# Python csv模块的使用

**简介：**

[CSV (Comma Separated Values)](http://zh.wikipedia.org/zh-cn/%E9%80%97%E5%8F%B7%E5%88%86%E9%9A%94%E5%80%BC" \t "http://blog.csdn.net/u012162613/article/details/_blank)，即逗号分隔值（也称字符分隔值，因为分隔符可以不是逗号），是一种常用的文本格式，用以存储表格数据，包括数字或者字符。csv虽然使用广泛，但却没有通用的标准，所以在处理csv格式时常常会碰到麻烦，幸好python内置了csv模块。下面简单介绍csv模块中最常用的一些函数。细节请访问[https://docs.python.org/2/library/csv.html#module-csv](https://docs.python.org/2/library/csv.html" \l "module-csv" \t "http://blog.csdn.net/u012162613/article/details/_blank)

也许你会说，我为什么要学习使用CSV模块呢？没有CSV模块我一样可以解析操作CSV文件，比如下面这种代码：

with open('stocks.csv') as f:  
for line in f:  
row = line.split(',')  
# process row

使用这种方式的一个缺点就是你仍然需要去处理一些棘手的细节问题。比如，如果某些字段值被引号包围，你不得不去除这些引号。另外，如果一个被引号包围的字段碰巧含有一个逗号，那么程序就会因为产生一个错误而停止。

默认情况下， CSV 库可识别 Microsoft Excel 所使用的 CSV 编码规则。这或许也是最常见的形式，并且也会给你带来最好的兼容性。然而，如果你查看 CSV 的文档，就会发现有很多种方法将它应用到其他编码格式上（如修改分隔字符等，用Tab分隔）。所以你应该总是优先选择 CSV 模块分割或解析 CSV 数据。

## **csv模块中的函数**

* **writer(csvfile, dialect='excel', \*\*fmtparams)**

参数说明：

csvfile，必须是支持迭代(Iterator)的对象，可以是文件(file)对象或者列表(list)对象，如果是文件对象，打开时需要加"b"标志参数。

dialect，编码风格，默认为excel的风格，也就是用逗号（,）分隔，dialect方式也支持自定义，通过调用register\_dialect方法来注册，下文会提到。

fmtparam，格式化参数，用来覆盖之前dialect对象指定的编码风格。

import csv

with open('test.csv','rb') as myFile:

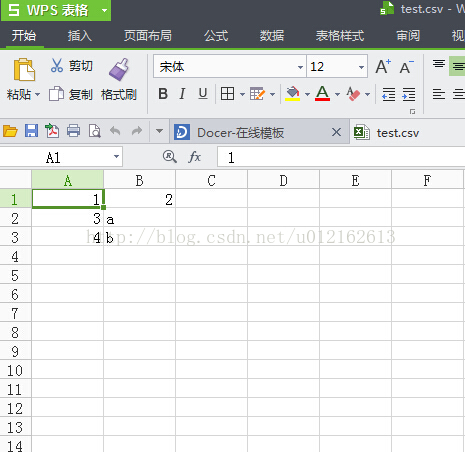
lines=csv.reader(myFile)

for line in lines:

print line

'test.csv'是文件名，‘rb’中的r表示“读”模式，因为是文件对象，所以加‘b’。open()返回了一个文件对象myFile，reader(myFile)只传入了第一个参数，另外两个参数采用缺省值，即以excel风格读入。reader()返回一个reader对象lines,lines是一个list，当调用它的方法lines.next()时，会返回一个string。上面程序的效果是将csv文件中的文本按行打印，每一行的元素都是以逗号分隔符','分隔得来。

在我的test.csv文件中，存储的数据如图：



程序输出：

['1', '2']  
['3', 'a']  
['4', 'b']

补充：reader对象还提供一些方法：line\_num、dialect、next()

* **writer(csvfile, dialect='excel', \*\*fmtparams)**

参数的意义同上，这里不赘述，直接上例程：

with open('t.csv','wb') as myFile:

myWriter=csv.writer(myFile)

myWriter.writerow([7,'g'])

myWriter.writerow([8,'h'])

myList=[[1,2,3],[4,5,6]]

myWriter.writerows(myList)

'w'表示写模式。

首先open()函数打开当前路径下的名字为't.csv'的文件，如果不存在这个文件，则创建它，返回myFile文件对象。

csv.writer(myFile)返回writer对象myWriter。

writerow()方法是一行一行写入，writerows方法是一次写入多行。

注意：如果文件't.csv'事先存在，调用writer函数会先清空原文件中的文本，再执行writerow/writerows方法。

补充：除了writerow、writerows，writer对象还提供了其他一些方法：writeheader、dialect

* **register\_dialect(name, [dialect, ]\*\*fmtparams)**

这个函数是用来自定义dialect的。(注册编码风格)

参数说明：

name,你所自定义的dialect的名字，比如默认的是'excel'，你可以定义成'mydialect'

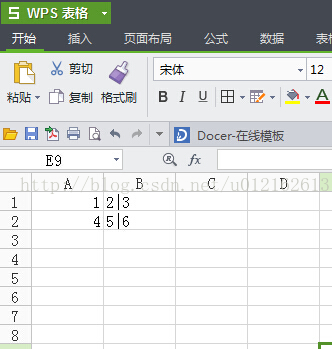
[dialect, ]\*\*fmtparams，dialect格式参数，有delimiter（分隔符，默认的就是逗号）、quotechar、

quoting等等，可以参考https://docs.python.org/2/library/csv.html#csv-fmt-params

csv.register\_dialect('mydialect',delimiter='|', quoting=csv.QUOTE\_ALL)

上面一行程序自定义了一个命名为mydialect的dialect，参数只设置了delimiter和quoting这两个，其他的仍然采用

默认值，其中以'|'为分隔符。接下来我们就可以像使用'excel'一样来使用'mydialect'了。我们来看看效果：



以'mydialect'风格打印：

[python] [view plain](http://blog.csdn.net/u012162613/article/details/41915859" \o "view plain) [copy](http://blog.csdn.net/u012162613/article/details/41915859" \o "copy)

1. with open('test.csv','rb') as myFile:
2. lines=csv.reader(myFile,'mydialect')
3. print lines.line\_num
4. for line in lines:
5. print line

输出：

['1,2', '3']  
['4,5', '6']

可以看到，现在是以'|'为分隔符，1和2合成了一个字符串（因为1和2之间的分隔符是逗号，而mydialect风格的分隔符是'|'），3单独一个字符串。对于writer()函数，同样可以传入mydialect作为参数，这里不赘述。

* **unregister\_dialect(name)**

这个函数用于注销自定义的dialect

此外，csv模块还提供get\_dialect(name)、list\_dialects()、field\_size\_limit([new\_limit])等函数，这些都比较简单，可以自己试试。比如list\_dialects()函数会列出当前csv模块里所有的dialect：

[python] [view plain](http://blog.csdn.net/u012162613/article/details/41915859" \o "view plain) [copy](http://blog.csdn.net/u012162613/article/details/41915859" \o "copy)

print csv.list\_dialects()

输出：

['excel-tab', 'excel', 'mydialect']

'mydialect'是自定义的，'excel-tab', 'excel'都是自带的dialect，其中'excel-tab'跟'excel'差不多，只不过它以tab为分隔符。