



文本分析

第4讲

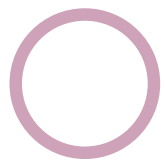
李田雨





数据七十二变





数据七十二变



在上一节我们讲了Pandas对数据的永久性保存以及对本地数据的读取。

我们分析的数据来源有很多种，例如：爬取、公司数据库、数据公司等。但是这些数据中有些数据项是我们不需要的，甚至可能会存在重复数据和空值的情况。

所以，本节我们将继续讲解数据七十二变，看一下数据是如何删除多余的数据和重复数据以及空值的处理。



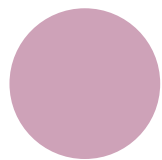
本节知识点

删除行和列

重复数据的处理

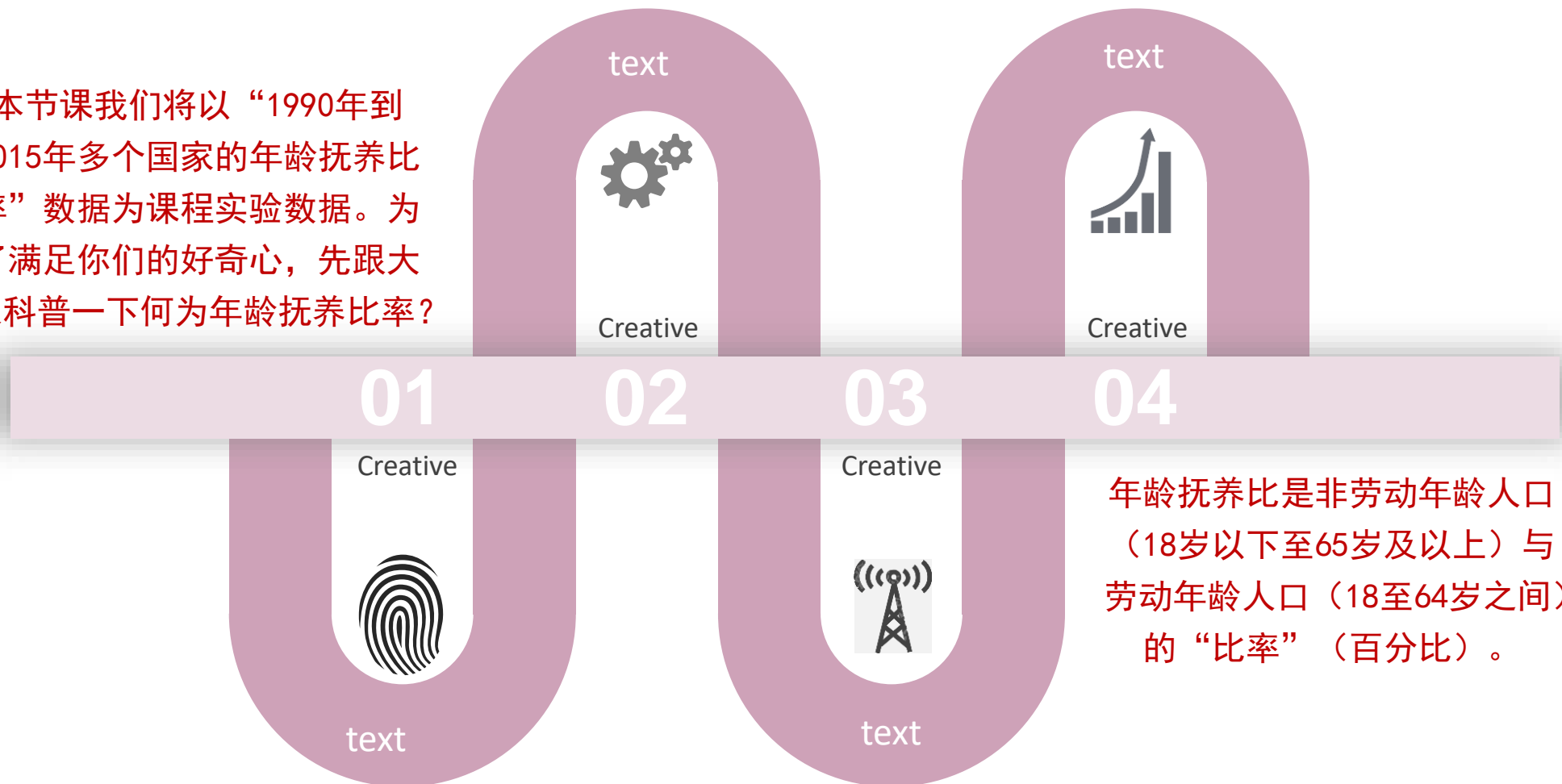
空值的处理





数据七十二变

本节课我们将以“1990年到2015年多个国家的年龄抚养比率”数据为课程实验数据。为了满足你们的好奇心，先跟大家科普一下何为年龄抚养比率？



年龄抚养比是非劳动年龄人口（18岁以下至65岁及以上）与劳动年龄人口（18至64岁之间）的“比率”（百分比）。



删除数据



数据的路径为/data/course_data/data_analysis/rate.xlsx，现在我们读取文件，了解一下数据的基本情况，运行下方代码查看结果：

```
+  ✂  📄  ⬆  ⬇  ▶ 运行

In [1]: 1 import pandas as pd
        2 df = pd.read_excel('/data/course_data/data_analysis/rate.xlsx')
        3 print(df.shape)
        4 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
```



	Country Name	Country Code	1990	2000	2007	2008	2009	2010	2011
0	Afghanistan	AFG	101.094930	103.254202	100.000371	100.215886	100.060480	99.459839	97.667911
1	Albania	ALB	61.808311	59.585866	50.862987	49.663787	48.637067	NaN	46.720288
2	Algeria	DZA	87.675705	62.886169	49.487870	48.910002	48.645026	48.681853	49.233576
3	American Samoa	ASM	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	Andorra	ADO	NaN	NaN	NaN	NaN	NaN	NaN	NaN

在结果中我们发现有多個NaN，那NaN表示的是什麼數據呢？

如果文件的單元格中沒有值時，在使用pandas讀取後就會用NaN表示，也就是我們常說的空值。



01

删除数据

在NumPy模块中提供了nan的值，如果你想要创建一个空值，可以使用下方代码：



代码实现

```
1 print(NaN+1)
```

所以，当我们不知道的情况下会影响我们的计算结果。



代码实现

```
print(type(NaN))
```

但是，当NaN可以参与到数据计算中，最终的结果却永远都是NaN。



代码实现

```
1 from numpy import nan as NaN
