重庆邮电大学

第7章 文件和I/O

主 讲 人：

2019/3/23

目录

概述

本章要解决的问题如下：

- 什么是文件
- 怎样操作文件
- 怎样操作目录

1. 文件基础知识

1.1 什么是文件

文件是存储在外部介质上的数据集合，与文件名相关联。

按文件中的数据组织形式可以把文件分为两类：

- 文本文件
- 二进制文件

1. 文件基础知识

1.2 文件的打开或创建

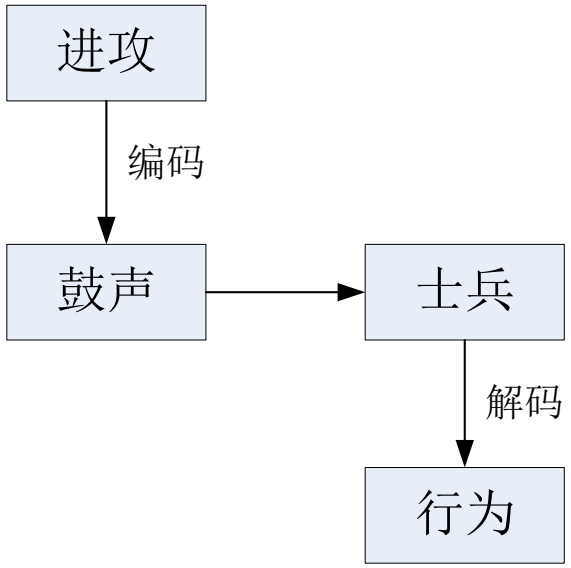
文件变量名=open(文件名[, 打开方式[, 缓冲区]])
示例如下：

```
• f1=open('file1.txt', 'r')
• f2=open('file2.txt', 'w')
```

1. 文件基础知识

1.3 字符编码

编码是用数字来表示符号和文字的一种方式，是符号、文字存储和显示的基础。信息传递与编码关系的例子如右图所示：



1. 文件基础知识

常见的编码

- UTF-8
- GB2312
- GBK
- CP936
- Unicode

1.4 文件的写入

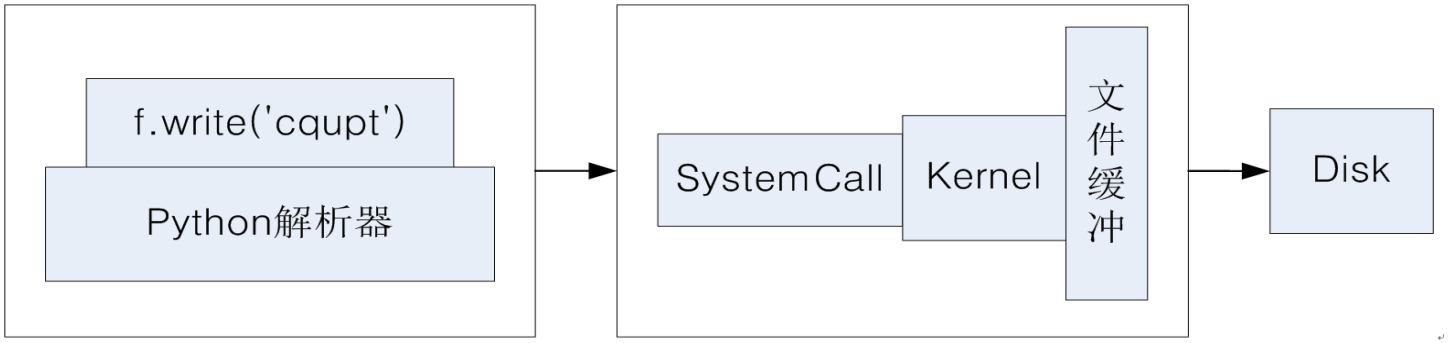
1. 文本文件的写入

以下两种方法可以进行文本文件的写入。

- write(str)：将字符串str写入文件。
- writelines(sequence_of_strings)：写多行到文件，其中sequence_of_strings是由字符串所组成的列表，或者迭代器。

1. 文件基础知识

文件的写过程与存储如下图：



1. 文件基础知识

示例

把字符串“重庆邮电大学123@cqupt”写入文件F7_1.txt中，采用GBK编码，显示文件的长度（总字节），默认采用的是GBK编码。

1. 文件基础知识

程序

```
#Exp7_1.py
#coding=GBK
f=open('F7_1.txt','w')
f.write('重庆邮电大学123@cqupt')
f.seek(0,2)       #把文件指针移到文件尾
length=f.tell()   #会返回文件尾的位置，其值刚好等于文件长度
f.close()
print ('文件长度=',length)
```

