实验2：数据分析库pandas的使用（二）

**什么是pandas？**

pandas 是基于NumPy 的一种工具，该工具是为了解决数据分析任务而创建的，可用于数据挖掘，同时也提供数据清洗功能。Pandas 纳入了大量库和一些标准的数据模型，提供了高效地操作大型数据集所需的工具。pandas提供了大量能使我们快速便捷地处理数据的函数和方法。它是使Python成为强大而高效的数据分析环境的重要因素之一。

# 1、Pandas迭代

Pandas对象之间的基本迭代的行为取决于类型。当迭代一个系列时，它被视为数组式，基本迭代产生这些值。其他数据结构，如：DataFrame和Panel，遵循类似惯例迭代对象的键。

简而言之，基本迭代(对于i在对象中)产生 -

* *Series* - 值
* *DataFrame* - 列标签
* *Pannel* - 项目标签

## 迭代DataFrame

迭代DataFrame提供列名。

**示例**

import pandas as pd

import numpy as np

N=20

df = pd.DataFrame({

'A': pd.date\_range(start='2016-01-01',periods=N,freq='D'),

'x': np.linspace(0,stop=N-1,num=N),

'y': np.random.rand(N),

'C': np.random.choice(['Low','Medium','High'],N).tolist(),

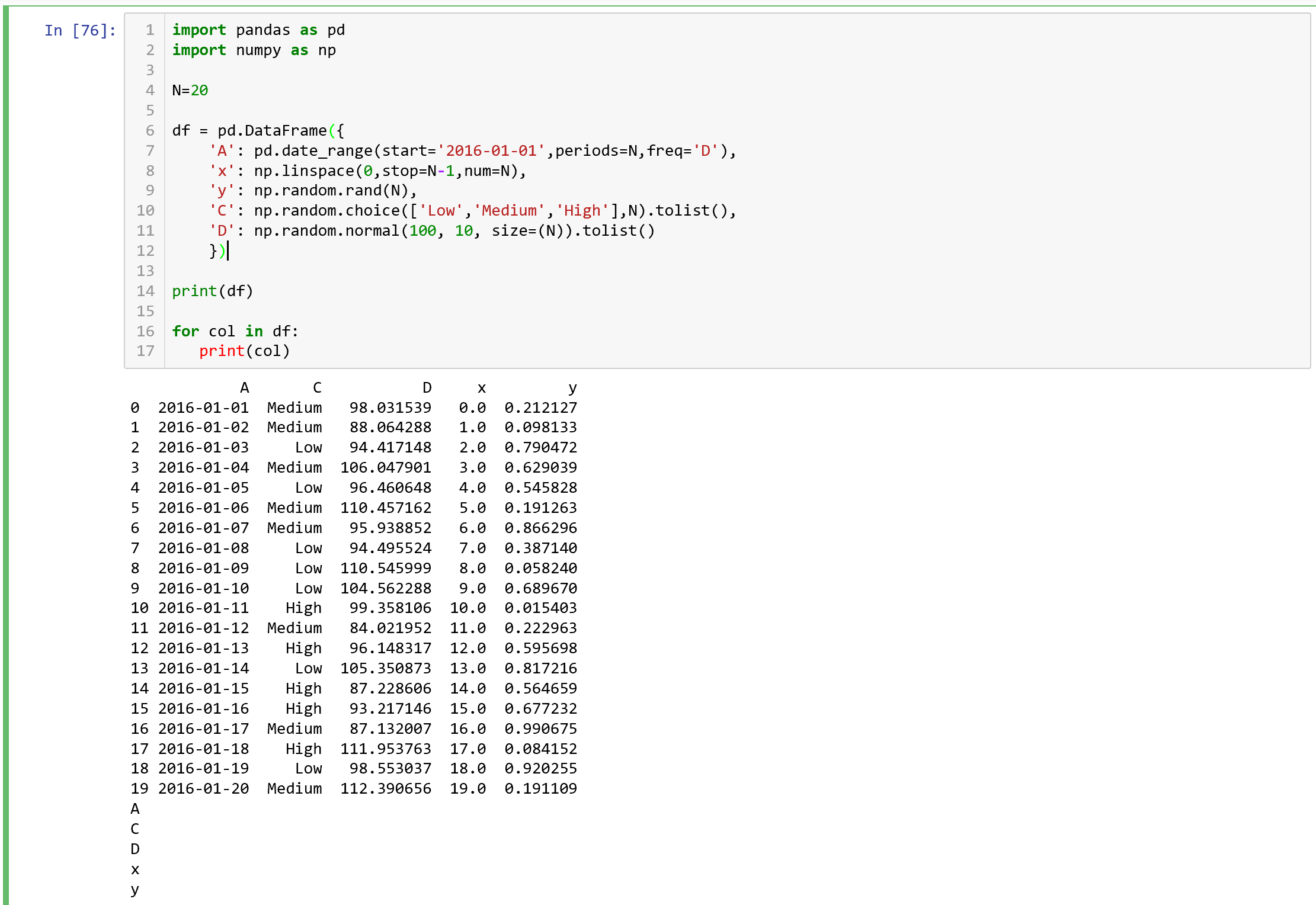
'D': np.random.normal(100, 10, size=(N)).tolist()

})

print(df)

for col in df:

print(col)



要遍历数据帧(DataFrame)中的行，可以使用以下函数 -

* iteritems() - 迭代(key，value)对
* iterrows() - 将行迭代为(索引，系列)对
* itertuples() - 以namedtuples的形式迭代行

**iterrows()示例**

iterrows()返回迭代器，产生每个索引值以及包含每行数据的序列。

**示例**

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(4,3),columns = ['col1','col2','col3'])

print(df,'\n')

for row\_index,row in df.iterrows():

print(row\_index)

print(row)

**itertuples()示例**

itertuples()方法将为DataFrame中的每一行返回一个产生一个命名元组的迭代器。元组的第一个元素将是行的相应索引值，而剩余的值是行值。

**示例**

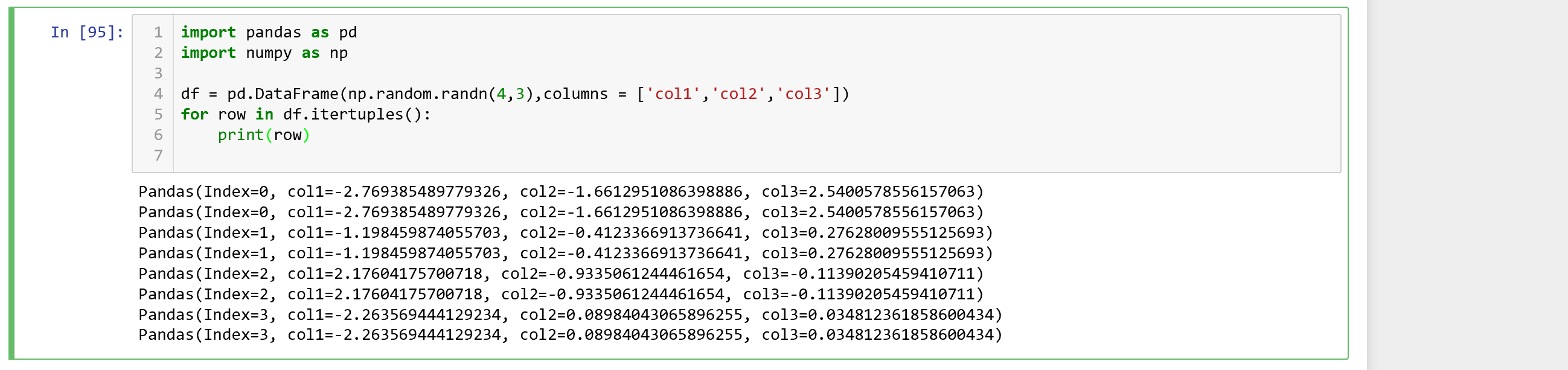
import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(4,3),columns = ['col1','col2','col3'])

for row in df.itertuples():

print(row)



