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

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

print(a, b, c)

>>> func(1, 2)

-🡪 this will print 1, 2, 8 as we have given the values of a and b as 1 and 2 while calling function and c has a default value of 8 and it is ok if we don’t pass the value for c.

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)

* It will print 1, 2, 3 as we have passed values 2 and 3 for b and c respectively while calling the function.

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

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

print(a, pargs)

>>> func(1, 2, 3)

🡪 this will print 1, (2,3) as 2, 3 are saved in pargs as separate elements.

4. What does this code print, and why?

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

print(a, kargs)

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

* It will return error as we have not declared parameters b and c in arguments. We can only pass one value and dictionary in function.

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))

🡪 it will print 1, 5, 6, 5

As \* will assign the values elementwise and d have default value 5.

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

* 1, [‘x’], {‘a’ : ‘y’}
* Here at first value of ‘a’ will be set to 1 which is the value of l, after entering the function it will change to 2, but there will no effect on value of 1.
* As b = m = [1] after function it will [‘x’] as both list shares same memory.
* Same is with the case of n = {‘a’ : ‘y’}