Built-in Python Operators, Functions, and Methods - Lab

Introduction

We've looked at some of the built-in methods, functions, and the operators in Python. These are all very powerful tools we can (and will) use in our code. Below, we'll put these new tools to use to solve the tests in this lab.

Objectives

In this lab you will:

- Use built-in Python functions and methods
- Use comparison operators to compare objects
- Use logical operators to incorporate multiple conditions

- Use identity operators to confirm the identity of an object

Instructions

Let's start by using some built-in functions and methods. Employ the appropriate functions and methods to get the intended result.

```
# Desired output: "HELLO, THERE"
yell_hello = "hello, there".upper()
yell_hello
```

```
'HELLO, THERE'
```

```
# Desired output: "psst, hey"
whisper_hey = "PSST, HEY".lower()
whisper_hey
```

```
'psst, hey'
```

```
# Desired output: "Learn. Love. Code"
flatiron_mantra = "LEARN. LOVE. CODE.".title()
flatiron_mantra
```

```
'Learn. Love. Code.'
```

```
# Desired output: str
type_string = type("i'm a string")
type_string
```

```
str
```

```
# Desired output: list
type_list = type(["i'm", "a", "list"])
type_list
```

```
list
```

```
# Desired output: 3
length_of_list = len(["i'm", "a", "list"])
length_of_list
```

```
3
```

```
# Desired output: "list"
longest_word_in_list = max(["i'm", "a", "list"])
longest_word_in_list
```

```
'list'
```

```
# Desired output: 1
smallest_number = min([1, 3, 4, 78])
smallest_number
```

```
1
```

```
# Desired output: 11
sum_of_numbers = sum([1, 2, 3, 5])
sum_of_numbers
```

```
11
```

Uncomment the code in each cell as you start working on them. For example, when you begin working on the first cell, remove `#` at the start of each line.

> **Note:** The `cmd+?` keyboard shortcut comments or uncomments a given line of code!

Replace `[COMPARISON]`, with the correct comparison operator to get the desired output, which you will find as a comment at the end of each line. See the example below.

```
# boolean_compare = False [COMPARISON] True # True
=> boolean_compare = False != True # True
OR
=> boolean_compare = False != True
```

Once uncommented, you can check the output to see if your comparisons match the answers provided in the ending comments.

> **Remember** the comparison operators are: `==` , `!=` , `<` , `>` , `<=` , `>=`

```
boolean_compare = True != True # False
boolean_compare2 = False == True # False
print(boolean_compare, boolean_compare2)
```

```
False False
```

```
number_compare = 10 == 10 # True
number_compare2 = -20 <= 30 # True
number_compare3 = 4 > 5 # False
print(number_compare, number_compare2, number_compare3)
```

```
True True False
```

```
string_compare = "stacy" > "STACY" # True
string_compare2 = "hey i love python!" == "hi love python" # False
string_compare3 = "this string is bigger than the other" > "that is true" # True
print(string_compare, string_compare2, string_compare3)
```

```
True False True
```

In the next section, do not use either `==` or `!=` operators

```
list_compare = [0, 0, 0, 0] != [0, 0, 0] # True
list_compare2 = [1, 0, 0] > [0, 0, 0] # True
list_compare3 = [0, 0, 0] > [0, 0, 3] # False
list_compare4 = [0, 0, 3, 0] > [0, 0, 3] # True
list_compare5 = [0, 0, 4, 0] < [0, 0, 3] # False
print(list_compare, list_compare2, list_compare3, list_compare4, list_compare5)
```

```
True True False True False
```

Practicing Identity and Logical Operators

In this next section, use the identity and logical operators to get the desired output as you did in the examples above using the comparison operators.

> **Remember** the **logical operators** are `and` , `or` , & `not` ; and the **identity operators** are `is` & `is not`

Use logical operators for this section

```
logical_compare = 2 and [] # []
logical_compare2 = not [] # True
logical_compare3 = 0 and [] # 0
logical_compare4 = True and 2 # 2
logical_compare5 = 2 or 3 # 2
logical_compare6 = not True # False
logical_compare7 = False and 2 # False
print(logical_compare, logical_compare2, logical_compare3, logical_compare4,
      logical_compare5, logical_compare6, logical_compare7)
```

```
[] True 0 2 2 False False
```

Use identity operators for this section

```
a = []
b = a
identity_compare = {} is {} # False
identity_compare2 = a is b # True
identity_compare3 = b is not [] # True
identity_compare4 = 9 is not 10 # True
identity_compare5 = "Same" is not "Same" # False
identity_compare6 = [1,3,4] is [1,2,3] # False
print(identity_compare, identity_compare2, identity_compare3, identity_compare4, ide
```

```
False True True True False False
```

Summary

Great work! After all that, there's nothing we can't compare. Well, I guess apples and oranges might still be off the table. We practiced using comparison, logical, and identity operators in Python to compare elements of the same and different datatypes and/or values. Going forward, there will be plenty of instances where we will need to compare elements. So, it is important to have a good understanding of how each of these operators works. Don't worry, as with all concepts in programming, the more we work with something the better we understand it.

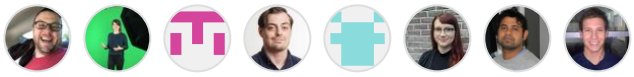
Releases

No releases published

Packages

No packages published

Contributors 8



Languages

- **Jupyter Notebook** 100.0%