

Notes - Monday 12/28/20

12/28/20 Monday

Let's keep going with the Udemmy course! We took a smol holiday break. Starting with boolean operators.

Boolean Comparison operators

`==` Equality: Checks to see if objects are equal to each other. Type of data matters so if you compare `"2" = 2` it won't yield True since you're comparing a string to an integer. If you compare `'Bye' = 'bye'` it won't yield true since case matters.

`!=` Inequality: Checks to see if objects are UNEqual to each other. So let's say you want to see if 4 is not equal to 5. Your syntax would be `4 != 5` and Python would yield True.

`>` Greater Than: Checks to see if the objects are greater than each other.

`<` Less Than: Checks to see if the objects are less than each other.

`>=` Greater Than or Equal to: as described. Like the greater than and equality check together.

`<=` Less Than or Equal to: as described like the less than and equality check together.

Chaining Boolean/Comparison Operators

So we discussed comparison operators above:

- Equality `==`
- Inequality `!=`
- Greater Than `>`
- Less Than `<`
- Greater Than or Equal to `>=`
- Less Than or Equal to `<=`

Now we're going to add in:

- AND
- OR
- NOT

Let's say that we want to compare 1, 2, and 3.

1. `1 < 2 < 3` will work and output True.
2. `1 < 2 > 3` will output False even though `1 < 2` is true.

AND Operand

In example #2, we could use the `AND` operator which would link the two comparisons together. So in other words, `AND` wants both the statements on either side to be True. The syntax would be:

```
1 < 2 AND 2 > 3
```

So this is checking if `1 < 2` is true AND THEN if `2 > 3` is true. If both are true then we'll get an output of true. But since `2 > 3` is false, the result will output as False.

OR Operand

This logic links 2 statements together and checks to see if either pass the logic presented. In other words, `OR` wants one of the statements on either side to be True. So if we use example #2 again, the `OR` operator can be used. And the syntax would look like:

```
1 < 2 OR 2 > 3
```

So this is checking if either of the comparison statements are true. Either statement on the left or right can yield a true output and the `OR` operator will output as True.

NOT Operand

This logic looks for the reverse of what's inside the statement. Another way to say this is that the `NOT` operand is looking for the opposite of the boolean or syntax or statement in the line of code. So let's say our example is:

```
100 == 1
```

So if we ran this as is, we would expect a False output as 100 does not equal 1. But if we used the `NOT` operand then we would expect a True output. This is because we're asking Python if the example statement is NOT equal. So the syntax would look like:

```
not 100 == 1
```

Future Usage

These `AND`, `OR`, and `NOT` operands can be used in these simpler boolean/comparison operators. It could also be used in If statements.

The syntaxes might change depending on the library used but these operands could be wrapped in a parantheses or not. For example the above `NOT` operand example could look like either one of these:

- `not (100 == 1)`
- `not 100 == 1`

For now either of these are acceptable but as stated, the syntax may change depending on the libraries used which will be covered later.

Revision #2

Created 22 December 2020 18:13:55 by cba88

Updated 28 December 2020 21:38:55 by cba88