



BÀI 8

Packages và Module

Python

Tóm Tắt Nội Dung

Trong bài học này chúng ta sẽ đi tìm hiểu lần lượt các nội dung

- 1 Khái niệm về Module trong Python
- 2 Tự tạo một Module, quy tắc đặt tên Module
- 3 Cách sử dụng Module
- 4 Làm quen một số Built-in Module trong Python
- 5 Tổng qua về Package trong Python

8.1 Tổng quan về Module



Giới thiệu về Module

- ◆ Python được xây dựng với một Core (nhân/Lõi) bao gồm các thành phần và tính năng mà ngôn ngữ Python cung cấp sẵn.
- ◆ Các thành phần cốt lõi này là những yếu tố căn bản và cần thiết để xây dựng các ứng dụng Python. Ngoài ra, Python cũng có một hệ sinh thái phong phú của các module và thư viện bên thứ ba, cho phép mở rộng và mở rộng thêm tính năng cho ngôn ngữ.
Các thư viện này được gọi chung là **module**.
- ◆ Chúng ta có thể sử dụng hàm dir với từ khóa `__builtins__` để liệt kê tất cả các hàm hệ thống có sẵn trong phần Core của Python

```
dir(__builtins__)
```

8.1 Tổng quan về Module



Cách import Module

- ◆ Cú pháp cơ bản:

```
import module_name
```

- ◆ Import với tên rút gọn (alias)

```
import math as m
print(m.sqrt(25))
```

8.2 Built-in Modules



Module Math

- ◆ Module **math** cung cấp một loạt các hàm và hằng số toán học trên các số thực (float) và số nguyên (integer), bao gồm các phép tính như căn bậc hai, lượng giác, lôgarit, số mũ, làm tròn, v.v
- ◆ min(), max(), abs() là các built-in math functions không nằm trong module Math

```
import math
x = math.sqrt(64)
x = math.ceil(1.4)
y = math.floor(1.4)

print(x)
print(x) # returns 2
print(y) # returns 1
```

8.2 Built-in Modules



Module cMath

- ◆ Cung cấp các hàm và hằng số để thực hiện các phép toán số học trên các số phức. Nó cung cấp các hàm và phép toán như căn bậc hai phức, lượng giác phức, logarit phức, v.v.
Module cmath làm việc với số phức, cho phép thực hiện các phép toán phức tạp

```
import cmath
x = cmath.sqrt(-1) # Căn bậc hai phức của -1
print(x) # Output: 1j
y = cmath.exp(cmath.pi * 1j) # Exp phức của π
print(y) # Output: (-1+1.2246467991473532e-16j)
z = cmath.log10(100 + 100j) # Logarit phức cơ số 10 của 100 + 100j
print(z) # Output: (2+0.7853981633974483j)
```

8.2 Builtin Modules



Module datetime

Cung cấp các lớp và hàm để làm việc với thời gian và ngày tháng

- ◆ **datetime**: Đại diện cho một điểm thời gian cụ thể, bao gồm cả ngày, tháng, năm và thời gian (giờ, phút, giây, microgiây)
- ◆ **date**: Đại diện cho một ngày duy nhất trong lịch, bao gồm năm, tháng và ngày
- ◆ **time**: Đại diện cho một thời gian cụ thể trong ngày, bao gồm giờ, phút, giây và microgiây
- ◆ **timedelta**: Đại diện cho một khoảng thời gian hoặc một sự chênh lệch giữa hai điểm thời gian

Ngoài ra, module datetime cho phép thực hiện định dạng hiển thị thời gian theo chuẩn, chuyển đổi múi giờ...

8.2 Built-in Modules



Module datetime

```
import datetime

# Lấy thời gian hiện tại
current_time = datetime.datetime.now()
print("Thời gian hiện tại:", current_time)

# Truy xuất các thành phần của thời gian
year = current_time.year #lấy được năm hiện tại
month = current_time.month #lấy được tháng hiện tại
day = current_time.day #lấy được ngày hiện tại
hour = current_time.hour #lấy được giờ hiện tại
minute = current_time.minute #lấy được phút hiện tại

# Định dạng thời gian theo kiểu "dd-mm-yyyy H:i:s"
formatted_time = current_time.strftime("%d-%m-%Y %H:%M:%S")
```

8.2 Built-in Modules



Module random

Cung cấp các hàm và phương thức để cho phép bạn tạo số ngẫu nhiên, chọn ngẫu nhiên từ một tập hợp, và thực hiện các tác vụ liên quan đến số ngẫu nhiên.

