Function, Module, Package

Nội Dung

- Function
- Module
- Package

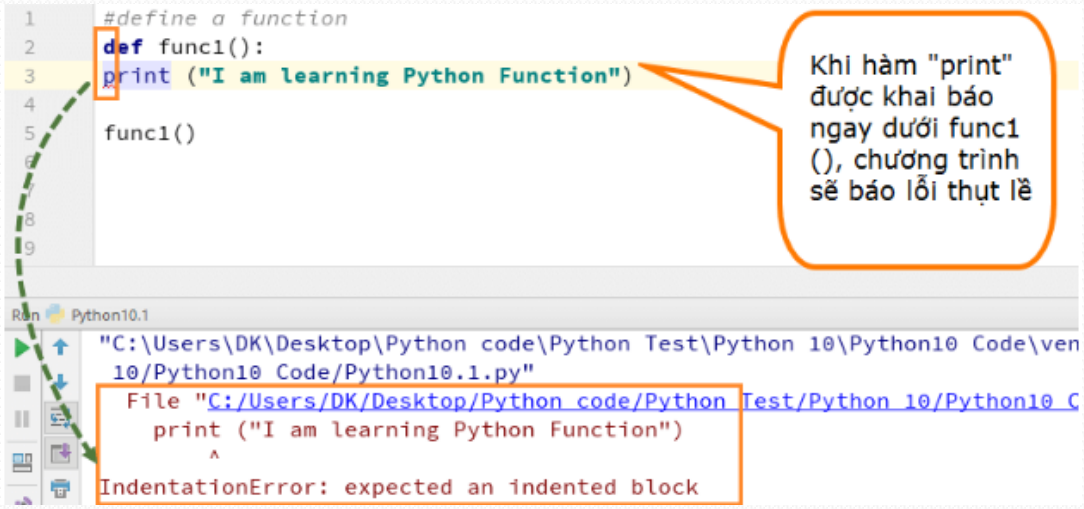
Function Trong Python

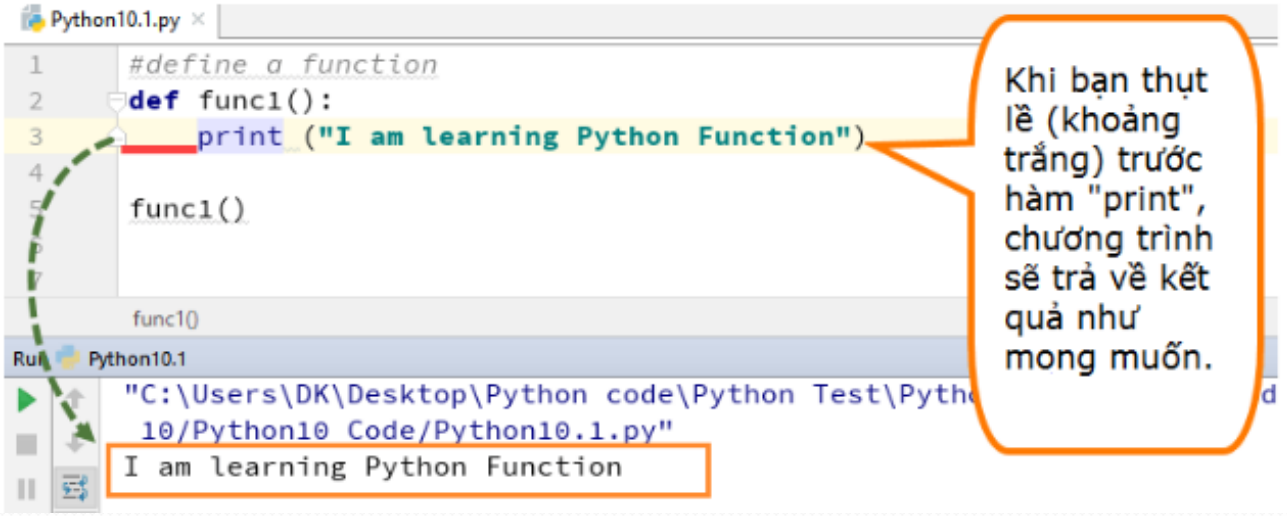
- Hàm là một khối các câu lệnh chỉ thực hiện khi được gọi
- Trong Python, hàm được định nghĩa bằng từ khóa **def** theo sau là **tên hàm** và dấu ngoặc đơn **( ):**

```
def my_function():
  print("Hello from a function")
```

