

Giới thiệu về Python

Python là một **ngôn ngữ lập trình thông dịch** (interpreter) và hướng đối tượng (object-oriented).

Python hỗ trợ module và gói (packages).

* Ghi chú:

- *Interpreter (trình thông dịch): Biên dịch và thực thi đồng thời từng dòng code*
- *Compiler (trình biên dịch): Biên dịch toàn bộ chương trình thành mã trung gian*

1.2 Lịch sử (1)

Python đã được hình thành vào cuối những năm 1980. Ban hành chính thức vào tháng 12 năm 1989 bởi **Guido van Rossum** tại **Centrum Wiskunde & Informatica (CWI)** ở Hà Lan.

Van Rossum là tác giả chính của **Python**, và ông là một trong những người tiên phong đưa ra hướng phát triển của **Python**.



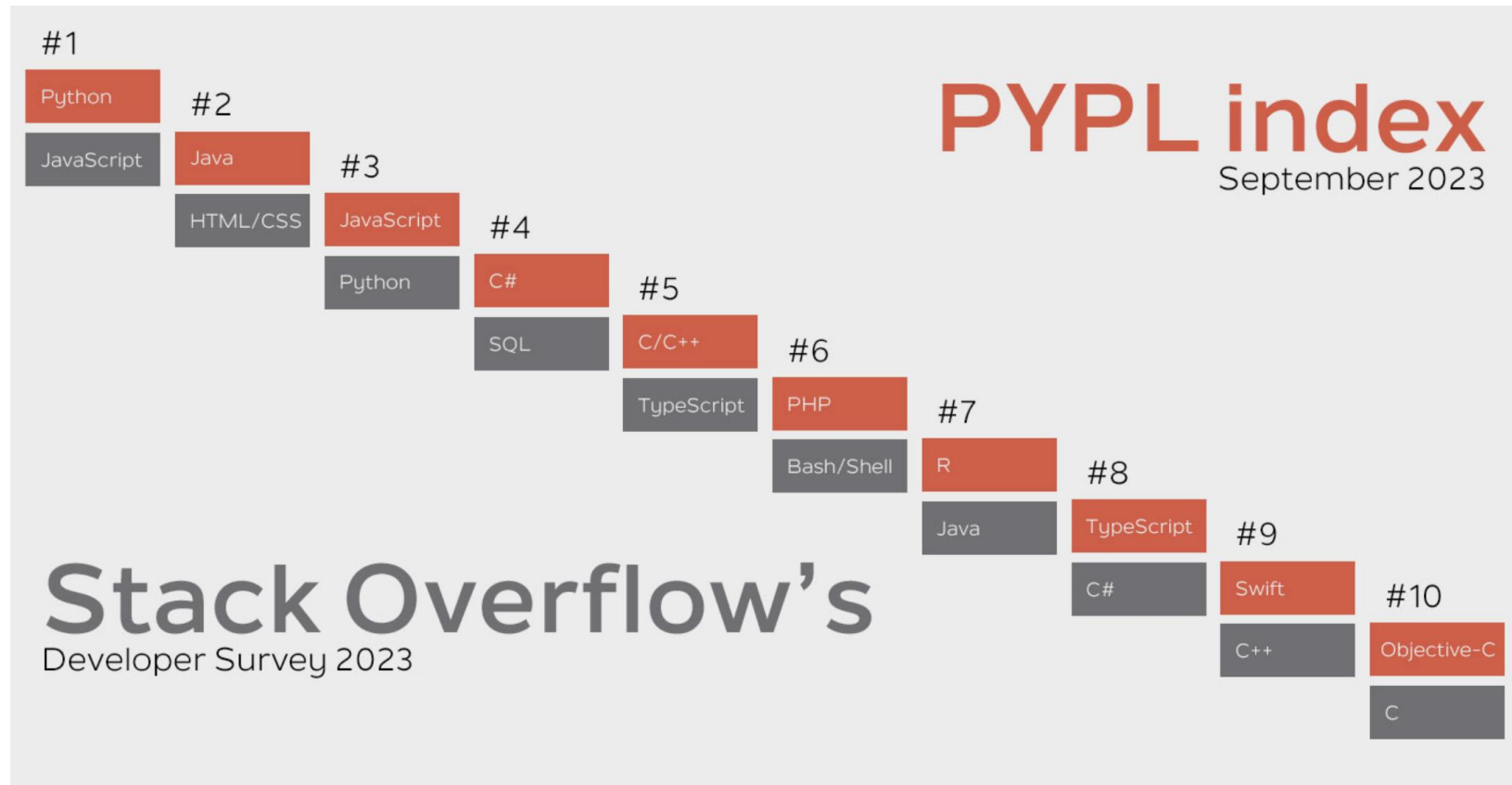
Python 2.0 được phát hành vào ngày 16 tháng 10 năm 2000 và có nhiều tính năng mới, trong đó có tính năng hỗ trợ Unicode.

Python 3.0 được phát hành năm 2008, sau một thời gian dài thử nghiệm.

Cho tới năm 2024, Python đang có phiên bản **3.12.x**, chuẩn bị release bản **3.13**

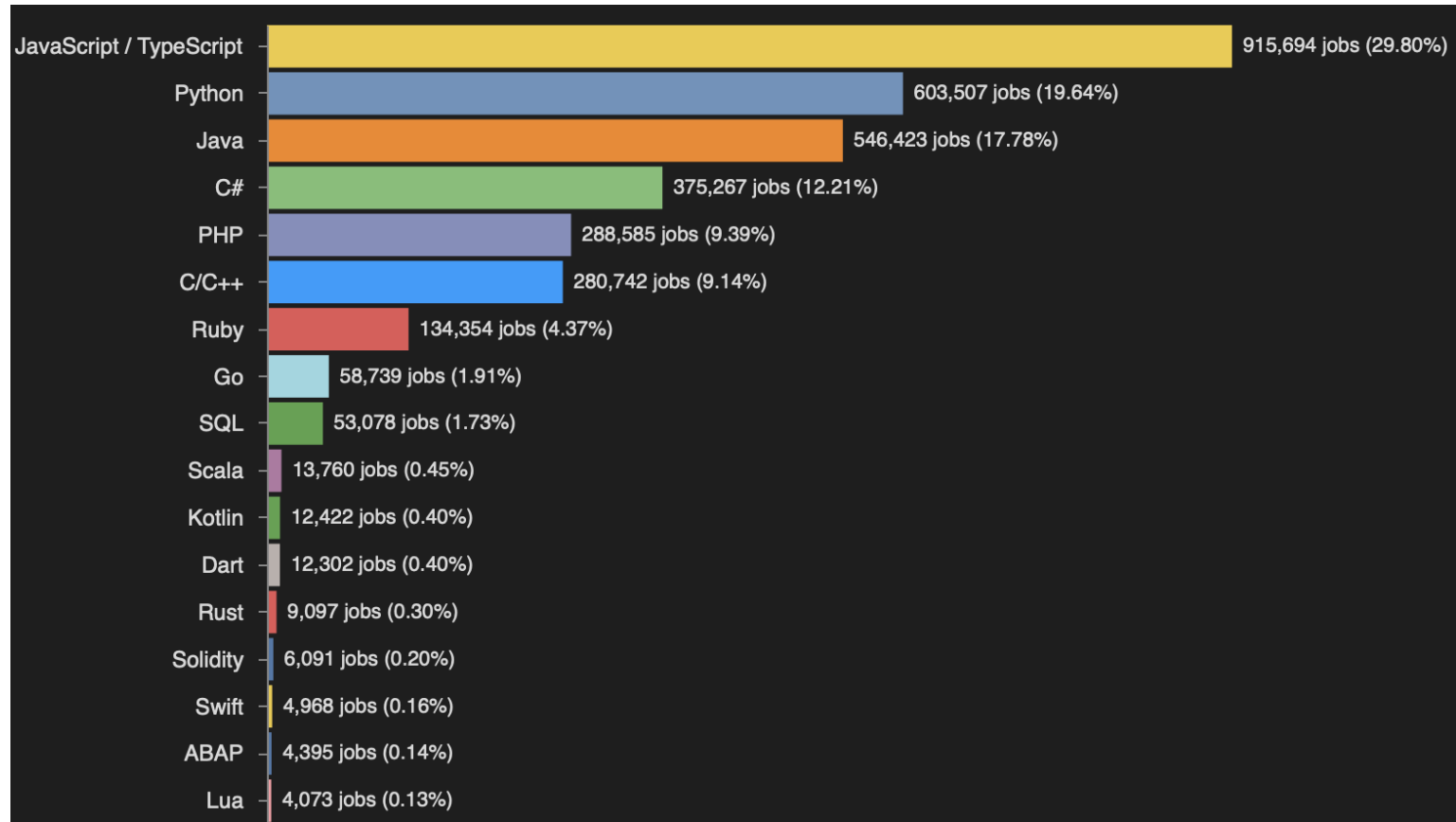
1.3 Hiện tại

Xếp hạng các ngôn ngữ lập trình phổ biến nhất

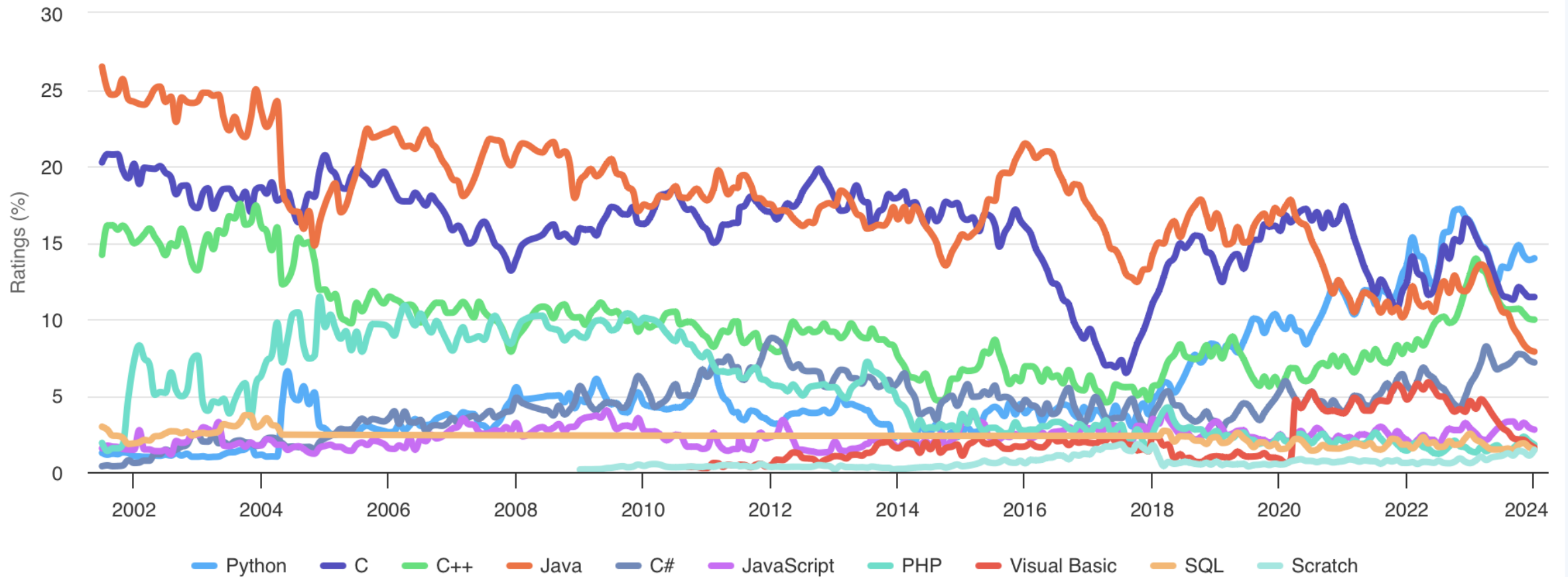


1.3 Hiện tại (tt)

Top các ngôn ngữ lập trình phổ biến nhất năm 2023 dựa trên số lượng các công việc được tuyển dụng.



1.4 Xu hướng



1.5 Đặc điểm

Python là ngôn ngữ có khả năng chạy trên nhiều nền tảng: Windows, Linux/Unix, Mac và những hệ điều hành khác. Thậm chí có cả những phiên bản được phát triển chạy trên .NET, máy ảo Java.

Python rất đơn giản và dễ học.

Python có cộng đồng lập trình lớn, hệ thống thư viện chuẩn, các thư viện mã nguồn mở được chia sẻ phổ biến trên internet.

