文本分析

第5讲

李田雨

数联璧合

数联璧合

我们经常需要将多个Excel文件，或者从多个渠道获得来的数据综合起来一起分析。

本节课，我们将学习如何利用Pandas合并多个DataFrame数据，以及筛选我们心仪的数据。

本节知识点

数据的合并

数据的筛选

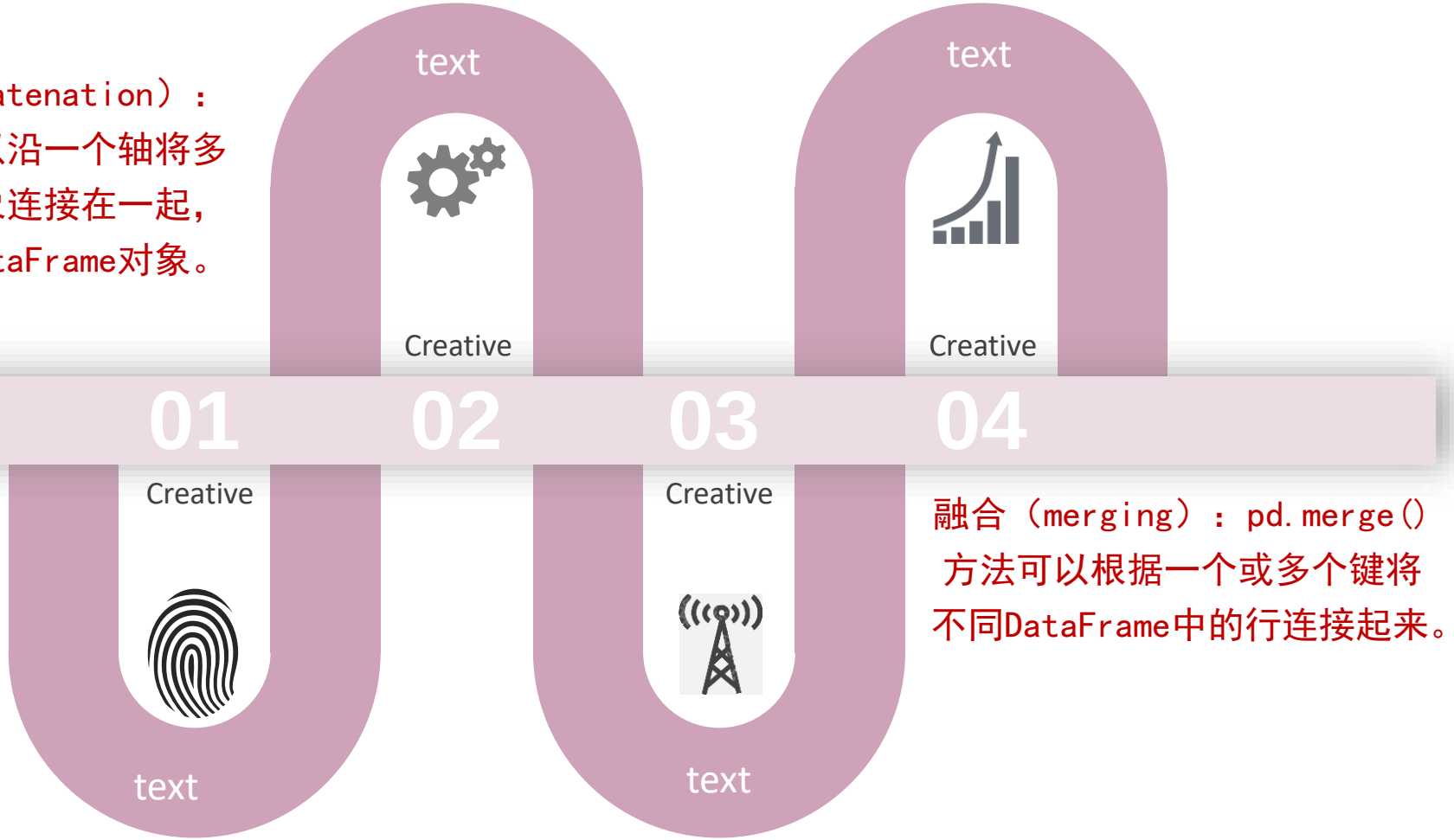
数据的排序

数据的合并

数据的合并

数据合并主要包括下面两种操作：

轴向连接（concatenation）：pd.concat()可以沿一个轴将多个DataFrame对象连接在一起，形成一个新的DataFrame对象。



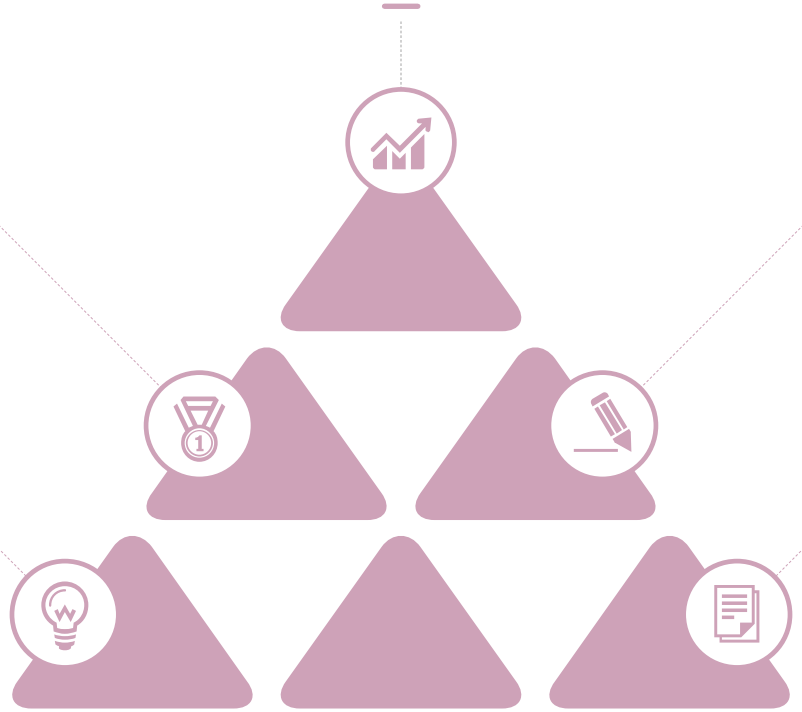
融合（merging）：pd.merge()方法可以根据一个或多个键将不同DataFrame中的行连接起来。

