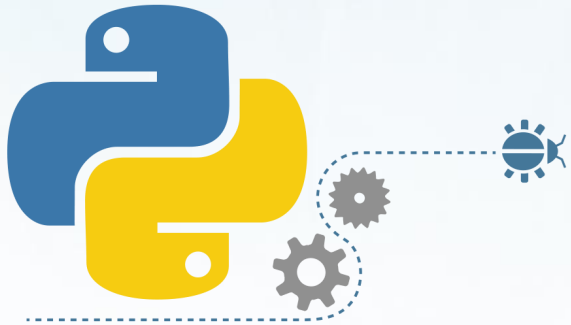
TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMC University of Technology and Education

KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN HỆ THỐNG THÔNG TIN

NHẬP MÔN LẬP TRÌNH PYTHON (IPPA233277)

SỬ DỤNG MODULES THƯ VIỆN TRONG PYTHON



GV. Trần Quang Khải

1. Nắm và vận dụng được khái niệm module
2. Hiểu về đường dẫn khi sử dụng module
3. Nắm và vận dụng được package trong python

1. Giới thiệu
2. Khai báo module
3. Đường dẫn khi sử dụng module
4. Package trong python

- Trong python, cũng như các ngôn ngữ lập trình khác, chương trình thường được chia thành nhiều tập tin mã nguồn khác nhau và được tổ chức theo dạng thư mục
- Python quản lý các tập tin và thư mục dưới dạng các module và package
- Module trong python bản chất là một tập tin chứa mã nguồn (lớp, hàm, biến,…) có phần đuôi mở rộng *.py

- Trong Python, có hai loại module được sử dụng:
  - ✓ Module được cung cấp sẵn (**Built-in Modules**): đi kèm với gói cài đặt Python, cung cấp một thư viện hỗ trợ gồm nhiều module khác nhau chứa các hàm dựng sẵn phong phú hữu ích
  - ✓ Module do người dùng tự định nghĩa (**User-defined Modules**): được thêm vào do chính người dùng hoặc cộng đồng đóng góp. Chúng ta hoàn toàn có thể tạo để sử dụng trong chương trình hoặc cho các chương trình khác chứa các lớp, hàm, biến,… theo đúng nhu cầu thiết kế

- Trong Python, câu lệnh `import` được sử dụng để thêm toàn bộ một module vào trong chương trình

```
import module_name
```

- Module sẽ được nạp một lần duy nhất dù có được khai báo ở bất kỳ đâu hoặc nhiều lần trong chương trình
- Khi cần sử dụng hàm hoặc đối tượng trong module, sử dụng cú pháp:

```
module_name.function_name()
```

- Nếu muốn sử dụng nhiều module, có thể import từng dòng hoặc dùng dấu phẩy (,) để ngăn cách giữa các module trong câu lệnh

Ví dụ:

```
import math, random # import 2 modules
print(math.factorial(5))
print(random.randint(10, 20))
```


- Nhiều khi tên module quá dài để truy xuất, có thể đổi tên module bằng từ khóa **as** và sử dụng tên này về sau

```
import math as m
print(m.sqrt(4))
```

- Đôi khi **không cần phải nạp toàn bộ module**, mà chỉ cần truy xuất một phần các đối tượng của module, ta sử dụng from…import…, lúc này sẽ không cần dùng tên module để truy xuất mà sử dụng trực tiếp

```
from math import sqrt, pi
print(sqrt(4 * pi)) # 3.5449077018110318
# hoặc đặt tên cho đối tượng
from math import sqrt as can
print(can(4)) # 2.0
```

- Nạp tất cả từ module sử dụng dấu (*) (nên hạn chế, thay thế bằng import module_name để trỏ đúng đối tượng)

```
from module_name import *
```


- Với from package import item, item có thể là submodule, subpackage, function, class, variable
- Với import item.subitem, subsubitem, mỗi item trừ cuối cùng bắt buộc phải là một package, item cuối cùng có thể là module, package nhưng không thể là function, class, variable

- Mỗi module có một thuộc tính __name__ chứa tên của module đó
- Có hai cách khi sử dụng module
  - ✓Thực thi trực tiếp (khi chạy module ở chế độ kịch bản): __name__được gán giá trị '__main__' là module chính của chương trình
  - ✓Thực thi gián tiếp: khi module được import vào một module khác (toàn bộ mã nguồn của module được thực thi một lần), __name__ chỉ là tên của module. Trường hợp này, Python sẽ lưu tạm một bản bytecode của module trong thư mục __pycache__ (nằm cùng thư mục với module được import)

Ví dụ: với module my_module, khi được import, Python sẽ tạo ra file __pycache__/my_module.cpython-39.pyc

Lưu ý:

- Việc lưu này, giúp cho những lần thực thi sau không cần dịch lại, giảm thời gian nạp/tải chương trình,
- Việc lưu này không xảy ra khi thực thi trực tiếp.
- File bytecode không làm chương trình chạy nhanh hơn nhưng làm giảm thời gian tải ứng dụng

