第5章

视频讲解

文件应用——公交查询系统

5.1 公交查询系统功能介绍

随着公交系统的逐渐庞大，人们很难得到准确的公交信息，这样给人们的出行就带来了不便。因此，急需一个方便、快捷的公交信息查询方式。本系统提供了换乘查询功能、路线查询功能。乘客可以方便地进行查询，以防乘错车次。本系统主要有4个模块：线路查询、站点查询、换乘查询和后台管理模块。

(1) 线路查询：可以获得要查询公交所通过的各个站点。

(2) 站点查询：通过输入的指定站点查询经过该站点的公交。

(3) 换乘查询：分为公交直达、公交一次换乘，主要体现那些不可直达需要转车的路线的所有换乘方法。

(4) 后台管理：用于管理员登录，添加、修改、删除公交线路等功能。

5.2 程序设计的思路

程序需要存储各公交线路站点信息，这里采用文件存储线路信息，形式如下：

1s%通利公交公司%长途客运西站%建设路国棉六厂%建设路桐柏路站%建设路文化宫路站%建设路工人路站%碧沙岗公园西门%绿城广场%嵩山路伊河路站%解放军测绘学院%市骨科医院%陇海路路寨%陇海路京广客运站%陇海路铁英街站%郑州铁路局%锦荣商贸城%福寿街大同路站%火车站%6：00-21：00 票价1元 ABCD卡有效

1x%火车站%一马路陇海路站%郑州铁路局%陇海路铁英街站%陇海路京广客运站%陇海路路寨%市骨科医院%解放军测绘学院%嵩山路伊河路站%绿城广场%碧沙岗公园西门%碧沙岗%建设路工人路站%建设路文化宫路站%建设路桐柏路站%建设路国棉六厂%华山路建设路站%通利公交公司%6：00-21：00 票价1元 ABCD卡有效

……

其中，1s 为上行，1x 为下行。站点信息之间用%分隔。

公交查询系统程序从文件中读取线路信息，其中线路信息存入 rote 字典中，线路名存入 rotename 列表中。rote 字典存储形式为{线路名：线路经过站点的列表}。

例如，以上线路信息存入 rote 字典时键名为线路名 1s，键对应内容为 1s 线路经过站点的列表：

```
print(rote['1s'])
print(rote['1x'])
```

结果如下：

```
['通利公交公司', '长途客运西站', '建设路国棉六厂', '建设路桐柏路站', '建设路文化宫路站', '建设路工人路站', '碧沙岗公园西门', '绿城广场', '嵩山路伊河路站', '解放军测绘学院', '市骨科医院', '陇海路路寨', '陇海路京广客运站', '陇海路铁英街站', '郑州铁路局', '锦荣商贸城', '福寿街大同路站', '火车站']
['火车站', '一马路陇海路站', '郑州铁路局', '陇海路铁英街站', '陇海路京广客运站', '陇海路路寨', '市骨科医院', '解放军测绘学院', '嵩山路伊河路站', '绿城广场', '碧沙岗公园西门', '碧沙岗', '建设路工人路站', '建设路文化宫路站', '建设路桐柏路站', '建设路国棉六厂', '华山路建设路站', '通利公交公司']
```

当线路信息存入 rote 字典后，线路查询、站点查询、换乘功能实现就可转换为对 rote 字典的相应操作。

视频讲解

5.3　Python 文件的使用

在程序运行时，数据保存在内存的变量里。内存中的数据在程序结束或关机后就会消失。如果想要在下次开机运行程序时还使用同样的数据，就需要把数据存储在不易失的存储介质中，比如硬盘、光盘或 U 盘里。不易失存储介质上的数据保存在以存储路径命名的文件中。通过读/写文件，程序就可以在运行时保存数据。本节学习使用 Python 在磁盘上创建、读写以及关闭文件。