注意 - 不要尝试在迭代时修改任何对象。迭代是用于读取，迭代器返回原始对象(视图)的副本，因此更改将不会反映在原始对象上。

**示例**

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(4,3),columns = ['col1','col2','col3'])

print(df)

for index, row in df.iterrows():

row['a'] = 10

print(df)



# 2、Pandas排序

*Pandas*有两种排序方式，它们分别是 -

* 按标签
* 按实际值

下面来看看一个输出的例子。

## 2.1 按标签排序

使用sort\_index()方法，通过传递axis参数和排序顺序，可以对DataFrame进行排序。 默认情况下，按照升序对行标签进行排序。

**示例**

import pandas as pd

import numpy as np

unsorted\_df = pd.DataFrame(np.random.randn(10,2),index=[1,4,6,2,3,5,9,8,0,7],columns = ['col2','col1'])

print(unsorted\_df,'\n')

sorted\_df=unsorted\_df.sort\_index(axis=0, ascending=True)

print(sorted\_df)



## 2.2 按值排序

像索引排序一样，sort\_values()是按值排序的方法。它接受一个by参数，它将使用要与其排序值的DataFrame的列名称。

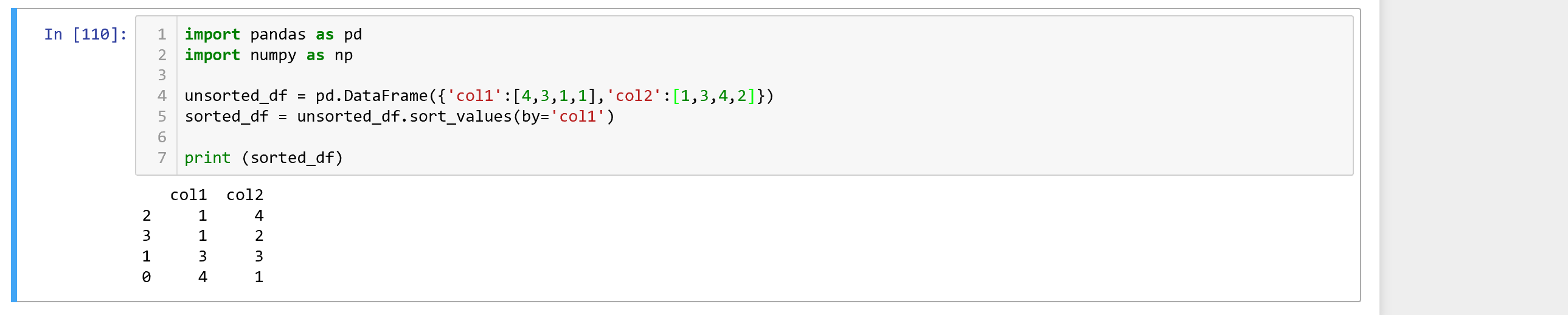
**示例**

import pandas as pd

import numpy as np

unsorted\_df = pd.DataFrame({'col1':[4,3,1,1],'col2':[1,3,4,2]})

sorted\_df = unsorted\_df.sort\_values(by='col1')

print(sorted\_df)

**示例**

import pandas as pd

import numpy as np

unsorted\_df = pd.DataFrame({'col1':[4,3,1,1],'col2':[1,3,4,2]})

sorted\_df = unsorted\_df.sort\_values(by=['col1','col2'])

print(sorted\_df)



# 3、Pandas字符串和文本数据

*Pandas*提供了一组字符串函数，可以方便地对字符串数据进行操作。 最重要的是，这些函数忽略(或排除)丢失/NaN值。

几乎这些方法都使用Python字符串函数(请参阅： <http://docs.python.org/3/library/stdtypes.html#string-methods> )。 因此，将Series对象转换为String对象，然后执行该操作。



*Pandas*提供了一组字符串函数，可以方便地对字符串数据进行操作。 最重要的是，这些函数忽略(或排除)丢失/NaN值。

import pandas as pd

import numpy as np

s = pd.Series(['Tom', 'William Rick', 'John', 'Alber@t', np.nan, '1234','SteveMinsu'])

print(s)

print (s.str.lower())



# 4、Pandas窗口函数

为了处理数字数据，Pandas提供了几个变体，如滚动，展开和指数移动窗口统计的权重。 其中包括总和，均值，中位数，方差，协方差，相关性等。

下来学习如何在DataFrame对象上应用上提及的每种方法。

## 2.1 .rolling()函数

这个函数可以应用于一系列数据。指定window=n参数并在其上应用适当的统计函数。

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randint(0,10,(10, 4)),

index = pd.date\_range('1/1/2020', periods=10),

columns = ['A', 'B', 'C', 'D'])

print(df)

rol = df.rolling(window=3)

print(rol.mean())