Python là ngôn ngữ mã nguồn mở được sử dụng và phân phối tự do, ngay cả trong việc thương mại.

1.5 Đặc điểm (tt)

Vừa hướng thủ tục (procedural-oriented), vừa hướng đối tượng (object-oriented).

Hỗ trợ module và hỗ trợ gói (package).

Xử lý lỗi bằng ngoại lệ (Exception).

Kiểu dữ liệu động.

Có khả năng tương tác với các ngôn ngữ lập trình khác: C/C++, Java, .Net ...

Nội dung bài 1

- 1. Cài đặt và thiết lập môi trường lập trình**
- 2. Xây dựng ứng dụng đầu tiên (Hello world!)**
- 3. Một số loại lỗi thường gặp trong lập trình**
- 4. Biến và khai báo biến trong Python**
- 5. Các kiểu dữ liệu cơ bản (Số, Ký tự, Danh sách, Boolean)**

1. Cài đặt và thiết lập môi trường lập trình

Sinh viên cài đặt Python dùng PyCharm Community Edition 2024.1.1 hoặc Visual Code

1.1 Môi trường & công cụ lập trình

Cài đặt môi trường:

OS: Windows, Unix (MacOs hoặc Ubuntu...)

Python: 3.x

IDE: PyCharm hoặc Visual Studio Code (VSCode)

1.2 Cài đặt Python

Bước 1: Download python 3.12.x tại
<https://www.python.org/downloads/>

Bước 2: Cài đặt theo hướng dẫn

Windows: <https://docs.python.org/3/using/windows.html>

Unix: <https://docs.python.org/3/using/unix.html>

1.2 Cài đặt Python (tt)

Lưu ý:

Chọn “Add python.exe to PATH”



1.3 Cài đặt Pycharm

IDE

Includes many Python tools

Text editor (code completion), Libraries, Compiler Test platform (unittest, pytest,...)

Database (SQL, NoSQL)

Link download:

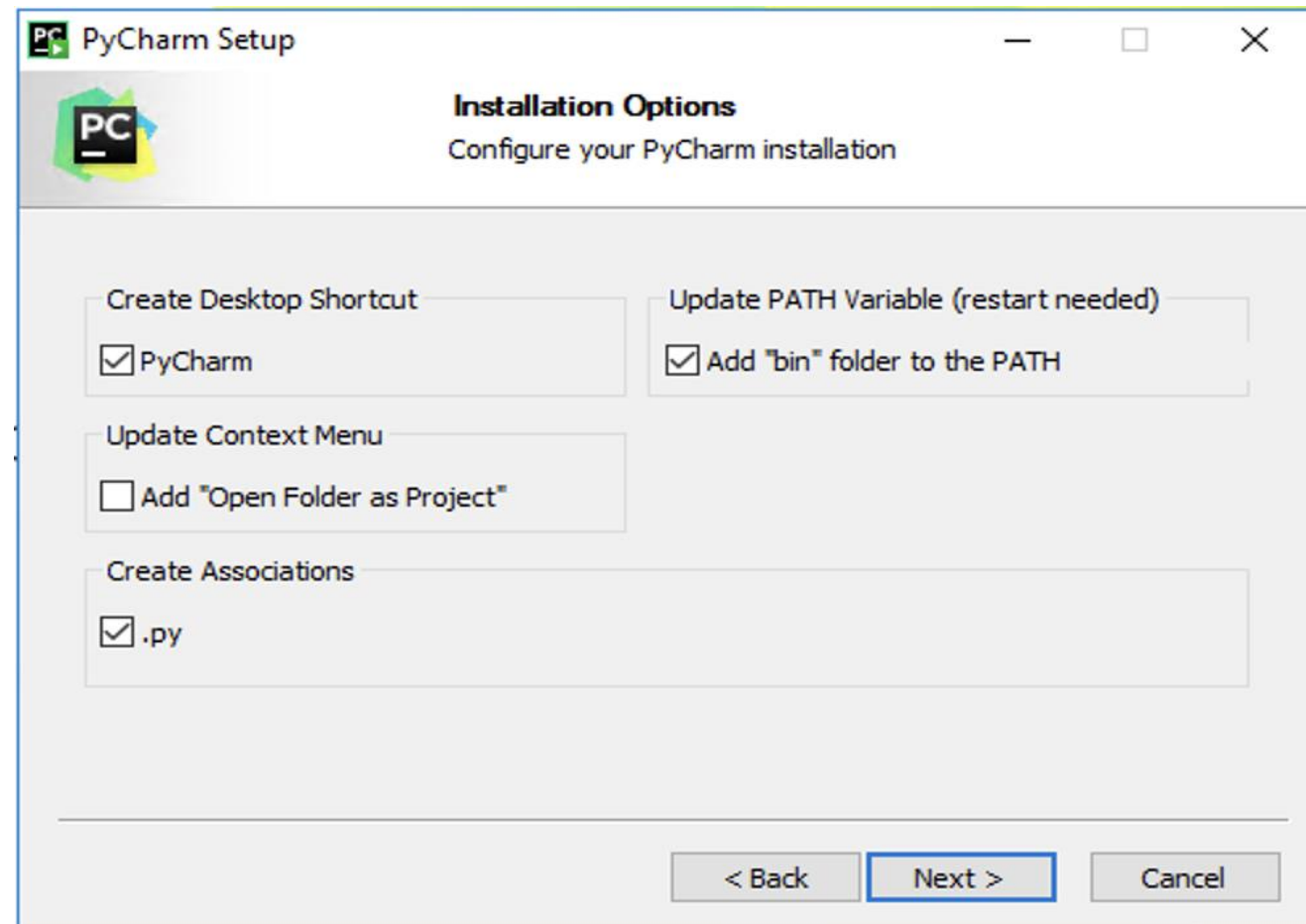
<https://www.jetbrains.com/pycharm/>



1.3 Cài đặt Pycharm (tt)

Lưu ý:

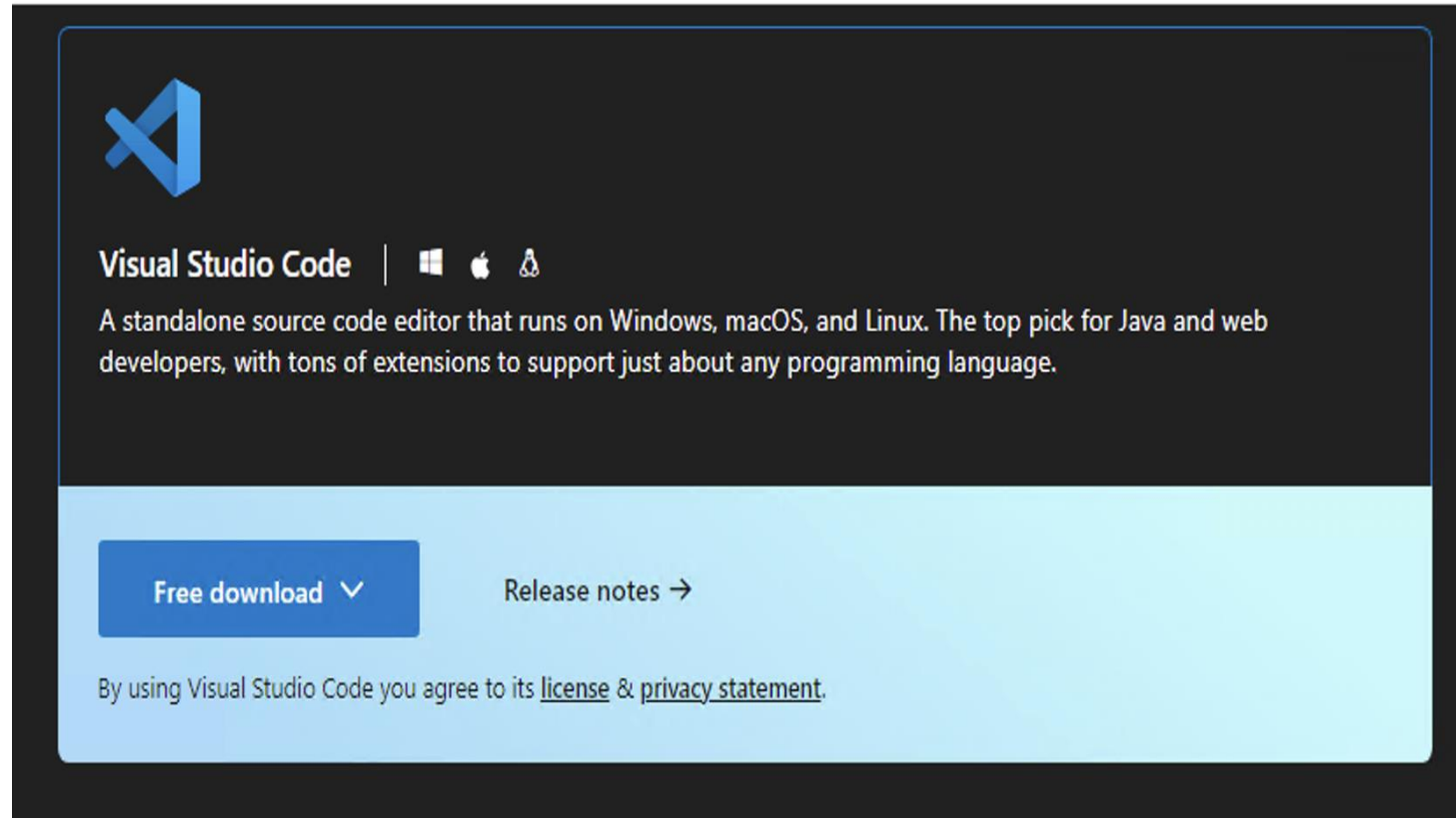
Chọn “Add bin folder to the PATH”



1.4 Cài đặt VSCode

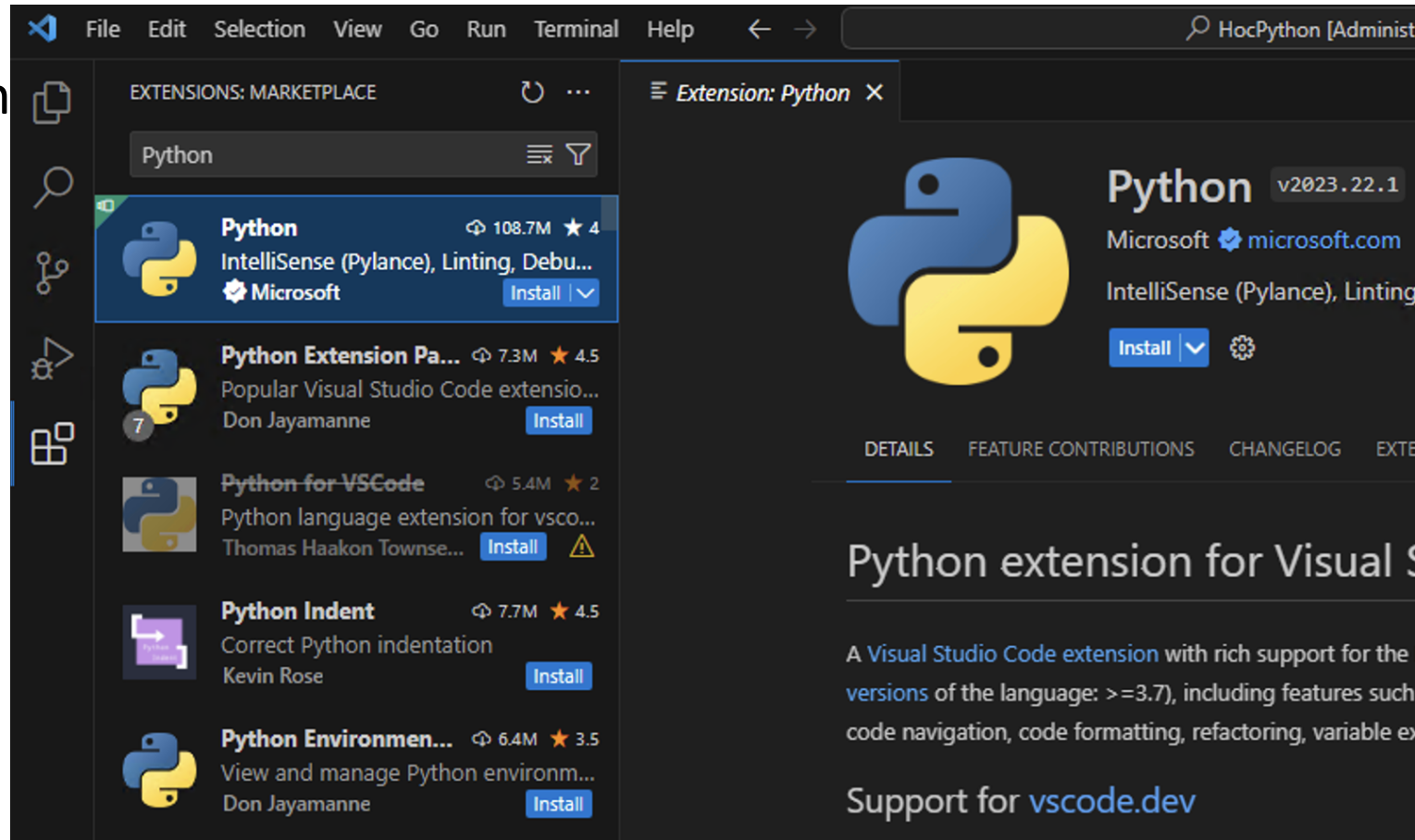
Link download:

