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

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

print(a, b, c)

>>> func(1, 2)
```

The result of the code is: 1 2 8.
b value redefined while calling the function func()

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)
```

The result of the code is: 1 2 3.
Value of the b & c is redefined while calling the function func() & print value assigned as a, b, c

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

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

print(a, pargs)

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

The result of the code is: 1 (2, 3)
It is due to usage of the *args (multiple argument) syntax the result will be as above.

4. What does this code print, and why?

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

print(a, kargs)

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

The result of the code is: 1 {'c':3,'b':2}
It is due to usage of the **kwargs (multiple argument with key value pair, it will act as a dictionary) syntax the result will be as above.

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))
```

The result of the code is: 1 5 6 5
The function is redefined, and values printed as pr the final input.

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
```

The result of the code is: (1, [‘x’], {‘a’:’y’})

While defining the function a is assigned as a variable, b is assigned as a list and c is assigned as a dictionary.