- Do đó, khi xây dựng module cần chú ý:
  - ✓ Mỗi module sẽ có một hàm main() chứa những lệnh cần thực thi theo kiểu trực tiếp
  - ✓ Kiểm tra nếu __name__ == '__main__' thì thực thi main()

```
def main():
    'Thực thi các logic của chương trình'
    # code cần chạy
if __name__ == '__main__':
    main()
```

  - ✓ Khi chạy gián tiếp qua import module_name sẽ không thể có giá trị __main__ vì vậy main() sẽ không thực thi

- Khi khai báo sử dụng module, trình thông dịch sẽ tìm các module tương ứng theo thứ tự thư mục:
  - ✓Thư mục hiện hành mà script đang gọi
  - ✓Các thư mục trong **PYTHONPATH** (nếu có), một biến môi trường với danh sách thư mục
  - ✓Các đường dẫn mặc định trên Linux/Unix (/usr /local /lib /python)
- Đường dẫn tìm kiếm module lưu trữ trong sys.path chứa thư mục hiện hành, PYTHONPATH, đường dẫn mặc định

Ví dụ:

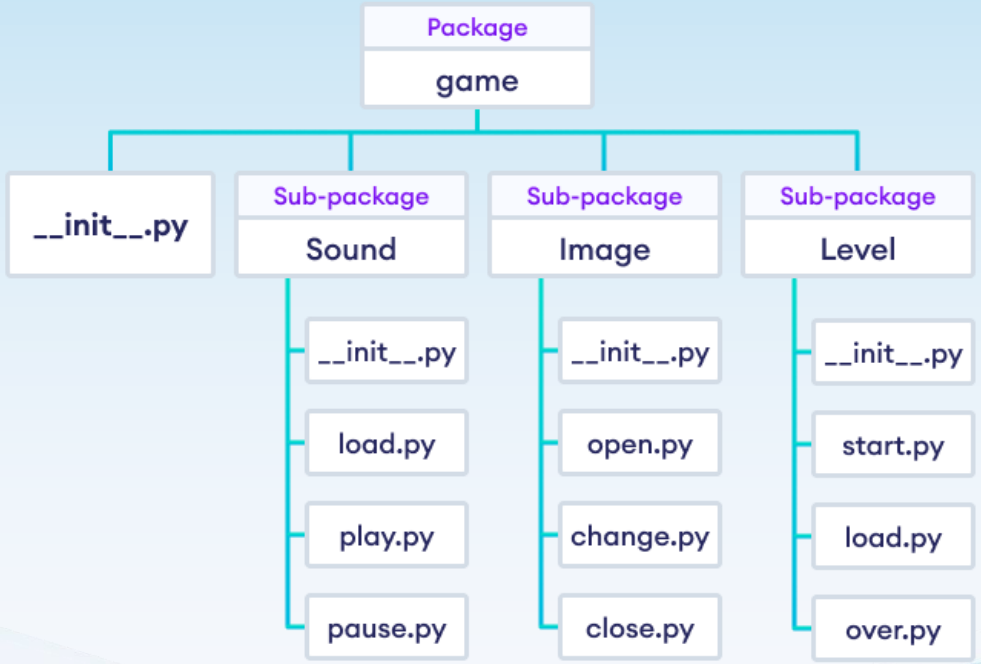
```
import sys
print(sys.path)
```

- Liệt kê thuộc tính, phương thức trong module sử dụng hàm dir()

```
import math
print(dir(math))
```

- Là một thư mục chứa một hoặc nhiều modules hay các packages khác nhau.
- Tạo package là tạo thư mục có tên <package> chứa __init__.py (như một constructor sẽ được thực thi đầu tiên khi import package)
- Cách thức khai báo package tương đồng với khai báo module

import <package_name>.<module_name> [as <alias_name>]



- Khai báo biến __all__ trong __init__.py của package

```
__all__ = [
        '<module_name>',
        …
]
```

Chứa danh sách các modules của package, trình thông dịch chỉ import các module có trong khai báo này

Ví dụ: `__all__ = ["echo", "surround", "reverse"]`

- Trường hợp không khai báo __all__ sẽ không import các submodules

Ví dụ: câu lệnh **from game.sound import *** sẽ **KHÔNG** import tất cả submodules của package game.sound vào mà chỉ đảm bảo là package game.sound được import, chạy tất cả code bên trong file __init__.py, và import tất cả các names được định nghĩa trong package. Những names này bao gồm tất cả function, class, variable được định nghĩa bên trong file __init__.py và các submodules được explicitly loaded

- Trong mỗi thư mục package đều có một tập tin __init__.py hay thư mục chứa tập tin __init__.py đều được xem lf một package với tên thư mục là tên của package
- Khi import một package cũng đồng thời import tất cả module của package đó và được chỉ định trong __init__.py

Lưu ý: trong __init__.py và trong module sử dụng package nên dùng cùng một cách import hoặc from .. Import

*(import * được xem là bad practice, from package import specific_submodule là cách được recommended)*

- ✓ Họ tên : Trần Quang Khải
- ✓ Email : khaitq@hcmute.edu.vn

- ✓ Zalo (mã Qr)