《Python统计计算》

（2021年秋季学期）

翟祥
北京林业大学

E-mail: zhaixbh@126.com

编程基础

第7章 文件操作

- 为了长期保存数据以便重复使用、修改和共享，必须将数据以文件的形式存储到外部存储介质(如磁盘、U盘、光盘等)或云盘中。
- 管理信息系统是使用数据库来存储数据的，而数据库最终还是要以文件的形式存储到硬盘或其他存储介质上。
- 应用程序的配置信息往往也是使用文件来存储的。
- 图形、图像、音频、视频、可执行文件等等也都是以文件的形式存储在磁盘上的。
- 因此，文件操作在各类应用软件的开发中均占有重要的地位。

- 按文件中数据的组织形式把文件分为文本文件和二进制文件两类。

（1）文本文件

文本文件存储的是常规字符串，由若干文本行组成，通常每行以换行符'\n'结尾。常规字符串是指记事本或其他文本编辑器能正常显示、编辑并且人类能够直接阅读和理解的字符串，如英文字母、汉字、数字字符串。文本文件可以使用字处理软件如gedit、记事本进行编辑。

（2）二进制文件

二进制文件把对象内容以字节串(bytes)进行存储，无法用记事本或其他普通字处理软件直接进行编辑，通常也无法被人类直接阅读和理解，需要使用专门的软件进行解码后读取、显示、修改或执行。常见的如图形图像文件、音视频文件、可执行文件、资源文件、各种数据库文件、各类office文档等都属于二进制文件。

7.1 文件基本操作

文件对象名=open(文件名[, 打开方式[, 缓冲区]])

（1）文件名指定了被打开的文件名称。

（2）打开模式指定了打开文件后的处理方式，见下页表格。

（3）缓冲区指定了读写文件的缓存模式。0表示不缓存，1表示缓存，如大于1则表示缓冲区的大小。默认值是缓存模式。

（4）open( )函数返回1个文件对象，该对象可以对文件进行各种操作。

例如：　f1 = open('file1.txt', 'r')

　　f2 = open('file2.txt', 'w')

7.1 文件基本操作

文件打开方式

模式	说明
r	读模式（默认模式，可省略），如果文件不存在则抛出异常
w	写模式，如果文件已存在，先清空原有内容
x	写模式，创建新文件，如果文件已存在则抛出异常
a	追加模式，不覆盖文件中原有内容
b	二进制模式（可与其他模式组合使用）
t	文本模式（默认模式，可省略）
+	读、写模式（可与其他模式组合使用）

文件常用的打开方式

- r 以只读模式打开文件，缺省模式为r
- w 以只写模式打开文件，且先把文件内容清空
- a 以添加模式打开文件，写文件的时候总是写到文件末尾，用seek也无用。打开的文件也是不能读的
- r+ 以读写方式打开文件，文件可读可写，可写到文件的任何位置
- w+ 和r+不同的是，“w+”会先把文件内容清空
- a+ 和r+不同的是，“a+”只能写到文件末尾

7.1 文件基本操作

- 文件常用属性

属性	说明
closed	判断文件是否关闭，若文件被关闭，则返回True
mode	返回文件的打开模式
name	返回文件的名称

7.1 文件基本操作

- 文件对象常用方法

方法	说明
flush()	把缓冲区的内容写入文件，不关闭文件
close()	把缓冲区的内容写入文件，关闭文件，释放文件对象
read([size])	从文件中读取size个字节的内容作为结果返回，如果省略size则表示一次性读取所有内容
readline()	从文本文件中读取一行内容作为结果返回
readlines()	把文本文件中的每行作为字符串插入列表中，返回该列表
seek(offset[,whence])	把文件指针移动到新的位置，offset表示相对于whence的位置。whence为0表示从文件头开始计算，1表示从当前位置开始计算，2表示从文件尾开始计算，默认为0
tell()	返回当前文件指针的位置
truncate([size])	删除从当前指针位置到文件末尾的内容。如果指定了size，则不论指针在什么位置都只留下前size个字节，其余的删除
write(s)	把字符串s的内容写入文件
writelines(s)	把字符串列表写入文本文件，不添加换行符

