PCS109　系统参数

概述

系统参数，即经常说的函式的形参。在定义函式时，若此函式带有参数，就把这些定义的参数叫做形式参数，当调用这个函式时传入形参的那些变量叫做实际参数。

应用

定义参数

Python 中函式参数的传递是通过赋值来进行的。在 Python 中函式参数的定义主要有四种方式：

F(arg1,arg2,...)是最常见的定义方式，一个函式可以定义任意个参数，每个参数间用逗号分割，用这种方式定义的函式在调用的时候也必须在函式名后的小括号里提供个数相等的值（实际参数），而且顺序必须相同，也就是说在这种调用方式中，形参和实参的个数必须一致，而且必须一一对应，即第一个形参必须对应第一个实参。例如：

```
        def a(x,y):
            print x,y
```

调用该函式，a(1,2)则 x 取 1，y 取 2，形参与实参相对应，如果是 a(1)或者 a(1,2,3)则会报错。

F(arg1,arg2=value2,...,argN = valueN)这种方式就是第一种的改进版，它提供了默认值。例如：

```
    def a(x,y=3):
        print x,y
```

调用该函式，a(1,2)同样还是 x 取 1，y 取 2，但是如果是 a(1)，则不会报错了，这个时候 x 还是 1，y 则为默认的 3。上面这两种方式，还可以更换参数位置，比如 a(y=8,x=3)用这种形式也是可以的。

上面两个方式是有多少个形参，就传进去多少个实参，但有时候会不确定有多少个参数，此时这种 F(*arg1)方式则比较有用。它以一个*加上形参名的方式来表示这个函式的实参个数不定，可能为 0 个也可能为 n 个。需要注意的一点是，不管有多少个，在函式内部都被存放在以形参名为标识符的 tuple 中。

```
>>> def a(*x):
...     if len(x)==0:
...         print 'None'
...     else:
...         print x
...
>>> a(1)
(1,)
>>> a()
None
>>> a(1, 2, 3)
(1, 2, 3)
>>> a(x=1, y=2, z=3)#但是若这样传递参数，会报类型错误
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: a() got an unexpected keyword argument 'x'
```

F(**arg1)形参名前加 2 个*表示参数在函式内部将被存放在以形式名为标识符的 dictionary 中，这时调用函式的方法则需要采用 arg1=value1,arg2=value2 这样的形式了。

```
>>> def a(**x):
...     if len(x)==0:
...         print 'None'
...     else:
...         print x
...
>>> a()
None
>>> a(x=1, y=2)
{'y': 2, 'x': 1}
>>> a(1, 2)    #使用这种方式会报错
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
TypeError: a() takes exactly 0 arguments (2 given)
```

参数解析

函式参数在调用过程中是按照上述四种方法的优先级依次降低的，也就是先把方式 1 中的 arg 解析，然后解析方式 2 中的 arg=value，再解析方式 3，即把多出来的 arg 这种形式的实参组成个 tuple 传进去，最后把剩下的 key=value 这种形式的实参组成一个 dictionary 传给带 2 个星号的形参，也就方式 4。

```
>>> def test(x, y=1, *a, **b):
...     print x, y, a, b
...
>>> test(1)
1 1 () {}
>>> test(1, 2)
1 2 () {}
>>> test(1, 2, 3, 4)
1 2 (3, 4) {}
>>> test(x=1, y=2)
1 2 () {}
>>> test(1, a=2)
1 1 () {'a': 2}
>>> test(1, 2, 3, a=4)
1 2 (3,) {'a': 4}
>>> test(1, 2, 3, y=4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: test() got multiple values for keyword argument 'y'
```

参数若不能被正确解析，程序会抛出类型错误异常。

小结

函式传递参数时使用*和**分别代表传递的是 tuple 和 dictionary，并且参数是按照一定顺序解析的。