[Bài đọc] Module

Module là gì?

Module (mô-đun) là một khái niệm trong Python chỉ đến những file chứa những câu lệnh Python và các định nghĩa.

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 thường xuyên được 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 và dễ dàng chia sẻ.

Tạo module

Save đoạn code sau vào 1 file có tên là my_module.py. Việc đặt tên cho module theo quy tắc đặt tên biến. Như vậy ta đã tạo ra 1 module đơn giản.

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

Sử dụng moduleĐể sử dụng module vừa tạo ở trên chúng ta sử dụng lệnh **import**. Chi tiết về **import** đọc bài 06.

```
import my_module
my_module.greeting("John")
```

Lưu ý: Trong ví dụ trên, khi import ta có thể có như sau: import my_module.greeting, khi đó lúc gọi hàm ta chỉ cần gọi mỗi tên hàm mà không cần tên module ở phía trước.

Biến trong module

Thông thường module thường chứa tập hợp các hàm, tuy nhiên nó cũng có thể chứa tất cả các biến của tất cả các kiểu dữ liệu (mảng, dictionary, đối tượng…)

Bổ sung đoạn code sau vào file my_module.py ở trên:

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

Lúc này ta có thể truy xuất đến các dữ liệu của biến person1 như sau

```
import my_module
a = my_module.person1["age"]
```

```
print(a)
```

[Bài đọc] Hàm và sử dụng hàm

Hàm là gì?

Hàm là một tập hợp các tập lệnh Python và nó giải quyết một nhiệm vụ cụ thể nào đó. Ví dụ hàm tính giai thừa của một số, ….

Lợi ích của việc định nghĩa hàm là làm cho chương trình được chia nhỏ thành những khối, module. Đối với một chương trình lớn, nhiều dòng code thì hàm làm cho chương trình được tổ chức một cách khoa học, dễ quản lý. Hơn nữa, nó giúp tránh trùng lặp khi viết code và hiệu quả trong việc tái sử dụng code.

Cú pháp

```
def function_name(parameters):
        """docstring"""
        statement(s)
```

Trong cú pháp trên, hàm có các thành phần sau:

1. Từ khóa def để bắt đầu tiêu đề của hàm
2. Function_name tên hàm là duy nhất. Tên hàm được đặt theo quy tắc đặt tên biến
3. Parameters là tham số của hàm. Nhiều hàm không có tham số, tham số là thành phần không bắt buộc khi khai báo hàm
4. Dấu (:) đánh dấu sự kết thúc tiêu đề của hàm
5. docstring – Là tài liệu mô tả về hàm, thường là mô tả chức năng của hàm này để làm gì, các tham số như thế nào… Tài liệu này rất hữu ích khi hàm được viết cho bên thứ 3 sử dụng, căn cứ vào tài liệu này, hàm được sử dụng một cách dễ dàng hơn. Thành phần này là không bắt buộc
6. statements – các khối lệnh của hàm

Ví dụ

```
def greet(name):
    """
    Hàm này in ra dòng chào
    tên người được truyền vào hàm
    :param name: Tên người
    """
    print("Xin chào! "+name)
```

Sử dụng hàm

Sau khi hàm đã được định nghĩa, khai báo, nó có thể được sử dụng (gọi) trong một chương trình khác, một hàm khác. Hàm được gọi thông qua tên của hàm với các tham số tương ứng như lúc khai báo hàm.

Ví dụ, với hàm được định nghĩa ở trên, hàm có thể được gọi như sau:

```
greet("Nguyễn Văn Bình")
```

Sử dụng docstring

Để hiển thị tài liệu mô tả về hàm, ta dùng thuộc tính __doc__ của hàm. Ví dụ, với hàm định nghĩa ở trên, ta hiển thị mô tả về hàm như sau:

```
print(greet.__doc__)
```

Và kết quả:

```
Hàm này in ra dòng chào
    tên người được truyền vào hàm
    :param name: Tên người
```

Hàm nặc danh

Hàm nặc danh là hàm không được định nghĩa theo cách bình thường với từ khóa def. từ khóa lambda được sử dụng để định nghĩa hàm nặc danh.

