Boolean Operations

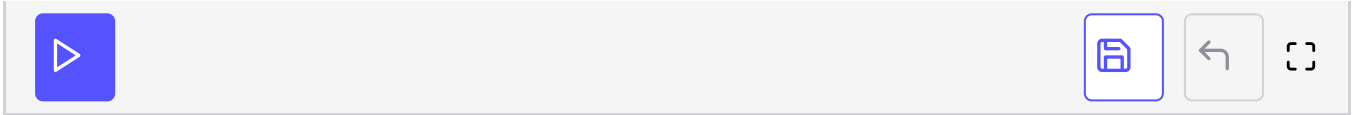
Boolean operations are used to combine multiple conditions. Let's learn how we are going to use boolean operators in python.

Now we're ready to learn about Boolean operations (and, or, not). According to the Python documentation, their order of priority is first **or**, then **and**, then **not**. Here's how they work:

> - **or** means that if any conditional that is "ored" together is True, then the following statement runs
> - **and** means that all statements must be True for the following statement to run
> - **not** means that if the conditional evaluates to False, it is True. This is the most confusing, in my opinion.

Let's take a look at some examples of each of these. We will start with **or**.

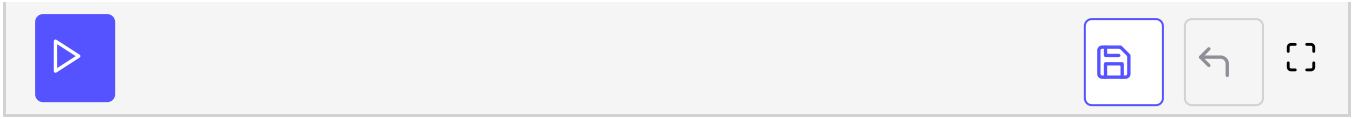
```
x = 10
y = 20

if x < 10 or y > 15:
    print("This statement was True!")
```



Here we create a couple of variables and test if one is less than ten or if the other is greater than 15. Because the latter is greater than 15, the print statement executes. As you can see, if one or both of the statements are True, it will execute the statement. Let's take a look at how **and** works:

```
x = 10
y = 10
if x == 10 and y == 15:
    print("This statement was True")
```

```
else:
    print("The statement was False!")
```

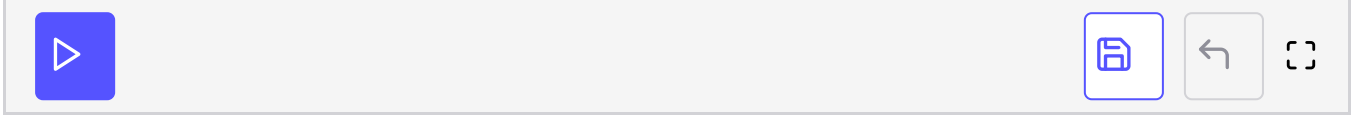


If you run the code above, you will see that first statement does not get run. Instead, the statement under the **else** is executed. Why is that? Well, it is because what we are testing is both x **and** y are 10 and 15 respectively. In this case, they are not, so we drop to the else. Thus, when you **and** two statements together, **both** statements have to evaluate to True for it to execute the following code. Also note that to test equality in Python, you have to use a double equal sign. A single equals sign is known as the **assignment operator** and is only for assigning a value to a variable. If you had tried to run the code above with one of those statement only having one equals sign, you would have received a message about invalid syntax.

Note that you can also **or** and **and** more than two statements together. However, I do not recommend that as the more statements that you do that too, the harder it can be to understand and debug.

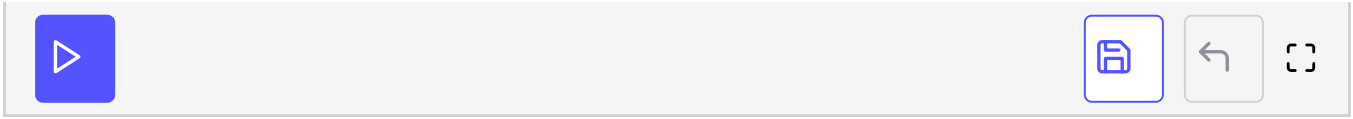
Now we're ready to take a look at the **not** operation.

```
my_list = [1, 2, 3, 4]
x = 10
if x not in my_list:
    print("'x' is not in the list, so this is True!")
```



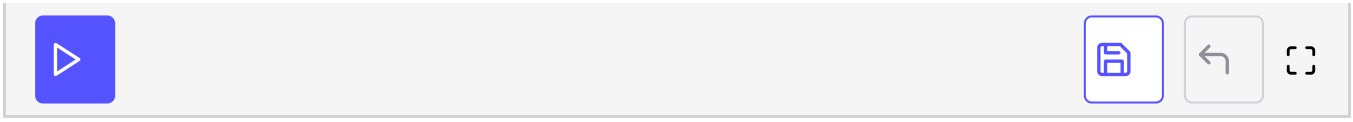
In this example, we create a list that contains four integers. Then we write a test that asks if "x" is not in that list. Because "x" equals 10, the statement evaluates to True and the message is printed to the screen. Another way to test for **not** is by using the exclamation point, like this:

```
x = 10
if x != 11:
    print("x is not equal to 11!")
```



If you want to, you can combine the **not** operation with the other two to create more complex conditional statements. Here is a simple example:

```
my_list = [1, 2, 3, 4]
x = 10
z = 11
if x not in my_list and z != 10:
    print("This is True!")
```



Personally, I find the **not** operator a little confusing and don't use it that much. But you will find it useful from time to time.