Checking for Nothing

Python has a way to depict the absence of a value. Let's learn about python keyword 'None'

Because we are talking about statements that evaluate to True, we probably need to cover what evaluates to False. Python has the keyword **False** which I've mentioned a few times. However an empty string, tuple or list also evaluates to False. There is also another keyword that basically evaluates to False which is called **None**. The None value is used to represent the absence of value. It's kind of analogous to Null, which you find in databases. Let's take a look at some code to help us better understand how this all works:

```
empty_list = []
empty_tuple = ()
empty_string = ""
nothing = None

if empty_list == []:
    print("It's an empty list!")

if empty_tuple:
    print("It's not an empty tuple!")

if not empty_string:
    print("This is an empty string!")

if not nothing:
    print("Then it's nothing!")
```

The first four lines set up four variables. Next we create four conditionals to test them. The first one checks to see if the **empty_list** is really empty. The second conditional checks to see if the **empty_tuple** has something in it. Yes, you read that right, The second conditional only evaluates to True if the tuple is **not** empty! The last two conditionals are doing the opposite of the second. The third is checking if the string **is** empty and the fourth is checking if the **nothing** variable is really None.

The **not** operator means that we are checking for the opposite meaning. In other words, we are checking if the value is NOT True. So in the third example, we check if the empty string is REALLY empty. Here's another way to write the same thing:

```
empty_string = ""

if empty_string == "":
    print("This is an empty string!")
```

To really nail this down, let's set the **empty_string** variable to actually contain something:

```
empty_string = "something"
if empty_string == "":
    print("This is an empty string!")
```

If you run this, you will find that nothing is printed as we will only print something if the variable is an empty string.

Please note that none of these variables equals the other. They just evaluate the same way. To prove this, we'll take a look at a couple of quick examples:

```
empty_list = []
empty_string = "something"
nothing = None

print(empty_list == empty_string) # False
print(empty_string == nothing) # False
```

As you can see, they do not equal each other. You will find yourself checking your data structures for data a lot in the real world. Some programmers actually like to just wrap their structures in an exception handler and if they happen to be empty, they'll catch the exception. Others like to use the strategy

mentioned above where you actually test the data structure to see if it has something in it. Both strategies are valid.