Bài 23: KIỂU DỮ LIỆU BOOLEAN TRONG PYTHON

Xem bài học trên website để ủng hộ Kteam: Kiểu dữ liệu Boolean 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 HÀM INPUT - một hàm giúp bạn yêu cầu nhập dữ liệu từ bàn phím

Ở bài này Kteam sẽ giới thiệu với các bạn **Kiểu dữ liệu Boolean trong Python**. Một kiểu dữ liệu cực kì cần thiết trong các phần sử dụng cấu trúc rẽ nhánh, vòng lặp.

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

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

- Giới thiệu về Boolean trong Python
- Boolean trong các toán tử so sánh
- NOT, AND và OR
- Các giá trị cũng là các Boolean
- Syntaxnic sugar cho việc so sánh trong Python

Giới thiệu về Boolean trong Python

Boolean là một kiểu dữ liệu mà các ngôn ngữ lập trình ngày này đều thường xuyên sử dụng. Python cũng không ngoại lệ.

Kiểu dữ liệu này chỉ có hai giá trị:

- Một là **True** – có nghĩa là đúng
- Nếu không thì là **False** – có nghĩa là sai.

Bạn cũng đã thấy nó rồi khi sử dụng toán tử in trong các bài kiểu dữ liệu chuỗi, list,...

Boolean trong các toán tử so sánh

So sánh giữa số với số

Bạn chắc biết so sánh là gì nhờ các tiết học toán ở trường. Ví dụ như

- **3 > 1** là đúng
- **69 < 10** là sai
- **241 = 141 + 100** là đúng
- **(5 x 0) ≠ 0** là sai.

Trong Python cũng có các toán tử như vậy. Tuy nhiên kí hiệu của chúng thì có khác đôi chút.

Bảng sau đây sẽ cho các bạn thông tin về những toán tử so sánh trong Python

Toán học	Python
>	>
<	<
=	==
≥	>=
≤	<=
≠	!=

Hãy xem ví dụ minh họa trong Python:

```
>>> 3 > 1 # 3 > 1 là đúng => True
True
>>> 69 < 10 # 69 < 10 là sai => False
False
>>> 241 == 141 + 100 # 241 = 141 + 100 là đúng => True
True
>>> (5 * 0) != 0  # 5 x 0 ≠ 0 là sai => False
False
```

Bạn cũng có thể so sánh nhiều kiểu dữ liệu khác nữa, không chỉ là giữa số với số.

So sánh giữa hai iterable cùng loại

Khi so sánh hai iterable cùng loại. Python sẽ lấy lần lượt từng phần tử trong iterable ra so sánh. Kteam sẽ lấy ví dụ về kiểu chuỗi:

```
>>> 'Kteam' == "Kteam"
True
>>> 'Free' == 'Education'
False
```

Lưu ý: Python so sánh các kí tự với nhau bằng cách đưa chúng về dưới dạng số bằng hàm **ord**. Bạn có thể tham khảo giá trị của nó trong **ASCII** Table.

```
>>> ord('A')
65
>>> ord('a')
97
```

Khi bạn so sánh bằng các toán tử **==**, **>=**, **<=** thì Python sẽ so sánh hết các phần tử.

Còn nếu bạn dùng các toán tử như **>**, **<**, **!=** thì nhiều lúc Python sẽ không cần phải đi hết các giá trị iterable. Nếu như ở vị trí **i** nào đó mà đã hai giá trị không bằng nhau thì nó sẽ dừng lại.

```
>>> 'a' > 'ABC'
# ord('a') không bằng ord('A'), không cần phải so sánh tiếp và ord('a') > ord('A') là đúng => True
True
>>> 'aaa' < 'aaAcv'
# ord('a') không bằng ord('A') ở vị trí thứ 2, không cần phải so sánh tiếp và ord('a') < ord('A') là sai => False
False
>>> 'aaa' < 'aaaAcv'
# 3 phần tử đầu tiên bằng nhau. Ở phần tử thứ tư, ta sẽ so sánh 0 và ord('A') và dĩ nhiên ord('A') > 0 => True
True
```

Toán tử is

Đây là một toán tử dễ nhầm lẫn với toán tử **==**. Nhưng thật sự thì nó rất đơn giản!

Ở đây, Kteam sẽ nói tới một phần kiến thức ở tiếng Anh để bạn có thể dễ phân biệt 2 toán tử trên. Từ **is** trong tiếng Việt (ở ngữ cảnh này – ngôn ngữ lập trình Python) có nghĩa là "**là**". Còn toán tử **==** có nghĩa là **bằng**.

Kteam sẽ đưa ra một ví dụ. Bạn cũng không nên khắt khe việc đúng sai trong ví dụ này, nó chỉ giúp bạn hiểu sự khác nhau giữa toán tử **==** và **is** thôi.

Thế nào là **bằng** (**==**)?

- **Bằng** là toán tử so sánh khi nói về mặt giá trị.
- **Ví dụ:** Chiều cao của Tèo **bằng** chiều cao của Tí

Thế nào là **là** (**is**)?