使用的文件与人们日常生活中所使用的记事本很相似。在使用记事本时需要先打开本子，使用后要合上它。打开记事本后，既可以读取信息，也可以向本子里写。不管哪种情况，都需要知道在哪里进行读/写。在记事本中既可以一页页从头到尾地读或写，也可以直接跳转到需要的地方进行读/写。

在 Python 中对文件的操作通常按照以下 3 个步骤进行：

(1) 使用 open()函数打开(或建立)文件，返回一个 file 对象。

(2) 使用 file 对象的读/写方法对文件进行读/写的操作。其中，将数据从外存传输到内存的过程称为读操作，将数据从内存传输到外存的过程称为写操作。

(3) 使用 file 对象的 close()方法关闭文件。

视频讲解

5.3.1　打开(建立)文件

在 Python 中要访问文件，必须打开 Python Shell 与磁盘上文件之间的

连接。当使用 open()函数打开或建立文件时,会建立文件和使用它的程序之间的连接,并返回代表连接的文件对象。通过文件对象,就可以在文件所在磁盘和程序之间传递文件内容,执行文件上所有后续操作。

open()函数用来打开文件。open()函数需要一个字符串路径,表明希望打开文件,并返回一个文件对象。语法如下:

```
fileobj = open(filename[,mode[,buffering]])
```

其中,fileobj 是 open()函数返回的文件对象。参数 filename(文件名)是必写参数,它既可以是绝对路径,也可以是相对路径。mode(模式)和 buffering(缓冲)可选。

mode 是指明文件类型和操作的字符串,可以使用的值如表 5-1 所示。

表 5-1 open()函数中 mode 参数常用值

值	描 述
'r'	读模式。如果文件不存在,则发生异常
'w '	写模式。如果文件不存在,则创建文件再打开;如果文件存在,则清空文件内容再打开
'a'	追加模式。如果文件不存在,则创建文件再打开;如果文件存在,打开文件后将新内容追加至原内容之后
'b'	二进制模式,可添加到其他模式中使用
'+'	读/写模式,可添加到其他模式中使用

下面举例说明 open()函数的使用。

先用记事本创建一个文本文件,取名为 hello.txt。输入以下内容并保存在文件夹 d:\python 中:

```
Hello!
Henan Zhengzhou
```

在交互式环境中输入以下代码:

```
>>> helloFile = open("d:\\python\\hello.txt")
```

这条命令将以读取文本文件的方式打开放在 D 盘下 Python 文件夹下的 hello.txt 文件。“读模式”是 Python 打开文件的默认模式。当文件以读模式打开时,只能从文件中读取数据而不能向文件写入或修改数据。

当调用 open()函数时将返回一个文件对象,在本例中文件对象保存在 helloFile 变量中。

```
>>> print (helloFile)
<_io.TextIOWrapper name = 'd:\\python\\hello.txt' mode = 'r' encoding = 'cp936'>
```

打印文件对象时可以看到文件名、读/写模式和编码格式。cp936 就是指 Windows 系统里第 936 号编码格式,即 GB 2312。接下来就可以调用 helloFile 文件对象的方法读取文件中的数据了。

5.3.2　读取文本文件

文件打开后，才能读/写或读并且写文件内容。读取文件内容可通过调用文件 file 对象的多个方法实现。

1. read()方法

不设置参数的 read()方法将整个文件的内容读取为一个字符串。read()方法一次读取文件的全部内容，性能根据文件大小而变化，比如 1GB 的文件读取时需要使用同样大小的内存。

【例 5-1】　调用 read()方法读取 hello.txt 文件中的内容。

```
helloFile = open("d:\\python\\hello.txt")
fileContent = helloFile.read()
helloFile.close()
print(fileContent)
```

输出结果：

```
Hello!
Henan Zhengzhou
```

也可以设置最大读入字符数来限制 read()函数一次返回的大小。

【例 5-2】　设置参数一次读取 3 个字符读取文件。

```
helloFile = open("d:\\python\\hello.txt")
fileContent = ""
while True:
    fragment = helloFile.read(3)
    if fragment == "":          #或者 if not fragment
        break
    fileContent += fragment
helloFile.close()
print(fileContent)
```