Function Trong Python

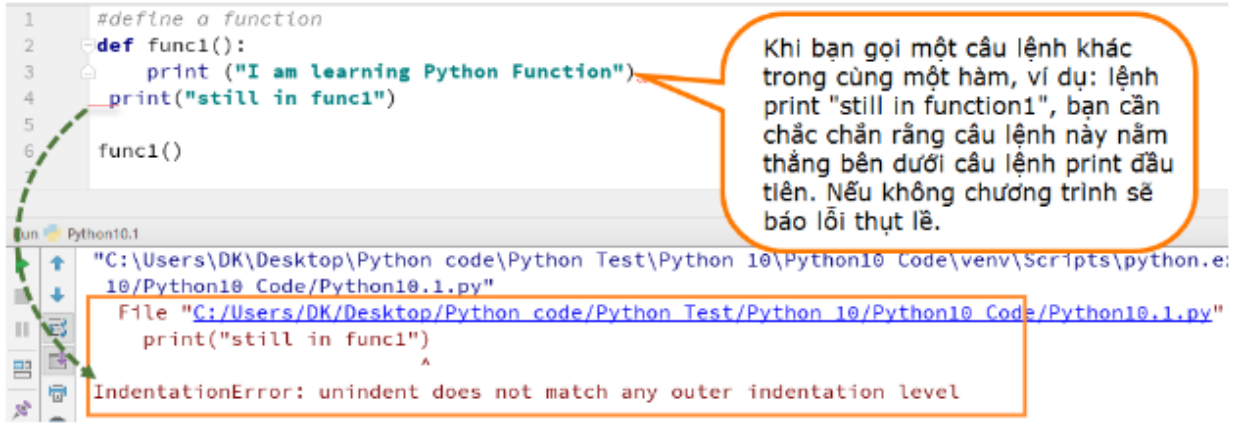
- Ý nghĩa canh lề trong hàm

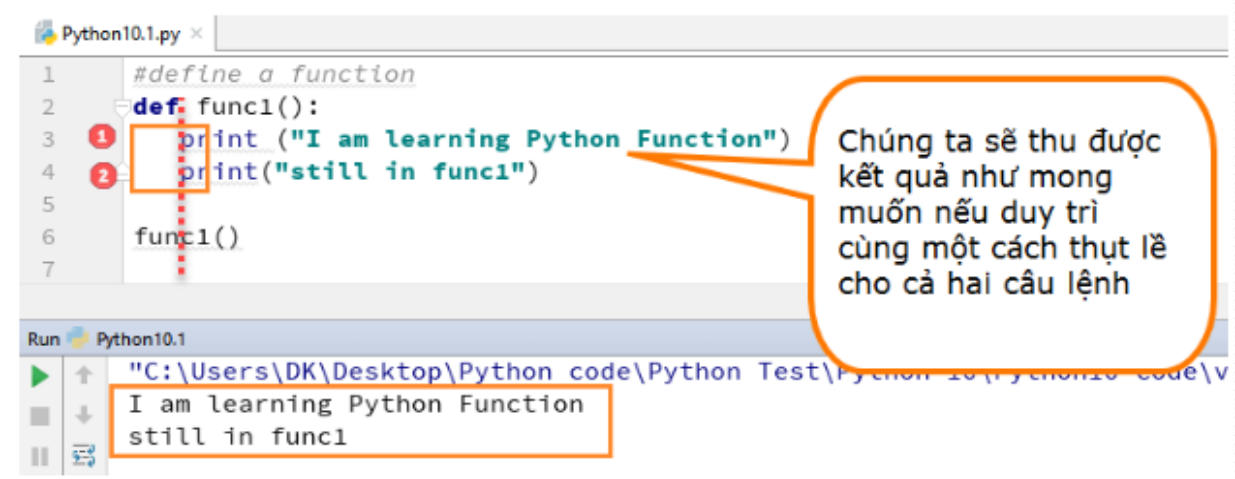




Function Trong Python

- Ý nghĩa canh lề trong hàm





Function Trong Python

- Gọi một hàm: Để gọi một hàm, chúng ta sử dụng tên hàm, theo sau là cặp dấu ngoặc đơn ():

```
def my_function():
  print("Hello, World!")
my_function()
# In ra màn hình: Hello, World!
```

