Bài 24: CẤU TRÚC RẼ NHÁNH TRONG PYTHON

Xem bài học trên website để ủng hộ Kteam: Cấu trúc rẽ nhánh trong 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

Trong bài trước, Kteam đã giới thiệu đến bạn KIỂU DỮ LIỆU BOOLEAN TRONG PYTHON.

Ở bài này Kteam sẽ giới thiệu với các bạn **Cấu trúc rẽ nhánh trong Python – Câu lệnh IF**. Một câu lệnh thường xuyên được sử dụng trong các chương trình.

Nội dung

Để đọc hiểu bài này tốt nhất bạn cần:

- Cài đặt sẵn MÔI TRƯỜNG PHÁT TRIỂN CỦA PYTHON.
- Xem qua bài CÁCH CHAY CHƯƠNG TRÌNH PYTHON.
- Nắm CÁCH GHI CHÚ và BIẾN TRONG PYTHON.
- CÁC KIỂU DỮ LIỆU ĐƯỢC GIỚI THIỆU TRONG PYTHON

Trong bài này, chúng ta sẽ cùng tìm hiểu những nội dung sau đây

- If là gì? Có ăn được không
- If
- If – else if

- If – else
- If – else if – else
- Block trong Python

If là gì? Có ăn được không?

If là một từ tiếng Anh thường gặp, khi dịch nó ra tiếng Việt ta sẽ được nghĩa là "Nếu" hoặc là "Giá mà", "Miễn là",... Dĩ nhiên là "Nếu" là một từ chẳng mấy xa lạ với các bạn. Chúng ta sử dụng nó cả trăm, ngàn lần một ngày.

- Nếu hôm nay chủ nhật, Tèo sẽ đi chơi.
- Nếu ủng hộ đủ 5000 điểm thì Kteam sẽ xuất bản khóa Kỹ Thuật Import/Export Cookie Selenium.
- Nếu được vote up câu hỏi thì bạn được cộng điểm, còn nếu bị vote down bạn sẽ bị trừ điểm, không có vote thì số điểm không thay đổi.

Python cũng biết nếu, có điều nếu khác chúng ta một tẹo. Để biết khác thế nào, chúng ta hãy cùng tìm hiểu!

If

Đây là ví dụ về câu lệnh if cơ bản nhất. **Nếu ... thì ...**

- Nếu **m – 1 < 0** thì **m < 1**

Từ đó, Python đã xây dựng một cấu trúc nếu tương tự như trên:

```
if expression:
    # If-block
```

Lưu ý: Tất cả các câu lệnh nằm trong if-block là các câu lệnh có lề thụt vào trong so với câu lệnh if. Chi tiết Kteam sẽ trình bày ở phần tiếp theo

Ở đây, nếu **expression** là một giá trị khi đưa về kiểu dữ liệu Boolean là **True** thì Python sẽ nhảy vào thực hiện các câu lệnh trong if-block. Còn nếu không thì không thì sẽ bỏ qua if-block đó.

```
>>> a = 0
>>> b = 3
>>>
>>> if a - 1 < 0: # (a – 1 < 0) có giá trị là True
...     print('a nhỏ hơn 1')
...
a nhỏ hơn 1
>>>
>>> if b - 1 < 0: # (b – 1  < 0) có giá trị là False
...     print('b nhỏ hơn 1')
...
>>>
```

If – else if

Đây là bản nâng cấp của cấu trúc **if** vừa rồi chúng ta tìm hiểu. Nó có cấu trúc như sau:

```
if expression:
    # If-block
elif 2-expression:
    # 2-if-block
elif 3-expression:
    # 3-if-block
...
elif n-expression:
```

> # **n**-if-block

Ở đây, bạn có thể đặt bao nhiêu lần **nếu** cũng được. Và từ câu lệnh **if** đến lần **elif** lần thứ **n – 1** (câu lệnh với **n**-expression) là một khối, ta sẽ đặt cho nó một cái tên là khối **BIG** để dễ hiểu. Nó sẽ hoạt động như sau:

Bước 1: Kiểm tra xem expression có phải là một giá trị Boolean **True** hay không?