7.2 文本文件基本操作

- 例1：向文本文件中写入内容。

```
f=open('sample.txt', 'a+')
s= '文本文件的读取方法\n文本文件的写入方法\n'
f.write(s)
f.close()
```

7.2 文本文件基本操作

- 更建议这样写：

```
s= '文本文件的读取方法\n文本文件的写入方法\n'
with open('sample.txt','a+') as f:
    f.write(s)
```

- 使用with自动关闭资源。
- 不论何种原因跳出with块，总能保证文件被正确关闭。

7.2 文本文件基本操作

- 例2：读取并显示文本文件的前5个字节。（Python 2）

```
f=open( 'sample.txt', 'r')
s=f.read(5)   #读取文件的前5个字节
f.close( )
print 's=',s
print  ‘字符串s的长度(字节个数)=', len(s)
```

7.2 文本文件基本操作

- 读取并显示文本文件的前5个字符。（Python 3）

```
f=open( 'sample.txt', 'r')
s=f.read(5)    #读取文件的前5个字符
f.close( )
print('s=',s)
print('字符串s的长度(字符个数)=', len(s))
```

7.2 文本文件基本操作

- 例3：读取并显示文本文件所有行。

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

7.2 文本文件基本操作

- 或者可以这样写：

```
f=open('F7_2.txt', 'r')
li=f.readlines()
for line in li:
    print(line,  end=' ')
f.close()
```

7.2 文本文件基本操作

- 例4：移动文件指针。

Python 2.x和Python 3.x对于seek()方法的理解和处理是一致的，即将文件指针定位到文件中指定字节的位置。但是由于对中文的支持程度不一样，可能会导致在Python 2.x和Python 3.x中的运行结果有所不同。例如下面的代码在Python 3.4.2中运行，当遇到无法解码的字符会抛出异常。

7.2 文本文件基本操作

```
>>> s = '中国山东烟台SDIBT'
>>> fp = open(r'D:\sample.txt', 'w')
>>> fp.write(s)
11
>>> fp.close()
>>> fp = open(r'D:\sample.txt', 'r')
>>> print(fp.read(3))
中国山
>>> fp.seek(2)
2
>>> print(fp.read(1))
国
```

```
>>> fp.seek(13)
13
>>> print(fp.read(1))
D
>>> fp.seek(3)
3
>>> print(fp.read(1))
```
出错信息

```
UnicodeDecodeError: 'gbk' codec can't decode byte 0xfa in position 0: illegal multibyte sequence
```

7.2 文本文件基本操作

- 而在Python 2.7.8中，则不抛出异常，而是输出乱码，例如下面的代码：

```
>>> s = '中国山东烟台SDIBT'
>>> fp = open(r'D:\sample.txt', 'w')
>>> fp.write(s)
>>> fp.close()
>>> fp = open(r'D:\sample.txt', 'r')
>>> print(fp.read(3))
Öй
>>> fp.seek(2)
>>> print(fp.read(3))
¤úÉ
>>> print(fp.read(2))
蕉
```

7.2 文本文件基本操作

- 例5：读取文本文件data.txt（文件中每行存放一个整数）中所有整数，将其按升序排序后再写入文本文件data_asc.txt中。

```
with open('data.txt', 'r') as fp:
    data = fp.readlines()
data = [int(line.strip()) for line in data]
data.sort()
data = [str(i)+'\n' for i in data]
with open('data_asc.txt', 'w') as fp:
    fp.writelines(data)
```

7.2 文本文件基本操作

- 例6：编写程序，保存为demo6.py，运行后生成文件demo6_new.py，其中的内容与demo6.py一致，但是在每行的行尾加上了行号。

```
filename = 'demo6.py'
with open(filename, 'r') as fp:
    lines = fp.readlines()
lines = [line.rstrip()+' '*(100-len(line))+'#'+str(index)+'\n' for index, line in enumerate(lines)]
with open(filename[:-3]+'_new.py', 'w') as fp:
    fp.writelines(lines)
```

