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

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

print(a, b, c)

>>> func(1, 2)

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)

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

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

print(a, pargs)

>>> func(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)

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))

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

Answer:

1. The result of the code is "1 2 8". The function func has three parameters, a, b, and c, where b and c have default values of 6 and 8, respectively. When func is called with arguments 1 and 2, the value 1 is assigned to parameter a and the value 2 is assigned to parameter b. Since no value is given for parameter c, its default value of 8 is used.
2. The result of the code is "1 2 3". The function func has three parameters, a, b, and c, where c has a default value of 5. When func is called with arguments 1, c=3, and b=2, the value 1 is assigned to parameter a, the value 2 is assigned to parameter b, and the value 3 is assigned to parameter c, overwriting its default value.
3. The result of the code is "1 (2, 3)". The function func has two parameters, a and \*pargs, where \*pargs is a tuple of any additional positional arguments. When func is called with arguments 1, 2, and 3, the value 1 is assigned to parameter a, and the tuple (2, 3) is assigned to \*pargs.
4. The code prints "1 {'c': 3, 'b': 2}". The function func has one parameter, a, and \*\*kargs is a dictionary of any additional keyword arguments. When func is called with keyword arguments a=1, c=3, and b=2, the value 1 is assigned to parameter a, and the dictionary {'c': 3, 'b': 2} is assigned to \*\*kargs.
5. The code prints "1 5 6 5". The function func has four parameters, a, b, c, and d. When func is called with arguments 1 and \*(5, 6), the value 1 is assigned to parameter a, the value 5 is assigned to parameter b, the value 6 is assigned to parameter c, and the default value of 5 is assigned to parameter d.
6. The result of the code is "1 [x] {'a': 'y'}". The function func has three parameters, a, b, and c. When func is called with arguments l, m, and n, the value 2 is assigned to local variable a, m[0] is assigned the value 'x', and n['a'] is assigned the value 'y'. Since l, m, and n were passed by reference, their values have been modified by the function. Therefore, the final values of l, m, and n are 1, ['x'], and {'a': 'y'}.