第6章 文件及目录操作

主要内容

- 文件的打开与关闭
- 文本文件的读写
- 二进制文件的读写
- 文件读写编程实例

➢ 文件的打开与关闭

◆ 打开

文件必须存在　　文件不存在，可以创建

fileObject=open(filename,mode)

filename: 包括路径名，如'c:\\pysrc\\msg.txt'

mode: r, w, a, rb, wb, ab, +

```
>>> f1=open('d:\\grade.txt')
>>> f1=open('d:\\grade.txt',"w+")
```

◆ 关闭

fileObject.close()

当打开文件有写入信息时，不关闭文件，在程序结束时，有可能出写文件不正确的情形。

➢ 文件的打开与关闭

◆ 关闭

```
with open (filename,mode) as f:
    <利用f读写文件>
```

with中的语句执行完毕后，自动关闭文件

```
with open('d:\\grade.txt',"w+") as f:
    f.write('very good!')
```

◆ 打开异常

```
try:
    open (filename,mode)
except FileNotFoundError:
    print("……")
```

```
import os
if not os.path.exists('abc.txt'):
    print('file is not exist!')
```

第6章 文件及目录操作

主要内容

- 文件的打开与关闭
- 文本文件的读写
- 二进制文件的读写
- 文件读写编程实例

➢ 向文件写入信息

【例6-1】利用文件读写函数写入字符串信息

```
try:
    with open("d:\\grade.txt",'w') as f:
        f.write('山东\n威海\n')
        L=['文化西路2号','哈尔滨工业大学(威海)']
        f.writelines(L)
        newL=[line +'\n' for line in L]
        f.writelines(newL)
except:
    print('文件创建失败')
```

➢ 从文件读取信息

【例6-2】利用文件读写函数读出字符串信息

```
try:
    with open("d:\\grade.txt",'r') as f:
        s1=f.read(2)
        print(s1)
        print(f.tell())
        f.seek(6,0)
        s1=f.read(4)
        print(s1)
except FileNotFoundError:
    print('文件未发现')
```

```
try:
    with open("d:\\grade.txt",'r') as f:
        i=1
        for line in f:
            print('第',str(i),'行:',line, end="")
            i=i+1
except FileNotFoundError:
    print('文件未发现')
```

第6章 文件及目录操作

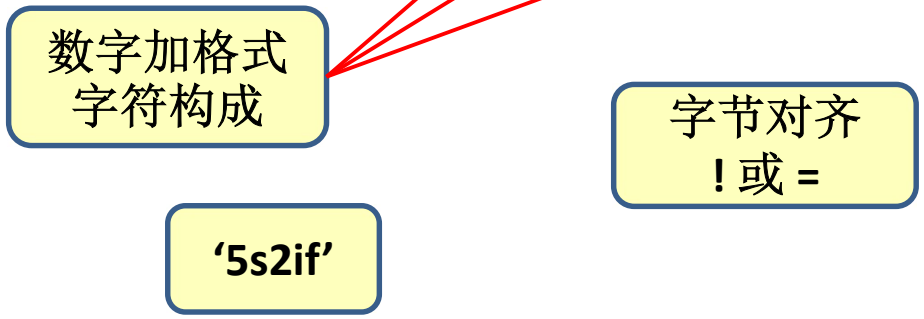
主要内容

- 文件的打开与关闭
- 文本文件的读写
- 二进制文件的读写
- 文件读写编程实例

➢ 使用struct读写二进制文件

- ◆ 字节流：以字节为单位的一系列二进制数据为字节流。
- ◆ **struct** 的功能：将一系列不同类型的数据封装成一段字节流，或者相反。

(1) 数据封装函数　pack(fmt, v1, v2, ...)
(2) 数据解封函数　unpack(fmt, string)
(3) 格式大小计算函数　calcsize(fmt)



➢ 使用struct读写二进制文件

【例6-3】使用struct模块读写二进制文件

```
import struct
def m():
    with open('d:\\grade.bin','wb') as f:
        s1='张三疯'.encode('utf-8')
        s2='机械无忌'.encode('utf-8')
        byte=struct.pack("!10s12si",s1,s2,128)
        f.write(byte)
    with open('d:\\grade.bin','rb') as f:
        a,b,c=struct.unpack("!10s12si",f.read(12+18+4))
        print(a.decode("utf-8",'ignore'),b.decode("utf-8",
                                                  'ignore'),c)

m()
```

