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. Output= 1 2 8 because func() is passing two values which will assign to first two variables i.e. a & b.**

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. Output= 1 2 3 because in the function we are passing the values of b & c specifically.**

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

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

print(a, pargs)

>>> func(1, 2, 3)

**Ans. Output= 1 (2,3) because \*pargs takes n no. of arguments and store them in the form of tuples.**

4. What does this code print, and why?

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

print(a, kargs)

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

**Ans. Output = 1 {'c': 3, 'b': 2} because \*\*kargs takes n no. of arguments and store them in the form of dictionary.**

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. 1 5 6 5 in this 1 is assigned to a and \*(5,6) assigned to variables after a i.e. b & c,**

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. Output** **= (1, ['x'], {'a': 'y'}** **by passing the values of m,n in func() it’s replaces their original values as m is a list and n is a dictionary as changes will also reflect back in list and dictionary.**