

CHƯƠNG I. Python và các lệnh cơ bản

I. Tạo chương trình và tải tài nguyên

1. Tạo chương trình

Trong Pycharm, ta có thể đặt nhiều thứ trong một Project. Hoặc ta cũng có thể tạo nhiều project riêng biệt. Để tạo một project, hãy chọn File/ New project...

Sau đó đặt tên cho Project.

Nếu muốn đặt project trong một thư mục nào đó, ta chọn biểu tượng để chọn thư mục chứa project, sau đó mới đặt tên Project. Ví dụ: Tạo project có tên MyProject và đặt trong ổ đĩa D.

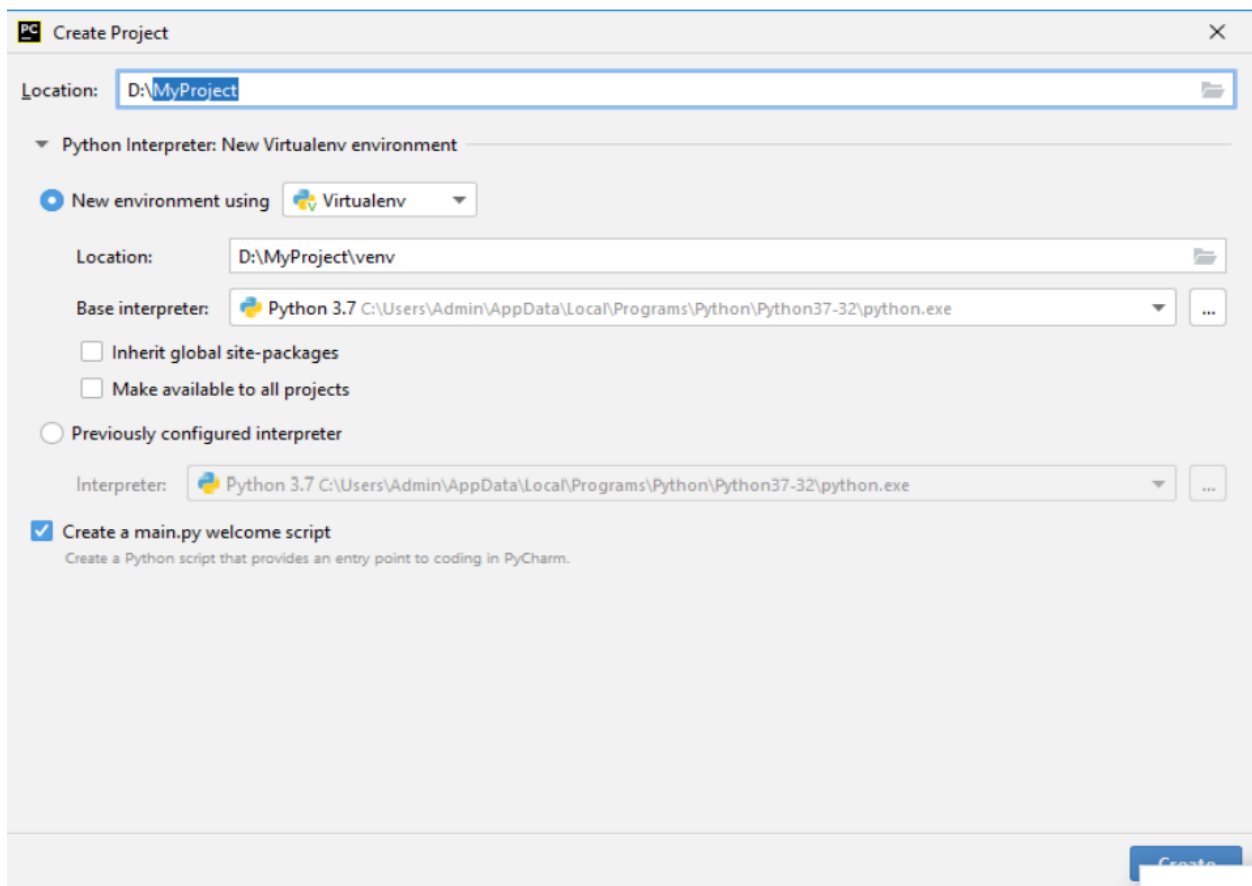


Figure 1: Tạo project mới trong PyCharm

1.1. Thêm file .py

Kích phải chuột lên tên Project tại cửa sổ quản lý Project bên tay trái, chọn New/ Python File, sau đó đặt tên file, Enter.

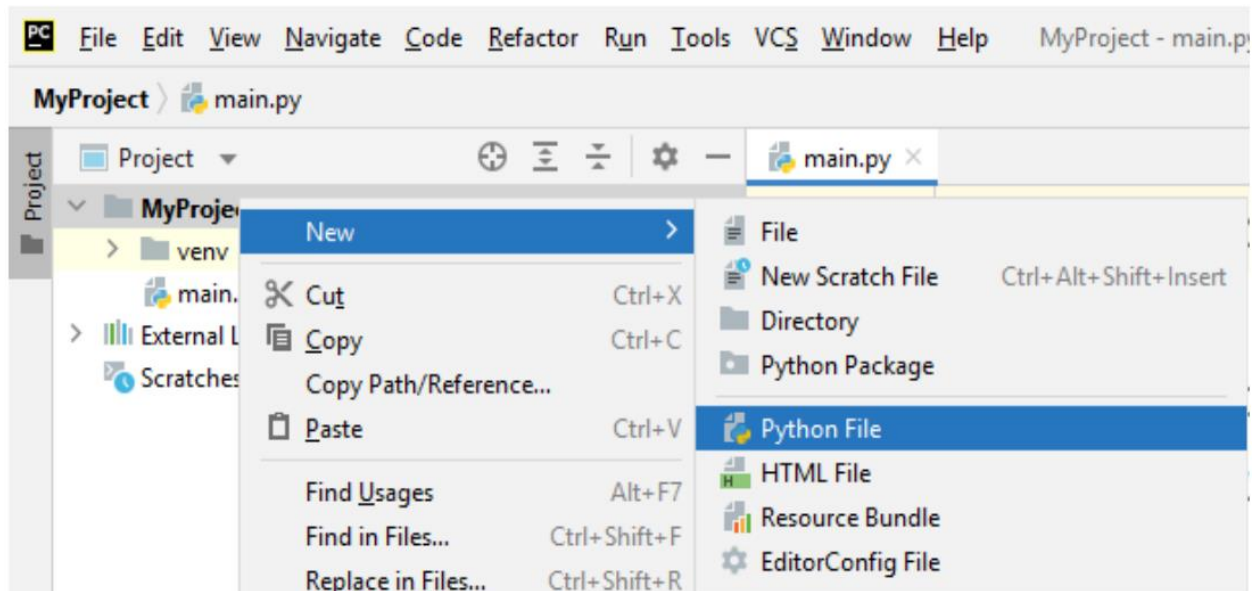



Figure 2: Thêm một file .py vào project

File .py đã sẵn sàng và ta có thể soạn thảo mã lệnh của chương trình.