程序运行结果

文件长度=21

1. 文件基础知识

示例

在Windows系统中，把字符串“重庆邮电大学123@cqupt ”用UTF-8编码写入文件F7_2.txt中，并显示文件的长度（总字节数）。

1. 文件基础知识

程序

```
import codecs          #自然语言编码转换模块
#Exp7_2.py
#coding=UTF-8
s='重庆邮电大学123@cqupt'
f=codecs.open('F7_2.txt','w','UTF-8') #UTF-8编码方式
f.write(s)
f.seek(0,2) #把文件指针移到文件尾
length=f.tell() #文件尾的位置，其值刚好等于文件长度（字节数）
f.close()
print('文件长度=',length)
```

程序运行结果

文件长度=27

1. 文件基础知识

示例

在文件F7_2.txt末尾追加两行内容。

```
#Exp7_3.py
f=open('F7_2.txt','a+')
s='重邮在山上\n重邮景色很美\n'
f.write(s)
f.close()
```

1.4 文件的写入

2. 二进制文件的写入

以下两种方法可以进行二进制文件的写入。

- 一种是通过struct模块的pack()方法把数字和布尔值转换成字节串（以字节为单位的字符串），然后用write()方法写入二进制文件中，字符串则可直接写入二进制文件中。pack()方法的语法是：pack（格式串，数字对象表）。
- 另外一种是用pickle模块的dump()方法直接把对象转换为字节串（bytes）并存入文件中。

1. 文件基础知识

示例

把1个整数、1个浮点数、1个布尔型对象、1个字符串存入二进制文件F7_4.dat中。

```
#Exp7_4.py
#coding=UTF-8
import struct
n=102400000
x=10.24
b=True
s='重庆邮电大学123@cqupt'
sn=struct.pack('if?', n, x, b)
#把整数n、浮点数x、布尔对象b依次转换为字节串
```

1. 文件基础知识

程序续

```
f=open('F7_4.dat','wb')
f.write(sn)   #写入字节串
f.write(s)    #字符串可直接写入
f.close()
```

1. 文件基础知识

示例

把1个整数、1个浮点数、1个字符串、1个列表、1个元组、1个集合、1个字典存入二进制文件F7_5.dat中。

```
#Exp7_5.py
#coding=UTF-8
import pickle
f=open('F7_5.dat','wb')
n=7
i=102400000
a=10.24
s='中国人民123abc'
lst=[[1,2,3],[4,5,6J,[7,8,9]]
```

1. 文件基础知识

```
tu=(-5,10,8)
coll={4,5,6}
dic={'a':'apple','b':'banana','g':'grape','o':'orange'}
try:
    pickle.dump(n,f) #表示后面将要写入的数据个数
    pickle.dump(i,f) #把整数i转换为字节串，并写入文件
    pickle.dump(a,f)
    pickle.dump(lst,f)
    pickle.dump(i,f)
    pickle.dump(a,f)
    pickle.dump(lst,f)
    pickle.dump(i,f)
    pickle.dump(tu,f)
    pickle.dump(coll,f)
    pickle.dump(dic,f)
except:
    print('写文件异常!')  #如果写文件异常则跳到此处执行
f.close()
```

1.5 文件的读取

1. 文本文件的读取

- read([size])：读取文件，如果文件大于size个字节，则只读取size个字节；如果小于size个字节，则读取完；如果不设置size，则默认读取全部。
- readline([size])：读取一行
- readlines([size])：读取完文件

1. 文件基础知识

示例

读取文件F7_1.txt的前8个字节，并显示：

```
#Exp7_6.py
f=open('F7_1.txt','r')
s=f.read(11)#读取文件的前11个字节
f.close()
print('s=',s)
print('字符串s的长度(字符个数)=', len(s))
```

程序运行结果

```
s=重庆邮电大学123@c
字符串s的长度(字符个数)= 11
```

1. 文件基础知识

示例

读取文件F7_1.txt的全部内容，并显示：

```
#Exp7_7.py
f=open('F7_1.txt','r')
s=f.read()#读取文件全部内容
f.close()
print('s=',s)
```

程序运行结果

s=重庆邮电大学123@cqupt

1. 文件基础知识

示例

使用readline()读取文件F7_2.txt的每一行，并显示：

```
#Exp7_8.py
f=open('F7_2.txt','r')
while True:
    line=f.readline()
    if line=='':
        break
print(line),
#逗号不会产生换行符，但文件中有换行符，因此会换行
f.close()
```

