Functions and Methods Introduction

by Sophia

WHAT'S COVERED

In this lesson, you will learn how to use functions and methods in Python. Specifically, this lesson covers:

1. Functions

A **function** is a section of code that runs when it is called. We have the ability to pass data into functions, through parameters, and this data can then be used within the function. After the function has been processed, it can return data as an option, although functions don't need to do so. We are already familiar with the print function, where we output variables like the following:

EXAMPLE

```
print('I\'ve used the print function before.')
```

The format of a function is the function name followed by parentheses, like:

```
myFunction()
```

Data can be passed into functions as arguments. These arguments are specified within the parentheses. You will see the terms parameter and argument used for information that is being passed into a function. A **parameter** is the actual variable name(s) when we define a function definition.

Note: You do not need to worry about creating functions now, but it's good to see how it is defined.

A function definition is the first line when we create a new function to be used in a program. An **argument** is the actual value(s) being passed into the function when it is called. We can add in multiple arguments as long as the function supports it. We would separate them by a comma.

↷ EXAMPLE

Let's look at an example to help clarify function items.

```
#first we define the function definition by using the reserved keyword "def" to define it
#then set up the function myFunction with three parameters firstName, middleInitial, and lastName.
#to end a function, we place a colon at the end

def myFunction(firstName, middleInitial, lastName):

#finally we will print to screen so we can see it work
  print(firstName, middleInitial, lastName)

#here is where we call myFunction and pass it the actual values as arguments
myFunction("John", "R", "Doe")
```

Here is the code snippet in code editor without comments:

```
def myFunction(firstName, middleInitial, lastName):
  print(firstName, middleInitial, lastName)
myFunction("John", "R", "Doe")
```

The output is as expected once we run the code.

```
John R Doe
```

Don't worry too much if you're still unsure about what all of the components of creating your own functions are, as we'll continue to cover them later on.

Note: When you do review Python documentation, you'll see the arguments often shortened as "args" in the documentation about Python.

There are many other built-in functions that we can use as part of the string. So, rather than immediately printing a string out, we can do other things. Let's look at a few functions.

1a. len() Function

We can calculate and print out the length or the number of characters in a string using the **len() function**.

↷ EXAMPLE

```
myString = 'Python is fun!'
print(len(myString))
```

Here is the output once the code is run.

```
14
```

The `len()` function counts all characters, including punctuation and spaces, which is why it prints out 14.

In past lessons, we've used data type functions to change variables. This is called type-casting, or casting. **Casting** is when you convert a variable's value from one data type to another. It is important to understand what they do since these functions allow us to convert data from one type to another so that we can perform the right operations on the data. For example, if we wanted to add 1 + 1, we want the result to be 2. However, if we wanted the result to be 11 using the same values, we would need to cast them to a string first so that the values are concatenated rather than added.

1b. int() Function

The **int() function** creates an integer number from an integer literal, a float literal (by removing all of the decimals), or a string literal (assuming the string represents a whole number).

EXAMPLE

```
print(int(100))
print(int(2.95))
print(int("200"))
```

Here is the output once the code is run.

```
100
2
200
```

It's important to note that the int for float removes the decimals rather than rounding the decimals. For example, the result of the int(2.95) is 2 and not 3.

1c. float() Function

The **float() function** can create a float number from an integer literal, float literal, or a string literal as long as the string represents a float or an integer.

EXAMPLE

```
print(float(100))
print(float(2.95))
print(float("200"))
```

Here is the output once the code is run.

```
100.0
2.95
200.0
```

1d. str() Function

The **str() function** creates a string from various data types, including strings, integer literals, and float literals.

EXAMPLE

```
print(str(100))
print(str(2.95))
print(str("200"))
```

Here is the output once the code is run.

```
100
2.95
200
```

TRY IT

Now that you have had a chance to see a few of the built-in functions in Python, try your hand at a few of them.

Directions: Using the IDE, try using the functions you just learned in the examples.

TERMS TO KNOW

Functions
A function is a piece of code that runs when it is called. We have the ability to pass in data into those functions, which are called parameters.

Parameter
A parameter is the actual variable name(s) when we define a function definition.

Argument
An argument is an actual value(s) that is being passed into the function when it is being called.

len()
The `len()` function counts all characters, including punctuation and spaces.

Casting (or Type-Casting) Casting is when you convert a variable's value from one data type to another.

int()
The `int()` function creates an integer number from an integer literal, a float literal (by removing all of the decimals), or a string literal (assuming the string represents a whole number).

float()
The `float()` function can create a float number from an integer literal, float literal, or a string literal as long as the string has a float or an integer.

str()
The `str()` function creates a string from various data types, including strings, integer literals, and float literals.

2. Built-In String Methods

We've looked at functions so far, but Python also has methods. Methods are similar to functions, but **methods** are a specialized type of procedure that is directly associated with an object. In Python, every item of data is an object. An **object** is an instance of a class that has properties and methods that are encapsulated or part of it. A class is a blueprint to create different objects. Again, don't worry, we will cover these things in detail in a later unit.

In the same way that a function is used, a method is called to perform a specific task. However, it is called by using the object name, a period, and the method name.

DID YOU KNOW

There are over 40 String Methods in Python! To learn more about what they are and their functions, check out: **Python String Methods**

That is a bunch of methods! We can't cover them all now, but we can show a few in the examples below.

