python读写文件，和设置文件的字符编码比如utf-8

python 文件 更多 2

一. python打开文件代码如下：

```
f = open("d:\test.txt", "w")
```

说明：
第一个参数是文件名称，包括路径；
第二个参数是打开的模式mode

> 'r'：只读（缺省。如果文件不存在，则抛出错误）
> 'w'：只写（如果文件不存在，则自动创建文件）
> 'a'：附加到文件末尾
> 'r+'：读写

如果需要以二进制方式打开文件，需要在mode后面加上字符"b"，比如"rb""wb"等

二、python读取文件内容`f.read(size)`

> 参数size表示读取的数量，可以省略。如果省略size参数，则表示读取文件所有内容。

f.readline()读取文件一行的内容 f.readlines()读取所有的行到数组里面[line1,line2,...lineN]。

在避免将所有文件内容加载到内存中，这种方法常常使用，便于提高效率。

三、python写入文件f.write(string)

将一个字符串写入文件，如果写入结束，必须在字符串后面加上"\n"，然后f.close()关闭文件

四、文件中的内容定位

f.read()读取之后，文件指针到达文件的末尾，如果再来一次f.read()将会发现读取的是空内容，如果想再次读取全部内容，必须将定位指针移动到文件开始：

```
f.seek(0)
```

这个函数的格式如下（单位是bytes）：`f.seek(offset, from_what)` from_what表示开始读取的位置，offset表示从from_what再移动一定量的距离，比如f.seek(10, 3)表示定位到第三个字符并再后移10个字符。

from_what值为0时表示文件的开始，它也可以省略，缺省是0即文件开头。下面给出一个完整的例子：

```
f = open('/tmp/workfile', 'r+')
f.write('0123456789abcdef')
f.seek(5)     # Go to the 6th byte in the file
f.read(1)
f.seek (-3, 2) # Go to the 3rd byte before the end
f.read(1)
```

五、关闭文件释放资源文件操作完毕，一定要记得关闭文件`f.close()`，可以释放资源供其他程序使

只是ASCII或者gbk编码格式的的文件读写，比较简单，读写如下：

```
# coding=gbk

f = open('c:/intimate.txt','r') # r 指示文件打开模式，即只读
s1 = f.read()
s2 = f.readline()
s3 = f.readlines() #读出所有内容

f.close()

f = open('c:/intimate.txt','w') # w 写文件
11 f.write(s1)
12 f.writelines(s2) # 没有writeline
13 f.close()
```

六. `f.writelines`不会输出换行符。

python unicode文件读写：

```
# coding=gbk
import codecs

f = codecs.open('c:/intimate.txt','a','utf-8')
f.write(u'中文')
s = '中文'
f.write(s.decode('gbk'))
f.close()

f = codecs.open('c:/intimate.txt','r','utf-8')
s = f.readlines()
f.close()
for line in s:
    print line.encode('gbk')
```

python代码文件的编码

py文件默认是ASCII编码，中文在显示时会做一个ASCII到系统默认编码的转换，这时就会出错：SyntaxError: Non-ASCII character。需要在代码文件的第一行或第二行添加编码指示：

1. `# coding=utf-8 ##以utf-8编码储存中文字符`
2. print '中文'像上面那样直接输入的字符串是按照代码文件的编码来处理的，如果用unicode编码，有以下2种方式：
    1. s1 = u'中文' #u表示用unicode编码方式储存信息
    2. s2 = unicode('中文','gbk')

unicode是一个内置函数，第二个参数指示源字符串的编码格式。

decode是任何字符串具有的方法，将字符串转换成unicode格式，参数指示源字符串的编码格式。

encode也是任何字符串具有的方法，将字符串转换成参数指定的格式。

python字符串的编码

用 u'汉字' 构造出来的是unicode类型，不用的话构造出来是str类型

str的编码是与系统环境相关的，一般就是sys.getfilesystemencoding()得到的值

所以从unicode转str，要用encode方法

从str转unicode，所以要用decode

例如：

```
# coding=utf-8   #默认编码格式为utf-8
```

```
s = u'中文' #unicode编码的文字
print s.encode('utf-8')  #转换成utf-8格式输出
print s #效果与上面相同，似乎默认直接转换为指定编码
```

我的总结：

```
u=u'unicode编码文字'
g=u.encode('gbk') #转换为gbk格式
print g #此时为乱码，因为当前环境为utf-8,gbk编码文字为乱码
str=g.decode('gbk').encode('utf-8')  #以gbk编码格式读取g（因为他就是gbk编码的）并转换为utf-8格式输出
print str #正常显示中文
```

安全的方法：

```
s.decode('gbk','ignore').encode('utf-8’) #以gbk编码读取（当然是读取gbk编码格式的文字了）并忽略错误的编码，转换成utf-
```

因为decode的函数原型是`decode([encoding], [errors='strict'])`，可以用第二个参数控制错误处理的策略，默认的参数就是strict，代表遇到非法字符时抛出异常；

如果设置为ignore，则会忽略非法字符；

如果设置为replace，则会用?取代非法字符；

如果设置为xmlcharrefreplace，则使用XML的字符引用。