程序运行结果

重庆邮电大学123@cqupt
重邮在山上
重邮景色很美

1. 文件基础知识

示例

使用readlines()读取文件F7_2.txt的每一行，并显示：

```
#Exp7_9.py
f=open('F7_2.txt','r')
s=f.readlines()
for line in s:
    print(line),
  #逗号不会产生换行符，但文件中有换行符，因此会换行
    f.close()
```

程序运行结果

```
重庆邮电大学123@cqupt
重邮在山上
重邮景色很美
```

1.5 文件的读取

2. 二进制文件的读取

- 用struct模块的pack()方法完成转换而写的文件，应该用read()方法读出相应数据的字节串，然后通过代码还原数据。字符串不用还原。
- 用pickle模块的dump()方法完成转换而写的文件，应该用pickle模块的load()方法还原对象。

1.5 文件的读取

（1）使用read()方法

- 字符串可以直接读出，数字和布尔对象需要用struct模块的unpack()方法还原。
- unpack()方法的语法是：

  unpack(格式串，字符串表)。

1. 文件基础知识

读取二进制文件F7_4.dat中的数据，并显示：

```
#Exp7_10.py
import struct
f=open('F7_4.dat','rb')
sn=f.read(9)
tu=struct.unpack('if?',sn)
#从字节串sn中还原出1个整数、1个浮点数和1个布尔值，并返回元组。
print(tu)
n=tu[0]
x=tu[1]
bl=tu[2]
print('n=',n)
print('x=',x)
print('bl=',bl)
s=f.read(9)
f.close()
print('s=',s)
```

程序运行结果

```
 (102400000,10.239999771118164,True)
n=102400000
x=10.239999771118164
bl=True
s='重庆邮电大学123@cqupt'
```

1.5 文件的读取

（2）使用pickle模块的load()方法

pickle模块的load(f)方法可以从二进制文件中读取对象的字节串并还原对象，使用起来非常方便。参数f是文件对象，该方法返回还原后的对象。

1. 文件基础知识

示例

读取二进制文件F7_5.dat中的数据，并显示：

```
#Exp7_11.py
import pickle
f=open('F7_5.dat','rb')
n=pickle.load(f)      #读取文件的数据个数
i=0
while i<n:
x=pickle.load(f)
print(x)
i=i+1
f.close()
```

```
102400000
10.24
中国人民123abc
[1,2,3],[4,5,6J,[7,8,9]
(-5,10,8)
set([4,5,6])
{'a':'apple','b':'banana','g':'grape','o':'orange'}
```

程序运行结果

1. 文件基础知识

1.5　文件的读取

3. 文件指针的移动

- 写入文件后，必须打开才能读取写入的内容。
- 读取文件后，无法再次读取读过的内容。
- seek(n)，其中n>=0，seek(0)表示文件指针移到文件头；n>0时，表示移动到文件头之后的位置，从任意位置读取内容时或从任意位置覆盖内容时需要这样做。
- seek(0,2)表示把文件指针移到文件尾，在追加新内容时需要这样做。

1. 文件基础知识

示例

把文件F7_1.txt中的“大”替换为“小”，再把“1”替换为“9”，最后在文件末尾增加“软件学院”。

```
#Exp7_12.py
#coding=GBK
f=open('F7_1.txt','r+')
f.seek(5)                #文件指针移到'大'的首字节上
f.write('小')            #用'小'覆盖'大'
f.seek(1)                #文件指针移到'1'上
f.write('9')             #用'9'覆盖'1'
f.seek(0,2)                          #文件指针移到文件尾
f.write('软件学院') #增加新内容
f.close()
```

1.5 文件的读取

4. 文件的关闭

① 关闭文件的原因如下：

- 将写缓存同步到磁盘。
- 操作系统每个进程打开文件的个数是有限的。
- 如果打开文件数到了系统限制，再打开文件就会失败。

② 最常用的方法就是，调用close()显式地关闭文件。

目录

2. 文件操作

2.1　常用的文件操作函数

一般而言，文件的基本操作都需要os模块和os.path模块。

2. 文件操作

2.2　文件的复制

复制文件有以下两种方式：

- 可以用read()与write()方法来实现
- 另外还可以用shutil模块实现文件的复制，该模块的copyfile()函数就可以实现文件的复制

2. 文件操作

示例

编写一个用来复制文件的函数。

```
#Exp7_16.py
#coding=GBK
def FileCopy(tar_File,res_File):  #定义1个函数以完成文件的复制
try:
    f=open(res_File,'rb')
    f2=open(tar_File,'wb')
except:
    print('打开文件异常！')
return -1
s=f.read()
f2.write(s)
f.close()
f2.close()
return 0
```