2a. .index() Method

Using the **.index() method**, we can find the location of a character or a string within another string. The index method is called from the object, which is `myString` in this following example.

EXAMPLE

```
myString = 'Python is fun!'
print(myString.index("n"))
```

Here is the output once the code is run.

```
5
```

This method prints out 5 because the first occurrence of the letter "n" ends up being 5 characters away from the first character. This may seem confusing, as the first "n" is the sixth character. In most programming languages, including Python, the first position starts at 0 rather than 1. This means that the index of the last character of the string will be the length of the string minus 1.

Using the same string, this table will show the indices for each character in the string.

P	y	t	h	o	n		i	s		f	u	n	!
0	1	2	3	4	5	6	7	8	9	10	11	12	13

It is important to note that the index function only finds the first instance of the character. Even though the second "n" appears in the 12th index, 5 is returned.

2a. .capitalize() Method

Using the **.capitalize() method**, we can convert the first character of the string to an uppercase with all of the other characters to lowercase.

EXAMPLE

```
outputString = 'I AM YELLING!'
```

```
print(outputString.capitalize())
```
Here is the output once the code is run.

```
I am yelling!
```

2b. .lower() Method

The **.lower() method** will convert all of the characters of a string to lowercase, including the first character.

EXAMPLE

```
outputString = 'ThIs Is My TeSt StRiNg!'
print(outputString.lower())
```
Here is the output once the code is run.

```
this is my test string!
```

2c. .upper() Method

The **.upper() method** is the opposite of the `.lower()` method and converts all of the characters of a string to uppercase.

EXAMPLE

```
outputString = 'ThIs Is My TeSt StRiNg!'
print(outputString.upper())
```
Here is the output once the code is run.

```
THIS IS MY TEST STRING!
```

2d. .swapcase() Method

The **.swapcase() method** returns a copy of the string with the uppercase characters converted to lowercase, and the lowercase characters converted to uppercase.

EXAMPLE

```
outputString = 'ThIs Is My TeSt StRiNg!'
print(outputString.swapcase())
```
Here is the output once the code is run.

```
tHiS iS mY tEsT sTrIng!
```

2e. .title() Method

Finally, we have the **.title() method**, which returns a copy of the string with the first letter of each word converted to uppercase and the other characters converted to lowercase.

EXAMPLE

```
outputString = 'ThIs Is My TeSt StRiNg!'
print(outputString.title())
```

Here is the output once the code is run.

```
This is My Test String!
```

TRY IT

Now that you have had a chance to see a few of the built-in methods in Python, see if you can test some you just learned.

Directions: Using the IDE, try using the methods you just learned in the examples.

TERMS TO KNOW

Object An object is an instance of a class that has properties and methods that are encapsulated or part of it.

Methods Methods are a specialized type of procedure that is directly associated with an object and called to perform a specific task by using the object name, a period, and then the method name.

.index() The `.index()` method can find the location of a character or a string within another string.

.capitalize() The `.capitalize()` method will convert the first character of the string to an uppercase with all of the other characters to lowercase.

.lower() The `.lower()` method will convert all of the characters of a string to lowercase, including the first character.

.upper() The `.upper()` method is the opposite of the .lower() method and converts all of the characters of a string to uppercase.

.swapcase() The `.swapcase()` method returns a copy of the string with the uppercase characters converted to lowercase and the lowercase characters converted to uppercase.

.title() The `.title()` method returns a copy of the string with the first letter of each word converted to uppercase and the other characters converted to lowercase.

SUMMARY

In this lesson, we learned about some basic Python **functions** and their use. This included the `len()`, `int()`, `float()`, and `str()` functions. We were also introduced to a few of the Python **built-in string methods** including the `index()`, `capitalize()`,`lower()`, `upper()`, `swapcase()`, and `title()` methods. We will learn how to use these functions and methods in a future lesson.

Best of luck in your learning!

TERMS TO KNOW

.capitalize()

The .capitalize() method will convert the first character of the string to an uppercase with all of the other characters to a lowercase.

.index()

The .index() method can find the location of a character or a string within another string.

.lower()

The .lower() method will convert all of the characters of a string to lowercase including the first character.

.swapcase()

The .swapcase() method returns a copy of the string with the uppercase characters converted to lowercase and the lowercase characters converted to uppercase.

.title()

The .title() method returns a copy of the string where the first letter of each word is converted to uppercase with the other characters converted to lowercase.

.upper()

The .upper() method is the opposite of the .lower() method and converts all of the characters of a string to uppercase.

Argument

An argument is an actual value(s) that is being passed into the function when it is being called.

Casting (or Type-Casting)

Casting is when you convert a variable’s value from one data type to another.

Functions

A function is a section of code that runs when it is called. We have the ability to pass in data into those functions, which are called parameters.

Methods

Methods are a specialized type of procedure that’s directly associated with an object and called to perform a specific task by using the object name, a period, and then the method name.

Object

An object is an instance of a class that has properties and methods that are encapsulated or part of it.

Parameter

A parameter is the actual variable name(s) when we define a function definition.

float()

The float() function can create a float number from an integer literal, float literal, or a string literal as long as the string has a float or an integer.

int()

The int() function creates an integer number from an integer literal, a float literal (by removing all of the decimals), or a string literal (assuming the string represents a whole number).

len()

The len() function counts all characters, including punctuation and spaces.

str()

The str() function creates a string from various data types, including strings, integer literals, and float literals.