Conditional Statements

In this lesson, we will go over conditional statements in Python.

WE'LL COVER THE FOLLOWING

- if statements
- if-else statements

A conditional statement is a `boolean` expression that, if `True`, executes a piece of code. It allows programs to branch out into different paths based on the outcome of `boolean` expressions:

```
if condition 1 is true:
    execute expression 1

if condition 2 is true:
    execute expression 2

else:
    execute expression 3
```

Conditional statements control the flow of the code and are classified as **control structures.**

if statements

The simplest conditional statement that we can write is the if statement. It is comprised of two parts:

```
if condition:
    code to be executed
```

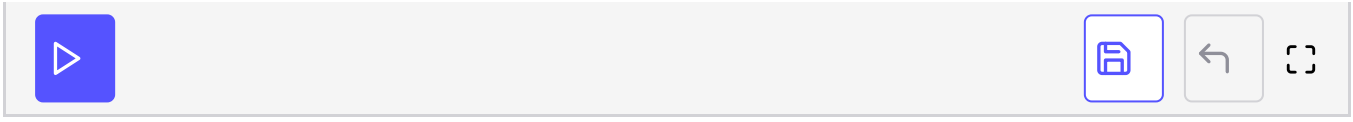
💡 **if** the **condition** holds `True`, execute the code to be executed.

Otherwise, skip it and move on.

```
num = 10

if num == 10: # The condition is true
  print ("The number is equal to 10") # The code is executed

if num > 10: # The condtion is false
  print ("The number is greater than 10") # The code is not executed
```



Our first condition simply checks if the value of `num` is `10` and the second condition checks whether the value of `num` is greater than `10`.

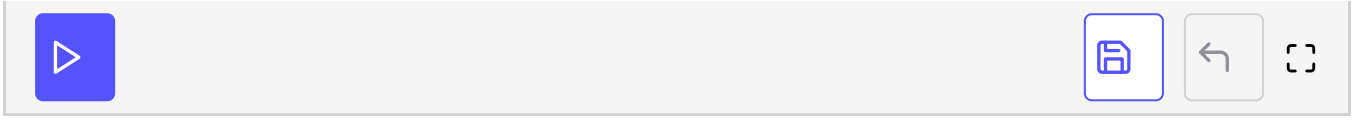
Since `num` is equal to `10`, the first expression returns `True`, and the compiler goes ahead and executes the print statement on line 4.

We can use logical operators and nested `if` statements to create more complex conditions in the `if` statement.

```
num = 72

if num % 2 == 0 and num % 3 == 0 and num % 4 == 0 :
    # Only works when num is a multiple of 2, 3, and 4
    print ("The number is a multiple of 2, 3, and 4")

    # This nested if only works when num is a multiple of 2, 3, 4 and 6
    if num % 6 == 0:
        print ("The number is also a multiple of 6")
```



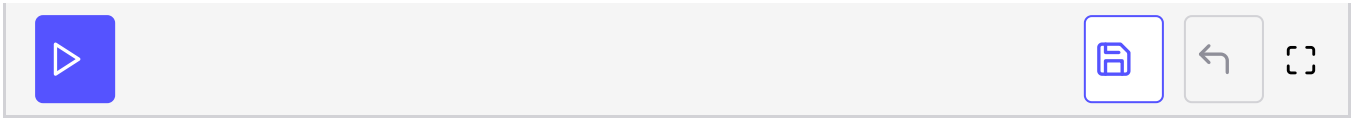
if-else statements

When we want to execute a different set of operations in case the `if` condition turns out to be `False`, we use the `if-else` statement. When the `if` condition turns out to be `False`, the code after the `else` keyword is executed. Let's see an example below:

```
num = 12

if (num == 10): # The condition is not true
  print ("The number is equal to 10") # The code is not executed
```

```
else:   # This will be automatically executed
  print ("The number is not equal to 10") # The code is executed
```



We'll learn about loops in Python in the next lesson.