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

Since, 1,2 values are passing to a and b, the existing value for b=6 will get overwritten because the priority will be given to function calling parameters. Also, c is not passing the default value of c will gets printed.

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

Here, no matter we are passing the keyword arguments, function parameters will automatically matches the key parameters. So, output gets printed.

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)

Here, first parameter value ‘1’ will be passed to function. Next two parameters 2,3 values will gets recognized as a tuple then output gets printed.

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}

Here, keyword parameter ‘a’ will gets recognized normally, but next two keyword parameters will recognize as a dictionary since \*\* is used.

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 unpacking operator and are operators that unpack the values from iterable objects in Python. The single asterisk operator \* can be used on any iterable that Python provides, while the double asterisk operator \*\* can only be used on dictionaries. In the example the value \*(5,6) will be unpacked and will be assigned to b and c and passed as arguments, d =5 will taken by defaults are keyword arguments.

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'})

Here in the code, the list and dict are passed as argument, and those are mutable. Here the list l and parameter b point to the same list in the memory location where as dict n and c point to the same memory location. Any updates to this list will update in the memory location. l = 1 , integer values, immutable, m is list, mutable, n is dict, mutablev and output will be = 1,['x'],{'a':'y'}