<https://code.visualstudio.com>

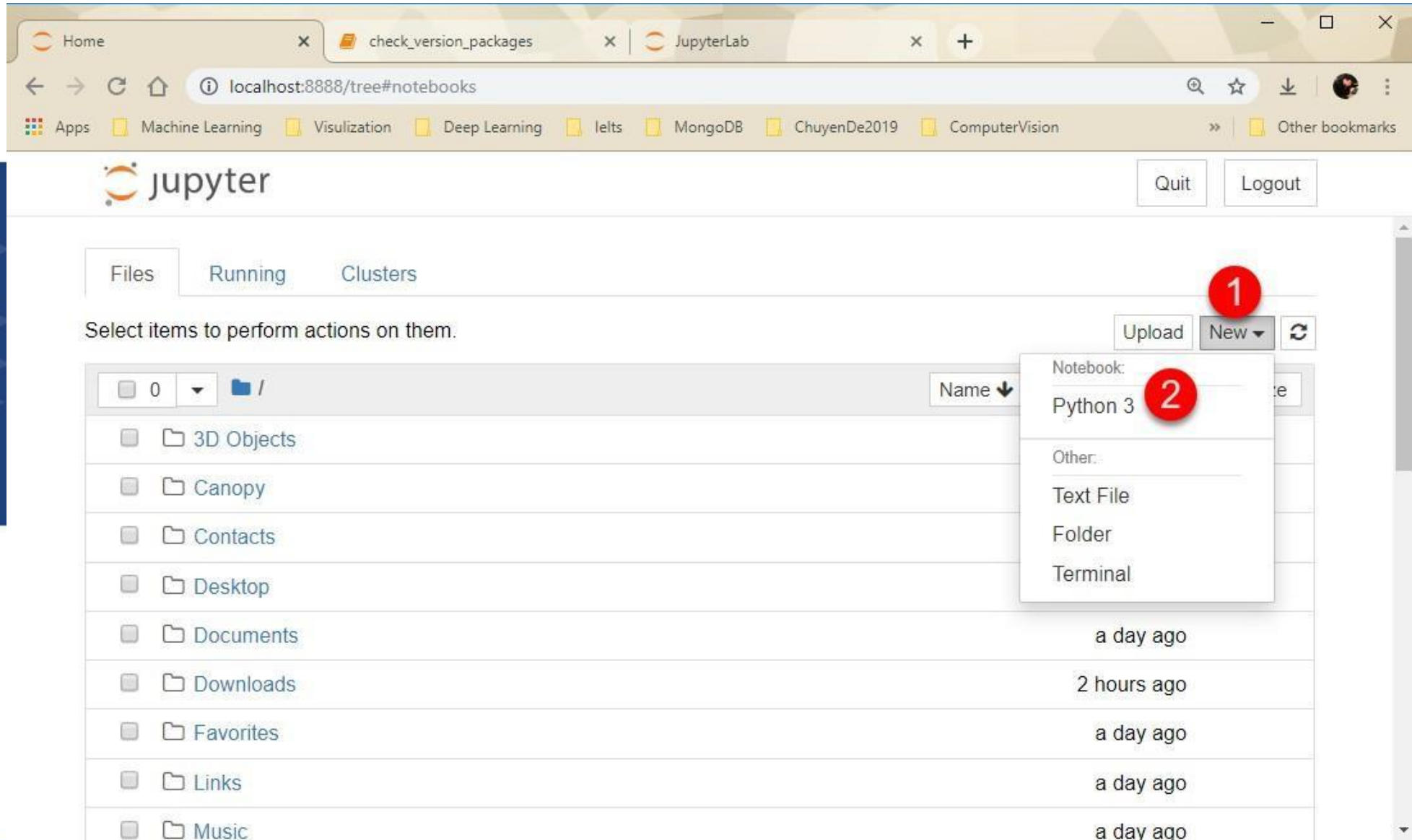


2.4 Cài đặt VSCode (tt)

Cài đặt thêm Extension
“python” trên VSCode



Ví dụ Tạo file Python mới với Jupyter Notebook



The screenshot displays the JupyterLab web interface in a browser. The address bar shows `localhost:8888/tree#notebooks`. The interface includes a top navigation bar with 'Home', 'check_version_packages', and 'JupyterLab' tabs. Below the navigation bar, there are tabs for 'Files', 'Running', and 'Clusters'. The 'Files' tab is active, showing a file browser with a list of folders: 3D Objects, Canopy, Contacts, Desktop, Documents, Downloads, Favorites, Links, and Music. A red circle with the number '1' highlights the 'New' button in the top right corner. A red circle with the number '2' highlights the 'Python 3' option in the dropdown menu that appears when the 'New' button is clicked. The dropdown menu also includes options for 'Text File', 'Folder', and 'Terminal'. The 'Python 3' option is selected, and a 'Notebook:' label is visible above it.

Home x check_version_packages x JupyterLab x +

localhost:8888/tree#notebooks

Apps Machine Learning Visulization Deep Learning Ielts MongoDB ChuyenDe2019 ComputerVision Other bookmarks

jupyter Quit Logout

Files Running Clusters

Select items to perform actions on them.

0 / Name

3D Objects

Canopy

Contacts

Desktop

Documents

Downloads

Favorites

Links

Music

Upload New

Notebook:

Python 3

Other:

