

Booleans and If Statements: Takeaways

by Dataquest Labs, Inc. - All rights reserved © 2018

Syntax

COMPARISON OPERATORS

- `==` returns:
 - **True** if both values are equivalent: `5 == 5`
 - **False** if they're different: `5 == 6`
- `!=` returns:
 - **True** if both values are different: `5 != 6`
 - **False** if they're equivalent: `5 != 5`
- `>` returns:
 - **True** if the first value is greater than the second value: `5 > 1`
 - **False** if the first value is less than the second value: `5 > 6`
- `<` returns:
 - **True** if the first value is less than the second value: `5 < 6`
 - **False** if the first value is greater than the second value: `5 < 1`
- `>=` returns:
 - **True** if the first value is greater than or equal to the second value: `1 >= 1`
 - **False** if the first value is less than the second value: `1 >= 3`
- `<=` returns:
 - **True** if the first value is less than or equal to the second value: `1 <= 2`
 - **False** if the first value is greater than the second value: `1 <= 0`

IF STATEMENTS

- Basic if statement syntax:

```
python
sample_rate = 749
if sample_rate > 750:
    print("Higher than 750")
```

- Nested if statements:

```
python
sample_rate = 749
if sample_rate > 750:
    if sample_rate < 748:
        print("Higher than 750 AND lower than 748")
```

- If statement within a for loop:

```
python
sample_list = [0,1,2]
which_index = 0
for s in sample_list:
    if s == 2:
        which_index = counter
    counter += 1
```

Concepts

- We can use conditional logic to add criteria to the code we write. Some examples of operations that use criteria include:
 - Finding all the integers in a list that are greater than 5.
 - Identifying which elements in a list are strings and printing only those values.
- Python has a class called **Boolean** that helps express conditional logic. There are only two Boolean values: **True** and **False**. Because the words *Boolean* values may look like strings, but they're an entirely separate class.

- To complement *boolean*, Python contains the *if* operator. We can use this operator to write a statement that tests whether certain conditions exist. Our *if* statement will evaluate to either *True* or *False* and only run the specified code when *True*.
- Similar to *for* loops we need to format *if* statements in the following way:
 - End the conditional statement with a colon (`:`)
 - Indent the code (that we want run when *True*) below the conditional statement
 - Also similar to *for* loops *if* statements can contain multiple lines in the body as long as their indentation aligns.

Resources

- [Python Documentation: Comparison Operators](#)
- [Python Documentation: If Statements](#)



Takeaways by Dataquest Labs, Inc. - All rights reserved © 2018