Parameter Passing

```
def f(a,b,x,y):
    print(a,b,x,y)
s = (10,20)
t = {'x' : 30, 'y' : 40}
f(*s, **t)
```

Ok. *s will unpack the tuple to corresponding positional parameters, so a gets 10 and b gets 20. Similarly, **t will unpack dictionary so x gets 30, y gets 40.

```
def f(a,b,c,d):
    print(a,b,c,d)
f(b=10, a = 20, 3, 4)
```

ERROR. In function call, we cannot have positional arguments passed after keyword arguments. Keyword arguments have to be rightmost in call.

```
def f(a,b,c,d):
    print(a,b,c,d)
f(1,2,a=10,c=20,d=40)
```

ERROR. Parameter a gets its value twice. In call, each parameter can get its value at most once.

```
def f(a,b,*c):
    print(a,b,c)
f(a=1, b=2, 3, 4)
```

ERROR. In function call, we cannot have positional arguments passed after keyword arguments. Keyword arguments have to be rightmost in call.

```
def f(a,b,*c):
    print(a,b,c)
f(a=10, b=20, c=(30,40))
```

ERROR. *-parameter cannot be passed a value through keyword.

```
def f(*c, a, b):
    print(a,b,c)
f(1,2,3,4,5)
```

ERROR. 1,2,3,4,5 will all get collected into c. However, a and b don't get any values. Particularly, whenever in def statement we have any parameters after the *-parameter, those parameters are keyword-only parameters. Thus, they have to be passed in via keyword.

```
def f(*c, a, b):
    print(a,b,c)
f(a=1, b=2, 3,4,5)
```

ERROR. In function call, we cannot have positional arguments passed after keyword arguments. Keyword arguments have to be rightmost in call.

```
def f(*c, a, b):
    print(a,b,c)
f(1, 2, 3, a=4, b=5)
```

Ok. 1,2,3 get collected into c. And a and b get values through keyword.

```
def f(*x=(10,20)):
    print(x)
f()
```

ERROR. *-parameters cannot take default values.

```
def f(**kw, x):
    print(x, kw)
f(x=10, a=20,b=30)
```

ERROR. In a def statement, nothing can appear after **-parameter. That has to be the last.

```
def f(x, **kw):
    print(x, kw)
f(x=10, a=20,b=30)
```

Ok. x gets assigned via keyword. The remaining two keyword assignments are collected into a dictionary in kw.

```
def f(x, **kw):
    print(x, kw)
f({'a':10, 'x':20})
```

Ok. There is nothing restricting the type of parameters. So parameter x can be any type. Here, f is called with one parameter, a dictionary. This whole dictionary gets assigned to x. Kw gets nothing. So it becomes an empty dictionary.

```
def f(x, **kw):
    print(x, kw)
f(30, {'a':10, 'x':20})
```

ERROR. This function takes only one positional argument x (x can be passed via keyword also if needed). But kw cannot be passed by position. Kw will get keyword parameters only. However, in this call, we are passing two arguments by position. The first one gets assigned to x, but the second one cannot get assigned to anything, since kw will not accept positional parameters, but only keyword.

```
def f(x, **kw):
    print(x, kw)
f(30, **{'a':10, 'x':20})
```

This has solved the problem of previous slide. But still, this gives ERROR. We use **-unpacking to unpack the dictionary and create keyword parameters a and x. However, x has already been assigned by position. Hence, x cannot be assigned once again. Thus, error.

```
def f(*x, y, **kw):
    print(x,y,kw)
f(c=10, d=20, a=30, b=40, y=50)
```

Ok. The *-parameter x collects all positional parameters. Here, in the call, there are no positional parameters. So x becomes empty tuple. What comes after *-parameter is keyword-only parameter. Thus, y HAS to be passed via keyword ONLY. This has happened in this call. So y gets assigned to 50. All the other remaining keyword parameters in the call get packed into **kw as a dictionary.