Python基础教程

第10章 文件和输入输出

10.1 文件对象

Python中的文件对象，主要对磁盘文件或者其他类型的抽象层面上的“文件”进行读取和写入，还有其他的操作。

文件只是连续的字节序列。

数据的传输经常会用到字节流，无论字节流是由单个字节还是大块数据组成。

10.2 文件内建函数 open( )

内建函数 **open( )** 提供了初始化输入/输出（**I/O**）操作通用接口。

open( ) 内建函数成功打开文件后会返回一个文件对象，否则会引发一个错误。

open(file, mode = 'r', buffering = -1, encoding = None, ...)

* **file**：必需的，字符串形式的文件，包括绝对或相对路径。

* **mode**：文件的打开模式。

* **encoding**：编码方式，在文本模式中使用，中文**Windows**平台的默认值为**'gbk'**。

常见文件操作模式

- **r：** 只读模式（默认）。
- **w：** 只写模式。

【不可读；不存在则创建；存在则删除原内容】

- **a：** 追加写模式。

【不可读； 不存在则创建；存在则只追加内容】

- **b：** 二进制模式。

【操作原始数据，字节字符串，默认为文本模式t】

- **“r+”**（读写方式打开）。
- **“w+”**（以写的方式打开并且还可以读）。
- **“a+”**（以追加的方式打开且还可以读）。
- **rb**（以二进制读模式打开 ）。
- **wb**（以二进制写模式打开） **(参见 w)**
- **ab**（以二进制追加模式打开） **(参见 a)**

```
>>> f1 = open("test1.txt")
>>> data = f1.read( )
>>> f1.close( )
>>> print(data)
hello 中国
>>> f1 = open(r"E:\data\test1.txt",'rb')
>>> data = f1.read()
>>> f1.close( )
>>> print(data)
b'hello \xd6\xd0\xb9\xfa'
```

- 问题：通过**open**并**read**的方式读文件会不会有问题？
- 请用记事本生成一个文档，并采用**Unicode**格式保存，然后读。
- 如何查看操作文件的编码方式？
- 文件对象的**encoding**属性。

10.2.2 通用换行符支持(UNS)

在不同的系统平台上，换行符是不同的，例如**Unix**下是**\n**，而**Windows**下是**\r\n**。

Python3中默认使用了通用换行符支持（**Universal NEWLINE Support**），会使用“**\n**”作为通用换行符，从而屏蔽了不同平台下的换行符差异。

可以通过**open**函数的**newline**参数进行修改。

10.3 文件内建方法

open( ) 成功执行并返回一个文件对象后，可以对该文件进行后续操作。

文件操作方法可以分为四类：

* 输入
* 输出
* 文件内移动
* 杂项操作。

10.3.1 输入

- **read( )**：

读取文件中的字符（不是字节）到字符串中，最多读取给定数目**size**个字符。

如果没有给定**size**或者**size**为负数，文件将被读取直至末尾。这个方法不推荐使用。

10.3.1 输入

- **readline( )：**
  - 读取文件中**size**个字符，返回一个字符串。
  - 如果没有给定**size**或者**size**为负数则返回一行（包括行结束符）。

10.3.1 输入

- **readlines( )：**
  - 读取剩余的所有的行（文件指针不一定在开始位置）并将其以一个字符串列表形式返回。
  - 此方法一次性读取文件所有的内容至内存中，适用于小型文件。

10.3.2 输出

- **write( )**：
  - 和 **read( )** 及 **readline( )** 相反，它把含有文本或字节字符串的字符串写入到文件中。

10.3.2 输出

- **writelines( )**：
  - 和 **readlines( )** 相反，它接收一个字符串列表作为参数并将其写入到文件中。
  - 每个字符串的行结束符不会被自动写入，如果需要的话，必须在调用**writelines( )** 前给每行加上行结束符。