Function Trong Python

- Tham số:
  - Các tham số có thể thêm sau tên hàm, bên trong cặp dấu ngoặc đơn ().
  - Chúng ta có thể thêm bao nhiêu tham số tùy thích, chỉ cần tách chúng bằng dấu phẩy.

```
def my_function (name):
  print ("Tên: " + name)
my_function ("Emil")
my_function ("Tobias")
my_function ("Linus")
""" In ra màn hình:
Tên: Emil
Tên: Tobias
Tên: Linus"""
```

Function Trong Python

- Tham số mặc định
  - Tham số mặc định là tham số đã có sẵn một giá trị trước khi hàm được gọi.
  - Nếu chúng ta gọi một hàm mà không truyền vào tham số, hàm sẽ sử dụng giá trị mặc định.
  - Ở trường hợp còn lại, tham số mặc định sẽ hoạt động như tham số thông thường.

```
def my_function(country = "Việt Nam"):
  print ("Tôi đến từ " + country)
my_function("Lào")
my_function("Campuchia")
my_function()
""" In ra màn hình:
Tôi đến từ Lào
Tôi đến từ Campuchia
Tôi đến từ Việt Nam"""
```

Function Trong Python

- Hàm có kiểu trả về
  - Để cho một hàm có thể trả về, chúng ta cần phải sử dụng từ khóa **return**:

```
def my_function (x):
  return 5 * x

print (my_function(3))
print (my_function(5))
print (my_function(9))
""" In ra màn hình:
15
25
45"""
```

Function Trong Python

- Trong phiên bản Python 3.6 có 68 hàm Python được tích hợp sẵn.

Hàm	Mô tả
abs()	Trả về giá trị tuyệt đối của một số
all()	Trả về True khi tất cả các phần tử trong iterable là đúng
any()	Kiểm tra bất kỳ phần tử nào của iterable là True
ascii()	Tả về string chứa đại diện (representation) có thể in
bin()	Chuyển đổi số nguyên sang chuỗi nhị phân
bool()	Chuyển một giá trị sang Boolean
bytearray()	Trả về mảng kích thước byte được cấp
bytes()	Trả về đối tượng byte không đổi
callable()	Kiểm tra xem đối tượng có thể gọi hay không
chr()	Trả về một ký tự (một chuỗi) từ Integer
classmethod()	Trả về một class method cho hàm
compile()	Trả về đối tượng code Python
complex()	Tạo một số phức
delattr()	Xóa thuộc tính khỏi đối tượng
dict()	Tạo Dictionary

Function Trong Python

- Hàm main() trong python: if __name __ == "__main__"

Function Trong Python

- Hàm Lambda trong Python:
  - Trong Python, **hàm vô danh** là hàm được định nghĩa mà không có tên.
  - Nếu các hàm bình thường được định nghĩa bằng cách sử dụng từ khóa def, thì **hàm vô danh** được định nghĩa bằng cách sử dụng từ khóa **lambda**
- Một hàm Lambda trong Python có cú pháp sau:

```
lambda tham_so: bieu_thuc
```

```
myfunc = lambda i: i*2
print(myfunc(2))
# In ra màn hình: 4
```

Function Trong Python

- Hàm vô danh có thể được định nghĩa với nhiều hơn một tham số đầu vào:

```
myfunc = lambda x,y: x*y
print(myfunc(3,6))
# In ra màn hình: 18
```

