实验2：数据分析库pandas的使用（一）

**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数据结构

Pandas处理以下三个数据结构 -

* 系列(Series)
* 数据帧(DataFrame)
* 面板(Panel)

这些数据结构构建在Numpy数组之上，这意味着它们很快。

**维数和描述**

考虑这些数据结构的最好方法是，较高维数据结构是其较低维数据结构的容器。 例如，DataFrame是Series的容器，Panel是DataFrame的容器。



构建和处理两个或更多个维数组是一项繁琐的任务，用户在编写函数时要考虑数据集的方向。 但是使用*Pandas*数据结构，减少了用户的思考。

例如，使用表格数据(DataFrame)，在语义上更有用于考虑索引(行)和列，而不是轴0和轴1。

**可变性**

所有*Pandas*数据结构是值可变的(可以更改)，除了系列都是大小可变的。系列是大小不变的。

注 - DataFrame被广泛使用，是最重要的数据结构之一。面板使用少得多。

# 2、数据结构之Series

系列(Series)是能够保存任何类型的数据(整数，字符串，浮点数，Python对象等)的一维标记数组。轴标签统称为索引。

*Pandas*系列可以使用以下构造函数创建 -

pandas.Series( data, index, dtype, copy)。

构造函数的参数如下 -



可以使用各种输入创建一个系列，如 -

* 数组
* 字典
* 标量值或常数

## 2.1 从ndarray创建一个系列

如果数据是ndarray，则传递的索引必须具有相同的长度。 如果没有传递索引值，那么默认的索引将是范围(n)，其中n是数组长度，即[0,1,2,3…. range(len(array))-1] - 1]。

**示例**

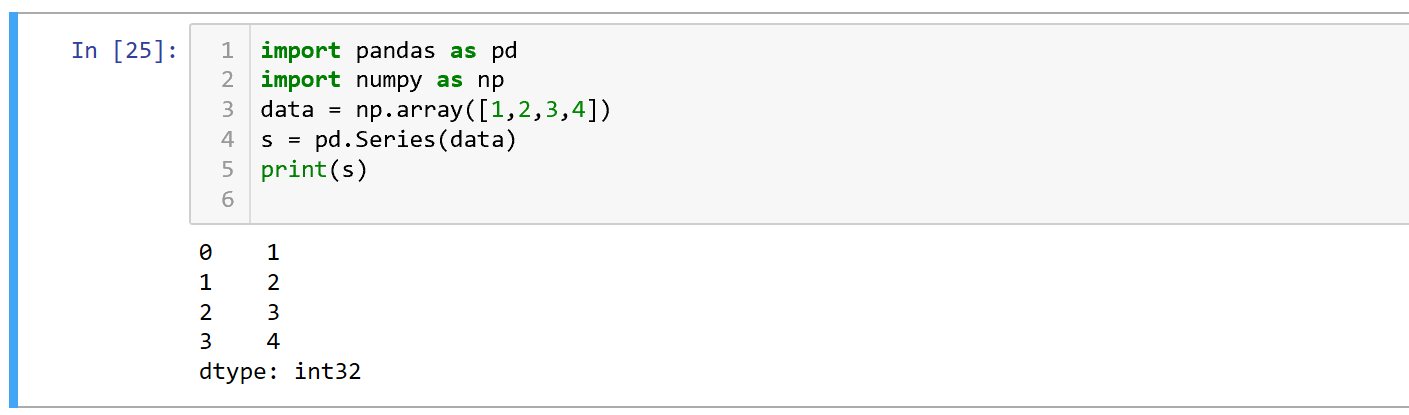
import pandas as pd

import numpy as np

data = np.array([1,2,3,4])

s = pd.Series(data)

print(s)

这里没有传递任何索引，因此默认情况下，它分配了从0到len(data)-1的索引，即：0到3。

**示例**

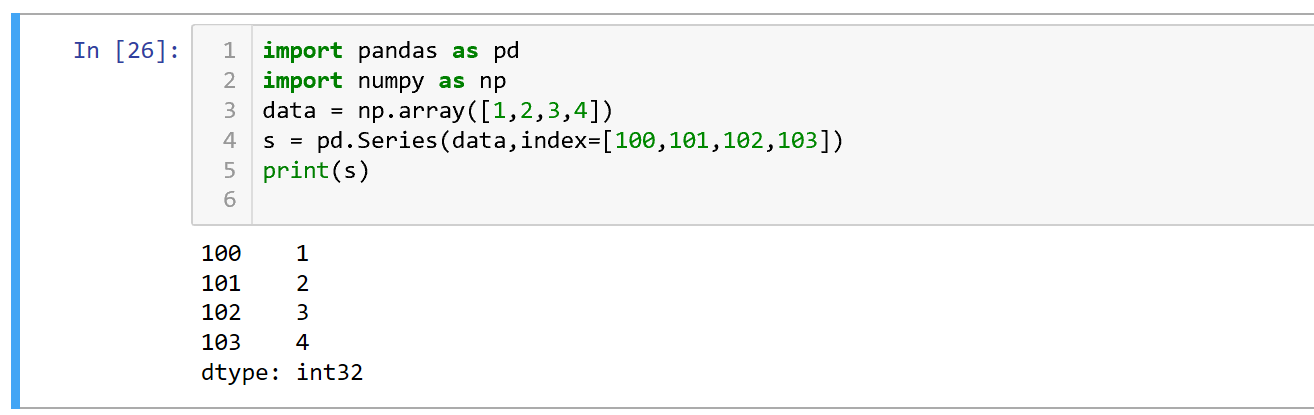
import pandas as pd

import numpy as np

data = np.array([1,2,3,4])

s = pd.Series(data,index=[100,101,102,103])

print(s)

在这里传递了索引值。现在可以在输出中看到自定义的索引值。

## 2.2 从字典创建一个系列

字典(dict)可以作为输入传递，如果没有指定索引，则按排序顺序取得字典键以构造索引。 如果传递了索引，索引中与标签对应的数据中的值将被拉出。

**示例**

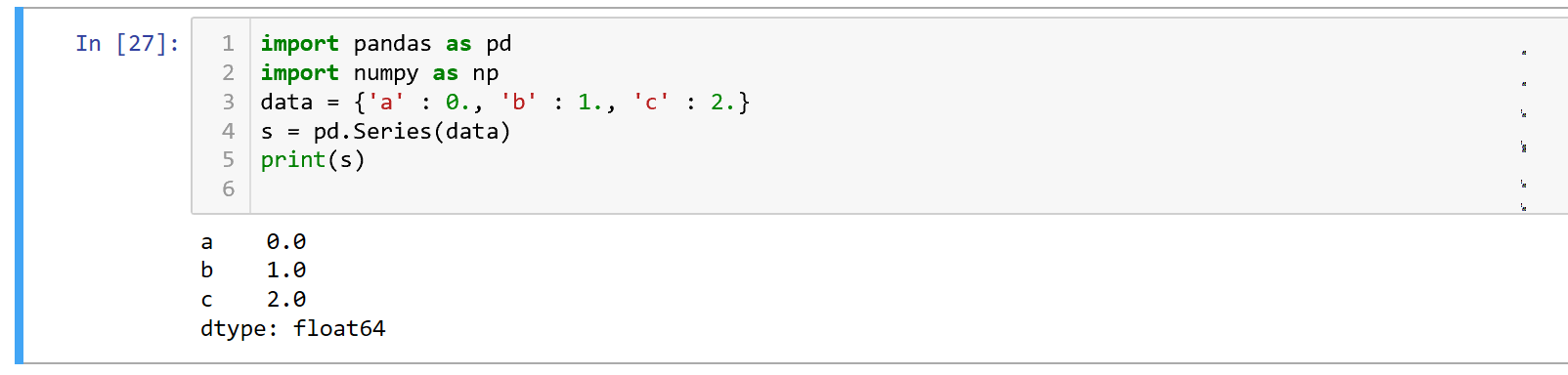
import pandas as pd

import numpy as np

data = {'a' : 0., 'b' : 1., 'c' : 2.}

s = pd.Series(data)

print(s)



**示例**

import pandas as pd

import numpy as np

data = {'a' : 0., 'b' : 1., 'c' : 2.}

s = pd.Series(data,index=['b','c','d','a'])

print(s)

执行上面示例代码，输出结果如下 -

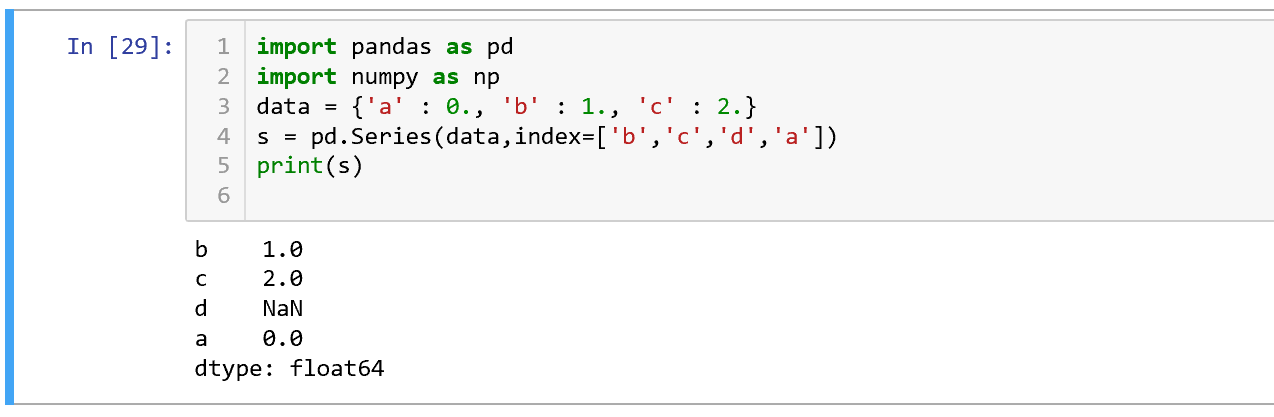
b 1.0

c 2.0

d NaN

a 0.0

dtype: float64

注意观察 - 索引顺序保持不变，缺少的元素使用NaN(不是数字)填充。

## 2.3从标量创建一个系列

系列中的数据可以使用类似于访问ndarray中的数据来访问。

**示例**

import pandas as pd

s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

print(s)

print('retrieve the first value')

print(s[0])

print('retrieve the first three elements')

print(s[:3])

print('retrieve the value with index')

print(s['c'])

print('retrieve elements with index')

