Lab - Python Classes Review (Instructor Version)

**Instructor Note**: Red font color or gray highlights indicate text that appears in the instructor copy only.

# Objectives

Part 1: Launch the DEVASC VM

Part 2: Review Functions, Methods, and Classes

Part 3: Define a Function

Part 4: Define a Class with Methods

Part 5: Review the circleClass.py Script

# Background / Scenario

In this lab, you review Python methods, functions, and classes. You then create a class and instantiate it several times with different values. Finally, you review the **Circle** class example used in the course.

# Required Resources

* 1 PC with operating system of your choice
* Virtual Box or VMWare
* DEVASC Virtual Machine

# Instructions

## Launch the DEVASC VM

If you have not already completed the **Lab - Install the Virtual Machine Lab Environment**, do so now. If you have already completed that lab, launch the DEVASC VM now.

## Review Functions, Methods, and Classes

In this part, you review the difference between functions and methods. You also review the basic structure of a class.

### What is a function?

As a quick review, recall that a function is an independently defined block of code that is called by name. In the following example, the function called **functionName** is defined and then called. Notice that it is an independent block of code. It is not encapsulated in any other code.

# Define the function

def functionName:

...blocks of code...

# Call the function

functionName()

### What is a method?

A method, however, cannot be called by itself. It is dependent on the object in which it is defined. In the following example, the class **className** is declared and three methods are defined. The class is instantiated and then each method of the class is called.

**Note**: This pseudo-code does not explicitly show the class constructor **\_\_init\_\_** method with the **self** variable. This special method is reviewed below.

# Define the class

class className

# Define a method

def method1Name

...blocks of code

# Define another method

def method2Name

...blocks of code

# Define yet another method

def method3Name

...blocks of code

# Instantiate the class

myClass = className()

# Call the instantiation and associated methods

myClass.method1Name()

myClass.method2Name()

myClass.method3Name()

## Define a Function

In this part, you will define a function with arguments and then call the function.

* + - 1. Open a new text file and save it as **myCity.py** in your **~/labs/devnet-src/python** directory.
      2. Define the function **myCity** with the argument **city** for city name. When the function is called with a specified city name, it prints a statement that includes the city name.

def myCity(city):

print("I live in " + city + ".")

* + - 1. Call the function **myCity** passing it different values for **city**, as shown in the following examples.

myCity("Austin")

myCity("Tokyo")

myCity("Salzburg")

* + - 1. Save and run the **myCity.py** file. You should get the following output.

devasc@labvm:~/labs/devnet-src/python$ **python3 myCity.py**

I live in Austin.

I live in Tokyo.

I live in Salzburg.

devasc@labvm:~/labs/devnet-src/python$

## Define a Class with Methods

In this part, you will define a class, use the \_\_init\_\_() function to define a method for the class, and then create instances of the class.

### Define and then instantiate a class with the \_\_init\_\_() method.

A Python class is used to create objects that have properties and methods. All Python classes typically include an explicitly defined **\_\_init\_\_()** function, although you can create a class without defining one. The **\_\_init\_\_()** function is always **init**iated when a class is instantiated. Instantiating a class creates a copy of the class which inherits all the class variables and methods.

**Note**: Although it is sometimes called the **\_\_init\_\_()** function, it is dependent on the class. Therefore, it is technically a method.

* + - 1. Open a new text file and save it as **myLocation.py**.
      2. Define a class with the name **Location** and press Enter. If you are working is VS Code, then the text editor should automatically indent four spaces.

class Location:

|<-- cursor should now be here

* + - 1. Next, define the **\_\_init\_\_()** function. By convention, the first parameter is called **self**. The **self** parameter is a reference to the current instance of the class itself and is used to access variables that belong to the entire class. The **\_\_init\_\_()** function is then assigned any variables the entire class needs. In the following example, define a name and country variable. Press Enter twice and then backspace twice to the left margin.

def \_\_init\_\_(self, name, country):

self.name = name

self.country = country

|<-- cursor should now be here

* + - 1. You can test that this class is now ready to use. Instantiate the class by assigning it a name of your choice. Then specify the values for the required class variables **name** and **country**. The following example uses the Location class to instantiate a class called **loc** with a **name** and **country** specified by you. Use your name and country.

loc = Location("Your\_Name", "Your\_Country")

* + - 1. To verify that the instantiated **loc** class now has your assigned name and country, add print statements to your script.

print(loc.name)

print(loc.country)

* + - 1. To verify the loc is indeed a class, add the following print statement that will print the data type for **loc**.

print(type(loc))

* + - 1. Save and run your script. You should get the following output except with your supplied name and country.

devasc@labvm:~/labs/devnet-src/python$ **python3 myLocation.py**

Your\_Name

Your\_Country

<class '\_\_main\_\_.Location'>

devasc@labvm:~/labs/devnet-src/python$

### Add a method to the Location class.

Now add a method to the **Location** class that can be called by a programmer when the class is instantiated. In this simple example, create a method to print the statement, “My name is [name] and I live in [country].”

