước tiên chúng ta cần cài đặt thư viện Matplotlib. Cách cài đặt đơn giản bằng dòng lệnh sau:

```
pip install matplotlib
```

Sau khi cài đặt xong, chúng ta tiến hành import vào chương trình để thực hiện vẽ đồ thị.

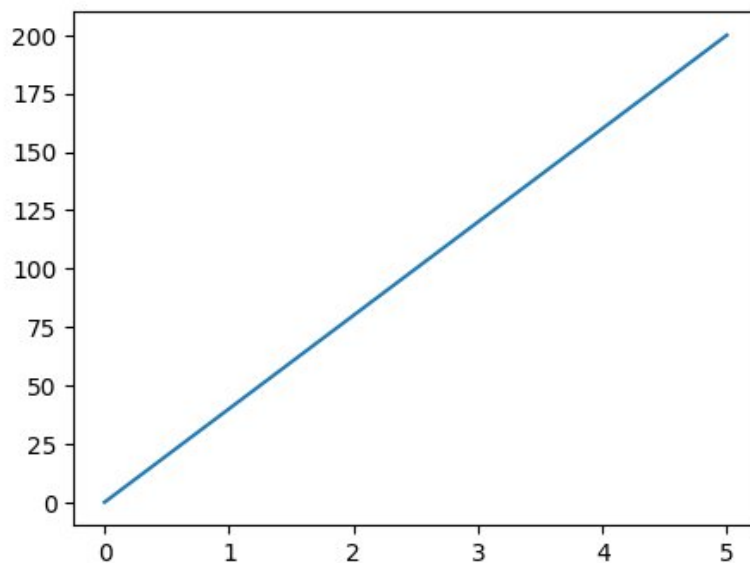
Để vẽ đồ thị với Matplotlib, chúng ta sử dụng module pyplot thông qua ví dụ đơn giản như sau:

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

x_points = np.array([0, 5])
y_points = np.array([0, 200])

plt.plot(x_points, y_points)
plt.show()
```

Kết quả sẽ như hình sau:



Ví dụ:

Chúng ta có bảng số liệu sau mô tả độ lỗi khi huấn luyện mô hình theo từng epoch. Hãy minh họa số liệu này dưới dạng đồ thị.

Epoch	1	5	10	15	20	25	30	35	40
Error Rate	0.98	0.67	0.45	0.33	0.21	0.11	0.08	0.01	0.002

Có thể thực hiện bằng đoạn code như sau:

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

x_points = np.array([1, 5, 10, 15, 20, 25, 30, 35, 40])
y_points = np.array([0.98, 0.67, 0.45, 0.33, 0.21, 0.11, 0.08, 0.01, 0.002])

plt.plot(x_points, y_points)
plt.title("Error Rate With Epoch")
plt.xlabel("Epoch")
plt.ylabel("Error Rate")
plt.show()
```

Kết quả sẽ là:

