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

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

**print(a, b, c)**

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

1,2,8 because a is positional argument so it take first value from during function call. We also provided value of keyword argument b but didn’t give value of c so it’s default value 8 was executed.

**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)**

1,2,3 when we pass value to positional argument by addressing as a=2, c=6 then its position doesn’t matter. That's why here we gave keyword argument before positional argument but still we got result.

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

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

**print(a, pargs)**

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

1,2,3 first value should be taken by positional argument then by \*args

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

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

**print(a, kargs)**

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

1 {‘c’ : 3, ‘b’ : 2} Because a is a positional argument and rest are from \*\*kwargs or keyword argument.

**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))**

1,5,6,5 Here a takes 1 as positional argument. 5 and 6 are given as a tuple but \* is used iterate over any iterable item so 5 goes to b and 6 goes to c. default value of d was printed as value was not given.

**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, [1], {'a': 0}) values of a, b and c are updated to l, m and n whose values were given before function call.