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

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

print(a, b, c)

>>> func(1, 2)

Result: 1,2,8

As the b and c is already assigned so when b or c is not defined specifically the intepreter will take the 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)

Result: 1,2,3

As the values of a, b, c are defined specifically therefore the default values will be ignored

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

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

print(a, pargs)

>>> func(1, 2, 3)

Result: 1, (2,3)

As after first digit all the values are considered as arguments so the value 2 and 3 are separate values

4. What does this code print, and why?

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

print(a, kargs)

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

Result: 1, {‘b’:2,’c’:3}

Since the elements after **a** are defined as dictionary i.e KWARGS

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))

Result: 1 5 6 5

As the 5 and 6 are being explained as argument in the function func. D is not defined so it will take the 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

Result: (1, [‘x’],{‘a’:’y’})

As the value of l is locally defined but since the values for the m and n are locally defined and also inside the function there values are in the form of list and dictionary. Hence there values are replaced with the function values while the value of is taken as same.