1、如果dict = {1:'a', 2:'b', 3:'c', 3:1}，出现这种写法，相同的key对应不同的value，那么后面的value会覆盖掉前面的value

最终字典为{1:'a', 2:'b', 3:1}

2、字典格式化字符串

对于key值是字符串类型的字典，可以采用下面的方式输出

dict = {‘a':'1', 'b':'2', 'c':'3'}

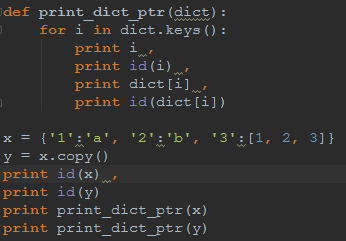
print "value %(a)s" %dict==>"value 1"

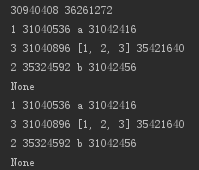
如果value是整形的，可以写成print "value %(a)d" %dict

但是如果key值类型不是字符串类型的，就不能使用这种方式输出

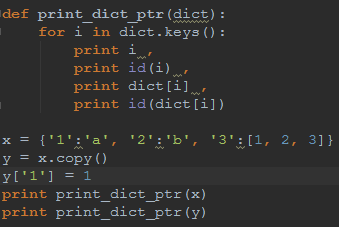
3、copy方法

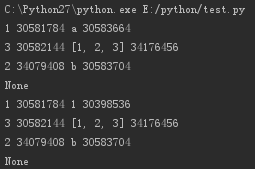
字典浅拷贝





拷贝之后x和y所有的key和value地址都是一样的，但是x和y却是两个不同的字典。由于有这个特性，就产生了下面几个结果。



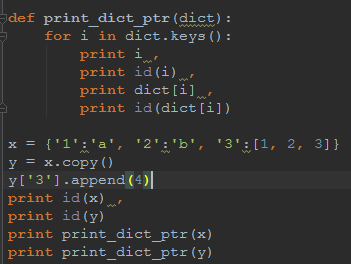


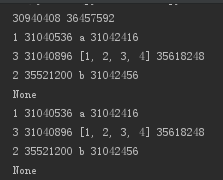
改变了y某一个value的值，x的不会变，因为修改y某一个value值实际上是创建了一个新的内存空间。并没有在原有value的地址上修改。

如果给y新增了一对key-value，x也不会改变，因为x和y是两个不同的数组。

将y中的某一个key删除，x也不会改变。

但是如果修改了y['3']这个value，x就跟一起改变。





同样y['3'][0] = 5或者y['3'].remove(2)也都会改变x里面的值。

因为x和y里面的这个列表实际上是指向同一内存的，y对这个列表的任何操作，都是对该内存的值做了修改，所以x也会一起变。

如果不是列表而是字典，也会产生同样的效果。

4、dict.fromkeys([...])

根据指定的key创造一个新的字典，每个key对应的value默认值为None

方法入参是一个列表，将列表里面的每一个成员作为新字典的key。

也可以根据需要指定默认的value，dict.fromkeys([...], 'unset')

5、dict.get()和dict[key]和dict.setdefault()

dict.get()好处在如果输入一个不存在的key，不会报错，而是返回None。如果用dict[key]这种方法如果key不存在程序就会报错。

dict.get(key, value)可以指定当key不存在时，方法的返回值value，默认是None

dict.setdefault和dict.get()类似

当输入一个不存在的key，dict.setdefault(key, value)方法会返回value，默认是None，同时还会将这个key-value新增到字典里。而dict.get()只会返回，不会修改原有字典

如果输入的key是存在的，那么dict.setdefault会返回其对应的value值，而且不会修改已有的值。