数据结构:

序列(Series):带标签(索引)的一维数组

如果标签是时间,时间序列.

数据帧(DataFrame):带行索引和列索引的二维数组

面板数据(Panel):带标签的三维数组

文件的读写:

1.文件打开三个动作:打开,读写,关闭

2.上下文管理语句:with语句,异常处理

1 Series

In [32]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

1.1 基本概念

In [37]:
```
import numpy as np  #np是numpy   pd是pandas
import pandas as pd #导入全部
pd.Series([1,2])   #这个是没有指定索引的,[1,2]是值的范围,取整数. 没有指定索引的时候用默认索引,就是从0~N-1.
#默认索引也称为隐式索引
```

Out[37]:
```
0    1
1    2
dtype: int64
```

In [3]:
```
pd.Series([1,2],index=['a','b']) #指定索引,index=['a','b']用来指定索引为a,b.  指定索引也称为显式索引
```

Out[3]:
```
a    1
b    2
dtype: int64
```

In [ ]:
```
1
```

In [4]:
```
1 from pandas import Series,DataFrame #只导入单个
2 s=Series([3,4,5,6])
3 s   #索引在左边，值在右边，默认索引
```

Out[4]:
```
0    3
1    4
2    5
3    6
dtype: int64
```

In [5]:
```
1 type(s)
```

Out[5]:
```
pandas.core.series.Series
```

In [6]:
```
1 s.index     #获得索引
```

Out[6]:
```
RangeIndex(start=0, stop=4, step=1)
```

In [7]:
```
1 s.values     #获得值，值为数组
```

Out[7]:
```
array([3, 4, 5, 6], dtype=int64)
```

1.1.1 索引

In [8]:
```
1 s=Series([3,4,5,6],index=['a','b','c','d']) #指定索引，显式索引
2 s          #一般值是数字,所以索引很多都会用字母来表示     序列是有序的,字典是无序的
```

Out[8]:
```
a    3
b    4
c    5
d    6
dtype: int64
```

In [9]:
```
s['a'] #通过索引，获得单个值
```

Out[9]: 3

In [10]:
```
s.a
```

Out[10]: 3

In [11]:
```
s.get('a')
```

Out[11]: 3

In [12]:
```
s[0]  #这里面虽然指定了索引,但是依然存在一个默认索引，隐式索引
```

Out[12]: 3

1.1.2 切片

In [22]:
```
s    #切片不会改变其数据类型
```

Out[22]:
```
a    3
b    4
c    5
d    6
dtype: int64
```

In [27]:
```
s['a':'c'] #切片，显示索引，两边闭合,切出来的还是序列,,可以在第三个参数中填入步长
```

Out[27]:
```
a    3
b    4
c    5
dtype: int64
```

In [13]:
```
s['a':'d':2]
```

Out[13]:
```
a    3
c    5
dtype: int64
```

In [28]:
```
s[0:3]  # 隐式索引，左闭右开,取不到3,取的是默认索引为0,1,2的序列
```

Out[28]:
```
a    3
b    4
c    5
dtype: int64
```

In [25]:
```
s[::2] #开始，结束，步长
```

Out[25]:
```
a    3
c    5
dtype: int64
```

In [29]:
```
s[['a','d']] #切片
```

Out[29]:
```
a    3
d    6
dtype: int64
```

In [14]:
```
s=Series([3,4,5,6],index=['a','c','c','d'])
s     #索引可以重复，尽量不要这样做
```

Out[14]:
```
a    3
c    4
c    5
d    6
dtype: int64
```

In [33]:
```
1 s['c']
```

Out[33]:
```
c    4
c    5
dtype: int64
```

In [ ]:
```
1
```

1.1.3 选择和过滤

In [15]:
```
1 s=Series([3,4,5,6],index=['a','b','c','d']) #指定索引
2 s
```

Out[15]:
```
a    3
b    4
c    5
d    6
dtype: int64
```

In [37]:
```
1 s>3  #判断  #选出值符合要求的,先对值进行判断
```

Out[37]:
```
a    False
b     True
c     True
d     True
dtype: bool
```

In [39]:
```
1 s[s>3] #选择大于3的所有序列 ,选出值符合要求的
```

Out[39]:
```
b    4
c    5
d    6
dtype: int64
```

In [16]:
```
s[(s>3)&(s<5)]   #选出值同时大于3而且小于5的序列
```

Out[16]:
```
b    4
dtype: int64
```

In [17]:
```
s
```

Out[17]:
```
a    3
b    4
c    5
d    6
dtype: int64
```

In [17]:
```
s.index=='a'   #判断索引是否等于a,等于a返回True,不等于a返回False  按照索引是否符合要求进行选择
```

Out[17]:
```
array([ True, False, False, False])
```

In [41]:
```
s[s.index=='a']   #选出索引符合要求的,即选出索引为a的序列
```

Out[41]:
```
a    3
dtype: int64
```

In [18]:
```
s[s.values == 6]  #选出值符合要求的,即选出值为6的序列
```

Out[18]:
```
d    6
dtype: int64
```

In [19]:
```
s=Series([3,4,5,6],index=['张三','李四','李五','王六'])
s
```

Out[19]:
```
张三    3
李四    4
李五    5
王六    6
dtype: int64
```

In [20]:
```
'李四'.startswith('李')  #判断索引字符串是不是以"李"开头   startswith对单个字符进行判断
```

Out[20]:
```
True
```

In [21]:
```
s[s.index.str.startswith('李')]   #str.startswith  对index里面的所有字符串进行判断是否以'李'开头
#选择索引以'李'开头的序列
```

Out[21]:
```
李四    4
李五    5
dtype: int64
```

In [23]:
```
s[s.index==str.startswith('李')]  #报错,字符串不能用等号连接,因为str.startswith('李')返回的是逻辑判断,True或False
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_22640/37242170.py in <module>
----> 1 s[s.index==str.startswith('李')]

TypeError: startswith() takes at least 1 argument (0 given)
```

In [24]:
```
s=Series([3,4,5,6],index=['张李','李四','李五','王六'])
s
```

Out[24]:
```
张李    3
李四    4
李五    5
王六    6
dtype: int64
```

In [25]:
```
s.index.str.contains('李')   #对索引中是否含有'李'字进行判断,返回True或False
```

Out[25]:
```
array([ True,  True,  True, False])
```

In [26]:
```
s[s.index.str.contains('李')]   #选择索引中含有'李'的序列
```

Out[26]:
```
张李    3
李四    4
李五    5
dtype: int64
```

In [112]:
```
s=Series([3,4,5,6],index=['张一李','李一四','李二五','王二六'])
s   #判断名字中间的字为"一"还是"二"进行选择
```

Out[112]:
```
张一李    3
李一四    4
李二五    5
王二六    6
dtype: int64
```

In [113]:
```
s.index.str.slice(1,2)=='一'  #判断名字中间的字为"一"返回True,不为"一"返回False
```

Out[113]:
```
array([ True,  True, False, False])
```

In [114]:
```
s[s.index.str.slice(1,2)=='一']  #对名字中间的字为"一"的序列进行选择
```

Out[114]:
```
张一李    3
李一四    4
dtype: int64
```

1.2 序列的运算

1.2.1 序列运算保留索引

In [39]:
```
s=Series([3,4,5,6],index=['a','b','c','d']) #指定索引
s
```

Out[39]:
```
a    3
b    4
c    5
d    6
dtype: int64
```

In [40]:
```
s1=s*2   #对值进行运算
s1
```

Out[40]:
```
a     6
b     8
c    10
d    12
dtype: int64
```

In [41]:
```
np.exp(s1)  #e的多少次方,值作为了指数
```

Out[41]:
```
a       403.428793
b      2980.957987
c     22026.465795
d    162754.791419
dtype: float64
```