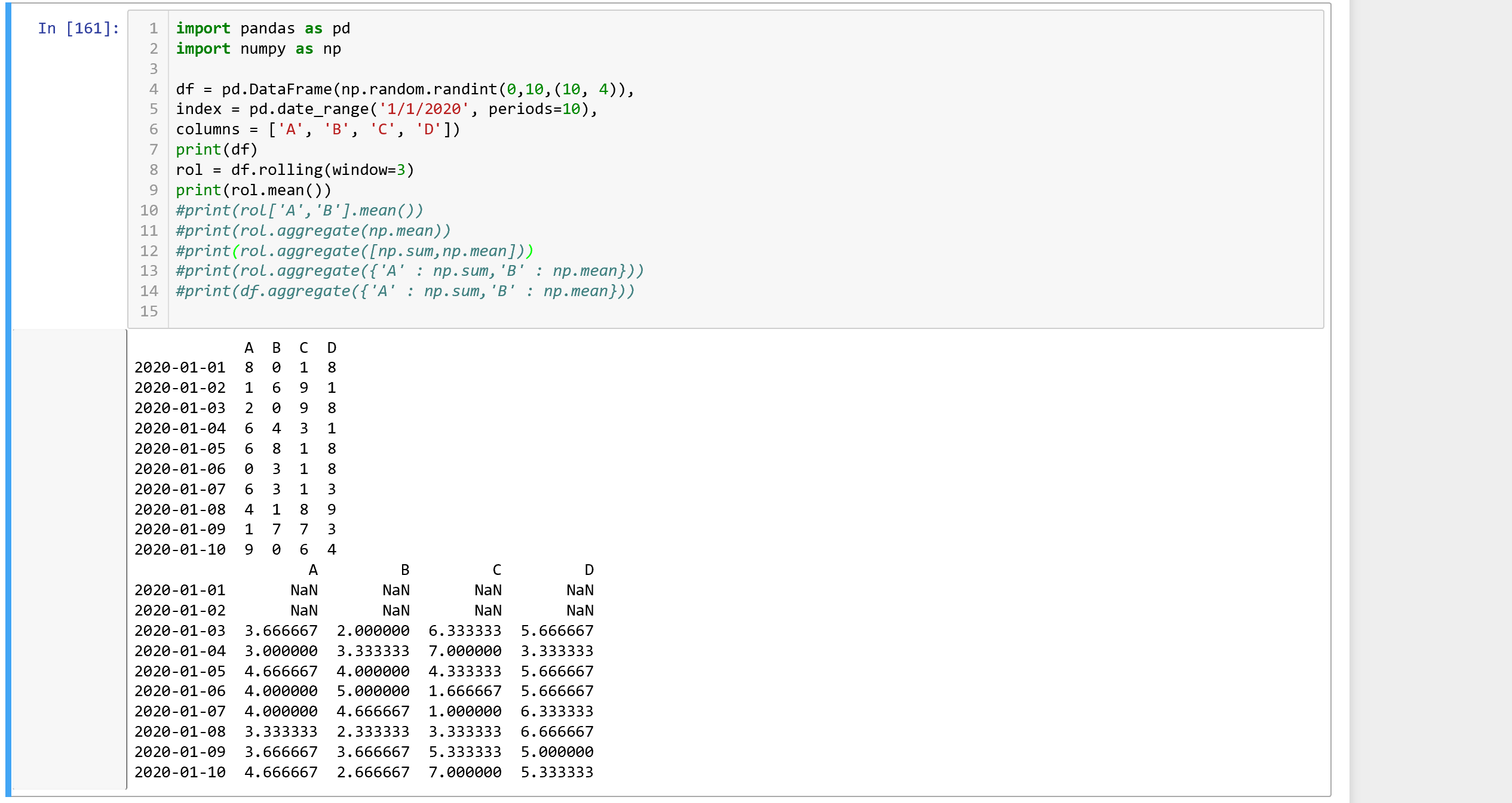
#print(rol['A','B'].mean())

#print(rol.aggregate(np.mean))

#print(rol.aggregate([np.sum,np.mean]))

#print(rol.aggregate({'A' : np.sum,'B' : np.mean}))

#print(df.aggregate({'A' : np.sum,'B' : np.mean}))



注 - 由于窗口大小为3(window)，前两个元素有空值，第三个元素的值将是n，n-1和n-2元素的平均值。这样也可以应用上面提到的各种函数了。

## 2.2 .expanding()函数

这个函数可以应用于一系列数据。 指定min\_periods = n参数并在其上应用适当的统计函数。

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randint(0,10,(10, 4)),

index = pd.date\_range('1/1/2020', periods=10),

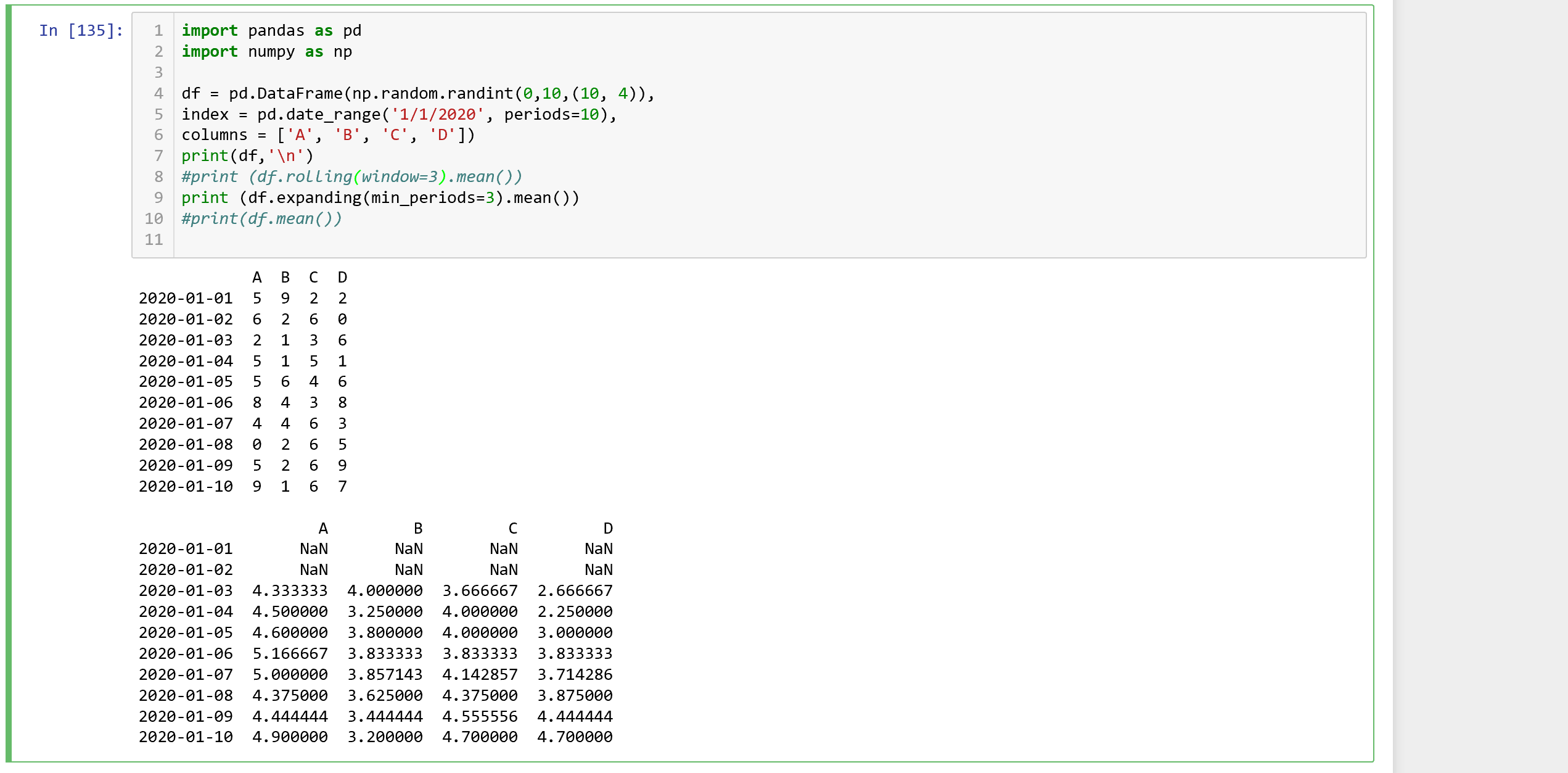
columns = ['A', 'B', 'C', 'D'])

print(df,'\n')

#print (df.rolling(window=3).mean())

print (df.expanding(min\_periods=3).mean())

#print(df.mean())



