# Bài: Các phương thức đặc biệt trong lập trình hướng đối tượng Python

*Xem bài học trên website để ủng hộ Kteam: Các phương thức đặc biệt trong lập trình hướng đối tượng 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 tìm hiểu về TÍNH KẾ THỪA TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG PYTHON.

Còn ở bài này, bạn đọc sẽ biết tới một số những **phương thức đặc biệt**. Tổng quát thì nó gần giống như một **regular method** nhưng nó đặc biệt hơn một vài chỗ.

---

## 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
- TẠO LỚP KẾ THỪA TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG VỚI PYTHON

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

- Giới thiệu chung về phương thức đặc biệt
- Giới thiệu một số phương thức đặc biệt

---

## Giới thiệu chung về phương thức đặc biệt

**Phương thức đặc biệt**, tiếng Anh là **Special method**, đôi lúc người ta còn gọi là **Magic method** hoặc **Dunder method**. Những phương thức này đã được quy ước sẵn tên. Định dạng chung của các phức thức này là **__tên phương thức __**. Và chúng ta đã tìm hiểu một **special method** rồi đấy. Nếu bạn đọc chưa nhớ ra thì đó chính là hàm **constructor** của chúng ta. Nó cũng là một **special method**.

Có rất nhiều **special method**, mỗi **special method** lại có một công dụng khác nhau, việc chúng ta là dựa theo khuôn mẫu đó để viết theo ý chúng ta. Nếu bạn đọc còn chưa mường tượng được vấn đề thì không sao cả. Chúng ta sẽ đến với một số ví dụ để hiểu hơn

---

## Giới thiệu một số phương thức đặc biệt

Trước tiên là ta tạo một **class** đơn giản kèm theo một đối tượng thuộc lớp 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

SN_A = SieuNhan('Sieu nhan Do', 'Kiem', 'Do')
```

Còn nhớ trong bài đầu LỚP & ĐỐI TƯỢNG, khi ta thử in đối tượng, nó sẽ ra một dòng mới nhìn vào là thấy chẳng có thông tin gì hữu ích cho ta:

:

```
print(SN_A)
```

Kết quả:

:

```
<__main__.SieuNhan object at 0x00BCA4D0>
```

Nếu bạn muốn khi in ra mà ta có một miêu tả rõ ràng về thứ này là gì, thì ta sẽ nên sử dụng hàm **__str__**. Bạn đọc xem ví dụ sau đây, ta sẽ viết thêm cho lớp một phương thức nữa:

:

```
def __str__(self):
    return 'Day la {}, su dung {}'.format(self.ten, self.vu_khi)
```

Thì khi in ra, hàm **__str__** này được gọi đến và trả về kết quả như bạn muốn

Kết quả:

:

```
Day la Sieu nhan Do, su dung Kiem
```

Phương thức **__str__** này còn có một anh em họ nữa là phương thức **__repr__**

:

```
def __repr__(self):
    return 'ten: {}\nvu khi: {}\nmau sac: {}'.format(self.ten, self.vu_khi, self.mau_sac)
```

Tuy nhiên các bạn lưu ý dùm mình, khi dùng hàm **print**. Nếu như lớp của bạn cùng có cả 2 phương thức **__str__** và **__repr__** thì hàm **print** ưu tiên dùng **__str__** hơn. Còn trên **interactive prompt** khi không dùng hàm **print** thì sẽ ưu tiên **__repr__** hơn. Tuy nhiên ta vẫn có cách để gọi **__repr__** nếu cần (ngoài cách gọi trực tiếp đối tượng**.__repr__()**). Vì mục đích của **__repr__** cho thông tin chi tiết cụ thể về đối tượng, còn **__str__** chỉ đơn giản là giá trị. Ta nghĩ đơn giản như ta là một đối tượng, mỗi khi ta gọi hàm **__str__** thì nó sẽ trả về hình ảnh bề da thịt của cơ thể chúng ta, còn nếu gọi **__repr__** thì trả về hình ảnh xương cốt, gân, cơ bắp của cơ thể chúng ta.

Đó là một quy ước ngầm, đương nhiên bạn vẫn có thể phá cách theo cách bạn muốn nhưng để có thể làm việc nhóm tốt, để người khác dễ dàng xử lí code của bạn thì nên theo những quy ước mà hầu hết mọi người đều theo.

:

```
print(SN_A)
print('%s' %SN_A)
print('%r' %SN_A)
```

Kết quả:

:

```
Day la Sieu nhan Do, su dung Kiem
Day la Sieu nhan Do, su dung Kiem
ten: Sieu nhan Do
vu khi: Kiem
mau sac: Do
```

Bạn còn nhớ hàm **len** chứ? Bản chất hàm **len** cũng là một **special method**. Bạn đọc xem ví dụ sau đây:

:

```
s = 'How Kteam'
print(len(s))
print(s.__len__())
```

Kết quả:

:

```
9
9
```

Và đương nhiên rồi, bạn hoàn toàn có thể tự tạo riêng cho lớp của mình một **special method** để có thể sử dụng được hàm **len** (mặc định thì chưa được)

:

```
def __len__(self):
    return len(self.ten)
```

Kết quả:

:

```
12
```

Trước khi ta tới một **special method** nữa, thì ta hãy tạo thêm một đối tượng siêu nhân nữa:

:

```
SN_B = SieuNhan('Sieu nhan Xanh', 'Cung', 'Xanh')
```

Bạn còn nhớ toán tử **+** của kiểu dữ liệu số hoặc chuỗi, list chứ? Bản chất các toán tử cũng là những **special method**.

:

```
print(2 + 3)
print(int.__add__(2, 3))
print('How ' + 'Kteam')
print(str.__add__('How ', 'Kteam'))
print([1, 2] + [3, 4])
print(list.__add__([1, 2], [3, 4]))
```

Kết quả:

:

```
5
5
How Kteam
How Kteam
[1, 2, 3, 4]
[1, 2, 3, 4]
```

Và dĩ nhiên, bạn cũng hoàn toàn có thể tạo riêng cho mình một toán tử, chẳng hạn + như ví dụ trên chẳng hạn.

:

```
def __add__(self, mot_sieu_nhan_khac):
    return self.suc_manh + mot_sieu_nhan_khac.suc_manh
```

Ta gọi phương thức bằng toán tử cũng như gọi trực tiếp phương thức:

:

```
print(SN_A + SN_B)
print(SieuNhan.__add__(SN_A, SN_B))
```

Kết quả:

:

```
100
100
```

Còn rất nhiều những special method, bạn đọc có thể tham khảo thêm tại:

> SPECIAL METHOD

## Kết luận

Bài này đã giúp bạn đọc tìm hiểu về những phương thức đặc biệt trong một lớp

Ở bài tiếp theo, ta sẽ tìm hiểu ba thuật ngữ quen thuộc của lập trình hướng đối tượng là GETTER, SETTER, DELETER.

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**"