Text File

Folder

Terminal

a day ago

2 hours ago

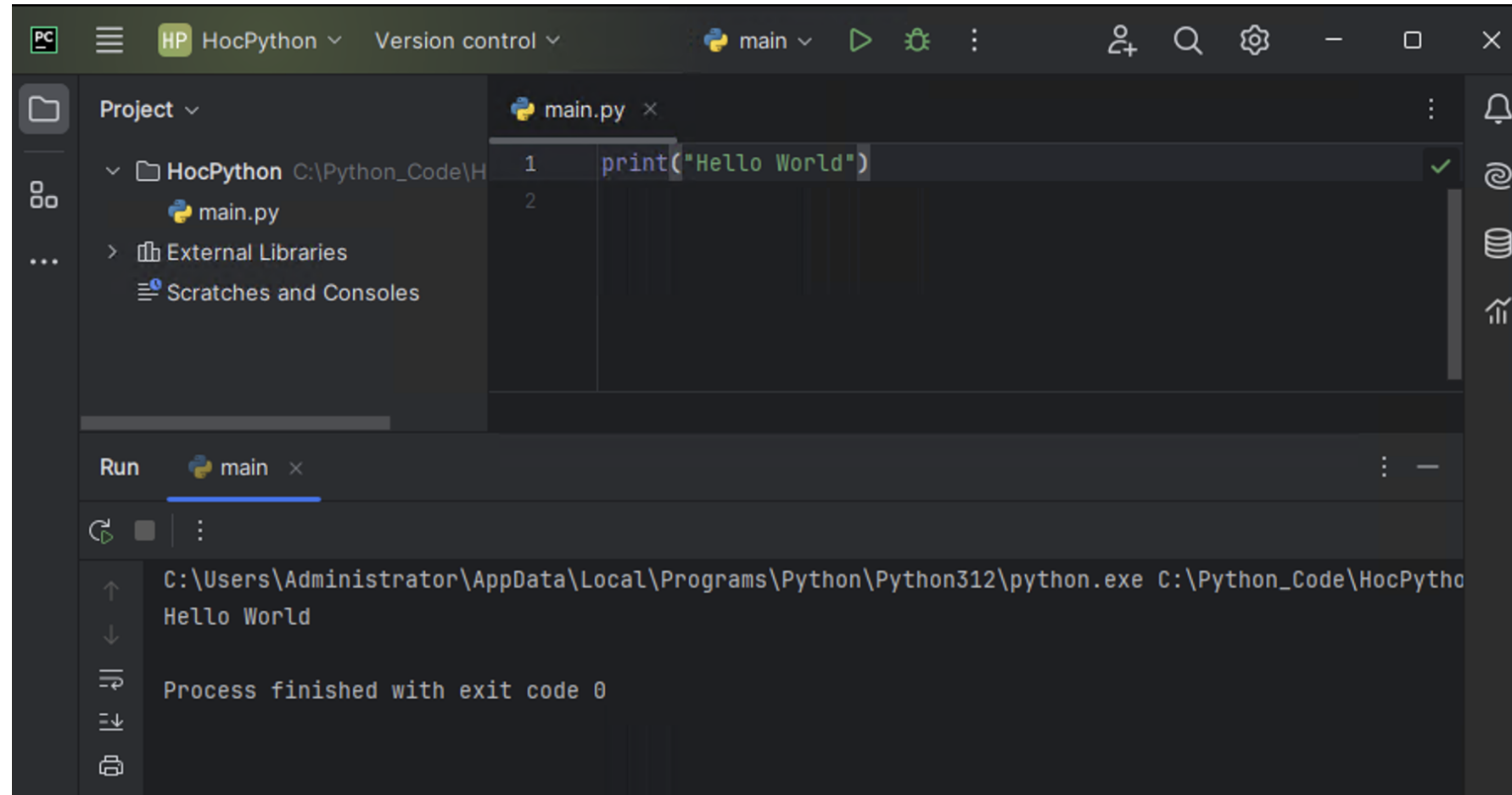
a day ago

a day ago

a day ago

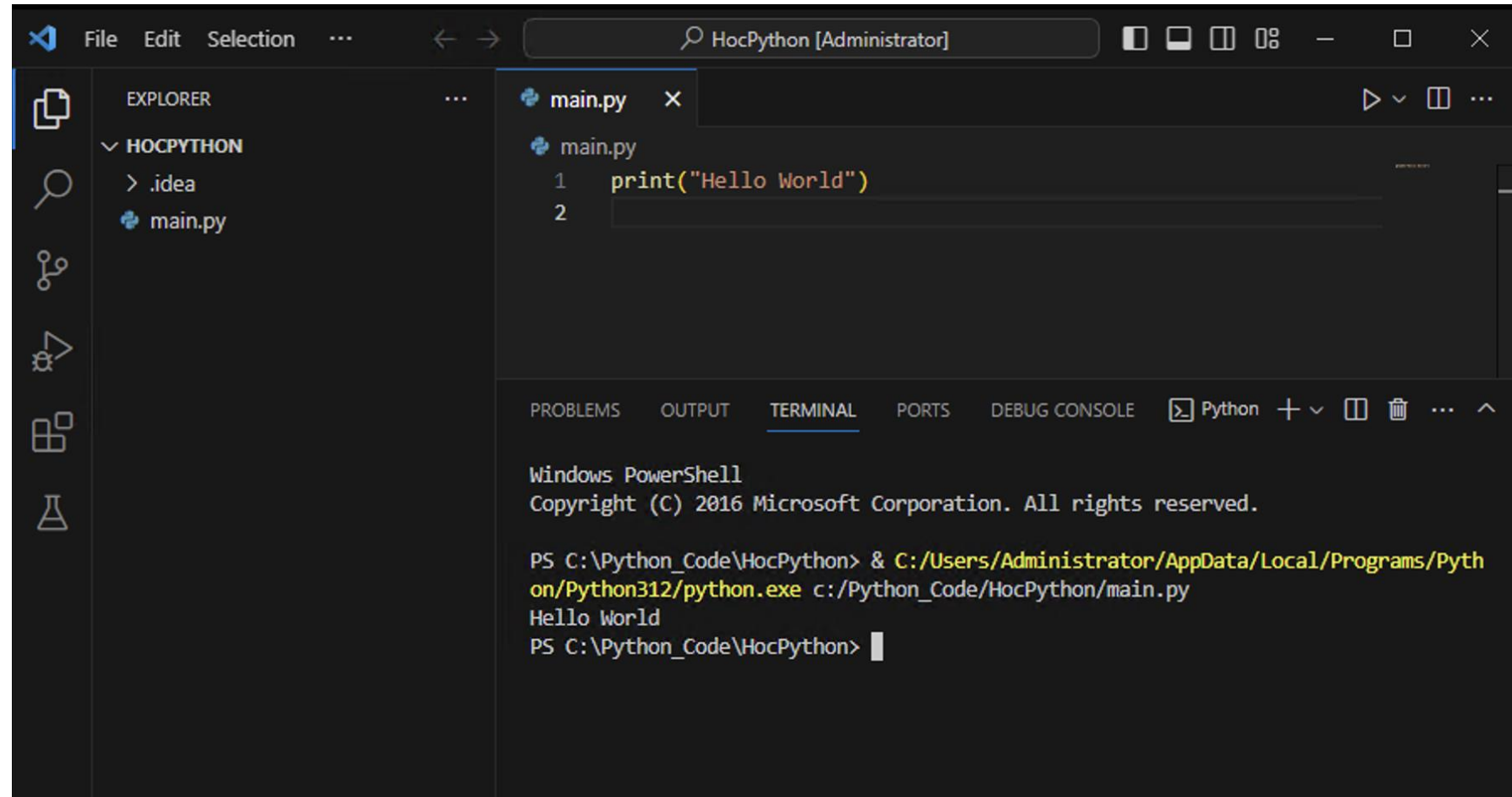
2. Chương trình đầu tiên

Sử dụng Pycharm,
tạo file `.py` và chạy
thử



2. Chương trình đầu tiên (tt)

Sử dụng VSCode,
tạo file **.py** và chạy
thử



The screenshot shows the Visual Studio Code (VS Code) interface. The Explorer panel on the left shows a project named 'HOCPYTHON' with a file named 'main.py'. The main editor window displays the code in 'main.py':

```
1 print("Hello World")
2
```

The TERMINAL panel at the bottom shows the command prompt output:

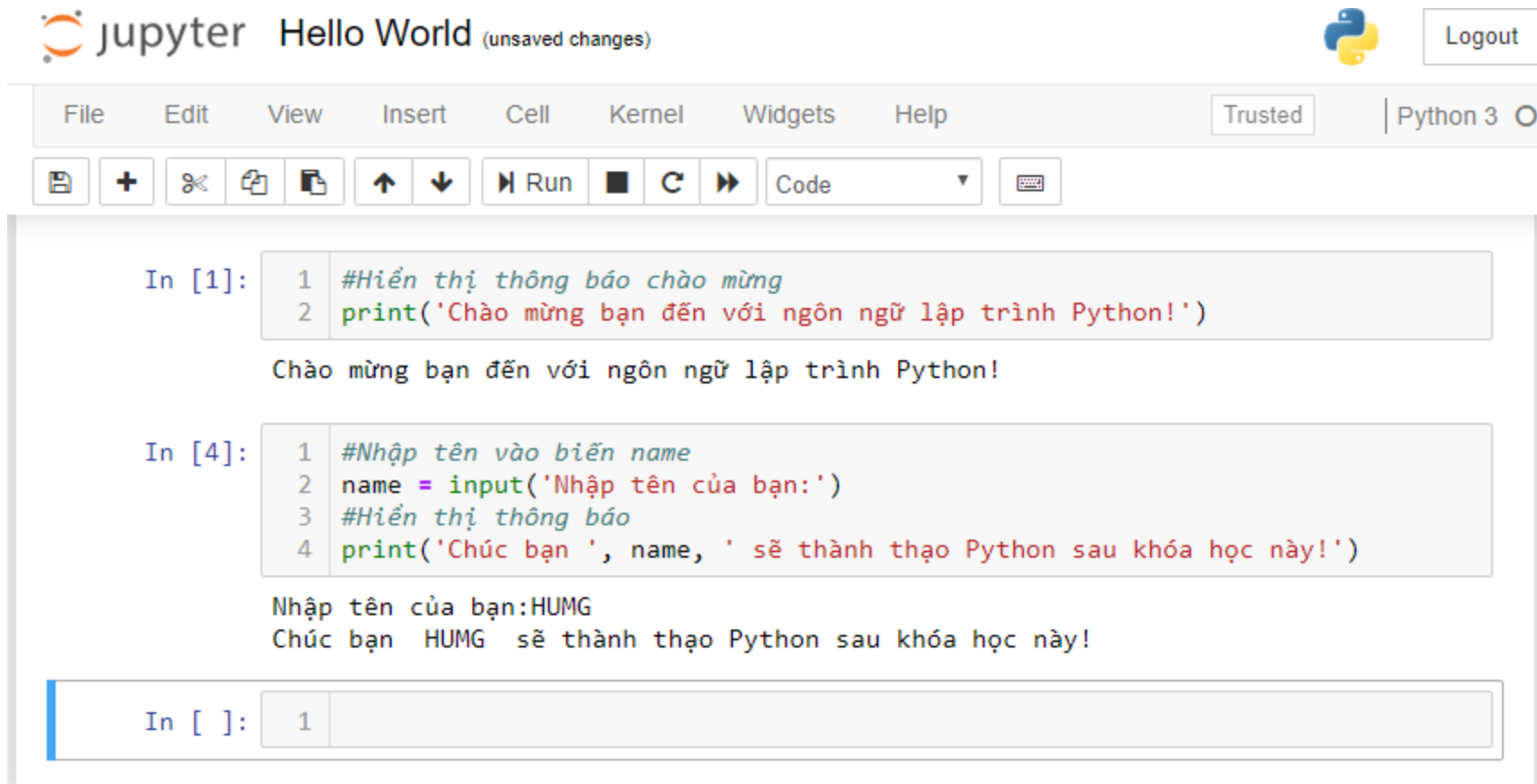
```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.


PS C:\Python_Code\HocPython> & C:/Users/Administrator/AppData/Local/Programs/Python/Python312/python.exe c:/Python_Code/HocPython/main.py
Hello World
PS C:\Python_Code\HocPython>
```

2. Chương trình đầu tiên (tt)












- Tạo thư mục chứa mã nguồn, tạo file đầu tiên “Hello World”

Hãy nhập và chạy những dòng code Python đầu tiên!



```
jupyter Hello World (unsaved changes)  Logout
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

           Code

In [1]:

```
1 #Hiển thị thông báo chào mừng
2 print('Chào mừng bạn đến với ngôn ngữ lập trình Python!')
```

Chào mừng bạn đến với ngôn ngữ lập trình Python!

In [4]:

```
1 #Nhập tên vào biến name
2 name = input('Nhập tên của bạn:')
3 #Hiển thị thông báo
4 print('Chúc bạn ', name, ' sẽ thành thạo Python sau khóa học này!')
```

Nhập tên của bạn:HUMG
Chúc bạn HUMG sẽ thành thạo Python sau khóa học này!

In []:

```
1
```

Thực hành 1

Thực hành 1.1

- Thêm một cell code mới, nhập đoạn mã dưới đây, chạy và cho biết kết quả!

```
In [3]: 1  #khởi tạo biến a = 0; A = 0
        2  a = 0
        3  A = 0
        4  for i in range(1,6):
        5      print(i)
        6      a = a + i
        7
        8  #Hiển thị kết quả biến a
        9  print('Giá trị của biến a = ',a)
       10  print('-----')
       11
       12  for i in range (1,6,2):
       13      print(i)
       14      A = A + i
       15
       16  #HIển thị kết quả biến A
       17  print('Giá trị của biến A = ',A)
       18
```

Thực hành 1.2

- Tạo một file mới, đặt tên: Calculator.ipynb
- Viết code: nhập vào 2 số a, b và thực hiện để hiển thị kết quả như sau:

In [1]:

```
1 #-----  
2 #{Code của bạn ở đây}  
3 #-----  
4
```

Nhập vào số a:9

Nhập vào số b:5

Tổng a + b = 14

Hiệu a - b = 4

Tích a * b = 45

Thương a / b = 1.8

Lưu ý: lệnh `num_a = int(a)` sẽ chuyển đổi kiểu dữ liệu của a sang kiểu int (số nguyên) và gán giá trị cho biến num_a

3. Một số lỗi thường gặp trong lập trình!

So sánh code C – Java - Python

“Hello, World”

- C

```
#include <stdio.h>

int main(int argc, char ** argv)
{
    printf("Hello, World!\n");
}
```

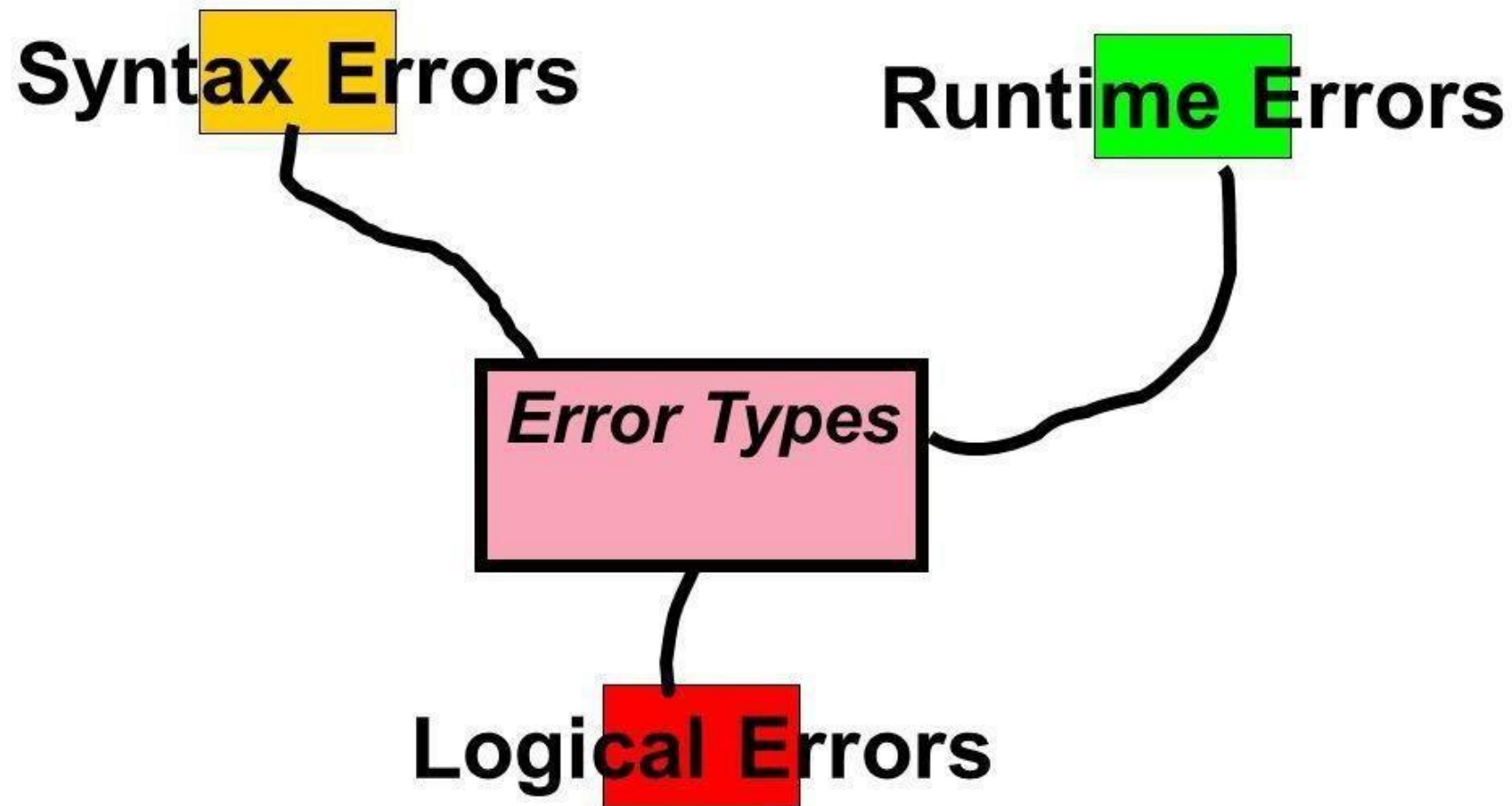
- Java

```
public class Hello
{
    public static void main(String argv[])
    {
        System.out.println("Hello, World!");
    }
}
```

- now in Python