非字符串不需编码

含中文时要加

➢ 使用struct读写二进制文件

【例6-4】读取位图文件的类型及大小

```
import struct
import os
def m():
    pathfile=os.getcwd()+tes.bmp
    with open(pathfile,'rb') as f:
        a=struct.unpack(“2s",f.read(2))
        b=struct.unpack("i",f.read(4))
    print(a[0].decode(),b[0])
m()
```

➢ 使用pickle实现数据序列化

◆ 序列化：将任意python对象转化为一系列的字节存储到文件中。反序列化恰好相反

◆ 序列化对象

(1) 基本类型：布尔型、整型、浮点型、复数型等

(2) 对象：字符串、列表、元组、字典和集合等

(3) 其他：函数、类、类的实例

◆ 序列化方法

(1) dump() 将对象转换成字节流存入文件

dump(obj, file[,protocal])

(2) load()解析文件 中的字节流(反序列化)

load(file)

➢ 使用pickle实现数据序列化

【例6-5】使用pickle模块实现序列化和反序列化

```
import pickle
data={'Jack':[32,'male','market'],'Suan':[28,'female','AD.']}
L=[1,2,3]
L.insert(1,'zxd')
with open('d:\\p1.txt','wb') as f:
    pickle.dump(data,f)
    pickle.dump(L,f,-1)
with open('d:\\p1.txt','rb') as f:
    data=pickle.load(f)
    print(data)
    L=pickle.load(f)
    print(L)
```

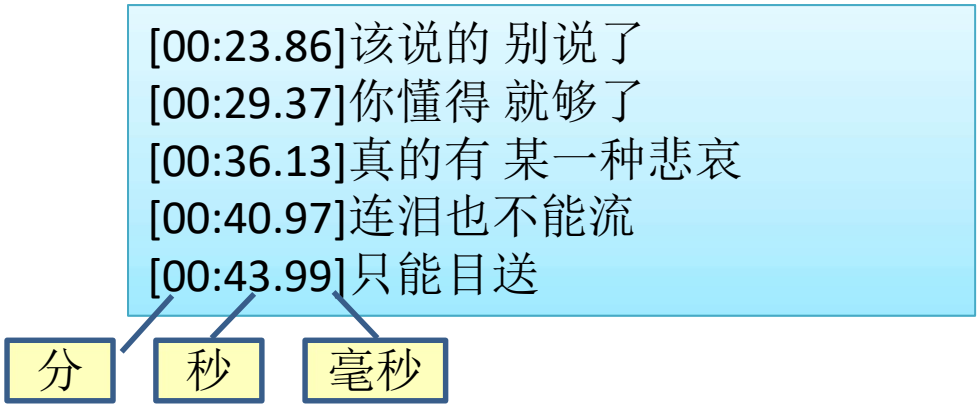
第6章 文件及目录操作

主要内容

- 文件的打开与关闭
- 文本文件的读写
- 二进制文件的读写
- 文件读写编程实例

【例6-5】读取mp3歌词文件，将时间标签转换成毫秒形式，并将每一句歌词读出来，按时间顺序以“时间（毫秒为单位）歌词”的形式显示每一句

【问题分析】mp3歌词为LRC格式，如下所示：



计算步骤：时间校验；时间转换；排序

【例6-6】读取mp3歌词文件

```
def readLRC(filename):
    with open(filename,'r',encoding='UTF-8') as f:
        res={}
        L=f.readline()
        while L!='':
            if(L[1:3].isdigit() and L[4:6].isdigit() and L[7:9].isdigit()
                and L[3]==':' and L[6]=='.'):  #校验
                t1=(int(L[1:3])*60+int(L[4:6]))*1000+int(L[7:9])*10 #转换
                res[t1]=L[10:].rstrip()
            L=f.readline()
        return res
def m():
    fname=input('输入MP3歌词文件名：')
    lrcD=readLRC(fname)
    for key in sorted(lrcD):  #排序
        print(key,lrcD[key])
m()
```

本章小结

- 文件的打开与关闭
- 文本文件的读写
- 二进制文件的读写
- 文件读写编程实例