**ANSWERS**

**Q1. 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 output of the above mentioned code is **“1 2 8”**. In this code, a function func() is defined with three parameters: a, b, and c, with default values 6 and 8 for b and c, respectively. When the function is called with func(1, 2), the value 1 is assigned to a, 2 is assigned to b, and the default value 8 is assigned to c. Thus, the output of print(a, b, c) is 1 2 8.

**Q2. 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 output of the above mentioned code is “1 2 3”. In this code, a function func() is defined with three parameters: a, b, and c, with a default value of 5 for c. When the function is called with func(1, c=3, b=2), the value 1 is assigned to a, 2 is assigned to b, and 3 is assigned to c. Thus, the output of print(a, b, c) is 1 2 3.

**Q3. How about this code: what is its result, and why?**

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

**print(a, pargs)**

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

**Ans.** The output of the above mentioned code is **”1 (2, 3)”**. In this code, a function func() is defined with a parameter a and a variable-length argument \*pargs, which collects any additional positional arguments into a tuple. When the function is called with func(1, 2, 3), the value 1 is assigned to a, and the additional positional arguments 2 and 3 are collected into a tuple and assigned to pargs. Thus, the output of print(a, pargs) is 1 (2, 3).

**Q4. What does this code print, and why?**

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

**print(a, kargs)**

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

**Ans.** The output of the above mentioned code is **“1 {‘c’ : 3, ‘b’ : 2}”**. In this code, a function func() is defined with a parameter a and a double-asterisk argument \*\*kargs, which collects any additional keyword arguments into a dictionary. When the function is called with func(a=1, c=3, b=2), the value 1 is assigned to a, and the additional keyword arguments c=3 and b=2 are collected into a dictionary and assigned to kargs. Thus, the output of print(a, kargs) is 1 {'c': 3, 'b': 2}.

**Q5. 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 output of the above mentioned code is **“1 5 6 5”**. In this code, a function func() is defined with four parameters: a, b, c, and d, with default values of 8 and 5 for c and d, respectively. When the function is called with func(1, \*(5, 6)), the value 1 is assigned to a, and the tuple (5, 6) is unpacked into b and c, while d retains its default value of 5. Thus, the output of print(a, b, c, d) is 1 5 6 5.

**Q6. 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.** The output of the above mentioned code is **“1 ['x'] {'a': 'y'}”.** In this code, a function func() is defined with three parameters: a, b, and c. Inside the function, a is reassigned to 2, b[0] is modified to 'x', and c['a'] is modified to 'y'. Then, the variables l, m, and n are assigned 1, [1], and {'a': 0} respectively. When the function func(l, m, n) is called, it modifies these variables according to the function's behavior. After the function call, l remains unchanged, m is modified to ['x'], and n is modified to {'a': 'y'}. This is why the print statement outputs 1 ['x'] {'a': 'y'}.