```

而且需要注意的是，NaN比较特殊点就是其本身是一种float类型数据。

请抄写代码到下面的代码框运行：



```
+ ✂ 📄 📁 ⬆ ⬇ ▶ 运行
```

```
In [1]: 1 from numpy import nan as NaN
        2 print(type(NaN))
        3 print(NaN+1)
```

```
<class 'float'>
nan
```

对于大批量的Series数据，使用肉眼很难判断空值的存在，这时我们可以先对空值进行过滤。请运行下方代码，查看过滤的方法：



```
In [1]: 1 from numpy import nan as NaN
        2 import pandas as pd
        3 se=pd.Series([4,NaN,8,NaN,5])
        4 print(se.notnull())
        5 print(se[se.notnull()])
        6
        7
        8 df = pd.read_excel('/data/course_data/data_analysis/rate.xlsx')
        9 print(df.shape)
       10 print(df.head())
       11
       12
       13 se=pd.Series([4,NaN,8,NaN,5])
       14 print(se.notnull())
       15 print(se[se.notnull()])
```

01

删除数据



通过结果我们发现，结果中依然存在空值，并没有过滤掉空值。

```
0    True
1   False
2     True
3   False
4     True
dtype: bool
0    4.0
2    8.0
4    5.0
dtype: float64
(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

0    True
1   False
2     True
3   False
4     True
dtype: bool
0    4.0
2    8.0
4    5.0
dtype: float64
```


01

删除数据



编辑标题

我们也可以使用**thresh**参数筛选想要删除的数据，**thresh=n**保留至少有n个非NaN数据的行。



如果想要对列做删除操作，需要添加**axis**参数，**axis=1**表示列，**axis=0**表示行。



代码片段

```
1 df1 = df.dropna()
```

dropna()是删除空值数据的方法，默认将只要含有**NaN**的整行数据删掉，如果想要删除整行都是空值的数据需要添加**how='all'**参数。

所以在**DataFrame**类型数据中，一般我们会将存在**NaN**的数据使用**dropna()**方法全部删掉：



到这里有同学会问，
如果我只是单纯的想删除两行数据该怎么做呢？






如果只是单纯的想删除数据，我们可以使用`df.drop()`方法，一起来了解一下该函数。

代码片段 `DataFrame.drop(labels=None,axis=0, index=None, columns=None, inplace=False)`



