Data science and Machine learning with Python

Designed by Abdur Rahman Joy - MCSD, MCPD, MCSE, MCTS, OCJP, Sr. Technical Trainer for VFX at IDB BISW (Scholarship program), and C#.net, R, Scala, Kotlin, JAVA, Android/IOS/Windows Mobile Apps, SQL server, Azure, Oracle, SharePoint Development, AWS , CEH, KALI Linux, Python, Data Science, Machine Learning ,Software Testing, Graphics, Multimedia and Game Developer at Joy Infosys and other premises like BITM, SkillsJob, PNTL, Leads Training and New Horizon inc , Cell #: +880-1712587348, email: jspaonline@gmail.com. Web URL: http://www.joyinfosys.com/me.

Function:

Function is a special command block, it is named and it helps program code to be more easily to read, and it is called to use in different places in the program. The function is a reuseable command block.

Syntax:

```
def functionName( parameters ):
   "Short description of the function"
   codes ...
   return [expression]
```

- A function starts with word "def" (define) and followed by the function's name.
- Next, it is the list of parameters in the brackets () and colon (:), the function can include 0, 1 or parameters, the parameters are separated by comma.
- The first line of the function body is a short description of the function (optional).

return statement:

The **return** statement is used to returns a value (or an expression), or "nothing". When the **return** statement is run, function will end. **Return** is a statement not required in the function body.

Example	Description
return 3	Function return a value, and ends
return	Function return nothing, and ends

Parameters:
Function includes 0, 1 or parameters separated by comma. There are four types of parameters:

1. Required Parameter
2. Default parameter
3. Variable-Length Parameter
4. Keyword Parameter

Function examples

For example, a function has a parameter, and returns "Nothing".

```
# Define a function:
def sayHello(name)  :

   # If name is empty or null.
   if not name   :
      print( "Hello every body!" )

   # If name is not empty and not null.
   else :
      print( "Hello " + name)

# Call function, pass parameters to function.
sayHello("")
```


Designed by Abdur Rahman Joy - MCSD, MCPD, MCSE, MCTS, OCJP, Sr. Technical Trainer for VFX at IDB BISW (Scholarship program), and C#.net, R, Scala, Kotlin, JAVA, Android/IOS/Windows Mobile Apps, SQL server, Azure, Oracle, SharePoint Development, AWS , CEH, KALI Linux, Python, Data Science, Machine Learning ,Software Testing, Graphics, Multimedia and Game Developer at Joy Infosys and other premises like BITM, SkillsJob, PNTL, Leads Training and New Horizon inc , Cell #: +880-1712587348, email: jspaonline@gmail.com. Web URL: http://www.joyinfosys.com/me.

```
sayHello("Python");

sayHello("Java");
```
Output:
Hello every body!
Hello Python
Hello Java

Next is the example of a function that has a parameter and returns the value.

```
# Define a function:
def getGreeting(name)  :

    # If name is empty or null (None).
    if not name   :
        # Returns a value.
        # And function ends here.
        return "Hello every body!"

    # If name is not empty and not null (not None),
    # this code will be executed.
    return "Hello " + name

# Call function, pass parameters to function.
greeting = getGreeting("")

print(greeting)

 greeting = getGreeting("Python")

print(greeting)
```
Output:
Hello every body!
Hello Python

Function with required parameters

The following example defines the **showInfo** function having 2 parameters. Both of the parameters are required. When you call for the function, you need to provide 2 parameters to the function. Conversely program will throw an error.

```
def showInfo(name, gender):

    print ("Name: ", name);
    print ("Gender: ", gender);


# Valid
showInfo("Tran", "Male")

# Invalid ==> Error!!
showInfo("Tran")
```

Output:

```
Hello every body!
Hello Python
```

Function with default parameters

Function can have a lot of parameters, including the required parameters and ones with default values.
The **showInfo** function below has three parameters ***(name, gender = "Male", country = "US")***:

1. **name** is a required parameter.
2. **gender** is the parameter with default value "Male".
3. **country** is a parameter with the default value "US".

```
def showInfo(name, gender = "Male", country ="US"):

    print ("Name: ", name)
    print ("Gender: ", gender)
    print ("Country: ", country)


# Valid
showInfo("Aladdin", "Male", "India")

print (" ------ ")

# Valid
showInfo("Tom", "Male")

print (" ------ ")

# Valid
showInfo("Jerry")
```

```
print (" ------ ")

# Valid
showInfo(name = "Tintin", country ="France")

print (" ------ ")
```

```
output:
Name:  Aladdin
Gender:  Male
Country:  India
 ------
Name:  Tom
Gender:  Male
Country:  US
 ------
Name:  Jerry
Gender:  Male
Country:  US
 ------
Name:  Tintin
Gender:  Male
Country:  France
 ------
```

Modules in Python

Function with Variable-Length parameter

Variable-length Parameter is a special parameter. When calling for function, you can pass 0, 1 or values to that parameter.

Note: **"Variable-length Parameter"** must always be the last parameter of the function.

Example:

sumValues function has three parameters:

- Parameters **a, b** are required.
- Parameters ***others** are **"Variable-Length Parameters"**.

```
def sumValues(a, b, *others):

    retValue = a + b

    # 'others' parameter like an array.
    for other in others :
       retValue = retValue + other

    return retValue
```

save it to: variableLengthParameterExample

How to call for functions:

```
from variableLengthParameterExample import sumValues

# Pass: *others = []
a = sumValues(10, 20)

print("sumValues(10, 20) = ", a);

# Pass: *others = [1]
a = sumValues(10, 20, 1);


print("sumValues(10, 20, 1 ) = ", a);

# Pass: *others = [1,2]
a = sumValues(10, 20, 1, 2);


print("sumValues(10, 20, 1 , 2) = ", a);

# Pass: *others = [1,2,3,4,5]
a = sumValues(10, 20, 1, 2,3,4,5);


print("sumValues(10, 20, 1, 2, 3, 4, 5) = ", a);
```

Output:

```
sumValues(10, 20) =  30
sumValues(10, 20, 1 ) =  31
sumValues(10, 20, 1 , 2) =  33
sumValues(10, 20, 1, 2, 3, 4, 5) =  45
```