Programming Fundamentals

Lecture 6 – Functions

Iresh Bandara

Learning Outcomes

- This lecture addresses LO1,LO2 and LO4 for the module
- On completion of this lecture, students are expected to explain and apply
  - Python functions with and without arguments
  - Return values from the functions
  - Global and local scope

Agenda

- Python function – introduction
- Functions with and without arguments
- Variations of passing arguments
- Return values
- Global and local variables
- Pass by value and reference

Python functions

- Function is a block of statements which runs when called
- May or may not take arguments to process and may or may not return results
- Function definition : function name + parameters list + statements function contain
- Advantages of using functions
  - Reusability : Can call the function more than once. Need to edit the logic once only
  - Clarity : Separated logic and easy to read

Functions and Arguments

```
#no arguments
def print_something():
     print(“Hello world”)

print_somthing() #function call
print_somthing() #call again
```

```
#single argument
def print_name(name):
     print(“Hello ”+ name)

print_name(“alex”) #function call
print_name(“jack”) #call again
```

```
#multiple arguments
def print_name(f_name,l_name):
     print(“Hello ”+ f_name + “ ”+ l_name)

print_name(“alex”,”peterson”) #function call
print_name(“john”,”doe”) #call again
```

ROBERT GORDON UNIVERSITY ABERDEEN | INFORMATICS INSTITUTE OF TECHNOLOGY

Exercise 1

- Flow of execution. Trace the problem?
- Solution ?

```
fucntion2()
def function1():
    print("function1")
def fucntion2():
    function1()
    print("function2")
```

Functions and Arguments contd..

```
#Keyword arguments
def my_function(arg1, arg2, arg3):
       print("first arg is" + arg1)

my_function(arg1 = ″1", arg2 = "2",
arg3 = "3")
```

```
#If args number unknown
def my_function(*args):
       print("first arg is" +
args[0])

my_function("1″,″2″,″3″)
```

```
#Default parameter value
def my_function(arg1="default″):
       print("first arg is" + arg1)

my_function("1″)
my_function()
```

Function that return

- Void functions do not have return values
- When a function is returning a value, it needs to be saved (res variable).

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


res=addition(3,4)
print(res)#print 7
```

```
def string_concat(para1,para2):
        return para1+para2

print(string_concat("Hello","World"))
#print HelloWorld
```

Global and local scope

- Local variables : defined within a function
- Global variables : available throughout the program

```
total = 0
def sum( arg1, arg2 ):
        total = arg1 + arg2;
        print(total) #total 30
        return total;

sum( 10, 20 );
print(total) #total 0
```

Local variable

Global variable

Pass by value and reference

- Immutable objects (float, int, string): pass by value.
  - Cannot retrieve the changes that were done inside a function
- Mutable objects (list, dictionaries) : pass by reference
  - Can retrieve the changes that were done inside a function

```
#pass by objects
def string_concat(para):
        para="changed"
#return

para="original"
string_concat(para)
print(para)
#output is “original”
```

```
#pass by reference.
def string_concat(para):
        para[0]="changed"

para = “original”
para_list=[para]
string_concat(para_list)
print(para_list[0])
#output is “changed”
```

Exercise 2

What will be the output of the following program?

```
def f():
    #city = "Munich"
    def g():
        global city
        city = "Zurich"
    print("Before calling g: " + city)
    print("Calling g now:")
    g()
    print("After calling g: " + city)

city = "Stuttgart"
f()
print("'city' in main: " + city)
```

Summary

- Functions = function name + args list + body
- Needs to define before use/call
- Functions are defined with or without arguments
- Several variations: multiple arguments/keyword args/default values/unknown number of args
- Functions can return an output after the execution.
- Global variables: can access throughout the program
- Difference between pass by value and reference were discussed
- Local variables: only inside the function
- **global** is used to define a variable inside a func, and thereafter in global scope