Assignment-23

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. Result: 1 2 8

Explanation: The function has default values for parameters b (6) and c (8). When called with func(1, 2), 1 is assigned to a, 2 overrides the default for b, and c keeps its default value of 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)
```

Ans. Result: 1 2 3

Explanation: This demonstrates keyword arguments. The function is called with one positional argument (1 for a) and two keyword arguments (c=3 and b=2). The order of keyword arguments doesn't matter.

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

```
>>> def func(a, *pargs):
print(a, pargs)
>>> func(1, 2, 3)
```

Ans. Result: 1 (2, 3)

Explanation: This function uses *pargs to collect any additional positional arguments into a tuple. When called with func(1, 2, 3), 1 is assigned to a, and (2, 3) becomes the tuple pargs.

4. What does this code print, and why?

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

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

Ans. Result: 1 {'c': 3, 'b': 2}

Explanation: This function uses **kargs to collect keyword arguments into a dictionary. When called with keyword arguments, 'a' is assigned to the parameter a, and the rest form a dictionary {'c': 3, 'b': 2}.

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. Result: 1 5 6 5

Explanation: The * operator unpacks the tuple (5, 6) into individual arguments. Therefore, this call is equivalent to func(1, 5, 6), with d keeping its default value of 5.

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

Explanation: This demonstrates mutability in Python:

- l is an integer (immutable), so assigning a=2 in the function doesn't affect the original l
- m is a list (mutable), so modifying b[0] in the function changes the original list
- n is a dictionary (mutable), so modifying c['a'] in the function changes the original dictionary