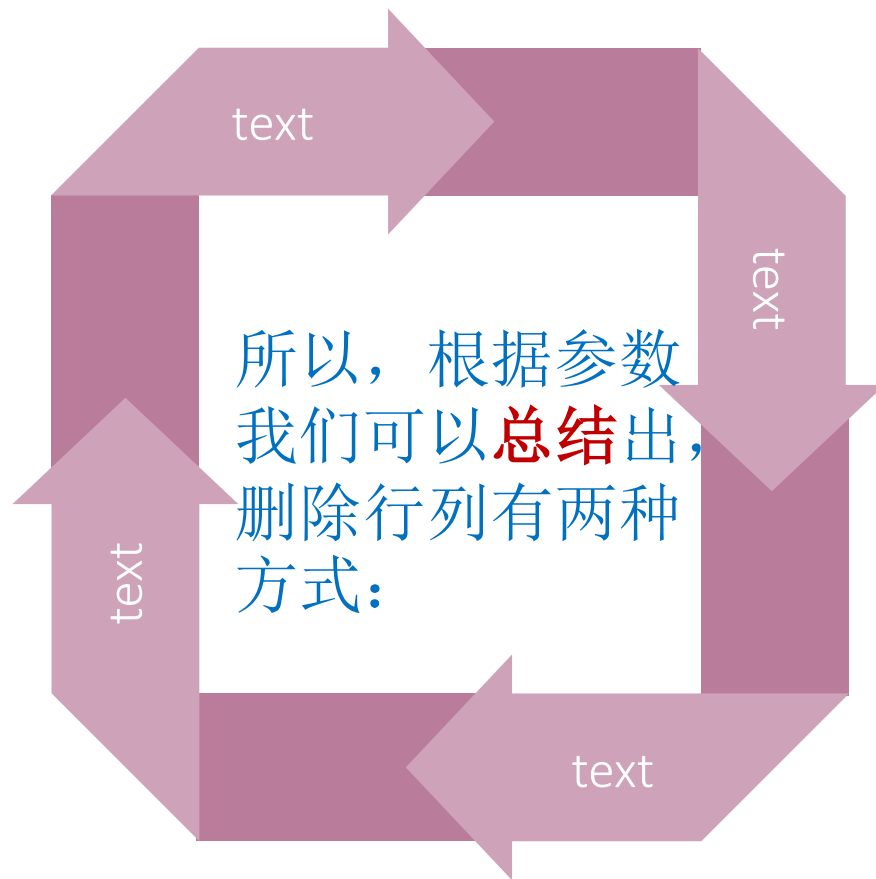
代码解释

- 01. `labels`：就是要删除的行列的名字，用列表给定。
 - 02. `index`：直接指定要删除的行。
 - 03. `columns`：直接指定要删除的列
 - 04. `inplace=False`：默认该删除操作不改变原数据，而是返回一个执行删除操作后的新`dataframe`。
 - 05. `inplace=True`：则会直接在原数据上进行删除操作，删除后无法返回。
- 

01

删除数据

1.labels=None,axis=0
的组合



2.index 或 columns 直接
指定要删除的行或列

01

删除数据

```
1 import pandas as pd
2 df = pd.read_excel('/data/course_data/data_analysis/rate.xlsx')
3 # 删除第0行和第1行
4 # df.drop(labels=[0,1],axis=0)
5
6 # 删除列名为1990的列
7 df.drop(axis=1,columns=1990)
```