```
print "Hello, World!"
```

Các loại lỗi thường gặp trong lập trình



Các loại lỗi thường gặp trong lập trình

1) Lỗi cú pháp (Syntax Error): Đây là loại lỗi sơ đẳng nhất trong lập trình. Thường là do gõ **sai cấu trúc của ngôn ngữ** (ví dụ như thiếu dấu kết thúc một câu lệnh, một số ngôn ngữ từ khoá phân biệt chữ hoa, chữ thường thì lại gõ chữ hoa, v.v. gọi là lỗi chính tả).

Chương trình sẽ không thể biên dịch được khi gặp lỗi này. Các trình biên dịch khi gặp lỗi ở dòng code nào thì sẽ báo lỗi. Việc tìm và sửa lỗi cú pháp rất đơn giản.

In [2]:

```
1 #Lỗi cú pháp
2 for i in range(1,e):
3     print('Biến :', i)
```

NameError

Traceback (most recent call last)

<ipython-input-2-e981b9106777> in <module>

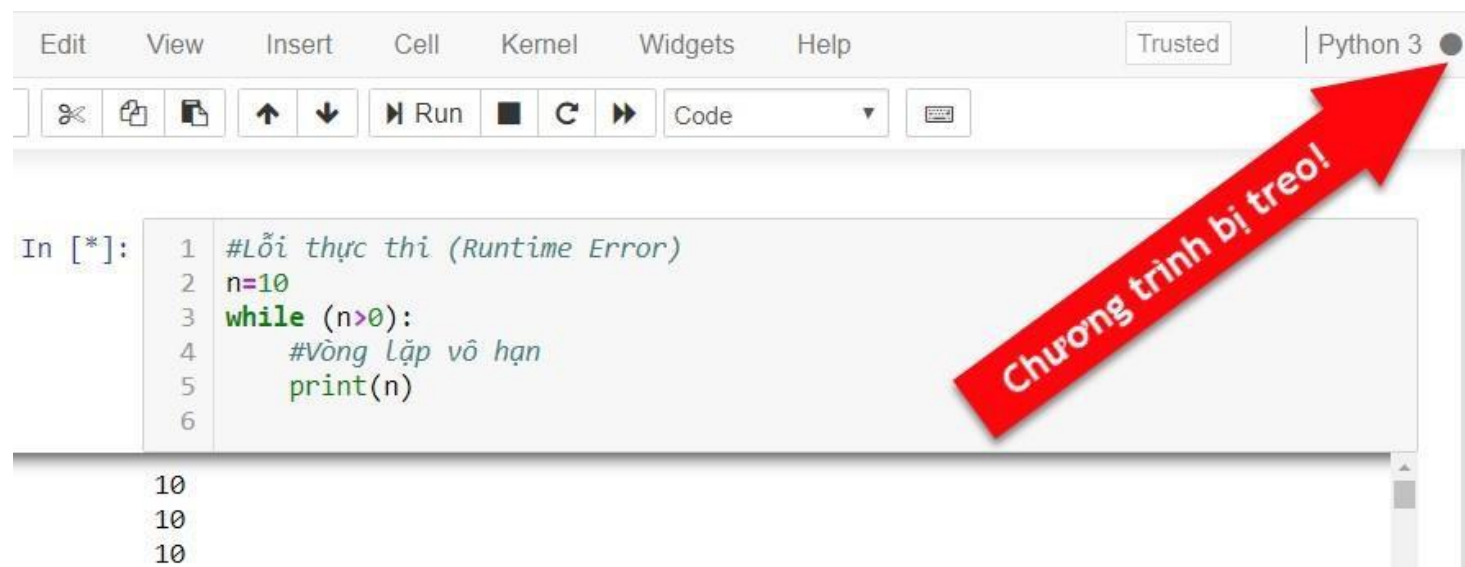
```
1 #Lỗi cú pháp
----> 2 for i in range(1,e):
3     print('Biến :', i)
```

NameError: name 'e' is not defined

(Tham khảo 2) Các loại lỗi thường gặp trong lập trình

2) Lỗi thực thi (Runtime error): Xảy ra bất ngờ khi chương trình đang chạy. Loại lỗi này thường xảy ra do người lập trình viết code ẩu, không lường hết các trường hợp xảy ra, khiến chương trình đang chạy thì bị lỗi **treo màn hình**, **thoát khỏi chương trình** hoặc thoát luôn chương trình, v.v. Lỗi này có thể dễ dàng phát hiện bằng cách Debug

Nên bổ xung *cấu trúc xử lý ngoại lệ* **try ... catch ... finally...** để hạn chế lỗi thực thi



```

Edit View Insert Cell Kernel Widgets Help Trusted Python 3
✂️ 📄 📁 ⬆️ ⬇️ ▶️ Run ■ ↺ ▶️ Code
In [*]: 1 #Lỗi thực thi (Runtime Error)
        2 n=10
        3 while (n>0):
        4     #Vòng lặp vô hạn
        5     print(n)
        6
        10
        10
        10

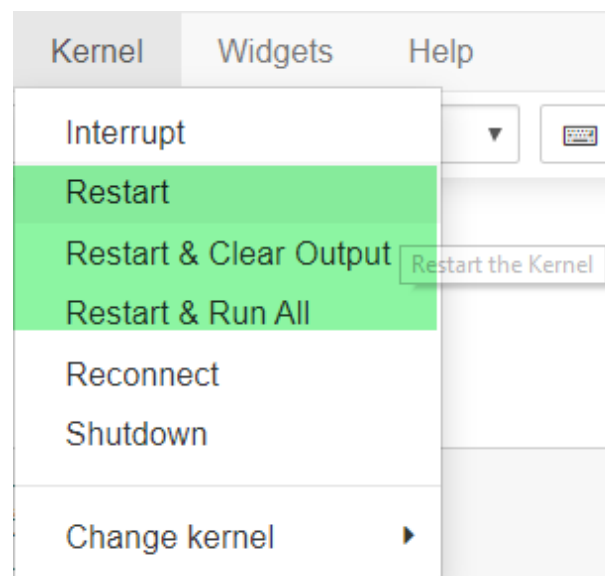
```

Chương trình bị treo!

(Tham khảo 2) Các loại lỗi thường gặp trong lập trình

Trong trường hợp dấu tròn bên cạnh chữ Python 3 chuyển sang màu đen trong thời gian dài và không thể thực hiện được bất kỳ thao tác nào khác với cell, khi đó chương trình đang bị treo.

Để thoát khỏi tình huống này có thể khởi động lại kernel (**restart the Kernel**)



(Tham khảo 2) Các loại lỗi thường gặp trong lập trình

3) Lỗi thuật toán (Logical Error):

Đây là lỗi do tư duy sai, thuật toán sai dẫn đến sai kết quả. Đây là loại lỗi **khó phát hiện nhất**, thường phải sử dụng chương trình, thậm chí là dùng lâu mới phát hiện ra được. Việc debug lỗi này cũng là một việc tốn khá nhiều thời gian và công sức. Để phát hiện lỗi này thì chương trình cần chạy nhiều lần với nhiều kết quả để xem nó có phù hợp hay không.

In [6]:

```
1 #Lỗi thuật toán (Logical Errors)
2 #Tìm quãng đường di chuyển của xe biết Vận tốc của xe: 60 km/h
3
4 thoigian = input('Nhập thời gian xe chạy (h):')
5 #Tính quãng đường
6 quangduong = thoigian*60
7
8 #Hiển thị kết quả
9 print('Quãng đường xe chạy được là: ', quangduong, ' (Km)')
10
```

Chương trình chạy nhưng kết quả không đúng!

Nhập thời gian xe chạy (h):3

[illegible]

Ví dụ lỗi Cú pháp

Python không cung cấp các dấu ngoặc ôm (`{}`) để chỉ các khối code cho định nghĩa lớp hoặc hàm hoặc điều khiển luồng.

Các khối code được nhận biết bởi độ thụt dòng code (indentation) trong Python và đây là điều bắt buộc.

Số khoảng trống trong độ thụt dòng có thể khác nhau, nhưng tất cả các lệnh bên trong khối phải được thụt cùng một số lượng khoảng trống như nhau.

Ví dụ lỗi Cú pháp

Tìm Lỗi cú pháp

```
if True:
    print ("True")
else:
    print ("False")
```

```
if True:
    print ("Answer")
    print ("True")
else:
    print ("Answer")
    print ("False")
```


Ví dụ lỗi Cú pháp

Code đúng:

```
if True:
    print ("True")
else:
    print ("False")
```

Code sai:

```
if True:
    print ("Answer")
    print ("True")
else:
    print ("Answer")
    print ("False")
```

4. Biến và khai báo biến trong Python



1.1 Biến (VARIABLE)

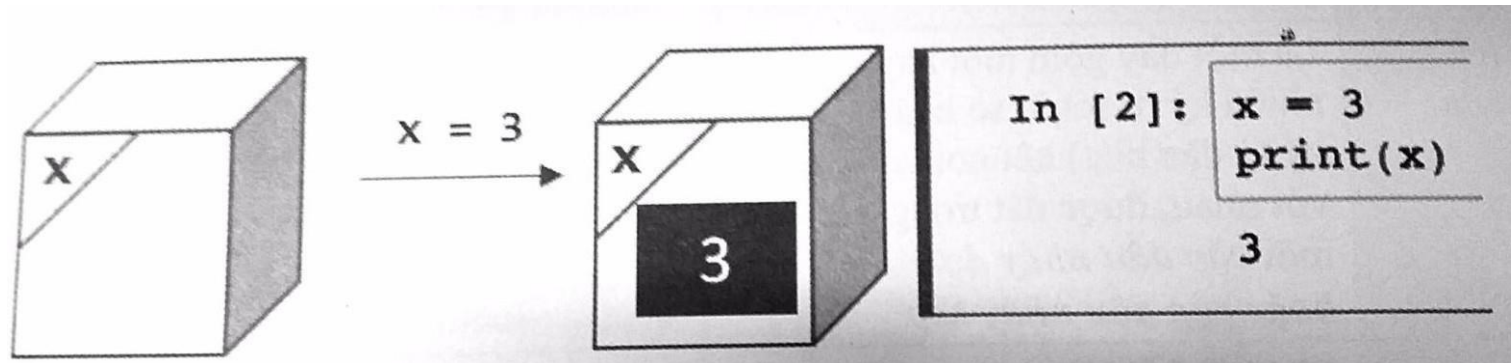
- Trong lập trình, biến (variable) là **tên của một vùng trong bộ nhớ RAM**, được sử dụng để **lưu trữ thông tin**.
- Biến giống như **một chiếc hộp** có thể giúp chúng ta lưu trữ dữ liệu. Ta có thể gán thông tin cho một biến, và có thể lấy thông tin đó ra để sử dụng.
- Khi một biến được khai báo, một vùng trong bộ nhớ sẽ dành cho các biến.

```
In [3]: message = 'CHXH' #tạo biến message nhận giá trị là 'CHXH'  
n = 17 # tạo biến n nhận giá trị là 17  
pi = 3.1415926535897931 # tạo biến pi nhận giá trị là 3.14...
```

Khai báo Biến (VARIABLE)

Cú pháp: **tên_biến =**
<giá trị>

```
In [3]: message = 'CHXH' #tạo biến message nhận giá trị là 'CHXH'  
n = 17 # tạo biến n nhận giá trị là 17  
pi = 3.1415926535897931 # tạo biến pi nhận giá trị là 3.14...
```



Vai trò của Biến

- Biến giúp chúng ta lưu trữ các dữ liệu và cho phép chúng ta lấy các dữ liệu của chúng để tính toán được thuận tiện và chính xác hơn.
- Khi muốn lấy dữ liệu từ một nguồn không cố định từ ngoài chương trình (ví dụ: dữ liệu người dùng nhập vào, dữ liệu load từ file ...) ta cần sử dụng biến số để lưu thông tin của các dữ liệu này

```
In [4]: 52348252408 + 523482034
```

```
Out[4]: 52871734442
```

```
In [5]: 52871734442 + 545354645577
```

```
Out[5]: 598226380019
```

```
In [6]: a = 52348252408  
b = 523482034  
c = 545354645577  
d = a + b + c  
print(d)
```

```
598226380019
```

Quy tắc đặt tên Biến

Một định danh (identifier) trong Python được sử dụng để nhận diện một biến, một hàm, một lớp, hoặc một đối tượng.

- Một định danh bắt đầu với một chữ cái từ A tới Z hoặc từ a tới z hoặc một dấu gạch dưới (_) hoặc các chữ số (từ 0 tới 9). Ví dụ: `_a`, `aB`, `Ab...`
- Python không hỗ trợ các ký tự đặc biệt chẳng hạn như `@`, `$` và `%` bên trong các định danh.

Ví dụ: Định danh biến nào sau đây là hợp lệ?
`variable_name`, `variableName`, `_variable`, `variable1`,
`1variable`, `variable-name`, `variable name`

```
In [9]: 76ab = 'asd'

File "<ipython-input-9-d8355cbf35f4>", line 1
      76ab = 'asd'
          ^
SyntaxError: invalid syntax
```

```
In [8]: them@ = 1000000

File "<ipython-input-8-0de45931b145>", line 1
      them@ = 1000000
          ^
SyntaxError: invalid syntax
```

```
In [10]: class = 'Advanced Theoretical Herpetology'

File "<ipython-input-10-9630266a2671>", line 1
      class = 'Advanced Theoretical Herpetology'
          ^
SyntaxError: invalid syntax
```

Gán giá trị cho biến

Trong Python, không cần khai báo biến một cách tường minh. Phép gán được thực hiện bởi toán tử =.

Toán hạng trái của toán tử = là tên biến và toán hạng phải là giá trị được lưu trữ trong biến.

