Bài 29: KIỂU DỮ LIỆU FUNCTION TRONG PYTHON - POSITIONAL VÀ KEYWORD ARGUMENT

Xem bài học trên website để ủng hộ Kteam: Kiểu dữ liệu Function trong Python - Positional và keyword argument

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 KIỂU DỮ LIỆU FUNCTION TRONG PYTHON.

Và ở bài này Kteam sẽ lại tìm hiểu với các bạn **Kiểu dữ liệu Function trong Python - Positional và keyword argument**.

Nội dung

Để đọc hiểu bài này tốt nhất bạn cần:

- Cài đặt sẵn MÔI TRƯỜNG PHÁT TRIỂN CỦA PYTHON.
- Xem qua bài CÁCH CHẠY CHƯƠNG TRÌNH PYTHON.
- Nắm CÁCH GHI CHÚ và BIẾN TRONG PYTHON.
- CÁC KIỂU DỮ LIỆU ĐƯỢC GIỚI THIỆU TRONG PYTHON
- CÂU ĐIỀU KIỆN IF TRONG PYTHON
- VÒNG LẶP WHILE và VÒNG LẶP FOR TRONG PYTHON
- NHẬP XUẤT TRONG PYTHON

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

- Positional argument và keyword argument
- Bắt buộc (force) positional argument và keyword argument

Positional argument và keyword argument

Với một hàm thông thường như sau

```
>>> def kteam(a, b):
...     pass # lệnh giữ chỗ.
...
```

Thì ta có thể **pass argument** vào cho hàm như sau

```
>>> kteam(3, 'Free Education')
```

Trong ví dụ trên, hai giá trị là số 3 và chuỗi 'Free Education' gọi là positional argument.

Còn với trường hợp dưới đây

```
>>> kteam(a=3, b='Free Education')
```

Thì hai giá trị trên (chính là số 3 và chuỗi 'Free Education') là những keyword argument.

Sau đây là những điều tuy nhỏ nhưng bạn cần phải biết. Khi **pass argument** theo positional argument. Thì các argument sẽ được gán LẦN LƯỢT cho các parameter. Riêng đối với keyword argument. Bạn đã tự mình gán giá trị cho các parameter. Vậy nên:

```
>>> def kteam(a, b):
...     print('a', a)
...     print('b', b)
...
>>> kteam(a=3,b=4)
a 3
b 4
>>> kteam(b=3,a=4)
a 4
b 3
```

Hai cách gọi hàm trên đều tương tự như nhau.

Một điều nữa là bạn không được phép để **positional** theo sau (follow) keyword.

Có nghĩa là bạn có thể **pass argument** vừa **positional** và keyword cùng một lúc được, nhưng những positional buộc phải đứng trước keyword.

Trường hợp ngớ ngẩn của Tèo sau đây sẽ cho bạn biết điều đó:

```
>>> def teo_with_sone(name, verb):
...     print('Teo', verb + 's', name)
...
>>> teo_with_sone('Python', 'love')
Teo loves Python
>>> teo_with_sone('HTML', verb='like')
Teo likes HTML
>>> teo_with_sone(verb='like', 'CSharp')
  File "<stdin>", line 1
SyntaxError: positional argument follows keyword argument
>>> teo_with_sone(name='Java', 'hate')
  File "<stdin>", line 1
```

```
SyntaxError: positional argument follows keyword argument
```

Bắt buộc (force) Positional argument và keyword argument

Keyword argument

Trong Python, có một số hàm bắt chúng ta buộc phải **pass argument** một cách rõ ràng rành mạch như hàm **sorted**.

```
>>> sorted([3, 4, 1], reverse=True)
[4, 3, 1]

>>> sorted([3, 4, 1], True)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must use keyword argument for key function
```

Bạn thấy đấy, ta không thể **pass argument** cho parameter reverse theo positional argument.

Việc thiết kế này cũng rất tiện lợi vì nhiều trường hợp nhiều parameter cùng một lúc đều có default argument value. Hãy xem vấn đề sau đây mà Tèo mắc phải.

Tèo có một hàm

```
>>> def Teo(a, b=2, c=3, d=4):
...     f = (a + d) * (b + c)
...     print(f)
...
>>> Teo(1)
25
>>> Teo(1, 2, 3, 5)
```

```
30
```

Tèo gọi hàm thì thấy kết quả theo đúng ý mình. Giờ Tèo muốn đổi giá trị của parameter **d** thành 5. Nên Tèo phải truyền lại các giá trị **2** và **3** cho các parameter **b** và **c**.

Vậy, ta có cách nào không phải truyền lại hai giá trị cho parameter b và c không?

Có, chính là **keyword argument**.

```
>>> Teo(1, d=5)
30
```

Đôi lúc, chúng ta nên sử dụng **keyword argument** để tiện lợi và rõ ràng.

Python cho phép chúng ta tạo ra các parameter chỉ nhận giá trị bằng việc pass argument theo kiểu keyword argument.

Cú pháp

```
def function (*, key_arg1, key_arg2):

# function-block
```

Khi tạo một hàm mà có một parameter *. Thì Python sẽ hiểu đó không phải là parameter mà chính là syntax để rồi nó biến các parameter sau * thành các keyword only argument (chỉ nhận giá trị theo kiểu keyword argument)

Ví dụ là dễ hiểu nhất!

```
>>> def kteam(pos_or_key_arg, *, key_arg1, key_arg2):
...     print(pos_or_key_arg)
...     print(key_arg1)
...     print(key_arg2)
...
```

```
>>> kteam(1, key_arg1=2, key_arg2='Kteam')
1
2
Kteam
>>> kteam(1, 2, key_arg2='Kteam')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: kteam() takes 1 positional argument but 2 positional arguments (and 1 keyword-only argument) were given

>>> kteam(1, 2, 'Kteam')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: kteam() takes 1 positional argument but 3 were given
```

Lưu ý: ta có thể thay thế dấu * bằng *identifier. Tuy nhiên phổ biến vẫn là *.

Positional argument

Bạn còn nhớ hàm input chứ? Kteam từng nói với bạn rằng cú pháp của hàm input có thể được viết như thế này.

```
input(prompt=None, /)
```

Dấu **/** chính là một syntax để force parameter prompt trở thành positional only argument. Có nghĩa là bạn chỉ có thể pass argument cho parameter prompt theo kiểu **positional**. Chính xác thì dấu **/** sẽ biến các parameter đứng trước nó thành positional only argument

Tuy nhiên Kteam sẽ không đi sâu vào positional argument vì ở phiên bản Python 3.6.X trở đi không hỗ trợ positional only argument.

Lưu ý: 3.6.X trở đi không hỗ trợ không có nghĩa những bạn cũ hơn một chút xíu như 3.5, 3.4 có hỗ trợ.

Củng cố bài học

Câu hỏi củng cố

Câu hỏi: Dùng hàm help để xem cú pháp của hàm sorted? Sau đó cho biết parameter nào là positional only? Parameter nào là keyword only?

Đáp án của phần này sẽ được trình bày ở bài tiếp theo. Tuy nhiên, Kteam khuyến khích bạn tự trả lời các câu hỏi để củng cố kiến thức cũng như thực hành một cách tốt nhất!

Kết luận

Qua bài viết này, Bạn đã biết một chút về hàm trong Python qua các khái niệm positional argument và keyword argument.

Ở bài viết sau. Kteam sẽ tiếp tục giới thiệu thêm với các bạn về HÀM TRONG PYTHON (FUNCTION).

Cảm ơn 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".**