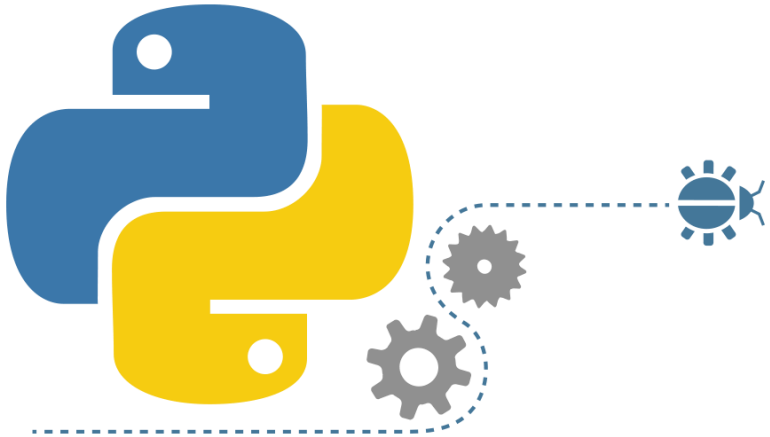
Module



a. Module trong Python là gì?

Module (mô-đun) đề cập đến một file chứa những câu lệnh Python và các định nghĩa. Một file chứa code Python, ví dụ ***mymodule.py*** được gọi là module và tên của module sẽ là ***mymodule***.

Module thường được sử dụng khi muốn chia chương trình lớn thành những file nhỏ hơn để dễ quản lý và tổ chức. Phổ biến nhất là những hàm Python hay phải sử dụng sẽ được định nghĩa trong một module và nhập vào Python thay vì sao chép định nghĩa trong những chương trình khác nhau. Nhờ thế, module cho phép tái sử dụng code.

Giờ thử tạo một module bằng cách nhập code sau vào file và lưu với tên ***mymodule.py***.

```
#Ví dụ về module Python
def them(p, q):
    """Module này thêm 2 số
    và trả về kết quả """

    ket_qua = p + q
    return ket_qua
```

Ở đây, chúng ta định nghĩa hàm them() trong module mymodule. Hàm sẽ lấy vào 2 số và trả về tổng của chúng.

b. Làm sao để nhập module trong Python?

Sử dụng từ khóa `import`

```
>>> import mymodule
```

Lệnh này không nhập tên của hàm được định nghĩa trong `mymodule` một cách trực tiếp trong bảng hiện tại, nó chỉ nhập tên của module mà thôi. Sử dụng tên module để truy cập vào hàm với toán tử (.). Ví dụ:

```
>>> mymodule.them(4.5,5.5)
10.0
```

Hãy chắc chắn bạn đã gõ đúng tên module, tên hàm (phân biệt chữ hoa, chữ thường) nếu bạn nhận được thông báo sau: "ModuleNotFoundError: No module named..."

Có sẵn rất nhiều module tiêu chuẩn. Bạn có thể kiểm tra danh sách đầy đủ tại địa chỉ: **https://docs.python.org/3/py-modindex.html**.

b1. Sử dụng lệnh import

Chúng ta có thể sử dụng lệnh import để nhập module và truy cập vào các định nghĩa bên trong nó, sử dụng toán tử . như mô tả bên trên. Đây là ví dụ:

```
# Lệnh import để nhập module có sẵn trong Python
import math
print("Giá trị của pi là: ", math.pi)
```

```
Giá trị của pi là: 3.141592653589793
```

b2. Nhập module và sửa tên khi nhập

```
# Nhập module và sửa tên nó
import math as m
print("Giá trị của pi là: ", m.pi)
```

Đổi tên module **math** là **m**, giúp tiết kiệm thời gian trong một số trường hợp.

Chú ý, việc đổi tên này chỉ áp dụng trong phạm vi lệnh, chứ không thực sự đổi tên module trong Lib.

Khi đã đổi tên, bạn phải gõ đúng tên module, **math** lúc này không được công nhận trong phạm vi lệnh nữa, mà bạn phải dùng **m** mới đúng

b3. Lệnh from...import trong Python

Chúng ta có thể nhập một tên cụ thể từ module mà không cần nhập toàn bộ module như ví dụ dưới đây:

```
# Chỉ nhập pi từ module math
from math import pi
print("Giá trị của pi là: ",pi)
```

Code trên chỉ nhập thuộc tính pi từ module math. Trong trường hợp này không cần sử dụng toán tử "."

Có thể nhập nhiều thuộc tính của module như ví dụ dưới đây:

```
# Nhập nhiều thuộc tính từ module math
>>> from math import pi, e
>>> pi
3.141592653589793
>>> e
2.718281828459045
```

b4. Nhập tất cả tên

Ta có thể nhập tất cả tên (các định nghĩa) từ một module sử dụng code sau:

```
# Nhập tất cả tên từ module math
form math import *
print("Giá trị của pi là: ",pi)
```

Chúng ta nhập tất cả các định nghĩa từ module math nên tất cả tên đều có thể nhìn thấy trong phạm vi này, ngoại trừ những tên bắt đầu bằng dấu gạch dưới _.

Nhập mọi thứ với dấu hoa thị * không phải là một thói quen lập trình tốt. Vì nó có thể dẫn đến những định nghĩa bị trùng lặp cho cùng một định danh và khiến cho việc đọc code trở nên khó khăn hơn

c. 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.

Nhập lần lượt các lệnh sau để xem đường dẫn:

```
>>> import sys
>>> sys.path
```

Chúng ta có thể chỉnh sửa danh sách này để thêm các path tùy chỉnh theo mong muốn.

d. Nạp lại module

Trình thông dịch Python chỉ nhập một module trong một phiên. Điều này làm cho mọi thứ trở nên hiệu quả hơn. Dưới đây là ví dụ minh họa cho cách hoạt động này.

Bạn viết code sau trong file mới, đặt tên là **module_1**:

```
print("Code đã hoạt động")
```

Giờ ta thử nhập module module_1 để xem sự hiệu quả của việc nhập module nhiều lần.

```
>>> import module_1
Code đã hoạt động
>>> import module_1
>>> import module_1
>>> import module_1
```

Code trên chỉ thực hiện một lần, nghĩa là `module_1` chỉ được nhập 1 lần trong 1 phiên làm việc đó. Bây giờ nếu sửa đổi code trong `module_1` thì phải nạp lại nó, phải khởi động lại trình thông dịch, nhưng cách này có vẻ không ổn lắm.
Python cung cấp cách thông minh hơn để làm điều này. Bạn có thể sử dụng hàm `reload()` trong module imp để nạp lại `module_1`, như dưới đây:

```
>>> import module_1
Code đã hoạt động
>>> import module_1
>>> import module_1
>>> import imp
>>> imp.reload(module_1)
Code đã hoạt động
<module 'module_1' from
'C:/Users/myusers/Python/Python36-32\\module_1.py'>
```

Các bài tập

Bài 1:

Yêu cầu:

Định nghĩa class Shape và class con là Rectangle trong file module_lib.py

Tạo file module_test.py, viết code để sử dụng các class trên.

Gợi ý:

- Shape có điểm gốc (x,y) , có hàm move truyền vào điểm (x1, y1) -> x = x +x1 và y = y + y1
- Rectangle kế thừa Shape, có thêm thuộc tính width và heigth và hàm **info** sẽ in chiều rộng và chiều cao và tọa độ điểm gốc
- Module sẽ import Shape và Rectangle, sau đó dùng lại hàm **move** và **info**