《数据科学与大数据导论》实验课程

第3章 使用Pandas进行数据分析

目录

3.1 Pandas安装

Pandas是什么？

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，它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提供了大量能快速便捷处理数据的函数和方法，是使Python成为强大而高效的数据分析的重要因素之一。

3.1 Pandas安装

在cmd下使用命令： pip install pandas

```
C:\Users\Administrator>pip install pandas
```

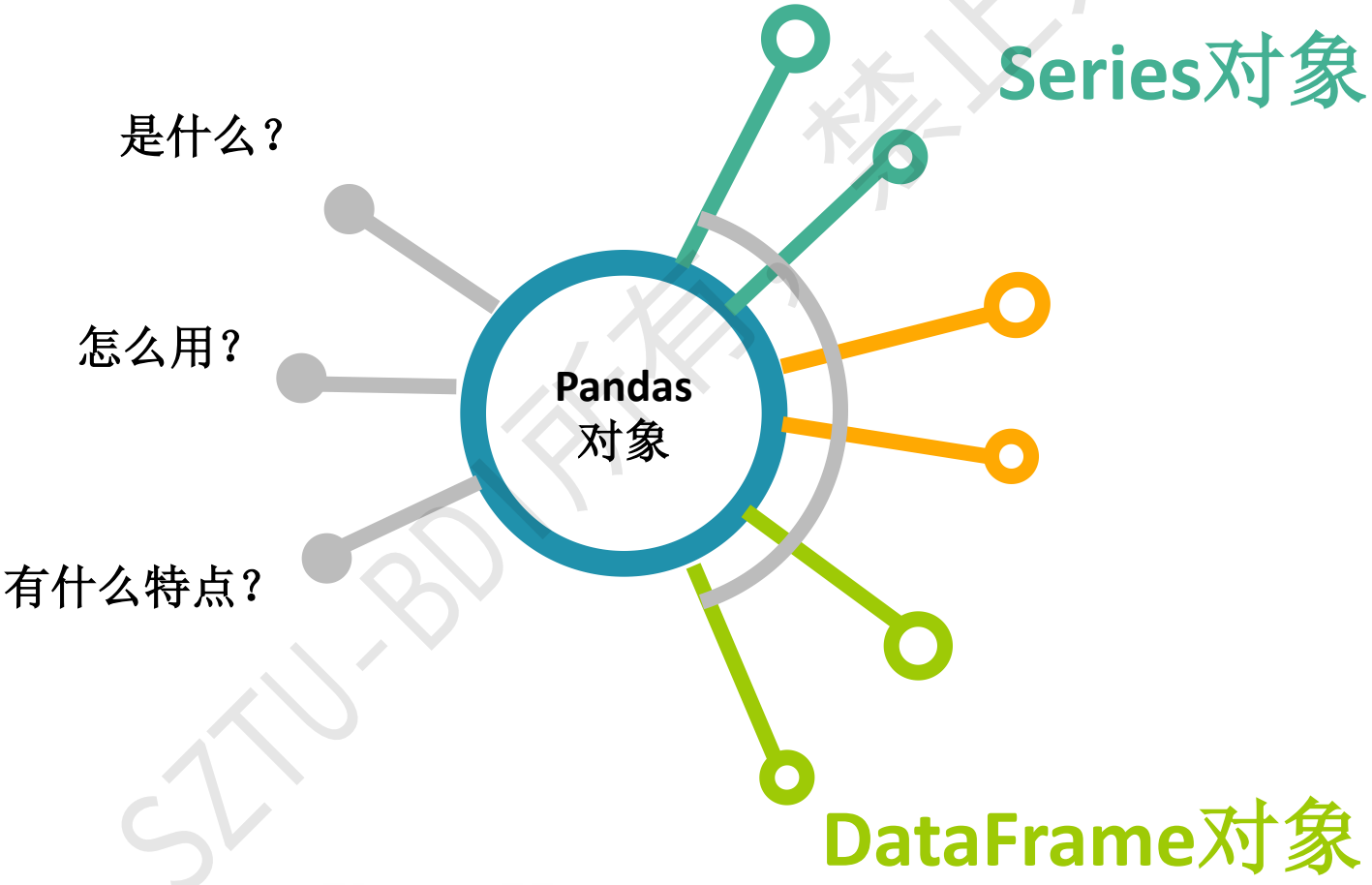
注意：若没有配置环境变量，需要先cd到python的安装目录

```
Collecting six>=1.5 (from python-dateutil>=2.6.1->pandas)
  Downloading https://files.pythonhosted.org/packages/ee/ff/48bde5c0f013094d729f
e4b0316ba2a24774b3ff1c52d924a8a4cb04078a/six-1.15.0-py2.py3-none-any.whl
Installing collected packages: pytz, six, python-dateutil, pandas
Successfully installed pandas-1.0.5 python-dateutil-2.8.1 pytz-2020.1 six-1.15.0

WARNING: You are using pip version 19.2.3, however version 20.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' comm
and.
```

安装完成后实验Pandas是否安装成功

3.2 Pandas的对象



3.2.1 Series对象

Series是Pandas中最基本的对象，类似于一维数组的对象，由一组数据和一组与之相关的数据标签（索引）组成。
Series对象相比于一维数据结构多了一些额外的功能，它的内部结构很简单，由两个相互关联的数组组成（values和index），其中values数组用来存放数据，主数组的每一个元素都有一个与之相关联的标签，这些标签存储在一个index的数组中。

```
>>> import pandas as pd
>>> x=pd.Series([1,3,5,7])          #创建一个Series对象
>>> x    #查看对象x内容
0    1
1    3
2    5
3    7
dtype: int64
>>>
```

3.2.1 Series对象

标签index内的内容也可以进行指定

```
>>> import pandas as pd
>>> x=pd.Series([1,3,5,7],index=['a','b','c','d'])
>>> x   #查看对象x内容
a    1
b    3
c    5
d    7
dtype: int64
>>>
```

可以尝试查看这个对象中的两个数组values和index里面的内容

```
>>> x.values                #查看x对象中values数组
array([1, 3, 5, 7], dtype=int64)
>>> x.index                 #查看x对象中index数组
Index(['a', 'b', 'c', 'd'], dtype='object')
>>>
```

3.2.1 Series对象

Series对象的values属性本来就是一个Numpy的一个数组对象，而Series对象中的values数组也是对Numpy中的ndarray对象的引用，如果改变原有对象的值，Series对象的值也会跟着改变。

```
>>> import numpy as np
>>> import pandas as pd
>>> array1=np.array([1,3,5,7])          #创建一个一维数组
>>> array1                  #查看创建的一维数组
array([1, 3, 5, 7])
>>> x=pd.Series(array1) #创建一个Series对象，用array1数组导入
>>> x                                   #查看对象x内容
0    1
1    3
2    5
3    7
dtype: int32
>>> y=pd.Series(x)      #使用对象x再创建一个对象
>>> y
0    1
1    3
2    5
3    7
dtype: int32
>>>
```

3.2.1 Series对象

因为Series对象的index对应values，所以可以用字典对象来构造Series对象。字典中的所有的键放在Series对象的index数组中，字典中的所有值放在Series对象的values数组中，仍然保持对应关系。下面的代码给出了一个字典实例，如果index中的值在字典中有对应的键，则生成的Series对象中该值对应的元素为在字典中对应的值，如果找不到，则其值为NaN，即空值。

```
>>> import pandas as pd
>>> z={"a":3,"b":4,"c":5}                    #创建字典数据类型
>>> z                    #查看字典z内容
{'a': 3, 'b': 4, 'c': 5}
>>> x=pd.Series(z,index=["a","b","c", "d"])
>>> x                    #查看对象x内容
a    3.0
b    4.0
c    5.0
d    NaN
dtype: float64
>>>
```

3.2.2 DataFrame对象

DataFrame对象的数据结构跟excel表相似，其目的是将Series的使用场景由一维扩展到多维，它由按一定顺序排列的多列数据组成，各列的数据类型可以有所不同。

DataFrame对象有两个索引数组（index和columns），第一个数组index与行相关，它与Series的索引数组极为相似，每个索引值都跟所在的一行相关联；第二个数组columns包含一系列列标签（每个值相当于列名）。DataFrame可以理解为一个由Series组成的字典，其中每一列的名称作为字典的键，形成DataFrame列的Series作为字典的值，每个Series的所有元素映射到称为index的标签数组中。

3.2.2 DataFrame对象

```
>>> import pandas as pd
>>> data={"a":[1,2,3,4],"b":[5,6,7,8],"c":[9,10,11,12]}
>>> x=pd.DataFrame(data)                    #使用字典创建DataFrame对象
>>> x                #查看对象x内容
   a  b   c
0  1  5   9
1  2  6  10
2  3  7  11
3  4  8  12
>>>
```

与Series对象一样，可以指定index数组的内容，下面的案例中指定了index数组的内容。

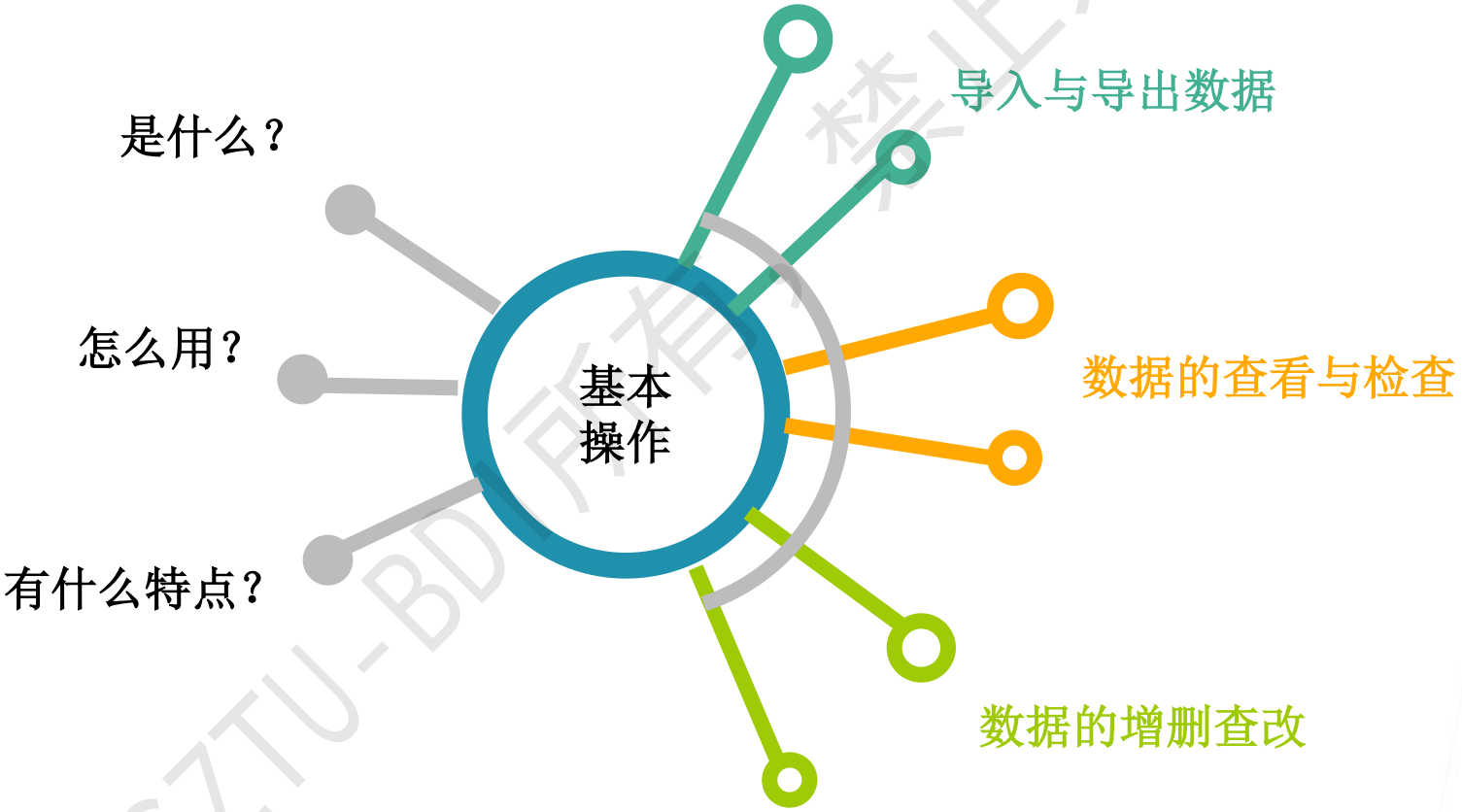
```
>>> y=pd.DataFrame(data,index=["one","two","three","four"])
>>> #使用字典创建DataFrame对象，并指定index属性内容
>>> y                    #查看对象y内容
       a  b   c
one    1  5   9
two    2  6  10
three  3  7  11
four   4  8  12
>>>
```

3.2.2 DataFrame对象

同样可以使用数组矩阵构造DataFrame对象

```
>>> import numpy as np
>>> import pandas as pd
>>> x=pd.DataFrame(np.arange(16).reshape((4,4)),index=["one","two","three","four
"],columns=["ball", "pen", "pencil", "paper"])
>>> #使用Numpy数组创建DataFrame对象，并指定index、columns标签内容
>>> x                          #查看对象x内容
       ball  pen  pencil  paper
one       0    1       2      3
two       4    5       6      7
three     8    9      10     11
four     12   13      14     15
>>>
```

3.3 Pandas基本操作



3.3.1 导入与导出数据

1.数据导入

（1）csv文件的导入

函数原型：read_csv(filrpath,sep,names,encoding)。

参数：①filepath完成导入csv文件的路径，一般使用绝对路径，且用“/”或者“\”表示；②sep表示分隔符，一般csv文件默认是逗号；③names表示导入的列和指定列的顺序，默认按顺序导入所有列；④encoding表示文件编码，大多时候会让参数encoding=‘utf-8’。

```
>>> import pandas as pd
>>> #读取csv文件pd.read_csv(文件路径)
>>> #df1为DataFrame
>>> df1=pd.read_csv(r"C:\data.csv")
```

3.3.1 导入与导出数据

1.数据导入

（2）txt文件的导入

函数原型：read_table(filrpath,sep,names,encoding)。

参数：①filepath完成导入csv文件的路径，一般使用绝对路径，且用“/”或者“\”表示；②sep表示分隔符，一般csv文件默认是逗号；③names表示导入的列和指定列的顺序，默认按顺序导入所有列；④encoding表示文件编码，大多时候会让参数encoding=‘utf-8’。

```
>>> import pandas as pd
>>> #因为data.txt没有列名。所以要加上header=None
>>> df1=pd.read_table(r"C:\data.txt",header=None)
```

3.3.1 导入与导出数据

1.数据导入

（3）Excel文件的导入

函数原型：read_excel(filrpath,sep,names)。

参数：①filepath完成导入csv文件的路径，一般使用绝对路径，且用“/”或者“\”表示；②sep表示分隔符，一般csv文件默认是逗号；③names表示导入的列和指定列的顺序，默认按顺序导入所有列。

```
>>> import pandas as pd
>>> #df1为DataFrame
>>> df1=pd.read_excel(r"C:\data.xlsx")
```

3.3.1 导入与导出数据

2.数据导出

函数原型：to_csv(filrpath,sep,names,encoding)。
参数：文件路径filrpath的末尾要写上.csv文件格式；分隔符sep是输出文件的分隔符，默认为逗号，也可以用制表符等；names是否输出索引，默认为输出索引，如果不想要可以改为False；encoding是否输出列名，默认为输出列名；编码默认为utf-8。

```
>>> import pandas as pd
>>> df1.to_csv(r"C:\data1.csv",index=True,header=True)
>>> #注意导出的文件后缀要写成.csv
>>> df1.to_csv(r"C:\data2.csv",index=False,header=True)
>>> #index和header默认为True
>>>
```

3.3.2 数据的查看与检查

1.Series对象数据的查看与检查

直接通过主数组的下标来获取，或者通过对象的index标签值来获取。

```
>>> import pandas as pd
>>> x=pd.Series([1,3,5,7],index=['a','b','c','d'])
>>> x[2]                    #通过下标查看数组
5
>>> x['c']          #通过index标签查看数组
5
```

可以一次性获取多个元素，同样也有和刚刚类似的两种方法，对于数组下标，只需要用“：”表示起始到终止的位置即可（这个结果只包含起始值但不包含终止值）。

```
>>> x[0:2]                  #通过下标连续引用多个数据
a    1
b    3
dtype: int64
>>> x[['a','b']]            #通过index标签连续引用多个数据
a    1
b    3
dtype: int64
>>>
```

3.3.2 数据的查看与检查

2.DataFrame对象数据的查看与检查

```
>>> import numpy as np
>>> import pandas as pd
>>> x=pd.DataFrame(np.arange(16).reshape((4,4)),index=["one","two","three","four
"],columns=["ball", "pen", "pencil", "paper"])
>>> x                                    #查看对象x内容
       ball  pen  pencil  paper
one       0    1       2      3
two       4    5       6      7
three     8    9      10     11
four     12   13      14     15
>>> x.columns                            #查看对象x的columns标签
Index(['ball', 'pen', 'pencil', 'paper'], dtype='object')
>>> type(x.columns)      #查看对象x的columns标签的数据类型
<class 'pandas.core.indexes.base.Index'>
>>> x.index                              #查看对象x的index标签
Index(['one', 'two', 'three', 'four'], dtype='object')
>>> type(x.index)        #查看对象x的index标签的数据类型
<class 'pandas.core.indexes.base.Index'>
>>> x.values                             #查看对象x的values标签
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
>>> type(x.values)       #查看对象x的values标签的数据类型
<class 'numpy.ndarray'>
>>>
```

3.3.2 数据的查看与检查

2.DataFrame对象数据的查看与检查

DataFrame对象也可以获得一列数据，获得一列数据有两种方法，一种是用“[]”，另一种使用符号“.”来连接。DataFrame对象也可以获得多行数据。

```
>>> x["pencil"]          #通过pencil标签获得一列数据
one      2
two      6
three   10
four    14
Name: pencil, dtype: int32
>>> x.pen                                  #通过.pen方法获得一列数据
one      1
two      5
three    9
four    13
Name: pen, dtype: int32
>>> x[0:2]               #通过下标连续引用多个数据
     ball  pen  pencil  paper
one     0    1       2      3
two     4    5       6      7
>>>
```

3.3.3 数据的增删查改

1.数据的增加

增加数据可以像字典一样直接添加。

```
>>> import pandas as pd
>>> x=pd.Series([1,3,5,7],index=['a','b','c','d'])#创建一个对象
>>> x                          #查看对象x内容
a    1
b    3
c    5
d    7
dtype: int64
>>> x['e']=9                   #增加对象标签为e的内容
>>> x
a    1
b    3
c    5
d    7
e    9
dtype: int64
>>>
```

也可以使用append()函数进行增加，增加效果类似，但区别在于append()函数添加元素后，原来的值没有改变。

```
>>> x.append(pd.Series([9],index=['e']))        #增加对象标签为e的内容
a    1
b    3
c    5
d    7
e    9
dtype: int64
>>> x
a    1
b    3
c    5
d    7
dtype: int64
>>>
```

3.3.3 数据的增删查改

2.数据的删除

del方法可用于删除元素

```
>>> x
a    1
b    3
c    5
d    7
dtype: int64
>>> del x['a']          #删除对象中标签为a的内容
>>> x
b    3
c    5
d    7
dtype: int64
>>>
```

Pandas也提供了删除函数pop()

```
>>> x=pd.Series([1,3,5,7],index=['a','b','c','d'])#创建一个对象
>>> x.pop('b')          #使用pop()函数进行删除
3
>>> x                   #查看对象x内容
a    1
c    5
d    7
dtype: int64
>>>
```

3.3.3 数据的增删查改

3.数据的查找与修改

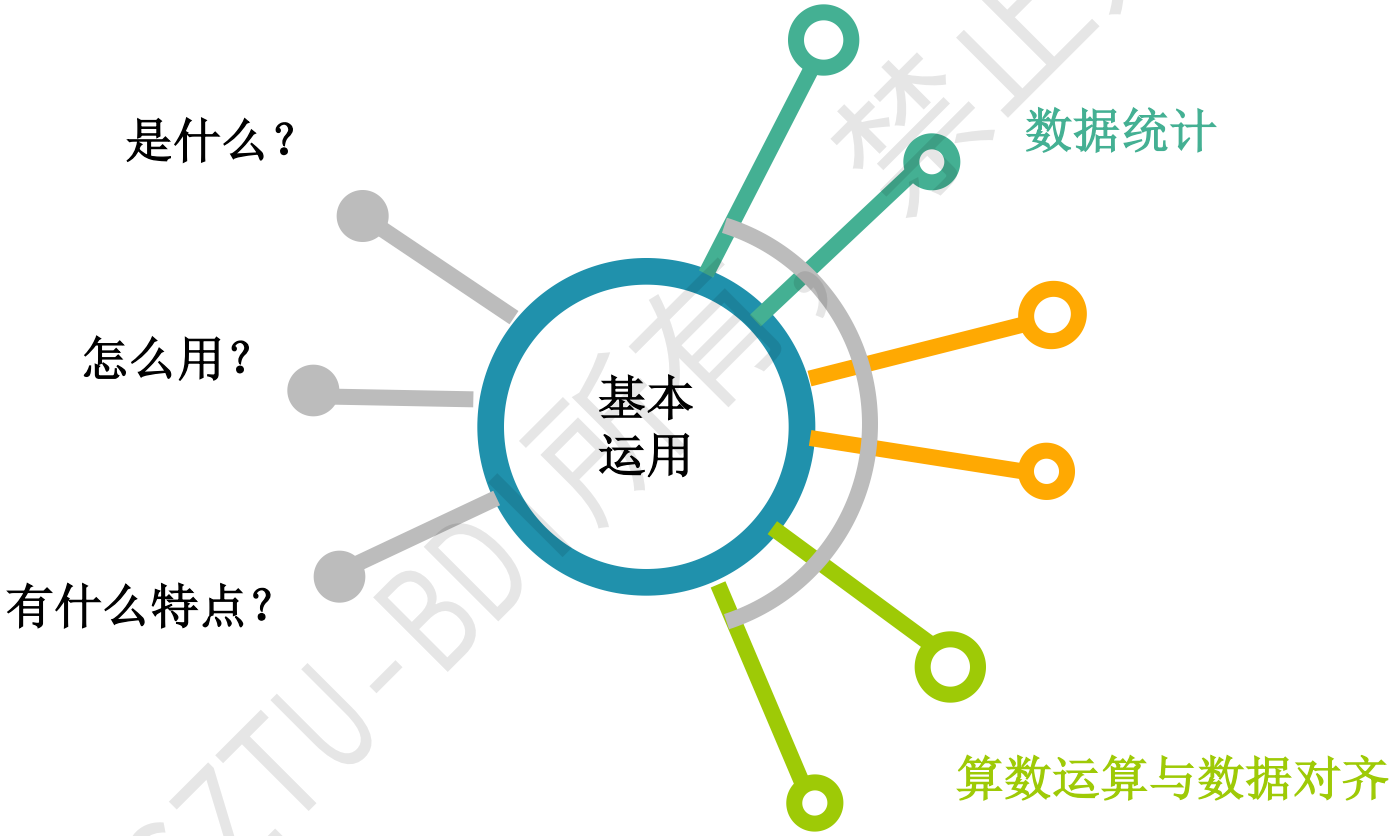
由于Pandas库是以Numpy库为基础开发的，所以Numpy数组的许多操作方法对Series对象也有效，例如数据的筛选。

```
>>> x[x>4]                    #筛选x>4的对象
c    6
d    7
dtype: int64
>>>
```

对于DataFrame对象的数据修改，也是类似。查找某个元素值，类似于二维数组的查找办法，需要用两个[]分别找它的行标和列标；同时，也可以找到它的位置然后修改它的值。

```
>>> x["pencil"][1]                    #查找对象中的值
6
>>> x["pencil"][1]=12                 #修改查找的值
>>> x["pencil"][1]                    #查看修改后对象中的值
12
>>>
```

3.4 Pandas的基本运用



3.4.1 数据统计

1.sum()函数与cumsum()函数

sum()函数用于对象求和。cumsum()函数用于累计求和，不同点在于，sum()只显示了单列或者单行结果，而cumsum()函数显示的累计求和的过程。

```
>>> import numpy as np
>>> import pandas as pd
>>> array1=np.array([1,2,3,4,5,6,7,8]).reshape(4,2)      #创建一个数组
>>> array1                          #查看创建的数组
array([[1, 2],
       [3, 4],
       [5, 6],
       [7, 8]])
>>> x=pd.DataFrame(array1,index=['a','b','c','d',],columns=['one','two'])
>>> x
   one  two
a    1    2
b    3    4
c    5    6
d    7    8
>>> x.sum()                         #使用sum()函数计算
one    16
two    20
dtype: int64
>>> x.sum(axis=1)          #使用sum()函数计算，axis=1
a     3
b     7
c    11
d    15
dtype: int64
>>>
```

```
>>> x.cumsum()   #使用cumsum()函数计算
   one  two
a    1    2
b    4    6
c    9   12
d   16   20
>>>
```

3.4.1 数据统计

2.idxmax()函数与idxmin()函数

这两个函数的功能是返回最大最小值的行名称

```
>>> x.idxmax()                    #使用idxmax()函数计算
one     d
two     d
dtype: object
>>> x.idxmin()                    #使用idxmin()函数计算
one     a
two     a
dtype: object
>>>
```

3.4.1 数据统计

3.unique()函数与value_counts()函数

unique()函数的功能是去除重复的元素，使用Series对象的unique()函数，返回一个Numpy数组。

value_counts()函数的功能是返回一个Series对象，index为原Series对象中不重复的元素，values为不重复的元素出现的次数。

```
>>> import numpy as np
>>> import pandas as pd
>>> x=pd.Series([1,3,5,7,2,4,3,5,7,6,7])          #创建一个对象
>>> x
0     1
1     3
2     5
3     7
4     2
5     4
6     3
7     5
8     7
9     6
10    7
dtype: int64
```

```
>>> x.unique()                                    #使用unique()函数进行去重复值
array([1, 3, 5, 7, 2, 4, 6], dtype=int64)
>>> type(x.unique())                             #查看去除重复值后的数据类型
<class 'numpy.ndarray'>
```

```
>>> x.value_counts()                              #使用value_counts()函数进行去重复值
7    3
3    2
5    2
1    1
2    1
4    1
6    1
dtype: int64
>>> type(x.value_counts())                        #查看去除重复值后对象的数据类型
<class 'pandas.core.series.Series'>
```

3.4.1 数据统计

4.isin()函数

isin()函数可用于筛选数据，判定Series中的每个元素中是否包含在给定的isin()的参数中，如果包含，则为True，否则为False。

```
>>> x.isin([2,4])                    #使用isin()函数进行数据筛选
0     False
1     False
2     False
3     False
4      True
5      True
6     False
7     False
8     False
9     False
10    False
dtype: bool
>>> x[x.isin([2,4])]
4    2
5    4
dtype: int64
>>>
```

3.4.2 算数运算与数据对齐

1.算术运算

对于使用Numpy数组中的运算符（如+、-、*、/）或者其他的数学函数，也适用于Pandas。

```
>>> import numpy as np
>>> import pandas as pd
>>> x=pd.Series([20,40,60,80])  #创建x对象
>>> x                           #查看x对象的内容
0    20
1    40
2    60
3    80
dtype: int64
>>> x/2                         #对x对象进行除以二的操作
0    10.0
1    20.0
2    30.0
3    40.0
dtype: float64
>>> np.log(x)                   #对x对象使用log()函数
0    2.995732
1    3.688879
2    4.094345
3    4.382027
dtype: float64
>>>
```

3.4.2 算数运算与数据对齐

2.数据对齐

Pandas的数据对齐是数据清洗的重要过程，可以按索引对齐进行运算，如果没对齐的位置，则补NaN，即空值，在数据的末尾也可以填充NaN。对象除了和标量之间可以进行运算，对象和对象之间也可以进行运算，这样就可能存在没有数据对齐的情况，如果index的值没有对齐，则没有对齐的元素运算之后的值为NaN。

```
>>> x=pd.Series({"a": 3, "b": 4, "c": 5})          #创建x对象
>>> y=pd.Series({"a": 1, "b": 7, "c": 2, "d": 11})
>>> x
a    3
b    4
c    5
dtype: int64
>>> y
a     1
b     7
c     2
d    11
dtype: int64
>>> x+y                    #对象之间的加法运算
a     4.0
b    11.0
c     7.0
d     NaN
dtype: float64
>>>
```

3.5 Pandas使用案例

1.学生成绩数组导入

```
>>> import numpy as np
>>> import pandas as pd
>>> f=open('C:\data.xlsx','rb')          #打开文件data
>>> x=pd.read_excel(f)                          #导入数据对象
>>> x              #查看对象的值
       number  Usual performance  Final exam  total points
0   194020066                 88        93.5           NaN
1   194020019                 90        93.5           NaN
2   194020014                 94        92.5           NaN
3   194020004                 90        90.5           NaN
4   194020010                 85        90.5           NaN
5   194020046                 93        89.5           NaN
6   194020002                 92        89.5           NaN
7   194020028                 90        87.5           NaN
8   194020045                 85        86.5           NaN
9   194020021                 92        86.5           NaN
10  194020011                 90        85.5           NaN
11  194020017                 94        85.5           NaN
12  194020058                 93        84.5           NaN
13  194020051                 90        84.5           NaN
14  194020015                 95        84.5           NaN
15  194020038                 88        84.5           NaN
16  194020020                 92        83.5           NaN
17  194020053                 91        83.5           NaN
18  194020063                 89        81.5           NaN
19  194020037                 92        80.5           NaN
```

3.5 Pandas使用案例

2.总分计算

```
>>> x["total points"]=x["Usual performance"]*0.5+x["Final exam"]*0.5
>>> x["total points"]           #查看学生成绩总分
0     90.75
1     91.75
2     93.25
3     90.25
4     87.75
5     91.25
6     90.75
7     88.75
8     85.75
9     89.25
10    87.75
11    89.75
12    88.75
13    87.25
14    89.75
15    86.25
16    87.75
17    87.25
18    85.25
19    86.25
Name: total points, dtype: float64
>>>
```

3.5 Pandas使用案例

3.数据查看

```
>>> x[1:10]                          #查看x的行
      number  Usual performance  Final exam  total points
1  194020019                 90        93.5         91.75
2  194020014                 94        92.5         93.25
3  194020004                 90        90.5         90.25
4  194020010                 85        90.5         87.75
5  194020046                 93        89.5         91.25
6  194020002                 92        89.5         90.75
7  194020028                 90        87.5         88.75
8  194020045                 85        86.5         85.75
9  194020021                 92        86.5         89.25
```

4.求和

```
>>> x.sum()                          #sum()求和
number               3.880401e+09
Usual performance    1.813000e+03
Final exam           1.738000e+03
total points         1.775500e+03
dtype: float64
>>> x.cumsum()                       #cumsum()求和
        number  Usual performance  Final exam  total points
0    194020066                 88        93.5         90.75
1    388040085                178       187.0        182.50
2    582060099                272       279.5        275.75
3    776080103                362       370.0        366.00
4    970100113                447       460.5        453.75
5   1164120159                540       550.0        545.00
6   1358140161                632       639.5        635.75
7   1552160189                722       727.0        724.50
8   1746180234                807       813.5        810.25
9   1940200255                899       900.0        899.50
10  2134220266                989       985.5        987.25
11  2328240283               1083      1071.0       1077.00
12  2522260341               1176      1155.5       1165.75
13  2716280392               1266      1240.0       1253.00
14  2910300407               1361      1324.5       1342.75
15  3104320445               1449      1409.0       1429.00
16  3298340465               1541      1492.5       1516.75
17  3492360518               1632      1576.0       1604.00
18  3686380581               1721      1657.5       1689.25
19  3880400618               1813      1738.0       1775.50
```

3.5 Pandas使用案例

5.行标处理

```
>>> x.idxmax()                #idxmax()求最大值行标
number                 0
Usual performance     14
Final exam             0
total points           2
dtype: int64
>>> x.idxmin()                #idxmin()求最小值行标
number                 6
Usual performance      4
Final exam            19
total points          18
dtype: int64
```

6.导出文件

```
>>> x.to_csv(r"C:\data1.csv",index=True,header=True)
```

Thank You!