In [42]:
```
s1
```

Out[42]:
```
a     6
b     8
c    10
d    12
dtype: int64
```

In [43]:
```
s1>8  #索引一直保留
```

Out[43]:
```
a    False
b    False
c     True
d     True
dtype: bool
```

1.2.2 序列运算索引自动对齐

In [47]:
```
s1=Series([3,4,5,6],index=['a','b','c','d']) #指定索引
s2=Series([4,5,6,3],index=['a','d','c','b']) #指定索引
#s1和s2初始序列的索引可以不是一个顺序的,但在计算的时候这两个序列的索引会自动对齐
```

In [48]:
```
s1
```

Out[48]:
```
a    3
b    4
c    5
d    6
dtype: int64
```

In [49]:
```
s2 #在进行计算的时候,索引会自动对齐位置,比如:a和a对齐,b和b对齐,c和c对齐,d和d对齐,所以相加是按照数组的方式进行相加
```

Out[49]:
```
a    4
d    5
c    6
b    3
dtype: int64
```

In [50]:
```
s1+s2    #索引自动对齐,,相对应的索引的值进行加和计算
```

Out[50]:
```
a     7
b     7
c    11
d    11
dtype: int64
```

In [51]:
```
#索引不存在，引入缺失值
s1=Series([3,4,5,6],index=['a','b','c','d']) #指定索引
s2=Series([3,4,5,6],index=['e','a','d','f']) #指定索引
s3=s1+s2
s3
#索引不会自动去除某一个或某一些,计算之后的两个序列的所有索引都会显示
#缺失值是特殊的浮点型,序列的数据类型需保持一直,所以所有的数据自动变成浮点型
```

Out[51]:
```
a     7.0
b     NaN
c     NaN
d    11.0
e     NaN
f     NaN
dtype: float64
```

In [52]:
```
s1=Series([3,4,5,6],index=['a','b','c','d'])
s2=Series([3,4,5,6],index=['e','a','d','f'])
s1
s2
```

Out[52]:
```
a    3
b    4
c    5
d    6
dtype: int64
```

Out[52]:
```
e    3
a    4
d    5
f    6
dtype: int64
```

In [53]:
```
s1.add(s2,fill_value=0)
#当对应索引没有缺失值时直接进行加和计算便可,当对应索引有某个是缺失值时,用0补足再进行加和计算
```

Out[53]:
```
a     7.0
b     4.0
c     5.0
d    11.0
e     3.0
f     6.0
dtype: float64
```

1.3 缺失值处理

In [54]:
```
1 s3
```

Out[54]:
```
a     7.0
b     NaN
c     NaN
d    11.0
e     NaN
f     NaN
dtype: float64
```

In [55]:
```
1 s3.isnull()  #缺失值的判断,,是缺失值返回True,不是缺失值返回False
```

Out[55]:
```
a    False
b     True
c     True
d    False
e     True
f     True
dtype: bool
```

In [56]:
```
1 s3.notnull()  #是缺失值返回False,不是缺失值返回True
```

Out[56]:
```
a     True
b    False
c    False
d     True
e    False
f    False
dtype: bool
```

In [57]:
```
1 s3
```

Out[57]:
```
a     7.0
b     NaN
c     NaN
d    11.0
e     NaN
f     NaN
dtype: float64
```

In [58]:
```
1 s3[s3.notnull()]  #选出不是缺失值的
```

Out[58]:
```
a     7.0
d    11.0
dtype: float64
```

In [59]:
```
1 s3.isnull().sum()
2 #统计有多少个缺失值
3 #返回的是True为1,False为0 ， sum求和是对布尔值进行求和(即1和0的和),这样就可以统计缺失值的个数
```

Out[59]:
```
4
```

In [60]:
```
1 s3.fillna(0) #用0填充缺失值
```

Out[60]:
```
a     7.0
b     0.0
c     0.0
d    11.0
e     0.0
f     0.0
dtype: float64
```

1.4 序列的名字和索引名字

In [61]:
```
s1=Series([89,90,91,92],index=['张三','李四','王五','老六']) #指定索引
s1
```

Out[61]:
```
张三    89
李四    90
王五    91
老六    92
dtype: int64
```

In [62]:
```
s1.name='学生成绩表'  #序列的名字(就是给序列起一个类似于表名的一个标题名)
s1
```

Out[62]:
```
张三    89
李四    90
王五    91
老六    92
Name: 学生成绩表, dtype: int64
```

In [63]:
```
s1.index
```

Out[63]:
```
Index(['张三', '李四', '王五', '老六'], dtype='object')
```

In [64]:
```
s1.index.name='学生姓名'  #(相当于加一个字段名)
```

In [65]:
```
s1   #值没有名字(即值不能增加类似于字段名这样的一个名字)
```

Out[65]:
```
学生姓名
张三    89
李四    90
王五    91
老六    92
Name: 学生成绩表, dtype: int64
```

In [ ]:
```

```

1.5 序列的创建

1.5.1 列表，元组(一维)

In [66]:
```
s1=Series([3,4,5,6],index=['a','b','c','d']) #可以用列表转换成序列
s1
```

Out[66]:
```
a    3
b    4
c    5
d    6
dtype: int64
```

In [67]:
```
s1=Series((3,4,5,6),index=['a','b','c','d']) #可以用元组转换成序列
s1
```

Out[67]:
```
a    3
b    4
c    5
d    6
dtype: int64
```

1.5.2 一维数组

In [68]:
```
s1=Series(np.arange(1,5),index=['a','b','c','d'])
s1   #可以用数组转换成序列,  arange(1,5)在1到5这个左闭右开的一个范围内取整数
```

Out[68]:
```
a    1
b    2
c    3
d    4
dtype: int32
```

1.5.3 标量

In [69]:
```
s1=Series(5,index=['a','b','c','d']) #标量，只有一个值5的时候,这个5会进行自动重复以匹配索引长度
s1
```

Out[69]:
```
a    5
b    5
c    5
d    5
dtype: int64
```

1.5.4 字典

In [70]:
```
dict1={'a':1,'b':3,'c':5,'d':8}    #这是一个字典
dict1
```

Out[70]:
```
{'a': 1, 'b': 3, 'c': 5, 'd': 8}
```

In [71]:
```
s1=Series(dict1)    #不指定索引，默认以字典的key作为索引，字典值作为值
s1          #字典转换成序列
```

Out[71]:
```
a    1
b    3
c    5
d    8
dtype: int64
```

In [72]:
```
s1=Series(dict1,index=['c','d','e','f'])
s1    #指定了索引,如果索引对应的值没有的话,会以缺失值进行显示
#原字典中的key在想要转换的序列的索引里没有出现,则此key及其对应的值形成的键值对 不会被转换到序列里
```

Out[72]:
```
c    5.0
d    8.0
e    NaN
f    NaN
dtype: float64
```

2 Dataframe

2.1 dataframe的创建

2.1.1 二维列表创建DataFrame(数据帧)

In [73]:
```
data1=[[1, 2, 3],[4, 5, 6],[7, 8, 9]] #可以用二维列表创建DataFrame(数据帧),或者可以用二维元组创建DataFrame(数据帧)
d1=DataFrame(data1)   #默认行索引和列索引
d1
#最左侧和最上方的黑色加粗部分为默认的行索引和列索引
```

Out[73]:

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

In [74]:
```
type(d1)
```

Out[74]:
```
pandas.core.frame.DataFrame
```

In [75]:
```
d1.index     #获取行索引用index
```

Out[75]:
```
RangeIndex(start=0, stop=3, step=1)
```

In [76]:
```
d1.columns   #获取列索引用columns
```

Out[76]:
```
RangeIndex(start=0, stop=3, step=1)
```

In [77]:
```
d1.values   #获取值用values
```

Out[77]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]], dtype=int64)
```

In [ ]:
```
1
```

In [78]:
```
1 data1=[[1, 2, 3],[4, 5, 6],[7, 8, 9]] #用二维列表创建DataFrame(数据帧),也可以用二维元组创建DataFrame(数据帧)
2 d1=DataFrame(data1,index=['a','b','c'],columns=['one','two','three'])#创建的时候加上行索引和列索引,即指定索引
3 d1
```

Out[78]:

	one	two	three
a	1	2	3
b	4	5	6
c	7	8	9

In [79]:
```
1 d1.index
```

Out[79]:
```
Index(['a', 'b', 'c'], dtype='object')
```

In [80]:
```
1 d1.columns
```

Out[80]:
```
Index(['one', 'two', 'three'], dtype='object')
```

In [81]:
```
1 d1=DataFrame(data1)
2 d1   #也可以在创建好以后再加上新的索引对之前的默认索引进行覆盖
```

Out[81]:

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

In [82]:
```
d1.index=['a','b','c']    #也可以在创建好以后再加上新的索引对之前的默认索引进行覆盖
d1.columns=['one','two','three']
d1
```

Out[82]:

	one	two	three
a	1	2	3
b	4	5	6
c	7	8	9

2.1.2 二维数组创建DataFrame(数据帧)

In [83]:
```
df2=DataFrame(np.arange(16).reshape(4,4)) #可以用数组转换为DataFrame
df2
```

Out[83]:

	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

In [ ]:
```

```

2.1.3 等长列表、 元组、 数组、 序列组成的字典创建DataFrame(数据帧)

In [86]:
```
data1={'a':[1,2],'b':[3,4]} #把key抽出形成columns,  key作为列的字段名(即索引)  ,行索引用默认索引
DataFrame(data1)            #用列表组成的字典可以转换为DataFrame(数据帧)
```

Out[86]:

	a	b
0	1	3
1	2	4

In [87]:
```
data1={'c':('1','2'),'a':('5','6')}
DataFrame(data1)    #用元组组成的字典可以转换为DataFrame(数据帧)
```

Out[87]:

	c	a
0	1	5
1	2	6

In [88]:
```
data1={'c':np.arange(4),'a':np.arange(4)}    #等长的数组创建DataFrame
DataFrame(data1)    #用数组组成的字典可以转换为DataFrame(数据帧)
```

Out[88]:

	c	a
0	0	0
1	1	1
2	2	2
3	3	3

In [90]:
```
data1={'c':Series([1,2,6]),'a':Series([3,4,5])}
DataFrame(data1)     #等长的序列创建DataFrame   #用序列组成的字典可以转换为DataFrame(数据帧)
```

Out[90]:

	c	a
0	1	3
1	2	4
2	6	5

2.1.4 字典组成的字典创建DataFrame

In [91]:
```
nest_dict={'shanghai':{2015:100,2016:101},'beijing':{2015:102,2016:103,2017:109}}
DataFrame(nest_dict)  #外层的key形成columns(列的字段名)，里层的key成为index(行的索引名)
#所有的key都会输出显示,没有值的地方用缺失值填充
```

Out[91]:

	shanghai	beijing
2015	100.0	102
2016	101.0	103
2017	NaN	109

2.1.5 字典或Series的列表

In [92]:
```
data = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]
DataFrame(data)
```

Out[92]:

	a	b	c
0	1	2	NaN
1	5	10	20.0

2.2 二维数据结构转换

1. 二维列表
2. 二维数组
3. 矩阵
4. 数据帧

2.2.1 二维列表转换为其他

In [93]:
```
L=[[1,2],[3,4]]  #这是一个二维列表
```

In [94]:
```
np.array(L)     #转换为数组
```

Out[94]:
```
array([[1, 2],
       [3, 4]])
```

In [95]:
```
np.mat(L)    #转换为矩阵
```

Out[95]:
```
matrix([[1, 2],
        [3, 4]])
```

In [96]:
```
DataFrame(L)    #转换为数据帧
```

Out[96]:

	0	1
0	1	2
1	3	4

2.2.2 二维数组转换为其他

In [98]:
```
a=np.arange(4).reshape(2,2)
a  #这是一个二维数组
```

Out[98]:
```
array([[0, 1],
       [2, 3]])
```

In [99]:
```
a.tolist()#转换为列表
```

Out[99]:
```
[[0, 1], [2, 3]]
```

In [100]:
```
1 np.mat(a)  #转换为矩阵
```

Out[100]:
```
matrix([[0, 1],
        [2, 3]])
```

In [101]:
```
1 DataFrame(a)  #转换为数据帧
```

Out[101]:

	0	1
0	0	1
1	2	3

2.2.3 二维矩阵转换为其他

In [102]:
```
1 m=np.mat([[1,2],[3,4]])
2 m  #这是一个矩阵
```

Out[102]:
```
matrix([[1, 2],
        [3, 4]])
```

In [103]:
```
1 m.tolist()  #转换为列表
```

Out[103]:
```
[[1, 2], [3, 4]]
```

In [104]:
```
1 np.array(m)#转换为数组
```

Out[104]:
```
array([[1, 2],
       [3, 4]])
```

In [105]:
```
1 DataFrame(m)  #转换为数据帧
```

Out[105]:

	0	1
0	1	2
1	3	4

2.2.4 数据帧转换为其他

In [108]:
```
1 d=DataFrame([[1,2],[3,4]])
2 d  #这是一个数据帧
```

Out[108]:

	0	1
0	1	2
1	3	4

In [109]:
```
1 d.values #转换为数组  将数据帧的值取出来就是转换为了数组
```

Out[109]:
```
array([[1, 2],
       [3, 4]], dtype=int64)
```

In [110]:
```
1 d.values.tolist() #转换为列表  将数据帧转换为数组后,就可以按照数组的转换方法进行转换成列表和其他
```

Out[110]:
```
[[1, 2], [3, 4]]
```

In [111]:
```
1 np.mat(d.values)   #转换为矩阵
```

Out[111]:
```
matrix([[1, 2],
        [3, 4]], dtype=int64)
```

2.3 数据的加载和查看

2.3.1 数据的读写

2.3.1.1 csv文件

In [263]:
```
data1=[[1, 2, 3],[4, 5, 6],[7, 8, 9]] #二维列表创建,二维元组
d1=DataFrame(data1,index=['a','b','c'],columns=['one','two','three']) #创建的时候加上行索引和列索引
d1   #创建一个DataFrame
```

Out[263]:

	one	two	three
a	1	2	3
b	4	5	6
c	7	8	9

In [264]:
```
d1.to_csv('d1.csv',sep=',', header=True)   #写入csv,带header(带有列的字段名即列的索引名)
#在d1.csv的前面引号的里面可以加上想要保存的指定路径,如果不加路径就是默认保存到当前路径
#sep=' '  引号里面可以用逗号也可以用空格
```

In [265]:
```
d1.to_csv('d2.csv',sep=',', header=False) #写入csv,不带header(没有列的字段名即列的索引名)
```

In [ ]:
```

```

In [266]:
```
pd.read_csv('d1.csv',index_col=0) #将第0列设置为索引,,有header的读法
```

Out[266]:

	one	two	three
a	1	2	3
b	4	5	6
c	7	8	9

In [267]:
```
pd.read_csv('d2.csv',index_col=0,names=['one','two','three']) #names参数增加列名,没有header的读法
```

Out[267]:

	one	two	three
a	1	2	3
b	4	5	6
c	7	8	9

2.3.1.2 excel文件

In [268]:
```
d1
```

Out[268]:

	one	two	three
a	1	2	3
b	4	5	6
c	7	8	9

In [269]:
```
d1.to_excel('d1.xlsx',sheet_name='d1') #写到excel中的某个表格,sheet_name定义excel的文件名为d1
```

In [270]:
```
pd.read_excel('d1.xlsx',sheet_name='d1',index_col=0) #读取excel中的某个表格,将第0列设置为索引
```

Out[270]:

	one	two	three
a	1	2	3
b	4	5	6
c	7	8	9

In [271]:
```
d2=pd.read_csv('iris.csv')
d2
```

0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [ ]:
```
#上下文管理语句
with open('说明.txt',encoding='utf-8') as f:
    txt=f.read()
txt
```

In [272]:
```
#写到多个表格，上下文管理语句
with pd.ExcelWriter('output.xlsx') as writer:
    d1.to_excel(writer, sheet_name='d1')
    d2.to_excel(writer, sheet_name='iris')

```

In [273]:
```
pd.read_excel('output.xlsx',sheet_name='iris',index_col=0) #读
```

0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

2.3.1.3 *非表格型文件的读写,看回放视频第二天的下午2点半左右的视频*

