TÁC GIẢ: 8 SYNC

CỘNG ĐỒNG

- Kevin Nguyễn
- Nhóm Chia Sẻ Công Nghệ
- Nhóm BlockChain
- Tiktok: 8 Sync
- Youtube: 8 Sync Dev
- Zalo

KHÓA HỌC:

- Fullstack Python
- Fullstack Nextjs
- Fullstack Android-IOS

Tài liệu sẽ được cập nhật định kì và thông báo trong group nên các bạn chú ý nhen .

BÀI 7. PYTHON - CHÚ THÍCH HÀM

Tính năng chú thích hàm của Python cho phép bạn thêm các siêu dữ liệu bổ sung về các đối số được khai báo trong định nghĩa hàm và cũng kiểu dữ liệu trả về.

Mặc dù bạn có thể sử dụng tính năng docstring của Python để tài liệu hóa một hàm, nhưng nó có thể trở nên lỗi thời nếu có các thay đổi trong nguyên mẫu của

hàm. Do đó, tính năng chú thích đã được giới thiệu trong Python là kết quả của PEP 3107.

Các chú thích không được Python thông dịch viên xem xét khi thực thi hàm. Chúng chủ yếu dành cho các IDE Python để cung cấp tài liệu chi tiết cho người lập trình.

Chú thích là bất kỳ biểu thức Python hợp lệ nào được thêm vào các đối số hoặc kiểu dữ liệu trả về. Ví dụ đơn giản nhất của chú thích là quy định kiểu dữ liệu của các đối số. Chú thích được đề cập như là một biểu thức sau khi đặt dấu hai chấm trước đối số.


```
def myfunction(a: int, b: int):
    c = a + b
    return c
```

Hãy nhớ rằng Python là một ngôn ngữ được gán động, và không thực hiện bất kỳ kiểm tra kiểu nào tại thời gian chạy. Do đó, việc chú thích các đối số với các kiểu dữ liệu không có bất kỳ hiệu ứng nào khi gọi hàm. Ngay cả khi có các đối số không phải là số nguyên được cung cấp, Python cũng không phát hiện ra lỗi nào.


```
def myfunction(a: int, b: int):
    c = a + b
    return c

print(myfunction(10, 20))
print(myfunction("Hello ", "Python"))
```

Nó sẽ tạo ra đầu ra sau:

```
30
Hello Python
```

CHÚ THÍCH HÀM VỚI KIỂU TRẢ VỀ

Chú thích được bỏ qua vào thời điểm chạy, nhưng hữu ích cho các IDE và các thư viện kiểm tra kiểu tĩnh như mypy.

Bạn cũng có thể đưa ra chú thích cho kiểu dữ liệu trả về. Sau dấu ngoặc đơn và trước dấu hai chấm, đặt một mũi tên (->) theo sau là chú thích. Ví dụ:


```
def myfunction(a: int, b: int) -> int:
    c = a + b
    return c
```

CHÚ THÍCH HÀM VỚI BIỂU THỨC

Vì việc sử dụng kiểu dữ liệu làm chú thích bị bỏ qua vào thời điểm chạy, bạn có thể đặt bất kỳ biểu thức nào làm chú thích để làm siêu dữ liệu cho các đối số. Do đó, hàm có thể có bất kỳ biểu thức tùy ý nào làm chú thích như trong ví dụ sau:


```
def total(x: 'marks in Physics', y: 'marks in Chemistry'):
    return x + y
```

CHÚ THÍCH HÀM VỚI ĐỐI SỐ MẶC ĐỊNH

Nếu bạn muốn chỉ định một đối số mặc định cùng với chú thích, bạn cần đặt nó sau biểu thức chú thích. Đối số mặc định phải đặt sau các đối số bắt buộc trong danh sách đối số.

< 8 Sync Dev />

```
def myfunction(a: "Physics", b: "Maths" = 20) -> int:
    c = a + b
    return c

print(myfunction(10))
```

THUỘC TÍNH ANNOTATIONS CỦA HÀM

Hàm trong Python cũng là một đối tượng, và một trong những thuộc tính của nó là **annotations**. Bạn có thể kiểm tra bằng hàm dir().

< 8 Sync Dev />

```
print(dir(myfunction))
```

Điều này sẽ in ra danh sách đối tượng myfunction chứa **annotations** như một trong những thuộc tính.

< 8 Sync Dev />

```
['__annotations__', '__call__', '__class__', '__closure__',
'__code__', '__defaults__', '__delattr__', '__dict__',
'__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__get__', '__getattribute__', '__globals__', '__gt__',
'__hash__', '__init__', '__init_subclass__', '__kwdefaults__',
'__le__', '__lt__', '__module__', '__name__', '__ne__',
'__new__', '__qualname__', '__reduce__', '__reduce_ex__',
```

```
'__repr__', '__setattr__', '__sizeof__', '__str__',
'__subclasshook__']
```

Thuộc tính **annotations** chính là một từ điển trong đó các đối số là các khóa và các chú thích là các giá trị

của chúng.

< 8 Sync Dev />

```
def myfunction(a: "Physics", b: "Maths" = 20) -> int:
   c = a + b
   return c

print(myfunction.__annotations__)
```

Nó sẽ tạo ra đầu ra sau:

< 8 Sync Dev />

```
{'a': 'Physics', 'b': 'Maths', 'return': <class 'int'>}
```

Bạn có thể có các đối số vị trí tùy ý và/hoặc các đối số từ khóa tùy ý cho một hàm. Chú thích cũng có thể được đưa ra cho chúng.

< 8 Sync Dev />

```
def myfunction(*args: "arbitrary args", **kwargs: "arbitrary
keyword args") -> int:
   pass

print(myfunction.__annotations__)
```

Nó sẽ tạo ra đầu ra sau:

< 8 Sync Dev />

```
{'args': 'arbitrary args', 'kwargs': 'arbitrary keyword args',
'return': <class 'int'>}
```

Trong trường hợp bạn cần cung cấp nhiều hơn một biểu thức chú thích cho một đối số hàm, đặt nó dưới dạng một đối tượng từ điển phía trước đối số đó.

< 8 Sync Dev />

```
def division(num: dict(type=float, msg='numerator'), den:
dict(type=float, msg='denominator')) -> float:
    return num / den

print(division.__annotations__)
```

Nó sẽ tạo ra đầu ra sau:

< 8 Sync Dev />

```
{'num': {'type': <class 'float'>, 'msg': 'numerator'}, 'den':
{'type': <class 'float'>, 'msg': 'denominator'}, 'return':
<class 'float'>}
```