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 is 1 2 8. In func(1, 2), the parameter a gets the value 1, the parameter b gets the value 2 and c gets the default value of 8.

1. 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 is 1 2 3. In func(1, c=3, b=2), the parameter a gets the value 1, the parameter b gets the value 2 and c gets the value of 3.

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

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

print(a, pargs)

>>> func(1, 2, 3)

**Ans:** The result is 1 (2, 3). This is due to pargs passes variable number of non-keyworded arguments list and on which operation of the list can be performed.

1. What does this code print, and why?

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

print(a, kargs)

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

**Ans:** The result is 1 {'c': 3, 'b': 2}. This is due to kargs passes variable number of keyword arguments dictionary to function on which operation of a dictionary can be performed.

1. 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 result is 1 5 6 5. In func(1, \*(5,6)), the parameter a gets the value 1, the parameter b gets the value 5, the parameter c gets the value 6 and d gets the value of 5.

1. 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:** Process finished with exit code 0 is the result because we not call print function hence nothing is displayed as a result.