10.3.3 文件内移动

seek(offset[,whence]) 方法类似于**C**中的**fseek**。

* **offset**为偏移量。

* **whence**代表相对位置，是一个可选参数，位置的默认值为**0**（文件开头），**1**代表当前位置，**2**代表文件末尾。

tell( ) 方法是对 **seek( )** 的补充，这个方法会告诉你当前文件指针在文件中的位置（从文件起始开始算起，单位为字符）。

10.3.4 文件迭代

迭代文件对象：*for eachLine in f:*

…

由于引入了迭代器和文件迭代，用户不必调用**read*( )** 方法就可以在**for**循环中迭代文件的每一行。

另外，可以使用迭代器的**next**方法，**file.next( )** 可以用来读取文件的下一行。

采用迭代方式读取文件更为高效。

10.3.5 其他

* close( )：通过关闭文件来结束对文件的访问。

* fileno( )：返回打开文件的文件描述符，这是一个整形，可以用于os模块的一些底层操作。

* flush( )：把输出缓冲区内的数据立即写入文件，调用close时会自动调用这个方法。

10.3.6 文件方法-open()实例

```
filename = input( ‘Enter file name: ’ )
f = open(filename)
allLines = f.readlines( )
f.close( )
for line in allLines:
    print(line)
```

这个例子在读完所有行以后才开始向屏幕输出数据，当文件很大时，这个方法并不好。

open()实例

更有效的办法是使用迭代器，每次只读取和显示一行：

```
filename = input( ‘Enter file name: ’ )
f = open(filename)
for line in f:
    print(line)
f.close( )
```

10.3.6 文件方法-write()实例

```
filename = input('Enter file name for write:    ')
f = open(filename,'w')
while True:
    line = input("Enter a line('.' to quit): ")
    if line != '.':
        f.write('%s\n' % line)
    else:
        break
f.close()
```

由于 input( ) 不会保留用户输入的换行符，调用 write( ) 方法时必须加上换行符。在键盘上很难输入一个EOF字符，所以程序使用(.)作为文件结束的标志，当用户输入句点以后会自动结束输入并关闭文件。

10.3.6 文件方法-seek()和tell()

```
>>> f = open('/tmp/x', 'w+')
>>> f.tell()
0
>>> f.write('test line 1\n')          # 加入一个长为 12 的字符串 [0-11]
>>> f.tell()
12
>>> f.write('test line 2\n')          # 加入一个长为 12 的字符串 [12-23]
>>> f.tell()                          # 告诉我们当前的位置
24
>>> f.seek(-12, 1)                    # 向后移 12 个字节
>>> f.tell()                          # 到了第二行的开头
12
>>> f.readline()
'test line 2\012'
>>> f.seek(0, 0)                      # 回到最开始
>>> f.readline()
'test line 1\012'
>>> f.tell()                          # 又回到了第二行
12
>>> f.readline()
'test line 2\012'
>>> f.tell()                          # 又到了结尾
>>> f.close()                         # 关闭文件
```

小实验

请检测用户给定的文件里是否含有敏感词，例如：“法轮功”。

10.4 文件内建属性

文件内建属性保存了文件对象相关的附加数据。

表 9.4 文件对象的属性

文件对象的属性	描　述
file.closed	表示文件已经被关闭，否则为 False
file.encoding[a]	文件所使用的编码——当 Unicode 字符串被写入数据时，它们将自动使用 file.encoding 转换为字节字符串；若 file.encoding 为 None 时使用系统默认编码
file.mode	Access 文件打开时使用的访问模式
file.name	文件名
file.newlines[a]	未读取到行分隔符时为 None，只有一种行分隔符时为一个字符串，当文件有多种类型的行结束符时，则为一个包含所有当前所遇到的行结束符的列表
file.softspace	为 0 表示在输出一数据后，要加上一个空格符，1 表示不加。这个属性一般程序员用不着，由程序内部使用

a. 2.3 版本中新增。

10.5 标准文件

Python中, 只要程序一执行，便可以访问三个标准文件：标准输入（一般是键盘），标准输出（到显示器的缓冲输出）和标准错误（到屏幕的非缓冲输出）。

“缓冲”和“非缓冲”是指 open( ) 函数的第三个参数， 沿用的是 C 语言中的命名, 分别为stdin，stdout和stderr。

可以通过 sys 模块来访问这些文件的句柄。导入sys模块以后，就可以使用sys.stdin，sys.stdout和sys.stderr访问。

sys.*是文件，所以必须处理好换行符。而 print 语句会自动在要输出的字符串后加上换行符。

10.6 命令行参数

sys模块通过sys.argv属性提供了对命令行参数的访问，sys.argv是命令行参数的列表，len(sys.argv)是命令行参数的个数（也就是argc）。

```
#test.py
import sys
print( ‘You entered %d arguments’ % len(sys.argv))
print( ‘They were:’ , str(sys.argv))
```

