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 => 1 2 8 because we are passing value of a and b from function call and c value already defined in arguments

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 =>1 2 3 because because we are assigning values in function call and function arguments

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) because it considering a as 1 and 23 as args value separated by comma means 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} because \*\*kargs returns dictionary structure always

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 => 1 5 6 5 because we are assigning it with \* so the values considered for b and c and d value already assigned for argument

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 => result will be blank because we are not returning anything from function call or not printing