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)
```

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)
```

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)
```

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)
```

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))
```

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
```

1. The code will print: 1 2 8. The function `func` takes three arguments, `a`, `b`, and `c`. The default values of `b` and `c` are 6 and 8 respectively. In the function call `func(1, 2)`, the value 1 is passed to `a`, and 2 is passed to `b`, so `b` is set to 2. Since no value is passed for `c`, it takes its default value of 8. Hence, the output is 1 2 8.
2. The code will print: 1 2 3. The function `func` takes three arguments, `a`, `b`, and `c`. In the function call `func(1, c=3, b=2)`, the value 1 is passed to `a`, 2 is passed to `b`, and 3 is passed to `c`. Since `b` and `c` are passed by keyword, their order in the function call doesn't matter. Hence, the output is 1 2 3.
3. The code will print: 1 (2, 3). The function `func` takes a variable number of arguments, with the first argument `a` being mandatory, and any additional arguments packed into a tuple called `pargs`. In the function call `func(1, 2, 3)`, 1 is passed to `a`, and 2 and 3 are packed into a tuple and passed as `pargs`. Hence, the output is 1 (2, 3).
4. The code will print: 1 {'c': 3, 'b': 2}. The function `func` takes a variable number of keyword arguments, which are packed into a dictionary called `kargs`. In the function call `func(a=1, c=3, b=2)`, 1 is passed to `a`, and the keyword arguments `c=3` and `b=2` are packed into a dictionary and passed as `kargs`. Hence, the output is 1 {'c': 3, 'b': 2}.

5. The code will print: 1 5 6 5. The function `func` takes four arguments, `a`, `b`, `c`, and `d`. In the function call `func(1, *(5, 6))`, 1 is passed to `a`, and the tuple `(5, 6)` is unpacked and passed as `b` and `c`, respectively. Since no value is passed for `d`, it takes its default value of 5. Hence, the output is 1 5 6 5.
6. The code will print: `1`, `['x']`, `{'a': 'y'}`. The function `func` takes three arguments, `a`, `b`, and `c`. Inside the function, `a` is assigned the value 2, which does not affect the value of `l` outside the function. `b` is a list, and the function modifies its first element by setting it to `'x'`. Since `m` is a reference to the same list object as `b`, the change made by the function affects `m` as well. `c` is a dictionary, and the function modifies its value for the key `'a'` by setting it to `'y'`. Since `n` is a reference to the same dictionary object as `c`, the change made by the function affects `n` as well. Hence, after the function call, `l`