10.7 文件系统

对文件系统的访问大多通过Python的os模块来实现。该模块是Python访问操作系统功能的主要接口。这些操作主要包括遍历目录树，删除/重命名文件，管理文件访问权限，系统进程管理等。

函　数	描　述
文件处理	
mkfifo()/mknod()a	创建命名管道/创建文件系统节点
remove()/unlink()	删除文件
rename()/renames()	重命名文件
*stat()	返回文件信息
symlink()	创建符号链接
utime()	更新时间戳
tmpfile()	创建并打开（‘w+b’）一个新的临时文件
walk()	生成一个目录树下的所有文件名

函　数	描　述
目录/文件夹	
chdir()/fchdir()	改变当前工作目录/通过一个文件描述符改变当前工作目录
chroot()	改变当前进程的根目录
listdir()	列出指定目录文件
getcwd()/getcwdu()	返回当前工作目录/功能相同，但 返回一个Unicode对象
mkdir()/mkdirs()	创建目录/创建多层目录
rmdir()/removedirs()	删除目录/删除多层目录
访问/权限	
access()	检验权限模式
chmod()	改变权限模式
chown()/lchown()	改变owner和group ID/功能相同，但不哈跟踪链接
umask()	设置默认权限模式
文件描述符操作	
open()	底层的操作系统open（对于文件，使用标准的内建open()函数）
read()/write()	根据文件描述符读取/写入数据
dup()/dup2()	复制文件描述符号/功能相同，但是是复制到另一个文件描述符
设备号	
makedev()	从major和minor设备号创建一个原始设备号
major()/minor()	从原始设备号获得major/minor设备号

10.7 文件系统

另一个模块**os.path**可以完成一些针对路径名的操作。

os.path模块中路径访问函数：

函　数	描　述
分隔	
basename()	去掉目录路径，返回文件名
dirname()	去掉文件名，返回目录路径
join()	将分离的各部分组合成一个路径名
split()	返回（dirname(),basename()）元组
splitdrive()	返回（drivename,pathname）元组
splitext()	返回（filename,extension）元组
信息	
getatime()	返回最近访问时间
getctime()	返回文件创建时间
getmtime()	返回最近文件修改时间
getsize()	返回文件大小（以字节为单位）

os.path模块中路径访问函数：

函　数	描　述
查询	
exists()	指定路径（文件或目录）是否存在
isabs()	指定路径是否为绝对路径
isdir()	指定路径是否存在且为一个目录
isfile()	指定路径是否存在且为一个文件
islink()	指定路径是否存在且为一个符号链接
ismount()	指定路径是否存在且为一个挂载点
samefile()	两个路径名是否指向同一个文件