7.3 二进制文件操作案例精选

- 数据库文件、图像文件、可执行文件、音视频文件、Office文档等等均属于二进制文件。对于二进制文件，不能使用记事本或其他文本编辑软件进行正常读写，也无法通过Python的文件对象直接读取和理解二进制文件的内容。必须正确理解二进制文件结构和序列化规则，才能准确地理解二进制文件内容并且设计正确的反序列化规则。
- 所谓序列化，简单地说就是把内存中的数据在不丢失其类型信息的情况下转成对象的二进制形式的过程，对象序列化后的形式经过正确的反序列化过程应该能够准确无误地恢复为原来的对象。
- Python中常用的序列化模块有struct、pickle、json、marshal和shelve，其中pickle有C语言实现的cPickle，速度约提高1000倍，应优先考虑使用。

7.3.1 使用pickle模块

- 例：写入二进制文件。

```
#7_8.py
import pickle
f=open('sample_pickle.dat', 'wb')
n=7
i=13000000
a=99.056
s='中国人民123abc'
lst=[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
tu=(-5, 10, 8)
coll={4, 5, 6}
dic={'a':'apple', 'b':'banana', 'g':'grape', 'o':'orange'}
try:
    pickle.dump(n, f)  #写入整数（用来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(s, f)
    pickle.dump(lst, 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( )
```

7.3.1 使用pickle模块

- 例：读取二进制文件。

```
#7_9.py
import pickle
f=open('sample_pickle.dat', 'rb')
n = pickle.load(f) #第一行的整数（刚才写入的表示后面数据个数）
i=0
while i<n:
    x = pickle.load(f)
    print(x)
    i=i+1
f.close( )
```

7.3.2 使用struct模块

- 例：使用struct模块写入二进制文件。

```
#7_10.py
import struct
n=1300000000
x=96.45
b=True
s='a1@中国'

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
# struct.pack(fmt, v1, v2, ...)，参数fmt是格式字符串，
# v1, v2, ...表示要转换的python值。

f=open('sample_struct.dat', 'wb')
f.write(sn)   #写入字节串
f.write(s.encode())   #encode将字符串转换成字节码
f.close( )
```

7.3.2 使用struct模块

- 例：使用struct模块读取二进制文件。

```
#7_11.py
import struct
f=open('sample_struct.dat', 'rb')
sn=f.read(9) #每个元素(整数、浮点数、布尔值)三个字节

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).decode()  #decode将字节码转换为字符串
#字符串中每个汉字3个字节，其它字符每个1个字节
f.close()
print('s=', s)
```

7.4 文件级操作

- 如果仅需要对文件内容进行读写，可以使用7.1节中介绍的文件对象；
- 如果需要处理文件路径，可以使用os.path模块中的对象和方法；
- 如果需要使用命令行读取文件内容可以使用fileinput模块；
- 创建临时文件和文件夹可以使用tempfile模块；
- 另外，Python 3.4开始的pathlib模块提供了大量用于表示和处理文件系统路径的类。

7.4.1 os与os.path模块

- os模块常用的文件处理函数

函数	使用说明
access(path, mode)	按照mode指定的权限访问文件
open(path, flags, mode=0o777, *, dir_fd=None)	按照mode指定的权限打开文件，默认权限为可读、可写、可执行
chmod(path, mode, *, dir_fd=None, follow_symlinks=True)	改变文件的访问权限
remove(path)	删除指定的文件
rename(src, dst)	重命名文件或目录
stat(path)	返回文件的所有属性
fstat(path)	返回打开的文件的所有属性
listdir(path)	返回path目录下的文件和目录列表
startfile(filepath [, operation])	使用关联的应用程序打开指定文件

7.4.1 os与os.path模块

- os.path模块常用的文件处理函数

函数名称	使用说明
abspath(path)	返回绝对路径
dirname(p)	返回目录的路径
exists(path)	判断文件是否存在
getatime(filename)	返回文件的最后访问时间
getctime(filename)	返回文件的创建时间
getmtime(filename)	返回文件的最后修改时间
getsize(filename)	返回文件的大小
isabs(path)、isdir(path)、isfile(path)	判断path是否为绝对路径、目录、文件
split(path)	对路径进行分割，以列表形式返回
splitext(path)	从路径中分割文件的扩展名
splitdrive(path)	从路径中分割驱动器的名称
walk(top,func,arg)	遍历目录