## 2.3 .ewm()函数

指数加权滑动（ewm），常用于时间序列数据。指定com，span，halflife参数，并在其上应用适当的统计函数。它以指数形式分配权重。

import pandas as pd

import numpy as np

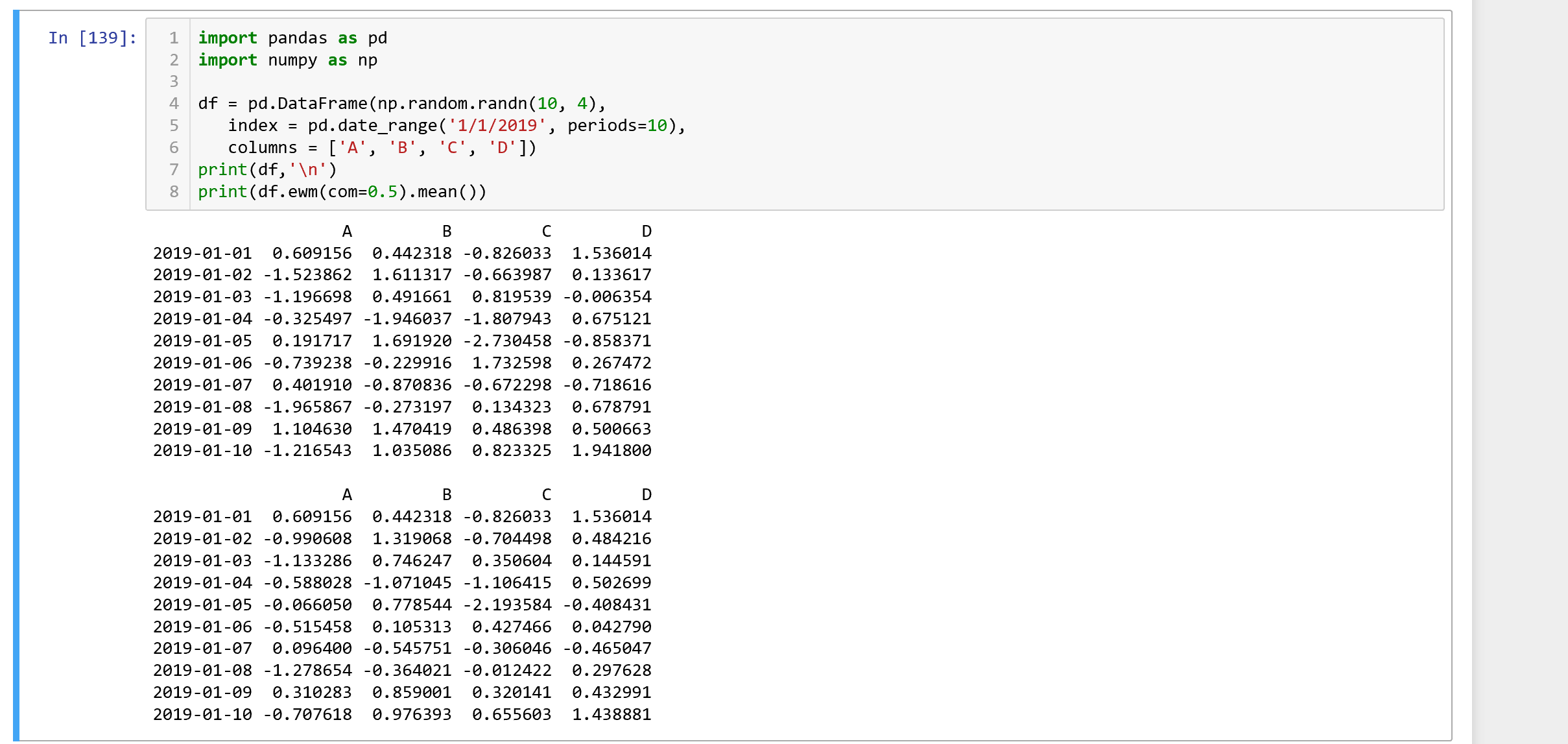
df = pd.DataFrame(np.random.randn(10, 4),

index = pd.date\_range('1/1/2019', periods=10),

columns = ['A', 'B', 'C', 'D'])

print(df,'\n')

print(df.ewm(com=0.5).mean())



窗口函数主要用于通过平滑曲线来以图形方式查找数据内的趋势。如果日常数据中有很多变化，并且有很多数据点可用，那么采样和绘图就是一种方法，应用窗口计算并在结果上绘制图形是另一种方法。 通过这些方法，可以平滑曲线或趋势。

# 5、Pandas分组（GroupBy）

为了处理数字数据，Pandas提供了几个变体，如滚动，展开和指数移动窗口统计的权重。 其中包括总和，均值，中位数，方差，协方差，相关性等。

任何分组(groupby)操作都涉及原始对象的以下操作之一。它们是 -

* 分割对象
* 应用一个函数
* 结合的结果

在许多情况下，我们将数据分成多个集合，并在每个子集上应用一些函数。在应用函数中，可以执行以下操作 -

* 聚合 - 计算汇总统计
* 转换 - 执行一些特定于组的操作
* 过滤 - 在某些情况下丢弃数据

下面来看看创建一个DataFrame对象并对其执行所有操作 -

import pandas as pd

ipl\_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],

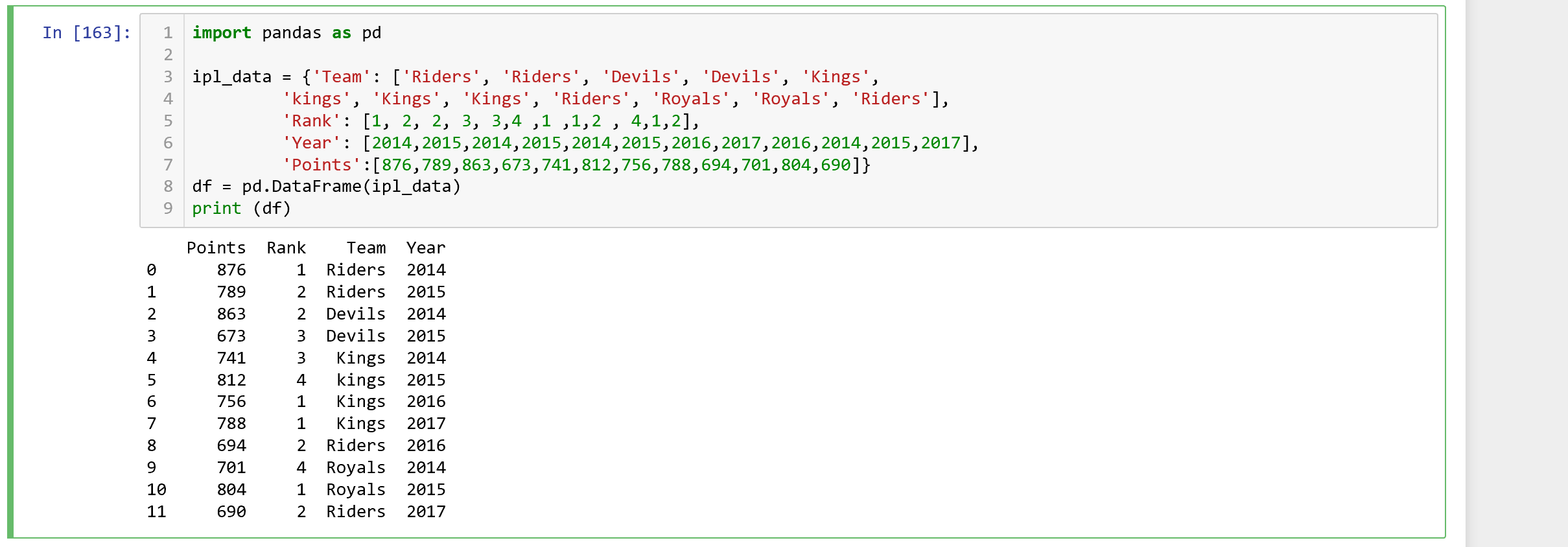
'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],