```
#参考这个网页
一\   https://blog.csdn.net/qq_35197351/article/details/84994089
二\   https://www.runoob.com/python/python-basic-syntax.html

```

In [ ]:
```
f=open('文件名.后缀',encoding='utf-8')  #执行后打开    前提是这个文件已经创建在当前路径下
```

In [ ]:
```
f.readlines()  #读取
```

In [ ]:
```
f.close()  #关闭   如果没有上下文管理语句,把文件读进来后再进行编辑后,不写close数据不会被保存.
```

In [ ]:
```
#上下文管理语句
with open('文件名.后缀',encoding='utf-8') as f:
    txt=f.read()
txt   #用上下文管理语句读进来,然后写,之后不用再写close也可以将数据保存
```

In [ ]:
```

```

2.3.1.4 html文件

In [117]:
```
df1=pd.read_html('http://stock.eastmoney.com/')
df1[6]      #从网页上读取表格  df1[6] 里面的参数指的是这个网页上的第几张表,不确定数字填几会有表
```

Out[117]:

	日期	名称	相关链接	涨跌幅	龙虎榜净买额(万)	上榜原因
0	2022-02-14	诚达药业	股吧详细	20.00%	6294.09	日涨幅达到15%的前5只证券
1	2022-02-14	保利联合	股吧详细	-4.85%	6052.98	日换手率达到20%的前5只证...
2	2022-02-14	德迈仕	股吧详细	20.01%	4517.69	日涨幅达到15%的前5只证券
3	2022-02-14	省广集团	股吧详细	10.05%	3679.36	日涨幅偏离值达到7%的前5...
4	2022-02-14	贵绳股份	股吧详细	9.97%	3103.98	非ST、*ST和S证券连续三个...

In [118]:
```
df1[6]
```

Out[118]:

	日期	名称	相关链接	涨跌幅	龙虎榜净买额(万)	上榜原因
0	2022-02-14	诚达药业	股吧详细	20.00%	6294.09	日涨幅达到15%的前5只证券
1	2022-02-14	保利联合	股吧详细	-4.85%	6052.98	日换手率达到20%的前5只证...
2	2022-02-14	德迈仕	股吧详细	20.01%	4517.69	日涨幅达到15%的前5只证券
3	2022-02-14	省广集团	股吧详细	10.05%	3679.36	日涨幅偏离值达到7%的前5...
4	2022-02-14	贵绳股份	股吧详细	9.97%	3103.98	非ST、*ST和S证券连续三个...

2.3.1.5 clipboard文件读取

In [123]:
```
1 pd.read_clipboard()#从粘贴板读取数据,在网页中直接复制表的内容,然后回到此页面不需要粘贴,直接运行即可
```

Out[123]:

	日期	名称	相关链接	涨跌幅	龙虎榜净买额(万)	上榜原因
0	2022-02-14	诚达药业	股吧详细	20.00%	6294.09	日涨幅达到15%的前5只证券
1	2022-02-14	保利联合	股吧详细	-4.85%	6052.98	日换手率达到20%的前5只证...
2	2022-02-14	德迈仕	股吧详细	20.01%	4517.69	日涨幅达到15%的前5只证券
3	2022-02-14	省广集团	股吧详细	10.05%	3679.36	日涨幅偏离值达到7%的前5...
4	2022-02-14	贵绳股份	股吧详细	9.97%	3103.98	非ST、*ST和S证券连续三个...

2.3.2 数据的查看

In [124]:
```
d2=pd.read_excel('学生成绩.xlsx') #读
d2
```

Out[124]:

	姓名	课程	成绩
0	孙志	英语	75
1	孙志	语文	58
2	孙志	英语	90
3	孙志	数学	89
4	赵昀艳	数学	28
...	...	...	...
195	李伟艳	数学	30
196	李琛志	数学	97
197	赵伟艳	英语	96
198	李东艳	语文	26
199	钱昀艳	语文	24

200 rows × 3 columns

In [125]:
```
d2.head(3)        #查看前三行
```

Out[125]:

	姓名	课程	成绩
0	孙志	英语	75
1	孙志	语文	58
2	孙志	英语	90

In [126]:
```
1 d2.tail(3)  #尾部,查看后几行
```

Out[126]:

	姓名	课程	成绩
197	赵伟艳	英语	96
198	李东艳	语文	26
199	钱昀艳	语文	24

In [127]:
```
1 d2.sample(n=4)   #随机的抽取几行
```

Out[127]:

	姓名	课程	成绩
95	钱琛艳	英语	38
143	钱坤	数学	95
117	李坤	数学	64
150	李琛艳	数学	13

In [128]:
```
1 d2.shape     #形状,查看行列数
```

Out[128]: (200, 3)

In [129]:
```
d2.info()  #数据类型，缺失值
#如果某列是字符串,数据类型显示为object
#如果某列是字符串与数字的组合,则数据类型也会显示为object,
#因为数组是有行列索引的,有位宽的限定,同一列的数据类型要一样指的就是位宽得一样,
#而以上两种无法确定它们的位宽,所以数据类型只能显示为object(对象)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   姓名      200 non-null    object
 1   课程      200 non-null    object
 2   成绩      200 non-null    int64 
dtypes: int64(1), object(2)
memory usage: 4.8+ KB
```

In [130]:
```
df=pd.DataFrame([[1,2,'a'],[1,'2','b']])
df
```

Out[130]:

	0	1	2
0	1	2	a
1	1	2	b

In [131]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   0       2 non-null      int64 
 1   1       2 non-null      object
 2   2       2 non-null      object
dtypes: int64(1), object(2)
memory usage: 176.0+ bytes
```

In [132]:
```
1 d2.dtypes  #查看数据类型
```

Out[132]:
```
姓名     object
课程     object
成绩      int64
dtype: object
```

In [133]:
```
1 d3=pd.read_excel('学生成绩-缺失.xlsx') #读
2 d3
```

Out[133]:

	姓名	课程	成绩
0	孙志	英语	75.0
1	孙志	语文	58.0
2	孙志	英语	90.0
3	孙志	数学	89.0
4	赵昀艳	NaN	28.0
...	...	...	...
195	李伟艳	数学	30.0
196	李琛志	数学	97.0
197	赵伟艳	英语	96.0
198	李东艳	语文	26.0
199	钱昀艳	语文	24.0

200 rows × 3 columns

In [134]:
```
1 d3.info() #缺失情况，数据类型
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   姓名      200 non-null    object 
 1   课程      198 non-null    object 
 2   成绩      189 non-null    float64
dtypes: float64(1), object(2)
memory usage: 4.8+ KB
```

2.4 索引和过滤

In [135]:
```
1 import numpy as np
2 np.random.seed(10)
3 df = DataFrame(np.random.randint(1,100,24).reshape(6,4))
4 df.index = ['a', 'b', 'c', 'd', 'e', 'f']
5 df.columns = ['one', 'two', 'three', 'four']
6 df
```

Out[135]:

	one	two	three	four
a	10	16	65	29
b	90	94	30	9
c	74	1	41	37
d	17	12	55	89
e	63	34	73	79
f	50	52	55	78

2.4.1 索引

2.4.1.1 行的选择

隐式索引

In [136]:
```
#选择行
df[0:1]  #选择第一行，默认的索引选择行 ， 左闭右开
```

Out[136]:

	one	two	three	four
a	10	16	65	29

In [137]:
```
df[2:3] #选择多行,左闭右开
```

Out[137]:

	one	two	three	four
c	74	1	41	37

In [138]:
```
df[2::2] #选择多行,左闭右开  ,设置步长
```

Out[138]:

	one	two	three	four
c	74	1	41	37
e	63	34	73	79

显示索引

In [139]:
```
df['a':'a'] #选择多行,指定的索引选择多行,两边闭合
```

Out[139]:

	one	two	three	four
a	10	16	65	29

In [140]:
```
1 df['a':'c'] #两边闭合
```

Out[140]:

	one	two	three	four
a	10	16	65	29
b	90	94	30	9
c	74	1	41	37

In [141]:
```
1 df['a':'f':2]  #可以设置步长
```

Out[141]:

	one	two	three	four
a	10	16	65	29
c	74	1	41	37
e	63	34	73	79

2.4.1.2 列的选择

In [ ]:
```
1 df['one':'three']  #这种方式不能用来选列,这是行的选择方式
```

In [142]:
```
#选择列
df
```

Out[142]:

	one	two	three	four
a	10	16	65	29
b	90	94	30	9
c	74	1	41	37
d	17	12	55	89
e	63	34	73	79
f	50	52	55	78

In [143]:
```
df['one']  #选出的是序列
```

Out[143]:
```
a    10
b    90
c    74
d    17
e    63
f    50
Name: one, dtype: int32
```

In [145]:
```
df[['one']] #两个中括号选择一列，是dataframe形式，  有行列索引
```

Out[145]:

	one
a	10
b	90
c	74
d	17
e	63
f	50

In [146]:
```
1 df[['one','three']] #选择多列,是dataframe形式,用逗号分隔
```

Out[146]:

	one	three
a	10	65
b	90	30
c	74	41
d	17	55
e	63	73
f	50	55

In [147]:
```
1 df
```

Out[147]:

	one	two	three	four
a	10	16	65	29
b	90	94	30	9
c	74	1	41	37
d	17	12	55	89
e	63	34	73	79
f	50	52	55	78

2.4.1.3 行，列的选择：loc方式 显式

In [148]:
```
df
```

Out[148]:

	one	two	three	four
a	10	16	65	29
b	90	94	30	9
c	74	1	41	37
d	17	12	55	89
e	63	34	73	79
f	50	52	55	78

In [149]:
```
df.loc['a':'c':2,'one':'three':2] #显式方式，两边闭合
#行列选择用逗号分开,前面是行(0轴)的选择,后面是列(1轴)的选择
```

Out[149]:

	one	three
a	10	65
c	74	41

In [150]:
```
df.loc['a':'c',['one','two','four']] #选择多行和多列,显式方式
#这个选择  行是一个范围,列是指定的某几列,用中括号括起来,列不是范围
```

Out[150]:

	one	two	four
a	10	16	29
b	90	94	9
c	74	1	37

In [151]:
```
df.loc[:,['one']] #选择多行和多列
#这个选择  行是所有行,列是第一列
```

Out[151]:

	one
a	10
b	90
c	74
d	17
e	63
f	50

In [152]:
```
df.loc[['a','c'],['one','four']] #选择多行和多列
#这个选择  行是指定的某几行,列是指定的某几列
```

Out[152]:

	one	four
a	10	29
c	74	37

In [ ]:
```
#如何选出c行?
```

In [153]:
```
df.loc[['c']]
```

Out[153]:

	one	two	three	four
c	74	1	41	37

In [154]:
```
1 df.loc[['c'],:]
```

Out[154]:

	one	two	three	four
c	74	1	41	37

In [155]:
```
1 df.loc['c':'c',:]
```

Out[155]:

	one	two	three	four
c	74	1	41	37

In [159]:
```
1 df.loc[:,['four']]  #选出第4列
```

Out[159]:

	four
a	29
b	9
c	37
d	89
e	79
f	78

2.4.1.4 行，列的选择：iloc方式 隐式

In [160]:
```
1 df
```

Out[160]:

	one	two	three	four
a	10	16	65	29
b	90	94	30	9
c	74	1	41	37
d	17	12	55	89
e	63	34	73	79
f	50	52	55	78

In [161]:
```
1 df.iloc[:2,:2] #隐式的方式，左闭右开
```

Out[161]:

	one	two
a	10	16
b	90	94

In [162]:
```
1 df.iloc[:3:2,:3:2]  #隐式的方式
```

Out[162]:

	one	three
a	10	65
c	74	41

In [163]:
```
df.iloc[:,[0,2]]  #所有行,指定列
```

Out[163]:

	one	three
a	10	65
b	90	30
c	74	41
d	17	55
e	63	73
f	50	55

In [164]:
```
df.iloc[[0,2],[0,2]]  #指定行,指定列
```

Out[164]:

	one	three
a	10	65
c	74	41

2.4.1.5 练习例子

In [167]:
```
df1=pd.read_clipboard() #从粘贴板读取数据
df1
```

Out[167]:

