Chapter 6 - Strings

Strings

String Comparison

Learning Objectives - String Comparison

- **Compare strings with `==` and `!=`**
- **Compare strings with `is` and `is not`**
- **Differentiate string comparisons with `==` and `is`**
- **Compare strings when the capitalizations are different**
- **Compare the alphabetical order of strings**

== & !=

Comparing with ==

The `==` operator can be used with strings just like it is with numbers or boolean values.

```
string1 = "It's Friday!"
string2 = "It's Friday!"
print(string1 == string2)
```

challenge

What happens if you:

- Change the value of `string1` to `"it's friday!"`?
- Change the value of `string2` to `"it\'s friday!"`?

Comparing with !=

You can also test for string inequality with the `!=` operator.

```
string1 = "It's Friday!"
string2 = "It's Monday."
print(string1 != string2)
```

challenge

What happens if you:

- Change the value of `string2` to `"It's Friday"`?
- Change the value of `string2` to `"It's Friday!"`?

Is & Is Not

Comparing with Is

Python uses the keyword `is` for comparison. It replaces `==`.

```
string1 = "It's Friday!"
string2 = "It's Friday!"
print(string1 is string2)
```

> challenge
>
> ## What happens if you:
>
> - Change `string1` and `string2` to the value `""`?
> - Change `string1` to the value `" "`?

Comparing with Is Not

If `==` can be replaced with `is`, then `!=` can be replaced with `is not`.

```
string1 = "It's Friday!"
string2 = "It's Monday."
print(string1 is not string2)
```

challenge

What happens if you:

- Change `string1` to `"\""` and change `string2` to `"""`?
- Change `string1` to `"\""` and change `string2` to `'"'`?

▼ **Why the EOF error?**

The string `“””` is actually a string that looks like this `"`. But if you put three double quotes in a row, that is starting a multi-line string. Python expects another triple quote to end the multi-line string. Python reached the end of the file (EOF) before finding the triple quote. That is why there is an error.

== vs. Is

Why Have Is and ==?

It may seem like `is` and `==` do the same thing, but there is a subtle difference between the two. Enter and run the code below.

```
a = 1
b = 1
print(id(a))
print(id(b))
```

You should see two identical numbers. These numbers are the object ID for the values stored in variables `a` and `b` (the `id` command returns the object ID). Each time you create a variable, Python takes some of the memory on your computer and reserves it for the variable. The more memory that is taken up by Python, the slower your program runs. Python increases performance by having two variables with the same value point to the same object in memory. Change your code to the following:

```
a = 1
b = 1
print(id(a))
print(id(b))
a += 1
print(id(a))
print(id(b))
```

Because the variable `a` has a different value, Python cannot use the same object ID. A new chunk of memory is used, which is why the object IDs are different.

Is and Object IDs

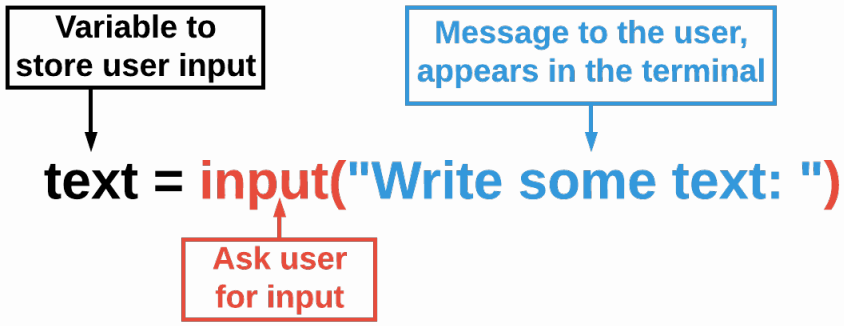
The `is` keyword compares object IDs, while `==` compares values. You should use `is` to compare strings and other objects (more on objects in another unit), and use `==` to compare floats and integers. The code below works, the `print` statements both return `True`. You will not get an error if you use `==` to compare strings. However, it is a good idea to get into the habit of use `is` for comparing strings and other objects.

```
string1 = "Hello"
string2 = "Hello"
print(string1 == string2)
print(string1 is string2)
```

Capitalization

User Input

To humans, the words “dog” and “Dog” are the same. Python, however, is case sensitive. That means a difference in capitalization implies two different words. When dealing with user input, the rules of capitalization are often not followed. The `input` command instructs Python to pause the program until the user has typed something and pressed `Return`. **Important**, Python treats all user input as a string. If you want a number, use typecasting to convert it to the proper data type.



User Input

▼ **The Terminal**

Collecting user input requires the terminal (also called the command line), which has not been used up until now. When you run the program below, you will see a new tab appear with the message to the user. Enter some text and press return. Click the “TRY IT” button to run the program again. If you want to edit your program, you can click on the tab with your code. You can also close the terminal (running the program again will launch the terminal).

```
text = input("Type something and press 'Return': ")
print(text)
```

challenge

What happens if you:

- Remove the space between the `:` and `"` in the `input` message?
- Press the `Ctrl` button and `C`?

▼ **Ctrl + C**

Pressing `Ctrl` and `C` on the keyboard will exit the program running in the terminal.

Comparing Text, Not Capitalization

You can compare the strings while ignoring capitalization by using the `lower` method. This will make all of the text lowercase. Make sure that both strings are lowercase for the comparison to work.

```
print("This program will check to see if two values are the same.")
string1 = input("Enter a value: ").lower()
string2 = input("Enter another value: ").lower()
if string1 == string2:
    print("They are the same!")
else:
    print("They are not the same.")
```

challenge

What happens if you:

- Enter `Texas` and `texas`?
- Enter `TeXaS` and `tExAs`?
- Change the `.lower()` to `.upper()` for both string variables?

▼ **Running Python code manually**

All the “TRY IT” button does is send a message to Codio to run your Python program. You can do the same from the terminal. The image below

explains how to run your code manually. If you see the `$` in the terminal, that means Python has finished running, and the terminal is waiting for the next command.

Alphabetical Order

Alphabetical Order

Comparing strings is more than determining if they are identical. You can also ask if a one string comes before another. Use the boolean operators `<` and `>` to determine alphabetical order.

```
string1 = "apple"
string2 = "cat"
if string1 < string2:
    print("{} comes before {}".format(string1, string2))
elif string1 is string2:
    print("{} is the same as {}".format(string1, string2))
else:
    print("{} comes after {}".format(string1, string2))
```

challenge

What happens if you:

- Change `string2` to `"apple"`?
- Change `string1` to `"zebra"`?
- Change `string2` to `"Zebra"`?