7.4.1 os与os.path模块

```
>>> import os
>>> import os.path
>>> os.path.exists('test1.txt')
False
>>> os.rename(‘c:\\test1.txt’,‘d:\\test2.txt’) #此时‘c:\\test1.txt’不存在，出错
>>> os.rename(‘c:\\dfg.txt’,‘d:\\test2.txt’) #os.rename可以实现文件的改名和移动
>>> os.path.exists('c:\\dfg.txt')
False
>>> os.path.exists('d:\\dfg.txt')
False
>>> os.path.exists('d:\\test2.txt')
True
>>> path='d:\\mypython_exp\\new_test.txt'
>>> os.path.dirname(path)
'd:\\mypython_exp'
>>> os.path.split(path)
('d:\\mypython_exp', 'new_test.txt')
>>> os.path.splitdrive(path)
('d:', '\\mypython_exp\\new_test.txt')
>>> os.path.splitext(path)
('d:\\mypython_exp\\new_test', '.txt')
```

7.4.1 os与os.path模块

- 列出当前目录下所有扩展名为pyc的文件
  - 其中os.getcwd()返回当前工作目录

```
>>> import os
>>> [fname for fname in os.listdir(os.getcwd()) if os.path.isfile(fname)
    and fname.endswith('.pyc')]
['consts.pyc', 'database_demo.pyc', 'nqueens.pyc']
```

7.4.1 os与os.path模块

- 将当前目录的所有扩展名为“html”的文件修改为扩展名为“htm”的文件：

```
import os
file_list=os.listdir(".")  #以列表形式返回当前路径下的文件和文件夹
for filename in file_list:
    pos=filename.rindex(".")
    if filename[pos+1:]=="html":
        newname=filename[:pos+1]+"htm"
        os.rename(filename,newname)
        print(filename+"更名为："+newname)
```

7.4.1 os与os.path模块

- 也可以改写为下面的简洁而等价的代码：

```
import os
file_list = [filename for filename in os.listdir(".") \
                    if filename.endswith('.html')]
for filename in file_list:
    newname = filename[:-4]+'htm'
    os.rename(filename, newname)
    print(filename+"更名为："+newname)
```

7.4.2 shutil模块

- 下面的代码使用shutil模块的copyfile()方法复制文件。

```
>>> import shutil
>>> shutil.copyfile('C:\\dir.txt', 'C:\\dir1.txt')
```

下面的代码将C:\Python34\Dlls文件夹以及该文件夹中所有文件压缩至D:\a.zip文件。

```
>>> shutil.make_archive('D:\\a', 'zip', 'C:\\Python34', 'Dlls')
'D:\\a.zip'
```

- 而下面的代码则将刚压缩得到的文件D:\a.zip解压缩至D:\a_unpack文件夹。

```
>>> shutil.unpack_archive('D:\\a.zip', 'D:\\a_unpack')
```

- 下面的代码使用shutil模块的方法删除刚刚解压缩得到的文件夹。

```
>>> shutil.rmtree('D:\\a_unpack')
```

7.5 目录操作

- os模块常用的目录操作函数

函数名称	使用说明
mkdir(path[,mode=0777])	创建目录
makedirs(path1/path2…,mode=511)	创建多级目录
rmdir(path)	删除目录
removedirs(path1/path2…)	删除多级目录
listdir(path)	返回指定目录下所有文件信息
getcwd()	返回当前工作目录
chdir(path)	把path设为当前工作目录
Walk(top,topdown=True,onrror=None)	遍历目录树

7.5 目录操作

```
>>> import os
>>> os.getcwd() #返回当前工作目录
'C:\\Python27'
>>> os.mkdir(os.getcwd()+'\\temp') #创建目录
>>> os.chdir(os.getcwd()+'\\temp') #改变当前工作目录
>>> os.getcwd()
'C:\\Python27\\temp'
>>> os.mkdir(os.getcwd()+'\\test')
>>> os.listdir('.')
['test']
>>> os.rmdir('test') #删除目录
>>> os.listdir('.')
[]
```