1.2. Thực thi chương trình

Kích phải chuột (tại tên file .py hoặc cửa sổ soạn thảo mã lệnh) hoặc chọn biểu

tượng  bên tay phải, hoặc chọn menu Run/ Run hoặc bấm tổ hợp phím Shift + F10 để thực thi chương trình.

2. Hướng dẫn cài đặt thêm các gói trong Pycharm

Python là môi trường mở để bên thứ 3 có thể dễ dàng bổ sung các gói thư viện của mình và chúng ta có thể sử dụng nó. Khi sử dụng một thư viện, nếu nó không có sẵn trong máy tính của bạn, hãy cài đặt bổ sung.

2.1. rê chuột lên tên gói và chọn install package

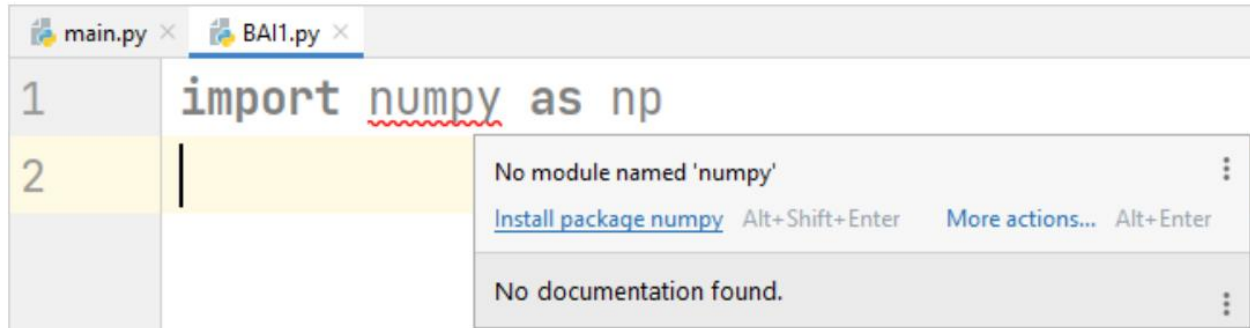


Figure 3: Cài đặt gói trực tiếp

2.2. Cài đặt trong cửa sổ quản lý gói

Chọn Python Packages (tab nằm phía dưới cùng), sau đó, tại cửa sổ tìm kiếm, ta gõ tên gói (ví dụ: “Numpy”). Nếu tìm thấy, kích chọn tên gói và chọn Install.

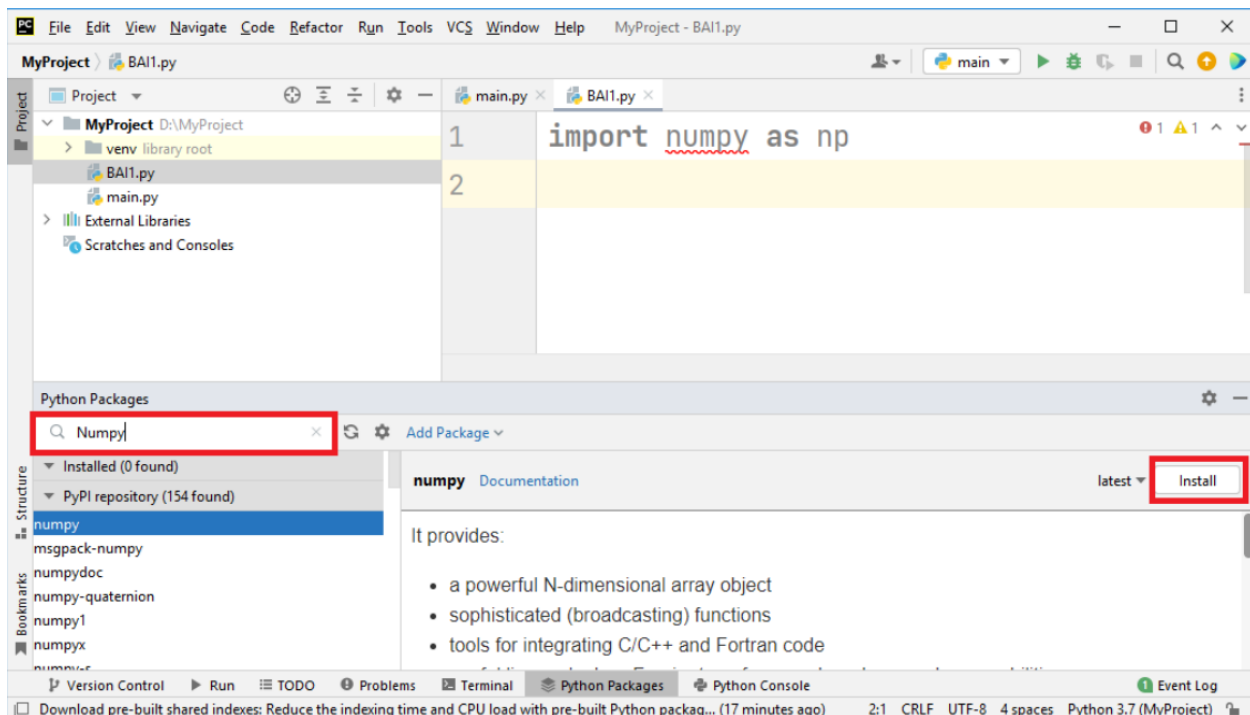


Figure 4: Cài đặt thêm gói bằng Python Packages

2.3. Cài đặt bằng pip

Chọn Terminal (tab nằm phía dưới cùng), sau đó gõ “pip install ” + Tên gói + Enter. Ví dụ: cài đặt thêm gói Numpy sẽ thao tác như Hình 5.

