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

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

print(a, b, c)

>>> func(1, 2)

A.Output is 1 2 8

This is because the function “func “ is called with the arguments 1 and 2 for a and b respectively. Since no value is provided for c, it will use the default value of 8. Inside the function, the values of a,b, and c are printed using the print function.

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)

A.Output is (1 2 3)

This is because the function func is called with the argument 1 for a, 2 for b and c=3 to override the default value of c. Inside the function, the values of a,b, and c.

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

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

print(a, pargs)

>>> func(1, 2, 3)

A.Output is 1 (2,3)

This is because the function func is called with the arguments 1,2 and 3. The value 1 is passed as the required argument a, and 2 and 3 are packed into the variable-length argument list **pargs.** Inside the function, the values of a and **pargs** are printed using the print function. So the output will be 1 (2,3)

4. What does this code print, and why?

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

print(a, kargs)

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

A.Output is 1 {‘c’:3, ’b’: 2}

This is because the function func is called with the keyword arguments a=1,c=3,and b=2. The values of a,c and b are packed into the variable-length keyword argument dictionary **kargs.** Inside the function, the values of a and **kargs** are printed using the print function. So the ouput will be 1 {‘c’:3, ’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))

A.Output is 1 5 6 5

This is because the function func is called with the arguments 1 for a and \*(5, 6) for b, which expands to (5, 6) and is unpacked into b. So the values 5 and 6 are assigned to b and c respectively and the default value of 5 is assigned to d. Inside the function, the values of a,b,c and d are printed using the print function. So the output will be 1 5 6 5.

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

A.Output is 1 {‘x’} {‘a’: ‘y’}

This is because the function func is called with 1,m and n as arguments. Inside the function, the value of a is set to 2,which does not affect the value of 1 outside the function. The first element of b is set to ‘x’, which changes the value of the first element of m outside the function. The value associated with the key ‘a’ in c is set to ‘y’, which changes the value associated with the key ‘a’ in n outside the function. When function call returns, the values of 1,m and n are printed using the print function. When the function call returns the values of 1, m and n are printed using the print function. So the output will be 1 {‘x’} {‘a’: ‘y’}.