2020

Free Python Course

O. Elijah
onlinepythonclass@gmail.com

1 FUNCTION AND MODULE

In this Chapter, we shall look at the example and application of function and module in Python. These include what code reuse means, types of functions and modules, how to create functions and modules.

1.1 Introduction to code reuse

Code reuse is a very important part of programming in any language and python is not an exception. The disadvantages of increasing code size make code:

- hard to maintain
- take longer time to debug
- consume extra computing resources which causes delay

For a large programming project to be successful, it is essential to abide by the **don't repeat yourself** or **DRY** principle**.** This is the reason **function** became an essential part of programming.

1.2 Function

What is Function? From the previous lesson, you've already used functions such as print, range, int as shown in the following examples :

```
print("Python")
range(0,20)
int("20")
```

Function is defined as "Any statement that consists of a word followed by information in **parentheses**". For example, "print" is a legit predefined function in python and the parentheses or "()" after the print make it a function. There are two main types of functions:

1. **Predefined function** – Function that predefined by python (e.g print(), int(), range() and etc
2. **User-define function** – Function that is defined (created) by user/programmer

Defining a function

You can define functions to provide the required functionality. Here are simple rules to define a function in Python.

- Function blocks begin with the keyword **def** followed by the function name and parentheses (( )).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.
- The code block within every function starts with a colon (:) and is indented.
- The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Syntax

```
def functionname( parameters ):
   "function_docstring"
   function_suite
   return [expression]
```

In this section, we shall focus on user-define function only. Also, we shall cover the following topics:

1. Function without argument
2. Function with argument
3. Function with return
4. Function as object

Note: When creating functions do not use key words in Python as name for your functions. See Section 1.3 for examples of keywords.

1.3 Function without argument

In this example, we shall create a function named `my_func()`. It takes no arguments and prints "Hello" three times and call the function.

```
def my_func():
    "This function prints hello word three times"
    print("hello")
    print("hello")
    print("hello")
```

To call the function type the `my_func()` on your command shell to see the results as follows

```
#Calling my_func()
```

```
my_func()
```
Results
```
hello
hello
hello
```

A function must be call in order to execute it. In conventional way, if we want to print "Hello" 12 times, we can write the code as follow

```
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
```

Alternatively, we can make it the code shorter by using our defined function `my_func()` that print 3 "Hello" and call it four times. Note open a new script file to run following code.

```
def my_func():
    print("hello")
    print("hello")
    print("hello")
my_func()
my_func()
my_func()
my_func()
```

Results

```
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
print("hello")
```

As you have seen, we reduced 4 line of code and it gives us the same result. Of course, the coding above doesn't not serve any purpose and we will see more in the next section. Please note that, **YOU MUST DEFINE FUNCTION BEFORE THEY ARE CALLED,** the code below will give u a "name error".

```
my_func()


def my_func():
    print("hello")
    print("hello")
    print("hello")
```

Results

NameError: name 'my_func' is not defined

1.4 Function with argument

In the previous section, all the function we've looked so far is a function with **zero argument** or **without argument** or **empty parentheses**. However, most function take arguments especially pre-defined argument like `print('Hello')` and `'Hello'` is an input argument of the function. The example below define a function with argument

```
def mix_exclamation(word):
    print(word + '!')

mix_exclamation('spam')
mix_exclamation('this')
mix_exclamation('world')
```

Results

```
spam!
this!
world!
```

Argument is defined INSIDE the parentheses can be any variable. For instance, the argument called 'word', its role is a middle-man who pass the message (value) to the function. Moreover, you also can define two or more arguments by separating them with commas.

Example: Lets create a `AddNum(x,y)` function to take two arguments and add them

```
def AddNum(x,y):
    z = x + y
    print(z)
#Calling my_AddNum()

AddNum(5,9)
```

Results

However, the variable inside the function cannot reference outside. The following code gives you a 'Name error'

```
def AddNum(x,y):
      z = x + y

#Calling my_AddNum()
AddNum(5,9)
```

Result

14

1.5 Function with return

In section 5.4, we discussed function with argument. However, we may want to extract the value inside the function to be used in the main code or another function. One of the ways is to use **return** statement inside the function. The example of function with return is shown as below

```
def AddNum(x,y):
    z = x + y
    return z
#Calling my_AddNum()
k = AddNum(5,9)
print(k)
```

More examples, like comparing two numbers to get the maximal

```
def max_(x,y):
    if x > y:
        return x
    else:
        return y
#Calling max_ ()
print(max_(4,9))
```

Results

9

Note that, the code after **return** statement will not be executed. For example

```
def AddNum(x,y):
    z = x + y
    return z
```

```
    print("this will not execute")

#Calling my_AddNum()
k = AddNum(5,9)
print(k)
```

Results
14

Please be reminded that, try not to create the **SAME FUNCTION NAME AS THE PREDEFINE FUNCTION LIKE** `print()`, you will destroy the function inside the script. (I dare you to try)

1.6 Function as object

Compare to previous section, this section will definitely blow your mind off. You have to spend some time to understand it. It is quite useful if you fully understand the basic concept. Function just like **any other kind of value (Hint!),** they can be assigned and reassigned to variable and later reference by those name. The example below shows the example of function as an object

```
def multiply(x,y):
    return x*y

a = 3
b = 7
operation = multiply
print(operation(a,b))
```

As you can see, we define the function name into the variable called 'operation'. The name 'operation' is used to call the function. Since the function can be used as an object. By the way, still remember the argument? Yes, function can be used as the argument too. Now, let me give u a mind-blowing example

```
def add_(x,y):
    return x+y

def add_another_time(func,x,y):
    return func(func(x,y),func(x,y))

#Calling add_another_time()

print(add_another_time(add_,10,5))
```

Want more examples? Let's go another round, shall we?

```
def factorial(ops,n):
    if n == 0:
        return 1
    else:
        return n*ops(ops,n-1)

k = 5
operation = factorial
print(operation(operation,k))
```

The above example is a function within a function

return 5*factorial(4) --> 5*4*factorial(3) --> 5*4*3*factorial(2) --> 5*4*3*2*factorial(1) --> 5*4*3*2*1*factorial(0) --> 5*4*3*2*1*1

Results

120

The above code can be simplified to

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n*factorial(n-1)

k = 5
operation = factorial
print(factorial(5))
```

Enjoy the looping and jumping? Okay, we are going to proceed to '**module**' in next section.

1.7 Module

Module is piece of code that other people have written to fulfil common task. From last chapter, '`import time`' where time is a module which is written by other people. The basic way to use module is to add `'import module_name'` at the top of your code. Then use `module_name.var` to access the function (var is the function inside the module). Below shows an example on creating a random number from 1 to 10

```
import random

value = random.randint(1,10)
print(value)
```

From the code, `random` is a module written by other people (actually is a python script called random.py hidden somewhere else). Inside the `random` module, there is one function called `randint`. The purpose of this function is to randomly return a range of integer by input two arguments.

There are other kinds of `import` that can be used if you only need certain function inside the module. The format is `'from module_name import var'`. Similar to previous explanation, var is a function. Let modify the code above

```
from random import randint

value = randint(1,10)
print(value)
```

Results

5

Note the function `randint(1,10)` will generate a random integer between 1 and 10

The difference is we do not have to put "random." as we had imported a specific function from the module. Please note that, import an unknown module or undefined module will give you an 'Import error', for example

```
import Elijah

ModuleNotFoundError: No module named 'elijah'
```

Moreover, you can change the name of the imported function. Reusing the code above, we are modify the 'randint' to 'justRandomInteger'. The format is `'from module_name import var as name'`. The 'name' is up to you to define.

For example,

```
from random import randint as justRandomInteger

value = justRandomInteger(1,10)
print(value)
```

Results

6

Before we end this chapter, there are three main types of module

- Module predefined or standard library in python (like 'random', 'math', 'time','os',socket'……..i lazy to list it all, find it by yourself)

- Module written by yourself
- Module from 3rd-party (external module download from internet or PyPI https://pypi.python.org/pypi)

By following previous example, we had encountered the predefined module. How about module that defined by yourself? First thing first, you require creating a module file called 'testme.py' and the code is as follow

```
#testme.py
def add_something(x,y):
    return x + y
```

Now, we are creating another python file called 'gettestme.py' to import the module file we just predefined (make sure 'testme.py' and 'gettestme.py' are in the same directory).