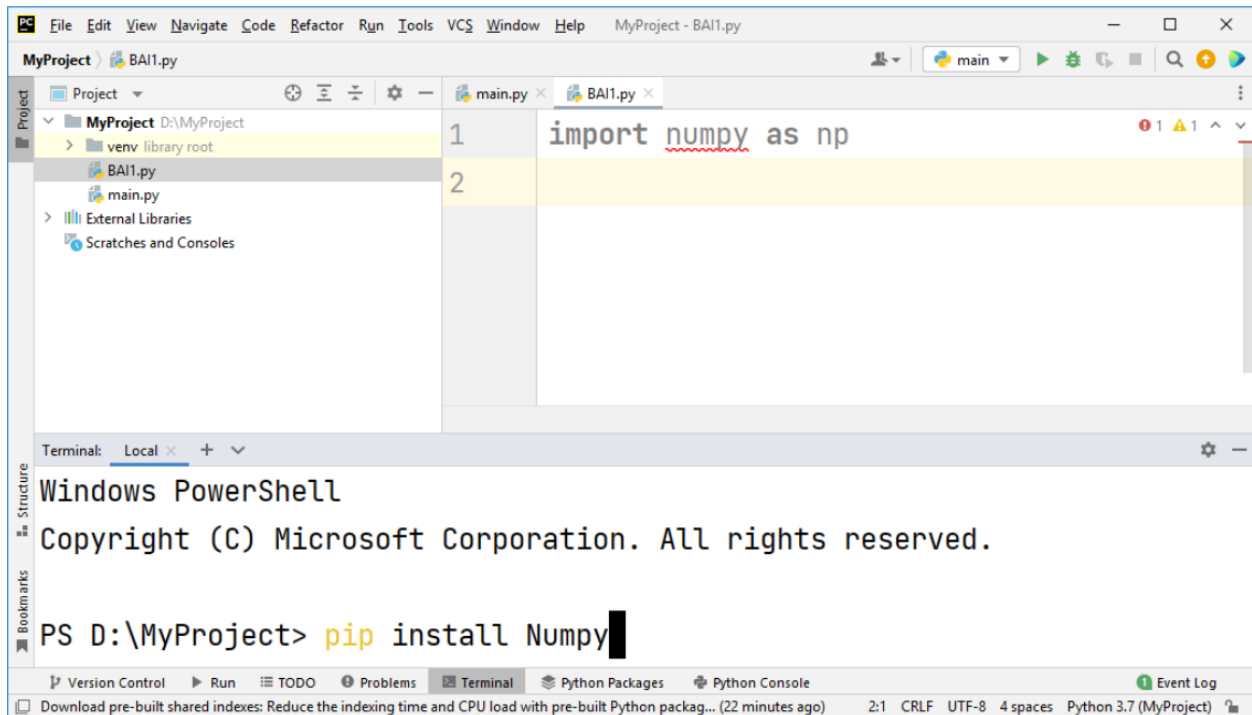




Figure 5: Cài đặt thêm gói bằng pip

2.4. Sử dụng các gói của Anaconda

Trong trường hợp máy tính đã cài sẵn Anaconda (một IDE tương tự như Pycharm) và không thể cài đặt bằng các cách trên, ta có thể sử dụng tạm thời các gói của Anaconda như sau:

- Chọn tên project ở góc dưới, bên phải màn hình, chọn Add interpreter...
- Chọn Conda Environment/ Existing Environment.
- Trong ô Interpreter, chọn biểu tượng  (Hình 6).
- Chọn C:/ ProgramData/ Anaconda 3/ Python.exe.

Lưu ý: Thư mục ProgramData là thư mục ẩn. Để hiện, ta chọn biểu tượng

trước khi chọn thư mục .

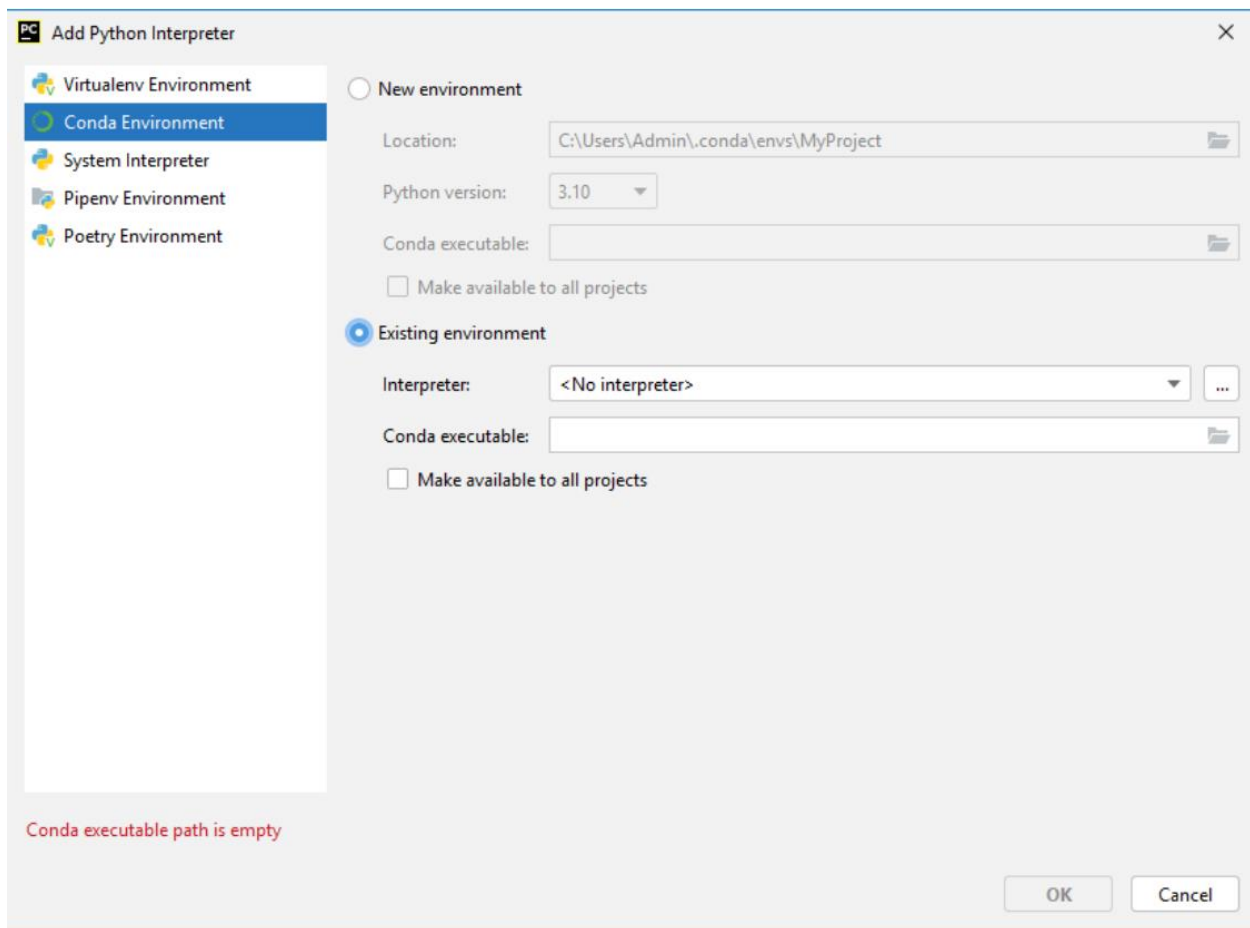


Figure 6: Chọn môi trường anacoda để thay thế

3. Tạo môi trường dùng chung

Để có thể tái sử dụng lại các thư viện đã tải từ các Project trước, chúng ta có thể tạo một môi trường dùng chung cho các Project.

