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: result is 1 2 8. in the function call we are passing two values. The function automatically assigns given two values to the first two parameters. for the 3rd parameter , in the function defenition itself a value has given, so that value will be printed for the third value '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)

Ans: when we pass some values along with the function call, and variable in which value passed and the variable passing in the funtion defenition are same, then while passing the variable order is not need to follow, using the variable name function will automaticallu identify the value.

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

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

print(a, pargs)

>>> func(1, 2, 3)

Ans: result is 1,(2,3) here one will be assigned to the first variable a, and other valuaes assigned to variable length argument \*pargs. here a is not a keyword argument its a positional argument.

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 1 will be passesd as a positional argument and will be assigned to the variable a, as a is defined in the function parameter defenition. the remaining arguments we passed are keyword arguments, \*\*kwargs collects keword arguments. Internally it acts like a dictionary which maps values to the keyword.

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 , Here no variable lengrth argument is defined in the function. the value passed in the function call just considered as positional arguments. Hence the output.

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'}).