Cú pháp: Hàm nặc danh chỉ chứa một câu lệnh như cú pháp bên dưới

```
lambda [arg1 [,arg2,.....argn]]:expression
```

Ví dụ: Hàm tính tổng hai số được định nghĩa với kiểu hàm nặc danh

```
sum = lambda num1, num2: num1+num2

print("Tổng 1 = ", sum(3, 5))
print("Tổng 2 = ", sum(-2, 14))
```

Hàm rỗng

Khi định nghĩa hàm, thân hàm không được để trống, nhưng đôi khi hàm được định nghĩa mà không có nội dung trong thân hàm, để tránh phát sinh ra lỗi cho chương trình, từ khóa pass được sử dụng để đặt trong thân hàm đó.

Ví dụ:

```
def my_function():
    pass
```

[Bài đọc] Phạm vi biến

Biến toàn cục (global), biến cục bộ (local)

Biến toàn cục là biến được khai báo ngoài phạm vi của hàm. Biến toàn cục có thể được sử dụng mọi nơi, có thể trong thân của hàm hoặc ngoài hàm.

Ví dụ: Khai báo biến toàn cục ở ngoài hàm nhưng được sử dụng ở trong hàm

```
x = "biến toàn cục x"
def my_function():
       print("Gọi biến toàn cục ở trong hàm: " + x)
my_function()
```

Nếu có một biến được khai báo trong hàm cùng tên với biến toàn cục đã được khai báo ở ngoài hàm, thì biến được khai báo trong hàm là biến cục bộ (local) và biến này chỉ được sử dụng trong hàm đó mà thôi. Chúng ta không thể sử dụng các biến được khai báo trong hàm ở bên ngoài phạm vi của hàm đó. Còn biến toàn cục về mặt giá trị của biến không thay đổi khi khai báo biến cục bộ cùng tên trong hàm.

Ví dụ:

```
x = "biến toàn cục x"
def my_function():
    x = "biến cục bộ, được khai báo trong hàm!"
    print(x)
my_function()
print(x)
```

Từ khóa global

Thông thường, biến được khai báo trong hàm là biến cục bộ và chỉ được sử dụng bên trong hàm đó. Tuy nhiên, chúng ta có thể khai báo một biến toàn cục ở trong hàm với từ khóa **global**.

Ví dụ: Tạo biến toàn cục ở trong hàm với từ khóa **global**

```
def my_function():
    global x
    x = "biến toàn cục, được khai báo trong hàm!"
    print(x)
my_function()
print(x)
```

[Bài đọc] Đối số của hàm

Đối số của hàm là gì?

Đối số của hàm là dữ liệu cần được truyền vào cho hàm để hàm thực thi. Số lượng đối số truyền vào cho hàm phải đúng như khi khai báo hàm. Không được truyền vào nhiều hơn hay ít hơn.

Lưu ý: Khái niệm đối số và tham số của hàm nhiều khi được dùng thay thế cho nhau. Tuy nhiên, chúng có sự khác biệt chút ít. Tham số của hàm là các biến của hàm được liệt kê trong dấu ngoặc khi định nghĩa hàm, còn đối số là giá trị của các tham số được truyền vào hàm khi gọi hàm đó.

Ví dụ: Cách khai báo và gọi hàm tính tổng hai số

```
def sum(a, b):
    return a + b


c = sum(4, 6)
print(c)
```

Ta thấy rằng, hàm sum tính tổng hai số khi định nghĩa gồm có 2 đối số a và b. Nếu khi sử

```
def sum(a, b):
    return a + b


c = sum(4)
print(c)
Traceback (most recent call last):
  File "C:/caocao/Python Project/Python Fundamental/hello10.py", line 5,
in <module>
    c = sum(4)
TypeError: sum() missing 1 required positional argument: 'b'
```

Từ khóa đối số

Khi chúng ta gọi hàm, các giá trị đối số truyền vào hàm có thể sử dụng cùng với tên của các tham số khi hàm đó được định nghĩa. Bằng cách này, khi truyền các giá trị của đối số vào hàm chúng ta có thể truyền theo thứ tự bất kì. Trình thông dịch có thể sử dụng tên của các đối số hàm để truyền vào các giá trị tương ứng.