Dưới đây là một số hàm và phương thức quan trọng trong module random:

- ◆ **random()**: Trả về một số ngẫu nhiên từ khoảng [0.0, 1.0)
- ◆ **randrange(start, stop, step)**: Trả về một số ngẫu nhiên trong khoảng từ start đến stop (không bao gồm stop) với bước nhảy step.
- ◆ **choice(sequence)**: Chọn ngẫu nhiên một phần tử từ một sequence (danh sách, tuple, chuỗi).
- ◆ **shuffle(sequence)**: Xáo trộn một sequence theo thứ tự ngẫu nhiên.
- ◆ **sample(population, k)**: Trả về một mẫu ngẫu nhiên gồm k phần tử từ một population (danh sách, tuple, chuỗi) mà không có sự trùng lặp.

8.2 Built-in Modules



Module random

Một số ví dụ

```
import random
```

```
# Tạo số ngẫu nhiên từ 0.0 đến 1.0
random_number = random.random()
```

```
# Tạo số ngẫu nhiên từ 1 đến 10
random_int = random.randint(1, 10)
```

```
# Chọn ngẫu nhiên một phần tử từ danh sách
my_list = [1, 2, 3, 4, 5]
random_element = random.choice(my_list)
```

8.2 Built-in Modules



Module os

Một số ví dụ

```
import random
```

```
# Tạo số ngẫu nhiên từ 0.0 đến 1.0
random_number = random.random()
```

```
# Tạo số ngẫu nhiên từ 1 đến 10
random_int = random.randint(1, 10)
```

```
# Chọn ngẫu nhiên một phần tử từ danh sách
my_list = [1, 2, 3, 4, 5]
random_element = random.choice(my_list)
```

8.1 Tổng quan về Module



Tự tạo một Module

Ngoài những Module có sẵn bạn có thể tự tạo cho mình một Module

Tạo một file đặt tên là **mymodule.py**

```
def greeting(name):
    print("Hello, " + name)
```

Module có thể là function, list, tuple, set, dictionaries...

```
person1 = {
    "name": "John",
    "age": 36,
    "country": "Norway"
}
```

8.1 Tổng quan về Module



Quy tắc đặt tên Module

- ◆ Theo quy ước, tên module trong Python nên được viết bằng **chữ thường** và sử dụng dấu gạch dưới (_) để phân tách các từ. Ví dụ: my_module, utils, data_processing, v.v.

Đặt tên theo kiểu **snake_case**

- ◆ Đảm bảo rằng tên module của bạn không trùng lặp với tên module có sẵn trong Python hoặc các module bên thứ ba. Điều này sẽ tránh xung đột và khó khăn trong việc tìm kiếm và sử dụng module
- ◆ Hãy chọn tên module có ý nghĩa và liên quan đến nội dung và chức năng của module. Điều này giúp người đọc mã dễ hiểu và tìm hiểu mục đích sử dụng của module.

8.1 Tổng quan về Module



Cách Sử dụng Module đã tạo ra

```
import mymodule  
mymodule.greeting("Jonathan")
```

Dùng từ khóa **import** để nạp module vào, với tên module chính là tên mà bạn đã đặt cho file. Sử dụng cú pháp **tên_module.properties**

Properties: có thể là function, biến

```
import mymodule as mx  
a = mx.person1["age"]  
print(a)
```

Bạn có thể **đổi tên** module với từ khóa **as** như ví dụ trên

8.1 Tổng quan về Module



Cách Sử dụng Module đã tạo ra

Trường hợp một module có nhiều thành phần như ví dụ sau:

```
def greeting(name):
    print("Hello, " + name)

person1 = {"name": "John", "age": 36, "country": "Norway"}
```

