KEY_Lesson09_Conditionals

February 4, 2020

1 Conditional Logic

In the last lesson, we learned about logic with booleans. Booleans can be used to determine if a certain line of code should be run. How would we do this? This is where we can use something called an if statement.

If statements follow the following formula:

```
if [condition]:
    [code]
```

The **condition** is a boolean, and the code under the condition only runs if the condition is true. These are called **if statements**

For example, say we had a list that we wanted to print out, but we don't want to print the list if it's too long. We could run the following:

[1]:
```
# create a list called my_list containing the numbers 0 to 7
my_list = [0, 1, 2, 3, 4, 5, 6, 7]

# write an if statement that only prints my_list if it has less than 10 items
if len(my_list) < 10:
    print(my_list)
```

```
[0, 1, 2, 3, 4, 5, 6, 7]
```

Let's test the conditional statement on its own to see what the result is:

[2]:
```
# print the boolean value of the condition in the if statement above
len(my_list) < 10
```

[2]:
```
True
```

We see that the condition is `True`, the code below runs.

What if we changed the condition above? How would this change our code?

[4]:
```
# write an if statement that only prints my_list if it has 10 or more items
if len(my_list) >= 10:
    print(my_list)
```

Nothing happened! Let's see why this is.

[5]:
```
# print the boolean value of the above if statement
len(my_list) >= 10
```

[5]:
```
False
```

In this case, the condition is `False`, so the code within the conditional did not run.

What if we wanted to run one block of code if the condition is `True`, and a different line of code if the condition is `False`? In other words, what if we wanted to run one block of code if our list has less than 10 items, and another block of code if the list has more than 10? This is where we use `if/else`.

An `if/else` looks like this:

```
if [condition]:
  [command 1]
else:
  [command 2]
```

The first two lines should look familiar; it's an if statement, which we just used! But after the if statement, we write `else:` and then some more python code. The code that comes after `else:` will run if the value of `condition` is false.

Let's see how this works! Let's write code that prints `my_list` if it has less than 10 items, and otherwise prints a message letting us know the list is too long to print.

[6]:
```
# write an if/else statement that prints my_list if my_list has less than 10␣
↪items
# and if my_list has 10 or more items, prints a message
if len(my_list) < 10:
    print(my_list)
else:
    print("The list is too long to print!")
```

```
[0, 1, 2, 3, 4, 5, 6, 7]
```

Because the list has less than 10 items, the list printed. But let's say we added more items to the list, making it longer than 10 items.

[7]:
```
# append the numbers 8, 9, 10, and 11 to my_list
my_list.append(8)
my_list.append(9)
my_list.append(10)
my_list.append(11)
```

[8]:
```
# print the length of my_list
len(my_list)
```

[8]:
```
12
```

Now, the length of my_list is above 10. How would the above line of code run now?

[9]:
```
# copy and paste the if/else statement we just wrote
if len(my_list) < 10:
    print(my_list)
else:
    print("The list is too long to print!")
```

```
The list is too long to print!
```

What if there were more than two different conditions? Using `if/elif/else` statements, we can command Python to run different code for any number of conditions. An `if/elif/else` statement looks like this:

```
if [condition 1]:
  [command 1]
elif [condition 2]:
  [command 2]
else:
  [command 3]

[more python code]
```

How does Python run this code? First it checks to see if `condition 1` is true. If it is, then it runs `command 1`, skips `command 2` and `command 3`, then continues running whatever code that comes after the `if/elif/else`.

If `condition 1` is false, but `condition 2` is true, then Python runs `command 2`, skips `command 3`, and continues with the code that comes after the `if/elif/else`.

Finally, if `condition 1` AND `condition 2` are both false, then `command 3` runs, and continues with the code that comes after the `if/elif/else`.

Obviously, Python is doing a lot to make these statements work! Here is a diagram of how these statements work:

We can do a lot of really powerful things with `if/elif/else` statements! Let's start by writing code that does the following: - If my_list has less than 10 items, print a message saying the list is "short" - If my_list has at least 10 items, but fewer than 15 items (so 10, 11, 12, 13, or 14 items), print a message saying the list is "medium" - Otherwise, print a message saying the list is "long"

[10]:
```
# use if/elif/else to print the length of my_list
if len(my_list) < 10:
    print("This list is short")
elif len(my_list) >= 10 & len(my_list) < 15:
    print("This list is medium length")
else:
    print("This list is really long!")
```

```
This list is medium length
```

So what happened here? `my_list` is 12 items long. We got to the first `if` statement, and because the length was not less than 10, we did not execute the line that says `This list is short`. The

next line tests if the length is between 10 and 15. This conditional is true, so it executes the line saying `This list is medium length`. Because we got to this line, we skip the `else` statement.

But what if our list was longer? Then what would happen?

[11]:
```
# append numbers 12, 13, and 14 to my_list
my_list.append(12)
my_list.append(13)
my_list.append(14)
my_list.append(15)
```

[12]:
```
# print the length of my_list
len(my_list)
```

[12]:
```
16
```

Let's copy and paste our `if/elif/else` statement from before and see how our code works now that my_list is longer!

[13]:
```
# copy and paste if/elif/else from before
if len(my_list) < 10:
    print("This list is short")
elif len(my_list) >= 10 and len(my_list) < 15:
    print("This list is medium length")
else:
    print("This list is really long!")
```

```
This list is really long!
```

Great job! You just learned how to use conditionals in Python. You learned: - How to use a boolean to only execute code under certain conditions using `if` statements - How to further control how code runs using `if/else` and `if/elif/else` statements.

[ ]:
```

```