Lesson-6: Functions

Program: 1

```
def got_called( ):
    print("Hello")

got_called( )
```

Output:

Program: 2

```
def got_called( ):
    print("Hello")
    return 5

r = got_called()
print(r)
```

Output:

Program: 3

```
def got_called( ):
    print("Hello")
    return

r = got_called( )
print(r)
```

Output:

Program: 4

```
def got_called(a,b):
    print("Hello")
    return (a+b)

a = 5
b = 6
r = got_called(a,b)
print(r)
```

Output:

Program: 5

```
def got_called(b,a):
        print("a = ", a)
        print("b = ", b)
        return (a+b)

a = 5
b = 6
r = got_called(a,b)
print(r)
```

Output:

Program: 6

```
def got_called(b,a):
    print("a = ", a)
    print("b = ", b)
    return (a+b)

r = got_called(a = 5,b = 6)
print(r)
```

Output:

Program: 7

```
def got_called(b,a,c=3):
    print("a = ", a)
    print("b = ", b)
    print("c = ", c)
    return (a+b+c)

r = got_called(5,6,7)
print(r)
r = got_called(5,6)
print(r)
```

Output:

Program: 8

```
def max_min(l):
    return (max(l), min(l))

l = [1,2,3,4,5]
max, min  = max_min(l)
print("max of ",l, "is ", max)
print("min of ",l, "is ", min)
```

Output:

Program: 9

```
def ftwo( ):
    n = 10
    print("n = ", n )

def fone( ):
    n = 20
    print("n = ", n )
    ftwo()
    print("n = ", n)

fone( )
```

Output:

Program: 10

```
def ftwo( ):
    print("n = ", n )

def fone( ):
    n = 20
    print("n = ", n )
    ftwo( )
    print("n = ", n)

fone( )
```

Output:

Program: 11

```
def fun_two( ):
    print("n = ", n )

def fun_one( ):
    n = 20
    print("n = ", n )
    fun_two( )
    print("n = ", n)

n = 9
fun_one( )
print(n)
```

Output:

Program: 12

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

n = int(input("Enter a number to find factorial of it\n"))
f = factorial(n)
print("The factorial of " , n ,  " is ",  f )
```

Output:

Program: 13

```
def fun(*args):
    for i in args:
        print(i, end=' ')
    print( )

fun(1)
fun(1,2)
fun(1,2,3)
fun(1,2,3,4)
```

Output:

Program: 14

```
f = lambda x: x * x
x = f(5)
print(x)
```

Output:

Program: 15

```
print((lambda x: x * x)(5))
```

Output:

Program: 16

```
f = lambda x, y: x if x > y else y
print(f(5,6))

print((lambda x, y: x if x > y else y)(5,6))
```

Output:

Program: 17

```
l = [1,2,3,4,5,6,7,8,9,0]
f = lambda x: x % 2 == 0
res = filter(f,l)
updated_list = list(res)
print(updated_list)
```

Output:

```
[2, 4, 6, 8, 0]
```

Program: 18

```
l = [1,2,3,4,5,6]
f = lambda x: x * x
res = map(f,l)
updated_list = list(res)
print(updated_list)
```

Output:

```
[1, 4, 9, 16, 25, 36]
```

Program: 19

```
lone = [1,2,3,4,5,6,7,8,9,0]
ltwo = [1,2,3,4,5,6,7,8,9,0]
f = lambda x, y: x * y
res = map(f,lone,ltwo)
updated_list = list(res)
print(updated_list)
```

Output:

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 0]
```

Program: 20

```
from functools import *

l = [1,2,3,4,5,6,7,8,9]
f = lambda x, y: x * y
res = reduce(f,l,)
#res = recuce(f,l,0)
print(res)
```

Output:

```
362880
```

Program: 21

```
def decorator( ready_for_decoration ):
    def helping_to_decorate( ):
        x = ready_for_decoration( )
        return x + 1
    return helping_to_decorate

def getting_decorated( ):
    return 10

 decorated = decorator(getting_decorated)
print(decorated( ))
```

Output:

```
11
```

Program: 22

```
def decorator( ready_for_decoration ):
    def helping_to_decorate( ):
        x = ready_for_decoration( )
        return x + 1
    return helping_to_decorate

@decorator
def getting_decorated( ):
    return 10

print(getting_decorated( ))
```

Output:

```
11
```

Program: 23

```
def mydecorator1( f ):
    def inner( ):
        x = f( )
        return x + 1
    return inner

def mydecorator2( f ):
    def inner( ):
        x = f( )
        return x * 2

    return inner

def myfunc():
    return 10


updated_func1 = mydecorator1(myfunc)
updated_func2 = mydecorator2(myfunc)

print(updated_func1())
print(updated_func2())
```