Bước 1: Truy cập **File > New Projects Setup > Settings for new Projects**

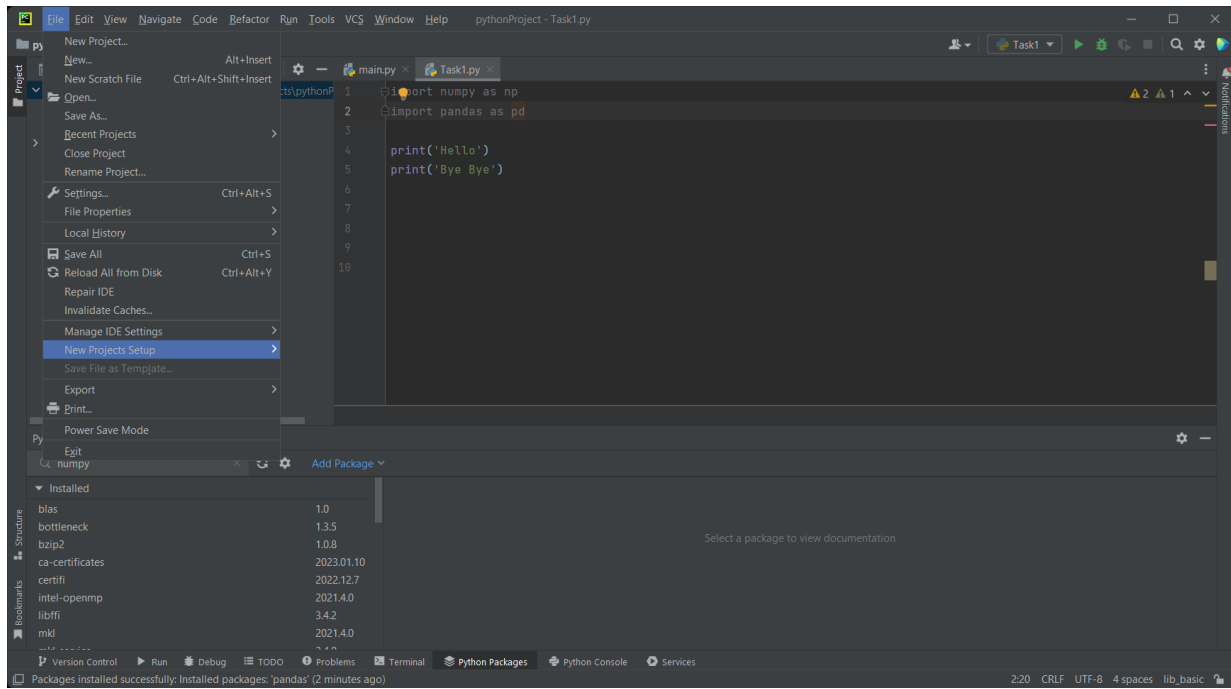


Figure 7: Tuy cập setting

Bước 2: Từ **Python Interpreter** chọn **Add Interpreter**

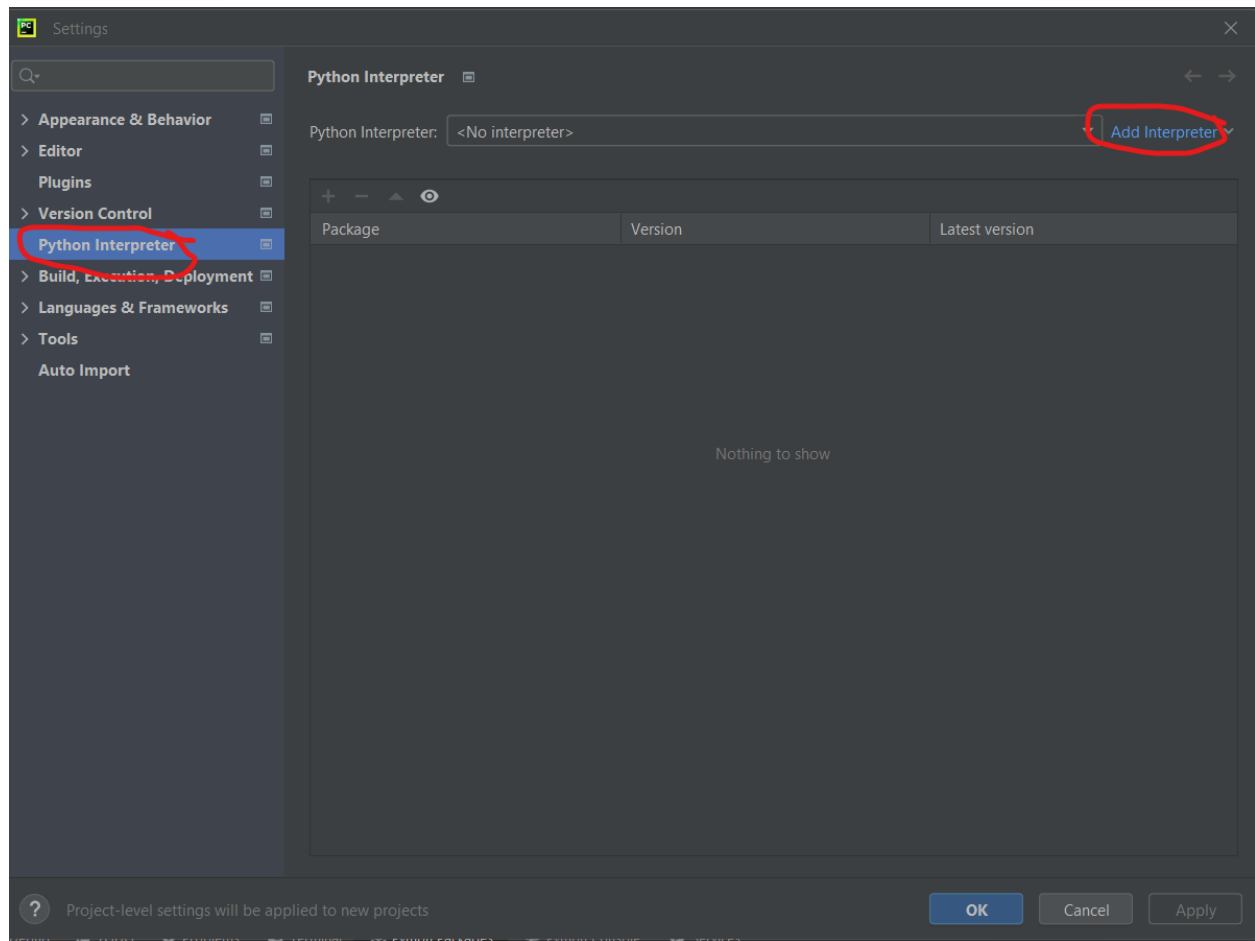


Figure 8: Tạo môi trường mới

