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

&gt;&gt;&gt; def func(a, b=6, c=8):

print(a, b, c)

&gt;&gt;&gt; func(1, 2)

A: 1 2 8

as per positional argument 1 and 2 will be assigned to a and b while 8 will be assigned to c

2. What is the result of this code, and why?

&gt;&gt;&gt; def func(a, b, c=5):

print(a, b, c)

&gt;&gt;&gt; func(1, c=3, b=2)

A: 1 2 3

1is assigned to a , while 2 and 3 is passed as values of arguments for b and c respectively

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

&gt;&gt;&gt; def func(a, \*pargs):

print(a, pargs)

&gt;&gt;&gt; func(1, 2, 3)

A: 1 (2, 3)

\* pargs will take no of arguments and return a tuple inside the function with the name of parameter.

4. What does this code print, and why?

&gt;&gt;&gt; def func(a, \*\*kargs):

print(a, kargs)

&gt;&gt;&gt; func(a=1, c=3, b=2)

1 {'c': 3, 'b': 2}

Arguments passed as a dictionary , with the name of the parameter excluding \*\*

5. What gets printed by this, and explain?

&gt;&gt;&gt; def func(a, b, c=8, d=5): print(a, b, c, d)

&gt;&gt;&gt; func(1, \*(5, 6))

A:

1 5 6 5

It will return as per positional arguments.

6. what is the result of this, and explain?

&gt;&gt;&gt; def func(a, b, c): a = 2; b[0] = &#39;x&#39;; c[&#39;a&#39;] = &#39;y&#39;

&gt;&gt;&gt; l=1; m=[1]; n={&#39;a&#39;:0}

&gt;&gt;&gt; func(l, m, n)

&gt;&gt;&gt; l, m, n

A:

(1, ['x'], {'a': 0, 0: 'y'})

Will return as per the data types.