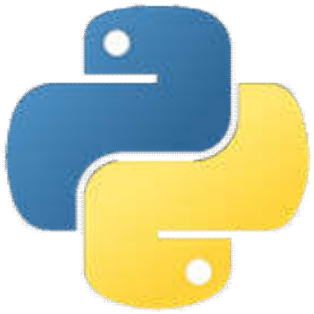

Hướng đối tượng trong Python

AUTHOR: LÊ TRUNG THỰC - EMAIL: THUCLT12A@GMAIL.COM

Nội dung

1. Hướng đối tượng (Object Oriented)
2. Class
3. Attribute & Method
4. Constructor
5. Bài tập thực hành

Tổng quan về OOP (1)

- Hướng thủ tục biểu hiện ở việc sử dụng các hàm trong Python. Lập trình viên có thể định nghĩa các hàm, và sử dụng tại các module khác trong chương trình Python.
- Hướng đối tượng trong **Python** biểu hiện ở việc sử dụng class.
- Class là một nguyên mẫu (prototype) để tạo ra các đối tượng (object hay instance).
- Tính chất cơ bản của OOP (Object-Oriented Programing):
  - Trừu tượng (Abstraction)
  - Đa hình (Polymorphism)
  - Đóng gói (Encapsulation)
  - Kế thừa (Inheritance)

Tổng quan về OOP (2)

- **Class** (lớp): Được định nghĩa cho một đối tượng bao gồm một tập các attributes (thuộc tính) đặc trưng cho tất cả các đối tượng của lớp.
- **Instance**: Một hiện thực cụ thể của một lớp, có thể gọi là thực thể.
- **object**: instance duy nhất chứa cấu trúc dữ liệu được định nghĩa bởi class. (Ví dụ: khai báo lớp HocSinh ta sẽ có object HocSinh, sau đó tạo biến nguyen_van_a = HocSinh() thì có thêm instance nguyen_van_a).
- **Attributes** gồm: *data members* và *methods* được gọi thông qua Instance.
- Methods: Hàm thực thi trong class.

Tổng quan về OOP (3)

- **Data members**:
  - **Class variable**: Biến dùng chung cho tất cả các đối tượng của lớp, được định nghĩa trong lớp mà không nằm trong methods (hàm thực thi) nào cả. Các biến này không được sử dụng thường xuyên.
  - **Instance variable**: biến được định nghĩa bên trong methods và chỉ thuộc về các instance (đối tượng thực thể của lớp).
- **Function overloading**: method định nghĩa các phép toán nhiều instances tham gia.
- **Operator overloading**: Phép toán cần nhiều method tham gia.

Class

- **Class** – Là một nguyên mẫu dùng để định nghĩa đối tượng.
- Cú pháp khởi tạo một class:

```
class ClassName:
  'Mô tả ngắn về class (Không bắt buộc)'
  # Code ...
```

Thuộc tính & Phương thức

- **Attribute – Thuộc tính** là biến nằm trong một lớp. Thuộc tính mô tả các đặc tính của một đối tượng.
- **Class variable** – Tương đương với khái niệm trường tĩnh (Static Field) như **Java. Biến của lớp** (gần như khái niệm biến toàn cục) có thể được truy cập thông qua tên lớp hoặc thông qua đối tượng từ một lớp khác trong cùng project.
- **Method – Phương thức** là các hàm được định nghĩa bên trong một class. Các phương thức được sử dụng để thực hiện các công việc cụ thể.

Phương thức khởi tạo

- **Constructor** – là một phương thức đặc biệt của class (trong python mặc định là **__init__**).
- ***self*** **-** Tham số đầu tiên của constructor (Một từ khóa trỏ chính đến class đó).
- Constructor được sử dụng để tạo ra **một đối tượng.**
- Constructor gán giá trị từ tham số vào thuộc tính của đối tượng.
- **Một class chỉ có một phương thức khởi tạo.**
- Nếu class không được định nghĩa constructor, Python mặc định hiểu constructor là **__init__(self).**

```
class NhanVien:
    'Lớp mô tả cho mọi nhân viên'
    dem = 0

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        NhanVien.dem += 1

    def hien_thi_so_luong(self):
        print("Tổng số nhân viên được tạo: %d" % NhanVien.dem)

    def hien_thi_nhan_vien(self):
        print("Tên: ", self.name, ", Lương: ", self.salary)
```

- Biến đếm là class variable được dùng chung cho mọi instances, được sử dụng bởi: NhanVien.dem từ trong hay ngoài class đều được.
- Hàm __init __ () là method đặc biệt, gọi là constructor. Hàm này được Python tự động gọi khi một instance mới được tạo ra.