01

数据的合并

The Research Progress

concat()函数可以将数据根据不同的轴作进行合并。我们先看一下concat()的常用参数：



代码片段

```
pd.concat(objs, axis=0, join='outer')
```

axis： 需要合并链接的轴，0是行，1是列，默认是0。

objs: series、dataframe或者是panel构成的序列list。

join：连接的方式inner，或者outer，默认是outer。

01 数据的合并

```
In [1]:
```

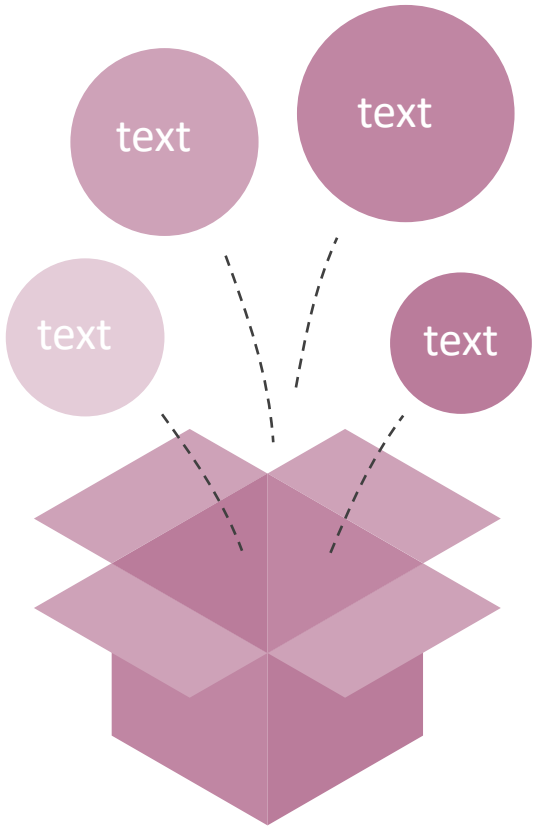
```python
import pandas as pd
dict1={
    'A': ['A0', 'A1', 'A2', 'A3'],
    'B': ['B0', 'B1', 'B2', 'B3'],
    'C': ['C0', 'C1', 'C2', 'C3']}
df1=pd.DataFrame(dict1)
print(df1)

dict2={
    'B': ['B0', 'B1', 'B2', 'B3'],
    'C': ['C0', 'C1', 'C2', 'C3'],
    'D': ['D0', 'D1', 'D2', 'D3']}
df2=pd.DataFrame(dict2)
print(df2)
```

```
    A   B   C
0  A0  B0  C0
1  A1  B1  C1
2  A2  B2  C2
3  A3  B3  C3
    B   C   D
0  B0  C0  D0
1  B1  C1  D1
2  B2  C2  D2
3  B3  C3  D3
```

```
In [ ]:
```

01

数据的合并



当concat()使用默认参数合并df1和df2时：

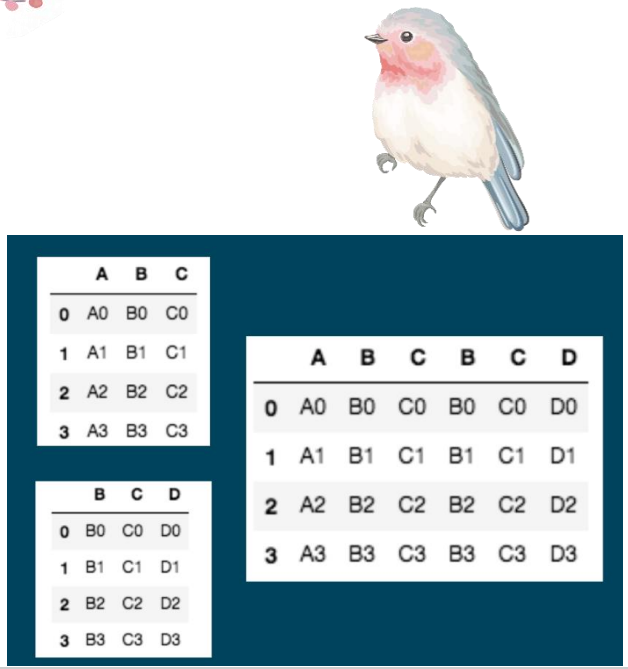
代码片段pd.concat(objs, axis=0, join='outer')

	A	B	C
0	A0	B0	C0
1	A1	B1	C1
2	A2	B2	C2
3	A3	B3	C3

	B	C	D
0	B0	C0	D0
1	B1	C1	D1
2	B2	C2	D2
3	B3	C3	D3

	A	B	C	D
0	A0	B0	C0	NaN
1	A1	B1	C1	NaN
2	A2	B2	C2	NaN
3	A3	B3	C3	NaN
0	NaN	B0	C0	D0
1	NaN	B1	C1	D1
2	NaN	B2	C2	D2
3	NaN	B3	C3	D3