Function Trong Python

- Sức mạnh của hàm vô danh được thể hiện khi chúng ta tạo các hàm ẩn trong thời gian chạy của chương trình:

```
def myfunc(n):
  return lambda i: i*n

doubler = myfunc(2)
tripler = myfunc(3)
val = 11
print("Doubled: " + str(doubler(val)) + ". Tripled: " + str(tripler(val)))
# In ra màn hình: Doubled: 22. Tripled: 33
```

Function Trong Python

- Ví dụ dùng hàm lambda với filter():
  - Hàm filter() sẽ lấy các tham số là **một hàm** và **một list**.
  - Hàm được gọi với tất cả các mục trong list và list mới sẽ được trả về, chứa các mục mà hàm đánh giá là True.

```
list_goc = [10, 9, 8, 7, 6, 1, 2, 3, 4, 5]
list_moi = list(filter(lambda a: (a%2 == 0) , list_goc))
# Kết quả: [10, 8, 6, 2, 4]
print(list_moi)
```

Function Trong Python

- Ví dụ dùng hàm Lambda với map():
  - Hàm map() cũng lấy các tham số là một hàm và một list.
  - Hàm được gọi với tất cả các mục trong list và list mới được trả về chứa các mục được hàm trả về tương ứng cho mỗi mục.

```
list_goc = [10, 9, 8, 7, 6, 1, 2, 3, 4, 5]
list_moi = list(map(lambda a: a*2 , list_goc))
# Kết quả: [20, 18, 16, 14, 12, 2, 4, 6, 8, 10]

print(list_moi)
```

Module Trong Python

- Module đề cập đến một file (.py) chứa những câu lệnh Python, các hàm và các định nghĩa.

- Một file chứa code Python, ví dụ **myfunction.py** được gọi là module và tên của module sẽ là **myfunction**.

myfunction.py

```
1 # -*- coding: utf-8 -*-
2
3 def my_add_function(x,y):
4     return x+y
5
6 def my_mul_function(x,y):
7     return x*y
```

Module

- Module thường được sử dụng khi muốn chia chương trình lớn thành những file nhỏ hơn để dễ quản lý và tổ chức.

- Module cho phép tái sử dụng code

- **Làm sao để nhập module trong Python?**
  - Chúng ta có thể nhập các định nghĩa từ module này vào module khác hoặc vào trình thông dịch trong Python.
  - Chúng ta sử dụng từ khóa **import** để thực hiện việc này

Module

- Ví dụ: tạo file **testfunction.py** (lưu cùng folder với module **myfuction**) để sử dụng module myfuction

myfunction.py

```
1 # -*- coding: utf-8 -*-
2
3 def my_add_function(x,y):
4     return x+y
5
6 def my_mul_function(x,y):
7     return x*y
```

myfunction.py | testfunction.py*

```
1 # -*- coding: utf-8 -*-
2 import myfunction
3 m = myfunction.my_add_function(1,2)
4 print(m) # 3
5 n = myfunction.my_mul_function(3,2)
6 print(n) # 6
```

Module

- Sử dụng lệnh import:
- Có thể gọi nhiều module

```
import  module1, module2,...
```

hoặc

```
import  module1
import  module2
Import  module3
```

Module

- Ví dụ: module **math** tích hợp sẵn trong Python

```
import math
a = 3.2
# làm tròn lên 1 số
print(math.ceil(a)) # 4
# làm tròn xuống 1 số
print(math.floor(a)) # 3
```

Module

- **Đặt lại tên cho module**: có thể tạo bí danh khi bạn nhập module bằng cách sử dụng từ khóa **as** :

```
import math as m
a = 3.2
# làm tròn lên 1 số
print(m.ceil(a)) # 4
# làm tròn xuống 1 số
print(m.floor(a)) # 3
```

Module

- **Lệnh from...import trong Python:** Bạn có thể chọn chỉ nhập các phần từ module (một vài hàm hoặc biến, ...) thay vì dùng tất cả thành phần trong module.

from modules import something, something2,...

```
from math import pi,e
print(pi) # 3.141592653589793
print(e) # 2.718281828459045
```

Module

import mỗi phương thức ceil ở trong module math.

```
from math import ceil
a = 3.2
print(ceil(a)) # kết quả: 4
print(floor(a)) # Kết quả: name 'floor' is not defined
```

Module

import tất cả mọi thứ được cho phép từ module math

```
from math import *
a = 3.2
print(ceil(a)) # kết quả: 4
print(floor(a)) # Kết quả: 3
```

• Đối với trường hợp các bạn sử dụng from ... import * thì mặc định python nó sẽ **không import** được các đối tượng có tên được **bắt đầu bằng ký tự _** .