Bước 3: Chọn nơi tạo môi trường, và đặt tên

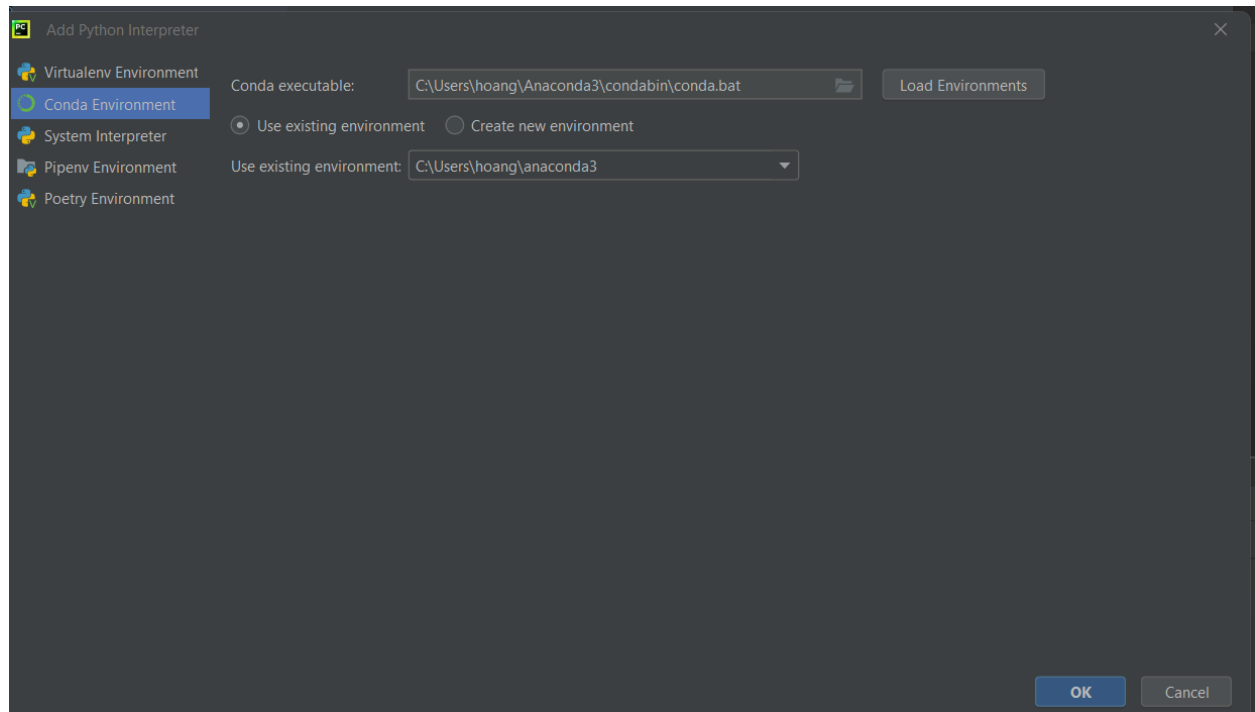


Figure 9: Đặt tên môi trường

Bước 4: Ấn biểu tượng  để cài đặt thư viện cần thiết.

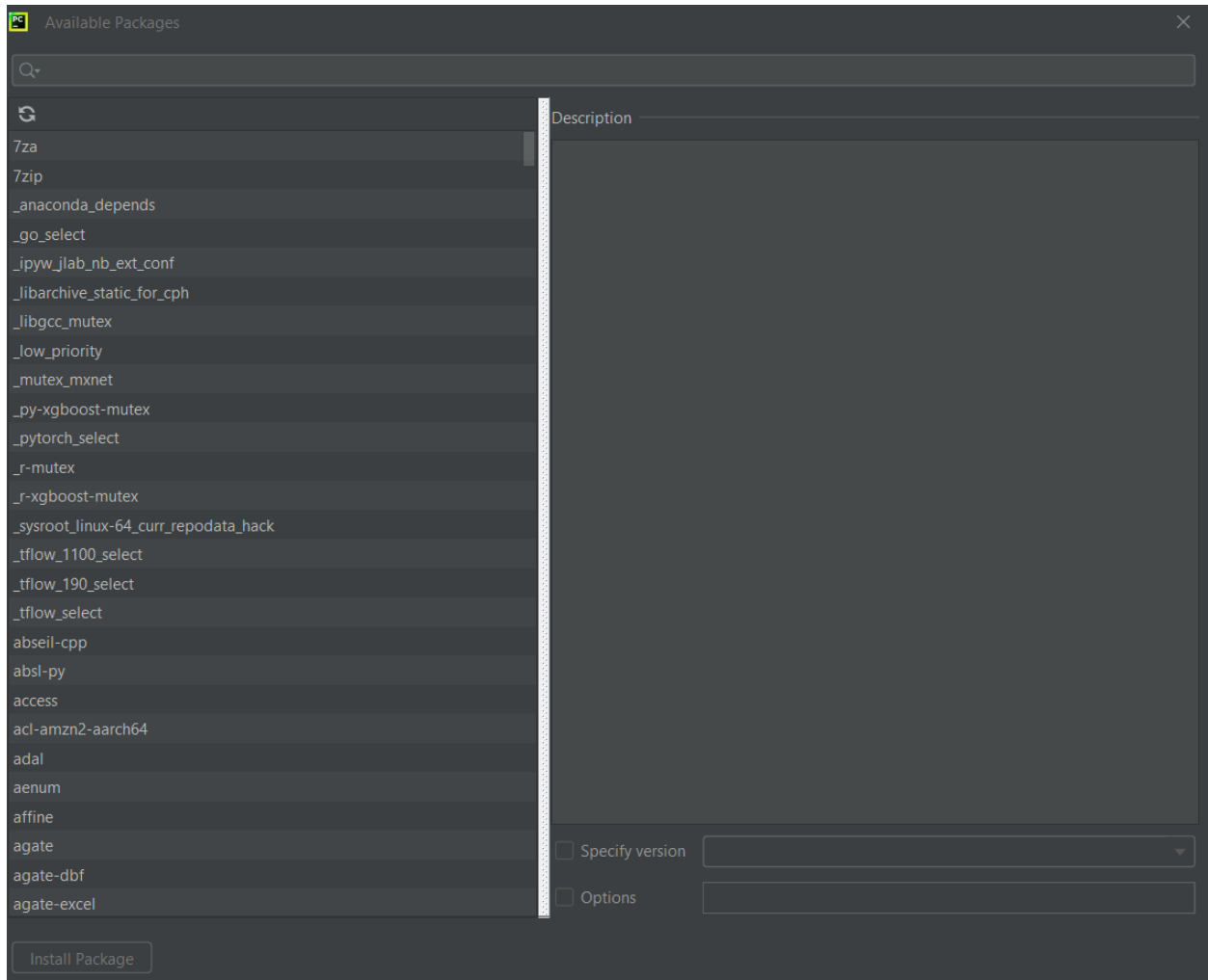
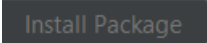