'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}

df = pd.DataFrame(ipl\_data)

print (df)



## 5.1 将数据拆分成组

Pandas对象可以分成任何对象。有多种方式来拆分对象，如 -

* obj.groupby(‘key’)
* obj.groupby([‘key1’,’key2’])
* obj.groupby(key,axis=1)

现在来看看如何将分组对象应用于DataFrame对象

**示例**

import pandas as pd

ipl\_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],

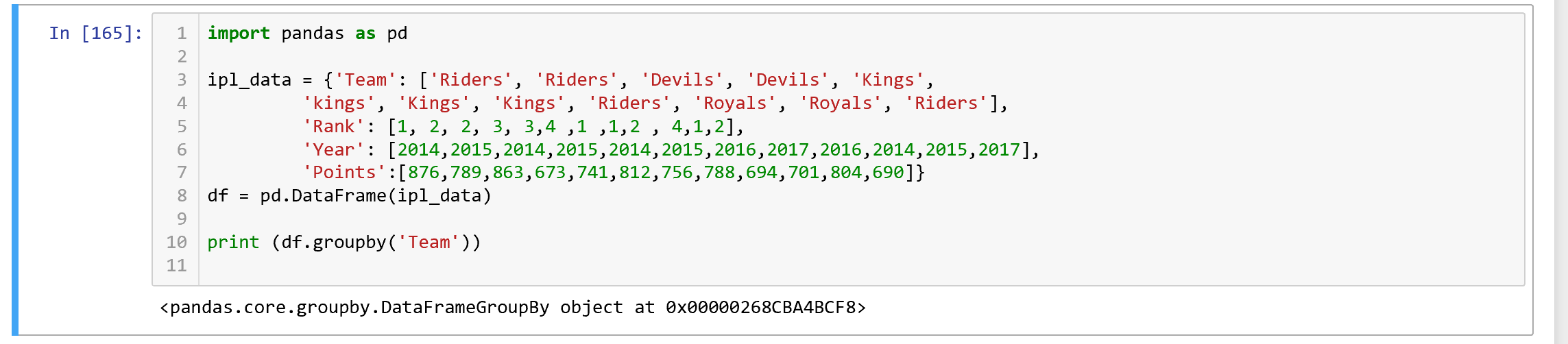
'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],

'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}

df = pd.DataFrame(ipl\_data)

print (df.groupby('Team'))



## 5.2 查看分组

import pandas as pd

ipl\_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],

'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],

'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}

df = pd.DataFrame(ipl\_data)

df.groupby('Team').groups



**示例**

按多列分组 -

import pandas as pd

ipl\_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],

'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],

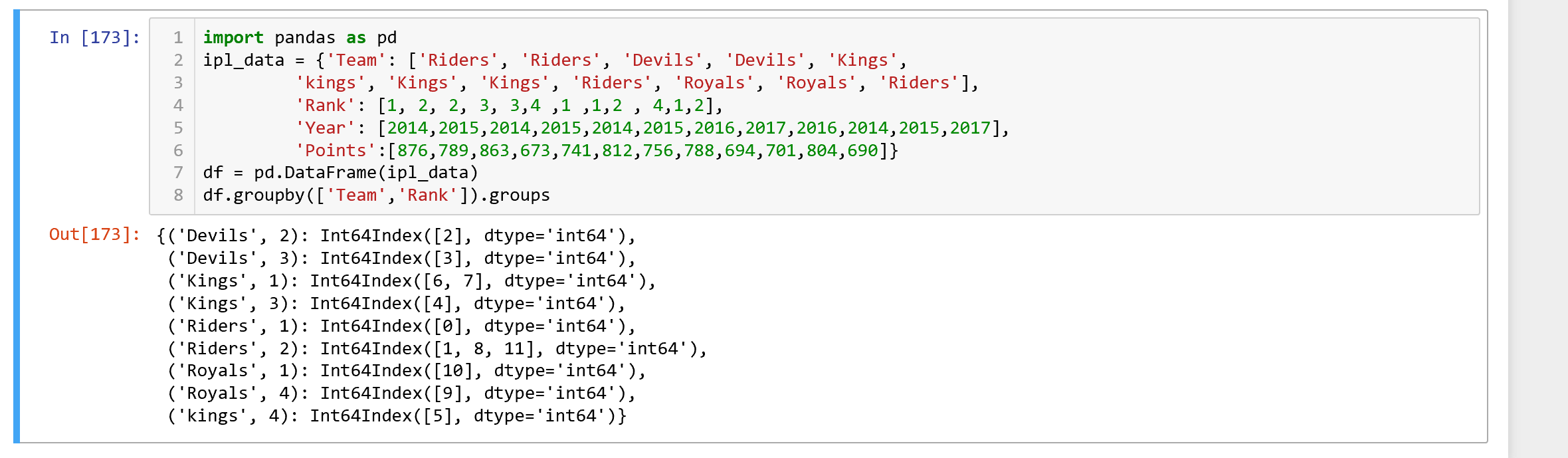
'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}

df = pd.DataFrame(ipl\_data)

df.groupby(['Team','Year']).groups

执行上面示例代码，得到以下结果 -



## 5.3 迭代遍历分组

使用groupby对象，可以遍历类似itertools.obj的对象。

import pandas as pd

ipl\_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],

'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],

'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}

df = pd.DataFrame(ipl\_data)

grouped = df.groupby('Year')

