**#QUESTION 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:

The result of the code will be:

The function func is defined with three parameters: a, b, and c.

The parameter b has a default value of 6, and the parameter c has a default value of 8.

When the function is called with func(1, 2), the value of a is assigned as 1, b is assigned as 2, and c retains its default value of 8.

The print statement inside the function will output the values of a, b, and c, which are 1, 2, and 8 respectively.

**#QUESTION 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:

The result of the code will be:

1 2 3

Explanation:

The function func is defined with three parameters: a, b, and c.

When the function is called with func(1, c=3, b=2), the value of a is assigned as 1, b is assigned as 2, and c is explicitly assigned as 3 using the keyword argument c=3.

The print statement inside the function will output the values of a, b, and c, which are 1, 2, and 3 respectively.

**#QUESTION 3:**

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

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

print(a, pargs)

>>> func(1, 2, 3)

Ans:

The result of the code will be:

1 (2, 3)

Explanation:

The function func is defined with a parameter a and \*pargs which is a variable-length argument.

When the function is called with func(1, 2, 3), the value 1 is assigned to a, and the values 2 and 3 are collected into pargs as a tuple.

The print statement inside the function will output the value of a and pargs, which is 1 and (2, 3) respectively.

**#QUESTION 4:**

What does this code print, and why?

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

print(a, kargs)

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

Ans:

The code will print the following output:

1 {'c': 3, 'b': 2}

Explanation:

The function func is defined with a parameter a and \*\*kargs which is a variable-length keyword argument.

When the function is called with func(a=1, c=3, b=2), the value 1 is assigned to a, and the keyword arguments c=3 and b=2 are collected into kargs as a dictionary.

The print statement inside the function will output the value of a and kargs, which is 1 and {'c': 3, 'b': 2} respectively.

**#QUESTION 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:

The code will print the following output:

1 5 6 5

Explanation:

The function func is defined with parameters a, b, c, and d, where c and d have default values of 8 and 5 respectively.

When the function is called with func(1, \*(5, 6)), the value 1 is assigned to a, and the tuple (5, 6) is unpacked and assigned to b.

Since there are no keyword arguments provided, the default values for c and d are used, resulting in c being 8 and d being 5.

The print statement inside the function will output the values of a, b, c, and d, which are 1, 5, 6, and 5 respectively.

**#QUESTION 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:

The code will print the following output:

1 ['x'] {'a': 'y'}

Explanation:

The function func takes three arguments: a, b, and c.

Inside the function, the variable a is assigned the value 2, but this assignment does not affect the original variable l because integers are immutable.

The first element of the list m is modified to 'x' using the assignment b[0] = 'x'. Since lists are mutable, the change is reflected in the original variable m as well.

The value associated with the key 'a' in the dictionary n is modified to 'y' using the assignment c['a'] = 'y'. Like lists, dictionaries are mutable, so the change is reflected in the original variable n too.

After calling the func function with the variables l, m, and n as arguments, the values of l, m, and n are printed.

The value of l remains unchanged since it is immutable.

The value of m is modified because it is a mutable list and was passed as a reference to the function.

The value of n is also modified because it is a mutable dictionary and was passed as a reference to the function.