Bài: Khai báo thuộc tính lớp trong lập trình hướng đối tượng với Python

Xem bài học trên website để ủng hộ Kteam: Khai báo thuộc tính lớp trong lập trình hướng đối tượng với Python

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Trong bài trước, Kteam đã giới thiệu đến bạn LỚP & ĐỐI TƯỢNG TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG PYTHON, cũng như cách tạo, sử dụng những **thuộc tính** (attribute) và những **phương thức** (method) của đối tượng đó.

Ở bài này, chúng ta sẽ tìm hiểu cách sử dụng và hiểu được cách **Khai báo thuộc tính (biến) trong class** (không thông qua hàm constructor)

Nội dung

Để theo dõi bài này một cách tốt nhất, bạn nên có có những kiến thức cơ bản về Python trong khóa LẬP TRÌNH PYTHON CƠ BẢN

Nếu bạn chưa có thời gian để học hết khóa trên thì hãy đảm bảo đã tìm hiểu những kiến thức sau đây

- BIẾN và CÁC KIỂU DỮ LIỆU CƠ BẢN của Python (Số, chuỗi, List, Tuple, Dict, Set, Range)
- Một số toán tử cơ bản (+, -, *, /, %)
- Khối lệnh điều kiện Khối vòng lặp như VÒNG LẶP FOR, VÒNG LẶP IF)
- HÀM

Và đương nhiên để học tiếp bài sau, bạn phải nắm vững kiến thức ở các bài trước:

- LỚP & ĐỐI TƯỢNG TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG PYTHON

Trong bài nay, Bạn và Kteam sẽ cùng tìm hiểu những nội dụng sau đây:

- Khai báo và sử dụng
- Cập nhật giá trị thuộc tính thông qua lớp
- Cập nhật giá trị thuộc tính thông qua đối tượng
- Sử dụng thuộc tính trong các phương thức

Khai báo và sử dụng

Ở bài trước, các bạn đã biết cách tạo ra các thuộc tính cho một đối tượng, bản chất các thuộc tính đó cũng chỉ là những biến.

Còn bài này, chúng ta sẽ tạo ra các **thuộc tính (các biến)** ở ngay trong lớp mà không cần phải thông qua hàm **constructor**.

Việc khai báo biến trong một lớp nó cũng tương tự như bạn khai báo biến ở trong một hàm hay một block nào đó. Ta sẽ tìm hiểu kĩ qua các ví dụ nhé!

:

```
class SieuNhan:
    suc_manh = 50
    def __init__(self, para_ten, para_vu_khi, para_mau_sac):
        self.ten = para_ten
        self.vu_khi = para_vu_khi
        self.mau_sac = para_mau_sac

sieu_nhan_A = SieuNhan("Sieu nhan do", "Kiem", "Do")
print(SieuNhan.suc_manh)
print(sieu_nhan_A.suc_manh)
```

Kết quả:

:

```
50
50
```

Được rồi, như bạn đã thấy, ta đã khai báo một thuộc tính **suc_manh** ở ngay hẳn trong lớp **SieuNhan**, và dĩ nhiên thuộc tính này không cần phải khai báo gián tiếp qua hàm **constructor**.

Lớp **SieuNhan** dĩ nhiên là có thuộc tính **suc_manh = 50**. Và vì sieu_nhan_A là một đối tượng thuộc lớp **SieuNhan**, nên những thuộc tính ở lớp **SieuNhan** mặc định sẽ được tự động gán cho đối tượng này, vì thế đối tượng này cũng có thuộc tính và cũng như là giá trị như ta khai báo trong "**khuôn mẫu**"

Cập nhật giá trị thuộc tính thông qua lớp

Tiếp nè, ta thử thay đổi một vài tí tẹo để xem điều gì đặc biệt

:

```
class SieuNhan:
    suc_manh = 50
    def __init__(self, para_ten, para_vu_khi, para_mau_sac):
        self.ten = para_ten
        self.vu_khi = para_vu_khi
        self.mau_sac = para_mau_sac

sieu_nhan_A = SieuNhan("Sieu nhan do", "Kiem", "Do")

print(SieuNhan.suc_manh)
print(sieu_nhan_A.suc_manh)

SieuNhan.suc_manh = 40

print(SieuNhan.suc_manh)
print(sieu_nhan_A.suc_manh)
```

Kết quả:

:

```
50
50
40
40
```

Như bạn thấy, rõ ràng ta chỉ thay đổi lại giá trị thuộc tính của lớp, tuy nhiên lại làm ảnh hưởng tới giá trị của đối tượng mặc dù đối tượng này được tạo ra từ lớp **SieuNhan** khi mà lớp **SieuNhan** này có giá trị thuộc tính **suc_manh** là **50**.