- Các method khi khai báo thì đều có param (đối số) self, tuy nhiên khi gọi thì không cần truyền giá trị vì Python sẽ tự động truyền instance vào biến self.
- Tạo instance và gọi (truy cập) vào thuộc tính, phương thức theo từng instance:

```
nhan_vien_dev = NhanVien('AnhDK', 1000)
nhan_vien_pm = NhanVien('LinhNM', 1200)
#Truy cap vao method cua Class
nhan_vien_dev.hien_thi_nhan_vien()
nhan_vien_pm.hien_thi_nhan_vien()
#Truy cap vao bien cua Class
print(nhan_vien_dev.dem)
#Truy cap vao thuoc tinh (attribute) cuar Class
print(nhan_vien_pm.name)
print(nhan_vien_dev.name)
```

- Có thể tạo mới, sửa, xoá attributes của class hay instance:

```
#Attribute moi
nhan_vien_pm.age = 28
nhan_vien_pm.new_attribute= 'Không có'
#Cap nhat attribute
nhan_vien_pm.salary = 15000
print(nhan_vien_pm.hien_thi_nhan_vien())
print(nhan_vien_pm.new_attribute)
#Xoa attribute cua íntance
del nhan_vien_pm.new_attribute
```

Truy cập class sử dụng hàm

- getattr(obj, name[, default]): Truy cập thuộc tính name của đối tượng obj, nếu obj này không có thuộc tính đó thì trả về giá trị default. Đối số default khi không được truyền sẽ gây lỗi …<> object has no attribute <>...

```
getattr(nhan_vien_pm, 'salary')
getattr(nhan_vien_pm, 'test')
```

- hasattr(obj, name): Trả về True/False, kiểm tra đối tượng obj có thuộc tính name hay không.

```
print(hasattr(nhan_vien_pm, 'salary')) #trả về True
print(hasattr(nhan_vien_pm, 'test')) #trả về False
```

- setattr(obj,name,value): gán giá trị cho thuộc tính. Hàm này tương đương với phép: obj.name = value

```
setattr(nhan_vien_pm, 'salary', 5500)
```

- delattr(obj, name): Xóa thuộc tính. Tương đương lệnh del obj.name

```
del nhan_vien_pm.new_attribute

delattr(nhan_vien_pm, 'new_attribute')
```

Tham số mặc định trong constructor

- Python cho phép tham số của phương thức khởi tạo có thể có giá trị mặc định.
- Chú ý: *Tất cả các tham số bắt buộc (required parameters) phải đặt trước tất cả các tham số có giá trị mặc định.*

```
class Person :

    # default value: age & gender.
    def __init__ (self, name, age = 1, gender = "Male" ):

        self.name = name
        self.age = age
        self.gender= gender


    def showInfo(self):

        print ("Name: ", self.name)
        print ("Age: ", self.age)
        print ("Gender: ", self.gender)
```

```
from person import Person

# Create Person object.
aimee = Person("Aimee", 21, "Female")
aimee.showInfo()

print (" ---------- ")

# age, gender default.
alice = Person( "Alice" )
alice.showInfo()

print (" ---------- ")
# gender default.
tran = Person("Tran", 37)
tran.showInfo()
```

Bài tập thực hành

Bài 1:

- a) Tạo các class **học viên** với các thuộc tính tương ứng như sau:
  - Học viên (Họ tên, ngày sinh, email, điện thoại, địa chỉ, đơn vị công tác)
- b) Tạo phương thức show_info trả về đầy đủ thông tin của cá nhân. Sau đó gọi phương thức này để in ra thông tin cá nhân của mình.
- c) Tạo phương thức get_info với các tham số mặc định truyền vào: *Địa chỉ = 'Hà Nội', Đơn vị công tác = 'ITPlus'.* Sau đó gọi hàm get_info và in ra thông tin trong 2 trường hợp:
  - TH1: Truyền vào các thông tin cá nhân nhưng không truyền vào giá trị Địa chỉ và Đơn vị công tác.
  - TH2: Truyền vào đầy đủ các thông tin cá nhân.

THANKS FOR ATTENDING!