在代码框中，尝试删除一行数据和一列数据



代码片段



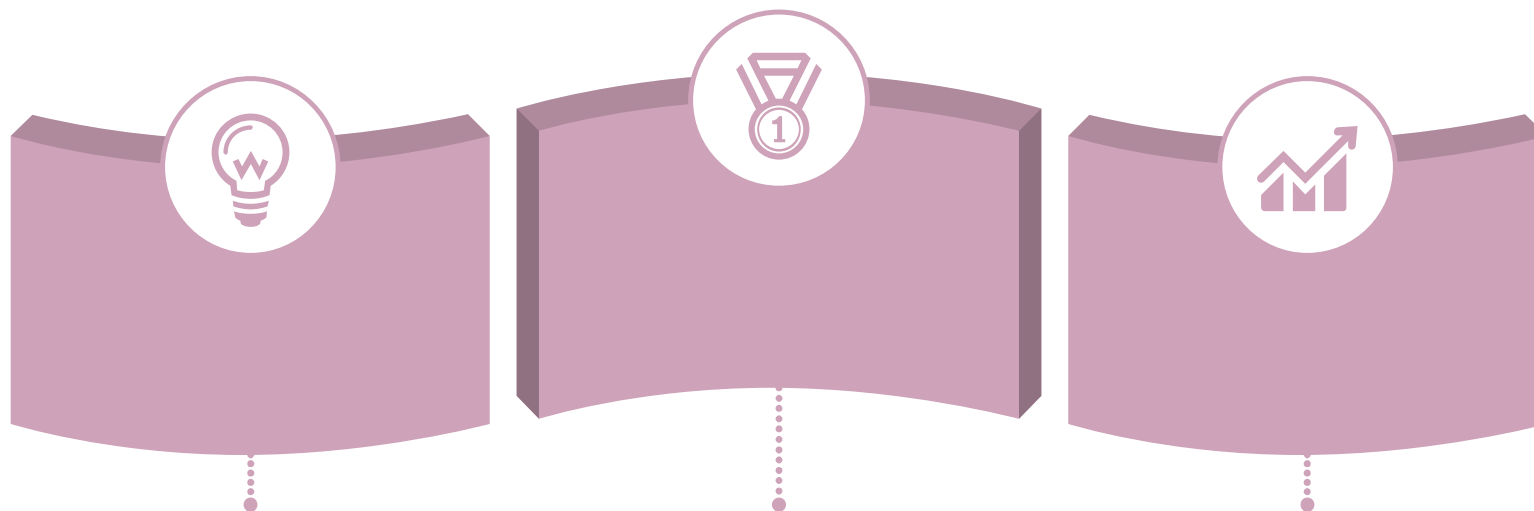
二

空值的处理



02

空值的处理



对于空值我们可以将整条数据删除，也可以使用`fillna()`方法对空值进行填充。

代码片段

```
df.fillna(value=None, method=None, axis=None, in
place=False, limit=None, downcast=None, **kwargs)
```

注意： `method` 参数不能与 `value` 参数同时出现。



三

重复数据的处理



03

重复数据的处理

重复数据的存在有时不仅会降低分析的准确度，也会降低分析的效率。所以我们在整理数据的时候应该将重复的数据删除掉。

利用 `duplicated()`函数可以返回每一行判断是否重复的结果（重复则为`True`）。

```
In [1]: import pandas as pd
        df = pd.read_excel('/data/course_data/data_analysis/rate.xlsx')
        # 返回重复的结果
        print(df.duplicated())

0      False
1      False
2      False
3      False
4      False
...
214    False
215    False
216    False
217    False
218    False
Length: 219, dtype: bool
```