当读到文件结尾时，read()方法会返回空字符串，此时 fragment==""成立，退出循环。

2. readline()方法

readline()方法从文件中获取一个字符串，每个字符串就是文件中的每一行。

【例 5-3】　调用 readline()方法读取 hello.txt 文件的内容。

```
helloFile = open("d:\\python\\hello.txt")
fileContent = ""
while True:
   line = helloFile.readline()
   if line == "":            #或者 if not line
       break
   fileContent += line
helloFile.close()
print(fileContent)
```

当读取到文件结尾时，readline()方法同样返回空字符串，使得 line==""成立，跳出循环。

3. readlines()方法

readlines()方法返回一个字符串列表，其中的每一项是文件中每一行的字符串。

【例 5-4】 使用 readlines()方法读取文件内容。

```
helloFile = open("d:\\python\\hello.txt")
fileContent = helloFile.readlines()
helloFile.close()
print(fileContent)
for line in fileContent:                    #输出列表
    print(line)
```

readlines()方法也可以设置参数，指定一次读取的字符数。

5.3.3 写文本文件

写文件与读文件相似，都需要先创建文件对象连接。所不同的是，写文件在打开文件时是以“写”模式或“添加”模式打开。如果文件不存在，则创建该文件。

视频讲解

与读文件时不能添加或修改数据类似，写文件时也不允许读取数据。“w”为写模式，打开已有文件时，会覆盖文件原有内容，从头开始，就像用一个新值覆写一个变量的值。

```
>>> helloFile = open("d:\\python\\hello.txt","w")        #"w"为写模式,打开已有文件时会覆盖
                                                         #文件原有内容
>>> fileContent = helloFile.read()
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    fileContent = helloFile.read()
IOError: File not open for reading
>>> helloFile.close()
>>> helloFile = open("d:\\python\\hello.txt")
>>> fileContent = helloFile.read()
>>> len(fileContent)
0
>>> helloFile.close()
```

由于写模式打开已有文件，文件原有内容会被清空，所以再次读取内容时长度为 0。

1. write ()方法

write()方法将字符参数写入文件。

【例 5-5】 用 write()方法写文件。

```
helloFile = open("d:\\python\\hello.txt","w")
helloFile.write("First line.\nSecond line.\n")
helloFile.close()
helloFile = open("d:\\python\\hello.txt","a")
helloFile.write("third line. ")
```

```
helloFile.close()
helloFile = open("d:\\python\\hello.txt")
fileContent = helloFile.read()
helloFile.close()
print(fileContent)
```

运行结果：

```
First line.
Second line.
third line.
```

当以写模式打开文件 hello.txt 时，文件原有内容被覆盖。调用 write()方法将字符串参数写入文件，这里“\n”代表换行符。关闭文件之后再次以添加模式打开文件 hello.txt，调用 write 方法写入的字符串"third line."被添加到了文件末尾。最终以读模式打开文件后读取到的内容共有三行字符串。

注意：write()方法不能自动在字符串末尾添加换行符，需要自己添加“\n”。

【例 5-6】 完成一个自定义函数 copy_file()，实现文件的复制功能。

copy_file()函数需要两个参数，指定需要复制的文件 oldfile 和文件的备份 newfile。分别以读模式和写模式打开两个文件，从 oldfile 一次读入 50 个字符并写入 newfile。当读到文件末尾时 fileContent==""成立，退出循环并关闭两个文件。

```
def copy_file(oldfile,newfile):
    oldFile = open(oldfile,"r")
    newFile = open(newfile,"w")
    while True:
        fileContent = oldFile.read(50)
        if fileContent == "":           #读到文件末尾时
            break
        newFile.write(fileContent)
    oldFile.close()
    newFile.close()
    return
copy_file("d:\\python\\hello.txt","d:\\python\\hello2.txt")
```