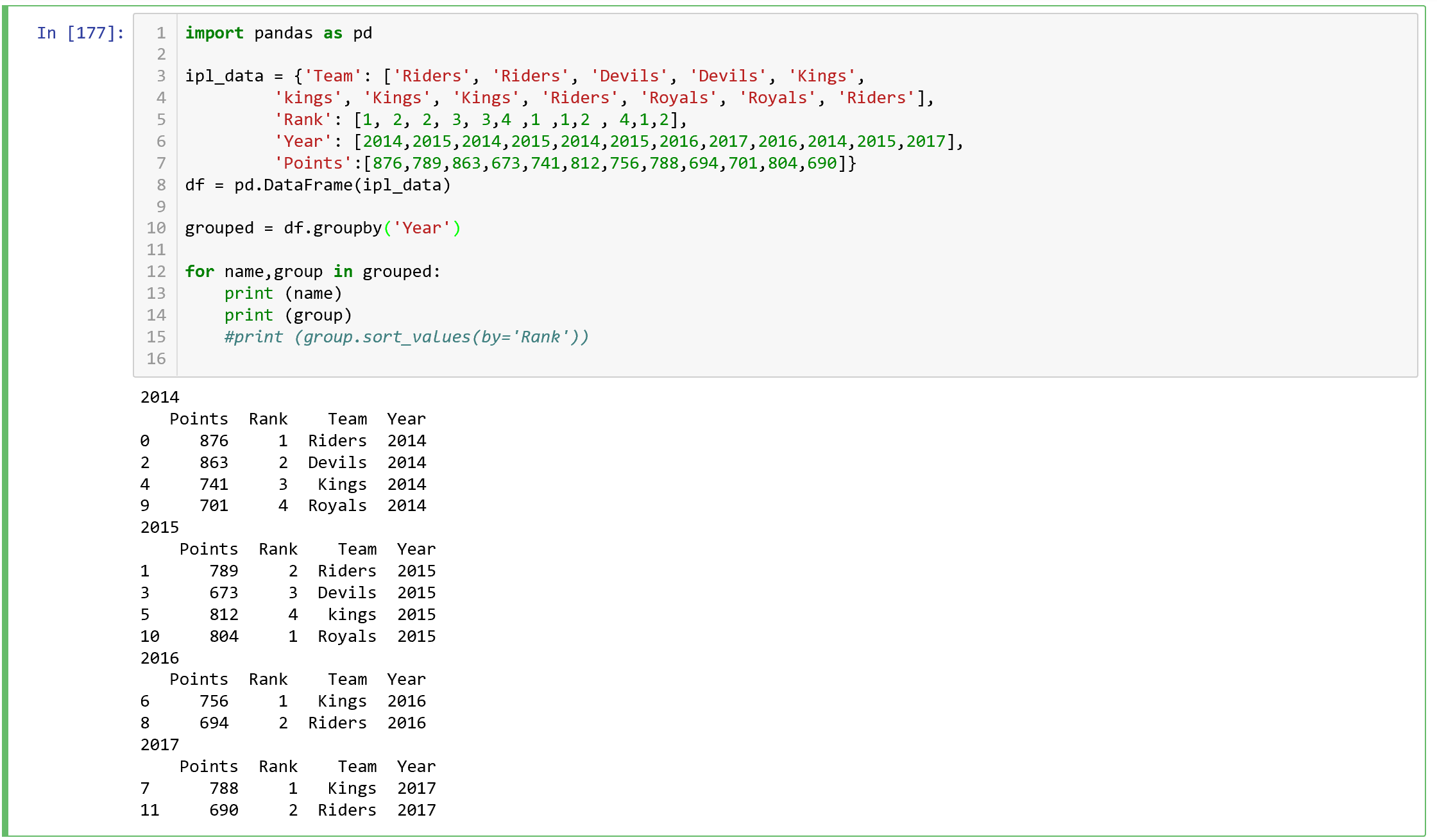
for name,group in grouped:

print (name)

print (group)

#print (group.sort\_values(by='Rank'))

执行上面示例代码，得到以下结果 -



默认情况下，groupby对象具有与分组名相同的标签名称。

## 5.4 选择一个分组

import pandas as pd

ipl\_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],

'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],

'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}

df = pd.DataFrame(ipl\_data)

grouped = df.groupby('Year')

print(grouped.get\_group(2014))



## 5.5 聚合

聚合函数为每个组返回单个聚合值。当创建了分组(group by)对象，就可以对分组数据执行多个聚合操作。

一个比较常用的是通过聚合或等效的agg方法聚合 -

import pandas as pd

import numpy as np

ipl\_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],

'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],

'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}

df = pd.DataFrame(ipl\_data)

grouped = df.groupby('Year')

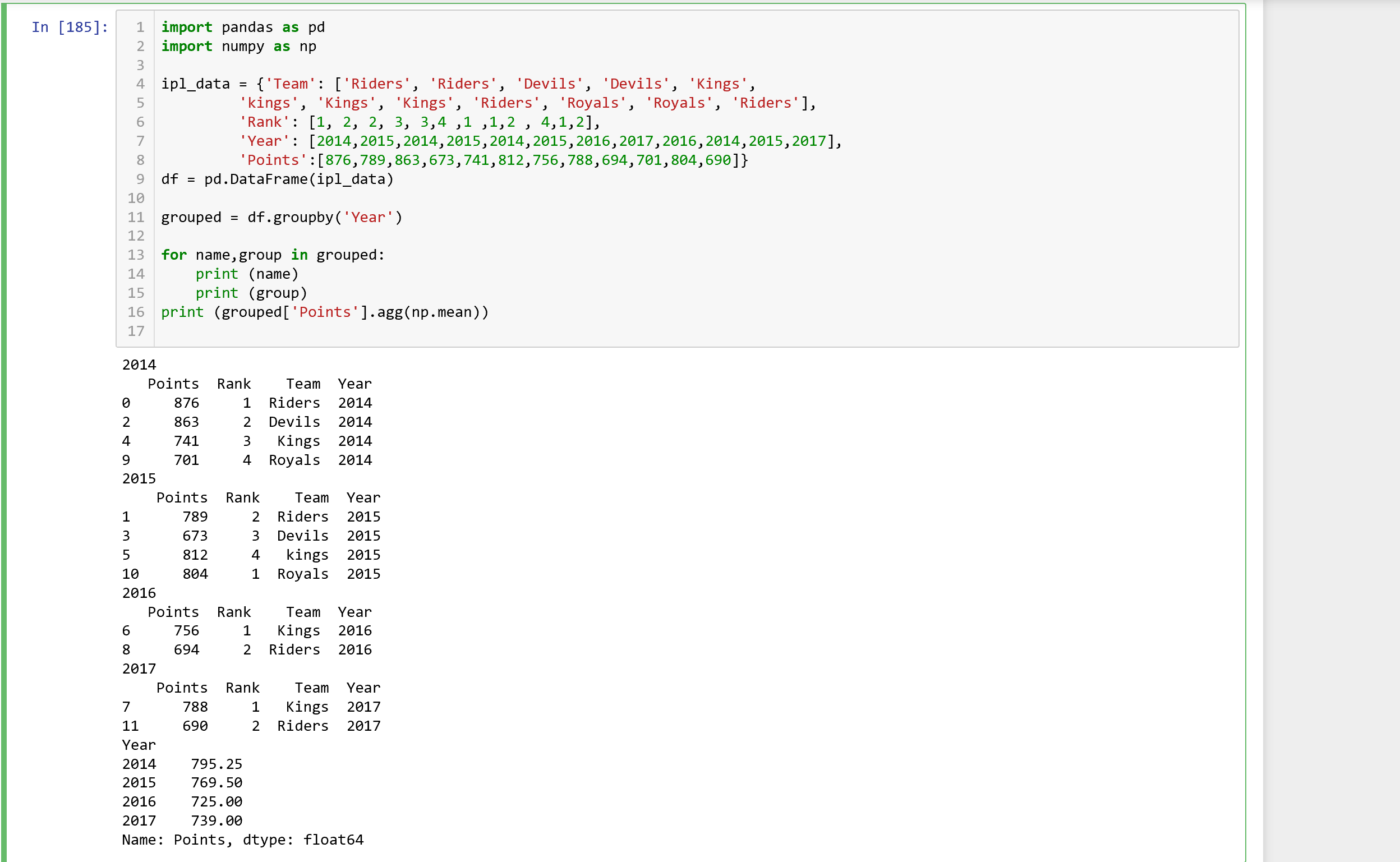
for name,group in grouped:

print (name)

print (group)

print (grouped['Points'].agg(np.mean))