* + - 1. Delete the code the begins with the instantiation of the **loc** class. Your **myLocation.py** script should now only include the following code.

class Location:

def \_\_init\_\_(self, name, country):

self.name = name

self.country = country

* + - 1. With your cursor at the end of the line **self.country = country**, press the Enter key twice and backspace once.

self.country = country

|<--Your cursor should be here

* + - 1. Define a new method call **myLocation** and assigned it the **self** parameter so that the new method can access the variables defined in the **\_\_init\_\_()** function. Then, define a print statement to print out the string specified above.

**Note**: The print statement should be on one line.

def myLocation(self):

print("Hi, my name is " + self.name + " and I live in " + self.country + ".")

* + - 1. Press the Enter key twice and backspace twice.
      2. Save and run your script to make sure there are no errors. You will not get any output yet.

### Instantiate the Location class multiple times and call the myLocation method.

Now that you have a class, you can instantiate it as many times as you like providing different values for the class variables each time.

* + - 1. Add the following code to your **myLocation.py** script to instantiate **Location** class and call the method. You do not need to add the comments.

# First instantiation of the class Location

loc1 = Location("Tomas", "Portugal")

# Call a method from the instantiated class

loc1.myLocation()

* + - 1. Save and run your script. You should get the following output.

devasc@labvm:~/labs/devnet-src/python$ **python3 myLocation.py**

Hi, my name is Tomas and I live in Portugal.

devasc@labvm:~/labs/devnet-src/python$

* + - 1. Add two more instantiations and then a fourth one where you specify the name and values for **your\_loc**.

loc2 = Location("Ying", "China")

loc3 = Location("Amare", "Kenya")

loc2.myLocation()

loc3.myLocation()

your\_loc = Location("Your\_Name", "Your\_Country")

your\_loc.myLocation()

* + - 1. Save and run your script. You should get the following output.

devasc@labvm:~/labs/devnet-src/python$ **python3 myLocation.py**

Hi, my name is Tomas and I live in Portugal.

Hi, my name is Ying and I live in China.

Hi, my name is Amare and I live in Kenya.

Hi, my name is Your\_Name and I live in Your\_Country.

devasc@labvm:~/labs/devnet-src/python$

### Review the complete myLocation.py script.

If you had any errors with your script, review the following example which includes all the code used in this part.

# Define a class with variables for \*\*name\*\* and \*\*country\*\*.

# Then define a method that belongs to the class. The method’s

# purpose is to print a sentence that uses the variables.

class Location:

def \_\_init\_\_(self, name, country):

self.name = name

self.country = country

def myLocation(self):

print("Hi, my name is " + self.name + " and I live in " + self.country + ".")

# First instantiation of the Location class

loc1 = Location("Tomas", "Portugal")

# Call a method from the instantiated class

loc1.myLocation()

# Three more instantiations and method calls for the Location class

loc2 = Location("Ying", "China")

loc3 = Location("Amare", "Kenya")

loc2.myLocation()

loc3.myLocation()

your\_loc = Location("Your\_Name", "Your\_Country")

your\_loc.myLocation()

## Review the circleClass.py Script

The example in the course shows how to create a class that calculates the circumference of a circle and then print out the calculated value. There are a few things to note in this script.

* The class includes three methods including the **\_\_init\_\_()** function. The **\_\_init\_\_()** function provides a method for entering the radius value.
* The **circumference** method calculates the circumference and returns the value storing it in the **circumferenceValue** variable.
* The **printCircumference** method prints a string. Notice that the variables are casted as strings with the **str()** function. Otherwise, the print statement would throw an error because **self.radius** and **myCircumference** are not strings.
* The Circle class instantiated three times.

# Given a radius value, print the circumference of a circle.

# Formula for a circumference is c = pi \* 2 \* radius

class Circle:

def \_\_init\_\_(self, radius):

self.radius = radius

def circumference(self):

pi = 3.14

circumferenceValue = pi \* self.radius \* 2

return circumferenceValue

def printCircumference(self):

myCircumference = self.circumference()

print ("Circumference of a circle with a radius of " + str(self.radius) + " is " + str(myCircumference))

# First instantiation of the Circle class.

circle1 = Circle(2)

# Call the printCircumference for the instantiated circle1 class.

circle1.printCircumference()

# Two more instantiations and method calls for the Circle class.

circle2 = Circle(5)

circle2.printCircumference()

circle3 = Circle(7)

circle3.printCircumference()

End of document