2. writelines()方法

writelines()方法向文件写入一个序列字符串列表，如果需要换行则要自己加入每行的换行符。

【例 5-7】 writelines()方法示例程序，实现将列表内容写入 log.txt 文件。

```
obj = open("log.txt","w")
list02 = ["11","test","hello","44","55"]
obj.writelines(list02)
obj.close()
```

运行结果是生成一个 log.txt 文件，内容是"11testhello4455"，可见没有换行。另外，应

注意 writelines()方法写入的序列必须是字符串序列,整数序列会产生错误。

5.3.4 文件内移动

无论读或写文件,Python 都会跟踪文件中的读/写位置。在默认情况下,文件的读/写都从文件的开始位置进行。Python 提供了控制文件读/写起始位置的方法,使得用户可以改变文件读/写操作发生的位置。

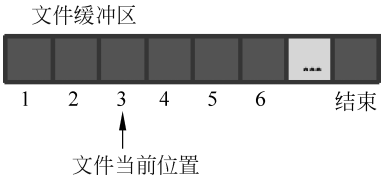


图 5-1 文件当前位置

当使用 open()函数打开文件时,open()函数在内存中创建缓冲区,将磁盘上的文件内容复制到缓冲区。文件内容复制到文件对象缓冲区后,文件对象将缓冲区视为一个大的列表,其中的每一个元素都有自己的索引,文件对象按字节对缓冲区索引计数。同时,文件对象对文件当前位置,即当前读/写操作发生的位置进行维护,如图 5-1 所示。许多方法隐式使用当前位置。比如调用 readline()方法后,文件当前位置移动到下一个回车处。

Python 使用一些函数跟踪文件当前位置。tell()函数可以计算文件当前位置和开始位置之间的字节偏移量。

```
>>> exampleFile = open("d:\\python\\example.txt","w")
>>> exampleFile.write("0123456789")
>>> exampleFile.close()
>>> exampleFile = open("d:\\python\\example.txt")
>>> exampleFile.read(2)
'01'
>>> exampleFile.read(2)
'23'
>>> exampleFile.tell()
4
>>> exampleFile.close()
```

这里 exampleFile.tell()函数返回的是一个整数 4,表示文件当前位置和开始位置之间有 4 字节偏移量。因为已经从文件中读取 4 个字符了,所以有 4 字节偏移量。

seek()函数设置新的文件当前位置,允许在文件中跳转,实现对文件的随机访问。

seek()函数有两个参数:第一个参数是字节数;第二个参数是引用点。seek()函数将文件当前指针由引用点移动指定的字节数到指定的位置。语法如下:

```
seek(offset[,whence])
```

说明:offset 是一个字节数,表示偏移量。引用点 whence 有如下 3 个取值。

- 文件开始处为 0,也是默认取值,意味着使用该文件的开始处作为基准位置,此时字节偏移量必须非负。
- 当前文件位置为 1,则是使用当前位置作为基准位置。此时偏移量可以取负值。
- 文件结尾处为 2,则该文件的末尾将被作为基准位置。

5.3.5　文件的关闭

文件使用后，必须调用 close()方法将其关闭。关闭文件是取消程序和文件之间连接的过程，内存缓冲区的所有内容将写入磁盘，因此必须在使用文件后关闭文件确保信息不会丢失。

要确保文件关闭，可以使用 try-finally 语句，在 finally 子句中调用 close()方法：

```
helloFile = open("d:\\python\\hello.txt","w")
try:
    helloFile.write("Hello,Sunny Day!")
finally:
    helloFile.close()
```

