String Operations

by Sophia

WHAT'S COVERED

In this lesson, you will learn about how to modify strings using basic operations. Specifically, this lesson covers:

1. Common Operations
2. Simple Data Conversion

1. Common Operations

One of the most common operations for strings is the use of **concatenation**. The concatenation operator should look familiar to you as it is the + operator (plus sign). The **+ operator** is the addition operator on integers; however, it can also function as a concatenation operator when applied to string variables, which means it joins the variables by linking them end to end. An error will appear if trying to use this operator on variables that are a mix of string and integer.

So, another way to put it is that with strings, this operator can combine values together to form a new one. The act of adding one string to another with the + operator can be referred to as string concatenation.

When we concatenate the strings, they simply get placed together.

TRY IT

Directions: Enter this code and run it.

```
myVar1 = 'Learning'
myVar2 = 'Python'
result = myVar1 + myVar2
print(result)
```

This should be the concatenated output screen.

```
LearningPython
```

Notice that the strings are concatenated, but there isn't a space in between them. If we did want to add a space, we can simply add the space in the concatenation.

TRY IT

Directions: Try the code again, this time adding a space ' '.

```
myVar1 = 'Learning'
myVar2 = 'Python'
result = myVar1 + ' ' + myVar2
print(result)
```

Now with the space added, we see:

```
Learning Python
```

Let's see what happens if we have the variables as integers (without quotes around the values).

TRY IT

Directions: Try the code using two integers.

```
myVar1 = 111
myVar2 = 222
result = myVar1 + myVar2
print(result)
```

With the following output screen.

```
333
```

We'll see that the `myVar1` was added to `myVar2`, and the results of it as we expected. Now, let's try this out again as strings, by adding quotes around each of the values.

TRY IT

Directions: Now try the code changing the integers into strings.

```
myVar1 = '111'
myVar2 = '222'
result = myVar1 + myVar2
print(result)
```

The output screen of the string values.

```
111222
```

As expected, since `myVar1` and `myVar2` are now strings, even though they contain numbers, they are concatenated together and placed into the result variable.

Another operation with strings is the `*` (asterisk sign) or multiplication operator. The * **operator** is unique to Python as it can multiply integers and can also be used as a replication operator with strings and integers (floats will not work). As a replication operator, it will repeat (make a copy of) the variable based on the number of times it is multiplied by.

TRY IT

Directions: Try this code with the `*` operator.

```
myString = 'Python is great'
result = myString * 5
print(result)
```
Output screen all on one line.

```
Python is greatPython is greatPython is greatPython is greatPython is great
```
This may not be what we intended. Adding in the newline character will help us ensure that we have each string on a new line.

TRY IT

Directions: Try the code below with the newline character.

```
myString = 'Python is great\n'
result = myString * 5
print(result)
```
Output screen with the `\n` character.

```
Python is great
Python is great
Python is great
Python is great
Python is great
```

That is much better!

Python also provides a membership operator that can be used in strings. The **in operator** is a reserved keyword and used as a membership operator that checks if the first operand is contained in the second. If it is contained, it returns true; otherwise, it returns false. It can also be used to iterate through a sequence in a for loop. More on loops in later tutorials.

TRY IT

Directions: Let's try the `in` operator. Add the following code to the IDE.

```
stringToFind = 'some'
sentence = 'Python is quite a fun language to learn. There can be some great string functions to use.'
result = stringToFind in sentence
print(result)
```
We have `stringToFind` setup as a variable for the string that we want to find in the sentence. In this case, we are looking for the word 'some'.

```
True
```
It does exist, hence the function returns True.

If we had passed in a string that didn't exist, it would return False. Let's add a string we know is NOT in the sentence.

Directions: Now try this `stringToFind` that we know is not in there.

```
stringToFind = 'grrrr'
sentence = 'Python is quite a fun language to learn. There can be some great string functions to use.'
result = stringToFind in sentence
print(result)
```

As expected, that string was not found.

```
False
```

We looked at the use of escape characters to add in a newline. However, we can also assign multiline strings without using the newline character by using three double quotes or three single quotes.

TRY IT

Directions: Let's try using the three double quotes.

```
outputString = """This is my first line
and now my second line
and lastly my last line."""
print(outputString)
```

The output screen looks like:

```
This is my first line
and now my second line
and lastly my last line.
```

Directions: Now try with three single quotes.

```
outputString = '''This is my first line
and now my second line
and lastly my last line.'''
print(outputString)
```

The output screen with the single quotes.

```
This is my first line
and now my second line
and lastly my last line.
```

In either case, they output the same.

Just be careful in the IDE when you enter in a double quote or single quote, as the IDE automatically adds in the second quote when you enter in the first. In the next challenge, we'll explore more string functions and methods.

TERMS TO KNOW

Concatenation
The operation of joining or merging two or more strings together.

+
The `+` operator (plus sign) is the addition operator on integers; however, it can also function as a concatenation operator when applied to string variables, which means it joins the variables by linking them end to end. An error will appear if trying to use this operator on variables that are a mix of string and integer.

*
The `*` operator (asterisk sign) or multiplication operator is unique to Python as it can multiply integers and can also be used as a replication operator with strings and integers (floats will not work). As a replication operator, it will repeat (make a copy of) the variable based on the number of times it is multiplied by.

in
The `in` operator is a reserved keyword and used as a membership operator that checks if the first operand is contained in the second. If it is contained, it returns true; otherwise, it returns false. It can also be used to iterate through a sequence in a for loop.

---

2. Simple Data Conversion

We can use the `str` function to convert other data types to a string instead.

EXAMPLE
For example, let's take the previous integer example.

```
myVar1 = 111
myVar2 = 222
result = myVar1 + myVar2
print(result)
```

That has 333 as the output screen.

```
333
```

If those variables were already set as integers, we can convert them to a string using the `str` function.

TRY IT

Directions: Try the code below using the `str` function.

```
myVar1 = 111
myVar2 = 222
result = str(myVar1) + str(myVar2)
print(result)
```

Now we see the output screen as two strings concatenated.

```
111222
```

Likewise, if we had those variables as strings, we can convert them to an `int` using the `int` function.

It originally looked like this:

```
myVar1 = '111'
myVar2 = '222'
result = myVar1 + myVar2
print(result)
```

Just a long string again.

```
111222
```

Directions: Now try using the `int` function to convert the strings to integers and add them.

```
myVar1 = '111'
myVar2 = '222'
result = int(myVar1) + int(myVar2)
print(result)
```

As expected, the sum of the two integers.

```
333
```

SUMMARY

In this lesson, we learned about the **common string operations**, including concatenation, the asterisk operator, and the membership operator `in`. We also learned how to do some **simple data conversion** using the `str` and `int` functions.

Best of luck in your learning!

TERMS TO KNOW

*

The * operator (asterisk sign) or multiplication operator is unique to Python as it can multiply integers and can also be used as a replication operator with strings and integers (floats will not work). As a replication operator, it will repeat (make a copy of) the variable based on the number of times it is multiplied by.

+

The + operator (plus sign) is the addition operator on integers; however, it can also function as a concatenation operator when applied to string variables, which means it joins the variables by linking them end to end. An error will appear if trying to use this operator on variables that are a mix of string and integer.

Concatenation

The operation of joining or merging two or more strings together.

in

The in operator is a reserved keyword and used as a membership operator that checks if the first operand is contained in the second. If it is contained, it returns true; otherwise, it returns false. It can also be used to iterate through a sequence in a for loop.