```
1. What is the result of the code, and why?
>>> def func(a, b=6, c=8):
print(a, b, c)
>>> func(1, 2)
```

In [1]:
```
def func(a,b=6,c=8):
    print(a,b,c)
func(1,2)
```

```
1 2 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)
```

In [2]:
```
def func(a,b,c=5):
    print(a,b,c)
func(1,c=3,b=2)
```

```
1 2 3
```

```
3. How about this code: what is its result, and why?
>>> def func(a, *pargs):
print(a, pargs)
>>> func(1, 2, 3)
```

In [3]:
```
def func(a, *pargs):
    print(a,pargs)
func(1,2,3)
```

```
1 (2, 3)
```

```
4. What does this code print, and why?
>>> def func(a, **kargs):
print(a, kargs)
>>> func(a=1, c=3, b=2)
```

In [4]:
```
def func(a,**kargs):
    print(a,kargs)
func(a=1,c=3,b=2)
```

```
1 {'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))
```

In [5]:
```
def func(a,b,c=8,d=5):
    print(a,b,c,d)
func(1,*(5,6))
```

```
1 5 6 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
```

In [6]:
```
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
```

Out[6]:
```
(1, ['x'], {'a': 'y'})
```

In [ ]: