文件和素材

打开文件

open函数

open(name[, mode[, buffering]])

使用一个文件名作为唯一的强制参数，返回一个文件对象

模式(mode)和缓冲(buffering)都是可选的

文件模式

若open函数只带一个文件名参数，可以获得能读取文件内容的文件对象；

若要向文件内写入内容，必须提供一个模式参数来显式声明；

明确地指定读模式和什么模式都不用效果一样；

使用写模式可以向文件写入内容

‘+’参数：可以用到其它任何模式中，指明读和写都是允许的；

‘r’：读模式

‘w’；写模式

‘a’：追加模式

‘b’：二进制模式(可添加到其它模式中)

‘+’：读/写模式(可添加到其它模式中)

‘b’模式改变处理文件的方法；

python假定处理的是文本文件(包含字符)

若处理其他类型的文件(二进制文件)，如声音剪辑或图像，在模式参数中增加’b’；’rb’：读取一个二进制文件

标准化换行符

python、unix：\n表示结束一行并另起一行；

windows：\r\n

通过在模式参数中使用U参数能够在打开文件时使用通用的换行符支持模式，在这种模式下，所有的换行符/字符串(\r\n,\r或者\n)都被转换成\n，不用考虑到运行的平台

缓冲

参数是0(或False)，I/O(输入/输出)是无缓冲的(所有读写操作都直接针对硬盘)；

参数是1(或True)，I/O(输入/输出)是有缓冲的(python用内存来代替硬盘，程序更快，只有使用flush或close时才会更新硬盘上的数据)；大于1的数字：缓冲区的大小(单位：字节)；-1(或任何负数)：使用默认的缓冲区大小

文件对象，流

类文件对象：支持一些文件的方法的对象，如file方法、read方法、write方法

urllib.urlopen返回的对象

sys模块，3种流，文件(类文件对象)：大部分文件对象可用的操作它们也可以使用

数据输入的标准源：sys.stdin，当程序从标准输入读取数据时，可以通过输入或使用管道把它和其它程序的标准输出链接起来提供文本

要打印的文本保存在sys.stout内，input和raw\_input函数的提示文字也是写入在sys.stout中，写入sys.stout的数据一般出现在屏幕上，但也能用管道连接到其它程序的标准输入

错误信息(如栈追踪)被写入sys.sterr

读/写

文件(或流)：提供或接受数据，若有一个名为f的类文件对象，可用f.write方法和f.read方法(以字符串形式)写入和读取数据

每次调用f.write(string)时，参数string会被追加到文件中已存在部分的后面

完成对一个文件的操作时，调用close

读取：告诉流要读多少字符(字节)

管式输出

使用管道可以在一个命令后面续写其它的多个命令

管道符号(|)：将一个命令的标准输出和下一个命令的标准输入连在一起

读写行

关闭文件

close方法关闭文件

一个文件对象在退出程序后自动关闭

写入过的文件总应该关闭，因为python可能会缓存(出于效率的考虑把数据临时存储在某处)写入的数据，若程序因为某些原因崩溃了，数据不会被写入文件；

要在使用完文件后关闭；

想确保文件被关闭了，使用try/finally语句，并在finally子句中调用close方法

#open your file here

try:

#write data to your file

finally:

file.close()

with语句

可以打开文件且将其赋值到变量上，之后将数据写入语句体中的文件

文件在语句结束后会被自动关闭，即使由于异常引起的结束也是如此

with open(‘somefile.txt’) as somefile:

do\_something(somefile)

数据可能被缓存了(在内存中临时性存储)，直到关闭文件才会被写入文件；

若想要继续使用文件(不关闭文件)，又想将磁盘上的文件进行更新，调用文件对象的flush方法(N.B. flush方法不允许其他程序使用该文件的同时访问文件)

with语句，允许使用上下文管理器，是一种支持 \_\_enter\_\_ 和 \_\_exit\_\_ 方法的对象

\_\_enter\_\_方法不带参数，它在进入with语句时被调用，返回值绑定在as关键字之后的变量

\_\_exit\_\_方法带有3个参数：异常类型、异常对象和异常回溯；在离开方法(带有通过参数提供的，可引发的异常)时，这个函数被调用；若\_\_exit\_\_返回false，所有的异常都不会被处理

文件可被用作上下文管理器；\_\_enter\_\_方法返回文件对象本身；\_\_exit\_\_方法关闭文件；

对文件内容进行迭代

按字节处理

在while循环中使用read方法

f = open('data.dat', 'r')

char = f.read(1)

while char:

print char

char = f.read(1)

f.close()

当到达文件的末尾时，read方法返回一个空字符串，但在那之前返回的字符串会包含一个字符(布尔值为真)，若char为真，表示没有到文件末尾

按行操作

readline方法

f = open(finename)

while True:

line = f.readline()

if not line: break

f.close()

读取所有文件内容

若文件不是很大，使用不带参数的read方法一次读取整个文件(把整个文件当作一个字符串)，或readlines方法(把文件读入一个字符串列表，在列表中每个字符串就是一行)

使用fileinput

xreadlines

文件迭代器