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

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

print(a, b, c)

>>> func(1, 2)
```

**Answer: Output is 1 2 8**
Because a and b are overwritten in the function call a=1,b=2, and c is already defined in the function scope as c=8.

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)
```

**Answer**: **Output is 1 2 3**

function is called with parameters c=3, b=3 and 1 in place of a, hence those values are printed

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

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

print(a, pargs)

>>> func(1, 2, 3)
```

**Answer**: **Output is 1 (2, 3)**

Because 1 is passed for 1, and 2,3 is passed as non keyword arguments for *pargs, which saves those values in a tuple

4. What does this code print, and why?

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

print(a, kargs)

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

**Answer: The output is 1 {'c': 3, 'b': 2}**

Here 1 is passed as a, the first parameter, the other value pairs are passed as keyword arguments, c=3, and b=2 is converted into key-value pairs, with c and b as keys and 3 and 2 their respective values.

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))
```

**Answer: The output is 1 5 6 5**

a takes the first parameter 1, and b and c are assigned the non keyword arguments (5 and 6) respectively. And d is assigned 6 in the scope of the function. Hence print(a,b,c,d) gives the above output.

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
```

**Answer: The output is (1, ['x'], {'a': 'y'})**

Here 1 is assigned to variable l, which is then passed to the functional variable a, which has a local value of 2.

Next, list m=[1] is assigned to variable b, wherein b is locally assigned the value b[0] =’x’. As lists are unambiguous and are always global in nature, we infer, **m=[a] = b = [‘x’], hence m = [‘x’].** Without the global nature of lists, the statement b[0] =’x’ will be false as the list has not been declared inside or outside the scope of the function.

Next the dictionary n={'a':0} is passed to the variable c, which is locally assigned c[‘a’] = ‘y’. As dictionary, like lists, are also mutable unambiguous objects which are always defined before calling, because otherwise assign a new value would throw an error if the dictionary wasnt defined globally

Hence we infer, **n={'a':0}** → **c[‘a’] = ‘y’** → **n={‘a’: ‘y’}**