	代码	名称	相关链接	最新价	涨跌幅	振幅	成交额	换手率
0	2349	精华制药	股吧 研报	13.71	10.03%	5.54%	10.84亿	9.68%
1	399006	创业板指	股吧 研报	2816.44	3.09%	2.87%	1849.90亿	2.56%
2	2335	科华数据	股吧 研报	29.52	5.69%	5.87%	3.75亿	3.25%
3	2639	雪人股份	股吧 研报	9.66	-0.82%	3.29%	4.44亿	8.21%

In [177]:
```
1 df1[['名称','最新价','涨跌幅','振幅']]  #选择指定的列,所有行
```

Out[177]:

	名称	最新价	涨跌幅	振幅
0	精华制药	13.71	10.03%	5.54%
1	创业板指	2816.44	3.09%	2.87%
2	科华数据	29.52	5.69%	5.87%
3	雪人股份	9.66	-0.82%	3.29%

In [174]:
```
1 df1.loc[:,['名称','最新价','涨跌幅','振幅']]  #选择指定的列,所有行
```

Out[174]:

	名称	最新价	涨跌幅	振幅
0	精华制药	13.71	10.03%	5.54%
1	创业板指	2816.44	3.09%	2.87%
2	科华数据	29.52	5.69%	5.87%
3	雪人股份	9.66	-0.82%	3.29%

In [175]:
```
1 df1.iloc[:,[1,3,4,5]]  #选择指定的列,所有行, 隐式方式
```

Out[175]:

	名称	最新价	涨跌幅	振幅
0	精华制药	13.71	10.03%	5.54%
1	创业板指	2816.44	3.09%	2.87%
2	科华数据	29.52	5.69%	5.87%
3	雪人股份	9.66	-0.82%	3.29%

2.4.2 过滤

In [178]:
```
1 df
```

Out[178]:

	one	two	three	four
a	10	16	65	29
b	90	94	30	9
c	74	1	41	37
d	17	12	55	89
e	63	34	73	79
f	50	52	55	78

2.4.2.1 选择满足条件的行

In [179]:
```
1 df['one']>20  #筛选行,需要用列值满足的某种条件来筛选相对应的行
```

Out[179]:
```
a    False
b     True
c     True
d    False
e     True
f     True
Name: one, dtype: bool
```

In [180]:
```
1 df[df['one']>20]   #选出one列大于20的行
```

Out[180]:

	one	two	three	four
b	90	94	30	9
c	74	1	41	37
e	63	34	73	79
f	50	52	55	78

In [205]:
```
1 df[(df['one']>20)&(df['one']<60)] #想选one列值大于20且小于60的行
```

Out[205]:

	one	two	three	four	five
f	50	52	55	78	three

In [206]:
```
1 df[(df['one']>20)&(df['three']<60)] #想选one列值大于20且three列值小于60的行
```

Out[206]:

	one	two	three	four	five
b	90	94	30	9	one
c	74	1	41	37	two
f	50	52	55	78	three

In [183]:
```
1 df[df.index=='a'] #选出行索引符合要求的列
```

Out[183]:

	one	two	three	four
a	10	16	65	29

2.4.2.2 选择满足条件的列

In [186]:
```
1 df
```

Out[186]:

	one	two	three	four
a	10	16	65	29
b	90	94	30	9
c	74	1	41	37
d	17	12	55	89
e	63	34	73	79
f	50	52	55	78

In [187]:
```
1 df.loc['a']>20  #筛选列,需要用行值满足的某种条件来筛选相对应的列
```

Out[187]:
```
one      False
two      False
three     True
four      True
Name: a, dtype: bool
```

In [188]:
```
1  df.loc[:,df.loc['a']>20]    #a行大于20的所有列
2 #逗号前面是用来选择行的,但是需要用列值所满足的条件来筛选相对应的行
3 #逗号后面是用来选择列的,但是需要用行值所满足的条件来帅选相对应的列
```

Out[188]:

	three	four
a	65	29
b	30	9
c	41	37
d	55	89
e	73	79
f	55	78

In [189]:
```
1 df.loc[df['one']>20,:]     #选择one列大于50的所有行
```

Out[189]:

	one	two	three	four
b	90	94	30	9
c	74	1	41	37
e	63	34	73	79
f	50	52	55	78

2.4.2.3 选择满足条件的行和列

In [190]:
```
1 df
```

Out[190]:

	one	two	three	four
a	10	16	65	29
b	90	94	30	9
c	74	1	41	37
d	17	12	55	89
e	63	34	73	79
f	50	52	55	78

In [192]:
```
1 df.loc[df['three']>40,df.loc['c']>40]  #逗号前面用来筛选行,逗号后面用来筛选列
```

Out[192]:

	one	three
a	10	65
c	74	41
d	17	55
e	63	73
f	50	55

In [191]:
```
1 df.loc[df['two']>20,df.loc['c']>40]
```

Out[191]:

	one	three
b	90	30
e	63	73
f	50	55

In [193]:
```
1 df['five']=['one','one','two','two','three','three']  #增加列
2 df
```

Out[193]:

	one	two	three	four	five
a	10	16	65	29	one
b	90	94	30	9	one
c	74	1	41	37	two
d	17	12	55	89	two
e	63	34	73	79	three
f	50	52	55	78	three

In [194]:
```
1 df.loc[df['five']=='two',:]#选择five列取值为two的所有行
```

Out[194]:

	one	two	three	four	five
c	74	1	41	37	two
d	17	12	55	89	two

In [198]:
```
1 df[df['five'].isin(['one','two'])] #选择five列取值为one和two的所有行
```

Out[198]:

	one	two	three	four	five
a	10	16	65	29	one
b	90	94	30	9	one
c	74	1	41	37	two
d	17	12	55	89	two

In [195]:
```
1 df[df['five'].str.endswith('e')] #选出five以e结尾的所有行
```

Out[195]:

	one	two	three	four	five
a	10	16	65	29	one
b	90	94	30	9	one
e	63	34	73	79	three
f	50	52	55	78	three

In [199]:
```
1 df.loc[df['five'].str.endswith('e'),:] #选出five以e结尾的所有行
```

Out[199]:

	one	two	three	four	five
a	10	16	65	29	one
b	90	94	30	9	one
e	63	34	73	79	three
f	50	52	55	78	three

2.4.2.4 例子

In [201]:
```
1 iris=pd.read_csv('iris.csv') #鸢尾花数据集
2 iris
```

Out[201]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [202]:
```
1 iris[(iris['species']=='setosa')&(iris['sepal_length']>5)]   #'&'相当于是拼接的效果,有对应关系,and不行
```

