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 19. TOÁN TỬ LOGIC TRONG PYTHON

Toán tử logic trong Python được sử dụng để tạo ra các biểu thức logic phức tạp. Mỗi toán hạng cho các toán tử logic này chính là một biểu thức logic. Ví dụ:

VÍ DỤ

```
age > 16 and marks > 80
percentage < 50 or attendance < 75
```

Cùng với từ khóa `False`, Python hiểu `None`, số không của tất cả các loại, và các chuỗi rỗng (chuỗi, tuple, list), từ điển rỗng và tập hợp rỗng là `False`. Tất cả các giá trị khác được coi là `True`.

Trong Python, có ba toán tử logic. Chúng là "and", "or" và "not". Chúng phải ở dạng viết thường.

TOÁN TỬ LOGIC "AND"

Đối với biểu thức logic phức tạp, cả hai toán hạng phải đều là `True` để biểu thức trở thành `True`. Nếu bất kỳ hoặc cả hai toán hạng đều đánh giá thành `False`, biểu thức trả về `False`.

BẢNG CHÂN TRỊ CỦA TOÁN TỬ LOGIC "AND"

Bảng dưới đây hiển thị các trường hợp:

a	b	a and b
F	F	F
F	T	F
T	F	F
T	T	T

TOÁN TỬ LOGIC "OR"

Ngược lại, toán tử or trả về `True` nếu bất kỳ một trong các toán hạng là `True`. Đối với biểu thức logic phức tạp, cả hai toán hạng phải đều là `False` để biểu thức trở

thành `False`.

BẢNG CHÂN TRỊ CỦA TOÁN TỬ LOGIC "OR"

Bảng dưới đây hiển thị kết quả của toán tử "or" với các điều kiện khác nhau:

a	b	a or b
F	F	F
F	T	T
T	F	T
T	T	T

TOÁN TỬ LOGIC "NOT"

Đây là một toán tử một ngôi. Trạng thái của toán hạng logic theo sau sẽ được đảo ngược. Kết quả là, not True trở thành False và not False trở thành True.

BẢNG CHÂN TRỊ CỦA TOÁN TỬ LOGIC "NOT"

a	not (a)
F	T
T	F

CÁCH TRÌNH DIỄN PYTHON ĐÁNH GIÁ CÁC TOÁN TỬ LOGIC?

Biểu thức "x and y" đầu tiên đánh giá "x". Nếu "x" là `False`, giá trị của nó được trả về; nếu không, "y" được đánh giá và giá trị kết quả được trả về.

Biểu thức "x or y" đầu tiên đánh giá "x"; nếu "x" là `True`, giá trị của nó được trả về; nếu không, "y" được đánh giá và giá trị kết quả được trả về.

VÍ DỤ VỀ TOÁN TỬ LOGIC PYTHON

Dưới đây là một số trường hợp sử dụng của các toán tử logic:

VÍ DỤ 1: CÁC TOÁN TỬ LOGIC VỚI ĐIỀU KIỆN BOOLEAN

< 8 Sync Dev />

```
x = 10
y = 20
print("x > 0 and x < 10:",x > 0 and x < 10)
print("x > 0 and y > 10:",x > 0 and y > 10)
print("x > 10 or y > 10:",x > 10 or y > 10)
print("x%2 == 0 and y%2 == 0:",x%2 == 0 and y%2 == 0)
print ("not (x+y>15):", not (x+y)>15)
```

Kết quả sẽ là:

< 8 Sync Dev />

```
x > 0 and x < 10: False
x > 0 and y > 10: True
x > 10 or y > 10: True
x%2 == 0 and y%2 == 0: True
not (x+y>15): False
```

VÍ DỤ 2: CÁC TOÁN TỬ LOGIC VỚI ĐIỀU KIỆN KHÔNG PHẢI BOOLEAN

Chúng ta có thể sử dụng các toán hạng không phải boolean với các toán tử logic. Ở đây, chúng ta cần nhớ rằng bất kỳ số khác không và các chuỗi không rỗng sẽ được đánh giá thành `True`. Do đó, các bảng chân trị của các toán tử logic cũng áp dụng.

Trong ví dụ dưới đây, các toán hạng số được sử dụng cho các toán tử logic. Các biến "x", "y" được đánh giá thành `True`, "z" là `False`.

< 8 Sync Dev />

```
x = 10
y = 20
z = 0
print("x and y:",x and y)
print("x or y:",x or y)
print("z or x:",z or x)
print("y or z:", y or z)
```

Kết quả sẽ là:

< 8 Sync Dev />

```
x and y: 20
x or y: 10
z or x: 10
y or z: 20
```

VÍ DỤ 3: CÁC TOÁN TỬ LOGIC VỚI CHUỖI VÀ TUPLE

Biến chuỗi được coi là `True` và tuple rỗng là `False` trong ví dụ dưới đây.

< 8 Sync Dev />

```
a="Hello"
b=tuple()
print("a and b:",a and b)
print("b or a:",b or a)
```

Kết quả sẽ là:

< 8 Sync Dev />

```
a and b: ()
b or a: Hello
```

VÍ DỤ 4: CÁC TOÁN TỬ LOGIC ĐỂ SO SÁNH CÁC CHUỖI (DANH SÁCH)

Cuối cùng, hai đối tượng danh sách dưới đây là không rỗng. Do đó, x and y trả về phần sau cùng, và x or y trả về phần đầu tiên.

< 8 Sync Dev />

```
x=[1,2,3]
y=[10,20,30]
print("x and y:",x and y)
print("x or y:",x or y)
```

Kết quả sẽ là:

< 8 Sync Dev />

```
x and y: [10, 20, 30]
x or y: [1, 2, 3]
```

Các ví dụ trên giúp bạn hiểu rõ về cách sử dụng và hoạt động của các toán tử logic trong Python. Hãy áp dụng chúng vào mã của bạn và tận dụng tính linh hoạt mà chúng mang lại!