Bước 2: Nếu có, thực hiện if-block sau đó kết thúc khối **BIG**. Không thì chuyển sang Bước 3.

Bước 3: Kiểu tra xem **2**-expression có phải là một giá trị Boolean **True** hay không?

Bước 4: Nếu có, thực hiện **2**-if-block sau đó kết thúc khối **BIG**. Không thì chuyển sang Bước 5.

Bước 5: Kiểm tra xem **3**-expression có phải là một giá trị Boolean **True** hay không?

Bước 6: Nếu có, thực hiện **3**-if-block sau đó kết thúc khối **BIG**. Không thì chuyển sang Bước 7

...

Bước (n - 1) x 2: Kiểm tra xem **n**-expression có phải là một giá trị Boolean **True** hay không?

Bước (n – 1) x 2 + 1: Nếu có, thực hiện **n**-if-block.

Bước (n – 1) x 2 + 2: Kết thúc khối **BIG**.

Ví dụ để các bạn dễ hiểu hơn

```
>>> a = 3
>>>
>>> if a - 1 < 0: # False, tiếp tục
...     print('a nhỏ hơn 1')
... elif a - 2 < 0: # False, tiếp tục
```

```
...     print('a nhỏ hơn 2')
... elif a - 3 < 0: # False, tiếp tục
...     print('a nhỏ hơn 3')
... elif a - 4 < 0: # True, kết thúc
...     print('a nhỏ hơn 4')
... elif a - 5 < 0: # Khối BIG đã kết thúc, dù đây là True nhưng không ý nghĩa
...     print('a nhỏ hơn 5')
...
a nhỏ hơn 4
```

If - else

Cấu trúc vừa rồi không biết có làm bạn đau đầu hay không. Nếu có, hãy thư giãn vì cấu trúc sau đây đơn giản hơn nhiều.

```
if expression:
    # If-block
else:
    # else-block
```

Nếu expression là một giá trị Boolean **True**, thực hiện if-block và kết thúc. Không quan tâm đến else-block. Còn nếu không sẽ thực hiện else-block và kết thúc.

Ví dụ:

```
>>> a = 0
>>> b = 3
>>>
>>> if a - 1 < 0:
...     print('a nhỏ hơn 1')
... else:
...     print('a lớn hơn hoặc bằng 1')
...
```

```
a nhỏ hơn 1
>>>
>>> if b - 1 < 0: # False, nên sẽ thực hiện else-block
...     print('b nhỏ hơn 1')
... else:
...     print('b lớn hơn hoặc bằng 1')
...
b lớn hơn hoặc bằng 1
```

If – else if - else

Nó không có gì mới mẻ nếu bạn nắm rõ 3 cấu trúc trên. Sau đây là cấu trúc của **if – else if – else**

```
if expression:
    # If-block
elif 2-expression:
    # 2-if-block
...
elif n-expression:
    # n-if-block
else:
# else-block
```

Bạn có thể đặt bao nhiêu lần **elif** cũng được nhưng **else** thì chỉ một. Và từ câu lệnh **if** đến câu lệnh **else** là một khối, ta cũng sẽ đặt cho nó một cái tên là khối **BIG** để dễ hiểu. Nó sẽ hoạt động như sau:

Bước 1: Kiểm tra xem expression có phải là một giá trị Boolean **True** hay không?

Bước 2: Nếu có, thực hiện if-block sau đó kết thúc khối **BIG**. Không thì chuyển sang Bước 3.

Bước 3: Kiểu tra xem **2**-expression có phải là một giá trị Boolean **True** hay không?

Bước 4: Nếu có, thực hiện **2**-if-block sau đó kết thúc khối **BIG**. Không thì chuyển sang Bước 5

...

Bước (n - 1) x 2: Kiểm tra xem **n**-expression có phải là một giá trị Boolean **True** hay không?

Bước (n – 1) x 2 + 1: Nếu có, thực hiện **n**-if-block sau đó kết thúc khối **BIG**.