也可以使用 with 语句自动关闭文件：

```
with open("d:\\python\\hello.txt") as helloFile:
    s = helloFile.read()
print(s)
```

with 语句可以打开文件并赋值给文件对象，之后就可以对文件进行操作。文件会在语句结束后自动关闭，即使是由于异常引起的结束也是如此。

5.4　程序设计的步骤

```
import math
rote = {}              #线路信息的字典
rotename = []          #线路名的列表
```

readRote()是读取线路信息的函数。readRote()从 gongjiao.txt 文件中读取线路信息(一行一个线路)，按%分隔后，第一个数据项是线路名(如 1s,201x)存入 rotename，以后数据项存入 rote 字典相应键值对中。其中 rote 字典中键为线路名，对应值为线路经过站点列表。

```
def readRote():
    fp = open("gongjiao.txt",'r', encoding = 'gbk')   #注意文本文件编码,也可能为 UTF-8
    #UnicodeDecodeError: 'gbk' codec can't decode byte 0xbf: illegal multibyte sequence
    fileContent = ""
    while True:
        line = fp.readline()
        if line == "":                          #或者 if not line
            break
        fileContent += line + '\n';
        list1 = line.split(" % ")               #按 % 分隔
        rotename.append(list1[0])               #线路名(如 1s,201x)存入 rotename
```

```
        rote[list1[0]] = list1[1:-1]        #list[:-1]   #增加键对
    fp.close()                              #文件关闭
    #print(rote.values())
    #print(rote['1x'])
```

findRote()是线路查询功能的函数。判断线路信息字典 rote 是否有此线路名的键,如果存在就打印出此线路名的键对应值(即此线路信息)。

```
def findRote():                             #线路查询功能
    findRoteName = input("请输入查询线路名")
    if findRoteName in rote:
        print(rote[findRoteName])
    else:
        print("输入有错误!没有你要查询的线路")
```

findStation()是站点查询功能的函数。仅仅需要遍历线路信息的字典 rote,判断键值 value(线路经过的站点)是否有查询站点名,如果有则打印出来键 key(线路名)。

```
def findStation():                          #站点查询功能
    stationName = input("请输入查询站点名")
    print('经过此站点的线路有:',end = ' ')
    #遍历字典
    for key,value in rote.items():
        if(stationName in value):
            print(key,end = '; ')
    print()
```

huanRote()是换乘查询功能的函数。此函数功能比较复杂,首先需要判断起始站点到终点之间是否有直达线路,无直达线路则需要换乘。

直达判断比较简单,仅仅需要判断起始站点到终点是否在同一个线路上。

换乘功能算法是找出经过起始站点的线路存入 S 列表,经过终点的线路存入 D 列表。然后判断 S 列表中 key1 线路和 D 列表中 key2 线路是否有相同站点 sameStationName。如果有相同站点 sameStationName,则乘客从起始站点经过 key1 线路到 sameStationName 相同站点,换成 key2 线路可以到达终点。为了便于用户选择换乘,并计算经过的站点数。

```
def huanRote():                                         #换乘查询功能
    startStationName = input("请输入起始站点名")        #"火车站"
    endStationName = input("请输入终点名")              #"绿城广场"
    canGo = False
    #直达判断
    for key,value in rote.items():
        if (startStationName in value) and (endStationName in value):   #在同一线路上
            canGo = True
            print("公交可直达线路",key)
    if canGo == False:
        print("公交不可直达")
        #换乘
```

```
        S = []       #经过起始站点的线路
        D = []       #经过终点的线路
        for key,value in rote.items():
            if (startStationName in value):
                S.append(key)
        for key,value in rote.items():
            if (endStationName in value):
                D.append(key)
        #判断线路之间是否有相同站点
        for key1 in S:
          for key2 in D:
              if hasSameStation(key1,key2):
                 sameStationName = hasSameStation(key1,key2)
                 n1 = stationNum(startStationName,sameStationName,key1)
                 n2 = stationNum(endStationName,sameStationName,key2)
                 print("经过 key1 线路" + key1 + n1 + "站到" + hasSameStation(key1,key2) + "换
成 key2 线路" + key2 + n2 + "站到达" + endStationName)

#判断线路之间是否有相同站点
def hasSameStation(key1,key2):
    for stationName in rote[key1]:
        if stationName in rote[key2]:
            return stationName
    return False
#两个站点之间的站数
def stationNum(Station1,Station2,roteName):
    for i in range(len(rote[roteName])):
        stationName = rote[roteName][i]
        if Station1 == stationName:
            i1 = i
        if Station2 == stationName:
            i2 = i
    return str(int(math.fabs(i1 - i2)))
```

