**一、准备工作**

import numpy as np

import pandas as pd

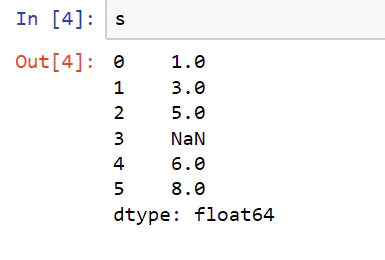
在进行下面的题目操作时，一定要先导入上面的两个数据分析包**pandas、numpy**

**二、生成对象**

**1. 如何用Python的列表创建一个series?**

s = pd.Series([1, 3, 5, np.nan, 6, 8])

**输出：**



一个series是一个一维的标记数组，可以容纳任何数据类型(整数、字符串、浮点数、Python对象等)。必须记住，与Python列表不同，一个series总是包含相同类型的数据。

**2.如何使用列表创建一个DataFrame?**

# 导入pandas

import pandas as pd

# 字符串列表

lst = ["Gamer47","Shox","Simple"]

# 在列表中调用DataFrame构造函数

df = pd.DataFrame(lst)

**输出：**

|  |
| --- |
|  |

**3.如何使用Series 字典对象生成 DataFrame?**

df2 = pd.DataFrame({'A': 1.,

'B': pd.Timestamp('20130102'),

'C': pd.Series(1, index=list(range(4)), dtype='float32'),

'D': np.array([3] \* 4, dtype='int32'),

'E': pd.Categorical(["test", "train", "test", "train"]),

'F': 'foo'})

**输出：**

|  |
| --- |
|  |

**4.如何在pandas中创建一个空的DataFrame？**  
要创建一个完全空的pandas dataframe，我们使用以下操作：

import pandas as pd

df = pd.DataFrame()

**输出：**

|  |
| --- |
|  |

**三.查看数据:**

已知有这样的数据，如何进行查看

dates = pd.date\_range('20130101', periods=6)

df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))

**输出：**

|  |
| --- |
|  |

**1. 如何查看头部数据? df.head()**

|  |
| --- |
|  |

**2.如何查看尾部数据? df.tail()**

|  |
| --- |
|  |

**3.如何快速查看数据的统计摘要? df.describe()**

|  |
| --- |
|  |

**4.如何查询索引和列名? 获取全部索引df.index,获取全部列名df.columns**

**四.索引**

**1.简述Pandas Index**  
在panda中建立索引意味着简单地从DataFrame中选择特定的数据行和列。  
pandas支持四种类型的多轴索引，它们是:

* Dataframe.[ ] 此函数称为索引运算符
* Dataframe.loc[ ] : 此函数用于标签
* Dataframe.iloc[ ] : 此函数用于基于位置或整数的
* Dataframe.ix[] : 此函数用于基于标签和整数的

它们统称为索引器。这些是迄今为止索引数据最常见的方法。这四个函数有助于从DataFrame获取元素、行和列。

**2.Pandas 定义重新索引(Reindexing)**  
重新索引会更改DataFrame的行标签和列标签。重新索引意味着使数据符合特定轴上给定的一组标签。  
多个操作可以通过像这样的索引来完成:

* 重新排序现有数据以匹配一组新的标签。
* 在不存在标签数据的标签位置插入缺失值(NA)标记。

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()

})

df\_reindexed = df.reindex(index=[0,2,5], columns=['A', 'C', 'B'])

print (df\_reindexed)

**输出:**

|  |
| --- |
|  |

**3.如何设置索引？**

**panda set\_index()**是一种将列表、序列或dataframe设置为dataframe索引的方法。  
**语法:**  
**DataFrame.set\_index(keys, inplace=False)**  
**参数:**

* keys：列标签或列标签/数组列表，需要设置为索引的列
* inplace：默认为False，适当修改DataFrame(不要创建新对象)

**改变索引列**  
在本例中，名称列被用作DataFrame的索引列

import pandas as pd

#就是读取csv文本文件到DataFrame变量中

data = pd.read\_csv("employees.csv")

data.set\_index("First Name", inplace = True)

#观察数据

data.head()

**输出:**  
如输出图像所示，以前索引列是一系列数字  
**Before Operation** **–**

|  |
| --- |
|  |

data.set\_index("First Name", inplace = True)

#观察改变索引列后的数据

data.head()

**After Operation**

|  |
| --- |
|  |

**4.如何重置索引?**  
**Pandas Series.reset\_index()**  
函数的作用是:生成一个新的DataFrame或带有重置索引的Series。

**例1：** 使用 Series.reset\_index() 函数重置给定Series对象的索引

# 导入pandas包

import pandas as pd

# 创建 Series

sr = pd.Series([10, 25, 3, 11, 24, 6])

# 创建索引

index\_ = ['Coca Cola', 'Sprite', 'Coke', 'Fanta', 'Dew', 'ThumbsUp']

# 设置索引

sr.index = index\_

# 打印series

print(sr)

**输出:**

|  |
| --- |
|  |

现在，我们将使用Series.reset\_index()函数来重置给定的series对象的索引

# 重置索引

result = sr.reset\_index()

# 打印

print(result)

**输出 :**

|  |
| --- |
|  |

从输出中可以看到，该 Series.reset\_index() 函数已将给定Series对象的索引重置为默认值。它保留了索引，并将其转换为列。

**五、选择需要的数据**

**1.获取数据**

**1.1先创建数据:**

dates = pd.date\_range('20130101', periods=6)

df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))

|  |
| --- |
|  |

**1.2选择单列，产生 Series**

|  |
| --- |
|  |

**1.3用 [ ] 切片行：**

|  |
| --- |
|  |

**2.按标签选择**

详见[按标签选择](https://links.jianshu.com/go?to=https%3A%2F%2Fpandas.pydata.org%2Fpandas-docs%2Fstable%2Fuser_guide%2Findexing.html%23indexing-label)。

**2.1用标签提取一行数据：**

|  |
| --- |
|  |

**2.2用标签选择多列数据：**

|  |
| --- |
|  |

**2.3用标签切片，包含行与列结束点：**

|  |
| --- |
|  |

**2.4返回对象降维：**

|  |
| --- |
|  |

**2.5提取标量值：**

|  |
| --- |
|  |

**2.6快速访问标量，与上述方法等效：**

|  |
| --- |
|  |

**3.按位置选择**

详见[按位置选择](https://links.jianshu.com/go?to=https%3A%2F%2Fpandas.pydata.org%2Fpandas-docs%2Fstable%2Fuser_guide%2Findexing.html%23indexing-label)。

**3.1用整数位置选择：**

|  |
| --- |
|  |

**3.2类似 NumPy / Python，用整数切片：**

|  |
| --- |
|  |

**3.3 类似 NumPy / Python，用整数列表按位置切片：**

|  |
| --- |
|  |

**3.4显式整行切片：**

|  |
| --- |
|  |

**3.5 显式整列切片：**

|  |
| --- |
|  |

**3.6显式提取值：**

|  |
| --- |
|  |

**3.7 快速访问标量，与上述方法等效：**

|  |
| --- |
|  |

**4.布尔索引**

**4.1用单列的值选择数据：**

|  |
| --- |
|  |

**4.2选择 DataFrame 里满足条件的值：**

|  |
| --- |
|  |

**4.3 用** [**isin()**](https://links.jianshu.com/go?to=https%3A%2F%2Fpandas.pydata.org%2Fpandas-docs%2Fstable%2Freference%2Fapi%2Fpandas.Series.isin.html%23pandas.Series.isin) **筛选：**

|  |
| --- |
|  |

**5.赋值**

**5.1用索引自动对齐新增列的数据：**

|  |
| --- |
|  |

**5.2 按标签赋值：**

|  |
| --- |
|  |

**5.3按位置赋值：**

|  |
| --- |
|  |

**5.4按 NumPy 数组赋值：**

|  |
| --- |
|  |

**5.5上述赋值结果：**

|  |
| --- |
|  |

**5.6用 where 条件赋值：**

|  |
| --- |
|  |

**六.运算**

**1.如何得到一个数列的最小值、第25百分位、中值、第75位和最大值?**

import pandas as pd

import numpy as np

from numpy import percentile

p = pd.Series(np.random.normal(14, 6, 22))

state = np.random.RandomState(120)

p = pd.Series(state.normal(14, 6, 22))

print(percentile(p, q=[0, 25, 50, 75, 100]))

**输出:**

|  |
| --- |
|  |

**2.如何获得panda DataFrame中一个列的平均值?**

**Pandas dataframe.mean(axis=None)**函数返回所请求轴(axis=0代表对列进行求平均值，axis=1代表对行进行求平均值)的值的平均值。

**示例:**使用 mean() 函数查找索引轴上所有观测值的平均值。

import pandas as pd

df = pd.DataFrame({"A":[12, 4, 5, 44, 1],

"B":[5, 2, 54, 3, 2],

"C":[20, 16, 7, 3, 8],

"D":[14, 3, 17, 2, 6]})

df

**输出:**

|  |
| --- |
|  |

让我们使用datafame .mean()函数来查找索引轴上的平均值。

|  |
| --- |
|  |

**3.如何将函数应用到DataFrame中的每个数据元素?**  
可以使用 apply() 函数以便将函数应用于给定dataframe中的每一行。让我们来看看我们完成这项任务的方式。  
**实例:**

import pandas as pd

def add(a, b, c):

return a + b + c

def main():

data = {

'A':[1, 2, 3],

'B':[4, 5, 6],

'C':[7, 8, 9] }

df = pd.DataFrame(data)

print("Original DataFrame:\n", df)

df['add'] = df.apply(lambda row : add(row['A'],

row['B'], row['C']), axis = 1)

print('\nAfter Applying Function: ')

print(df)

if \_\_name\_\_ == '\_\_main\_\_':

main()

**输出：**

|  |
| --- |
|  |

**4.如何在panda中获得一个DataFrame的行数和列数?**

import numpy as np

raw\_data = {'name': ['Willard Morris', 'Al Jennings', 'Omar Mullins', 'Spencer McDaniel'],

'age': [20, 19, 22, 21],

'favorite\_color': ['blue', 'red', 'yellow', "green"],

'grade': [88, 92, 95, 70]}

df = pd.DataFrame(raw\_data, columns = ['name', 'age', 'favorite\_color', 'grade'])

**输出:**

|  |
| --- |
|  |

获取df的行和列计数

df.shape

**输出:**

|  |
| --- |
|  |

**5.如何在panda DataFrame中获得列值的总和?**  
**Pandas dataframe.sum()**函数返回所请求轴的值的和

**语法:** DataFrame.sum(axis=None, skipna=None, )  
**参数:**

* axis : {index (0), columns (1)},axis=0代表对列进行求和，axis=1代表对行进行求和
* skipna : 计算结果时排除NA /空值

**示例1:**使用 sum() 函数查找索引轴上所有值的总和

#导入数据

import pandas as pd

df = pd.read\_csv("NBAPlayers.txt",delimiter="\t")

现在求出沿索引轴的所有值的和。我们将跳过计算和时的NaN值。

# 查找索引轴的和，默认情况下，轴设置为0

df.sum(axis = 0, skipna = True)

**输出:**

|  |
| --- |
|  |

**数据链接:**  
[https://pan.baidu.com/s/1vAUxFlNizVTYH03j-O-VVg](https://links.jianshu.com/go?to=https%3A%2F%2Fpan.baidu.com%2Fs%2F1vAUxFlNizVTYH03j-O-VVg)  
提取码：ykbo

**七.合并:**

**如何将新行追加到pandas DataFrame？**  
**Pandas dataframe.append()**函数的作用是:将其他dataframe的行追加到给定的dataframe的末尾，返回一个新的dataframe对象。

**语法:**  
DataFrame.append( ignore\_index=False,)  
**参数:**

* ignore\_index : 如果为真，就不要使用索引标签

**示例1：** 创建两个数据框，然后将第二个附加到第一个。

importpandas as pd

# 使用dictionary创建第一个Dataframe

df1 =df =pd.DataFrame({"a":[1, 2, 3, 4],

"b":[5, 6, 7, 8]})

# 使用dictionary创建第二个Dataframe

df2 =pd.DataFrame({"a":[1, 2, 3],

"b":[5, 6, 7]})

现在将df2附加到df1的末尾

df1.append(df2)

**输出:**

|  |
| --- |
|  |

请注意，第二个DataFrame的索引值保留在附加的DataFrame中。如果我们不希望发生这种情况，则可以设置ignore\_index = True。

df.append(df2, ignore\_index = True)

**输出 :**

|  |
| --- |
|  |

**八.分组（Grouping）**

“group by” 指的是涵盖下列一项或多项步骤的处理流程：

* **分割**：按条件把数据分割成多组；
* **应用**：为每组单独应用函数；
* **组合**：将处理结果组合成一个数据结构。

详见[分组](https://links.jianshu.com/go?to=https%3A%2F%2Fpandas.pydata.org%2Fpandas-docs%2Fstable%2Fuser_guide%2Fgroupby.html%23groupby)。

df = pd.DataFrame({'A': ['foo', 'bar', 'foo', 'bar',

'foo', 'bar', 'foo', 'foo'],

'B': ['one', 'one', 'two', 'three',

'two', 'two', 'one', 'three'],

'C': np.random.randn(8),

'D': np.random.randn(8)})

df

**输出:**

|  |
| --- |
|  |

**1.先分组，再用** [**sum()**](https://links.jianshu.com/go?to=https%3A%2F%2Fpandas.pydata.org%2Fpandas-docs%2Fstable%2Freference%2Fapi%2Fpandas.DataFrame.sum.html%23pandas.DataFrame.sum)**函数计算每组的汇总数据：**

df.groupby('A').sum()

**输出:**

|  |
| --- |
|  |

**2.多列分组后，生成多层索引，也可以应用 sum 函数：**

df.groupby(['A', 'B']).sum()

**输出:**

|  |
| --- |
|  |

**九.重塑（Reshaping）**

**如何将numpy数组转换为给定形状的DataFrame?**

import pandas as pd

import numpy as np

p = pd.Series(np.random.randint(1, 7, 8))

**输出:**

|  |
| --- |
|  |

将series p重新组合成一个2行4列的dataframe

info = pd.DataFrame(p.values.reshape(2,4))

**输出:**

|  |
| --- |
|  |

**十.数据透视表**

透视表是一种可以对数据动态排布并且分类汇总的表格格式，在pandas中它被称作pivot\_table。  
**pivot\_table(data, values=None, index=None, columns=None)**  
**参数:**

* Index: 就是层次字段，要通过透视表获取什么信息就按照相应的顺序设置字段
* Values: 可以对需要的计算数据进行筛选
* Columns: 类似Index可以设置列层次字段，它不是一个必要参数，作为一种分割数据的可选方式。

详见:[数据透视表](https://links.jianshu.com/go?to=https%3A%2F%2Fpandas.pydata.org%2Fpandas-docs%2Fstable%2Fuser_guide%2Freshaping.html%23reshaping-pivot)

import pandas as pd

import numpy as np

df = pd.DataFrame({'A': ['one', 'one', 'two', 'three'] \* 3,

'B': ['A', 'B', 'C'] \* 4,

'C': ['foo', 'foo', 'foo', 'bar', 'bar', 'bar'] \* 2,

'D': np.random.randn(12),

'E': np.random.randn(12)})

**打印输出：**

|  |
| --- |
|  |

用上述数据生成数据透视表非常简单：

pd.pivot\_table(df, values='D', index=['A', 'B'], columns=['C'])

**返回结果:**

|  |
| --- |
|  |

**十一.panda DataFrame里的操作**

**1.如何将列添加到pandas DataFrame？**  
**源数据:**

#导入pandas

import pandas as pd

# 定义一个包含学生数据的字典

data = {"Name": ["Jai","Princi", "Gaurav" ,"Anuj"],

"Height": [5.1, 6.2, 5.1, 5.2],

"Qualification": ["Msc", "Ma","Msc", "Msc"]}

# 将字典转换成DataFrame

df = pd.DataFrame(data)

**输出:**

|  |
| --- |
|  |

**添加列:**

# 声明要转换为列的列表

address = ["Delhi", "Bangalore", "Chennai" ,"Patna"]

# 使用“address”作为列名

df["Address"] = address

# 观察结果

df

**输出：**

|  |
| --- |
|  |

**2.如何向panda DataFrame添加索引、行或列?**  
**向DataFrame添加索引**  
如果您创建了一个DataFrame, panda允许将输入添加到索引参数中。它将确保您拥有所需的索引。否则，在默认情况下，DataFrame包含一个数值索引，该索引从0开始，在DataFrame的最后一行结束。

**向DataFrame添加行、列**  
我们可以使用.loc、iloc和ix将行、列插入到DataFrame中。

**添加具有特定索引名的行:**

import pandas as pd

employees = pd.DataFrame(

data={'Name': ['John Doe', 'William Spark'],

'Occupation': ['Chemist', 'Statistician'],

'Date Of Join': ['2018-01-25', '2018-01-26'],

'Age': [23, 24]},

index=['Emp001', 'Emp002'],

columns=['Name', 'Occupation', 'Date Of Join', 'Age'])

print("\n------------ BEFORE ----------------\n")

print(employees)

employees.loc['Emp003'] = ['Sunny', 'Programmer', '2018-01-25', 45]

print("\n------------ AFTER ----------------\n")

print(employees)