• Trong trường hợp này, nếu như bạn muốn import được các đối tượng đó thì bạn sẽ phải chỉ đích danh các đối tượng đó.

Module

```python
# file mathplus.py
def _get_sum (a, b):
    return a + b
```

```python
from mathplus import _get_sum

print(_get_sum(5,7))
# kết quả: 12
```

```python
from mathplus import *

print(_get_sum(5,7))
# kết quả: name '_get_sum' is not defined
```

Module

- **Sử dụng hàm dir():**
  - Có chức năng liệt kê tất cả các tên hàm (hoặc tên biến) trong một module.

```
import math
x = dir(math)
print(x)
```

```
['__doc__', '__loader__', '__name__', '__package__',
'__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan',
'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh',
'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs',
'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma',
'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf',
'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p',
'log2', 'modf', 'nan', 'pi', 'pow', 'radians',
'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh',
'tau', 'trunc']
```

Module

❖ **Đường dẫn tìm kiếm module Python**

- Khi nhập module, Python sẽ tìm một vài nơi. Trình thông dịch tìm các module có sẵn, nếu không thấy nó sẽ vào danh sách các thư mục được định nghĩa trong sys.path. Thứ tự tìm kiếm sẽ là:
  - Thư mục hiện tại.
  - PYTHONPATH (một biến môi trường với danh sách thư mục).
  - Thư mục mặc định có vị trí phụ thuộc vào chọn lựa trong quá trình cài đặt.

```
import sys
print(sys.path)
```

Module

- Ví dụ: nếu module myfunction và file testfunction được lưu cùng một folder (C:\Users\sony\Desktop\python_co_ban\test\mymodule) thì import ko lỗi

myfunction.py

```
1 # -*- coding: utf-8 -*-
2
3 def my_add_function(x,y):
4     return x+y
5
6 def my_mul_function(x,y):
7     return x*y
```

myfunction.py | testfunction.py*

```
1 # -*- coding: utf-8 -*-
2 import myfunction
3 m = myfunction.my_add_function(1,2)
4 print(m) # 3
5 n = myfunction.my_mul_function(3,2)
6 print(n) # 6
```

- Ví dụ 2: Tạo một module **myfunction2.py** và lưu tại ổ **F:\myfunction_2**

```python
# -*- coding: utf-8 -*-

def my_sub_function(x,y):
    return x-y

def my_div_function(x,y):
    return x/y
```

- Hàm testfunction.py được lưu cùng một folder (**C:\Users\sony\Desktop\python_co_ban\test\mymodule**)

testfunction.py | myfunction_2.py

```python
# -*- coding: utf-8 -*-
import myfunction, myfunction2

m = myfunction.my_add_function(1,2)
print(m) # 3
n = myfunction.my_mul_function(3,2)
print(n) # 6
k = myfunction2.my_sub_function(1,2)
print(k) # -1
l = myfunction2.my_div_function(3,2)
print(l) # 1.5
```

```
  File "C:/Users/sony/Desktop/python_co_ban/test/
mymodule/testfunction.py", line 2, in <module>
    import myfunction, myfunction2

ModuleNotFoundError: No module named 'myfunction2'
```

- Để khắc phục thì chúng ta phải thêm đường dẫn của thư mục **F:\myfuntion_2** chứa **module myfunction2** vào biến môi trường cho python (PYTHONPATH) dùng **sys.path**