```
a = 10          # Mot phép gan so nguyen
b = 10.01       # Mot so thuc
name = "Hoang"  # Mot chuoì
print (a)
print (b)
print (name)
```

Phép đa gán

Python cho phép gán một giá trị đơn cho một số biến đồng thời. Python hỗ trợ hai kiểu đa gán sau:

Gán giá trị đơn cho nhiều biến.

```
a = b = c = 3
```

Gán nhiều giá trị cho nhiều biến (Trong trường hợp này, các giá trị sẽ được gán theo thứ tự mà các biến xuất hiện).

```
one, two, three = 1, 2, 3
```


Từ khóa

Là các từ dành riêng và KHÔNG thể sử dụng để khai báo các hằng, biến hoặc cho bất kỳ tên định danh nào. Tất cả từ khóa trong Python là dạng chữ thường.

Bảng dưới liệt kê các từ khóa trong Python:

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

5. Các kiểu dữ liệu cơ bản trong Python

1.2 Kiểu dữ liệu trong Python

- Python hỗ trợ nhiều kiểu dữ liệu khác nhau

Type	Example
• Numeric: Integer, Float	<code>x = 10 x = 1.0</code>
• String	<code>x= 'Mike'</code>
• Boolean	<code>y = True x = False</code>
• List	<code>my_list = [10, 20, 30]</code>
• Tuple	<code>my_tuple = ('Brett', 'Cisco', 'Cary', 2015)</code>
• Dictionary	<code>my_dict = {"one":1, "two":2}</code>
• Lists in Lists	<code>my_list2=[[10,20,30], ['Cisco Live', 'May', 2016]]</code>

Python
Data types

a) Kiểu dữ liệu số

Bao gồm 2 kiểu:

- Số nguyên – int (ví dụ: 1, 4, 111)
- Số thực – float (ví dụ: 1.1, 3.23, 11.01)

In [13]:

```
a=11  
b=12  
c=a/b  
print(c)
```

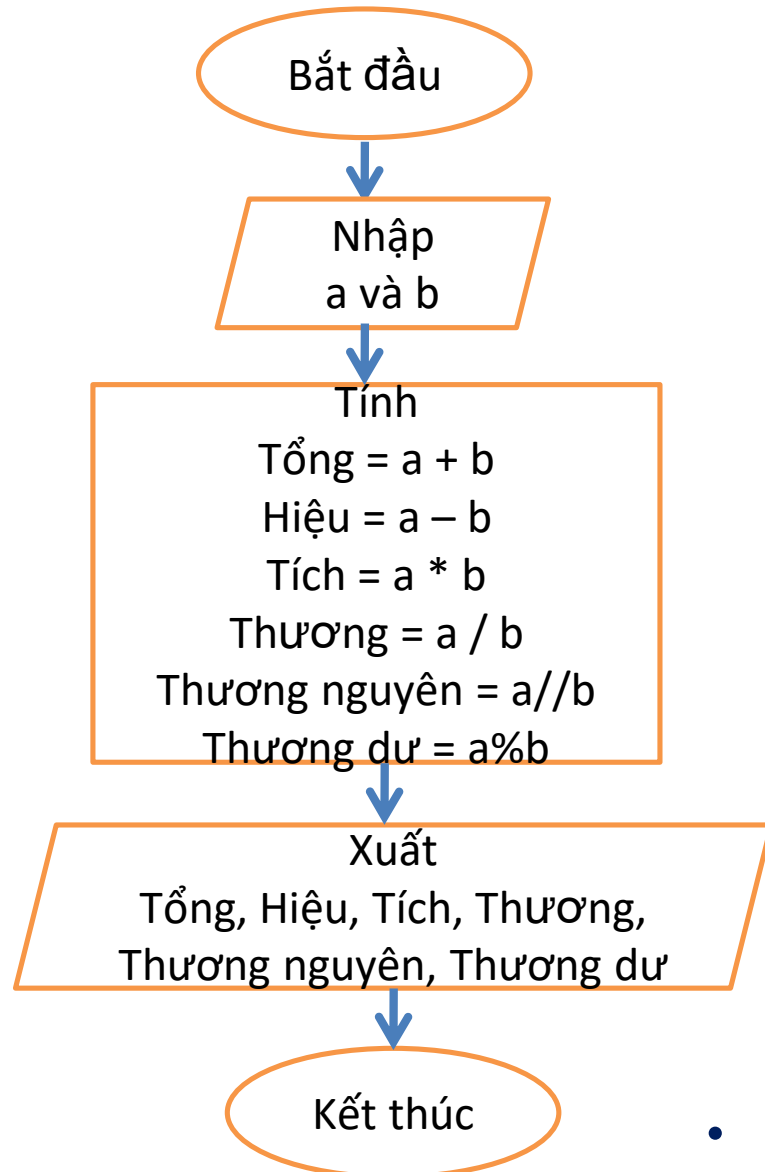
0.9166666666666666

In [14]:

```
a=11  
b=12  
c=int(a/b)  
print(c)
```

0

Các phép toán với số



```
1 a = 10
2 b = 8
3 #-----
4 tong = a + b      # Tổng của hai số (+)
5 hieu = a - b      # Hiệu của hai số (-)
6 tinh = a*b        # Tích của hai số (*)
7 thuong = a/b       # Thương của hai số (/)
8 thuong_nguyen = a//b # Phép chia lấy phần nguyên (//)
9 thuong_du = a % b  # Phép chia lấy phần dư (%)
10 mu = a**b         # Tính giá trị a lũy thừa b (**)
```

- **Sinh viên nhập code và đọc kết quả của các phép toán ở trên!**

Kiểm tra kiểu dữ liệu của biến

- Biến trong Python rất linh hoạt, có thể chứa giá trị thuộc nhiều kiểu dữ liệu khác nhau.

Cú pháp: **type(biến)** sẽ giúp ta biết được kiểu dữ liệu của biến

```
1  #Kiểm tra kiểu dữ liệu của biến
2  x = 1985
3  y = 3.1415926535
4  z = 'Đại học Mo Dia chat'
5  n = [5, 7, 9, 8]
6  b = True
7  #-----
8  print('Kiểu dữ liệu biến x: ', type(x))
9  print('Kiểu dữ liệu biến y: ', type(y))
10 print('Kiểu dữ liệu biến z: ', type(z))
11 print('Kiểu dữ liệu biến n: ', type(n))
12 print('Kiểu dữ liệu biến b: ', type(b))
```

```
Kiểu dữ liệu biến x: <class 'int'>
Kiểu dữ liệu biến y: <class 'float'>
Kiểu dữ liệu biến z: <class 'str'>
Kiểu dữ liệu biến n: <class 'list'>
Kiểu dữ liệu biến b: <class 'bool'>
```

b) Kiểu dữ liệu chuỗi

- Bên cạnh số, Python cũng có thể thao tác với chuỗi ký tự
- Chuỗi có thể được để trong dấu nháy đơn ('...') hoặc kép ("...")

```
In [2]: c='hung là trường đại học hàng đầu tại Việt Nam'
        print(c)
```

hung là trường đại học hàng đầu tại Việt Nam

```
In [3]: c="hung là trường đại học hàng đầu tại Việt Nam"
        print(c)
```

hung là trường đại học hàng đầu tại Việt Nam

```
1 #Kiểu dữ liệu số: a = 123
2 a = 123
3 print('Số: ', a)
4
5 #Kiểu dữ liệu chuỗi ký tự: A = '123'
6 A='123'
7 print('Chuỗi: ', A)
```

Số: 123

Chuỗi: 123

- Cần **phân biệt** kỹ giữa **kiểu dữ liệu chuỗi** và **số**:

Truy cập phần tử trong chuỗi

- Các chuỗi có thể được lập chỉ mục với ký tự đầu tiên được đánh số 0.
- Chỉ số cũng có thể là số âm, bắt đầu đếm từ bên phải
- Ngoài việc đánh số, thì cắt lát cũng được hỗ trợ. Trong khi index được sử dụng để lấy các ký tự riêng lẻ thì cắt lát sẽ được dùng để lấy chuỗi con.

```
+---+---+---+---+---+---+
| P | y | t | h | o | n |
+---+---+---+---+---+---+
0   1   2   3   4   5
      -5  -4  -3  -2  -1
```

```
In [4]: print(c[1])
```

u

```
In [5]: print(c[-1])
```

m

```
In [6]: print(c[0:3])
```

hum

Một số thao tác với chuỗi

- Cho biết số ký tự của chuỗi: **len(st)**

```
1 st = 'HUMG là Trường đại học hàng đầu tại Việt Nam'  
2 print('Số ký tự trong chuỗi st là: ', len(st))
```

Số ký tự trong chuỗi st là: 44

- Chuyển chuỗi st thành chữ hoa (upper) – chữ thường (lower): **st.upper()** – **st.lower()**

```
1 st = 'HUMG là Trường đại học hàng đầu tại Việt Nam'  
2 print('Chữ hoa: ', st.upper())  
3 print('Chữ thường: ', st.lower())
```

Chữ hoa: HUMG LÀ TRƯỜNG ĐẠI HỌC HÀNG ĐẦU TẠI VIỆT NAM

Chữ thường: humg là trường đại học hàng đầu tại việt nam

Một số thao tác với chuỗi (t)

- Loại bỏ khoảng trắng ở đầu và cuối chuỗi: **st.strip()**

```
1 st1 = '  Tôi  yêu Việt Nam!  '
2 st1.strip()
```

'Tôi yêu Việt Nam!'

- Thay thế một chuỗi s thành s1 trong chuỗi st: **st.replace(s,s1)**

```
1 st = 'HUMG là Trường đại học hàng đầu tại Việt Nam'
2 st_new= st.replace('Việt Nam', 'VN')
3 print(st_new)
```

HUMG là Trường đại học hàng đầu tại VN

Một số thao tác với chuỗi (t)

Để kiểm tra một chuỗi con có trong chuỗi st hay không? hãy dùng từ khóa **in**

```
1 st = 'HUMG là Trường đại học hàng đầu tại Việt Nam'
2 x = 'HUMG' in st
3 y = 'Python' in st
4 #Trả về True nếu có trong chuỗi st
5 #Trả về False nếu không có trong chuỗi st
6 print('Kết quả kiểm tra HUMG:', x)
7 print('Kết quả kiểm tra Python:', y)
```

Kết quả kiểm tra HUMG: True

Kết quả kiểm tra Python: False

Một số thao tác với chuỗi (t)

- Tách một chuỗi thành một danh sách (mảng 1 chiều):

`string.split('separator')`

```
1 st = 'HUMG là Trường đại học hàng đầu tại Việt Nam'
2 #Mặc định sẽ tách chuỗi tại vị trí dấu cách
3 list_st = st.split()
4 print(list_st)
```

```
['HUMG', 'là', 'Trường', 'đại', 'học', 'hàng', 'đầu', 'tại',  
'Việt', 'Nam']
```

```
1 st1 = 'Mai Phương Thúy, 1985, Hà Nội'
2 #Tùy chọn vị trí tách theo tham số đưa vào
3 list_st = st1.split(',')
4 print(list_st)
```