**输出:**

|  |
| --- |
|  |

**3.如何在panda DataFrame上进行迭代?**  
您可以通过结合使用for循环和对DataFrame的iterrows()调用来遍历DataFrame的行。

import pandas as pd

import numpy as np

df = pd.DataFrame([{'c1':10, 'c2':100}, {'c1':11,'c2':110}, {'c1':12,'c2':120}])

for index, row in df.iterrows():

print(row['c1'], row['c2'])

**输出:**

|  |
| --- |
|  |

**4.我们如何排序DataFrame？**  
我们可以通过以下几种有效地在DataFrame中执行排序：

* 按标签
* 按实际值

(1)按标签  
可以使用sort\_index()方法对数据dataframe进行排序。可以通过传递axis参数和排序顺序来实现。默认情况下，按升序对行标签进行排序。

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'])

sorted\_df=unsorted\_df.sort\_index()

print (sorted\_df)

**输出:**

|  |
| --- |
|  |

**排序顺序**  
通过将布尔值传递给升序参数，可以控制排序的顺序。让我们考虑下面的例子来理解这个问题

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'])

sorted\_df = unsorted\_df.sort\_index(ascending=False)

print(sorted\_df)

**输出:**

|  |
| --- |
|  |

**对列进行排序**  
通过传递值1的axis参数，可以对列标签进行排序，让我们考虑下面的例子来理解这个问题。

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'])

sorted\_df = unsorted\_df.sort\_index(axis=1)

print(sorted\_df)

**输出:**

|  |
| --- |
|  |

(2)按照实际值  
**DataFrame.sort\_values(by, axis=0, ascending=True, inplace=False)**  
sort\_values提供了一个功能，我们可以在其中指定要排序的值的DataFrame列名。它是通过传递“by”参数来完成的。

**参数解析:**

* axis:{0 or ‘index’, 1 or ‘columns’}, default 0，默认按照索引排序，即纵向排序，如果为1，则是横向排序
* by:如果axis=0，那么by="列名"；如果axis=1，那么by="行名"；
* ascending:布尔型，True则升序，可以是[True,False]，即第一字段升序，第二个降序
* inplace:布尔型，是否用排序后的数据框替换现有的数据框

import numpy as np

import pandas as pd

from pandas import DataFrame

data=[["a",2,301],["b",1,201],["c",2,201],["d",1,301],["e",2,301]]

df=pd.DataFrame(data,columns=["A","B","C"])

**查看源数据:**

|  |
| --- |
|  |

df.sort\_values(by=["C","B"],ascending=[False,True],inplace=True)

**输出:**

|  |
| --- |
|  |

data\_1=[[300,2,301],[2,1,201],[3,300,201],[100,1,301],[500,2,301]]

df\_1=pd.DataFrame(data\_1,columns=["A","B","C"])

**输出:**

|  |
| --- |
|  |

df\_1.sort\_values(by=0,axis=1,inplace=True)

**输出:**

|  |
| --- |
|  |

**7.如何删除panda DataFrame中的行?**

import pandas as pd

data = {'name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],

'year': [2012, 2012, 2013, 2014, 2014],

'reports': [4, 24, 31, 2, 3]}

df = pd.DataFrame(data, index = ['Cochice', 'Pima', 'Santa Cruz', 'Maricopa', 'Yuma'])

df

**输出:**

|  |
| --- |
|  |

**删除行**

df.drop(['Cochice', 'Pima'])

**输出 :**

|  |
| --- |
|  |

**8.如何删除panda DataFrame中的列?**  
**创建源数据**

test\_dict = {'id':[1,2,3,4,5,6],'name':['Alice','Bob','Cindy','Eric','Helen','Grace '],'math':[90,89,99,78,97,93],'english':[89,94,80,94,94,90]}

#直接写入参数test\_dict

test\_dict\_df = pd.DataFrame(test\_dict)

#字典型赋值

test\_dict\_df = pd.DataFrame(data=test\_dict)

test\_dict\_df

**输出:**

|  |
| --- |
|  |

**删除列**  
删除列，我们使用drop()函数，则要增加参数axis=1

test\_dict\_df.drop(['id'],axis=1)

**输出:**

|  |
| --- |
|  |