```
#gettestme.py
from testme import add_something

value = add_something (4,11)
print(value)
```

As you can see, 'gettestme.py' try to import the function `add_something()` from 'testme.py'. The purpose of module is to reuse the code.

PyPI a.k.a python package index is a repository currently contains 131,533 (as on 8/3/2018) packages for your to discover. The best way to download it is using pip (google it!!).

1.8 Application

Example 1

We would like to write a python code using function to solve the Example 3 in Chapter 3

A domestic electricity usage will charge with basic rate (kWh) #220 (Naira Nigerian currency). Calculate the bill by giving the power consumed (kWh). Print the total amount bill and if it is less than or equal to #2000, print "Your bill is free".

First, we need to define the function to calculate the electricity bill. Second, we write the main program to call the defined function.

The python code is as follows

```
#Example 1 Program to compute electricity bill


#user defined function to calculate the bills

def cal_tariff(amount): # function with argument or input parameter

    bill = kWh* tariff

    return bill         # return  the result to calling function


tariff = 200

kWh = int(input("enter total consumption (kWh): " ))

cost = cal_tariff(kWh)  # call the function

print("Your total bill is #", cost)

if (cost <=2000):

      print("\nYour bill is free!!!")
```

Result

```
enter total consumption (kWh): 5
Your total bill is # 1000

Your bill is free!!!
```

Example 2

Now let us write a program to create a user menu to (1) - add, (2) - multiply and (3) - exit from Example 2 in Chapter 3

First, we need to create our user defined function

1. To display menu
2. To add
3. To multiple

Second, we write the main program to call the functions

The python code is as follows

```
# Example 2 function without argument
# function to display the Program menu
def display_menu():  # user defined function
    print(" ********** MENU  **************")
    print("  (1) Addition                  ")
    print("  (2) Multiplication            ")
    print("  (3) Exit                      ")



#function to compute the sum of two numbers
def mysum(a,b):     # user defined function
    result = a+b
    return result
#fucntion to compute the multiplication of two numbers
def mymultiplication(a,b):   # user defined fucntion
    result = a*b
    return result


# main program
display_menu()  # calling function
menu = int(input(" Enter the option from the MENU  "))
while (menu !=3):
    first_num = int(input("Enter first number: "))
    second_num = int(input("Enter Second number: "))
    if (menu == 1):
        total = mysum(first_num, second_num)  # calling function
        print(first_num, " + ", second_num ,"= " ,total )
    else:
```

```
        total =  mymultiplication(first_num,second_num) # calling fucntion
        print(first_num, " x ", second_num, "= " ,total )


    display_menu()  # calling function
    menu = int(input("Enter the option from the MENU  "))
print("Thank you and bye!")
```

Result

```
enter total consumption (kWh): 5
Your total bill is # 1000

Your bill is free!!!
```

1.9 Reference

[1] https://code.tutsplus.com/tutorials/python-from-scratch-functions-and-modules--net-21045

[2] https://www.tutorialspoint.com/python/python_functions.htm

[3] https://www.tutorialspoint.com/python/python_modules.htm

Thanks to Kelvin for helping to preparing this material.

1.10 Exercise

1) What is DRY? Why is function an essential part of programming?

2) Create a simple calculator with add, subtract, multiply and divide using a menu. The menu should have the following

********** MENU **********

(1) Add
(2) Sub
(3) Multiply
(4) Divide
(5) Exit
****

The calculator should be able to compute two or three different numbers.

Example

2+ 2 + 3 = 7

5 x 5 x 5 = 125

7 – 1 – 2 = 4

16 /2/2 = 4

3) Practice the following codes to import the module Matplotlib

```
from matplotlib import pyplot as plt

x = range(1,10)

y = range(31,40)

plt.plot(x,y)

plt.show()
```

Now use the following example of the plot function above to plot the maths scores and english scores of the students in the following table. Refer to your chapter 3

1. The test score of Class A students are given as follows

Names	Maths Score	English score
John	67	98
Annis	88	80
Fatimah	78	85
Caleb	75	89
Aruna	60	93