2. 文件操作

示例

用FileCopy()函数把文本文件F7_1.txt复制到文件F7_1_2.txt中，把二进制文件F7_9.dat复制到F7_9_2.dat中。

```
#Exp7_17.py
from Exp7_16 import import FileCopy
#导入文件Exp7_16.py的方法FileCopy
FileCopy('F7_9_2.dat','F7_9.dat')
#调用导入的FileCopy方法
FileCopy('F7_1_2.txt','F7_1.txt')
#调用导入的FileCopy方法
```

2. 文件操作

示例

用shutil模块实现7_17.py的功能。

```
#Exp7_18.py
import shutil
shutil.copyfile('F7_9_2.dat','F7_9.dat')
#复制文件F7_9.dat到F7_9_2.dat
shutil.copyfile('F7_1_2.txt','F7_1.txt')
#复制文件F7_1.txt到F7_1_2.txt
```

2.3 文件的删除

文件的删除，需要调用os模块的remove()函数实现，我们使用os.path模块的exists()函数来确保被删除文件存在。示例如下：

```
import os,os.path
filename='test1.txt'
file(filename,'w')
if os.path.exists(filename): #确认文件是否存在
    os.remove(filename) #如果存在则删除
else:
    print('%s does not exist!'%filename)
```

2. 文件操作

2.4　文件的重命名

使用os模块的rename()函数可实现对文件或者目录的重命名。

```
os.listdir(".")          #列出当前目录的所有文件
os.rename("hi.txt","hello.txt")        #重命名文件
```

2. 文件操作

示例

问题描述：若当前目录存在文件名为test1.txt的文件，将其重新命名为mytest1.txt，若mytest1.txt已存在，则给出是否需要继续更名的提示。若不要，则提示更名不成功，退出程序；若要，则再次输入更名信息，检测新名是否已经存在，不存在则执行更名操作，输出更名成功提示信息，若存在，则再次询问是否更名。

2. 文件操作

```
#Exp7_19.py
import os,os.path
filename='test1.txt'
rename='mytest1.txt'
file_list=os.listdir('.')
print(file_list)
if filename in file_list:  #判断需要重命名的文件是否存在
    while(rename in file_list):                #更名是否存在
        choice=input('有重命名，继续吗？（Y/N）：')
        if choice in ['Y','y']:
            rename=input('请重新输入更新文件名：')
        else:
        break
    else:                          #更名不存在，则进行更名
        os.rename(filename,rename)
        print('重命名成功')
else:
    print('需要更名的文件不存在！')
```

程序运行结果

```
['7-1.py', 'F7_1.txt', 'F7_14.dat',
'F7_1_2.txt', 'F7_2.txt', 'F7_4.dat', 'jp.py',
'mytest1.txt', 'ps.py', 'README.txt',
'test1.txt', 'test7-1.txt', 'test7-2.txt',
'Untitled 1', '__pycache__']
```

2.5 文件的比较

前面说明了文件内容的查找统计与内容的替换，这里将介绍如何利用difflib模块实现对序列或文件的比较。

2. 文件操作

hello.txt的内容为：helloworld
hi.txt的内容为：hihello

```
#Exp7_22.py
import difflib
import os
A=open('hello.txt','r')
B=open('hi.txt','r')
contextA=A.read()
contextB=B.read()
s=difflib.SequenceMatcher(lambda x:x=="",contextA,contextB)
result=s.get_opcodes()
for tag,i1,i2,j1,j2 in result:
    print("%s contextA[%d:%d]=%s contextB[%d:%d]=%s"%\
        (tag,i1,i2,contextA[i1:i2],j1,j2,contextB[j1:j2]))
```

程序运行结果

```
insert contextA[0:0]= contextB[0:3]=hi
equal contextA[0:5]=hello
contextB[3:8]=hello
delete contextA[5:11]= world
contextB[8:8]=
```

目录

3. 目录操作

3.1　目录的创建

（1）用mkdir(path)创建一个指定目录。

```
>>> import os
>>> os.listdir('f:/')
['$RECYCLE.BIN', '360Downloads', 'KuGou', 'SogouDownLoad',
   'System Volume Information', 'Youku Files']
>>> os.mkdir('f:/mynewdir')  #创建mynewdir目录
>>> os.listdir('f:/')
['$RECYCLE.BIN', '360Downloads', 'KuGou', 'mynewdir',
   'SogouDownLoad', 'System Volume Information', 'Youku
   Files']
```