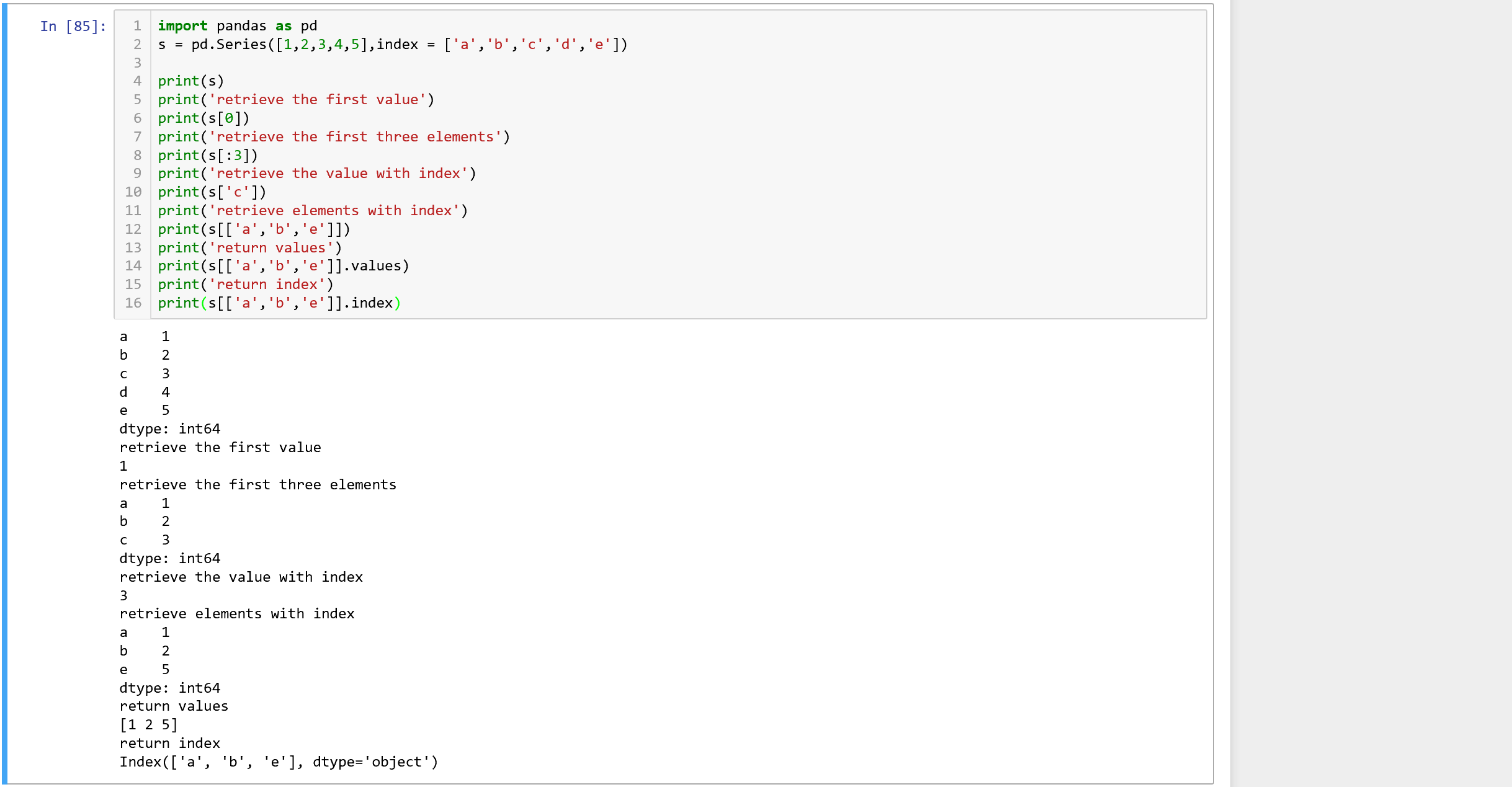
print(s[['a','b','e']])

print('return values')

print(s[['a','b','e']].values)

print('return index')

print(s[['a','b','e']].index)



# 3、数据结构之Dataframe

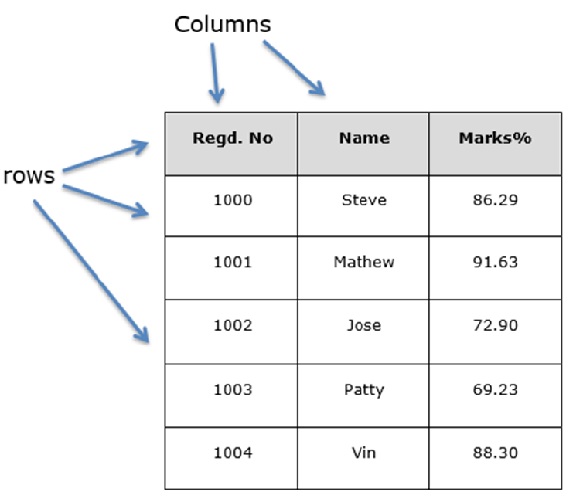
数据帧(DataFrame)是二维数据结构，即数据以行和列的表格方式排列。

数据帧(DataFrame)的功能特点：

* 潜在的列是不同的类型
* 大小可变
* 标记轴(行和列)
* 可以对行和列执行算术运算

**结构体**

假设要创建一个包含学生数据的数据帧。参考以下图示 -



可以将上图表视为SQL表或电子表格数据表示。

**pandas.DataFrame**

pandas中的DataFrame可以使用以下构造函数创建 -

pandas.DataFrame( data, index, columns, dtype, copy)



## 3.1 从列表创建DataFrame

可以使用单个列表或列表的列表创建数据帧(DataFrame)。

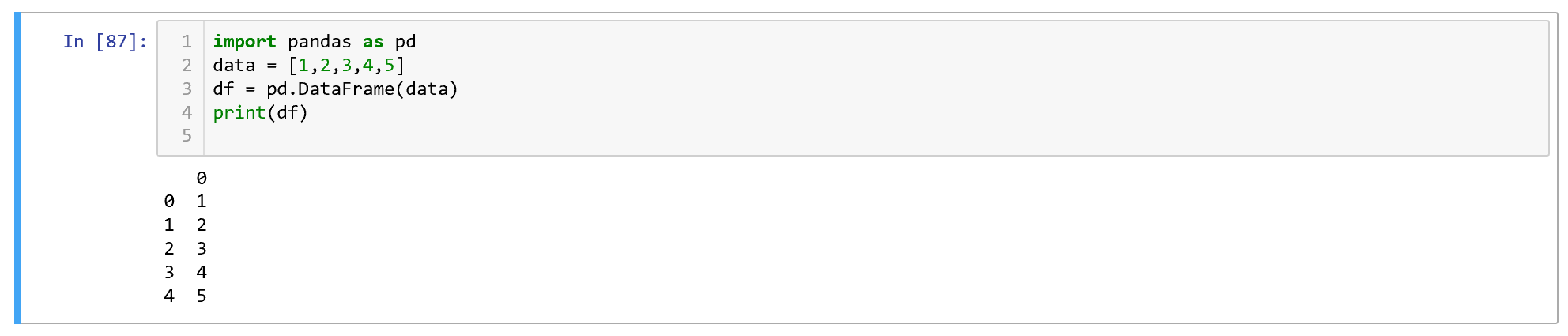
**实例**

import pandas as pd

data = [1,2,3,4,5]

df = pd.DataFrame(data)

print(df)



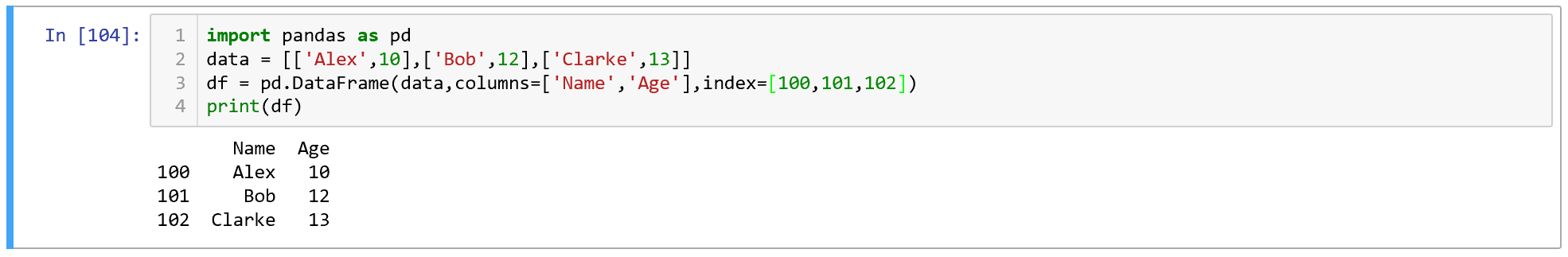
**实例**

import pandas as pd

data = [['Alex',10],['Bob',12],['Clarke',13]]

df = pd.DataFrame(data,columns=['Name','Age'],index=[100,101,102])

print(df)



**实例**

import pandas as pd

data = [['Alex',10],['Bob',12],['Clarke',13]]

df = pd.DataFrame(data,columns=['Name','Age'],index=[100,101,102],dtype=float)

print(df)

print(df.index)

print(df.columns)

注意 - 可以观察到，dtype参数将Age列的类型更改为浮点。

## 3.2 从ndarrays/Lists的字典来创建DataFrame

所有的ndarrays必须具有相同的长度。如果传递了索引(index)，则索引的长度应等于数组的长度。

如果没有传递索引，则默认情况下，索引将为range(n)，其中n为数组长度。

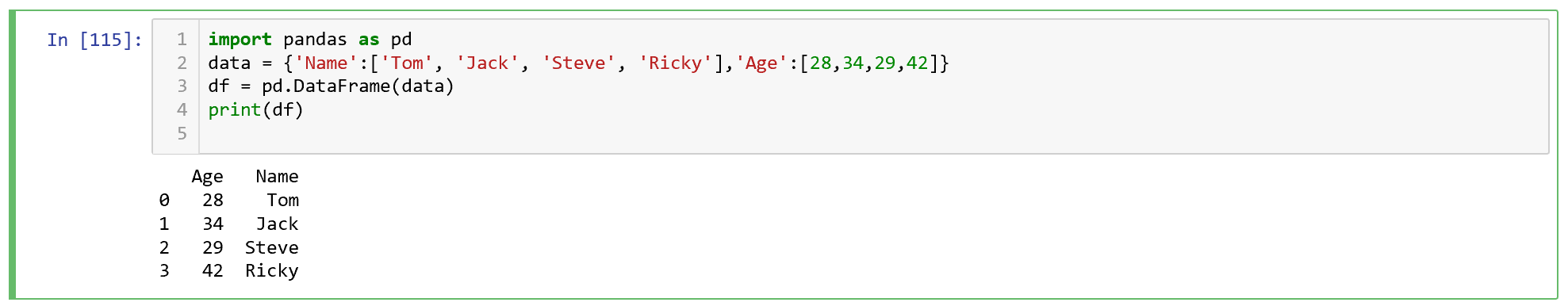
**实例**

import pandas as pd

data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}

df = pd.DataFrame(data)

print(df)

注 - 观察值0,1,2,3。它们是分配给每个使用函数range(n)的默认索引。

**示例**

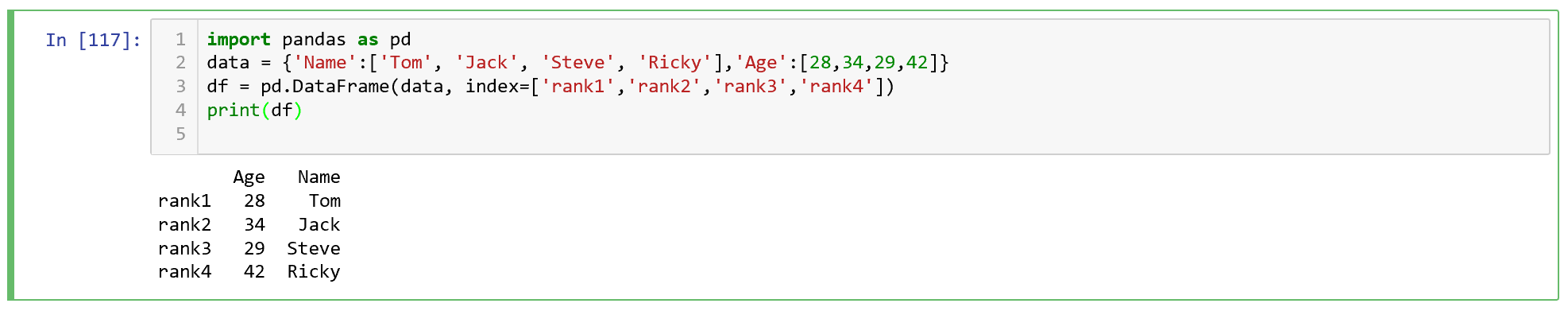
使用数组创建一个索引的数据帧(*DataFrame*)。

import pandas as pd

data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}

df = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])

print(df)



## 3.3 从列表创建数据帧DataFrame

字典列表可作为输入数据传递以用来创建数据帧(*DataFrame*)，字典键默认为列名。

**实例**

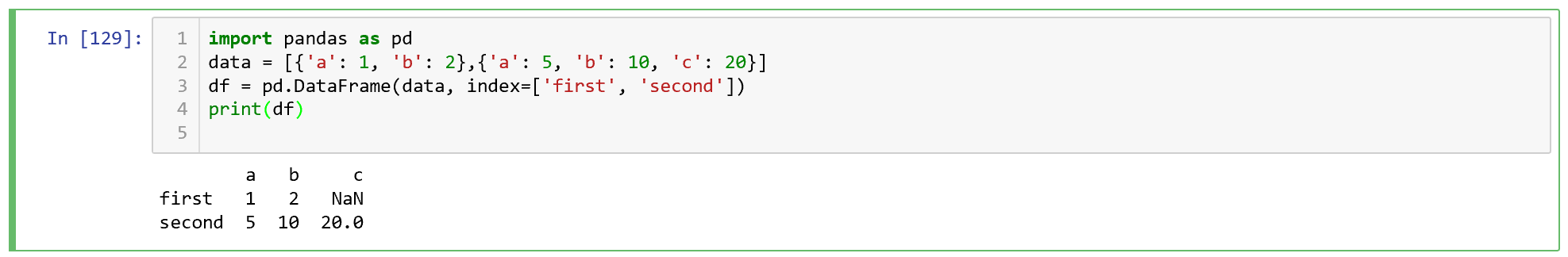
以下示例显示如何通过传递字典列表来创建数据帧(*DataFrame*)。

import pandas as pd

data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]

df = pd.DataFrame(data, index=['first', 'second'])

print(df)



注意 - 观察到，NaN(不是数字)被附加在缺失的区域。

**实例**

以下示例显示如何使用字典，行索引和列索引列表创建数据帧(*DataFrame*)。

import pandas as pd

data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]

#With two column indices, values same as dictionary keys

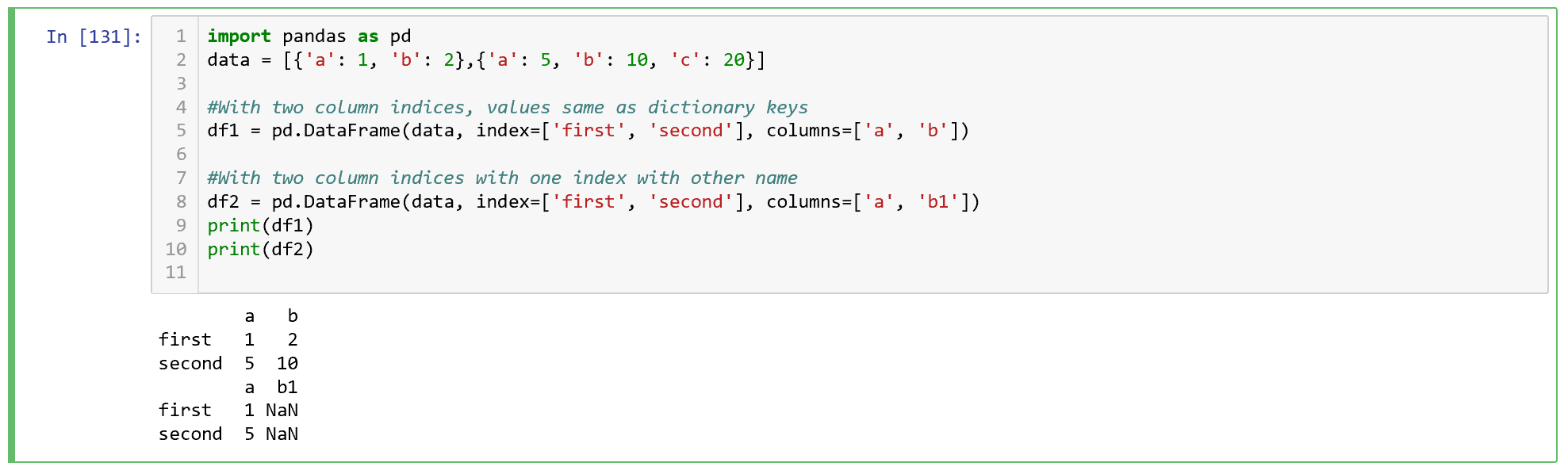
df1 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b'])

#With two column indices with one index with other name

df2 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b1'])

print(df1)

print(df2)