Ví dụ: Hàm tính tổng khai báo ở trên có thể được gọi bằng các cách sau với từ khóa:

```
def sum (a, b):
    return a + b
```

```
c = sum(b=6, a=4)
d = sum(a=4, b=6)
print(c)
print(d)
```

Giá trị mặc định của đối số

Giá trị mặc định của đối số hàm là giá trị của đối số sẽ được hàm sử dụng trong trường hợp người dùng không truyền vào khi gọi hàm.

Ví dụ:

```
def sum(a, b=10):
    return a + b


d = sum(a=4)
print(d)
```

Ở ví dụ trên ta thấy, khi định nghĩa hàm, b có giá trị mặc định =10. Khi Gọi hàm bên dưới, người dùng chỉ truyền vào giá trị của a mà không có giá trị của b. Trong trường hợp này, giá trị mặc định của b = 10 sẽ được hàm sử dụng cho phép tính tổng a và b. Kết quả chương trình là 14.

Đối số *args

Khi định nghĩa hàm, có những lúc chúng ta không xác định được trước số lượng đối số của hàm, hoặc số lượng có thể thay đổi. Khi đó, lúc định nghĩa ta thêm dấu * trước tên của đối số.

Ví dụ:

```
def sum(*matrix):
    s = 0
    for i in range(len(matrix)):
        s += matrix[i]
    return s


d = sum(1, 2, 3)
print(d)
```

[Thực hành] Hàm tính chu vi, diện tích hình tròn

Mục tiêu:

Luyện tập với cách thức định nghĩa hàm và gọi hàm

Mô tả bài toán

Xây dựng hàm tính chu vi hình tròn với tham số là bán kính. Xây dựng tiếp hàm tính diện tích hình tròn với tham số là bán kính hình tròn. Sau khi các hàm được xây dựng xong, tiến hành gọi các hàm với giá trị bán kính được nhập bởi người dùng.

[Thực hành] Chuyển đổi nhiệt độ

Mục tiêu:

Luyện tập với cách thức định nghĩa hàm và gọi hàm

Mô tả bài toán

Xây dựng hàm chuyển đổi nhiệt độ Fahrenheit (độ F) sang độ Celsius (độ C). Sau khi hàm được xây dựng xong, tiến hành gọi hàm với các giá trị tham số khác nhau

Sử dụng công thức f = (9*(int(c))/5)+32 để chuyển đổi từ độ C sang độ F

[Thực hành] Vẽ hình đa giác sử dụng hàm

Mục tiêu:

Luyện tập với cách thức định nghĩa hàm và gọi hàm

Mô tả bài toán

Xây dựng hàm vẽ hình đa giác đều với tham số là số cạnh của đa giác, và chiều dài của các cạnh. Sau khi hàm được xây dựng xong, tiến hành gọi hàm với các giá trị tham số khác nhau.

[Thực hành] Hàm tính diện tích hình tròn

Mục tiêu:

Luyện tập với cách thức định nghĩa hàm và gọi hàm

Mô tả bài toán

Xây dựng hàm vẽ hình tròn với tham số là bán kính. Xây dựng tiếp hàm tính diện tích hình tròn với tham số là bán kính hình tròn. Sau khi các hàm được xây dựng xong, tiến hành gọi các hàm với giá trị bán kính được nhập bởi người dùng.

[Thực hành] Vẽ hình vuông sử dụng hàm

Mục tiêu:

Luyện tập với cách thức định nghĩa hàm và gọi hàm

Mô tả bài toán

Xây dựng hàm vẽ hình vuông với tham số là độ dài các cạnh. Sau khi hàm được xây dựng xong, tiến hành gọi hàm với các giá trị tham số khác nhau.