python 字典

字典比列表更适用的情况:
1，表征游戏棋盘的状态，每个键都是由坐标组成的元组。
2，存储文件修改次数，用文件名作为键。
3，数字电话，地址薄。

例如: （字典创建方式）
phonebook={'a':'001','b':'002','c':'003'}
说明:
字典是由多个键及其对应的值构成的对组成（可以把键/值对称为项）
注意:
字典中的键是唯一的（其他类型的映射也是如此），而值不唯一。

---

dict函数（通过关键字创立字典）

基本字典操作;
1.len(d)返回d中项（键-值)的数量。
2.d[k]返回关联到键k上的值。
3.d[k]=v讲值v关联到键k上。
4.del d[k]删除键为k的项。
5. k in d 检查d中是否有键为k的项。

字典和列表的重要区别;
1,键类型
字典的键不一定为整形数据，也可能是字符串，元组或浮点型。
2,自动添加
即使那个键起初在字典中并不存在，也可为他分配一个值，这样字典就会建立新的项。而（在不使用append方法或者其他类似的操作情况下）不能将值关联到列表范围之外的索引上。
3.成员资格
表达式 k in d(字典)查找的是键，而不是值。
表达式 v in l(列表）则用来查找值，而不是索引。

代码清单4-1 字典示例

```
# 简单数据库

# 使用人名作为键的字典。每个人用另一个字典来表示，其键'phone'和'addr'分别表示他们的电话号码和地址。

people = {

    'Alice': {
        'phone': '2341',
        'addr': 'Foo drive 23'
    },

    'Beth': {
        'phone': '9102',
        'addr': 'Bar street 42'
    },

    'Cecil': {
        'phone': '3158',
        'addr': 'Baz avenue 90'
    }
}

# 针对电话号码和地址使用的描述性标签，会在打印输出的时候用到
labels = {
    'phone': 'phone number',
    'addr': 'address'
}

name = raw_input('Name: ')

# 查找电话号码还是地址？使用正确的键：

# 使用正确的键：
if request == 'p': key = 'phone'
if request == 'a': key = 'addr'

# 如果名字是字典中的有效键才打印信息：
if name in people: print "%s's %s is %s." % \
     (name, labels[key], people[name][key])
```

下面是程序的运行示例：

```
Name: Beth
Phone number (p) or address (a)? p
Beth's phone number is 9102.
```

转换说明符的应用:

```
>>> template = '''<html>
    <head><title>%(title)s</title></head>
    <body>
    <h1>%(title)s</h1>
    <p>%(text)s</p>
    </body>'''
>>> data = {'title': 'My Home Page', 'text': 'Welcome to my home page!'}
>>> print template % data
<html>
<head><title>My Home Page</title></head>
<body>
<h1>My Home Page</h1>
<p>Welcome to my home page!</p>
</body>
```

注意 string.Template类（第3章提到过）对于这类应用也是非常有用的。

字典的方法:
1.clear
2.copy
3.fromkey
4.get
.....

多态
"多种形态"，即使不知道变量所引起的对象类型是什么，还可以对其进行操作，而且可以对类或对象的不同表现出不同的行为。

八皇后问题

```
def conflict(state,nextX):
nextY=len(state)
for i in range(nextY):
if abs(state[i]-nextX) in (0,nextY-1):
return True
return False
def queen(num=8,state=()):
for pos in range(num):
if not conflict(state,pos):
if len(state)==num-1:
yield(pos,)
else:
for result in queen(num,state+(pos,)):
yield (pos,)+result
def printso(solution):
def line(pos,len=len(solution)):
return '.'*(pos)+'x'+'.'*(len-pos-1)
for pos in solution:
print line(pos)
import random
printso(random.choice(list(queen(8))))
```

---

shelve方法