Bài: Tạo lớp kế thừa trong lập trình hướng đối tượng với Python

Xem bài học trên website để ủng hộ Kteam: Tạo lớp kế thừa 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

Ở các bài trước, bạn đọc đã được tìm hiểu sơ bộ về một số khái niệm cơ bản của một lớp như việc TẠO RA MỘT LỚP, DỰNG HÀM CONSTRUCTOR, GÁN THUỘC TÍNH CÁC PHƯƠNG THỨC.

Còn ở bài này, bạn đọc sẽ được biết tới cách **kế thừa lớp** chúng ta vừa viết. Nói nôm na đơn giản là ta không phải viết lại một nùi code.

Tại sao lại như vậy? từ từ ta sẽ dần tìm hiểu

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
- KHAI BÁO THUỘC TÍNH LỚP TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG PYTHON
- CÁC PHƯƠNG THỨC LỚP TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG PYTHON

Bạn và Kteam sẽ cùng tìm hiểu những nội dụng sau đây

- Tạo lớp kế thừa
- Kế thừa thuộc tính
- Kế thừa hàm constructor
- Kế thừa phương thức

Tạo lớp kế thừa

Bạn có biết siêu nhân gao chứ? Siêu nhân gao là siêu nhân, tuy nhiên nó cũng không phải siêu nhân. Ý mình là, lớp siêu nhân cũng có thể miêu tả được siêu nhân gao, tuy nhiên một siêu nhân gao còn cần phải có một vài thứ khác để phân biệt với các siêu nhân khác mà ta không thể viết nó chung một lớp với các siêu nhân khác được vì như vậy sẽ thiếu một số đặc điểm riêng.

Vậy ý tưởng của ta là tạo ra một lớp mới là một lớp siêu nhân gao. Lớp này có đặc điểm là một lớp nâng cấp của lớp siêu nhân. Có nghĩa là nó có nhiều thứ mà lớp siêu nhân chưa có.

Trước mắt là chúng ta cứ viết lại một lớp siêu nhân gao có đầy đủ những thứ của lớp siêu nhân có cái đã. Nhưng viết thế nào? Dễ mà, copy paste là được. Như vậy thì chẳng giống lập trình tí nào, vậy nên ở lập trình hướng đối tượng không riêng gì Python, nó cho phép bạn gọi là kế thừa, lấy những gì mà lớp cũ có. Vậy ta muốn tạo lớp siêu nhân gao kế thừa từ lớp siêu nhân thì như thế nào?

Mời bạn đọc xem ví dụ:

:

```
class SieuNhan:
    pass

class SieuNhanGao(SieuNhan):
    pass

gao_do = SieuNhanGao()
print(gao_do)
```

Kết quả:

:

```
<__main__.SieuNhanGao object at 0x0118F630>
```

Kế thừa thuộc tính

Tiếp theo, ta sẽ chỉnh sửa ở lớp siêu nhân một chút xem ở lớp siêu nhân gao có được "hưởng" từ đó hay không. Đơn giản bằng việc tạo một biến ở lớp siêu nhân

:

```
class SieuNhan:
    suc_manh = 50

class SieuNhanGao(SieuNhan):
    pass

gao_do = SieuNhanGao()
print(gao_do.suc_manh)
```

Kết quả:

:

```
50
```

Giả sử như bạn không muốn thừa hưởng giá trị từ lớp ta kế thừa thì phải làm sao? Đơn giản mà, ta viết lại thôi

:

```
class SieuNhan:
    suc_manh = 50

class SieuNhanGao(SieuNhan):
    suc_manh = 40

gao_do = SieuNhanGao()
print(gao_do.suc_manh)
```

Kết quả:

:

```
40
```

Kế thừa hàm constructor

Ta vừa mới thử thừa kế các thuộc tính ở lớp ta thừa kế (lớp cha), giờ ta thử thừa kế hàm **constructor**. Nếu lớp bạn thừa kế có hàm **constructor** thì khi bạn thừa kế lớp đó bạn nghiễm nhiên đã có một hàm **constructor**, và dùng như dùng ở lớp kế thừa. Mời bạn đọc xem ví dụ:

:

```
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

class SieuNhanGao(SieuNhan):
    suc_manh = 40

gao_do = SieuNhanGao("Gao do", "Cung", "Do")
print(gao_do.__dict__)
print(gao_do.suc_manh)
```

Kết quả:

:

```
{'ten': 'Gao do', 'vu_khi': 'Cung', 'mau_sac': 'Do'}
40
```

Vậy câu hỏi đặt ra là ta muốn thêm một số thuộc tính nữa cho lớp siêu nhân gao thì sao? Giả sử ta cũng muốn thêm một thuộc tính nữa là sư thú mà siêu nhân gao đó có khi khởi tạo. Và muốn thay đổi thì ta viết lại thôi.