#print (grouped.agg(np.mean))



另一种查看每个分组的大小的方法是应用size()函数 -

import pandas as pd

import numpy as np

ipl\_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],

'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],

'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}

df = pd.DataFrame(ipl\_data)

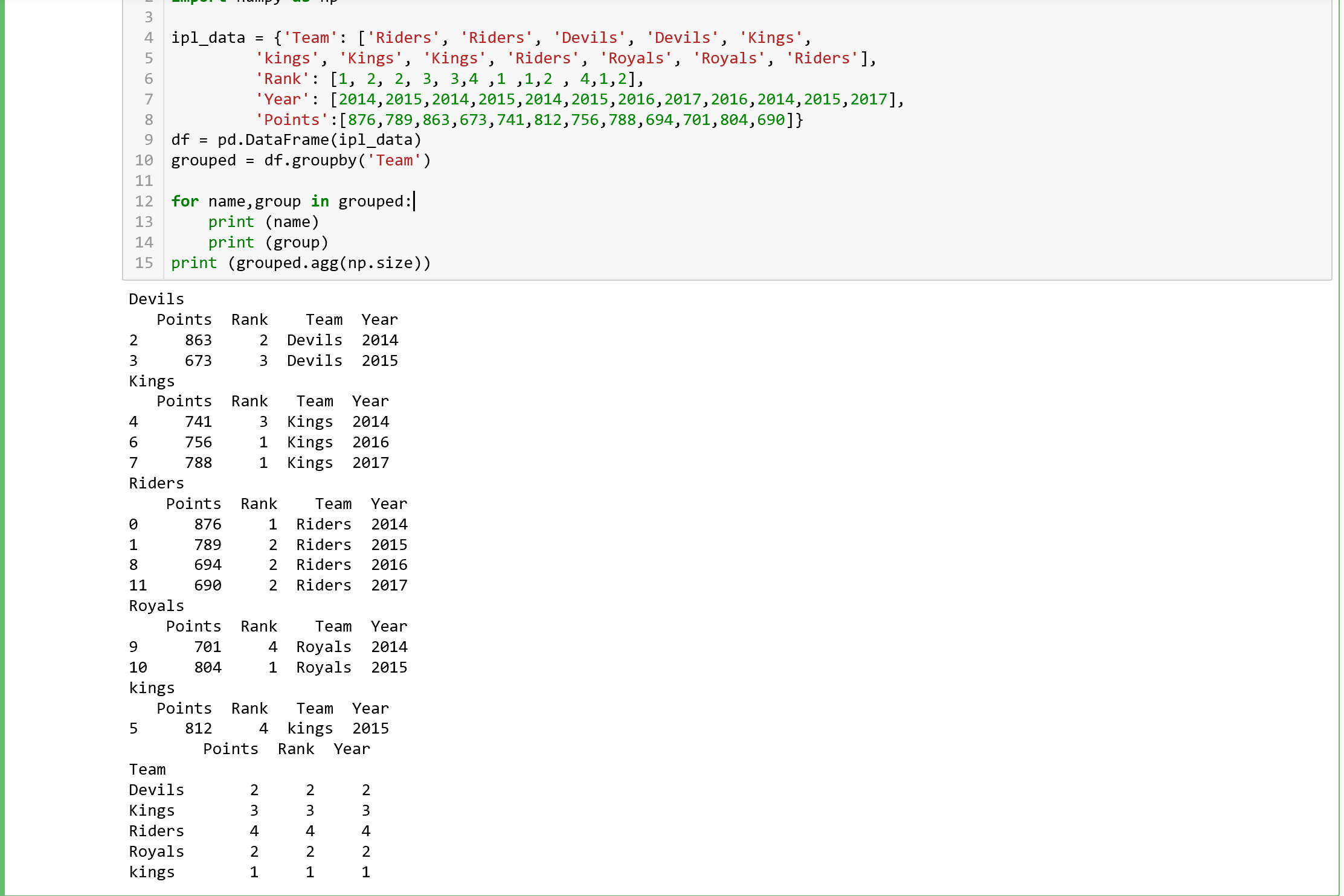
grouped = df.groupby('Team')

for name,group in grouped:

print (name)

print (group)

print (grouped.agg(np.size))



## 5.6 一次应用多个聚合函数

通过分组系列，还可以传递函数的列表或字典来进行聚合，并生成DataFrame作为输出 -

import pandas as pd

import numpy as np

ipl\_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],

'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],

'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}

df = pd.DataFrame(ipl\_data)

grouped = df.groupby('Team')

agg = grouped['Points'].agg([np.sum, np.mean, np.std])

print(agg)



## 5.7 转换

分组或列上的转换返回索引大小与被分组的索引相同的对象。因此，转换应该返回与组块大小相同的结果。

import pandas as pd

import numpy as np

ipl\_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],

'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],

'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}

df = pd.DataFrame(ipl\_data)

grouped = df.groupby('Team')

for name,group in grouped:

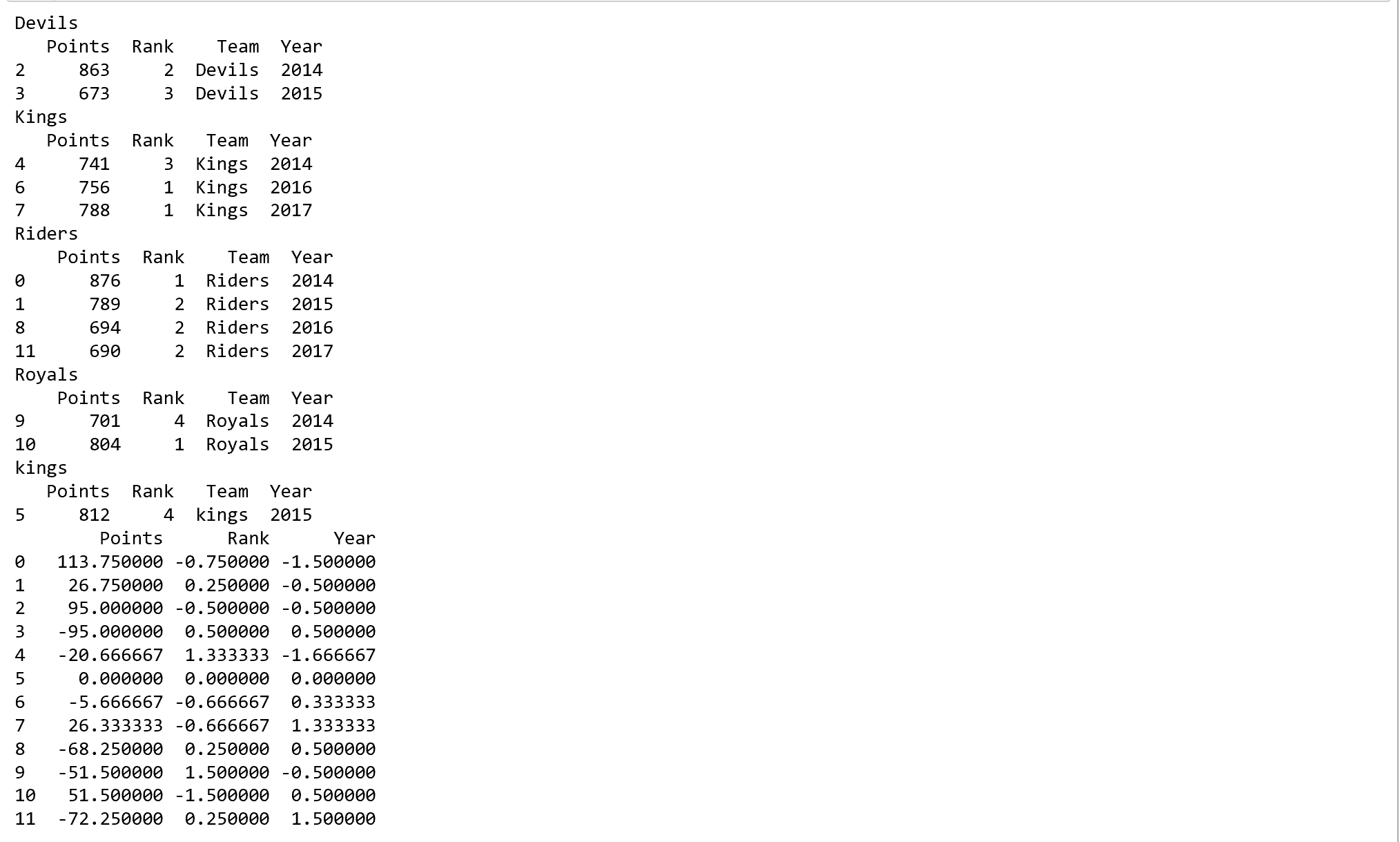
print (name)

