BÀI 4

Hàm (Function) Python

Tóm Tắt Nội Dung

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

1. Tìm hiểu và cách định nghĩa một function (hàm) trong Python
2. Function có chứa tham số, tham số tùy chọn
3. Function có trả về dữ liệu với câu lệnh return
4. Phạm vi sử dụng biến trong function
5. Anonymous function (A lambda function) trong Python

4.1 Tìm hiểu về Function trong Python

Định nghĩa: function là gì ?

Trong lập trình, **Function** (hàm) là một khối mã có tên được sử dụng để thực hiện một tác vụ cụ thể.

Function giúp chia nhỏ code thành các phần nhỏ hơn, dễ quản lý và sử dụng lại (tái sử dụng)

Cú pháp định nghĩa 1 function

```
def  name_function():
        #Khối lệnh được thực thi khi hàm được gọi
```

Name_function: Đặt theo quy tắc tên biến trong Python

Khối lệnh của hàm: phải thụt vào ít nhất 1 kí tự trắng so với lệnh def

4.1 Tìm hiểu về Function trong Python

Ví dụ về function

Một hàm có thể có hoặc không có giá trị trả về (return value)

```
def my_function():
    print("Lệnh 1")
    print("Lệnh 2")

my_function() #Gọi hàm, cho hàm khởi chạy
```

Trong ví dụ trên hàm chỉ thực thi các lệnh mà không trả lại giá trị gì thì thường được gọi là hàm **void** (Hàm thủ tục)

4.2 function có chứa tham số, tham số tùy chọn

4.2.1 Function có chứa tham số (parameter)

- **Tham số (Parameters):** là các biến được khai báo trong phần khai báo hàm, nhằm đại diện cho các giá trị sẽ được truyền vào khi gọi hàm
- Tham số được đặt tên và có thể có hoặc không có giá trị mặc định
- Khi bạn định nghĩa hàm, bạn có thể khai báo một hoặc nhiều tham số, phân tách chúng bằng dấu phẩy

```
def hello_name(name):
    print("Hello " + name)
```

4.2 function có chứa tham số, tham số tùy chọn

4.2.1 Function có chứa tham số (parameter)

- **Đối số (Arguments):** là các giá trị cụ thể được truyền vào hàm khi bạn gọi nó
- Số lượng và thứ tự của các đối số phải tương ứng với số lượng và thứ tự của các tham số tương ứng trong phần định nghĩa hàm
- Bạn có thể truyền đối số bằng giá trị cụ thể hoặc biến chứa giá trị

```
def hello_name(name):
    print("Hello " + name)

hello_name("Jonh") #Gọi hàm, cho hàm khởi chạy

name = 'Jonh'
hello_name(name) #đối số có thể là biến
```

4.2 function có chứa tham số, tham số tùy chọn

4.2.1 Function có chứa tham số tùy chọn/mặc định

- Giá trị mặc định chỉ được sử dụng nếu không có đối số tương ứng được truyền vào khi gọi hàm

```
def hello_name(name = ‘Guest’):
      print(“Hello “ + name)

hello_name(“Jonh”) #Kết quả: Hello Jonh
hello_name() #Kết quả: Hello Guest
```

Trong ví dụ trên, tham số name của hàm hello_name() có giá trị mặc định là "Guest".
Nếu không có đối số được truyền, giá trị mặc định "Guest" sẽ được sử dụng

4.2 function có chứa tham số, tham số tùy chọn

4.2.1 Function có nhiều tham số

```
def hello_name(fname, lname, age = 18):
    print(fname + " " + lname , age)
```

- Khi bạn định nghĩa hàm, bạn có thể khai báo một hoặc nhiều tham số, phân tách chúng bằng dấu phẩy
- Tham số mặc định luôn luôn được định nghĩa sau cùng

4.3 function có trả về return

Cách định nghĩa

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

- Hàm nhận a, b là tham số đầu vào
- Hàm trả về (return) giá trị tổng của 2 số a và b

Ví dụ gọi hàm sum() và sử dụng giá trị trả về:

```
result = sum(5, 3)
print(result) # Kết quả: 8
```

4.4 Phạm vi sử dụng biến trong Function

Local Scope (Phạm vi cục bộ)

Biến được khai báo bên trong khối lệnh của hàm thì chỉ sử dụng được bên trong phạm vi khối hàm đó, kể cả hàm con của nó.

```
def myFunction():
        x =  3;
        print(x)
        def subFunction():
                print(x)
```

Qua ví dụ trên biến **x** có thể sử dụng được cả trong hàm con subFunction

4.4 Phạm vi sử dụng biến trong Function

Global Scope (Phạm vi toàn cục)

Ví dụ thứ 2: Biến toàn cục với từ khóa global

```
def myFunction():
    global x
    x = 3;
    print(x)

print(x)
```

Qua ví dụ trên biến **x** dùng được trong hàm myFunction, vừa dùng được bên ngoài hàm myFunction

4.4 Phạm vi sử dụng biến trong Function

Global Scope (Phạm vi toàn cục)

Để thay đổi giá trị của một biến toàn cục bên trong một hàm, hãy tham chiếu đến biến đó bằng cách sử dụng từ khóa toàn cục:

```
x =  2;
def myFunction():
      global x
      x =  3;
      print(x)

print(x)
```

Qua ví dụ trên biến **x** cuối cùng được in ra là x = 3

4.5 Hàm nặc danh (lambda function)

Trong Python, Lambda function (còn được gọi là hàm vô danh) là một loại hàm đặc biệt mà không cần định nghĩa bằng từ khóa def như các hàm thông thường. Lambda function được tạo ra bằng cú pháp ngắn gọn và linh hoạt, thường được sử dụng trong các trường hợp đơn giản khi bạn chỉ cần một hàm nhỏ mà không cần đặt tên

Cú pháp

lambda arguments: expression

- **lambda** là từ khóa để định nghĩa Lambda function
- **arguments** là danh sách các tham số (có thể không có hoặc có nhiều tham số)
- **expression** là biểu thức mà Lambda function sẽ thực hiện và trả về kết quả, và chỉ duy nhất 1 dòng lệnh

4.5 Hàm nặc danh (lambda function)

Ví dụ về lambda function

lambda với một tham số a, và khi đó a + 10 chính là return của hàm

```
x = lambda a : a + 10
print(x(5))
```

lambda với hai tham số a, b – các tham số cách nhau bằng dấu phẩy

```
x = lambda a, b : a * b
print(x(5, 6))
```

4.5 Hàm nặc danh (lambda function)

Ví dụ về lambda function

lambda bên trong một function khác

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

mydoubler = myfunc(2)
print(mydoubler(12))
```

Qua ví vụ này chúng ta phải gọi 2 lần để thực thi được chương trình

Tổng kết lại bài 4

1. Nắm được cách định nghĩa một function (hàm) trong Python
2. Cách định nghĩa một Function có chứa tham số, tham số tùy chọn
3. Cách định nghĩa một Function có trả về dữ liệu với câu lệnh return
4. Hiểu thế nào là phạm vi sử dụng biến trong function
5. Cách tạo Anonymous function (A lambda function) trong Python