1. What is the result of the code, and why?

>>> def func(a, b=6, c=8):

print(a, b, c)

>>> func(1, 2)

Ans:

Result will be 1,2,8

Value of a will be as per the passing argument i.e 1

Default value of b will be changed as we are explicitly passing the value 2

Default value of c will be printed as we have not passed the third argument.

2. What is the result of this code, and why?

>>> def func(a, b, c=5):

print(a, b, c)

>>> func(1, c=3, b=2)

Ans:

Result 1 2 3

Here we are explicitly passing the c variable with value 3 and b value 2, therefore ordering does not matter.

3. How about this code: what is its result, and why?

>>> def func(a, \*pargs):

print(a, pargs)

>>> func(1, 2, 3)

Ans:

Result- 1 (2, 3)

It returns the value of a variable normally and then returns a tuple containing the b and c values

4. What does this code print, and why?

>>> def func(a, \*\*kargs):

print(a, kargs)

>>> func(a=1, c=3, b=2)

Ans:

Result- 1 {'c': 3, 'b': 2}

It returns the value of a variable normally

After that \*\*kargs returns dictionary containing the key as name of the variable and value as argument value which was passed.

5. What gets printed by this, and explain?

>>> def func(a, b, c=8, d=5): print(a, b, c, d)

>>> func(1, \*(5, 6))

Ans:

Result- 1 5 6 5

It returns the value of a variable normally

b and c value are passed as per the tuple and d value as default value as per definition

6. what is the result of this, and explain?

>>> def func(a, b, c): a = 2; b[0] = 'x'; c['a'] = 'y'

>>> l=1; m=[1]; n={'a':0}

>>> func(l, m, n)

>>> l, m, n

Ans:

1 ['x'] {'a': 'y'}

As l is of type int it is immutable therefore the function can not change the value of l variable and the variable value of 1 persists.

Fucn function modifies the content of list which is defined by variable m and generates a value of [‘x’]

Fucn function modifies the value of dictionary which is defined by variable n and updates the key (a) value to ‘y’