Assignment-23

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

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

print(a, b, c)

>>> func(1, 2)

Ans1- ouput= 1 2 8

The values of a, b, and c in the function call func(1, 2) are 1, 2, and 8, which are printed by the print statement within the 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)

Ans2- output= 1 2 3

The values of a, b, and c in the function call func(1, c=3, b=2) are 1, 2, and 3, which are printed by the print statement within the function.

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

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

print(a, pargs)

>>> func(1, 2, 3)

Ans3- output=1 (2, 3)

The func function prints the value of a and the pargs tuple, which contains the additional positional arguments passed to the function.

4. What does this code print, and why?

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

print(a, kargs)

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

Ans4- output= 1 {'c': 3, 'b': 2}

The func function prints the value of a and the kargs dictionary, which contains the additional keyword arguments passed to the function.

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))

Ans5- output= 1 5 8 5

The func function prints the values of a, b, c, and d, where a and b come from the function call, and c and d use their default values.

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

Ans6- output= 1 ['x'] {'a': 'y'}

l remains 1 because the local variable a inside the func function is separate from the global variable l.

m becomes ['x'] because the list m was modified inside the func function.

n becomes {'a': 'y'} because the dictionary n was modified inside the func function.