```
['Mai Phương Thúy', ' 1985', ' Hà Nội']
```

Phương thức kiểm tra với chuỗi

- **st.isalpha()** : Kiểm tra chuỗi ký tự st có phải chỉ chứa ký tự thuộc bảng chữ cái hay không (a-z, A-Z)?
- **st.isdigit()** : Kiểm tra chuỗi ký tự st có phải chỉ chứa ký tự chữ số hay không (0-9)?
- **st.isalnum()** : Kiểm tra chuỗi ký tự st có phải chỉ chứa ký tự chữ cái (a-z, A-Z) và ký tự số (0-9) hay không?
- **st.isupper()** : Kiểm tra chuỗi ký tự st có phải chỉ chứa ký tự chữ được viết hoa hay không (A-Z)?
- **st.islower()** : Kiểm tra chuỗi ký tự st có phải chỉ chứa ký tự chữ được viết hoa hay không (a-z)?

Chuyển đổi kiểu dữ liệu

- **str(biến_a)** : Chuyển kiểu dữ liệu của biến_a về kiểu chuỗi
- **int(biến_b)** : Chuyển kiểu dữ liệu của biến_b về kiểu số nguyên
- **float(biến_c)** : Chuyển kiểu dữ liệu của biến_c về kiểu số thực

```
1 a_chu = '1985'
2 a_so = 2019
3 #Chuyển từ chuỗi sang số
4 so_nguyen = int(a_chu)
5 so_thuc = float(a_chu)
6 print('Kiểu dữ liệu biến so_nguyen: ', type(so_nguyen), ' - giá trị: ', so_nguyen)
7 print('Kiểu dữ liệu biến so_thuc: ', type(so_thuc), ' - giá trị: ', so_thuc)
8
9 #Chuyển đổi từ số sang chuỗi
10 chuoi = str(a_so)
11 print('Kiểu dữ liệu biến chuoi: ', type(chuoi), ' - giá trị: ', chuoi)
```

```
Kiểu dữ liệu biến so_nguyen: <class 'int'> - giá trị: 1985
Kiểu dữ liệu biến so_thuc: <class 'float'> - giá trị: 1985.0
Kiểu dữ liệu biến chuoi: <class 'str'> - giá trị: 2019
```


Thực hành 2

Bài 1: Chia kẹo cho học sinh lớp 1

Cô có n cái kẹo, có m học sinh trong lớp. Hãy giúp cô chia đều số kẹo cho tất cả học sinh.

- **Yêu cầu:**
 - Nhập vào số kẹo của cô – N
 - Nhập vào số học sinh trong lớp – M
- **Cho biết:**
 - Mỗi học sinh được nhận bao nhiêu cái kẹo?
 - Cô còn lại bao nhiêu cái kẹo?

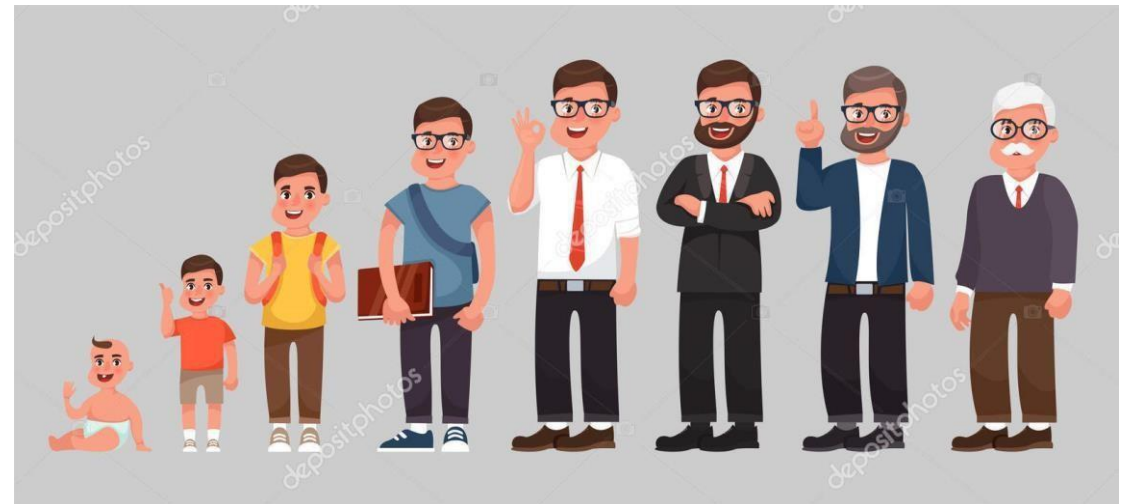


Bài 2: Tính tuổi

Nhập vào tên và năm sinh của một người, tính tuổi của người đó và hiển thị ra màn hình thông báo

- **Yêu cầu:**
 - Nhập vào họ tên: mai phương thúy
 - Nhập vào năm sinh: 2000
- **Hiển thị thông báo kết quả:**

Bạn “MAI PHƯƠNG THÚY” năm nay 24 tuổi!



Bài 3: Cho biết số thỏ trong rừng

Ban đầu trong rừng có một cặp thỏ, biết rằng cứ sau 1 tháng thì số lượng thỏ trong rừng tăng lên gấp đôi. Hỏi sau x tháng thì trong rừng có bao nhiêu con thỏ (Giả thiết các con thỏ không chết)

- Yêu cầu:
 - Nhập vào số tháng x : 3
- **Hiển thị thông báo kết quả:**

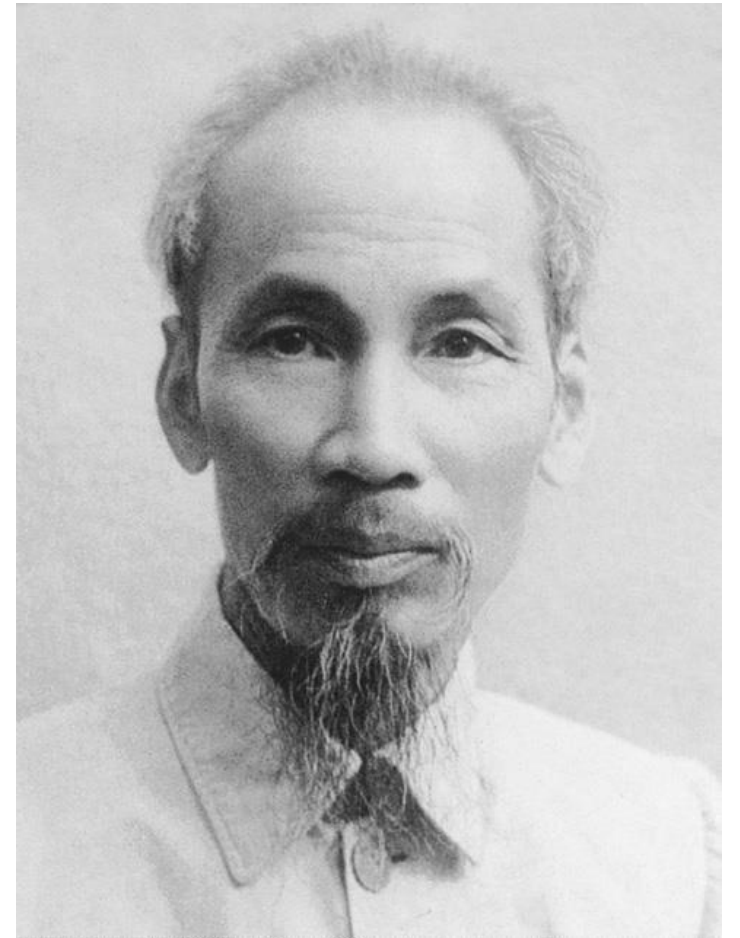
Trong rừng có: 16 con thỏ



Bài 4: Chuỗi văn bản

“Nước Việt Nam là một, dân tộc Việt Nam là một. Sông có thể cạn núi có thể mòn, song chân lý ấy không bao giờ thay đổi. (HỒ CHÍ MINH, 1890 – 1969)”

- **Yêu cầu:**
 - Cho biết số ký tự có trong đoạn văn trên.
 - Cho biết trong đoạn có chứa từ nào sau đây không (Không phân biệt chữ hoa, chữ thường)?
 - “hồ chí minh”
 - “non sông”
 - Tách đoạn văn thành các câu bởi dấu .
 - Cho biết trong đoạn văn trên có ký tự nào khác ký tự chữ và số hay không?
 - Thay thế các từ ‘Việt Nam’ bằng ‘VIỆT NAM’ trong đoạn văn trên.

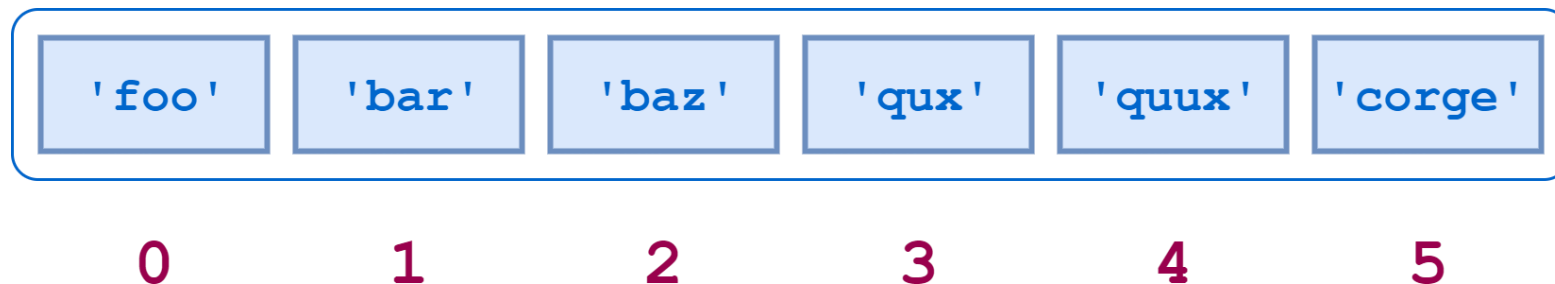


c) Kiểu dữ liệu danh sách (list)

- Giả sử ta có nhu cầu lưu danh sách tên tất cả học sinh trong lớp học. Nếu lớp học có:

```
1 #khai báo danh sách học sinh
2 hoc_sinh1 = 'Lê Thùy Dung'
3 hoc_sinh2 = 'Trần Đức Hùng'
4 hoc_sinh3 = 'Nguyễn Lan Anh'
```

- Nhưng nếu trong lớp có hàng chục, hàng trăm học sinh thì điều đó thật sự rất khó khăn, khó kiểm soát, quản lý.
- ➔ Kiểu dữ liệu danh sách (list) ra đời giúp chúng ta có thể khai báo, lưu trữ, truy xuất một dãy các đối tượng.



c) Kiểu dữ liệu danh sách (list)

- Khai báo list:

<tên_danh_sách> = [<Phần tử 1>,<Phần tử 2>,...,<Phần tử n>]

```
1  #KHAI BÁO DANH SÁCH LIST
2  #khai báo danh sách hoc_sinh gồm các chuỗi tên của học sinh
3  hoc_sinh=['Lê Thùy Dung','Trần Đức Hùng','Nguyễn Lan Anh','Mai Phương Thúy']
4  print(hoc_sinh)
5
6  #Khai báo danh sách diem_chu gồm các chuỗi ký tự
7  diem = ['A+','A','B+','B','C+','C','D+','D','F']
8  print(diem)
9
10 #Khai báo danh sách gồm các số nguyên
11 list_so = [9,5,8,13,0,4,7,-9,11]
12 print(list_so)
13
14 #Khai báo danh sách với nhiều kiểu dữ liệu khác nhau
15 person_info = ['Mary',1998,'Tokyo, Japan', 1.70, 65]
16 print(person_info)
17
18 #Khai báo danh sách rỗng
19 list_rong = []
20 print(list_rong)
```

['Lê Thùy Dung', 'Trần Đức Hùng', 'Nguyễn Lan Anh', 'Mai Phương Thúy']

['A+', 'A', 'B+', 'B', 'C+', 'C', 'D+', 'D', 'F']

[9, 5, 8, 13, 0, 4, 7, -9, 11]

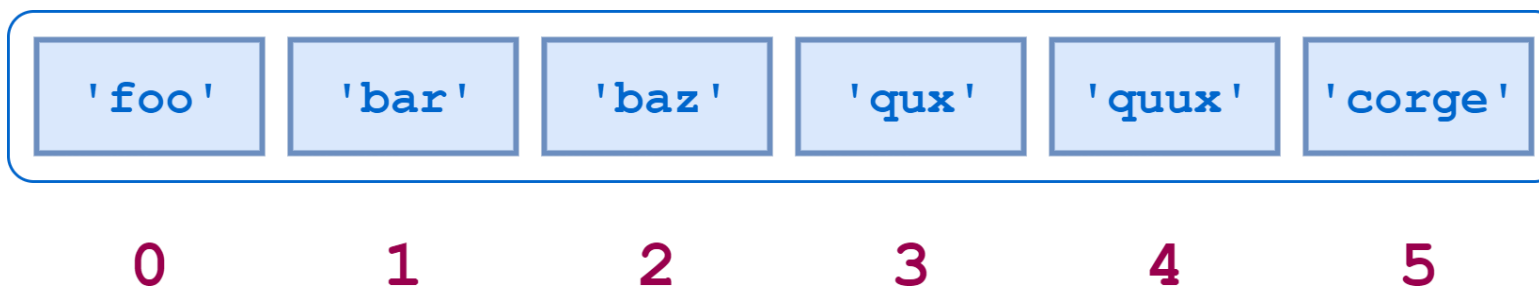
['Mary', 1998, 'Tokyo, Japan', 1.7, 65]

[]

Truy xuất phần tử trong danh sách

Để truy cập lấy giá trị tại một phần tử trong danh sách

<tên_danh_sách> [<chỉ số>]