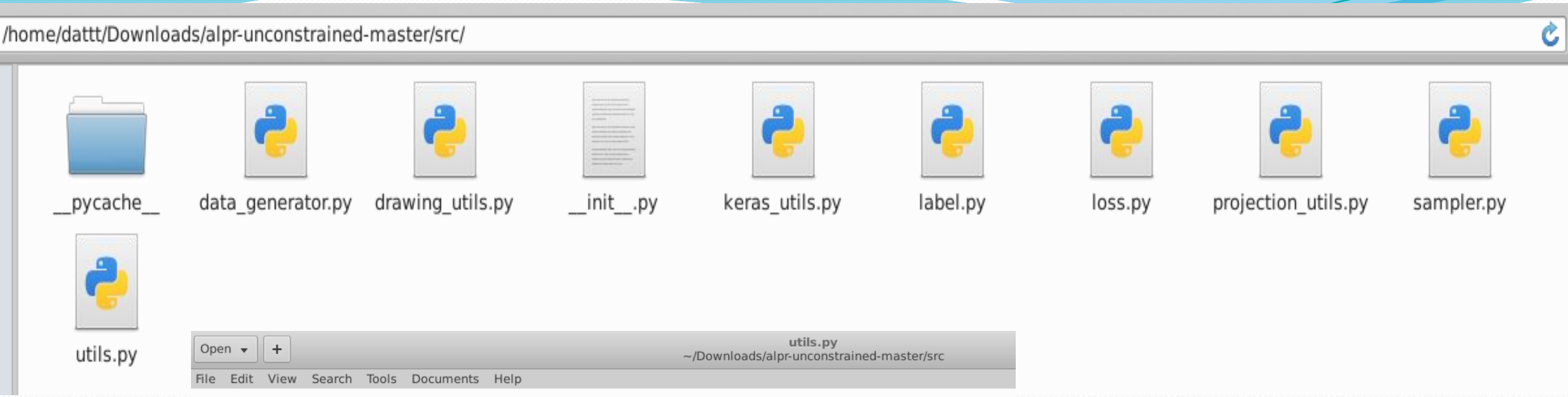
```
import sys
mymodule_path = 'F:\myfunction_2'
# thêm thư mục cần load vào trong hệ thống
sys.path.append(mymodule_path)
```

testfunction.py* | myfunction2.py

```
1  # -*- coding: utf-8 -*-
2  import sys
3  mymodule_path = 'F:\myfunction_2'
4  # thêm thư mục cần load vào trong hệ thống
5  sys.path.append(mymodule_path)
6
7  import myfunction, myfunction2
8  m = myfunction.my_add_function(1,2)
9  print(m) # 3
10 n = myfunction.my_mul_function(3,2)
11 print(n) # 6
12 k = myfunction2.my_sub_function(1,2)
13 print(k) # -1
14 l = myfunction2.my_div_function(3,2)
15 print(l) # 1.5
```

Package

- Các file tương tự hoặc cùng liên quan đến một chủ đề nào đó sẽ được để trong cùng một thư mục.
- Python có các **package cho thư mục** và **module cho file**.
- Khi chương trình đang code ngày càng lớn với rất nhiều module, chúng ta sẽ đặt những module giống nhau vào một package, và những nhóm module khác vào package khác.
- Trong một package có thể có package con và các module khác.
- Một thư mục phải chứa file có tên **__init__.py** để Python hiểu thư mục này là một package. File này có thể để trống.



Open + utils.py ~/Downloads/alpr-unconstrained-master/src

File Edit View Search Tools Documents Help

```
import numpy as np
import cv2
import sys

from glob import glob


def im2single(I):
        assert(I.dtype == 'uint8')
        return I.astype('float32')/255.


def getWH(shape):
        return np.array(shape[1::-1]).astype(float)


def IOU(tl1,br1,tl2,br2):
        wh1,wh2 = br1-tl1,br2-tl2
        assert((wh1>=.0).all() and (wh2>=.0).all())

        intersection_wh = np.maximum(np.minimum(br1,br2) - np.maximum(tl1,tl2),0.)
        intersection_area = np.prod(intersection_wh)
        area1,area2 = (np.prod(wh1),np.prod(wh2))
        union_area = area1 + area2 - intersection_area;
        return intersection_area/union_area
```

Ta có thể nhập các module từ package sử dụng toán tử dấu chấm (.)

Ví dụ:

```
import src.label
import src.utils
from src import loss
from src.utils import IOU,getWH
```