Out[202]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
10	5.4	3.7	1.5	0.2	setosa
14	5.8	4.0	1.2	0.2	setosa
15	5.7	4.4	1.5	0.4	setosa
16	5.4	3.9	1.3	0.4	setosa
17	5.1	3.5	1.4	0.3	setosa
18	5.7	3.8	1.7	0.3	setosa
19	5.1	3.8	1.5	0.3	setosa
20	5.4	3.4	1.7	0.2	setosa
21	5.1	3.7	1.5	0.4	setosa
23	5.1	3.3	1.7	0.5	setosa
27	5.2	3.5	1.5	0.2	setosa
28	5.2	3.4	1.4	0.2	setosa
31	5.4	3.4	1.5	0.4	setosa
32	5.2	4.1	1.5	0.1	setosa
33	5.5	4.2	1.4	0.2	setosa
36	5.5	3.5	1.3	0.2	setosa
39	5.1	3.4	1.5	0.2	setosa
44	5.1	3.8	1.9	0.4	setosa
46	5.1	3.8	1.6	0.2	setosa
48	5.3	3.7	1.5	0.2	setosa

In [203]:
```
1 iris[(iris['species']=='setosa')&(iris['sepal_length']>5)]
```

Out[203]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
10	5.4	3.7	1.5	0.2	setosa
14	5.8	4.0	1.2	0.2	setosa
15	5.7	4.4	1.5	0.4	setosa
16	5.4	3.9	1.3	0.4	setosa
17	5.1	3.5	1.4	0.3	setosa
18	5.7	3.8	1.7	0.3	setosa
19	5.1	3.8	1.5	0.3	setosa
20	5.4	3.4	1.7	0.2	setosa
21	5.1	3.7	1.5	0.4	setosa
23	5.1	3.3	1.7	0.5	setosa
27	5.2	3.5	1.5	0.2	setosa
28	5.2	3.4	1.4	0.2	setosa
31	5.4	3.4	1.5	0.4	setosa
32	5.2	4.1	1.5	0.1	setosa
33	5.5	4.2	1.4	0.2	setosa
36	5.5	3.5	1.3	0.2	setosa
39	5.1	3.4	1.5	0.2	setosa
44	5.1	3.8	1.9	0.4	setosa
46	5.1	3.8	1.6	0.2	setosa
48	5.3	3.7	1.5	0.2	setosa

2.5 索引设置

In [207]:
```
1 d2=pd.read_excel('学生成绩.xlsx') #读
2 d2
```

Out[207]:

	姓名	课程	成绩
0	孙志	英语	75
1	孙志	语文	58
2	孙志	英语	90
3	孙志	数学	89
4	赵昀艳	数学	28
...	...	...	...
195	李伟艳	数学	30
196	李琛志	数学	97
197	赵伟艳	英语	96
198	李东艳	语文	26
199	钱昀艳	语文	24

200 rows × 3 columns

2.5.1 将某列设置为索引:set_index

In [208]:
```
d3=d2.set_index('姓名')
d3.head(3)
```

Out[208]:

	课程	成绩
姓名		
孙志	英语	75
孙志	语文	58
孙志	英语	90

In [212]:
```
d3=d2.set_index(['姓名','课程'])
d3.head(5)    #设置双索引,不常用
```

Out[212]:

		成绩
姓名	**课程**	
孙志	**英语**	75
	语文	58
	英语	90
	数学	89
赵昀艳	**数学**	28

2.5.2 恢复默认索引:reset_index

In [213]:
```
1 d3.reset_index()   #恢复默认索引用reset
```

Out[213]:

	姓名	课程	成绩
0	孙志	英语	75
1	孙志	语文	58
2	孙志	英语	90
3	孙志	数学	89
4	赵昀艳	数学	28
...	...	...	...
195	李伟艳	数学	30
196	李琛志	数学	97
197	赵伟艳	英语	96
198	李东艳	语文	26
199	钱昀艳	语文	24

200 rows × 3 columns

In [230]:
```
1 d3.reset_index(drop=True)#恢复默认索引，丢弃原来索引
```

Out[230]:

	课程	成绩
0	英语	75
1	语文	58
2	英语	90
3	数学	89
4	数学	28
...	...	...
195	数学	30
196	数学	97
197	英语	96
198	语文	26
199	语文	24

200 rows × 2 columns

In [214]:
```
1 iris[(iris['species']=='setosa')&(iris['sepal_length']>5)].reset_index()
```

Out[214]:

	index	sepal_length	sepal_width	petal_length	petal_width	species
0	0	5.1	3.5	1.4	0.2	setosa
1	5	5.4	3.9	1.7	0.4	setosa
2	10	5.4	3.7	1.5	0.2	setosa
3	14	5.8	4.0	1.2	0.2	setosa
4	15	5.7	4.4	1.5	0.4	setosa
5	16	5.4	3.9	1.3	0.4	setosa
6	17	5.1	3.5	1.4	0.3	setosa
7	18	5.7	3.8	1.7	0.3	setosa
8	19	5.1	3.8	1.5	0.3	setosa
9	20	5.4	3.4	1.7	0.2	setosa
10	21	5.1	3.7	1.5	0.4	setosa
11	23	5.1	3.3	1.7	0.5	setosa
12	27	5.2	3.5	1.5	0.2	setosa
13	28	5.2	3.4	1.4	0.2	setosa
14	31	5.4	3.4	1.5	0.4	setosa
15	32	5.2	4.1	1.5	0.1	setosa
16	33	5.5	4.2	1.4	0.2	setosa
17	36	5.5	3.5	1.3	0.2	setosa
18	39	5.1	3.4	1.5	0.2	setosa
19	44	5.1	3.8	1.9	0.4	setosa
20	46	5.1	3.8	1.6	0.2	setosa
21	48	5.3	3.7	1.5	0.2	setosa

In [274]:
```
1 iris[(iris['species']=='setosa')&(iris['sepal_length']>5)].reset_index(drop=True)
```

Out[274]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	5.4	3.9	1.7	0.4	setosa
2	5.4	3.7	1.5	0.2	setosa
3	5.8	4.0	1.2	0.2	setosa
4	5.7	4.4	1.5	0.4	setosa
5	5.4	3.9	1.3	0.4	setosa
6	5.1	3.5	1.4	0.3	setosa
7	5.7	3.8	1.7	0.3	setosa
8	5.1	3.8	1.5	0.3	setosa
9	5.4	3.4	1.7	0.2	setosa
10	5.1	3.7	1.5	0.4	setosa

2.5.3 重新索引：reindex

In [218]:
```
df1 = DataFrame([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]],
    columns=['one','two','three']
)
df1
```

Out[218]:

	one	two	three
0	1	2	3
1	4	5	6
2	7	8	9

In [219]:
```
df1.reindex([1,3,2])#重索引行，没有引入缺失值
#相当于增加了一行原来没有的行,没有定义值的话用缺失值NaN代替
```

Out[219]:

	one	two	three
1	4.0	5.0	6.0
3	NaN	NaN	NaN
2	7.0	8.0	9.0

In [220]:
```
df1.reindex([1,3,2],fill_value=0) #重索引行，引入缺失值
#对于增加的新的一行,用0这个值填充缺失值
```

Out[220]:

	one	two	three
1	4	5	6
3	0	0	0
2	7	8	9

2.6 dataframe的增加和删除

In [221]:
```
data1=[[1, 2, 3],[4, 5, 6],[7, 8, 9]] #二维列表创建,二维元组
d1=DataFrame(data1,index=['a','b','c'],columns=['one','two','three'])
d1
```