Bước (n – 1) x 2 + 2: Nếu không thì thực hiện else-block và kết thúc khối **BIG**.

Ví dụ:

```
>>> a = 0
>>> if a - 1 < 0:
...     print('a nhỏ hơn 1')
... elif a - 1 > 0:
...     print('a lớn hơn 1')
... else:
...     print('a bằng 1')
...
a nhỏ hơn 1
>>>
>>> b = 2
>>> if b - 1 < 0:
...     print('b nhỏ hơn 1')
... elif b - 1 > 0:
...     print('b lớn hơn 1')
... else:
...     print('b bằng 1')
...
b lớn hơn 1
>>>
```

```
>>> c = 1
>>> if c - 1 < 0:
...     print('c nhỏ hơn 1')
... elif c - 1 > 0:
...     print('c lớn hơn 1')
... else:
...     print('c bằng 1')
...
c bằng 1
```

Block trong Python

Với đa số ngôn ngữ lập trình hiện nay, thường dùng cặp dấu ngoặc **{ }** để phân chia các block.

Riêng đối với Python lại sử dụng việc **định dạng code để suy ra các block**. Đây là điều giúp code Python luôn luôn phải đẹp mắt.

Một số điều lưu ý về việc định dạng code block trong Python:

- Câu lệnh mở block kết thúc bằng dấu hai chấm (**:**), sau khi sử dụng câu lệnh có dấu hai chấm (**:**) buộc phải xuống dòng và lùi lề vào trong và có tối thiểu một câu lệnh để không bỏ trống block.
- Những dòng code cùng lề thì là cùng một block.
- Một block có thể có nhiều block khác.
- Khi căn lề block không sử dụng cả tab lẫn space.
- Nên sử dụng **4 space** để căn lề một block

Sau đây là một hình minh họa của Kteam.

Các câu lệnh nằm trong một khung màu là một block, và block đó được mở bởi câu lệnh nằm ngay bên trên khung màu.

```python
a = 3

if a - 1 < 0:
    print('a nhỏ hơn 1')
    if a < 0:
        print('a nhỏ hơn 0')
    else:
        print('a không nhỏ hơn 0')
elif a - 1 > 0:
    print('a lớn hơn 1')
    if a - 2 > 0:
        print('a lớn hơn 2')
        if a - 3 > 0:
            print('a lớn hơn 3')
    elif a - 2 == 0:
        print('a bằng 2')
    else:
        print('1 < a < 2')
else:
    print('a bằng 1')

```

Lưu ý: Kteam có đề cập đến việc sau khi sử dụng câu lệnh có dấu hai chấm (**:**) buộc phải xuống dòng và lùi lề vào trong. Tuy nhiên, Bạn vẫn có thể đi ngược lại điều này trong một vài trường hợp

Ví dụ:

```
>>> a = 3
>>> if a - 1 > 0: print('a lớn hơn 1')
...
a lớn hơn 1
>>> if a - 1 > 0: print('a lớn hơn 1'); print('có thể a lớn hơn 2')
...
a lớn hơn 1
có thể a lớn hơn 2
```

Tuy nhiên, việc sử dụng như vậy không được khuyến khích vì chỉ tiết kiếm được một vài dòng code mà lại gây khó đọc thì không đáng để tiết kiệm.

Và bạn cũng đã biết thêm một điều Python không hề cấm dấu chấm phẩy (;). Nó vẫn là một cú pháp hợp lệ. Nếu bạn quen tay có thể dùng dấu chấm phẩy (;) thoải mái.

Củng cố bài học

Câu hỏi củng cố

Nhập từ bàn phím 3 số, in ra số lớn nhất (cố gắng ít dòng code nhất có thể - ở đây không tính việc nhập dữ liệu)

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 CÂU ĐIỀU KIỆN IF TRONG PYTHON.

Ở bài viết sau. Kteam sẽ nói về khái niệm vòng lặp và biết tới CẤU TRÚC VÒNG LẶP WHILE TRONG PYTHON.

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**".