3. 目录操作

3.1　目录的创建

（2）用makedirs(path1/path2...)创建多个目录。

```
>>> os.mkdir('./Newdir/subdir') #试图用mkdir创建两级目录：
    Newdir与下级目录subdir
Traceback(most recent call last):
   File "<stdin>", line 1. in <module>
FileNotFoundError: [WinError 3] 系统找不到指定的路径。：
    './Newdir/subdir'
>>> os.makedirs('./Newdir/subdir')  #用makedirs成功创建两级
    目录
```

3. 目录操作

3.2　目录的删除

删除目录的函数有以下两个。

- os.rmdir("dir")：只能删除空目录。
- shutil.rmtree("dir")：空目录、有内容的目录都可以删除。

3. 目录操作

3.2 目录的删除

（1）用rmdir(path)删除一个目录。

```
>>> import os
>>> os.listdir('f://') #检查F盘下的目录与文件信息
['$RECYCLE.BIN', '360Downloads', 'KuGou', 'mynewdir',
   'SogouDownLoad', 'System Volume Information', 'Youku
   Files']
>>> os.rmdir('f://mynewdir') #删除mynewdir目录
>>> os.listdir('f://')          #检查是否已经删除
['$RECYCLE.BIN', '360Downloads', 'KuGou', 'SogouDownLoad',
   'System Volume Information', 'Youku Files']
```

（2）用removedirs(path1/path2/...)删除多级目录。

```
>>> os.removedirs('./Newdir/subdir')
        #用removedirs成功删除两级目录
```

3. 目录操作

3.3 目录的遍历

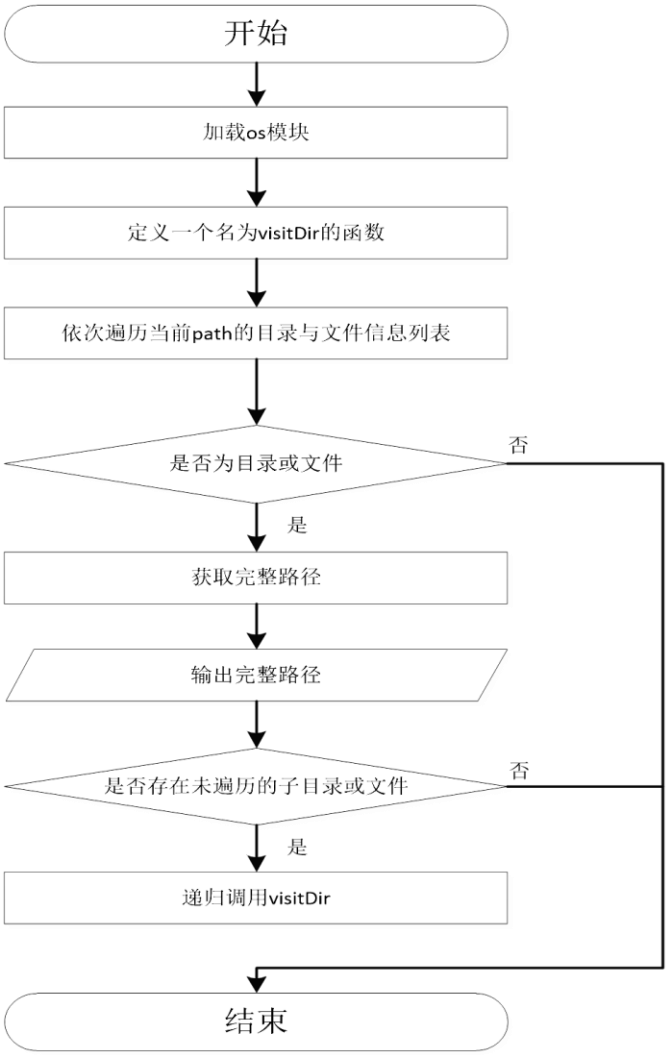
用listdir(path)函数可以查看指定路径下的目录及文件信息，如果我们希望查看指定路径下全部子目录的所有目录和文件信息，就需要进行目录的遍历，常用方法如下：

- 递归法
- os.walk()函数

3. 目录操作

1．递归法

分析：采用os.path.join函数获取文件或者目录的完整信息，并输出显示，然后判断该信息是否为目录，若是，则依据该目录进行递归，获取其下一级目录及文件的信息。具体的流程图如右。



3. 目录操作

2. os.walk()函数法

os.walk()函数将返回该路径下的所有文件及子目录信息元组，将该信息列表分成文件、目录逐行进行显示。程序具体的流程图如右所示。