- **Là** (**is**) trong trường hợp này là liên từ diễn giải định nghĩa, tính chất của một sự vật/sự việc/con người.
- **Ví dụ:** Ta không thể nói "Chiều cao của Tèo là chiều cao của Tí" vì của Tèo là của Tèo, đâu phải của Tí. Nên nói là "Chiều cao của Tèo là chiều cao của Tèo" hoặc "Chiều cao của Tí là chiều cao của Tí"

Ta hãy trở lại với Python bằng việc khởi tạo hai List

```
>>> lst = [1, 2, 3]
>>> lst_ = [1, 2, 3]
```

Chúng đều có giá trị là một List gồm ba phần tử 1, 2 và 3. Vậy chúng có bằng nhau? Đương nhiên là **có**. Thử luôn là biết.

```
>>> lst == lst_
True
```

Nhưng **lst** có phải là **lst_**? Đương nhiên là **không**. Vì đó là hai List khác nhau không liên quan đến nhau.

```
>>> lst is lst_
False
```

Vậy nếu ta có một List khác

```
>>> _lst = lst
>>> _lst
[1, 2, 3]
```

Thì **_lst** có phải là **lst** không? Nếu bạn còn nhớ một số điều lưu ý khi sử dụng List trong bài KIỂU DỮ LIỆU LIST TRONG PYTHON – PHẦN 1 thì chắc chắn là bạn còn nhớ, 2 List này đang trỏ chung vào một địa chỉ. Do đó, chúng là một, chỉ khác nhau cái nhãn thôi.

```
>>> _lst is lst
True
```

Từ đây, ta có thể suy ra một kết luận. Khi so sánh hai giá trị (đối tượng) bằng toán tử **==** thì Python sẽ **so sánh bằng giá trị** của chúng. Còn nếu so sánh bằng toán tử **is** thì Python sẽ lấy **giá trị của hàm id để so sánh**.

Lưu ý toán tử is

Bạn không nên so sánh 2 số như thế này

```
>>> 699 is 699
True
```

Kết quả luôn là **True**. Bạn sẽ chỉ thấy khác biệt khi:

```
>>> a = 699
>>> b = 699
>>> a is b
False
```

Nhưng, có một số trường hợp bạn cần biết:

```
>>> a = -5
>>> b = -5
>>> a is b
True
```

```
>>> c = 256
>>> d = 256
>>> c is d
True
>>> a = 'abc'
>>> b = 'abc'
True
```

Các số từ -5 đến 256 hoặc là một số chuỗi có số kí tự dưới 20 thì các biến có cùng một giá trị sẽ có cùng một giá trị trả về từ **hàm id**.

NOT, AND và OR

Not là **phủ định**.

Đây là cách bạn có thể đổi giá trị Boolean. Trong một số trường hợp đặc biệt. Việc kiểm tra giá trị Boolean đó là **False** hay là **True** hơi phức tạp, rườm ra trong khi đó việc kiểm tra giá trị ngược lại thì dễ dàng, đơn giản hơn.

Value	Not
True	False
False	True

And là **và**.

Or là **hoặc**.

Bạn cần nằm lòng bảng sau để có thể kết hợp những điều kiện một cách nhuần nhuyễn. Từ đó, bạn có thể sử dụng linh hoạt các câu lệnh điều kiện, đặt expression cho các vòng lặp một cách hiệu quả.

Bạn hãy xem bảng sau đây:

Left-Value	Right-Value	And	Or
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

Ví dụ: để rõ hơn nhé. Đầu tiên là **and**

```
>>> True and True
True
>>> True and False
False
>>> False and True
False
>>> False and False
False
```

Tiếp đến là **or**

```
>>> True or True
True
>>> True or False
True
>>> False or True
True
>>> False or False
False
```

Cuối cùng là **not**

```
>>> not True
False
>>> not False
True
```

Các giá trị cũng là các Boolean

Thật vậy, các giá trị đều là các boolean. Và đương nhiên, bạn có thể chuyển đổi chúng thành các Boolean bằng hàm **bool**.

Mọi giá trị khi chuyển về Boolean đều là **True** trừ một số trường hợp sau

- Số 0
- None
- Rỗng

Ví dụ: để hiểu hơn

```
>>> bool(0)
False
>>> bool(None)
False
>>> bool('')
False
>>> bool([])
False
>>> bool(())
False
>>> bool(set())
False
>>> bool({})
False
```

Thêm một số trường hợp **True**

```
>>> bool(1)
True
>>> bool('abc')
True
>>> bool([1, 2, 3])
True
```

1 là True, 0 là False

Không quá quan trọng, nhưng cũng nên biết

```
>>> True + 1
2
>>> False + 1
1
>>> int(True)
1
>>> int(False)
0
```

Syntaxnic sugar cho việc so sánh trong Python

Nếu bạn từng học một số ngôn ngữ lập trình khác. Bạn đôi lúc phải kiểm tra những trường hợp như kiểu tra một số **n** có nằm trong khoảng **(a; b)**, đoạn **[a; b]**, nửa khoảng **(a; b]**, nửa khoảng **[a; b)** hay không? hoặc là kiểm tra xem một số **k** có bằng một trong những số như x, y hoặc z hay không. Đương nhiên, những lần làm như vậy cũng làm bạn hơi cực

```
>>> n = 5
>>> n > 1 and n < 6 # kiểm tra xem n có nằm trong khoảng (1; 6) hay không
True
>>> n > 1 and n < 4 # kiểm tra xem n có nằm trong khoảng (1; 4) hay không
False
```

Nhưng với Python, bạn có thể làm thế này.

```
>>> 1 < a < 6
True
>>> b = -4
>>> b < -3 < -1 < 0 < a < 6 # thậm chí là dài như thế này
True
```

Với trường hợp nếu bạn muốn kiểm tra xem một số **k** có bằng **x** hoặc **y** hoặc là **z** hay không thì thường bạn phải viết khá dài.

```
>>> k = 4
>>> k == 3 or k == 4 or k == 5
True
```

Tuy nhiên, bạn cũng có thể

```
>>> k in (3, 4, 5) # nên dùng () hơn là [] hoặc thứ gì khác
True
```

Kết luận

Bài viết này đã giới thiệu sơ cho các bạn KIỂU DỮ LIỆU BOOLEAN TRONG PYTHON.

Ở bài sau, Kteam sẽ giới thiệu đến bạn CẤU TRÚC RẼ NHÁNH 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**".