```
1 #Truy xuất dữ liệu trong danh sách:
2 #Hiển thị tên học sinh ở vị trí thứ 3
3 print('Học sinh vị trí thứ 3: ', hoc_sinh[2])
4
5 #hiển thị tên người - chiều cao trong biến person_info
6 print('Họ tên: ', person_info[0], ' ---Chiều cao:', person_info[3])
7
8 #Truy xuất nhiều phần tử trong danh sách
9 print(list_so[3:8])
```

Học sinh vị trí thứ 3: Nguyễn Lan Anh

Họ tên: Mary ---Chiều cao: 1.7

[13, 0, 4, 7, -9]

Thay đổi giá trị phần tử trong list

Để thay đổi giá trị tại một phần tử trong danh sách:

<tên_danh_sách> [<chỉ số>] = <giá trị mới>

```
1 #Thay đổi giá trị của một phần tử trong danh sách
2 hoc_sinh=['Lê Thùy Dung','Trần Đức Hùng','Nguyễn Lan Anh','Mai Phương Thúy']
3 print('Ban đầu: ', hoc_sinh)
4 hoc_sinh[2] = 'Nguyễn Thị Lan Anh'
5 print('Thay đổi: ', hoc_sinh)
```

Ban đầu: ['Lê Thùy Dung', 'Trần Đức Hùng', 'Nguyễn Lan Anh', 'Mai Phương Thúy']
Thay đổi: ['Lê Thùy Dung', 'Trần Đức Hùng', 'Nguyễn Thị Lan Anh', 'Mai Phương Thúy']

```
1 list_so = [9,5,8,13,0,4,7,-9,11]
2 print('Ban đầu: ', list_so)
3
4 #Thay giá trị của phần tử cuối cùng trong list = 0
5 list_so[-1] = 0
6 print('Thay đổi:', list_so)
7
```

Ban đầu: [9, 5, 8, 13, 0, 4, 7, -9, 11]
Thay đổi: [9, 5, 8, 13, 0, 4, 7, -9, 0]

Các thao tác khác với list

- **Nối hai danh sách (+):** kết quả trả về một danh sách mới.

```
1 list_a = [8,4,8,2]
2 list_b = [3,0,7,6,5]
3 #----Cộng hai danh sách (+)----:
4 list_ab = list_a + list_b
5 print('List mới: ', list_ab)
```

List mới: [8, 4, 8, 2, 3, 0, 7, 6, 5]

- **Lấy độ dài của danh sách len(list):** kết quả trả về một số nguyên là độ dài của danh sách

```
1 #Lấy độ dài của danh sách list_ab: len(list_ab)
2 print(list_ab, ' Có số phần tử là: ', len(list_ab))
```

[8, 4, 8, 2, 3, 0, 7, 6, 5] Có số phần tử là: 9

Các thao tác khác với list (tt)

- **Kiểm tra một phần tử có thuộc danh sách hay không (in):** kết quả trả về một danh sách mới.

```
1  #Kiểm tra phần tử có thuộc danh sách ko?
2  print(list_ab)
3
4  #Kiểm tra phần tử 0:
5  bol_0 = 0 in list_ab
6  print('Phần tử 0 có thuộc list_ab ? ', bol_0)
7
8  #Kiểm tra phần tử 9:
9  bol_9 = 9 in list_ab
10 print('Phần tử 9 có thuộc list_ab ? ', bol_9)
```

[8, 4, 8, 2, 3, 0, 7, 6, 5]

Phần tử 0 có thuộc list_ab ? True

Phần tử 9 có thuộc list_ab ? False

Các thao tác khác với list (tt)

- **Thêm một phần tử vào danh sách:** có hai cách

- Thêm phần tử vào cuối danh sách:

`<danh_sách>.append(<giá trị>)`

- Thêm phần tử vào vị trí bất kỳ trong danh sách:

`<danh_sách>.insert(<vị trí>,<giá trị>)`

```
1  #Thêm phần tử vào danh sách:
2  print('Danh sách ban đầu: \n', list_ab)
3
4  #Thêm vào cuối danh sách:
5  list_ab.append(10)
6  print('Danh sách thêm vào cuối:\n', list_ab)
7
8  #Thêm vào vị trí bất kỳ trong danh sách
9  list_ab.insert(1,100)
10 print('Danh sách thêm vào vị trí thứ 2: \n', list_ab)
```

Danh sách ban đầu:

[8, 4, 8, 2, 3, 0, 7, 6, 5]

Danh sách thêm vào cuối:

[8, 4, 8, 2, 3, 0, 7, 6, 5, 10]

Danh sách thêm vào vị trí thứ 2:

[8, 100, 4, 8, 2, 3, 0, 7, 6, 5, 10]

Các thao tác khác với list (t)

- **Xóa phần tử khỏi danh sách:**
 - Xóa phần tử tại vị trí cuối cùng trong danh sách:
`<danh_sách>.pop()`
 - Xóa phần tử tại vị trí xác định trong danh sách:
`del <danh_sách>[<vị trí>]`
 - Xóa phần tử có giá trị xuất hiện đầu tiên trong danh sách:
`<danh_sách>.remove(<giá trị>)`

Các thao tác khác với list (t)

```
1  #Xóa phần tử khỏi danh sách:
2  print('Danh sách ban đầu: \n', list_ab)
3
4  #Xóa phần tử cuối
5  list_ab.pop()
6  print('Danh sách xóa phần tử cuối:\n', list_ab)
7
8  #Xóa phần tử tại vị trí số 2
9  del list_ab[1]
10 print('Danh sách xóa phần tử ở vị trí số 2:\n', list_ab)
11
12 #Xóa phần tử có giá trị 0 xuất hiện đầu tiên
13 list_ab.remove(0)
14 print('Xóa phần tử có giá trị 0 đầu tiên:\n', list_ab)
```

Danh sách ban đầu:

[8, 100, 4, 8, 2, 3, 0, 7, 6, 5, 10]

Danh sách xóa phần tử cuối:

[8, 100, 4, 8, 2, 3, 0, 7, 6, 5]

Danh sách xóa phần tử ở vị trí số 2:

[8, 4, 8, 2, 3, 0, 7, 6, 5]

Xóa phần tử có giá trị 0 đầu tiên:

[8, 4, 8, 2, 3, 7, 6, 5]

Các thao tác khác với list (t)

- Đếm số lần xuất hiện của một phần tử trong danh sách:

<danh_sách>.count(<giá trị>)

```
1  #Đếm số lần xuất hiện của một phần tử trong danh sách:
2  print('Danh sách ban đầu: \n', list_ab)
3
4  #Số lần xuất hiện số 8 trong danh sách
5  print(' Số 8 xuất hiện: ', list_ab.count(8))
6
7  #Số lần xuất hiện số 1 trong danh sách
8  print(' Số 1 xuất hiện: ', list_ab.count(1))
```

Danh sách ban đầu:

[8, 4, 8, 2, 3, 7, 6, 5]

Số 8 xuất hiện: 2

Số 1 xuất hiện: 0

Các thao tác khác với list (t)

- Lưu ý về việc gán danh sách bởi 1 danh sách khác

```
1 #Trường hợp số, chuỗi
2 a = 10      #Khai báo biến a có giá trị =10
3 b = a      # Gán giá trị của biến a cho biến b
4 b = 5      # thay đổi giá trị của biến b, bằng giá trị mới
5 #-----
6 print('Giá trị của biến a: ', a)
7 print('Giá trị của biến b: ', b)
```

Giá trị của biến a: 10

Giá trị của biến b: 5

```
1 #Trường hợp danh sách:
2 ds_a = [4,5,8,9] #Khai báo danh sách ds_a
3 ds_b = ds_a      #Gán giá trị của biến ds_a cho ds_b
4 ds_b[1] = 10     #Thay đổi giá trị vị trí số 2 trong ds_b
5 #-----
6 print('Biến ds_a: ', ds_a)
7 print('Biến ds_b: ', ds_b)
```

Biến ds_a: [4, 10, 8, 9]

Biến ds_b: [4, 10, 8, 9]

Với mỗi sự thay đổi ở một trong 2 danh sách quản lý bởi biến ds_a, ds_b đều dẫn đến sự thay đổi ở danh sách còn lại.

Các thao tác khác với list (t)

- Để tạo một danh sách độc lập với danh sách hiện có, các phần tử trong danh sách mới được sao chép giống hoàn toàn các phần tử trong danh sách hiện có:

<danh sách>.copy()

```
1  #Sao chép một danh sách độc lập:
2  ds_a = [4,5,8,9]    #Khởi tạo danh sách ds_a
3  ds_b = ds_a.copy()  #Sao chép ds_a cho ds_b
4  ds_b[1] = 10        #Thay đổi giá trị vị trí số 2 trong ds_b
5  #-----
6  print('Biến ds_a: ', ds_a)
7  print('Biến ds_b: ', ds_b)
```

```
Biến ds_a:  [4, 5, 8, 9]
Biến ds_b:  [4, 10, 8, 9]
```

d) Kiểu dữ liệu boolean

- Kiểu dữ liệu Boolean chỉ có hai giá trị True (đúng) và False (sai):

```
1  #Khai báo biến kiểu dữ liệu Boolean:
2  x = True
3  y = False
4  #Khai báo biến kiểu dữ liệu boolean qua biểu thức
5  z = 5 > 8
6  w = 12 == 12
7  #-----
8  print ('Kiểu dữ liệu của biến x:', type(x), ', Giá trị: ', x)
9  print ('Kiểu dữ liệu của biến y:', type(y), ', Giá trị: ', y)
10 print ('Kiểu dữ liệu của biến z:', type(z), ', Giá trị: ', z)
11 print ('Kiểu dữ liệu của biến w:', type(w), ', Giá trị: ', w)
```

```
Kiểu dữ liệu của biến x: <class 'bool'> , Giá trị: True
Kiểu dữ liệu của biến y: <class 'bool'> , Giá trị: False
Kiểu dữ liệu của biến z: <class 'bool'> , Giá trị: False
Kiểu dữ liệu của biến w: <class 'bool'> , Giá trị: True
```


Thực hành

Bài 5: Thống kê điểm Học viên

Khởi tạo một danh sách gồm các điểm thi môn “Python for Analysis” của lớp AIV_EVNICT (Điểm chữ: A, B, C, D, F)

- **Yêu cầu:**

- 1) Cho biết số sinh viên trong lớp
- 2) Có bao nhiêu sinh viên phải học lại môn này (**điểm F**).
- 3) Có bao nhiêu sinh viên có điểm từ B trở lên.
- 4) Sinh viên đầu tiên và cuối cùng trong lớp đã nghỉ học, tạo một bảng điểm mới và loại bỏ điểm của các sinh viên này ra khỏi danh sách điểm.



Bảng điểm lớp Japan01:

```
['C', 'B', 'D', 'A', 'F', 'A', 'B', 'F', 'B', 'B', 'C', 'A', 'D', 'F', 'B']
```

-----THỐNG KÊ-----

1.Tổng số học sinh trong lớp: 15

2.Số học sinh phải học lại là: 3

3.Số học sinh có điểm từ B trở lên là: 8

4.Bảng điểm mới: ['B', 'D', 'A', 'F', 'A', 'B', 'F', 'B', 'B', 'C', 'A', 'D', 'F']