通过上面的结果可以发现，当join='outer'，axis参数为0时，列进行并集处理，纵向表拼接，缺失值由NaN填充，并且会保留原有数据的行索引。



	A	B	C
0	A0	B0	C0
1	A1	B1	C1
2	A2	B2	C2
3	A3	B3	C3

	B	C	D
0	B0	C0	D0
1	B1	C1	D1
2	B2	C2	D2
3	B3	C3	D3

	A	B	C	B	C	D
0	A0	B0	C0	B0	C0	D0
1	A1	B1	C1	B1	C1	D1
2	A2	B2	C2	B2	C2	D2
3	A3	B3	C3	B3	C3	D3

01.如果两个表的index都没有实际含义，使用ignore_index参数，置true，重新整理一个新的index。

02.代码实现

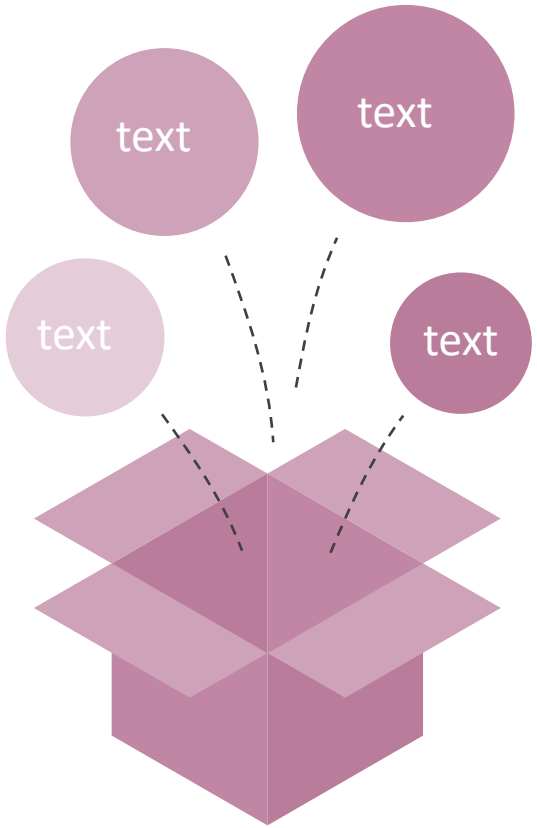
```
pd.concat([df1,df2],axis=0,join='outer',ignore_index=True)
```

03.当concat()的axis参数为1合并df1和df2时：

04.代码实现pd.concat([df1,df2],axis=1,join='outer')

05.当join='outer'，axis参数为1时，行进行并集处理，横向表拼接，缺失值由NaN填充。

01 数据的合并



当concat()的join参数为inner时合并df1和df2时：

代码片段

```
pd.concat([df1, df2], axis=0, join='inner')
```

	A	B	C
0	A0	B0	C0
1	A1	B1	C1
2	A2	B2	C2
3	A3	B3	C3

	B	C	D
0	B0	C0	D0
1	B1	C1	D1
2	B2	C2	D2
3	B3	C3	D3

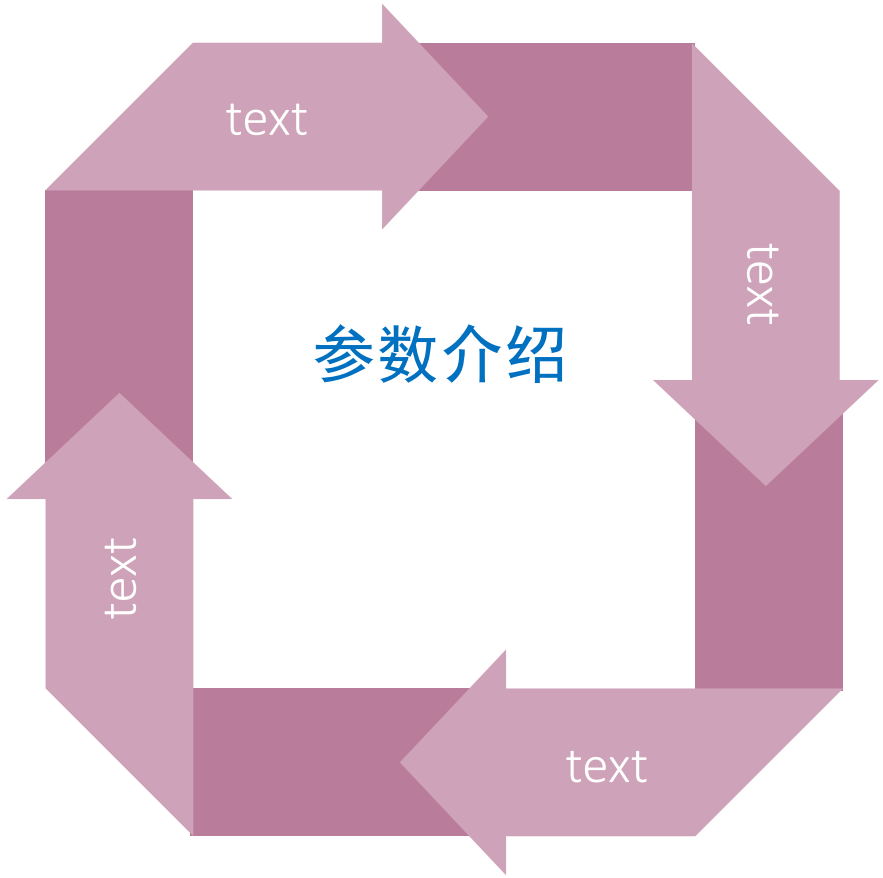
	B	C
0	B0	C0
1	B1	C1
2	B2	C2
3	B3	C3
4	B0	C0
5	B1	C1
6	B2	C2
7	B3	C3

通过上面的结果可以看出，如果为inner，得到的是两表的交集，如果是outer，得到的是两表的并集。

01 数据的合并

merge()函数通过指定连接键拼接列数据，我们先看一下merge的常用参数：

代码片段merge(left, right, how='inner', on=None)



1. left和right：两个要合并的DataFrame
2. how：连接方式，有inner、left、right、outer，默认为inner
3. on：指的是用于连接的列索引名称，必须存在于左右两个DataFrame中，如果没有指定且其他参数也没有指定，则以两个DataFrame列名交集作为连接键

01 数据的合并

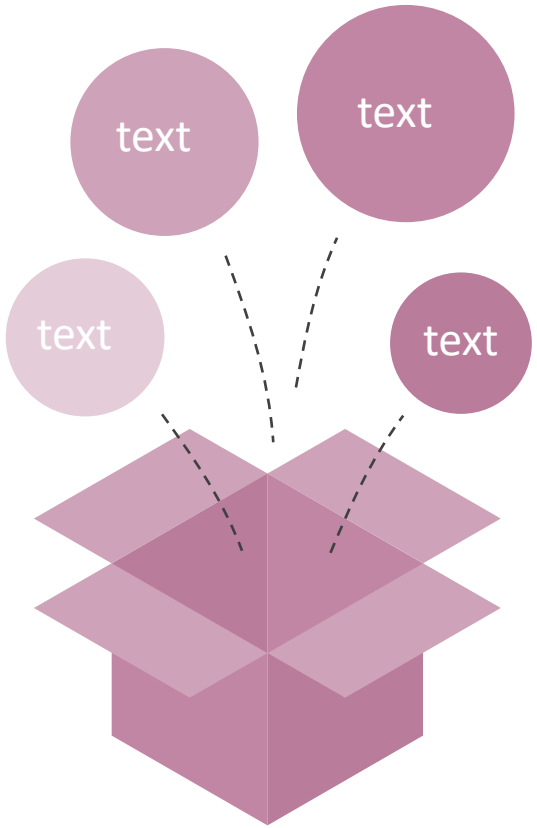
运行右面代码，了解数据的基本情况。

```
In [1]: import pandas as pd
        left = pd.DataFrame({'key':['a','b','b','d'],'data1':range(4)})
        print(left)

        right = pd.DataFrame({'key':['a','b','c'],'data2':range(3)})
        print(right)
```

```
  key  data1
0   a      0
1   b      1
2   b      2
3   d      3
  key  data2
0   a      0
1   b      1
2   c      2
```

01

运行结果



当merge()使用默认参数连接两个DataFrame时：

代码片段pd.merge(left, right)

	key	data1
0	a	0
1	b	1
2	b	2
3	d	3

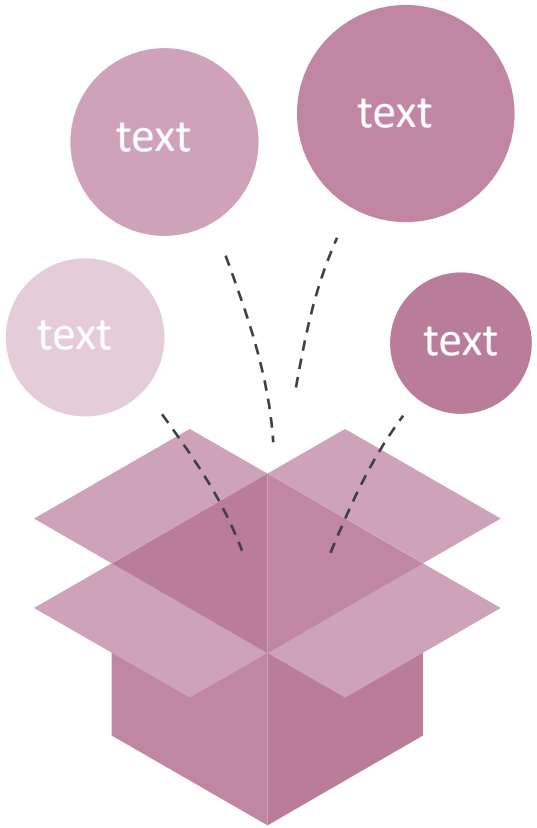
	key	data2
0	a	0
1	b	1
2	c	2

	key	data1	data2
0	a	0	0
1	b	1	1
2	b	2	1

merge()默认做inner连接，并且使用两个DataFrame的列名交集（key）作为连接键，同样，最终连接的数据也是两个DataFramekey列数据的交集。

01

运行结果



当两个DataFram使用做outer连接时：

代码片段`pd.merge(left,right,on=['key'],how='outer')`

	key	data1
0	a	0
1	b	1
2	b	2
3	d	3

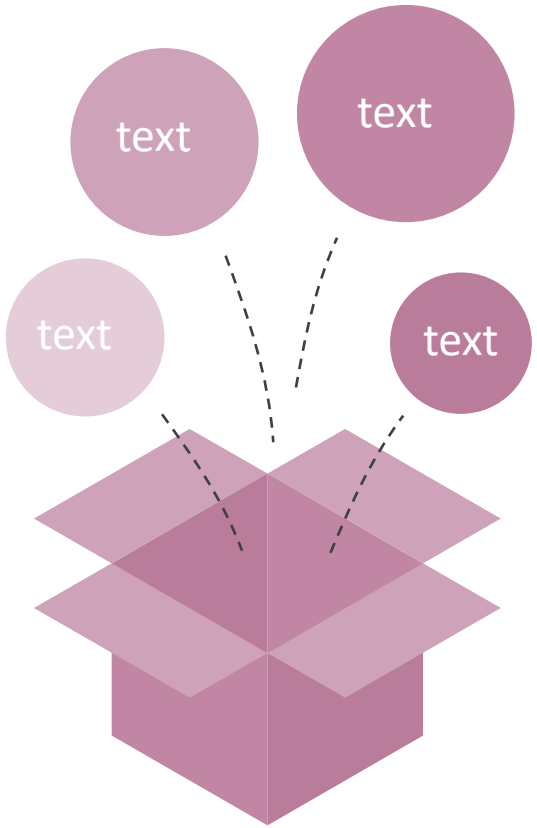
	key	data2
0	a	0
1	b	1
2	c	2

	key	data1	data2
0	a	0.0	0.0
1	b	1.0	1.0
2	b	2.0	1.0
3	d	3.0	NaN
4	c	NaN	2.0

当merge()做outer连接时最终连接的数据是两个DataFramekey列数据的并集，缺失的内容由NaN填充。

01

运行结果



当两个DataFram使用left做连接时：

代码片段`pd.merge(left,right,on=['key'],how='left')`

	key	data1
0	a	0
1	b	1
2	b	2
3	d	3

	key	data2
0	a	0
1	b	1
2	c	2

	key	data1	data2
0	a	0	0.0
1	b	1	1.0
2	b	2	1.0
3	d	3	NaN

当merge()做left连接时，最终连接的数据将以left数据的链接键为准合并两个数据的列数据，缺失的内容由NaN填充

那么，当merge()做right连接时，最终的链接数据是什么样呢？运行下面的代码，验证你的想法：

```
In [1]:
```

```
1  import pandas as pd
2  left = pd.DataFrame({'key':['a','b','b','d'],'data1':range(4)})
3  right = pd.DataFrame({'key':['a','b','c'],'data2':range(3)})
4  result = pd.merge(left,right,on=['key'],how='right')
5  print(result)
```

```
  key  data1  data2
0   a    0.0      0
1   b    1.0      1
2   b    2.0      1
3   c    NaN      2
```