main()是主函数，是公交查询系统程序的入口函数。主要通过循环实现用户功能选择。

```
def main(): #主函数
     readRote()
     while True:
           print( "**********")
           print( u"--------菜单----------")
           print( u"线路查询------------1")
           print( u"站点查询------------2")
           print( u"换乘查询------------3")
           print( u"添加线路信息--------4")
           print( u"退出程序------------0")
           print( "**********")
           nChoose = input("请输入你的选择:")
```

```
            if nChoose == "1":
                findRote()
            elif nChoose == "2":
                findStation()
            elif nChoose == "3":
                huanRote()
            elif nChoose == "4":
                  #添加线路信息
                  fp = open("gongjiao.txt",'a', encoding = 'gbk')
                  addRoteName = input("请输入添加线路名")
                  addRoteContent = input("请输入添加线路经过的站点及运行时间,%分隔\n")
                  fp.write(addRoteName + " % " + addRoteContent + "\n")
                  fp.close()
rotename.append(addRoteName)                          #添加到线路名的列表
    rote[addRoteName] = addRoteContent                #添加到线路信息的字典
elif nChoose == "0":
rotename.append(addRoteName)                          #添到线路名的列表
             rote[addRoteName] = addRoteContent       #添到线路信息的字典
        elif nChoose == "0":
    break
```

最后是调用 main()函数。

```
main()          #该程序的入口函数
```

运行结果如下：

```
**********************
--------菜单---------
线路查询------------1
站点查询------------2
换乘查询------------3
添加线路信息--------4
退出程序------------0
**********************
请输入你的选择:1
请输入查询线路名 3s
['黄岗寺', '武警医院分院(嵩山路)站', '亚星盛世家园', '长江路淮南街站', '长江路大学路站', '长江路建云路站', '长江路京广路站', '长江路碧云路站', '长江路邱寨', '长江路花寨路站', '十里铺', '紫荆山南路', '紫荆山路城东路站', '紫荆山路新郑路站', '紫荆山路航海路站', '紫荆山路金城街站', '二里岗南街紫荆山路站', '二里岗南街城东路站', '城东路豫筑路站', '城东路货栈街站', '陇海路货栈北街站', '陇海路东明路站', '陇海路未来路站', '未来路凤凰路站', '中博家具中心', '郑汴路商品大世界', '郑汴路商贸路站', '东建材']

请输入查询站点名十里铺
经过此站点的线路有:3x; 38s; 80s; 80x; 38x; 3s;

请输入你的选择:3
请输入起始站点名市第五人民医院
请输入终点名绿城广场
公交不可直达
经过 key1 线路 K811x 7 站到火车站换成 key2 线路 12s 4 站到达绿城广场
经过 key1 线路 K811x 7 站到火车站换成 key2 线路 68x 5 站到达绿城广场
```

```
经过 key1 线路 K811x 4 站到大石桥换成 key2 线路 103x 5 站到达绿城广场
经过 key1 线路 K811x 7 站到火车站换成 key2 线路 B12x 5 站到达绿城广场

请输入你的选择:4
请输入添加线路名 100s
请输入添加线路经过的站点及运行时间,%分隔
郑上路站%元通纺织城(绕城路)%绕城路铁炉%化工路绕城路站%化工路腊梅路站%化工路怡红路站%化工路雪松路站%化工路西百炉屯站%化工路东百炉屯站%中原制药厂%化工中路%化工路瑞达路站%化工路西流湖%化工路西环路站%化工路白庄%冉屯路秦岭路站%冉屯路五龙口%冉屯路冉屯东路站%冉屯路桐柏路站%农业路朱屯东路站%农业路沙口路站%农业路南阳路站%南阳路群英路站%南阳路群办路站%南阳路东风路站%丰乐路东风路站%丰乐路宋寨南街站%南阳路张寨%6:30-20:00 票价 1 元 ABCD 卡有效
```

