Bài 32: KIỂU DỮ LIỆU FUNCTION TRONG PYTHON - RETURN

Xem bài học trên website để ủng hộ Kteam: Kiểu dữ liệu Function trong Python - Return

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 – BIẾN LOCALS & GLOBALS.

Và ở bài này Kteam sẽ lại tìm hiểu với các bạn **KIỂU DỮ LIỆU FUNCTION TRONG PYTHON – Return**.

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

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

- Vấn đề
- Giới thiệu lệnh return
- Dùng return để trả về nhiều giá trị một lúc

Vấn đề

Giả sử bạn viết một hàm xử lí một công việc và bạn muốn sao lưu kết quả sau khi xử lí xong ra một biến. Nhưng bạn lại không thể làm điều đó. Vì nếu tạo ra một biến và lưu ngay trong hàm thì như ta đã biết, nó không thể sử dụng được ở mức toàn chương trình (**global**).

Giá mà bạn có thể ném cái dữ liệu sau khi xử lí xong ra ngoài nhỉ?

Có đấy, hầu như mọi ngôn ngữ lập trình bây giờ điều cho phép làm điều đó và đương nhiên Python không phải ngoại lệ.

Giới thiệu lệnh return

Đây là lệnh chỉ sử dụng được ở trong hàm (nếu sử dụng ở ngoài hàm sẽ có nhắc lỗi)

```
SyntaxError: 'return' outside function
```

Lệnh return có cú pháp như sau

```
return [object]
```

Ở đây, **object** là một đối tượng bất kì của một lớp nào đó, có thể là số (number), chuỗi (string), list, tuple, hàm (sẽ biết rõ hơn khi tìm hiểu decorator), lớp (class) hoặc thậm chí là bỏ trống – trường hợp bỏ trống thì **object** return về được tính là **None**.

Khi **return** được gọi, hàm được kết thúc và kết quả được trả ra ngoài. Kết quả trả ra ngoài nên được đưa cho một biến nào đó hứng, nếu không thì coi như bạn gọi hàm không để làm gì.

```
def cal_rec_per(width, height):
    per = (width + height) * 2
    return per

rec_1_width = 5
rec_1_height = 3

# khởi tạo một biến để hứng kết quả
rec_1_per = cal_rec_per(rec_1_width, rec_1_height)

print(rec_1_per)

# trường hợp này là khi bạn không cần tái sử dụng nó ở lần sau
print(cal_rec_per(7, 4))
```


```
def _return_ter_func():
    print('chúng ta sử dụng return để ngắt hàm')
    # dòng dưới đây tương tự như bạn viết return None
    return
    print('Hàm print này dĩ nhiên không được gọi')

none = _return_ter_func()
print(type(none))
```

```python
def _return_ter_func():
    print('chúng ta sử dụng return để ngắt hàm')
    # dòng dưới đây tương tự như bạn viết return None
    return
    print('Hàm print này dĩ nhiên không được gọi')

none = _return_ter_func()
print(type(none))

```

```
chúng ta sử dụng return để ngắt hàm
<class 'NoneType'>
[Finished in 0.4s]
```

Dùng return để trả về nhiều giá trị một lúc

Với Python, việc bạn có thể return nhiều giá trị một lúc bản chất nó không nằm ở câu lệnh Python, mà là do Python thiết kế đặc biệt để có thể **unpack** các object trả về. Bạn hãy xem ví dụ về khai báo sau đây

```
>>> one, two, three = 'how', 'Kteam', 69
>>> one
'how'
>>> two
'Kteam'
>>> three
69
>>> h, o, w = ('k', 'team', 96) # ở đây, cũng có thể sử dụng list hoặc một container bất kì
>>> h, o, w
('k', 'team', 96)
```

Tận dụng điều trên, ta có thể "**return nhiều giá trị cùng một lúc**"

```
def cal_rec_area_per(width, height):
    perimeter = (width + height) * 2
    area = width * height
    return perimeter, area
```

```
rec_width = 3
rec_height = 9
rec_per, rec_area = cal_rec_area_per(rec_width, rec_height)

print(rec_per, rec_area)
```

```
def cal_rec_area_per(width, height):
    perimeter = (width + height) * 2
    area = width * height
    return perimeter, area

rec_width = 3
rec_height = 9
rec_per, rec_area = cal_rec_area_per(rec_width, rec_height)

print(rec_per, rec_area)
```

```
24 27
[Finished in 0.2s]
```

Câu hỏi củng cố

1. Như các bạn đã biết khái niệm hàm số, với hàm số **y = f(x)** thì đồ thị hàm số **y = f(x)** đi qua điểm **M(x_0, y_0)** nếu như **y_0 = f(x_0)**.

Cho một list, mỗi phần tử là một tuple gồm hoành độ (x_0) và tung độ (y_0), kiểm tra xem đồ thị hàm số **y = f(x)** có đi qua điểm đó hay không. Nếu có thì đưa sang list A, trường hợp không thì đưa phần tử đó sang list B.

Sau khi kết thúc, tính tổng các tung độ (y_0) của hai list A và B rồi in ra trị tuyệt đối của hiệu tổng tung độ hai list đó.

Ví dụ: với hàm y = f(x) như sau

$$y = x^3 + 2x^2 - 4x + 1$$

Và một List các điểm

```
[(-5, -20), (-4, -15), (-3, 4), (-2, 9), (-1, 7), (0, 1)
, (1, -7), (2, -9), (4, 81), (5, 130)]
```

Thì kết quả in ra là **21**

2. Cho 5 biến với giá trị mỗi biến là một số tự tự nhiên, gọi **m** là giá trị lớn nhất trong 5 số đó. In ra màn hình 2m -1

Ví dụ: Với 5 biến như sau, kết quả in ra sẽ là 117

```
a = 32
b = 59
c = 8
d = 24
e = 15
```

> **Lưu ý:** Không dùng các hàm tìm **min max** hỗ trợ bởi thư viện, chương trình có sẵn, không sử dụng bất kì container nào. Và chương trình không quá 3 câu lệnh điều kiện (if hoặc elif hoặc else)

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 về lệnh return trong hàm.

Ở bài tiếp theo, Kteam sẽ nói đến một câu lệnh nữa có cách sử dụng rất giống return nhưng phức tạp rất nhiều - KIỂU DỮ LIỆU FUNCTION TRONG PYTHON – YIELD

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**".