01 数据的合并

上面我们了解两种合并数据的方式，
接下来我们看一下他们的应用场景：

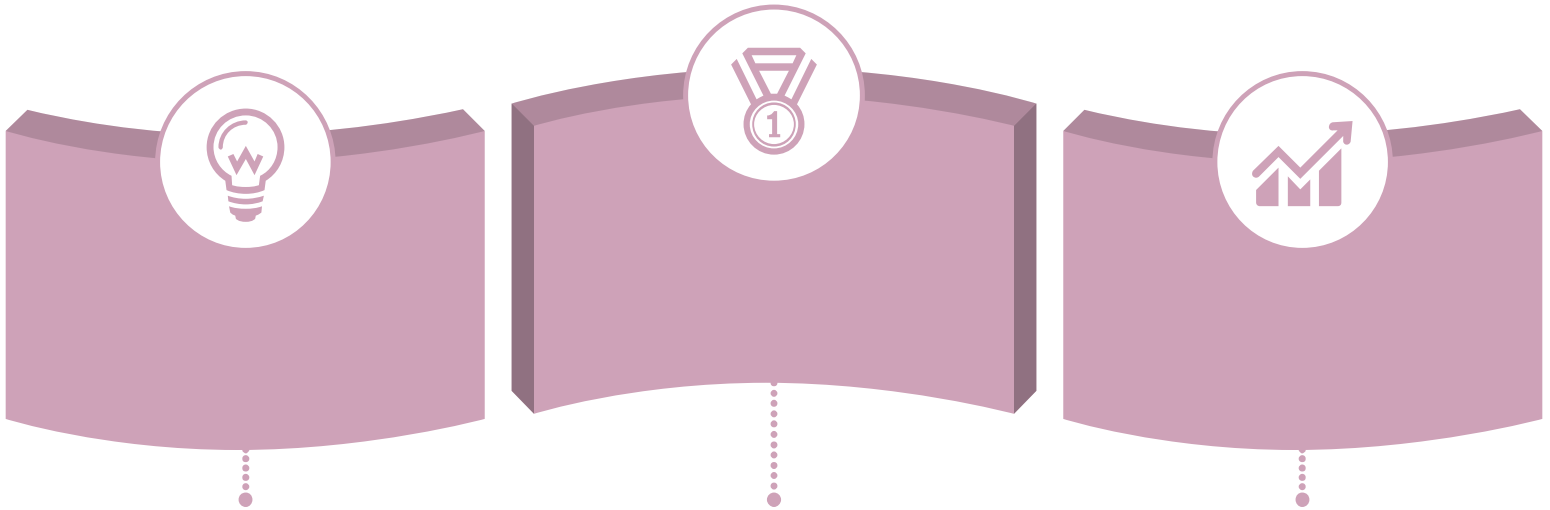
例如： 现在有两张表格分别存储了9月和10月份的成交信息，那么这个时候我们就可以使用concat( )将两个表沿着0轴合并。

例如： 现在有两张表格，一个是成交信息，包含订单号、金额、客户ID等信息；第二个是客户信息，包含客户ID、姓名、电话号等信息，那么这个时候我们就可以使用merge()根据客户ID将两个表合并成一个完整的表。

二

数据的筛选

02 数据的筛选



前面的课程中，我们学习了如何获取一条数据或者连续的多条数据，但是实际工作中我们经常需要处理上万条数据，特别是合并后的数据甚至上亿条，那么我们如何能快速筛选出符合条件的数据呢？

接下来，我们用某网站2017年用户数据为实验数据，来筛选我们想要的数据，路径为/data/course_data/data_analysis/mouhu_users_2017.csv。

数据详情：id、关注的收藏夹数量、关注数量、关注者数量（粉丝数）、关注的问题数量、关注的话题数量、关注的专栏数量。

运行下面代码，了解数据的基本情况。

```
In [1]:
1  import pandas as pd
2  df = pd.read_csv('/data/course_data/data_analysis/mouhu_users_2017.csv')
3  df.head()
```

	_id	关注的收藏夹	关注	关注者	关注的问题	关注的话题	关注的专栏
0	587598f89f11daf90617fb7a	52	17	1	30	58	2
1	587598f89f11daf90617fb7c	27	73	15	87	26	1
2	587598f89f11daf90617fb7e	72	94	1	112	20	4
3	587598f89f11daf90617fb80	174	84	8	895	30	7
4	587598f89f11daf90617fb82	3	236	64	119	44	17

```
In [ ]:
1
```

了解了数据的基本情况之后，第一个需求是将关注者超过100的用户数据获取出来。

我们先来看看筛选逻辑，然后运行代码，验证筛选结果：

```
In [1]:
```

```python
import pandas as pd
df = pd.read_csv('/data/course_data/data_analysis/mouhu_users_2017.csv')
bools= df['关注者']>100
df1 = df[bools]
print(df1.shape)
print(df1)
```

```
(5501, 7)
                          _id  关注的收藏夹   关注   关注者  关注的问题  关注的话题  关注的专栏
79     587598f89f11daf90617fc18       8   111  3049   1216      45      17
101    587598f89f11daf90617fc44      61   123   205    670      24      18
112    587598f99f11daf90617fc5a       1   804   197    830      39      26
114    587598f99f11daf90617fc5e      44    62   201    314       8       3
121    587598f99f11daf90617fc6c      44   628  6597   1566     144      69
...                         ...     ...   ...   ...    ...     ...     ...
72694  5878399e9f11da09c8fe690e       3   172   219    466      68      11
72700  5878399e9f11da09c8fe691a      21    39   281     51      45       2
72713  5878399e9f11da09c8fe6934      49   635   248   1772     151     123
72726  5878399e9f11da09c8fe694e     447    62   728     14       1      21
72741  5878399e9f11da09c8fe696c    3546   390   142     25       4     126
```

