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: if we consider that the indentions are proper the function call will generate the result as (1,2,8)

With the call ,

a=1,

b=2(the default value of 6 has been updated with the calling argument value),

c= 8 (default value)

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:

function call will generate the result as (1,2,3)

a=1,

b=2(key value parameter),

c= 3(default value of 5 is overwritten by 3)

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)

With the call

a=1,

2 and 3 are passed as a part of the \*pargs, which returns the result as a tuple

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}

A=1,

kwargs creates a dictionary with the key and the value pair values (b,c)

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:

a = 1

\*(5, 6) are argument for next 2 parameters ie b,c

b =5,

c =6

D=5(default value)

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:

|  |  |  |
| --- | --- | --- |
| func(l, m, n) | func(1, [1], {‘a’:0}) | a=1, b=[1], c={‘a’:0} |
|  | a = 2; b[0] = 'x'; c['a'] = 'y' | a=2, b=[‘x’], c={‘a’:’y’} |

as l,m,n are copies of a,b,c the updates done in a,b,c will be updated for the l, m, n

l,m,n🡪 2, [‘x’], {‘a’:’y’}