print (group)

#score = lambda x: (x - x.mean()) / x.std()\*10

score = lambda x: x - x.mean()

print (grouped.transform(score))



## 5.8 过滤

过滤根据定义的标准过滤数据并返回数据的子集。filter()函数用于过滤数据。

import pandas as pd

import numpy as np

ipl\_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',

'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],

'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],

'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],

'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}

df = pd.DataFrame(ipl\_data)

for name,group in grouped:

print (name)

print (group)

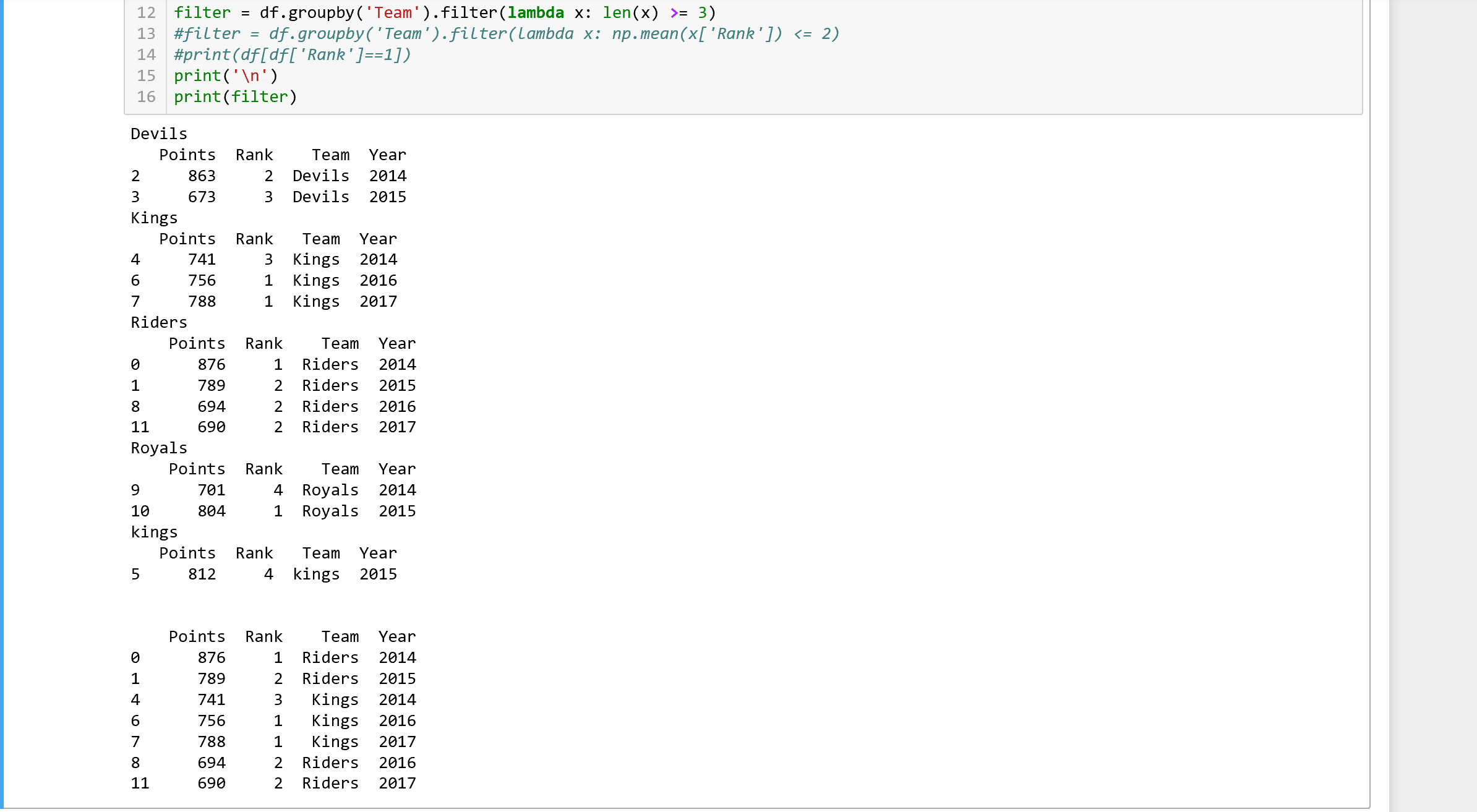
filter = df.groupby('Team').filter(lambda x: len(x) >= 3)

#filter = df.groupby('Team').filter(lambda x: np.mean(x['Rank']) <= 2)

#print(df[df['Rank']==1])

print('\n')

print(filter)



# 练习

以下三题采用学生数据集，如下：

import pandas as pd

stu\_dic = {'Age':[14,13,13,14,14,12,12,15,13,12,11,14,12,15,16,12,15,11,15],

'Height':[69,56.5,65.3,62.8,63.5,57.3,59.8,62.5,62.5,59,51.3,64.3,56.3,66.5,72,64.8,67,57.5,66.5],

'Name':['Alfred','Alice','Barbara','Carol','Henry','James','Jane','Janet','Jeffrey','John','Joyce','Judy','Louise','Marry','Philip','Robert','Ronald','Thomas','Willam'],

'Sex':['M','F','F','F','M','M','F','F','M','M','F','F','F','F','M','M','M','M','M'],

'Weight':[112.5,84,98,102.5,102.5,83,84.5,112.5,84,99.5,50.5,90,77,112,150,128,133,85,112]}

student = pd.DataFrame(stu\_dic)

1.按照性别分类，计算各组别中学生身高和体重的平均值

2.按照性别与年龄一起分类，计算各组别身高和体重的平均值和中位数

3.按照学生的年龄升序排序，年龄一样则按照身高降序排序