



BÀI GIẢNG: CLASS & OBJECT TRONG PYTHON

✍️ Tác giả: Đặng Kim Thi



MỤC LỤC

1. Class và Object trong Python
 - 1.1. Khái niệm Class và Object
 - 1.2. Cách tạo Class và Object
 - 1.3. Ví dụ minh họa
 - 1.4. Các thuộc tính và phương thức
2. Các nguyên tắc cơ bản trong OOP
 - 2.1. Tính đóng gói (Encapsulation)
 - 2.2. Tính kế thừa (Inheritance)
 - 2.3. Tính đa hình (Polymorphism)
 - 2.4. Tính trừu tượng (Abstraction)
3. Các phương thức đặc biệt trong Python
 - 3.1. Phương thức `__init__`
 - 3.2. Phương thức `__str__`
 - 3.3. Phương thức `__repr__`
 - 3.4. Phương thức `__del__`
4. Bài tập thực hành
 - 4.1. Bài tập về Class và Object
 - 4.2. Bài tập về kế thừa
 - 4.3. Bài tập về đa hình

◆ 1. CLASS VÀ OBJECT TRONG PYTHON

1.1. Khái niệm Class và Object

- **Class** là một khuôn mẫu để tạo ra các đối tượng (Object), chứa thuộc tính (biến) và phương thức (hàm).
- **Object** là một thể hiện cụ thể của một class.

💡 Ví dụ:

```
class Car:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

car1 = Car("Toyota", "Camry")
print(car1.brand) # Output: Toyota
```

1.2. Cách tạo Class và Object

Cú pháp:

```
class ClassName:
    def __init__(self, parameters):
        self.attribute = parameters
```

◆ 2. CÁC NGUYÊN TẮC CƠ BẢN TRONG OOP

2.1. Tính đóng gói (Encapsulation)

- Hạn chế truy cập trực tiếp vào dữ liệu trong class.

```
class Account:
    def __init__(self, balance):
        self.__balance = balance # Thuộc tính private
```

◆ 3. CÁC PHƯƠNG THỨC ĐẶC BIỆT TRONG PYTHON

3.1. Phương thức `__init__`

- Hàm khởi tạo, tự động chạy khi tạo object.

```
class Person:
    def __init__(self, name):
        self.name = name
```

🏆 4. BÀI TẬP THỰC HÀNH

4.1. Bài tập về Class và Object

Bài 1: Viết một class `Student` có thuộc tính `name`, `age` và phương thức hiển thị thông tin sinh viên.

📌 **Input:**

```
s = Student("Thi", 22)
s.display()
```

📌 **Output:**

Tên: Thi, Tuổi: 22

TỔNG KẾT

- **Class** giúp tổ chức mã nguồn tốt hơn.
- **Object** là thể hiện cụ thể của class.
- **4 nguyên tắc OOP** giúp lập trình hướng đối tượng mạnh mẽ.

 Tác giả: Đặng Kim Thi