**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 - 1 2 8 is the result beacausw a is assigned the value 1 b is assigned the value 2 and c retains its default value of 8.

**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 – 1 2 3 is the result bacause a is assigned the value 1 b is assigned the value 2 and c is assigned the value 3 as specifoed in the function call.

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

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

**print(a, pargs)**

**>>> func(1, 2, 3)**

Ans – 1 (2,3) is the result this is because a is asssigned the value 1 and pargs is tuple having the remaining arguments (2,3) .

**4. What does this code print, and why?**

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

**print(a, kargs)**

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

Ans – 1{‘c’: 3 , ‘b’ :2}

this is because a is assigned the value 1 and kargs is a dictionary having the remaining keywords arguments c = 3 and b= 2.

**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 is the result because a ais assigne dthe value 1 , b is assigned the value ‘5’ c is assigned the value 6.

**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’}

This is because the value of ‘1’ remains unchanged the list ‘m’ is modified to [‘x’] and the dictionary ‘n’ is modified to [‘a’: ‘y’}