Nhưng bạn chỉ muốn sử dụng một thành phần trong module đó thì dùng từ khóa **from** như dưới đây

```
from mymodule import person1
print (person1["age"])
```

8.2 Built-in Modules



Nguồn tham chiếu

- ◆ Tổng quan về Module

https://www.w3schools.com/python/python_modules.asp

- ◆ Cách sử dụng Module Datetime

https://www.w3schools.com/python/python_datetime.asp

- ◆ Cách sử dụng Module Math

https://www.w3schools.com/python/python_math.asp

- ◆ Cách sử dụng Module Random

https://www.w3schools.com/python/module_random.asp

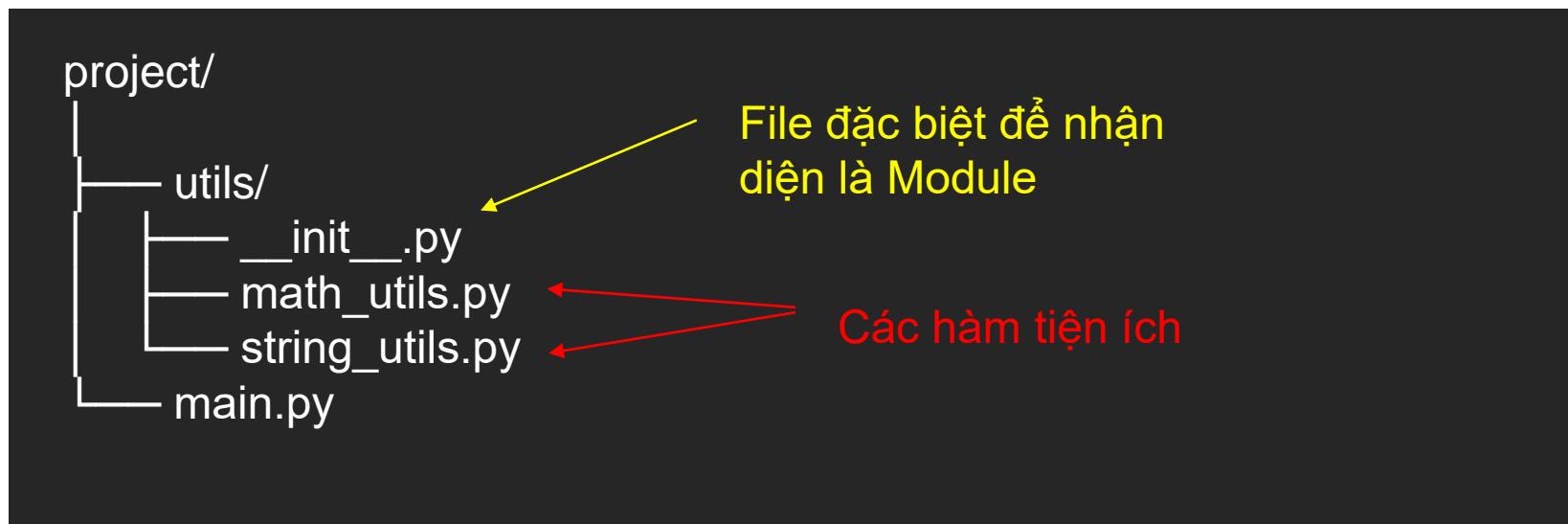
8.3 Package



Định nghĩa

Package là một **thư mục chứa nhiều module**, có file đặc biệt `__init__.py` (có thể rỗng).

Cấu trúc ví dụ:



8.4 Review Lessons 1-7

- ◆ Biến & Kiểu dữ liệu
- ◆ Cấu trúc điều khiển
- ◆ Vòng lặp
- ◆ Cấu trúc dữ liệu: list, tuple, set, dict
- ◆ Hàm
- ◆ Xử lý ngoại lệ
- ◆ File I/O

Tổng kết lại bài 8

- 6 Hiểu được thế nào là Module trong Python
- 7 Biết cách tự tạo một Module, nắm được quy tắc đặt tên Module
- 8 Nắm được cách sử dụng Module
- 9 Làm quen một số Built-in Module trong Python