Figure 10: Cài đặt thư viện

Chọn thư viện muốn cài, rồi ấn  ở góc cuối để hoàn thành.

II. Cú pháp Python

1. Từ khóa và định danh

1.1. Từ khóa

Từ khóa là các từ dành riêng trong Python và các định danh là tên được đặt cho các biến, hàm, v.v.).

Trong Python, các từ khóa có phân biệt chữ hoa chữ thường. Có 33 từ khóa trong Python 3.7.

Tất cả các từ khóa ngoại trừ **True**, **False** và **None** đều ở dạng chữ thường và chúng phải được viết chính xác.

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Figure 11: Danh mục các từ khóa của Python

1.2. Định danh

Định danh là tên được đặt cho các thực thể như lớp, hàm, biến, v.v. Nó giúp phân biệt thực thể này với thực thể khác.

Quy tắc đặt tên trong Python như sau:

1. Tên có thể bao gồm: các chữ cái viết thường (a đến z) hoặc viết hoa (A đến Z) hoặc chữ số (0 đến 9) hoặc dấu gạch dưới _.
2. Tên không thể bắt đầu bằng một chữ số.

3. Tên không được trùng với từ khóa.
4. Tên có thể có độ dài bất kỳ.
5. Nếu tên gồm nhiều từ: Các từ có thể được phân tách bằng dấu gạch dưới, như “this_is_a_long_variable”.

2. Câu lệnh Python (Python Statement)

Câu lệnh nhiều dòng: Trong Python, phần cuối của một câu lệnh được đánh dấu bằng một ký tự dòng mới (không sử dụng dấu chấm phân “;”). Nhưng chúng ta có thể tạo một câu lệnh kéo dài qua nhiều dòng với ký tự tiếp tục dòng (\).

Ví dụ:

```
a = 1 + 2 + 3 + \  
    4 + 5 + 6 + \  
    7 + 8 + 9
```

Tuy nhiên, ta cũng có thể sử dụng các dấu ngoặc như: (), [], {} để kéo dài câu lệnh trên nhiều dòng, trong một số trường hợp, ví dụ:

```
a = (1 + 2 + 3 +  
    4 + 5 + 6 +  
    7 + 8 + 9)
```

Nhiều câu lệnh trên một dòng: Chúng ta cũng có thể đặt nhiều câu lệnh trong một dòng bằng cách sử dụng dấu chấm phẩy, như sau:

```
a = 1; b = 2; c = 3
```

3. Tab đầu dòng trong Python (Python Indentation)

Hầu hết các ngôn ngữ lập trình như C, C++ và Java sử dụng dấu ngoặc nhọn {} để xác định một khối lệnh. Python, tuy nhiên, sử dụng tab đầu dòng (thụt lề).

```
if True:
    print('Hello')
    a = 5
```

4. Comment trong Python (chú thích)

Trong Python, chúng ta sử dụng ký hiệu thăng (#) để bắt đầu viết dòng chú thích. Các chú thích là để người lập trình hiểu rõ hơn về một chương trình. Trình thông dịch Python bỏ qua các dòng chú thích này.

Ví dụ:

```
#This is a comment
#print out Hello
print('Hello')
```

Chúng ta có thể có những nhận xét kéo dài đến nhiều dòng. Một cách đơn giản là sử dụng ký hiệu thăng (#) ở đầu mỗi dòng. Tuy nhiên, ta có thể bắt đầu và kết thúc khối chú thích nhiều dòng bằng ba dấu nháy đơn hoặc nháy kép: ''' hoặc """.

Ví dụ:

```
"""This is also a
perfect example of
multi-line comments"""
```

III. Biến và biểu thức

1. Biến

Cũng như các ngôn ngữ lập trình khác, biến trong Python là các ô nhớ được đặt tên, dùng để lưu trữ các giá trị sẽ sử dụng trong chương trình.

Python là một ngôn ngữ mà có thể tự suy ra kiểu của biến, vì vậy bạn không cần phải xác định rõ ràng kiểu biến. Tùy theo kiểu của dữ liệu được gán vào biến mà Python tự xác định kiểu cho biến.

Ví dụ:

```
website = "apple.com"
```

Khi đó, Python tự động biết rằng "apple.com" là một chuỗi và khai báo biến website là kiểu chuỗi ký tự.

2. Gán giá trị cho biến

Trong Python, ta sử dụng toán tử gán '=' để gán các giá trị cho biến. Khởi gán một giá trị cho nhiều biến: Ta có thể khởi gán liên tiếp cùng một giá trị cho nhiều biến bằng cách viết như trong ví dụ sau (biến x, y, z được gán cùng một giá trị 'Hello'):

```
x = y = z = 'Hello'
```

Khởi gán nhiều giá trị cho nhiều biến: Để khởi gán nhiều giá trị cho nhiều biến, ta có cách viết nhanh như sau (biến x sẽ được khởi gán là 1, biến y được gán giá trị là 2.5, biến z được gán giá trị là 'Hello'):

```
x, y, z = 1, 2.5, 'Hello'
```

Lưu ý: Trong Python, chúng ta không thực sự gán giá trị cho các biến. Thay vào đó, Python cung cấp tham chiếu của đối tượng (giá trị) cho biến.

3. Hằng trong Python

Hằng là một loại biến có giá trị không thể thay đổi. Sẽ rất hữu ích khi nghĩ về hằng số như là các thùng chứa thông tin mà sau này không thể thay đổi được.