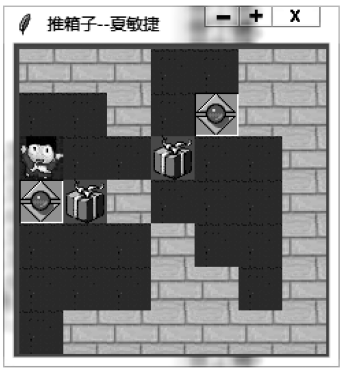
请读者完善此程序,增加以下功能:

```
删除线路信息 ---------5
修改线路信息 ---------6
```

5.5　文件使用拓展实例——游戏地图存储

在游戏开发中往往需要存储不同关卡的游戏地图信息,例如推箱子、连连看等游戏。这里以推箱子游戏地图存储为例来说明游戏地图信息如何存储到文件中并读取出来。

图 5-2 所示的推箱子游戏,可以看成 7×7 的表格,这样如果按行/列存储到文件中,就可以把这一关游戏地图存入到文件中了。



墙	墙	墙			墙	墙
		墙			墙	墙
						墙
		墙				墙
			墙			墙
			墙	墙		墙
	墙	墙	墙	墙	墙	墙

图 5-2　推箱子游戏

为了表示方便,每个格子状态值分别用常量 Wall(0)代表墙,Worker(1)代表人,Box(2)代表箱子,Passageway(3)代表路,Destination(4)代表目的地,WorkerInDest(5)代表人在目的地,RedBox(6)代表放到目的地的箱子。文件中存储的原始地图中格子的状态值采用相应的整数形式存放。图 5-2 所示的推箱子游戏界面的对应数据如下:

0	0	0	3	3	0	0
3	3	0	3	4	0	0
1	3	3	2	3	3	0
4	2	0	3	3	3	0
3	3	3	0	3	3	0
3	3	3	0	0	3	0
3	0	0	0	0	0	0

1. 将地图信息写入文件

只需要使用 write()方法按行/列(这里按行)存入到文件 map1.txt 中即可。

```
import os
myArray1 = []
#将地图信息写入文件
helloFile = open("map1.txt","w")
helloFile.write("0,0,0,3,3,0,0\n")
helloFile.write("3,3,0,3,4,0,0\n")
helloFile.write("1,3,3,2,3,3,0\n")
helloFile.write("4,2,0,3,3,3,0\n")
helloFile.write("3,3,3,0,3,3,0\n")
helloFile.write("3,3,3,0,0,3,0\n")
helloFile.write("3,0,0,0,0,0,0\n")
helloFile.close()
```

2. 从地图文件读取信息

只需要按行从文件 map1.txt 中读取即可得到地图信息。本例中将信息读取到二维列表中存储。

```
#读文件
helloFile = open("map1.txt","r")
myArray1 = []
while True:
    line = helloFile.readline()
    if line == "":                          #或者 if not line
        break
    line = line.replace("\n","")            #将读取的 1 行中最后的换行符去掉
    myArray1.append(line.split(","))
helloFile.close()
print(myArray1)
```

结果是：

```
[['0', '0', '0', '3', '3', '0', '0'], ['3', '3', '0', '3', '4', '0', '0'], ['1', '3', '3', '2', '3','3',
'0'], ['4', '2', '0', '3', '3', '3', '0'], ['3', '3', '3', '0', '3', '3', '0'], ['3', '3', '3', '0', '0',
'3', '0'], ['3', '0', '0', '0', '0', '0', '0']]
```

在图形化推箱子游戏中，根据数字代号用对应图形显示到界面上，即可完成地图读取任务。