注意 - 观察，df2使用字典键以外的列索引创建DataFrame; 因此，附加了NaN到位置上。

## 3.4 从系列的字典来创建DataFrame

字典的系列可以传递以形成一个DataFrame。 所得到的索引是通过的所有系列索引的并集。

**示例**

import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),

'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)

print df



注意 - 对于第一个系列，观察到没有传递标签'd'，但在结果中，对于d标签，附加了NaN。

现在通过实例来了解列选择，添加和删除。

# 4、Dataframe增删改查操作

## 4.1 列选择

下面将通过从数据帧(DataFrame)中选择一列。

**示例**

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),

'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd']),

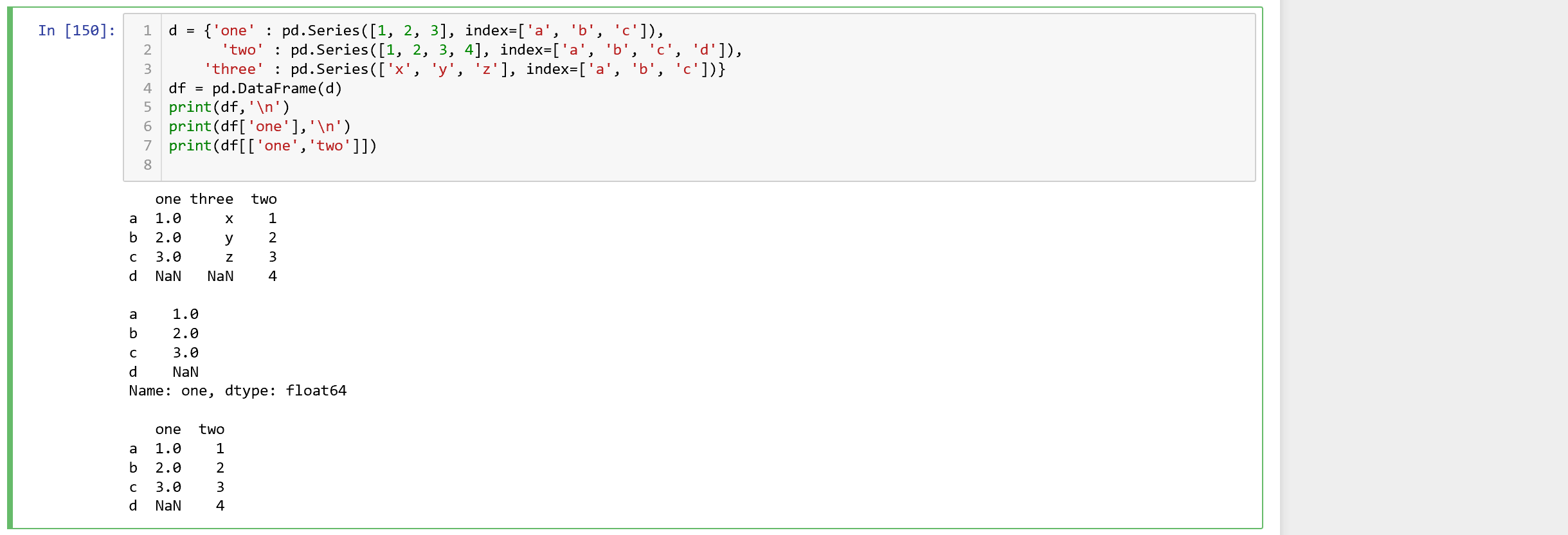
'three' : pd.Series(['x', 'y', 'z'], index=['a', 'b', 'c'])}

df = pd.DataFrame(d)

print(df,'\n')

print(df['one'],'\n')

print(df[['one','two']])



## 4.2列添加

下面将通过向现有数据框添加一个新列来理解这一点。

**示例**

import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),

'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)

# Adding a new column to an existing DataFrame object with column label by passing new series

print ("Adding a new column by passing as Series:")

df['three']=pd.Series([10,20,30],index=['a','b','c'])

print(df)

print ("Adding a new column using the existing columns in DataFrame:")

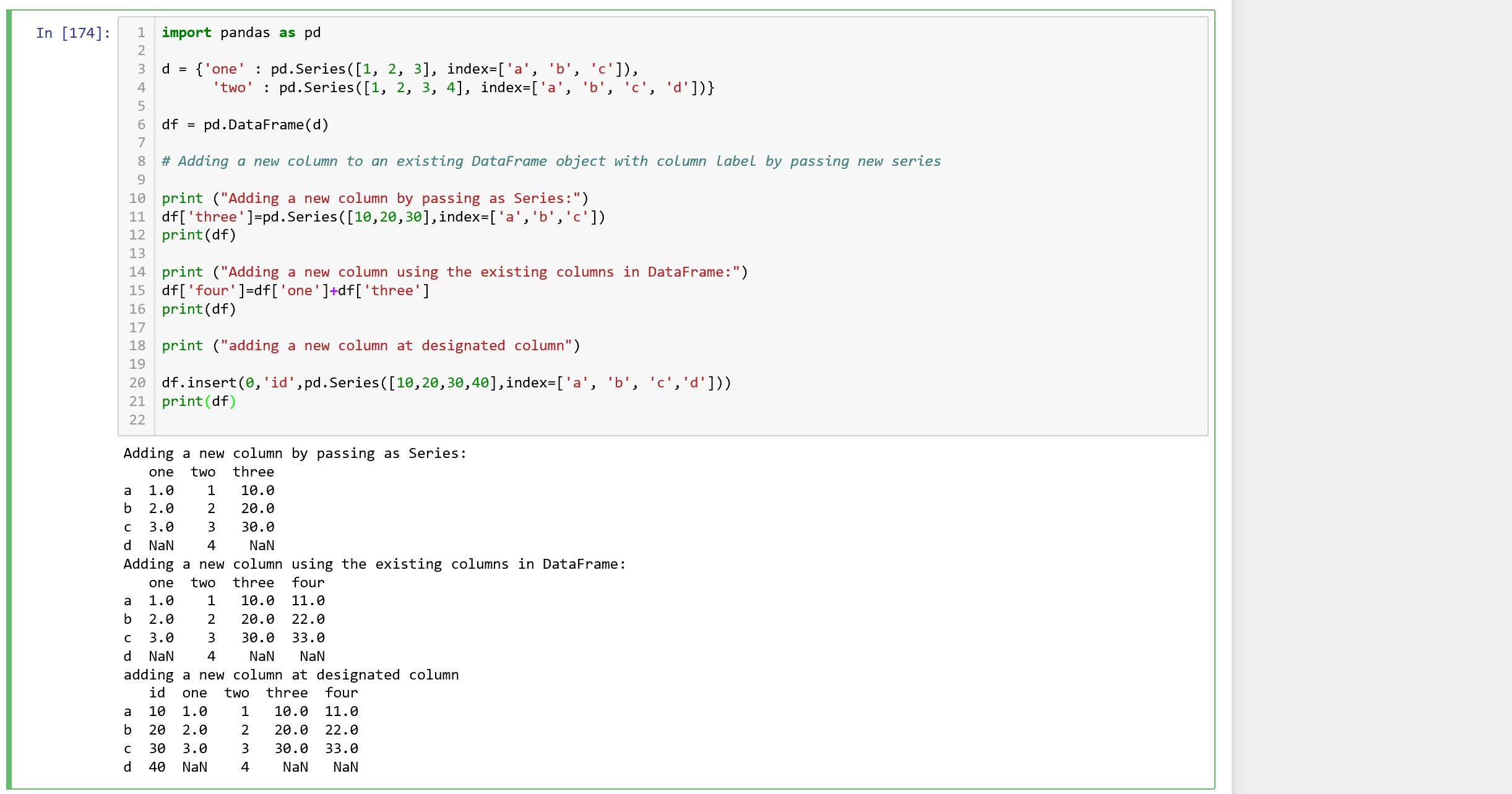
df['four']=df['one']+df['three']

print(df)

print ("adding a new column at designated column")

df.insert(0,'id',pd.Series([10,20,30,40],index=['a', 'b', 'c','d']))

print(df)



## 4.3列删除

列可以删除或弹出; 看看下面的例子来了解一下。

**例子**

import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),

'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd']),

'three' : pd.Series([10,20,30], index=['a','b','c'])}

df = pd.DataFrame(d)

print ("Our dataframe is:")

print(df)

# using del function

print ("Deleting the first column using DEL function:")

del df['one']

print(df)

# using pop function

print ("Deleting another column using POP function:")

df.pop('two')

print(df)



## 4.4列改名

**例子**

import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),

'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd']),

'three' : pd.Series([10,20,30], index=['a','b','c'])}

df = pd.DataFrame(d)

print ("Our dataframe is:")

print(df,'\n')

# change with

df.rename(columns={'one':'A','two':'B'})

print(df,'\n')

# using pop function

df.rename(columns={'one':'A','two':'B'},inplace = True)

print(df)





## 4.5 行选择

**标签选择**

可以通过将行标签传递给loc()函数来选择行。参考以下示例代码 -

import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),

'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)

print(df.loc['b'])



结果是一系列标签作为DataFrame的列名称。 而且，系列的名称是检索的标签。

**按整数位置选择**

可以通过将整数位置传递给iloc()函数来选择行。参考以下示例代码 -

import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),

'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)

print(df.iloc[2])