运行左面的代码，观察结果是否有重复数据：

通过结果我们发现，返回的是一个值为`Bool`类型的`Series`，如果当前行所有列的数据与前面的数据是重复的就返回`True`；反之，则返回`False`。

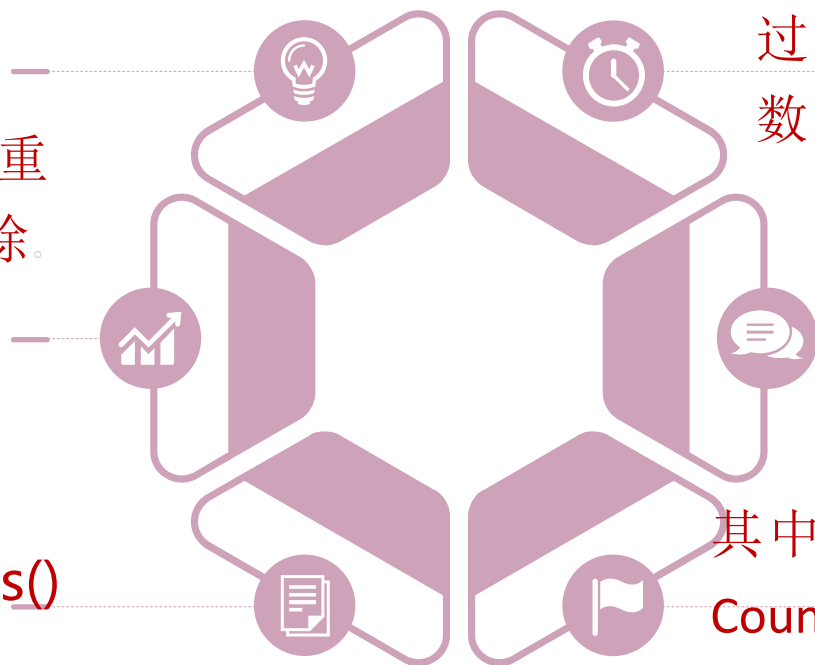
03

重复数据的处理

可以使用
`drop_duplicates()`将重
复的数据行进行删除。

代码片段

`df.drop_duplicates()`



我们也可以只可以通过判断某一列的重复数据，然后进行删除。

代码片段

```
df.drop_duplicates(['CountryName'], inplace=False)
```

其中['CountryName']表示对比CountryName例数据是否有重复，inplace用来控制是否直接对原始数据进行修改。



本节总结

第四课 知识点总结

删除数据

`df.drop()`删除一行或者一列

`axis`共有0/1两个值,0标识删除行,1标识删除列,默认0

`inplace`:是否在当前`df`中执行此操作;
`True`表示在原来数据技术上修改;
`Flase`表示返回一个新的值,不修改原有数据

空值的处理

`df.dropna()`只要含有`NaN`的整行数据删掉

参数`how='all'`,删除整行都是空值的数据

`axis`参数, `axis=1`表示列, `axis=0`表示行

参数`thresh=n`保留至少有`n`个非`NaN`数据的行

移除重复数据

`uplicated()`检测数据是否为重复的数据行

`a`通过`DataFrame`的`uplicated()`只判断某一列数据中是否重复

`f.drop_duplicates(['A'])`去除重复行数据



知识点回顾

删除数据

1.df.drop()删除一行或者一列

2.axis共有0/1两个值，0表示删除行，1表示删除列，默认0

3.inplace:

是否在当前df中执行此操作；

True表示在原来数据基础上修改；

False表示返回一个新的值，不修改原有数据。

知识点回顾

空值的处理

1. `df.dropna()` 只要含有NaN的整行数据删掉
2. 参数 `how='all'`, 删除整行都是空值的数据
3. `axis` 参数, `axis=1` 表示列, `axis=0` 表示行
4. 参数 `thresh=n` 保留至少有n个非NaN数据的行

知识点回顾

移除重复数据

1. `df.duplicated()` 监测数据是否为重复的数据行
2. 通过 `DataFrame` 的 `duplicated()` 去除重复行数据
3. `f.drop_duplicates(['A'])` 只判断某一列数据中是否重复



练习

如果你不需要，就让它消失！

题目要求

从自2009-2010赛季以来的英格兰当地足球比赛结果数据中，删除2009/2010赛季的所有数据以及country列。文件路径为/data/course_data/data_analysis/score_game.xlsx



03 解析

第一步:明确目标 ▲

从自2009-2010赛季以来的英格兰当地足球比赛结果数据中，删除2009/2010赛季的所有数据以及country列。文件路径为/data/course_data/data_analysis/score_game.xlsx

