LAB MANUAL 7

```
# # Python built-in Functions
'''for i in range(1, 10):
    print(i, end=' ')
# Output 1 2 3 4 5 6 7 8 9'''




# function
'''def message():
    print("Welcome to PYTHON MODULE")
    for i in range(1, 10):
        print(i, end=' ')


# call function using its name
message()
message()
message()
message()'''

#Creating a function with parameters

# function
'''def course_func(name, course_name):
    print("Hello", name, "Welcome to PYnative")
    print("Your course name is", course_name)

# call function
course_func('John', 'Python')'''

#Creating a function with parameters and return value

# function
'''def calculator(a, b):
    add = a + b
    print(a, b)
    # return the addition
    return add

# call function
# take return value in variable
returnfromadd = calculator(22, 66)

print("Addition :", returnfromadd)
calculator(22, 66)'''

#Calling a function

# function
'''def even_odd(n):
    # check numne ris even or odd
    if n % 2 == 0:
        print('Even number')
    else:
        print('Odd Number')

# calling function by its name
even_odd(45)
```

```
even_odd(90)

even_odd(99)'''

#

# import randint function
'''from random import randint

# call randint function to get random number
print(randint(10, 20))'''

#single line docstring
'''def factorial(x):
    """This function returns the factorial of a given number."""
    return x

# access doc string

print(factorial.__doc__)'''


# Multi-line Docstring
'''def any_fun(parameter1):
    """
       Description of function

       Arguments:
       parameter1(int):Description of parameter1

       Returns:
       int value
    """


print(any_fun.__doc__)'''

# Return outcome of the function

'''def is_odd(list1):
    odd_num = []
    for n in list1:
        if n % 2 == 1:
            odd_num.append(n)
    # return a list
    return odd_num

# Pass list to the function
odd_num = is_odd([2, 3, 42, 51, 62, 70, 5, 9, 5, 7,8, 9, 10, 14])
print("Even numbers are:", odd_num)'''


# Multiple return values
'''def arithmetic(num1, num2):
    add = num1 + num2
    sub = num1 - num2
    multiply = num1 * num2
    division = num1 / num2
    # return four values
    return add, sub, multiply, division
```

```
# read four return values in four variables
a, b, c, d = arithmetic(10, 2)


print("Addition: ", a)
print("Subtraction: ", b)
print("Multiplication: ", c)
print("Division: ", d)'''

# The pass statement for function
'''def addition(num1, num2):
    # Implementation of addition function in comming release
    # Pass statement
    pass'''


# Varibale scope and areas

'''global_lang = 'DataScience'

def var_scope_test():
    local_lang = 'Python'
    a = "hello inside the function"
    print(local_lang)
    print(a)

var_scope_test()
# Output 'Python'

# outside of function
print(global_lang)'''
# Output 'DataScience'''

# NameError: name 'local_lang' is not defined


'''def function1():
    # local variable
    loc_var = 888
    print("Value is :", loc_var)

def function2():

    print("Value is :", loc_var)

function1()
function2()'''

# Global Variable use
'''global_var = 999

def function1():
    print("Value in 1nd function :", global_var)

def function2():
    print("Value in 2nd function :", global_var)

function1()
function2()'''
```

```
# Global variable
'''global_var = 5

def function1():
    print("Value in 1st function :", global_var)

def function2():
    # Modify global variable
    # function will treat it as a local variable
    global_var = 555
    print("Value in 2nd function :", global_var)

def function3():
    print("Value in 3rd function :", global_var)

function1()
function2()
function3()'''

#As you can see, function2() treated global_var as a new
# variable (local variable). To solve such issues or access/modify
# global variables inside a function, we use the global keyword.

# Global variable
x = 5

# defining 1st function
def function1():
    print("Value in 1st function :", x)

# defining 2nd function
def function2():
    # Modify global variable using global keyword
    global x
    x = 555
    print("Value in 2nd function :", x)

# defining 3rd function
def function3():
    print("Value in 3rd function :", x)

function1()
function2()
function3()
```