[Bài đọc] Lớp và đối tượng

Lớp là gì?

Lớp (Class) là tập hợp tất cả các đối tượng có chung các thuộc tính và hành vi.

Ví dụ: Lớp ô tô có các thuộc tính như màu xe, loại động cơ, hãng sản xuất, loại nhiên liệu tiêu thụ, vận tốc tối đa…Các hành vi của xe có thể kể đến như chuyển động, chở hàng hóa…

Ở thế giới thực, thuộc tính là những tính chất, hình dáng bên ngoài, màu…của đối tượng. Còn hành vi là những chức năng, hành động của đối tượng có thể thực hiện được.

Trong ngôn ngữ lập trình Python, thuộc tính chính là dữ liệu của lớp, còn hành vi là các hàm của lớp. Từ đây ta có thể thấy rằng, lớp trong Python là một khái niệm trừu tượng và gồm có 2 thành phần chính: Dữ liệu và hàm

Cú pháp khai báo lớp

```
class tên_lớp:
    # các biến của lớp

    # Các hàm của lớp
```

Ví dụ: Khai báo lớp ô tô

```
class Car:
    # Liệt kê các thuộc tính của lớp ô tô
    engine = "V8"
    color = "black"
    max_speed = 100
    # Liệt kê các hàm của lớp
    def run(self):
        print("Chuyển động của ô tô")

    def load(self):
        print("Có thể chở hàng")

    def show(self):
        print("Màu của ô tô = ", self.color)
        print("Động cơ ô tô = ", self.engine)
        print("Vận tốc tối đa ô tô = ", self.max_speed)
```

Đối tượng là gì?

Đối tượng là một thể hiện cụ thể nào đó của lớp và nó có đầy đủ các thuộc tính và hành vi của lớp đó. Trái ngược với lớp là khái niệm trừu tượng thì đối tượng là một khái niệm

cụ thể. Ví dụ, khi nói đến lớp ô tô thì chúng ta không thể nói rõ được là ô tô đó màu gì, động cơ loại nào…nhưng khi nói đến đối tượng ô tô cụ thể nào đó (ví dụ ô tô của Nguyễn Nhật Vượng) thì chúng ta có thể nói rõ được là ô tô đó màu gì, dùng động cơ nào, chạy xăng hay dầu…

Khai báo đối tượng: Sau khi xây dựng lớp, chúng ta có thể tạo ra các đối tượng (instance) của lớp. Và sử dụng lớp đã xây dựng như một kiểu dữ liệu mới (kiểu dữ liệu do người dùng tự định nghĩa, để phân biệt với các kiểu dữ liệu có sẵn trong Python).

```
# Tạo đối tượng xe Santafe
santafe = Car()
santafe.color = "White"
santafe.max_speed = 250
santafe.show()

# Tạo đối tượng xe Sonata
sonata = Car()
sonata.color = "Blue"
sonata.max_speed = 200
sonata.engine = 'V6'
sonata.show()
```

Self trong lớp Python

Self là một đối tượng cụ thể của lớp. Sử dụng self chúng ta có thể truy xuất đến các thuộc tính và hàm của lớp.

Self luôn luôn chỉ đến đối tượng hiện tại.

```
class Car:
    # Liệt kê các thuộc tính của lớp ô tô
    engine = "V8"
    color = "black"
    max_speed = 100
    # Liệt kê các hàm của lớp
    def run(self):
        print("Chuyển động của ô tô")

    def load(self):
        print("Có thể chở hàng")

    def show(self):
        print("Màu của ô tô = ", self.color)
        print("Động cơ ô tô = ", self.engine)
        print("Vận tốc tối đa ô tô = ", self.max_speed)
```

Ở ví dụ trên ta thấy rằng, ở hàm show() ta dùng self để truy cập đến các thuộc tính của đối tượng hiện tại.

Khi xây dựng hàm của lớp, self là một tham số mặc định của hàm và nó luôn đứng đầu trong danh sách tham số của hàm, tuy nhiên khi gọi hàm, ta không cần phải truyền giá trị cho tham số self này mà Python truyền tham số này một cách tự động. Chương trình sẽ phát sinh lỗi khi chúng ta cố truyền giá trị cho tham số này.