10.8 永久存储模块

永久性存储模块可以把用户的数据归档保存以供以后使用，而又比完整的数据库管理系统简单，实用性很高。

大部分永久性存储模块是用来存储字符串数据的，但是也有方法来归档**Python**对象。

10.8.1 pickle和marshal模块

pickle和**marshal**模块可以用来转换并储存**Python**对象：

将比基本类型复杂的对象转换成一个二进制数据集合，这样就可以把数据集合保存起来，或通过网络发送，然后再重新把数据集合恢复原来的对象格式。

这个过程也称数据的序列化。

10.8.1 pickle和marshal模块

pickle和**marshal**模块的区别在于：

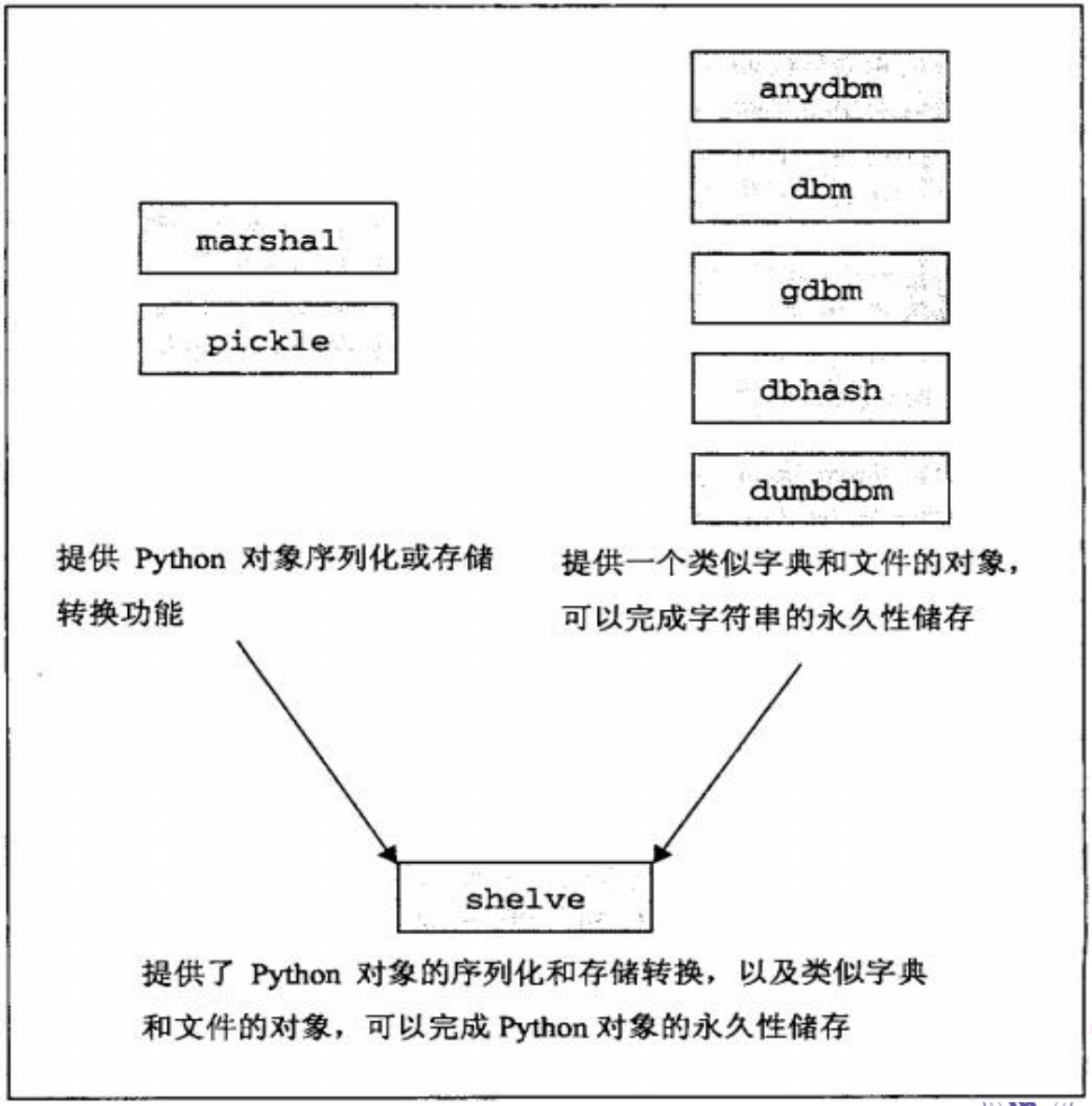
marshal只能处理简单的**Python**对象（数字、序列、映射以及代码对象）；

pickle还可以处理递归对象，被不同地方多次引用的对象，以及用户定义的类和实例。

10.8.3 shelve模块

shelve模块允许使用**anydbm**模块寻找合适的**DBM**模块，然后使用**cPickle**完成对存储转换的过程。

shelve模块原理：



10.9 相关模块

* **fileinput**模块：遍历一组输入文件，每次读取它们内容的一行，类似**Perl**语言中的不带参数的“<>”操作符。如果没有明确给定文件名，则默认从命令行读取文件名。

* **glob**模块和**fnmatch**模块：提供**Unix shell**样式文件名的模式匹配，例如使用星号(*)通配符代表任意字符串，用问号(?)匹配任意单个字符。

10.9 相关模块

* gzip模块和zlib模块：提供对zlib压缩库直接访问的接口。

* zipfile模块：修改和读取zip归档文件。

* shutil模块：提供高级的文件访问功能，包括复制文件、复制文件的访问权限等。

* tempfile模块：用于生产临时文件（名）。

理解文件系统
熟练掌握文件操作