Bài: Lớp và đối tượng trong lập trình hướng đối tượng với Python

Xem bài học trên website để ủng hộ Kteam: Lớp và đối tượng 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 bài này, chúng ta sẽ đến với những khái niệm cơ bản của LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG PYTHON chính là **Lớp (class) và đối tượng (object)**. Trong đối tượng, ta sẽ biết cách tạo, sử dụng những **thuộc tính** (attribute) và những **phương thức** (method) của đối tượng đó.

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

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

- Lớp là gì ?
- Thuộc tính là gì ?
- Hàm constructor (initialize method)
- Phương thức là gì ?

Lớp là gì ?

Nếu hỏi một bạn đã biết về **lập trình hướng đối tượng** (bất kể ngôn ngữ nào) rằng bạn nghĩ tới từ nào đầu tiên khi nói về OOP có lẽ hầu hết câu trả lời nhận được sẽ là: **class** (Hay tiếng Việt là lớp).

Vậy, class là gì? Nói đơn giản nó giống như là một bản mẫu, một khuôn mẫu. Ở đó ta khai báo các **thuộc tính (attribute)** và **phương thức (method)** nhằm miêu tả để từ đó ta tạo ra được những **object (đối tượng)**

> **Lưu ý:** đôi khi object người ta cũng có thể ghi là instance, tuy nó không sát nghĩa cho lắm. Bạn không cần bận tâm lắm đâu vì vào ví dụ ta sẽ hiểu thêm, còn nếu bạn muốn hiểu kĩ thì hãy nghiền ngẫm câu tiếng Anh sau: "**Objects are instances of types. 42 is an instance of the type int is equivalent to 42 is an int object**"

Cú pháp để tạo một lớp

```
class <tên_lớp>:

    # code
```

Giả sử giờ ta tạo một lớp để miêu tả siêu nhân.

:

```
class SieuNhan:
    pass # lệnh giữ chỗ
```

> **Lưu ý:** theo chuẩn PEP8 về đặt tên của lớp (class) thì sẽ được viết theo kiểu CapWords. Bạn có thể theo hoặc không theo, vì đây chỉ là một chuẩn format code Python thôi.

Rồi nào, ta đã có một khuôn mẫu của siêu nhân rồi, cái ta cần là một đối tượng thuộc lớp siêu nhân.

:

```
class SieuNhan:
    pass

sieu_nhan_A = SieuNhan() # sieu_nhan_A chính là một object thuộc lớp SieuNhan
print(sieu_nhan_A)
```

Kết quả:

:

```
<__main__.SieuNhan object at 0x0106CD10>
```

__main__.SieuNhan nghĩa là đây là đối tượng thuộc lớp **SieuNhan** ở hàm main (có nghĩa là ở file ta đang chạy thực thi) kèm theo cái nơi cư trú của nó – thứ mà ta không cần bận tâm lắm lúc này.

Thuộc tính là gì?

Siêu nhân (SN) của ta chưa có thuộc tính gì, ta cần phải giúp SN có thêm một vài thuộc tính. Khi khai báo thuộc tính cho một đối tượng, bạn phải nghĩa ra những thuộc tính để mà giúp ta có thể phân biệt nó với những đối tượng khác cùng lớp, ví dụ như giữa 2 thằng con trai đừng lấy giới tính ra để phân biệt mà nên dùng hơn là sử dụng tên.

Vậy nghĩ tới siêu nhân, ta nghĩ tới cái gì? Tên, vũ khí, màu sắc,...

Bạn đọc xem đoạn code ví dụ dưới đây để biết khai báo thuộc tính **ĐƠN GIẢN** và cách lấy thuộc tính

:

```
class SieuNhan:
    pass

sieu_nhan_A = SieuNhan()

sieu_nhan_A.ten = "Sieu nhan do"
sieu_nhan_A.vu_khi = "Kiem"
sieu_nhan_A.mau_sac = "Do"

print("Ten cua sieu nhan la:",sieu_nhan_A.ten)
print("Sieu nhan mau:", sieu_nhan_A.mau_sac)
print("Su dung vu khi:", sieu_nhan_A.vu_khi)
```

Kết quả:

:

```
Ten cua sieu nhan la: Sieu nhan do
Sieu nhan mau: Do
Su dung vu khi: Kiem
```

> **Lưu ý** là thuộc tính nào có mới lấy ra được nhé, chứ cái class của chúng ta không tự động sinh ra thuộc tính đâu

:

```
print("Chi so suc manh: ", sieu_nhan_A.suc_manh)
```

Kết quả:

:

```
AttributeError: 'SieuNhan' object has no attribute 'suc_manh'
```

Hàm constructor (initialize method)

Mở rộng vấn đề, ta cần khai báo khoảng 1000 siêu nhân. Giải sử một siêu nhân có 3 thuộc tính như trên vị chi ta sẽ mất 3000 dòng khai báo. Tuy là bạn vẫn có thể khai báo chỉ bằng 1000 dòng bằng cách khai báo one-liner của Python tuy nhiên đôi lúc những thuộc tính của đối tượng không dễ dàng để khai báo một cách đơn giản như vậy.