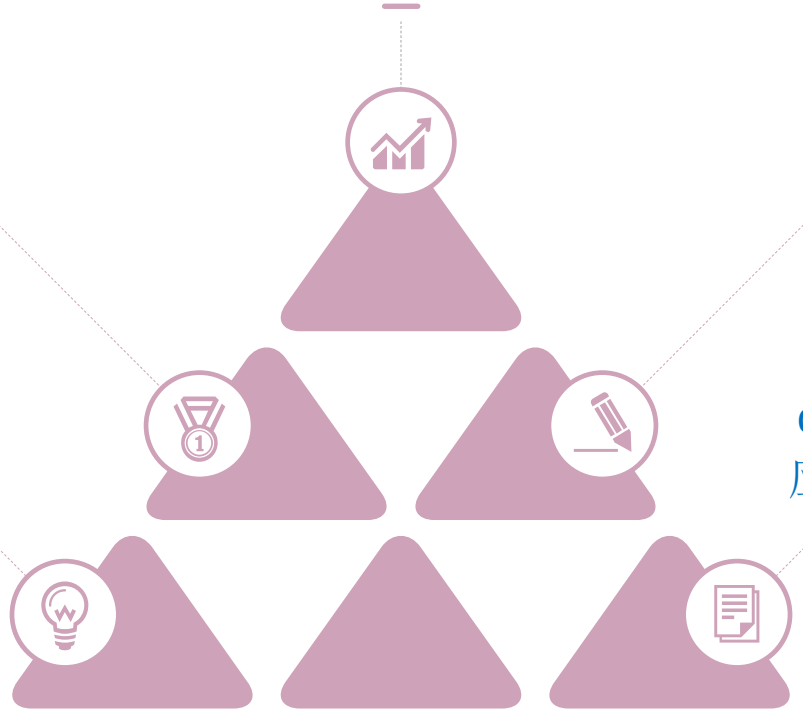
我们已经准确获取到所有关注者超过100的用户数据，下面我们看一下代码的逻辑。

03 课题研究的进展
The Research Progress

代码片段

```
bools= df['关注者']>100
```

首先判断每个用户的关注者数量是否大于100，大于则会返回True，表示该行被标记为True，否则被标记为False。bools记录了每一行是否符合筛选条件，是一个Series对象，其中的值是bool类型。



代码片段

```
df1 = df[bools]
```

然后，根据bools每行的值来对df进行筛选，值为True，表示对应的行会留下，否则，则去除。

最后打印的df1数据就是关注者超过100的用户数据。这是pandas根据某列的值进行筛选的基本逻辑。

根据某列的值进行筛选的逻辑我们已经掌握了，
如何进行多条件的联合筛选呢？

第二个需求是：获取关注者超过300并且关注的超过100的用户数据。运行下面的代码。

```
import pandas as pd
df = pd.read_csv('/data/course_data/data_analysis/mouhu_users_2017.csv')
bool1= df['关注者']>300
bool2= df['关注']>100
df2 = df[bool1 & bool2]
df2.head()
```

	_id	关注的收藏夹	关注	关注者	关注的问题	关注的话题	关注的专栏
79	587598f89f11daf90617fc18	8	111	3049	1216	45	17
121	587598f99f11daf90617fc6c	44	628	6597	1566	144	69
155	587598f99f11daf90617fcb0	194	250	1103	10	1	19
228	587598f99f11daf90617fd42	48	224	1450	360	128	20
261	587598f99f11daf90617fd84	1125	842	24848	108	33	27

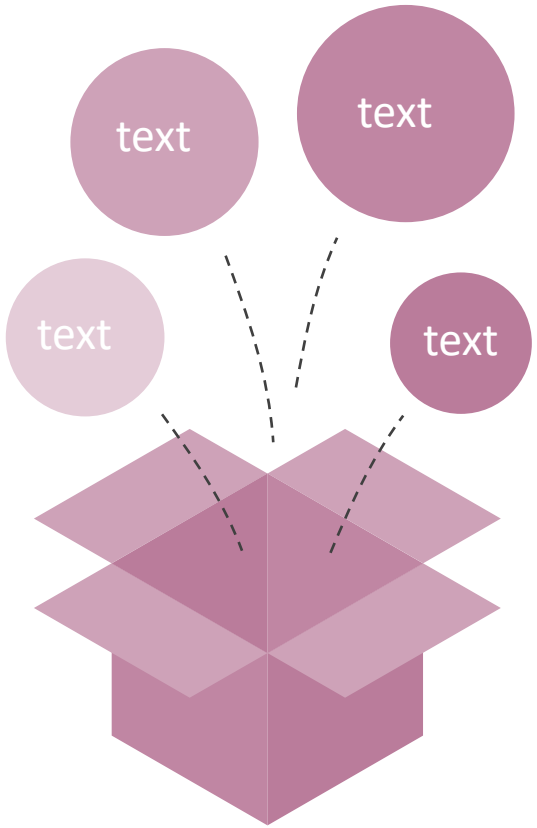
上面的这段代码里，我们通过了2个限制条件df['关注者']>300和 df['关注']>100，分别得到 bool1和bool2这2个Series。