7.5 目录操作

- 递归遍历文件夹

```
import os

def visitDir(path):
    if not os.path.isdir(path):
        print('Error:"',path,'" is not a directory or does not exist.')
        return

    for lists in os.listdir(path):
        sub_path = os.path.join(path, lists)
        print(sub_path)
        if os.path.isdir(sub_path):
            visitDir(sub_path)


visitDir('E:\\test')
```

7.5 目录操作

使用os.walk函数遍历

```
import os
def visitDir2(path):
    if not os.path.isdir(path):
        print('Error:"',path,'" is not a directory or does not exist.')
        return

    #os.walk返回一个元组，包括3个元素：
    #所有路径名、所有目录列表与文件列表
    list_dirs = os.walk(path)

    for root, dirs, files in list_dirs:  #遍历该元组的目录和文件信息
        for d in dirs:
            print(os.path.join(root, d))   #获取完整路径
        for f in files:
            print(os.path.join(root, f))   #获取文件绝对路径

visitDir2('h:\\music')
```

7.5 目录操作

- 下面的代码可以用来删除当前文件夹以及所有子文件夹中特定类型的文件，其中要删除的文件类型可以在当前文件夹下的配置文件filetypes.txt中进行定义，每个文件类型的扩展名占一行。

```
from os.path import isdir, join, splitext
from os import remove, listdir, getcwd
filetypes = []
def delCertainFiles(directory):
    for filename in listdir(directory):
        temp = join(directory, filename)
        if isdir(temp):
            delCertainFiles(temp)
        elif splitext(temp)[1] in filetypes:
            # check file extension name
            remove(temp)
            print(temp, ' deleted....')
def readFileTypes():
    global filetypes
    with open('filetypes.txt', 'r') as fp:
        filetypes = fp.readlines()
    filetypes = [ext.strip() for ext in filetypes]
def main():
    readFileTypes()
    print(filetypes)
    delCertainFiles(getcwd())

main()
```

7.6 高级话题

- 计算文本文件中最长行的长度

方法一：

```
f=open('d:\\test.txt','r')
allLineLens=[len(line.strip()) for line in f]
f.close()
longest=max(allLineLens)
print longest
```

方法二：

```
f=open('d:\\test.txt','r')
longest=max(len(line.strip()) for line in f)
f.close()
print longest
```

7.6 高级话题

计算MD5值(Python 2.7.x版本)

```
>>> import hashlib
>>> md5value=hashlib.md5() #创建hash对象
>>> md5value.update(‘12345’) #以字符串为参数，更新哈希对象
>>> md5value=md5value.hexdigest() #返回十六进制数字字符串
>>> print md5value
827ccb0eea8a706c4c34a16891f84e7b

>>> import md5
>>> md5value=md5.md5()
>>> md5value.update('12345')
>>> md5value=md5value.hexdigest()
>>> print md5value
827ccb0eea8a706c4c34a16891f84e7b
```

如果在Python 3.x环境运行，需要使用encode()方法对unicode字符串进行编码，如'12345'.encode()。

7.6 高级话题

- 对上面的代码稍加完善，即可实现自己的MD5计算器，

例如：

```
import hashlib
import os
import sys
fileName = sys.argv[1]
if os.path.isfile(fileName):
    with open(fileName, 'r') as fp:
        lines = fp.readlines()
    data = ''.join(lines)
    print(hashlib.md5(data).hexdigest())
```

7.6 高级话题

- 也可以使用ssdeep工具来计算文件的模糊哈希值，或者编写Python程序调用ssdeep提供的API函数来计算文件的模糊哈希值，模糊哈希值可以用来比较两个文件的相似百分比。

```
>>> from ssdeep import ssdeep
>>> s = ssdeep()
>>> print s.hash_file(filename)
```

- 对于某些恶意软件来说，可能会对自身进行加壳或加密，真正运行时再进行脱壳或解密，这样一来，会使得磁盘文件的哈希值和内存中脱壳或解密后进程的哈希值相差很大。因此，根据磁盘文件和其相应的进程之间模糊哈希值的相似度可以判断该文件是否包含自修改代码，并以此来判断其为恶意软件的可能性。

7.6 高级话题

- 判断一个文件是否为GIF图像文件

```
def is_gif(fname):
    f=open(fname,'r')
    first4=tuple(f.read(4)) #读取文件内容的开头四个字符
    f.close()
    return first4==('G','I','F','8')
>>> is_gif('c:\\test.gif')
True
>>> is_gif('c:\\dir.txt')
False
```

