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

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

print(a, b, c)

func(1, 2)

**Result:**

1 2 8

**Explanation:**

* The argument a is explicitly passed the value 1.
* The argument b is explicitly passed the value 2.
* The argument c is not passed, so it takes 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)

**Result:**

1 2 3

**Explanation:**

* The argument a is explicitly passed the value 1.
* The argument b is explicitly passed the value 2.
* The argument c is explicitly passed the value 3, which overrides the default value of 5.

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

def func(a, \*pargs):

print(a, pargs)

func(1, 2, 3)

**Result:**

1 (2, 3)

**Explanation:**

* The argument a is explicitly passed the value 1.
* The \*pargs syntax collects all extra positional arguments into a tuple. In this case, 2 and 3 are captured in the pargs tuple, so pargs becomes (2, 3).

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

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

print(a, kargs)

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

**Result:**

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

**Explanation:**

* The argument a is explicitly passed the value 1.
* The \*\*kargs syntax collects all extra keyword arguments into a dictionary. In this case, c=3 and b=2 are captured in the dictionary, so kargs becomes {'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))

**Result:**

1 5 6 5

**Explanation:**

* The argument a is explicitly passed the value 1.
* The \* operator unpacks the tuple (5, 6) and assigns b = 5 and c = 6.
* Since d is not passed, it takes 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

**Result:**

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

**Explanation:**

* The variable l is passed as 1, but it's a primitive value, so modifying a inside the function does not affect it. The value of l remains 1.
* The variable m is passed as a list. Lists are mutable, so modifying m[0] to 'x' inside the function affects the original list. After the function call, m becomes ['x'].
* The variable n is passed as a dictionary. Dictionaries are mutable, so modifying c['a'] to 'y' inside the function affects the original dictionary. After the function call, n becomes {'a': 'y'}.