BÀI 5

Hàm (Function) và Module 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

Tóm Tắt Nội Dung

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

6 Khái niệm về Module trong Python

7 Tự tạo một Module, quy tắc đặt tên Module

8 Cách sử dụng Module

9 Làm quen một số Built-in Module trong Python

5.1 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

5.1 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)

5.2 function có chứa tham số, tham số tùy chọn

5.2.1 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)
```

5.2 function có chứa tham số, tham số tùy chọn

5.2.1 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
```

5.2 function có chứa tham số, tham số tùy chọn

5.2.1 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

5.2 function có chứa tham số, tham số tùy chọn

5.2.1 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

5.3 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
```

5.4 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

5.4 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

5.4 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

5.5 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

5.5 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))
```

5.5 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

5.5 Hàm nặc danh (lambda function)

Nguồn tham chiếu

- Tổng quan về Function

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

- Hàm nặc danh Lambda

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

- Danh mục các hàm dựng sẵn - Built in Functions

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

5.1 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__)
```

5.1 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"
}
```

5.1 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.

5.1 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 phép **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

5.1 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"])
```

5.2 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
```

5.2 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 πi
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)
```

5.2 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…

5.2 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")
```

5.2 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.

5.2 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)
```

5.2 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

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

Tổng kết lại bài 4

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