Gán giá trị cho hằng: Trong Python, hằng thường được khai báo và gán giá trị cùng lúc, trong một mô-đun. Ở đây, mô-đun là một tệp mới riêng biệt được sử dụng trong tệp chính. Bên trong mô-đun, các hằng được định nghĩa với tên hằng được viết hoa toàn bộ. Ví dụ: trong tệp **constants.py**, ta khai báo:

```
PI = 3.14
GRAVITY = 9.8
```

Khi đó, trong tệp chính main.py, ta có thể sử dụng các hằng này:

```
import constant

print(constant.PI)
print(constant.GRAVITY)
```

4. Kiểu dữ liệu của biến

Mặc dù không cần chỉ rõ kiểu dữ liệu của biến khi ta khai báo biến, nhưng mọi giá trị trong Python đều có kiểu dữ liệu. Vì mọi thứ đều là một đối tượng trong lập trình Python. Các kiểu dữ liệu là các lớp và các biến là thể hiện (đối tượng) của các lớp này.

Có nhiều kiểu dữ liệu khác nhau trong Python. Một số loại quan trọng được liệt kê dưới đây:

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

Kiểu có cấu trúc (cấu trúc dữ liệu): list, tuple, set, dictionary, string (các cấu trúc này sẽ được giới thiệu ở bài sau).

5. Lấy kiểu dữ liệu của một biến:

```
type (<Tên_biến>)
```

Ví dụ:

```
a = 10  
print (type (a) )
```

Kết quả ta thu được:

```
<class 'int'>
```

Kiểm tra một biến có thuộc một kiểu nào đó hay không

```
isinstance (<Tên_biến>, <Kiểu>)
```

Hàm này sẽ trả về True nếu biến có kiểu như chỉ định, ngược lại, hàm trả về False. Ví dụ:

```
a = 10  
print (isinstance (a, int) )
```

Kết quả thu được:

```
True
```

Chúng ta có thể chuyển đổi giữa các kiểu dữ liệu khác nhau bằng cách sử dụng các hàm chuyển đổi kiểu khác nhau như: `int ()`, `float ()`, `str ()`, v.v. Ví dụ:

```
a = 10  
print (type (a) )  
b = float (a)  
print (type (b) )
```

Kết quả ta thu được:

```
<class 'int'>
<class 'float'>
```

6. Biểu thức:

Để tính toán một đại lượng, ta sử dụng các biểu thức. Biểu thức trong Python cũng bao gồm hai thành phần là: Các toán tử và các toán hạng.

Một số toán tử:

- o Arithmetic operators (toán tử số học): +, -, *, /, %, **, //
- o Assignment operators (toán tử gán): =, +=, -=, *=,
- o Comparison operators (toán tử so sánh): ==, !=, >, <, >=, <=
- o Logical operators (toán tử logic): and, or, not

Toán hạng:

- o Hằng: Gồm nhiều loại như hằng số, hằng chuỗi ký tự
- o Biến: Ta có thể sử dụng các biến trong biểu thức, giá trị của biến sẽ được sử dụng để tính toán giá trị của biểu thức.
- o Hàm: Các hàm trả về một giá trị cũng có thể được dùng tương tự một biến trong biểu thức.

Ví dụ:

```
import math
a = 10
print(a + 2*math.sqrt(a))
```

Kết quả thu được:

16.32

IV. Nhập xuất dữ liệu

Một số hàm như `input ()` và `print ()` được sử dụng rộng rãi cho các hoạt động đầu vào và đầu ra tiêu chuẩn tương ứng.

1. Xuất dữ liệu

Xuất dữ liệu: Sử dụng hàm `print()`

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

- `*objects`: là các thành phần dữ liệu cần xuất.
- `sep`: là ký tự ngăn cách giữa hai dữ liệu, mặc định là dấu cách.
- `end`: Sau khi tất cả dữ liệu đã được in ra thiết bị xuất, ký tự được chỉ định trong `end` sẽ được in ra. Thông thường nó có thể là dấu xuống dòng `'\n'`.
- `file`: Chỉ định thiết bị xuất (màn hình, file,...). Mặc định là `sys.stdout` (màn hình).

Ví dụ:

```
print(1, 2, 3, 4, sep='#', end='&')
```

Kết quả thu được:

```
1#2#3#4&.
```

2. Định dạng xuất

Sử dụng phương thức `str.format()`

Ta có thể dùng ký hiệu giữ chỗ `{}` để xuất dữ liệu từ các biến, sử dụng phương thức `format()` như sau:

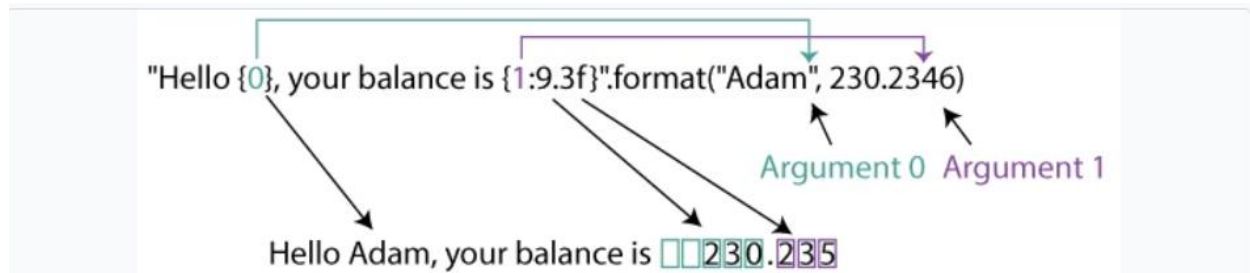
```
a = 5
print('a = {}'.format(a))
```

Kết quả thu được:

```
a = 5
```

2.1. Position parameter

- Lệnh format đọc lần lượt các giá trị truyền vào theo position
- Ví dụ:



Ví dụ trên có 2 position là 0 và 1; 2 giá trị truyền vào được format theo các position tương ứng

Khi không viết giá trị position sẽ ngầm hiểu bắt đầu từ 0

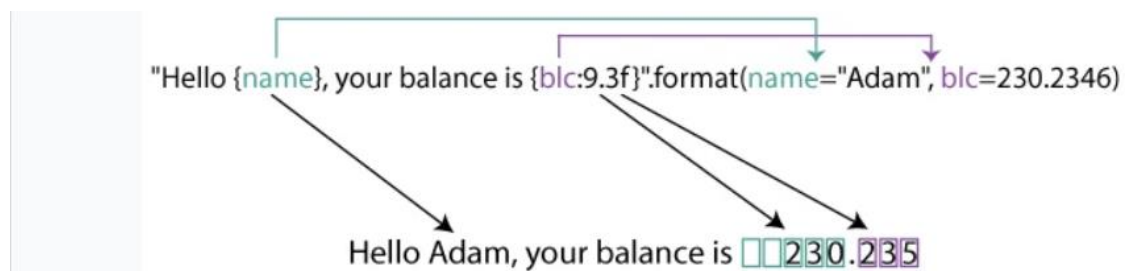
Position do người dùng tự viết, có thể viết không theo thứ tự

```
1 a = 5
2 print('a = {2}\nb = {1}\nc = {0}'.format(a, 1, 2.32443))
```

```
a = 2.32443
b = 1
c = 5
```

2.2. Keyword parameter

Tương tự position parameter, nhưng thay vì dùng position, hàm print sẽ format theo keyword.