Output:

```
11
20
```

Program: 24

```
def mydecorator1( f ):
    def inner():
        x = f()
        return x + 1
    return inner

def mydecorator2( f ):
    def inner( ):
        x = f( )
        return x * 2

    return inner

@mydecorator2
@mydecorator1
def myfunc( ):
    return 10

print(myfunc())
```

Output:

```
22
```

Program: 25

```
def my_generator(x):
    for i in range(x):
        r = 5 + i
        yield r

g = my_generator(5)

for i in g:
    print(i)
```

Output:

```
5
6
7
8
9
```

Guess The Output:

1)

```
def avg(n1, n2, n3):
        return (n1 + n2 + n3)/3.0

print(avg(10,25,40))
print(avg(10,25,40) + 10)
print(avg(avg(2,4,6),8,12))
```

2)

```
def avg(n1, n2, n3):
        return (n1 + n2 + n3)/3.0
```

```
if avg(10, 25, -40) < 0:
        print "Invalid Average”
```

3)

```
def maxmin(l):
        return (max(l), min(l))

l = [10,20,30]
max_min = maxmin(l)
print(max_min[0])
print(max_min[1])
```

4)

```
def change( l ):
        l[0] = 5

def call( ):
        l = [1,2,3]
        change(l)
        print(l)

call( )
```

5)

```
def change( ):
        ll[0] = 5

def call( ):
        ll = [1,2,3]
        change( )
        print(ll)

ll = [1,2,3]
call( )
print(ll)
```

6)

```
def call(z, a, b, c ):
print(a,b,c,z)
call(c = 6, b = 3, a = 1, 5 )
```

7)

```
def call( a = 10, b, c ):
print(a,b,c)

call( 5,6 )
```

8)

```
print( "Hello" )

def hel():
    print("Bello")

print("Kello")
```

9)

```
print( "Hello" )

def hel( ):
    print("Bello")

print("Kello")
hel()
```

10)

```
def hel( n ):
    print(id(n))
    n = n + 1
    print(id(n))

n = 5
print(id(n))
hel(n)
print(id(n))
```

11)

```
def hel( n ):
    print(id(n))
    n[0]=5
    print(id(n))

n = [1,2,3]
print(id(n))
hel(n)
print(id(n))
```

12)

```
def hel( ):
    n = 6
    print(id(n))
    n = n + 1
    print(id(n))
n = 5
print(id(n))
```

```
hel()
print(id(n))
```

13)

```
def ret_example():
    return 5, 6, 7

r = ret_example()
print(r)
```

14)

```
def fun( ):

    def infun():
        print("Hello")

    infun()
    return 5

r = fun( )
print(r)
```

15)

```
def fun():

    def infun():
        print("Hello")

    infun()
    return infun

f  = fun()
f()
```

16)

```
def fun():
    lst = [1,2,3]
    print(id(lst))

lst = [1,2,3]
fun()
print(id(lst))
```

17)

```
def fun():
    a = 2
    print(a)

a = 3
print(a)
fun()
print(a)
```

18)

```
def fun():
    global a
    a = 2
    print(a)

a = 3
print(a)
fun()
print(a)
```

19)

```
def fun():
    x = globals()['a']
    a = 2
    print(a, x)

a = 3
print(a)
fun()
print(a)
```

20)

```
def fun():
    x = globals()['a']
    y = globals()['b']
    a = 2
    print(a, x, y)

a = 3
b = 5
print(a)
fun()
print(a)
```

21)

```
lone = [1,2,3,4,5,6,7,8,9]
f = lambda x: 0
res = filter(f,lone)
```

```
print(list(res))
```

22)

```
lone = [1,2,3,4,5,6,7,8,9]
f = lambda x: 0
res = map(f,lone)
print(list(res))
```

23)

```
from functools import *

lone = [1,2,3,4,5,6,7,8,9]
f = lambda x,y: x + 0
res = reduce(f,lone)
print(res)
```

Programming Assignments:

1. WAP to convert temperature from (Fahrenheit to Celcius) or (Celcius to Fahrenheit) based on user input for a specific range 'start' and 'end'
   [**Formula**: 'c = (f - 32) * 5/9' and 'f = (5/9 * c) + 32'
2. Write all previous programs using functions.
3. WAP with function names 'checkZero' that is given three integers, and returns 'True' if any of the integers is 0, otherwise it returns 'False'.
4. WAP with function named 'checkLowToHigh' that is passed three integers, and returns 'True' if the three integers are in order from smallest to largest, otherwise it returns 'False'.
5. WAP with function named 'isDivisible' that is given a positive integer, n, and a second positive integer, m <= n, and returns how many numbers between 1 and n are evenly divisible by m.
6. WAP with function named 'print_name' that display "I love you <name>" for any given name passed to the function.