在我们的需求中，需要的数据是同时满足两个条件，所以我们使用逻辑与运算连接两个值，最后获取同时满足两个条件的值。

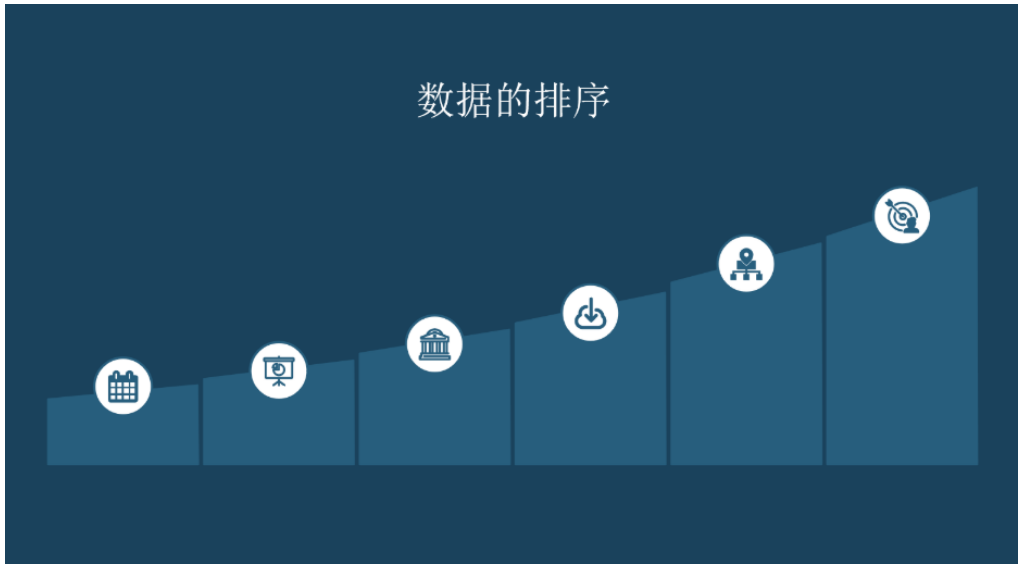
三

数据的排序

03 数据的排序



在数据获取过程中，数据的排序也是我们经常需要处理的问题。例如：我们需要找出关注者数量前十的用户信息。



可以使用sort_index()、sort_values()两个方法对数据进行排序，并且这两个方法Series和DataFrame都支持。

DataFrame的sort_index( )方法是按照行索引进行排序，sort_values()可以指定具体列进行排序。

接下来，我们使用的世界年龄抚养比率数据为行索引排序实验数据，是否还记得这个数据呢？运行下面的代码，我们一起回顾一下这个数据

In [1]:

```
import pandas as pd
df = pd.read_excel('/data/course_data/data_analysis/rate.xlsx')
print(df.shape)
print(df.head())
```

```
(219, 15)
      CountryName Country Code        1990        2000        2007  \
0     Afghanistan          AFG  101.094930  103.254202  100.000371
1         Albania          ALB   61.808311   59.585866   50.862987
2         Algeria          DZA   87.675705   62.886169   49.487870
3  American Samoa          ASM         NaN         NaN         NaN
4         Andorra          ADO         NaN         NaN         NaN

         2008        2009       2010       2011       2012       2013  \
0  100.215886  100.060480  99.459839  97.667911  95.312707  92.602785
1   49.663787   48.637067        NaN  46.720288  45.835739  45.247477
2   48.910002   48.645026  48.681853  49.233576  49.847713  50.600697
3         NaN         NaN        NaN        NaN        NaN        NaN
4         NaN         NaN        NaN        NaN        NaN        NaN

        2014       2015  Change 1990-2015  Change 2007-2015
0  89.773777  86.954464        -14.140466        -13.045907
1  44.912168  44.806973        -17.001338         -6.056014
2  51.536631  52.617579        -35.058127          3.129709
3        NaN        NaN               NaN               NaN
4        NaN        NaN               NaN               NaN
```

我们先来看一下，如何根据国家名称来进行排序，并且Country Code这一列被设置成了行索引。

```
In [1]:
import pandas as pd
df = pd.read_excel('/data/course_data/data_analysis/rate.xlsx',index_col='Country Code')
df.sort_index(inplace=True,ascending=True)
df.head()
```

Country Code	CountryName	1990	2000	2007	2008	2009	2010	201
ABW	Aruba	47.500831	44.420778	43.475981	43.957839	44.531061	45.157235	44.993173
ADO	Andorra	NaN	NaN	NaN	NaN	NaN	NaN	NaN
AFG	Afghanistan	101.094930	103.254202	100.000371	100.215886	100.060480	99.459839	97.667911
AGO	Angola	101.394722	100.930475	102.563811	102.609186	102.428788	102.035690	102.10675
ALB	Albania	61.808311	59.585866	50.862987	49.663787	48.637067	NaN	46.720288

通过结果发现，原来排序这么简单！接下来我们一起分析一下代码。

read_excel()中的参数index_col='Country Code'作用是在读取文件的时候指定Country Code这一列数据为行索引。

inplace=True参数和我们之前见过的作用一样，用来控制是否直接对原始数据进行修改。

ascending可以控制排序的顺序，默认值为True从小到大排列，当它被设置为False的时候就可以实现倒序排列。

03 数据的排序

现在就需要你马上动手更改上面的代码，实现数据根据索引进行倒序排列。

现在我们来完成第一个问题，获取某乎关注者数据前十的用户数据。那么，我们就需要根据关注者这一列数据进行排序。

我们成功的获取了关注者数据前十的用户数据，按照惯例我们一起分析一下代码。

by:决定了是按数据中的哪一列进行排序，将需要按照某列排序的列名赋值给by即可。

ascending=False:将数据按照从大到小的顺序排列。

inplace=True:用来控制是否直接对原始数据进行修改。

根据排序后的结果，我们发现最高的关注者数量已经达到了585787，并远远的看着第二名的356246，确有大V的潜质啊。

代码片段

```
import pandas as pd
df = pd.read_csv('/data/course_data/data_analysis/mouhu_users_2017.csv')
df.sort_values(by='关注者',ascending=False,inplace=True)
print(df.head(10))
```

04 本节总结

数据的合并、筛选和排序，是数据整理中比较重要的技能，就像将自行车变跑车，会大大提高你的工作效率，成功没有捷径，必须反复练习，勤于总结。下面我们对这节的知识点进行以下总结。

知识点回顾

数据的合并

1.pd.concat()沿一个轴将多个DataFrame对象连接在一起

2.axis参数值：0表示行，1表示列

3.join参数：outer并集处理，inner交集处理

4.merge()函数通过指定连接键拼接列数据

5.how：连接方式，有inner,left,right,outer，默认为inner

6.on:指的是要于连接的列索引名称

知识点回顾

数据的筛选

1.df['列名']>100，返回每一行是否符合该条件的Bool类型的Series

2.df[bool1&bool2],如果多个条件可以将判断条件之间使用逻辑运算符

知识点回顾

数据的排序

1.sort_index()是按照行索引进行排序

2.sort_values()可以指定具体列进行排序

3.sort_values的by参数:决定了是按数据中的哪一列进行排序

4.ascending=False:将数据按照从大到小的顺序排列，

默认True，升序

5.inplace=True:用来控制是否直接对原始数据进行修改

练习一一

最长的时间是多久

题目要求

现在我们有2017年度1月份和2月份的共享单车历史骑行数据，路径为/data/course_data/data_analysis/2017_1_data.csv和/data/course_data/data_analysis/2017_2_data.csv。

05 练习一解析

第一步:明确目标

现在我们有2017年度1月份和2月份的共享单车历史骑行数据，路径为/data/course_data/data_analysis/2017_1_data.csv和/data/course_data/data_analysis/2017_2_data.csv。