7.6 高级话题

- 使用xlwt写入Excel文件

```
#xlswrite.xls
from xlwt import *
book = Workbook()
sheet1 = book.add_sheet("First")
al=Alignment()
al.horz=Alignment.HORZ_CENTER
al.vert=Alignment.VERT_CENTER
borders=Borders()
borders.bottom=Borders.THICK
style=XFStyle()
style.alignment=al
style.borders=borders
row0=sheet1.row(0)
row0.write(0,'test',style=style)
book.save(r'd:\test.xls')
```

7.6 高级话题

- 使用xlrd读取Excel文件

```
>>> import xlrd
>>> book = xlrd.open_workbook(r'd:\test.xls')
>>> sheet1 = book.sheet_by_name('First')
>>> row0 = sheet1.row(0)
>>> print(row0[0])
text:u'test'
>>> print(row0[0].value)
test
```

7.6 高级话题

- 使用Pywin32操作Excel文件

```
#打开EXCEL
xlApp = win32com.client.Dispatch('Excel.Application')

xlBook = xlApp.Workbooks.Open('D:\\1.xls')
xlSht = xlBook.Worksheets('sheet1')
aaa = xlSht.Cells(1,2).Value
xlSht.Cells(2,3).Value = aaa
xlBook.Close(SaveChanges=1)
del xlApp
```

pywin32_excel.py

7.6 高级话题

- 使用扩展库openpyxl读写Excel 2007及更高版本的Excel文件。

```
import openpyxl
from openpyxl import Workbook
fn = r'f:\test.xlsx'                          #文件名
wb = Workbook()                              #创建工作簿
ws = wb.create_sheet(title='你好，世界')       #创建工作表
ws['A1'] = '这是第一个单元格'                  #单元格赋值
ws['B1'] = 3.1415926
wb.save(fn)                                  #保存Excel文件
wb = openpyxl.load_workbook(fn)              #打开已有的Excel文件
ws = wb.worksheets[1]                        #打开指定索引的工作表
print(ws['A1'].value)                        #读取并输出指定单元格的值
ws.append([1,2,3,4,5])                       #添加一行数据
ws.merge_cells('F2:F3')                      #合并单元格
ws['F2'] = "=sum(A2:E2)"                     #写入公式
for r in range(10,15):
    for c in range(3,8):
        _ = ws.cell(row=r, column=c, value=r*c)   #写入单元格数据
wb.save(fn)
```

7.6 高级话题

- 编写程序，进行文件夹增量备份。

程序功能与用法：指定源文件夹与目标文件夹，自动检测自上次备份以来源文件夹中内容的改变，包括修改的文件、新建的文件、新建的文件夹等等，自动复制新增或修改过的文件到目标文件夹中，自上次备份以来没有修改过的文件将被忽略而不复制，从而实现增量备份。

7.6 高级话题

```
import os
import filecmp
import shutil
import sys

def usage():
    print('scrDir and dstDir must be existing absolute path of certain directory')
    print('For example:{0} c:\\olddir c:\\newdir'.format(sys.argv[0]))
    sys.exit(0)
```

7.6 高级话题

```
def autoBackup(scrDir, dstDir):
    if ((not os.path.isdir(scrDir)) or (not os.path.isdir(dstDir)) or
      (os.path.abspath(scrDir)!=scrDir) or (os.path.abspath(dstDir)!=dstDir)):
        usage()
    for item in os.listdir(scrDir):
        scrItem = os.path.join(scrDir, item)
        dstItem = scrItem.replace(scrDir,dstDir)
        if os.path.isdir(scrItem):
            #创建新增的文件夹，保证目标文件夹的结构与原始文件夹一致
            if not os.path.exists(dstItem):
                os.makedirs(dstItem)
                print('make directory'+dstItem)
            autoBackup(scrItem, dstItem)
        elif os.path.isfile(scrItem):
            #只复制新增或修改过的文件
            if ((not os.path.exists(dstItem)) or
               (not filecmp.cmp(scrItem, dstItem, shallow=False))):
                shutil.copyfile(scrItem, dstItem)
                print('file:'+scrItem+'==>'+dstItem)
```