## 4.6 行切片

可以使用:运算符选择多行。参考以下示例代码 -

import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),

'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)

print(df[2:4])



## **4.7 添加行**

使用append()函数将新行添加到DataFrame。 此功能将附加行结束。

import pandas as pd

df = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])

df2 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a','b'])

df = df.append(df2)

print(df,'\n')

df = df.append([{'a':1,'b':1}])

print(df)



## **4.8 删除行**

使用索引标签从DataFrame中删除或删除行。 如果标签重复，则会删除多行。

如果有注意，在上述示例中，有标签是重复的。这里再多放一个标签，看看有多少行被删除。

import pandas as pd

df = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])

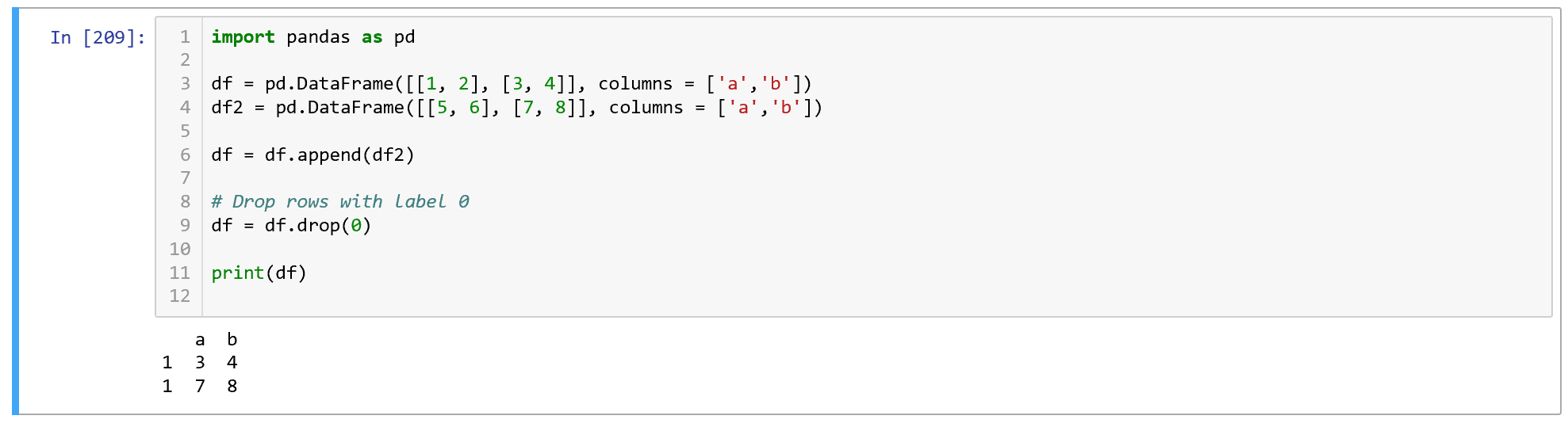
df2 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a','b'])

df = df.append(df2)

# Drop rows with label 0

df = df.drop(0)

print(df)



## **4.8 行改名**

1、采用rename()方法。看看下面的例子。

import pandas as pd

import numpy as np

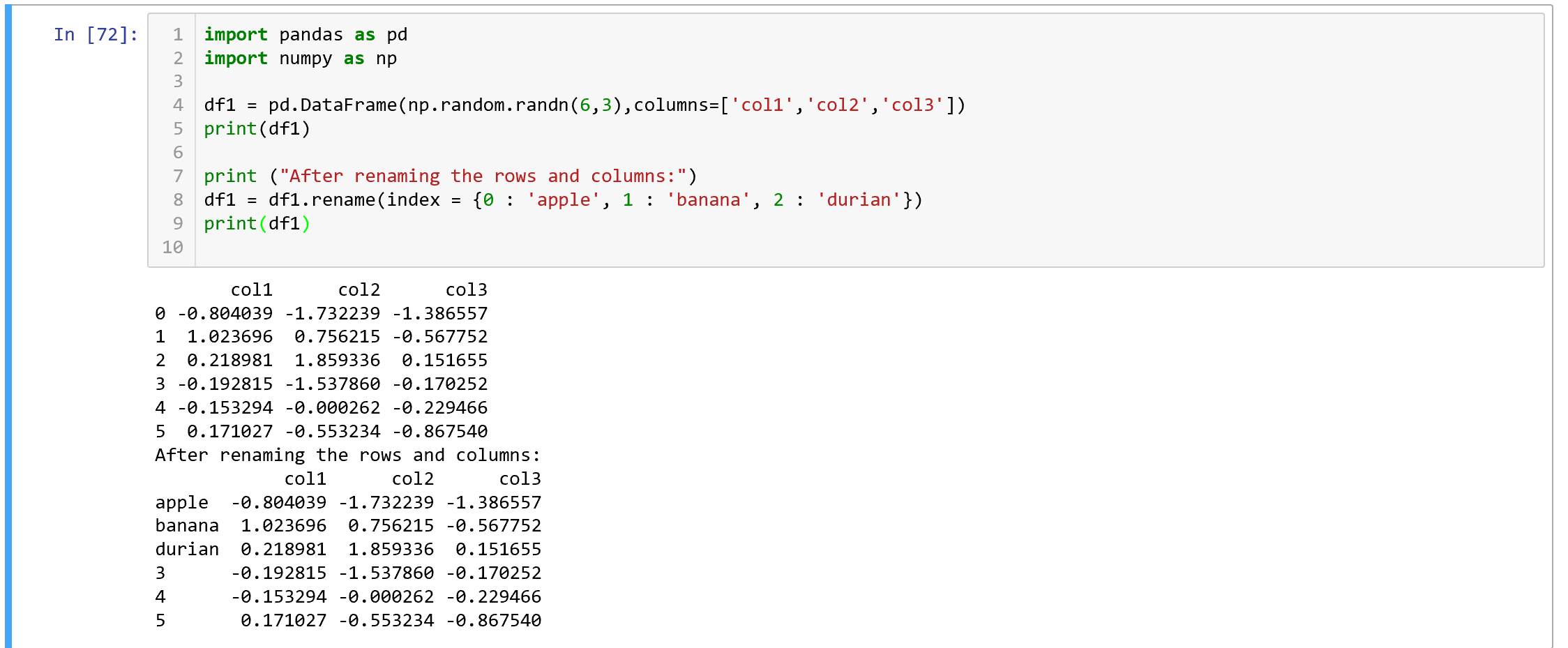
df1 = pd.DataFrame(np.random.randn(6,3),columns=['col1','col2','col3'])

print(df1)

print ("After renaming the rows and columns:")

df1 = df1.rename(index = {0 : 'apple', 1 : 'banana', 2 : 'durian'})

print(df1)



2、重置索引的方法

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

df2 = pd.DataFrame(np.random.randn(3,3),columns=['col1','col2','col3'])

df = df.append(df2)

print(df,'\n')

df\_reindexed = df.reset\_index()

print(df\_reindexed[['col1','col2','col3']])



## **4.9选择与修改元素**

直接使用索引，或者loc定位都可以

df = pd.DataFrame([[1, 2, 3], [3, 4, 5]], columns = ['a','b','c'])

df2 = pd.DataFrame([[5, 6, 7], [7, 8, 9]], columns = ['a','b','c'])

df = df.append(df2)

print(df,'\n')

# 直接用索引选 先列再行

print(df['a'][0])

print('\n')