第二步:分析过程

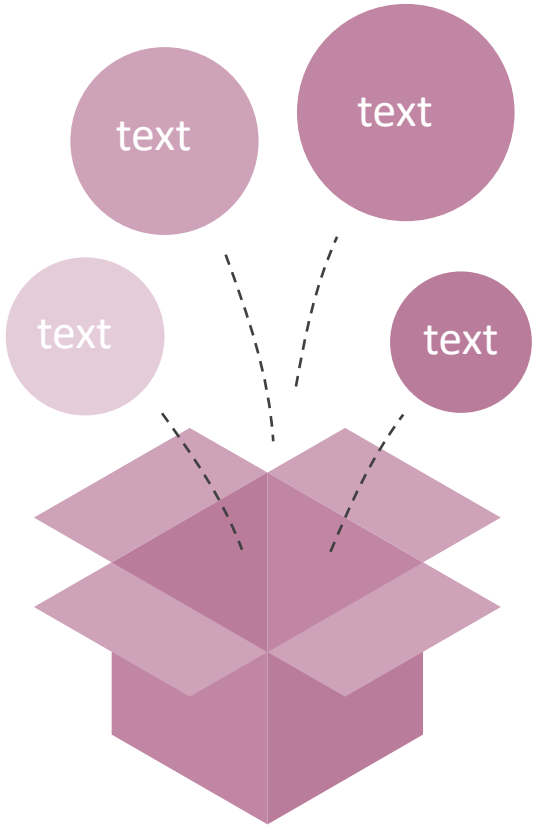
将两个csv数据合并成一个数据，并按骑行时间进行倒序排列，获取最长的骑行时间。

参数据描述

- Duration (ms)：骑行时间， 以毫秒为单位
- Start date：开始骑行时间
- End date：结束骑行时间
- Start station：开始地点
- Start date：开始骑行时间
- End station：结束地点
- Bike number：共享单车车号
- Member type：用户类别： 会员（Member/casual 非会员）

```python
import pandas as pd
df1 = pd.read_csv('/data/course_data/data_analysis/2017_1_data.csv')
print(df1.shape)
df2 = pd.read_csv('/data/course_data/data_analysis/2017_2_data.csv')
print(df2.shape)
# 合并数据
df3 = pd.concat([df1,df2])
print(df3.shape)
print(df3.head())
# 倒序排列时间
df3.sort_values(by='Duration (ms)',ascending=False,inplace=True)
print(df3.head())
# 获取最长时间
long_time = df3.iloc[0]['Duration (ms)']
print(df3.head())
print(long_time)
```

05 练习一运行结果




```
(42405, 7)
(72880, 7)
(115285, 7)
   Duration (ms)     Start date       End date  \
0         221834  2017/1/1 0:00  2017/1/1 0:04
1        1676854  2017/1/1 0:06  2017/1/1 0:34
2        1356956  2017/1/1 0:07  2017/1/1 0:29
3        1327901  2017/1/1 0:07  2017/1/1 0:29
4        1636768  2017/1/1 0:07  2017/1/1 0:34

                                Start station               End station  \
0                         3rd & Tingey St SE   M St & New Jersey Ave SE
1                           Lincoln Memorial             8th & D St NW
2  Henry Bacon Dr & Lincoln Memorial Circle NW  New York Ave & 15th St NW
3  Henry Bacon Dr & Lincoln Memorial Circle NW  New York Ave & 15th St NW
4                           Lincoln Memorial             8th & D St NW

  Bike number Member type
0      W00869      Member
1      W00894      Casual
2      W21945      Casual
3      W20012      Casual
4      W22786      Casual
       Duration (ms)      Start date        End date  \
24905       84876226  2017/1/6 10:59  2017/1/7 10:33
44077       84070936  2017/2/7 10:44  2017/2/8 10:05
35857       83789379  2017/2/6 12:35  2017/2/7 11:51
35874       83662979  2017/2/6 12:38  2017/2/7 11:53
29204       82897521  2017/1/7 11:42  2017/1/8 10:44

                           Start station                     End station  \
24905  17th & K St NW / Farragut Square                     19th & K St NW
44077                     Thomas Circle                     4th & C St SW
35857           Fleet St & Ritchie Pkwy  E Montgomery Ave & Maryland Ave
35874           Fleet St & Ritchie Pkwy  E Montgomery Ave & Maryland Ave
29204                     8th & H St NW           4th St & Madison Dr NW

      Bike number Member type
24905      W23232      Member
44077      W21615      Casual
35857      W22565      Casual
35874      W21846      Casual
29204      W21095      Casual
84876226
```

练习二

全国人口最多的城市

题目要求

根据第六次全国人口普查中的常住人口数据，获取全国人口最多的10个城市。数据信息：共包含省、地区、结尾、常住人口4个字段。文件路径为/data/course_data/data_analysis/liupu.csv

05 练习二解析及答案

第一步:明确目标

根据第六次全国人口普查中的常住人口数据，获取全国人口最多的10个城市。数据信息：共包含省、地区、结尾、常住人口4个字段。文件路径为/data/course_data/data_analysis/liupu.csv

第二步:分析过程

难点：根据结尾字段的数据，过滤掉省，保留市。

第三步:代码实现

```
In [1]:
```

```python
import pandas as pd
df = pd.read_csv('/data/course_data/data_analysis/liupu.csv')
# 查看数据
print(df.head())

# series.str会将每一个数据转换成字符串
# contains()判断字符串是否含有指定子串，返回的是bool类型
bools = df['结尾'].str.contains("市")

# 根据人口数倒序排列
df = df[bools].sort_values('常住人口',ascending=False)
# 获取前十个数据
print(df.head(10))
```

```
    省    地区 结尾        常住人口
0  安徽省   安徽省  省   59500468.0
1  安徽省   安庆市  市    5311379.0
2  安徽省   蚌埠市  市    3164467.0
3  安徽省   亳州市  市    4850657.0
4  安徽省   巢湖市  市    3873102.0
      省    地区 结尾        常住人口
567  上海市   上海市  市   23019196.0
23   北京市   北京市  市   19612368.0
569  四川省   成都市  市   14047625.0
601  天津市   天津市  市   12938693.0
73   广东省   广州市  市   12701948.0
157  河北省   保定市  市   11194382.0
232 黑龙江省  哈尔滨市  市  10635971.0
378  江苏省   苏州市  市   10459890.0
95   广东省   深圳市  市   10358381.0
204  河南省   南阳市  市   10263660.0
```

text here

text here

text here

text here

THANK YOU

欢迎进入下一章节的学习