**Pandas 读写csv**，逗号分隔值（Comma-Separated Values，CSV，有时也称为字符分隔值，因为分隔字符也可以不是逗号），其文件以纯文本形式存储表格数据（数字和文本）,CSV是一种通用的、相对简单的文件格式，被用户、商业和科学广泛应用。Pandas的下列函数专门用来处理这种文件类型。

* read\_csv
* read\_table
* to\_csv

**从csv读取数据**

根据通常经验，对数据分析人员来说，最常执行的操作是从CSV文件或其他类型的文本文件中读取数据。我们在工作目录中创建一个短小的csv文件，文件内容如下:  
  
这个文件默认以逗号作为分隔符，因此我们可以使用read\_csv()函数读取它的内容，同时将它转换为DataFrame对象。

import pandas as pd

csvframe = pd.read\_csv("testcsv\_01.csv")

print(csvframe)

Python

Copy

输出结果如下:

white red blue green animal

0 1 5 2 3 cat

1 2 7 8 5 dog

2 3 3 6 7 horse

3 2 2 8 3 duck

4 4 4 2 1 mouse

Shell

Copy

如上所示，读取csv文件中的数据很简单，csv文件中的数据为列表数据，位于不同的列的数据用逗号隔开。既然csv文件被视为文本文件，也可以使用read\_table()函数读取数据，此函数需要指定分隔符。如下例所示：

import pandas as pd

csvframe = pd.read\_table("testcsv\_01.csv",",")

print(csvframe)

Python

Copy

输出结果如下:

white red blue green animal

0 1 5 2 3 cat

1 2 7 8 5 dog

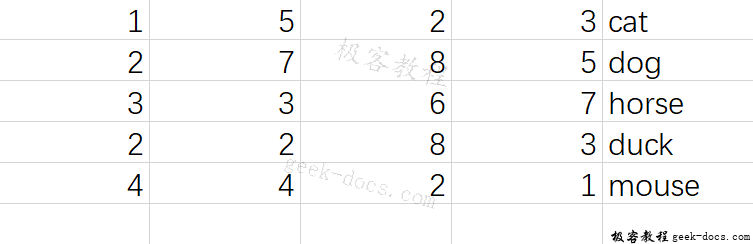
2 3 3 6 7 horse

3 2 2 8 3 duck

4 4 4 2 1 mouse

Shell

Copy

从上例可知，标识各列名称的表头位于csv文件的第一行，但一般情况并非如此，往往csv文件的第一行就是列表数据，如下图所示：  
  
对于没有表头这种情况，使用header选项，将其值置为None，Pandas会为其添加默认表头。

import pandas as pd

csvframe = pd.read\_csv("testcsv\_02.csv",header=None)

print(csvframe)

Python

Copy

输出结果如下:

0 1 2 3 4

0 1 5 2 3 cat

1 2 7 8 5 dog

2 3 3 6 7 horse

3 2 2 8 3 duck

4 4 4 2 1 mouse

Shell

Copy

此外，还可以使用names选项指定表头，直接把存在各列名称的数组赋给它即可。

import pandas as pd

csvframe = pd.read\_csv("testcsv\_02.csv",names=['white','red','blue','green','animal'])

print(csvframe)

Python

Copy

数据结果如下:

white red blue green animal

0 1 5 2 3 cat

1 2 7 8 5 dog

2 3 3 6 7 horse

3 2 2 8 3 duck

4 4 4 2 1 mouse

Shell

Copy

假如你想读取csv文件，创建一个具有等级结构的DataFrame对象，为此，可以添加index\_col选项，扩展read\_csv()函数的功能，把所有想转换为索引的列名称赋值给index\_col。如下图所示:  


import pandas as pd

csvframe = pd.read\_csv("testcsv\_03.csv",index\_col=['color','status'])

print(csvframe)

Python

Copy

输出结果如下:

Item1 Item2 Item3

color status

black up 3 4 6

down 2 6 7

white up 5 5 5

down 3 3 2

left 1 2 1

red up 2 2 2

down 1 1 4

Shell

Copy

**往csv文件写入数据**

从文件读取数据很常用，把计算结构或数据结构所包含的数据写入数据文件也是常用的必要操作。

例如，你可能想把DataFrame中数据写入csv中，在写入过程中，就要用到to\_csv()函数，其参数为即将生成的文件名。如下所示:

import pandas as pd

frame = pd.DataFrame([[0,1,2,3],[4,5,6,7],[8,9,10,11],[12,13,14,15]], columns=['ball', 'pen', 'pencil', 'paper'])

print(frame)

frame.to\_csv("testcsv\_04.csv")

Python

Copy

输出结果如下:

ball pen pencil paper

0 0 1 2 3

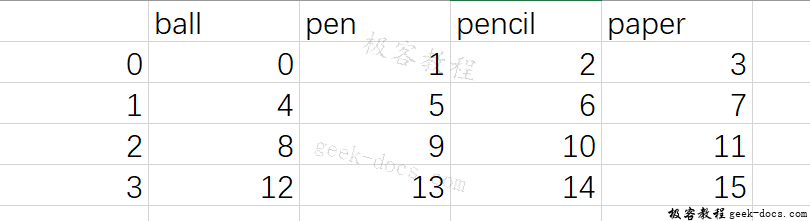
1 4 5 6 7

2 8 9 10 11

3 12 13 14 15

Shell

Copy

生成的testcsv\_04.csv文件，显示内容如下:  


如上面例子所示，把DataFrame写入文件时，索引和列名称连同数据一起写入，使用index和header选项，把它们的值设置为false，可取消这一默认行为。

import pandas as pd

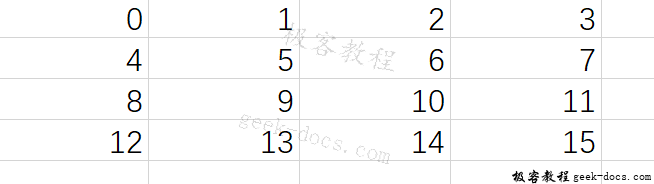
frame = pd.DataFrame([[0,1,2,3],[4,5,6,7],[8,9,10,11],[12,13,14,15]], columns=['ball', 'pen', 'pencil', 'paper'])

print(frame)

frame.to\_csv("testcsv\_05.csv", index=False,header=False)

Python

Copy

生成的testcsv\_05.csv文件，显示内容如下:  


需要注意的是，数据结构中的NaN写入文件后，显示为空字段，如下所示:

import pandas as pd

import numpy as np

frame = pd.DataFrame([[0,1,2,3],[np.NaN,np.NaN,np.NaN,7],[8,9,10,11],[12,13,14,15]], columns=['ball', 'pen', 'pencil', 'paper'])

print(frame)

frame.to\_csv("testcsv\_06.csv", index=False,header=False)

Python

Copy

生成的testcsv\_06.csv文件，显示内容如下:  
  
可以通过to\_csv()函数的na\_rep选项把空字段替换成你需要的值，常用值有NULL, 0, NaN，如下所示：

import pandas as pd

import numpy as np

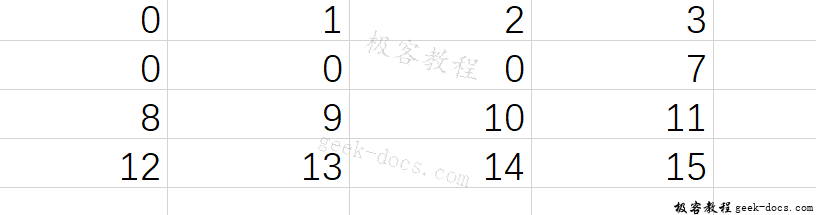
frame = pd.DataFrame([[0,1,2,3],[np.NaN,np.NaN,np.NaN,7],[8,9,10,11],[12,13,14,15]], columns=['ball', 'pen', 'pencil', 'paper'])

print(frame)

frame.to\_csv("testcsv\_07.csv", index=False,header=False, na\_rep='0')

Python

Copy

生成的testcsv\_07.csv文件，显示内容如下:  


注:DataFrame是我们本章讨论的主题，因为通常需要将这些数据结构写入文件，但是，这些函数同样适用于Series