Out[221]:

	one	two	three
a	1	2	3
b	4	5	6
c	7	8	9

2.6.1 增加行

In [223]:
```
d1.loc['d']=[6,9,6]  # 增加一行，如果原来的行存在则进行修改
d1  #loc[]中括号里的行索引不能与原有行索引一样,否则不会增加行,而是会直接覆盖原有行,相当于修改
```

Out[223]:

	one	two	three
a	1	2	3
b	4	5	6
c	7	8	9
d	6	9	6

In [226]:
```
d1.loc['c']=[7,8,9]
d1  #没增加,只是修改了原有行
```

Out[226]:

	one	two	three
a	1	2	3
b	4	5	6
c	7	8	9
d	6	9	6
e	9	9	9

In [227]:
```
d1.loc['e']=9  #标量自动扩展
d1
```

Out[227]:

	one	two	three
a	1	2	3
b	4	5	6
c	7	8	9
d	6	9	6
e	9	9	9

2.6.2 增加列

In [228]:
```
data1=[[1, 2, 3],[4, 5, 6],[7, 8, 9]] #二维列表创建,二维元组
d1=DataFrame(data1,index=['a','b','c'],columns=['one','two','three'])
d1
```

Out[228]:

	one	two	three
a	1	2	3
b	4	5	6
c	7	8	9

In [229]:
```
d1['four']=[7,8,9]          #增加第4列,取值的长度要一致
d1
```

Out[229]:

	one	two	three	four
a	1	2	3	7
b	4	5	6	8
c	7	8	9	9

In [230]:
```
d1['five']=10 #标量自动扩展,增加第5列,用10自动填充
d1
```

Out[230]:

	one	two	three	four	five
a	1	2	3	7	10
b	4	5	6	8	10
c	7	8	9	9	10

In [231]:
```
d1['six']=d1['five']+d1['one']
d1   #用计算的方式增加列,增加第6列
```

Out[231]:

	one	two	three	four	five	six
a	1	2	3	7	10	11
b	4	5	6	8	10	14
c	7	8	9	9	10	17

In [232]:
```
d1.insert(0,'zero',[1,1,1]) #在某列插入  #在某个位置插入列  是一个原地操作
#第一个参数表示插入的列所在的索引位置,第二个参数是插入的列的列名,第三个参数是插入的列的值(从上到下的值)
```

In [233]:
```
d1
```

Out[233]:

	zero	one	two	three	four	five	six
a	1	1	2	3	7	10	11
b	1	4	5	6	8	10	14
c	1	7	8	9	9	10	17

In [234]:
```
d1.insert(1,'o',d1['zero']+d1['one']) #在某列插入,值用的是两列计算的最终值
```

In [235]:
```
d1
```

Out[235]:

	zero	o	one	two	three	four	five	six
a	1	2	1	2	3	7	10	11
b	1	5	4	5	6	8	10	14
c	1	8	7	8	9	9	10	17

2.6.3 删除

In [252]:
```
1 d1
```

Out[252]:

	zero	one	two	three	four	five	six
a	1	1	2	3	7	10	11
b	1	4	5	6	8	10	14
c	1	7	8	9	9	10	17

In [237]:
```
1 d1.drop('a',axis=0)   #删除行，默认情况，axis=0，新生成操作(不是原地改变)
2 #第一个参数是要删除的行或列的索引,第二个参数是表示删除的东西是在哪个轴上的,axis就等于几
3 #d1.drop('a',axis=0)表示我要删除的是位于0轴上索引为'a'的这一整个行
4 #所以axis=0代表0轴
```

Out[237]:

	zero	o	one	two	three	four	five	six
b	1	5	4	5	6	8	10	14
c	1	8	7	8	9	9	10	17

In [253]:
```
1 d1.drop('b',axis=0)
```

Out[253]:

	zero	one	two	three	four	five	six
a	1	1	2	3	7	10	11
c	1	7	8	9	9	10	17

In [255]:
```
1 d1.drop('c',axis=0)
```

Out[255]:

	zero	one	two	three	four	five	six
a	1	1	2	3	7	10	11
b	1	4	5	6	8	10	14

In [241]:
```
1 d1.drop(['a','b'],axis=0) #删除多行
```

Out[241]:

	zero	o	one	two	three	four	five	six
c	1	8	7	8	9	9	10	17

In [238]:
```
1 d1.drop('o',axis=1) #删除列
2 #d1.drop('o',axis=1)代表我要删除的是位于1轴上索引为'o'的这一整个列
3 #所以axis=1代表1轴
```

Out[238]:

	zero	one	two	three	four	five	six
a	1	1	2	3	7	10	11
b	1	4	5	6	8	10	14
c	1	7	8	9	9	10	17

In [240]:
```
1 d1.drop(['one','six'],axis=1) #删除多列
```

Out[240]:

	zero	o	two	three	four	five
a	1	2	2	3	7	10
b	1	5	5	6	8	10
c	1	8	8	9	9	10

In [257]:
```
1 d1
```

Out[257]:

	zero	one	two	three	four	five	six
a	1	1	2	3	7	10	11
b	1	4	5	6	8	10	14
c	1	7	8	9	9	10	17

In [258]:
```
1 d1.drop('one',axis=1,inplace=False)
2 #inplace=True代表原地删除,inplace=False代表不是原地删除,会新生成操作,如果第三个参数不写,默认inplace=False
```

Out[258]:

	zero	two	three	four	five	six
a	1	2	3	7	10	11
b	1	5	6	8	10	14
c	1	8	9	9	10	17

In [260]:
```
1 d1
```

Out[260]:

	zero	one	two	three	four	five	six
a	1	1	2	3	7	10	11
b	1	4	5	6	8	10	14
c	1	7	8	9	9	10	17

In [261]:
```
1 d1.drop('six',axis=1,inplace=True) #删除列,原地删除
2 #inplace=True代表原地删除,inplace=False代表不是原地删除,会新生成操作,如果第三个参数不写,默认inplace=False
```

In [262]:
```
1 d1
```

Out[262]:

	zero	one	two	three	four	five
a	1	1	2	3	7	10
b	1	4	5	6	8	10
c	1	7	8	9	9	10

2.6.4 例子

In [246]:
```
1 df=pd.read_clipboard() #从粘贴板读取数据
2 df
```

Out[246]:

	代码	名称	相关链接	最新价	涨跌幅	振幅	成交额	换手率
0	2349	精华制药	股吧 研报	13.71	10.03%	5.54%	10.84亿	9.68%
1	399006	创业板指	股吧 研报	2816.44	3.09%	2.87%	1849.90亿	2.56%
2	2335	科华数据	股吧 研报	29.52	5.69%	5.87%	3.75亿	3.25%
3	2639	雪人股份	股吧 研报	9.66	-0.82%	3.29%	4.44亿	8.21%

In [247]:
```
1 df.drop(['相关链接','代码'],axis=1)
```

Out[247]:

	名称	最新价	涨跌幅	振幅	成交额	换手率
0	精华制药	13.71	10.03%	5.54%	10.84亿	9.68%
1	创业板指	2816.44	3.09%	2.87%	1849.90亿	2.56%
2	科华数据	29.52	5.69%	5.87%	3.75亿	3.25%
3	雪人股份	9.66	-0.82%	3.29%	4.44亿	8.21%