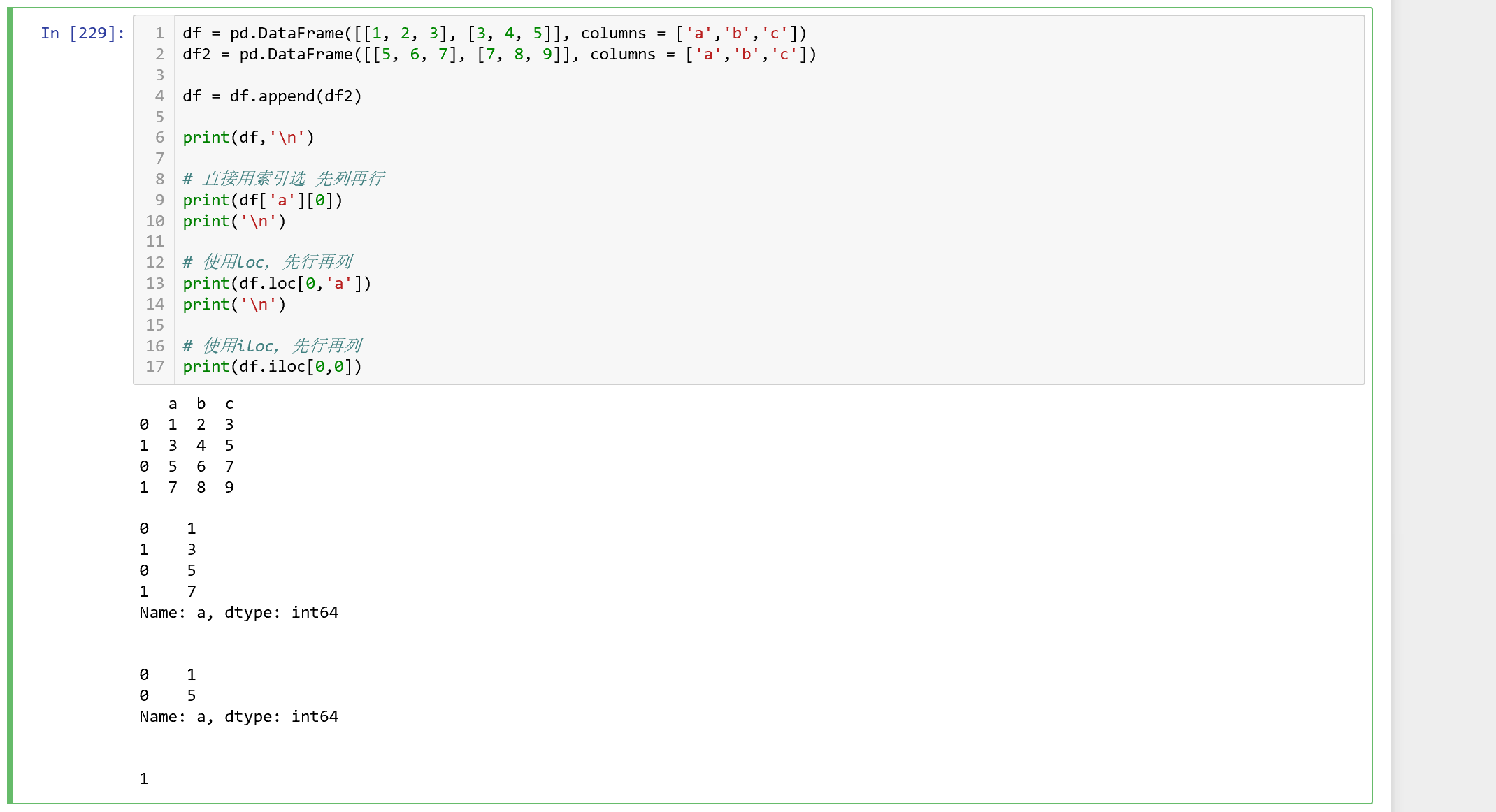
# 使用loc，先行再列

print(df.loc[0,'a'])

print('\n')

# 使用iloc，先行再列

print(df.iloc[0,0])



# 5、DataFrame基本功能

下面来看看数据帧(DataFrame)的基本功能有哪些？下表列出了DataFrame基本功能的重要属性或方法。



3轴(axis)这个名称旨在给出描述涉及面板数据的操作的一些语义。它们是 -

* *items* - axis 0，每个项目对应于内部包含的数据帧(DataFrame)。
* *major\_axis* - axis 1，它是每个数据帧(DataFrame)的索引(行)。
* *minor\_axis* - axis 2，它是每个数据帧(DataFrame)的列。

## **5.1 转置**

import pandas as pd

import numpy as np

# Create a Dictionary of series

d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Minsu','Jack']),

'Age':pd.Series([25,26,25,23,30,29,23]),

'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

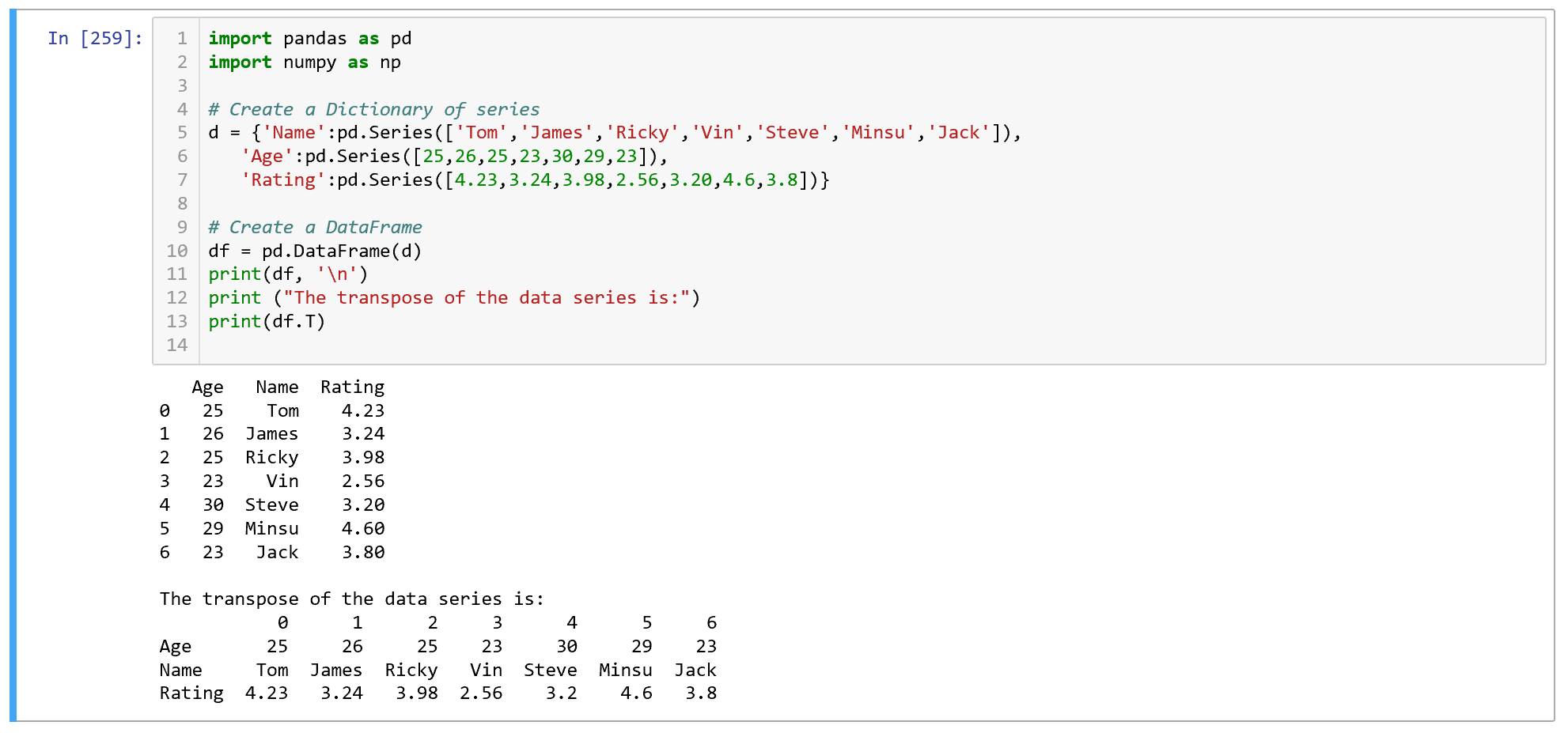
# Create a DataFrame

df = pd.DataFrame(d)

print(df, '\n')

print ("The transpose of the data series is:")

print(df.T)



## **5.2 axes示例**

import pandas as pd

import numpy as np

# Create a Dictionary of series

d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Minsu','Jack']),

'Age':pd.Series([25,26,25,23,30,29,23]),

'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

# Create a DataFrame

df = pd.DataFrame(d)

print(df, '\n')

print ("The transpose of the data series is:")

print(df.axes)



## **5.3 其他属性**

import pandas as pd

import numpy as np

# Create a Dictionary of series

d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Minsu','Jack']),

'Age':pd.Series([25,26,25,23,30,29,23]),

'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

# Create a DataFrame

df = pd.DataFrame(d)

print(df, '\n')

print ("The transpose of the data series is:")

print(df.shape)

print(df.size)

print(df.values)

print(df.head(2))

print(df.tail(2))



# 6、Pandas描述性统计

有很多方法用来集体计算DataFrame的描述性统计信息和其他相关操作。 其中大多数是sum()，mean()等聚合函数，但其中一些，如cumsum()，产生一个相同大小的对象。 一般来说，这些方法采用轴参数，就像ndarray.{sum，std，...}，但轴可以通过名称或整数来指定：

* *数据帧(DataFrame)* - “index”(axis=0，默认)，columns(axis=1)