Ta cần phải cần một cái khuôn mẫu mà chỉ cần đưa các giá trị thuộc tính vào còn việc gán giá trị thì để cho khuôn mẫu làm. Dĩ nhiên khuôn mẫu có thể, nếu ta xây dựng cho nó một hàm **constructor**.

:

```
class SieuNhan:
    def __init__(self):
        pass
```

> **Lưu ý:** 2 dấu gạch "_" bắt đầu và kết thúc

Giải thích một vài điều, đây là tên hàm được quy ước, nếu bạn đặt tên hàm như vậy, bạn mặc định nói với chương trình rằng đây là **constructor** (nó là gì thì bạn từ từ sẽ biết). Trong Python, một số hàm trong lớp sẽ được tự động gọi khi ta khai báo một đối tượng và constructor là một trong số những hàm đó.

Từ khóa **self** hay cụ thể ở đây là **parameter self** là một quy ước (lưu ý là hoàn toàn sẽ không bị bắt lỗi cú pháp nếu dùng từ khóa khác), bạn có thể dùng một từ khóa khác. Tuy nhiên từ trước tới giờ mình chưa thấy ai dùng một tứ khóa khác ngoài **self**. Nếu bạn không muốn gây hiểu lầm cho người khác thậm chí khiến người khác nghĩ là bạn viết code sai thì bạn nên sử dụng từ khóa **self**.

Vậy, từ khóa **self** là gì? Không ngẫu nhiên mà người ta lại lấy từ self. Ý nghĩa của nó là chính đối tượng đó. Hơi khó hiểu nhỉ? Coi ví dụ đã, bạn sẽ dần tự hiểu ra từ khóa này.

:

```
class SieuNhan:
    def __init__(self, para_ten, para_vu_khi, para_mau_sac):
        self.ten = "Sieu nhan " + para_ten
        self.vu_khi = para_vu_khi
        self.mau_sac = para_mau_sac

sieu_nhan_A = SieuNhan("do", "Kiem", "Do")
```

Đầu tiên, từ khóa **self** sẽ nhận giá trị chính là đối tượng đã gọi hàm đó. Ủa? Hàm **__init__** có đối tượng nào gọi đâu? Đương nhiên là không cần gọi, nó đã được tự động gọi khi bạn khởi tạo đối tượng rồi, có nghĩa là khi bạn dung lớp **SieuNhan** khởi tạo ra đối tượng **sieu_nhan_A** mặc định bạn đã kêu đối tượng **sieu_nhan_A** gọi hàm **__init__**. Và đương nhiên, **self** được gán bằng đối tượng **sieu_nhan_A**, các **argument** "**do**", "**Kiem**", "**Do**" còn lại sẽ được truyền vào theo tứ tự. Bạn hãy thử xem lại cách thủ công khai báo thuộc tính lúc ban đầu bạn sẽ thấy nó tương tự.

Bạn nên nhớ rằng mỗi khi có một đối tượng nào đó gọi một hàm thì luôn luôn tối thiểu sẽ có một argument được gửi vào hàm đó chính là chính đối tượng đó, nếu hàm đó không có parameter nhận thì sẽ sinh lỗi, còn nếu dư argument (vì ta không lường trước được có một argument là chính đối tượng được ngầm gửi vào) thì vẫn sẽ có lỗi tràn argument. Còn nếu mà gửi vào vẫn không có lỗi thì...Bug này nặng khó fix đây.

Khi ta thử in ra các thuộc tính

:

```
print("Ten cua sieu nhan la:",sieu_nhan_A.ten)
print("Sieu nhan mau:", sieu_nhan_A.mau_sac)
print("Su dung vu khi:", sieu_nhan_A.vu_khi)
```

Kết quả :

:

```
Ten cua sieu nhan la: Sieu nhan do
Sieu nhan mau: Do
Su dung vu khi: Kiem
```

Phương thức là gì?

Giờ ta thử giúp **SN** của chúng ta có một câu giới thiệu. Ta tạo một hàm để làm điều đó.

:

```
class SieuNhan:
    def __init__(self, para_ten, para_vu_khi, para_mau_sac):
        self.ten = "Sieu nhan " + para_ten
        self.vu_khi = para_vu_khi
        self.mau_sac = para_mau_sac
    def xin_chao(self):
        return "Xin chao, ta chinh la " + self.ten

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

print(sieu_nhan_A.xin_chao()) # vì nó là hàm nên nhớ là hãy thêm () để gọi hàm
print(SieuNhan.xin_chao(sieu_nhan_A)) # một cách gọi khác nhưng rất không phổ biến
```

Kết quả:

:

```
Xin chao, ta chinh la Sieu nhan do
Xin chao, ta chinh la Sieu nhan do
```

Thông thường, khi nói tới hàm của lớp, người ta hay gọi là phương thức (method). Khi nói tới hàm thì nó là một chương trình bé bé chờ bạn thực thi, còn khi gọi nó là phương thức thì nó là hàm nhưng liên quan tới lớp thôi. Người ta thường hay **« call function », « invoke method »**. Nhớ đừng nhầm lẫn nhé, người ta cười cho đấy.

Kết luận

Bài này đã giúp bạn nắm được những khái niệm cơ bản của lớp và đối tượng trong Python

Ở bài tiếp theo, ta sẽ tìm hiểu về KHAI BÁO THUỘC TÍNH TRONG CLASS.

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**"