Tới đây, bạn có thể đã biết được rằng, khi thay đổi giá trị một thuộc tính được khai báo trong lớp thông qua lớp thì thuộc tính ở toàn bộ đối tượng thuộc lớp đó sẽ được cập nhật lại giá trị mới được thay đổi.

Và hay một cái là, bạn có thể cập nhật giá trị này ngay trong hàm **constructor**.

:

```
class SieuNhan:
    so_thu_tu = 1
    suc_manh = 50
    def __init__(self, para_ten, para_vu_khi, para_mau_sac):
        self.ten = para_ten
        self.vu_khi = para_vu_khi
        self.mau_sac = para_mau_sac
        self.stt = SieuNhan.so_thu_tu
        SieuNhan.so_thu_tu += 1

sieu_nhan_A = SieuNhan("Sieu nhan do", "Kiem", "Do")
sieu_nhan_B = SieuNhan("Sieu nhan vang", "bua", "Vang")

print(sieu_nhan_A.stt)
print(sieu_nhan_B.stt)
print(SieuNhan.so_thu_tu)
```

Kết quả:

:

```
1
2
3
```

Như bạn thấy, thuộc tính **so_thu_tu** được thay đổi qua mỗi lần tạo đối tượng mới vì mỗi lần tạo đối tượng mới là ta lại gọi hàm **constructor**, do đó gián tiếp thay đổi giá trị **so_thu_tu** của lớp.

Ta gán giá trị này cho một thuộc tính của đối tượng đó ngay trong hàm **constructor** chứ không để cho lớp giữ. Vì nếu để cho lớp giữ thì như bạn đã biết, nó sẽ thay đổi chứ không hề giữ nguyên sau các lần tạo đối tượng mới.

Cập nhật giá trị thuộc tính thông qua đối tượng

Ta lại thử thay đổi thêm một tí nào

:

```
class SieuNhan:
    suc_manh = 50
    def __init__(self, para_ten, para_vu_khi, para_mau_sac):
        self.ten = para_ten
        self.vu_khi = para_vu_khi
        self.mau_sac = para_mau_sac

sieu_nhan_A = SieuNhan("Sieu nhan do", "Kiem", "Do")

print(SieuNhan.suc_manh)
print(sieu_nhan_A.suc_manh)

sieu_nhan_A.suc_manh = 40

print(SieuNhan.suc_manh)
print(sieu_nhan_A.suc_manh)
```

Kết quả:

:

```
50
50
50
40
```

Đã có sự khác biệt. Khi bạn thay đổi giá trị thuộc tính của một đối tượng, thì chỉ có đối tượng đó bị thay đổi, còn cái "**khuôn mẫu**" của chúng ta vẫn như vậy. Và dĩ nhiên nếu như có nhiều đối tượng khác nó cũng vẫn sẽ không bị ảnh hưởng chung.

Sử dụng thuộc tính trong các phương thức

Và đương nhiên rồi, như đã nói thì khi bạn khai báo thuộc tính của một đối tượng ở ngay trong "**khuôn mẫu**" luôn thì nó cũng chỉ vẫn là thuộc tính, vì thế bạn vẫn có thể sử dụng nó ở trong các phương thức một cách bình thường như những thuộc tính được khởi tạo ngay trong hàm **constructor**

:

```
class SieuNhan:
    suc_manh = 50
    def __init__(self, para_ten, para_vu_khi, para_mau_sac):
        self.ten = para_ten
        self.vu_khi = para_vu_khi
        self.mau_sac = para_mau_sac
    def xin_chao(self):
        print("Xin chao, ta la", self.ten)
        print("Suc manh cua ta la", self.suc_manh)
sieu_nhan_A = SieuNhan("Sieu nhan do", "Kiem", "Do")

sieu_nhan_A.xin_chao()
```

Kết quả:

:

```
Xin chao, ta la Sieu nhan do
Suc manh cua ta la 50
```

Kết luận

Bài này đã giúp bạn nắm được cách khai báo thuộc tính ngay trong lớp và một số lưu ý về những thuộc tính này

Ở bài tiếp theo, ta sẽ tìm hiểu sâu về NHỮNG PHƯƠNG THỨC CỦA MỘT ĐỐI TƯỢNG.

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**"