## **6.1 统计函数：以sum()方法为例**



import pandas as pd

import numpy as np

#Create a Dictionary of series

d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Minsu','Jack',

'Lee','David','Gasper','Betina','Andres']),

'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),

'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

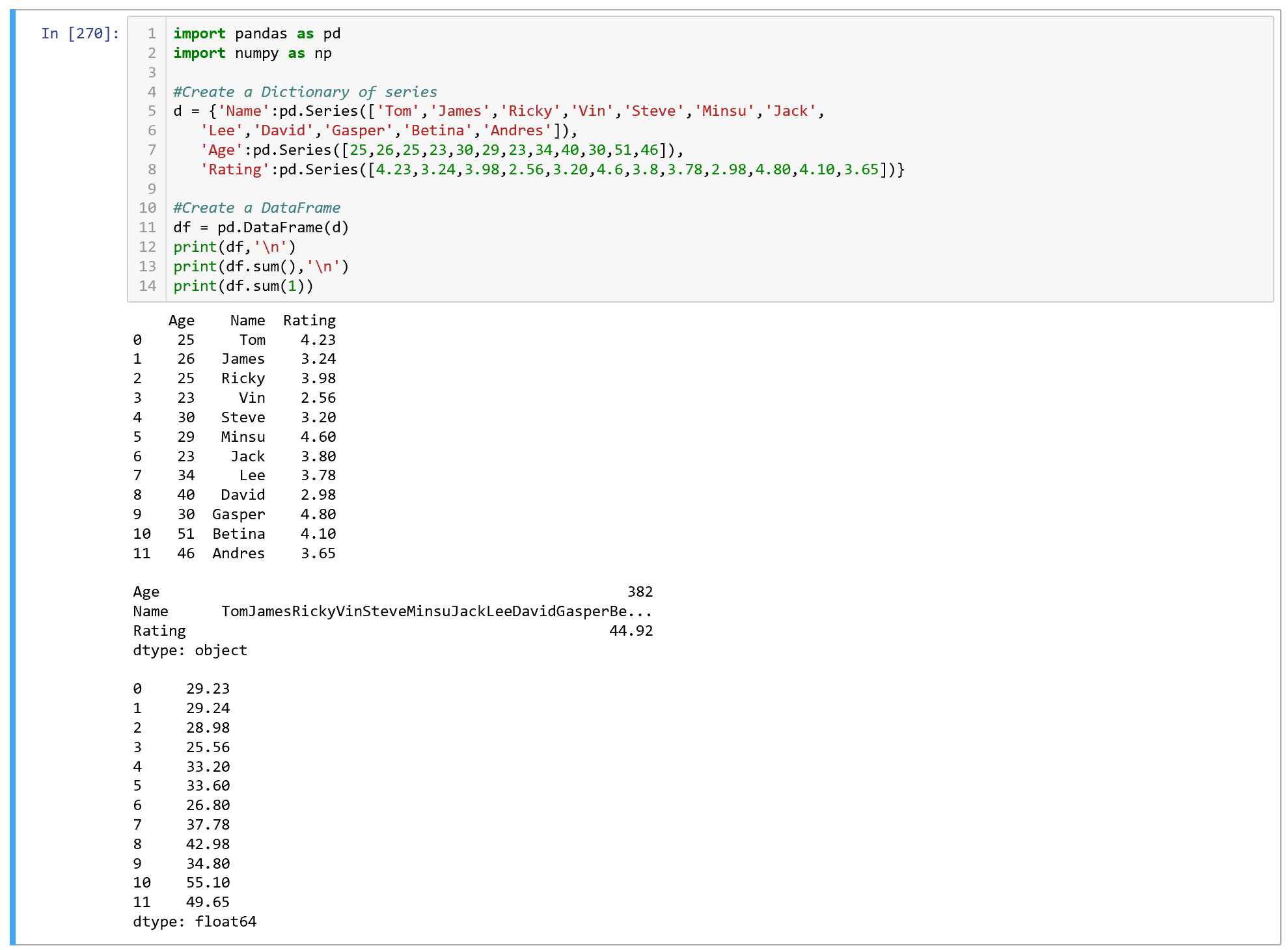
#Create a DataFrame

df = pd.DataFrame(d)

print(df,'\n')

print(df.sum(),'\n')

print(df.sum(1))



## **6.2 汇总数据**

describe()函数是用来计算有关DataFrame列的统计信息的摘要。

import pandas as pd

import numpy as np

#Create a Dictionary of series

d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Minsu','Jack',

'Lee','David','Gasper','Betina','Andres']),

'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),

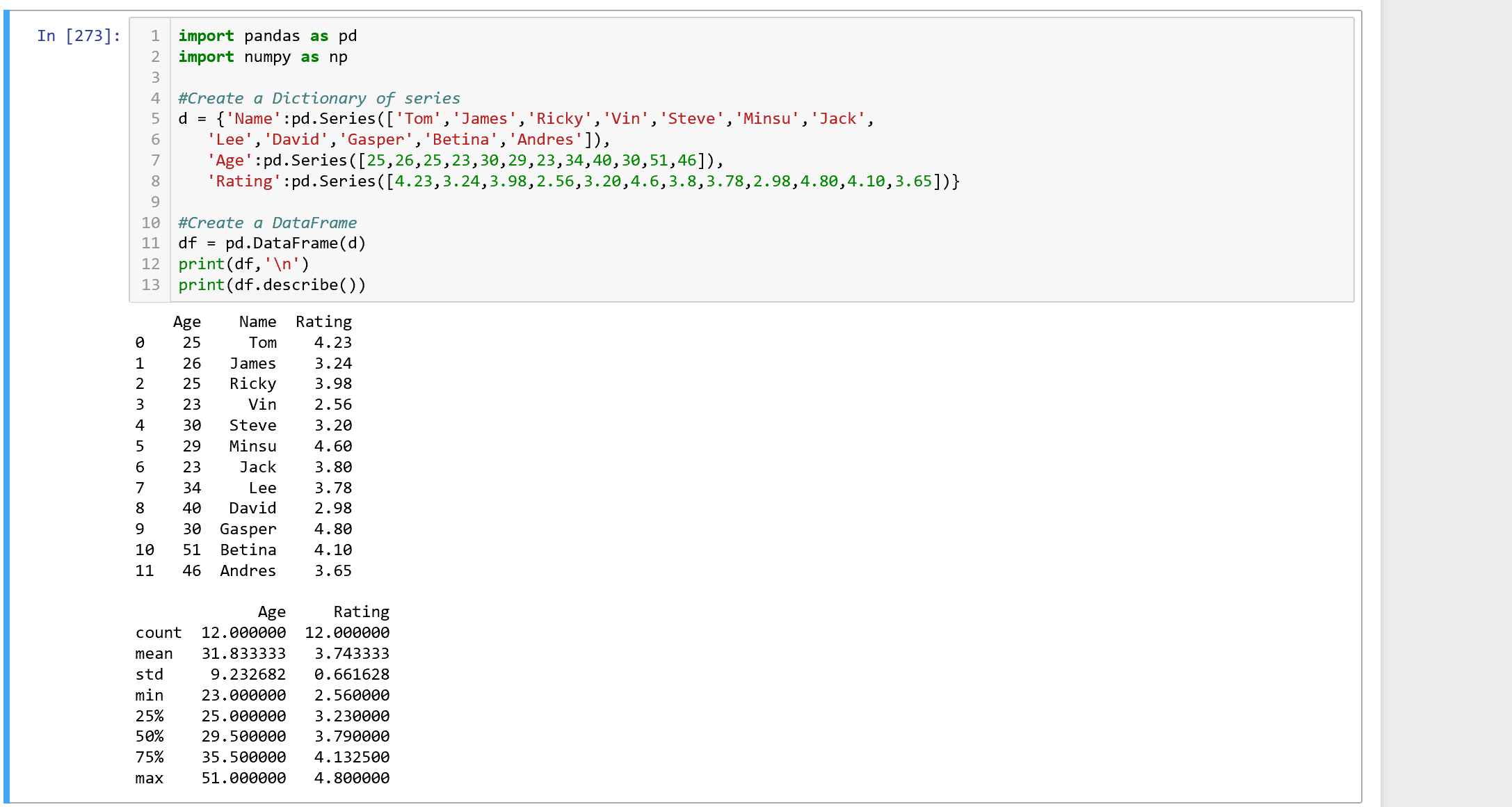
'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

#Create a DataFrame

df = pd.DataFrame(d)

print(df,'\n')

print(df.describe())



# 7、Pandas面板（Panel）

面板(Panel)是3D容器的数据。面板数据一词来源于计量经济学，部分源于名称：Pandas - pan(el)-da(ta)-s。

3轴(axis)这个名称旨在给出描述涉及面板数据的操作的一些语义。它们是 -

* *items* - axis 0，每个项目对应于内部包含的数据帧(DataFrame)。
* *major\_axis* - axis 1，它是每个数据帧(DataFrame)的索引(行)。
* *minor\_axis* - axis 2，它是每个数据帧(DataFrame)的列。

**pandas.Panel()**

可以使用以下构造函数创建面板 -

pandas.Panel(data, items, major\_axis, minor\_axis, dtype, copy)