Lưu ý: Self là một tham số chứ không phải là một từ khóa trong Python, chúng ta có thể dùng bất cứ tên nào khác thay thế self ở vị trí của nó.

```
class Car:
    # Liệt kê các thuộc tính của lớp ô tô
    engine = "V8"
    color = "black"
    max_speed = 100
    # Liệt kê các hàm của lớp
    def run(self):
        print("Chuyển động của ô tô")

    def load(self):
        print("Có thể chở hàng")

    def show(my_self):
        print("Màu của ô tô = ", my_self.color)
        print("Động cơ ô tô = ", my_self.engine)
        print("Vận tốc tối đa ô tô = ", my_self.max_speed)
```

Ở ví dụ trên ta thấy hàm show() sử dụng tham số my_self thay thế cho self ở ví dụ trên, kết quả không thay đổi.

Tổng kết:

Qua bài biết này, chúng ta đã tìm hiểu:

- Lớp là gì
- Cách xây dựng lớp
- Đối tượng là gì
- Cách khai báo đối tượng

[Bài đọc] Constructors in Python

Constructors (Hàm khởi tạo) là hàm được sử dụng để khởi tạo đối tượng của lớp. Chức năng chính của hàm khởi tạo là để khởi tạo các giá trị cho các dữ liệu thành viên của lớp (các thuộc tính) khi tạo đối tượng của lớp. Trong Python phương thức __init__() luôn được tự động gọi khi tạo đối tượng của lớp.

Cú pháp:

```
def __init__(self):
```

```
    # Thân của hàm khởi tạo
```

Phân loại Constructor:

Có 2 loại hàm khởi tạo: Hàm khởi tạo mặc định (không có tham số) và hàm khởi tạo có tham số.

Ví dụ: Định nghĩa hàm khởi tạo mặc định

```
class Car:
    # Liệt kê các thuộc tính của lớp ô tô
    engine = "V8"
    color = "black"
    max_speed = 100

    # Hàm khởi tạo mặc định
    def __init__(self):
        self.color = "default: white color"
        print("Hàm khởi tạo mặc định được gọi khi tạo đối tượng")

# Hàm khởi tạo mặc định được gọi
# khi khởi tạo đối tượng
sonata = Car()
```

Kết quả của đoạn mã:

Hàm khởi tạo mặc định được gọi khi tạo đối tượng

Ví dụ: Định nghĩa làm khởi tạo có tham số

```
class Car:
    # Liệt kê các thuộc tính của lớp ô tô
    engine = "V8"
    color = "black"
    max_speed = 100

    # Hàm khởi tạo mặc định
    def __init__(self, engine, color, max_speed):
        self.engine = engine
        self.color = color
        self.max_speed = max_speed

# Tạo đối tượng bằng hàm khởi tạo có
# tham số
sonata = Car("V8.1", "yellow", 300)
sonata.show()
```

Kết quả của đoạn mã:

Màu của ô tô = yellow

Động cơ ô tô = V8.1

Vận tốc tối đa ô tô = 300

Tổng kết:

Qua bài biết này, chúng ta đã tìm hiểu:

- Hàm khởi tạo, chức năng của hàm khởi tạo
- Cách xây dựng hàm khởi tạo

[Bài đọc] Destructors in Python

Destructors (Hàm hủy) Hàm hủy được gọi khi hủy đối tượng của lớp. Trong Python hàm hủy đóng vai trò không quan trọng như một số ngôn ngữ khác (như C++ chẳng hạn) vì Python có trình dọn rác (garbage collector) sẽ đảm nhiệm chức năng tự hủy đối tượng khi kết thúc chương trình và quản lý bộ nhớ một cách tự động. Hàm __del__() là một hàm hủy trong Python.

Cú pháp:

```
def __del__(self):

 # Thân của hàm hủy
```

Lưu ý: Mọi tham chiếu đến đối tượng cũng được hủy khi đối tượng đó được hủy hoặc khi chương trình kết thúc.

Ví dụ 1: Xây dựng hàm hủy

```
class Car:
    # Liệt kê các thuộc tính của lớp ô tô
    engine = "V8"
    color = "black"
    max_speed = 100

    # Hàm khởi tạo mặc định
    def __init__(self):
        self.color = "default: white color"
        print("Hàm khởi tạo mặc định được gọi khi tạo đối tượng")

    def __del__(self):
        print("Hủy đối tượng")

# Tạo đối tượng
sonata = Car()
del sonata
```

Lưu ý: Hàm hủy được gọi sau khi chương trình kết thúc hoặc tất cả các tham chiếu đến đối tượng đã được hủy.

Tổng kết:

Qua bài biết này, chúng ta đã tìm hiểu:

- Hàm hủy, chức năng của hàm hủy
- Cách xây dựng và sử dụng hàm hủy