7.6 高级话题

```
if __name__=='__main__':
    if len(sys.argv)!=3:
        usage()
    scrDir, dstDir= sys.argv[1], sys.argv[2]
    autoBackup(scrDir, dstDir)
```

7.6 高级话题

- 编写程序，统计指定文件夹大小以及文件和子文件夹数量。

```
import os

totalSize = 0
fileNum = 0
dirNum = 0

def visitDir(path):
    global totalSize
    global fileNum
    global dirNum
    for lists in os.listdir(path):
        sub_path = os.path.join(path, lists)
        if os.path.isfile(sub_path):
            fileNum = fileNum+1                    #统计文件数量
            totalSize = totalSize+os.path.getsize(sub_path)  #统计文件总大小
        elif os.path.isdir(sub_path):
            dirNum = dirNum+1                      #统计文件夹数量
            visitDir(sub_path)                     #递归遍历子文件夹
```

7.6 高级话题

```
def main(path):
    if not os.path.isdir(path):
        print('Error:"', path, '" is not a directory or does not exist.')
        return
    visitDir(path)

def sizeConvert(size):                                  #单位换算
    K, M, G = 1024, 1024**2, 1024**3
    if size >= G:
        return str(size/G)+'G Bytes'
    elif size >= M:
        return str(size/M)+'M Bytes'
    elif size >= K:
        return str(size/K)+'K Bytes'
    else:
        return str(size)+'Bytes'
```

7.6 高级话题

```
def output(path):
    print('The total size of '+path+'
is:'+sizeConvert(totalSize)+'('+str(totalSize)+' Bytes)')
    print('The total number of files in '+path+' is:',fileNum)
    print('The total number of directories in '+path+' is:',dirNum)

if __name__=='__main__':
    path = r'd:\idapro6.5plus'
    main(path)
    output(path)
```

7.6 高级话题

- 编写程序，递归删除指定文件夹中指定类型的文件。

```
from os.path import isdir, join, splitext
from os import remove, listdir
import sys
filetypes = ['.tmp', '.log', '.obj', '.txt']        #指定要删除的文件类型
def delCertainFiles(directory):
    if not isdir(directory):
        return
    for filename in listdir(directory):
        temp = join(directory, filename)
        if isdir(temp):
            delCertainFiles(temp)
        elif splitext(temp)[1] in filetypes:  #检查文件类型
            remove(temp)
            print(temp, ' deleted....')
def main():
    directory = r'E:\new'
    #directory = sys.argv[1]
    delCertainFiles(directory)
main()
```

7.6 高级话题

- 如果文件夹中有带特殊属性的文件或子文件夹，上面的代码可能会无法删除带特殊属性的文件，利用Python扩展库pywin32可以解决这一问题。

```
import win32con
import win32api
import os
From win32con import FILE_ATTRIBUTE_NORMAL

def del_dir(path):
    for file in os.listdir(path):
        file_or_dir = os.path.join(path,file)
        if os.path.isdir(file_or_dir) and not os.path.islink(file_or_dir):
            del_dir(file_or_dir)        #递归删除子文件夹及其文件
        else:
            try:
                os.remove(file_or_dir) #尝试删除该文件，
            except:                    #无法删除，很可能是文件拥有特殊属性
                win32api.SetFileAttributes(file_or_dir, FILE_ATTRIBUTE_NORMAL)
                os.remove(file_or_dir) #修改文件属性，设置为普通文件，再次删除
    os.rmdir(path) #delete the directory here

del_dir("E:\\old")
```

7.6 高级话题

- Python标准库zipfile提供了对zip和apk文件的访问。

```
>>> import zipfile
>>> fp = zipfile.ZipFile(r'D:\Jakstab-0.8.3.zip')
>>> for f in fp.namelist():
        print(f)
>>> fp.close()
```

- Python扩展库rarfile（可通过pip工具进行安装）提供了对rar文件的访问。

```
>>> import rarfile
>>> r = rarfile.RarFile(r'D:\asp网站.rar')
>>> for f in r.namelist():
        print(f)
>>> r.close()
```

谢谢Q/A