构造函数的参数如下 –



## 7.1 从3D ndarray创建

输入jupyter notebook，通过网址http://localhost:8888/tree访问。

# creating an empty panel

import pandas as pd

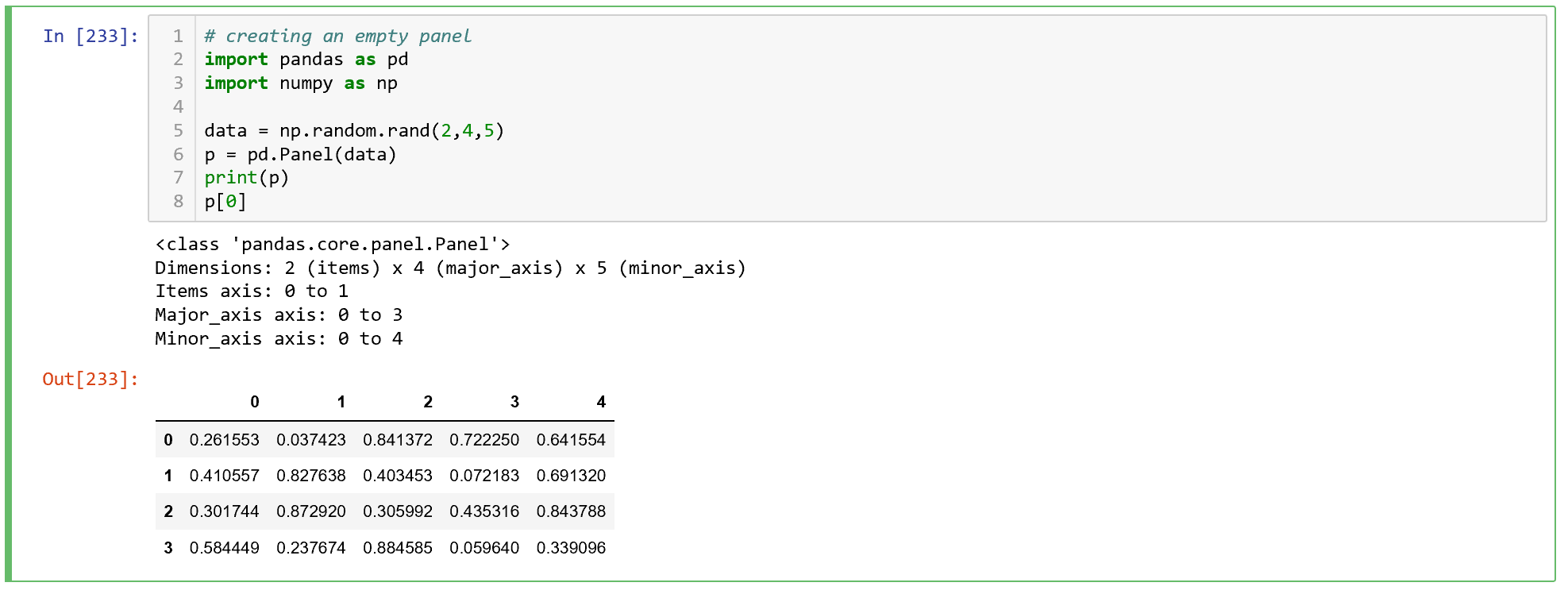
import numpy as np

data = np.random.rand(2,4,5)

p = pd.Panel(data)

print(p)

p[0]



## 7.2从DataFrame对象的dict创建面板

#creating an empty panel

import pandas as pd

import numpy as np

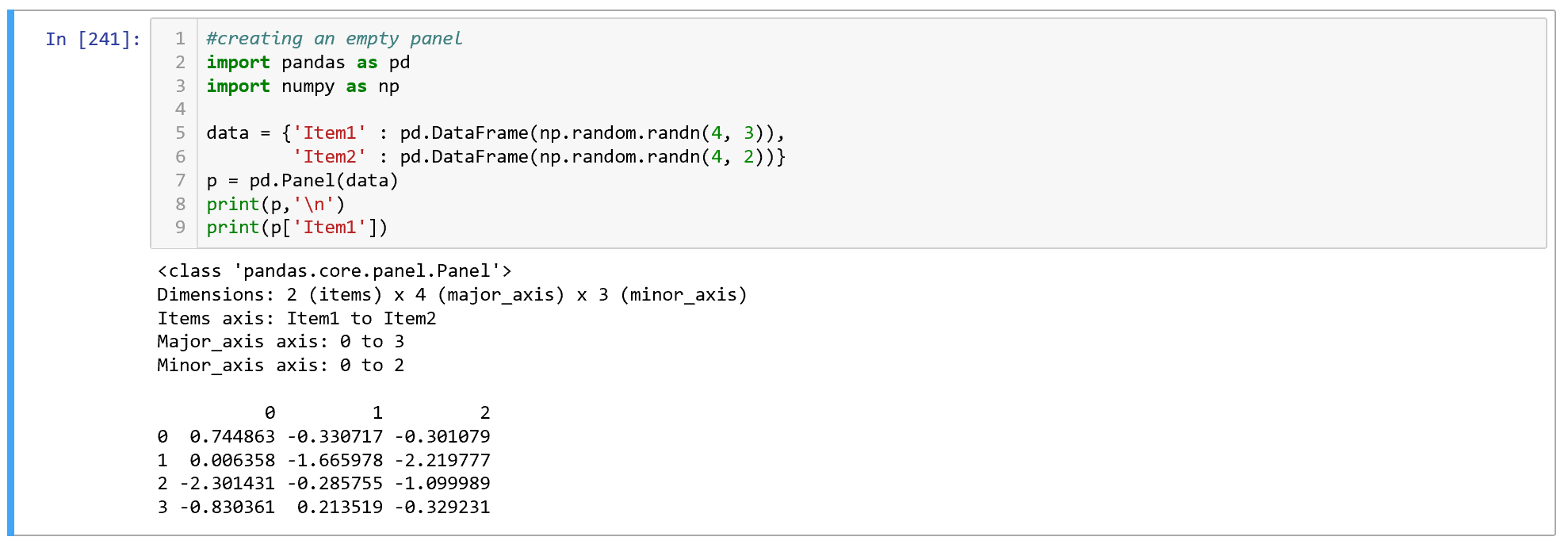
data = {'Item1' : pd.DataFrame(np.random.randn(4, 3)),

'Item2' : pd.DataFrame(np.random.randn(4, 2))}

p = pd.Panel(data)

print(p,'\n')

print(p['Item1'])



## 7.3 使用axis访问数据

# creating an empty panel

import pandas as pd

import numpy as np

data = {'Item1' : pd.DataFrame(np.random.randn(4, 3)),

'Item2' : pd.DataFrame(np.random.randn(4, 2))}

p = pd.Panel(data)

print(p,'\n')

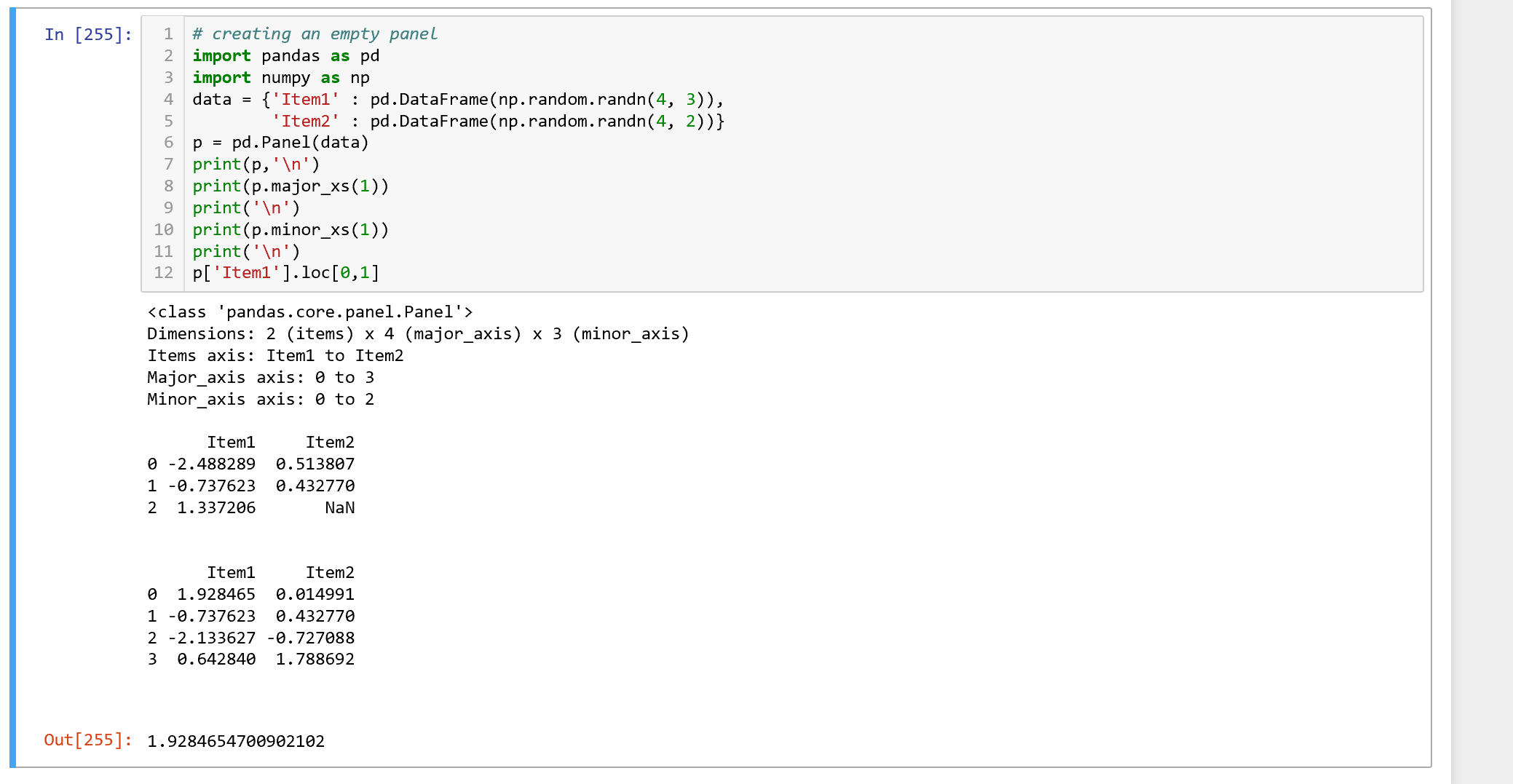
print(p.major\_xs(1))

print('\n')

print(p.minor\_xs(1))

print('\n')

p['Item1'].loc[0,1]



# 练习：

第一题和第二题采用学生数据集：

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. 查询Name,Height,Weight列的前五行；查询第1，3，5，6，8行信息。
2. 查询所有12岁以上的女生姓名、身高、体重；删除1，3，5，6行数据。
3. 随机生成均值为3，方差为4，大小100的正态分布，输出它的描述性统计指标。（提示：numpy.random.normal()）