Có thể kết hợp position và keyword trong cùng 1 lệnh print.

2.3. 1 số đặc tả

Đặc tả cho số

d	Kiểu số nguyên
c	Unicode
b	Nhi phân
o	Hệ 8
x	Hệ 16(chữ thường)
X	Hệ 16(chữ hoa)
e	Ký hiệu mũ(chữ thường)
E	Ký hiệu mũ(chữ hoa)
f	Số thực(mặc định 6)
g	Định dạng chung (6 chữ số có nghĩa)
%	Định dạng dạng phần trăm

Đặc tả căn chỉnh

<	Căn trái
>	Căn phải
^	Căn giữa
=	Đẩy dấu của số ra ngoài cùng trái

3. Nhập dữ liệu

Trong Python, chúng ta có hàm `input()` để cho phép điều này. Cú pháp cho `input ()` là:

```
input([Câu thông báo])
```

Ví dụ:

```
a = input()
```

Hoặc:

```
a = input('Nhập một giá trị nguyên a:')
```

Nhập nhiều biến bằng một lệnh input:

Ta có thể nhập giá trị cho nhiều biến chỉ bằng 1 lệnh input. Khi đó, mỗi giá trị nhập vào được ngăn cách bởi một dấu cách, và ta sử dụng phương thức `split()` để tách chúng ra và gán vào từng biến.

```
a, b = input().split()
print('a =', a)
print('b =', b)
```

Kết quả ta thu được: Chương trình cho phép nhập hai giá trị bất kỳ, ngăn cách bởi dấu cách. Giả sử ta nhập '3 5Enter'. Ta thu được:

```
a = 3
```

```
b = 5
```

V. Cấu trúc điều khiển, lặp

1. Cấu trúc rẽ nhánh

Việc ra quyết định là bắt buộc khi chúng ta chỉ muốn thực thi một đoạn code khi một điều kiện nhất định được thỏa mãn. Câu lệnh `if... elif... else` được sử dụng trong trường hợp này.

- Cấu trúc `if`

```
if <Điều_Kiện>:  
    Lệnh
```

- Cấu trúc `if ... else:`

```
if <Điều_Kiện>:  
    Lệnh 1  
else:  
    Lệnh 2
```

- Cấu trúc `if...elif...else:`

```
if <Điều_Kiện>:  
    Lệnh 1  
elif:  
    Lệnh 3  
else:  
    Lệnh 2
```

2. Cấu trúc lặp

2.1. Lặp *for*

Vòng lặp `for` trong Python được sử dụng để lặp qua một chuỗi (ví dụ như danh sách, tuple, chuỗi ký tự) hoặc các đối tượng có thể lặp khác. Lặp theo một trình tự được gọi là duyệt.

Cú pháp:

```
for <Biến> in <Dãy>:  
    Lệnh_Lặp
```

Ví dụ:

```
for x in (1, 2, 5):  
    print(x)
```

Kết quả thu được:

```
1  
2  
5
```

Trong đó:

<Biến> là một tên do ta tùy ý đặt. Biến sẽ lần lượt nhận từng giá trị trong <Dãy> và thực hiện Lệnh_lặp.

Lệnh_lặp: có thể là 1 lệnh, một khối lệnh, một cấu trúc điều khiển hoặc một khối cấu trúc điều khiển.

2.2. Hàm range()

Hàm range() thường được dùng kèm với lệnh for khi nó cho phép ta khởi tạo một dãy với nhiều dạng khác nhau: range(n) Khởi tạo dãy với n phần tử nguyên liên tiếp, từ 0 tới n-1.

Ví dụ:

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

Kết quả thu được:

0
1
2
3
4

`range(start, stop, step)`

Khởi tạo một dãy các số nguyên từ start cho tới stop-1, nhưng với bước nhảy step.

2.3. Lặp while

Vòng lặp while trong Python được sử dụng để lặp qua một lệnh/ một khối lệnh miễn là biểu thức điều kiện là đúng. Chúng ta thường sử dụng vòng lặp này khi chúng ta không biết trước số lần lặp lại.

Cú pháp:

```
while <điều_kiện_lặp>:  
    <Lệnh_lặp>
```

Vòng lặp while cũng có thể kết hợp với else:. Chừng nào mà điều kiện lặp nhận giá trị true, nó thực hiện <Lệnh_lặp>. Ngược lại, khi điều kiện lặp nhận giá trị False, nó thực hiện >Lệnh> sau else:

```
while <điều_kiện_lặp>:  
    <Lệnh_lặp>  
else:  
    <Lệnh>
```

VI. Làm tròn trong Python

1. round()

- Là hàm được xây dựng sẵn trong python
- Làm tròn số tiến về 0

2. Math.ceil()

- Làm tròn lên
- Là hàm nằm trong gói math
- Làm tròn số về dương vô cùng

3. Math.floor()

- Làm tròn xuống
- Là hàm nằm trong gói math
- Làm tròn số về âm vô cùng

VII. Bài tập vận dụng

Bài 1: Nhập vào từ bàn phím hai số nguyên a, b. Tính và in ra màn hình tổng, hiệu, tích, thương của a và b với mỗi kết quả in trên 1 dòng; kết quả của thương là số thực có độ chính xác 3 chữ số hàng thập phân.

Bài 2: Nhập vào tọa độ của hai điểm A(x1, y1) và B(x2, y2). Tính và in ra khoảng cách Euclidean giữa A và B theo công thức:

$$d(A, B) = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}.$$

Bài 3: Giải phương trình bậc 2 với các hệ số a, b, c nhập từ bàn phím.

Bài 4: Nhập vào một số nguyên n cho tới khi n thuộc [20, 30], nhập vào số thực x. Tính và in ra:

$$P = 2022 x^n + \frac{1}{x} + \frac{2}{x^2} + \dots + \frac{n}{x^n}$$

Bài 5: Nhập vào một số nguyên dương n. kiểm tra xem n có phải là số nguyên tố hay không?

Bài 6: Nhập vào một số nguyên n thuộc [100, 200]. Nếu dữ liệu nhập vào không thỏa mãn điều kiện đó, cho phép nhập lại cho tới khi thỏa mãn.

Tìm một số nguyên dương là lũy thừa của 2, bé nhất, mà lớn hơn n