MCQ and Answer Key on Python Programming (Set 3)

1. Find Output:

```
myd={"Kolkata":700001,"Mumbai":200001,"Delhi":100001}
print("Kolkata" in myd, "mumbai" in myd, "Delhi " in myd)
```

a) True False True
b) True False False
c) True True True
d) None

2. Find Output

```
q = set("I am a very good boy")
print(len(q))
```

a) 20
b) 12
c) 14
d) 18

3. myfile.txt contains:

One
Two
Three

Find output:

```
fp = open("MYFILE.txt","r")
print(fp.readline())
```

A) ['One\n', 'Two\n', 'Three']
b) One
c) Error : No such file or directory: 'MYFILE.txt'
d) None

4. Find Output:

```
t1 = ('1','2','3','3','2','1')
t1[4] = 5
print(t1[2]+t1[4])
```

a) 35
b) 32
c) Error
d) None

5. Find Output:

```
def func(a,b,c,d=None):
    if a == 1:
        a,b,c = 1,2,3
        print(a,b,c,d)
    elif b == 2:
        a,b,c=4,5,6
        print(a,b,c,d)
    elif c == 3:
        a,b,c = 7,8,9
        print(a,b,c,d)
    else:
        print(a,b,c,d)

func('1','5','9')
```

a) 1 2 3 None

b) 1 5 9 None

c) None

d) Error

6. Find Output:

```
nested_list=[123,['1','2','3'],[[1,2,3],[4,5,6]]]
print(len(nested_list))
```

a) 10

b) 4

c) 3

d) None

7. Find Output:

```
def func(Loc,Gender,Address="Delhi"):
    print("Location: ",Loc,end=" ")
    print("Gender: ",Gender,end=" ")
    print("Address: ",Address,end=" ")
    return;

func('M','Mahipalpur')
```

a) Location: Mahipalpur Gender: M Address: Delhi

b) Location: M Gender: Mahipalpur Address: Delhi

c) Location: Mahipalpur Gender: Delhi Address: M

d) None

8) Find Output:

```
txt = "I like bananas"
x = txt.replace("apples", "bananas")
print(x)
```

a) I like bananas

b) I like apples

c) Error

d) None

9) Find Output:

```
def func(x,y):
    if x ==0:
        return y
    else:
        return func(x-1,y+1)

print(func(4,1))
```

a) 4
b) 5
c) Error
d) None

10) Find Output:

```
def func(x,y):
    if x ==0:
        return y
    else:
        #print(x)
        return x*func(x-1,y+1)

func(4,1)
```

a) 20
b) 24
c) 120
d)None

11) Find output:

```
def recursion(k):
  if(k > 0):
    result = k + recursion(k - 1)
    print(result, end =" ")
  else:
    result = 0
  return result

recursion(5)
```

a) 1 3 6 10
10
b) 1 3 6 10 15
15
c) 1 2 3 4 5
15

d) None

12) Find output

```
def my_function(*args):
  print("The 2nd element is :" + args[1])
  print("The No of elements :" + str(len(args)))
my_function("Abc", '123, 123.45')
```

a) The 2nd element is :123, 123.45
The No of elements :2
b) The 2nd element is :123
The No of elements :3
c) The 2nd element is :123.45
The No of elements :3
d) None

13) Find output:

```
def my_function(arg3, arg2, arg1):
  print("The value of 3rd argument is " + str(arg3))

my_function(arg1 = "val1", arg2 = 2, arg3 = 3.0)
```

a) The value of 3rd argument is val1
b) The value of 3rd argument is 3.0
c) Error
d) None

14) Find Output:

```
def my_function(**args):
  print("His first name is " + args["fname"].upper() + " and last name is " +
args["lname"].upper())

my_function(fname = "Tushar", lname = "Singh")
```

a) His first name is Tushar and last name is Singh
b) His first name is TUSHAR and last name is SINGH
c) Error
d) None

15) Find Output:

```
def myfunction():
    pass
```

a) Error
b) No output
c) pass
d) None

Answer Key

1 – b
2 – b [Hints : set doesn't allow duplicate elements]
3 – b [Hints : file name is not case sensitive in file handing in python coding]
4 – c [Hints: Typeerror 'tuple' object does not support item assignment]
5 – b [Hints : parameter passed through the function for a,b,c are strings and default value of d = None]
6 – c
7- b
8 – a
9 – b [hints : when value of x = 4 then y = 1, similarly x = 3 then y = 2, x = 2, y = 3, x = 1, y =4 and at last when x = 0, y = 5]

10 – c

11 – b [Hints : for k = 5: result=5 + recursion(4) ; for k = 4: result=4+ recursion(3) ; for k = 3: result=3+ recursion(2) ; for k = 2: result=2+ recursion(1) ; for k = 1: result=1+ recursion(0) and as recursion(0) = 0 ; for k = 1, result = 1, for k = 2, result = 3, for k = 3, result = 6, for k = 4, result = 10, for k = 5, result = 15]

12- a

[Hints : **Arbitrary Arguments, *args**
If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition. This way the function will receive a tuple of arguments, and can access the items accordingly]

13 – b
14 - b [Hints : **Arbitrary Keyword Arguments**, ****kwargs**
If you do not know how many keyword arguments that will be passed into your function, add two asterisk: (**) before the parameter name in the function definition.This way the function will receive a *dictionary* of arguments, and can access the items accordingly]

15- b [Hints: function definitions cannot be empty, but if you for some reason have a function definition with no content, put in the pass statement to avoid getting an error.]