:

```
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

class SieuNhanGao(SieuNhan):
    suc_manh = 40
    def __init__(self, para_ten, para_vu_khi, para_mau_sac, para_su_thu):
        self.ten = para_ten
        self.vu_khi = para_vu_khi
        self.mau_sac = para_mau_sac
        self.su_thu = para_su_thu

gao_do = SieuNhanGao("Gao do", "Cung", "Do", "Su tu")
print(gao_do.__dict__)
print(gao_do.suc_manh)
```

Kết quả

:

```
{'ten': 'Gao do', 'vu_khi': 'Cung', 'mau_sac': 'Do', 'su_thu': 'Su tu'}
40
```

Bạn thấy rườm rà không nhỉ? Tại sao ta phải ghi lại gần như là hết cả hàm **constructor** như lớp cha. Có cách nào nhanh gọn giúp ta đỡ việc copy paste lại phương thức từ hàm **constructor** lớp cha mà chỉ cần thêm những thuộc tính mới thôi không? Có đấy. Ví dụ sau đây sẽ dùng phương pháp đó và đây là cách người ta hay dùng chứ không phải cách bên trên.

:

```
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

class SieuNhanGao(SieuNhan):
    suc_manh = 40
    def __init__(self, para_ten, para_vu_khi, para_mau_sac, para_su_thu):
        # doi luc nguoi ta cung co the
        # SieuNhan.__init__(para_ten, para_vu_khi, para_mau_sac)
        # nhung dung super van la chu yeu
        super().__init__(para_ten, para_vu_khi, para_mau_sac)
        self.su_thu = para_su_thu

gao_do = SieuNhanGao("Gao do", "Cung", "Do", "Su tu")
print(gao_do.__dict__)
print(gao_do.suc_manh)
```

Kết quả:

:

```
{'ten': 'Gao do', 'vu_khi': 'Cung', 'mau_sac': 'Do', 'su_thu': 'Su tu'}
40
```

> **Lưu ý:** Đây là cách dùng cho hàm super cho Python 3.X, với Python 2.X sẽ có một chút khác biệt. Với ví dụ trên nếu là Python 2 thì bạn sẽ gọi hàm super như sau:
>
> :
>
> ```
> super(SieuNhanGao, self).__init__(para_ten, para_vu_khi, para_mau_sac)
> ```

Kế thừa phương thức

Thế còn kế thừa những phương thức thì sao nhỉ ? Hoàn toàn tương tự như kế thừa thuộc tính. Nếu lớp trước có những phương thức gì, bạn được kế thừa toàn bộ. Nếu bạn muốn thêm thì viết thêm ở lớp kế thừa. Còn nếu muốn chỉnh sửa, thì ta viết lại phương thức đó:

:

```
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)
    def show_suc_manh(self):
        print("Suc manh cua ta la", self.suc_manh)

class SieuNhanGao(SieuNhan):
    suc_manh = 40
    def __init__(self, para_ten, para_vu_khi, para_mau_sac, para_su_thu):
        super().__init__(para_ten, para_vu_khi, para_mau_sac)
        self.su_thu = para_su_thu
    def show_suc_manh(self):
        print("Suc manh cua ta la", self.suc_manh)
        print("Su dung su thu", self.su_thu)

sieu_nhan_A = SieuNhan("Sieu nhan do", "Kiem", "Do")
gao_do = SieuNhanGao("Gao do", "Cung", "Do", "Su tu")

sieu_nhan_A.xin_chao()
gao_do.xin_chao()

sieu_nhan_A.show_suc_manh()
gao_do.show_suc_manh()
```

Kết quả:

:

```
Xin chao, ta la Sieu nhan do
Xin chao, ta la Gao do
Suc manh cua ta la 50
Suc manh cua ta la 40
Su dung su thu Su tu
```

Kết luận

Bài này đã được tìm hiểu về kế thừa trong lập trình hướng đối tượng Python

Ở bài tiếp theo, ta sẽ tìm hiểu về SPECIAL METHOD - một loại phương thức trong lớp của Python nữa ngoài regular method, class method và static method.

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**"