第二步:分析过程 ▲

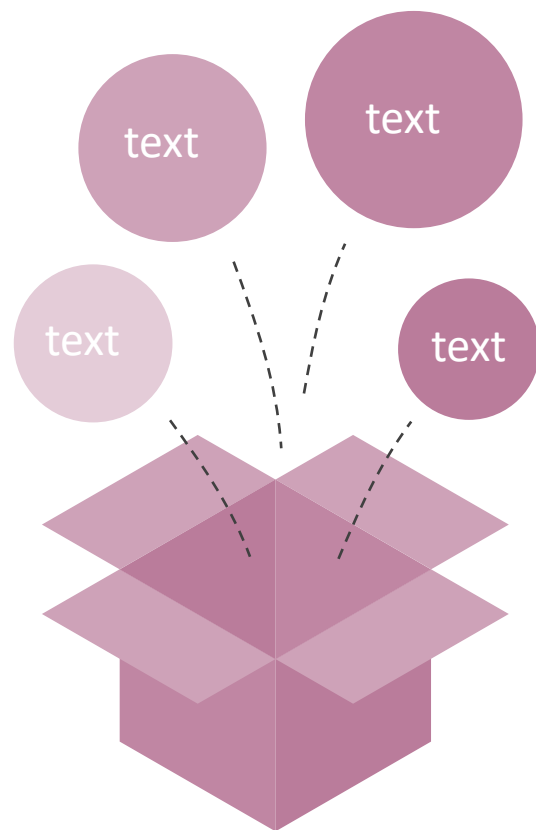
名词解释

- game_date-比赛时间
- country-国家
- tournament-锦标赛
- season-赛季
- home_field-主场场地
- home_team-主场球队
- away_team-客场球队
- home_team_score-主场球队得分
- away_team_score-客场球队得分
- home_team_score_extra_time-主场球队加时赛
- away_team_score_extra_time-客场球队加时赛

```
In [1]: 1 import pandas as pd
2 df = pd.read_excel('/data/course_data/data_analysis/score_game.xlsx')
3 print(df.shape)
4
5 # 通过遍历获取2009/2010赛季的数据索引（下节课我们将学习其他获取方式）
6 index_list = []
7 for index, row_data in df.iterrows():
8     if row_data['season'] == '2009/2010':
9         index_list.append(index)
10
11 # 删除2009/2010赛季的数据
12 df1 = df.drop(index_list, axis=0)
13 # print(df1)
14
15 # 删除country列
16 df2 = df1.drop('country', axis=1)
17 print(df2)
```

05

运行结果



	game_date	tournament	season	home_field	home_team \
36	2009-08-09	FA Community Shield	2009/2009	False	Chelsea
2632	2010-08-06	Championship	2010/2011	True	Norwich
2633	2010-08-07	League Two	2010/2011	True	Bury
2634	2010-08-07	League Two	2010/2011	True	Crewe
2635	2010-08-07	League Two	2010/2011	True	Gillingham
...
27485	2019-11-09	FA Cup	2019/2020	True	Colchester
27486	2019-11-09	FA Cup	2019/2020	True	Cheltenham
27487	2019-11-09	FA Cup	2019/2020	True	Carshalton
27488	2019-11-09	FA Cup	2019/2020	True	MK Dons
27489	2019-11-09	FA Cup	2019/2020	True	Tranmere

	away_team	home_team_score	away_team_score \
36	Manchester Utd	2	2
2632	Watford	2	3
2633	Port Vale	0	1
2634	Hereford	0	1
2635	Cheltenham	1	1
...
27485	Coventry	0	2
27486	Swindon	1	1
27487	Boston Utd	1	4
27488	Port Vale	0	1
27489	Wycombe	2	2

	home_team_score_extra_time	away_team_score_extra_time
36	1.0	0.0
2632	NaN	NaN
2633	NaN	NaN
2634	NaN	NaN
2635	NaN	NaN
...
27485	NaN	NaN
27486	NaN	NaN
27487	NaN	NaN
27488	NaN	NaN
27489	NaN	NaN

[24859 rows x 10 columns]



练习

算你狠

题目要求

观察文件 (/data/course_data/data_analysis/04Books.xlsx)
中的商品的单价 (OnePrice) 和购买数量 (Count)



02

课题研究的思路与方法 Ideas And Methods

第一步:明确目标 ▲

观察文件 (/data/course_data/data_analysis/04Books.xlsx) 中的商品的单价 (OnePrice) 和购买数量 (Count)



第二步:分析过程 ▲

计算出对应的总价 (Price) 。



```
# 计算Price的值(这种方法是列与列之间对齐后进行计算)
# books["Price"] = books['OnePrice'] * books['Count']
# print(books)
# 如果只想算某一段就可以，使用循环迭代（是单元格与单元格之间的操作）
# for i in range(5,16):
#     books["Price"].iloc[i] = books["OnePrice"].iloc[i] * books["Count"].iloc[i]
# print(books)
```

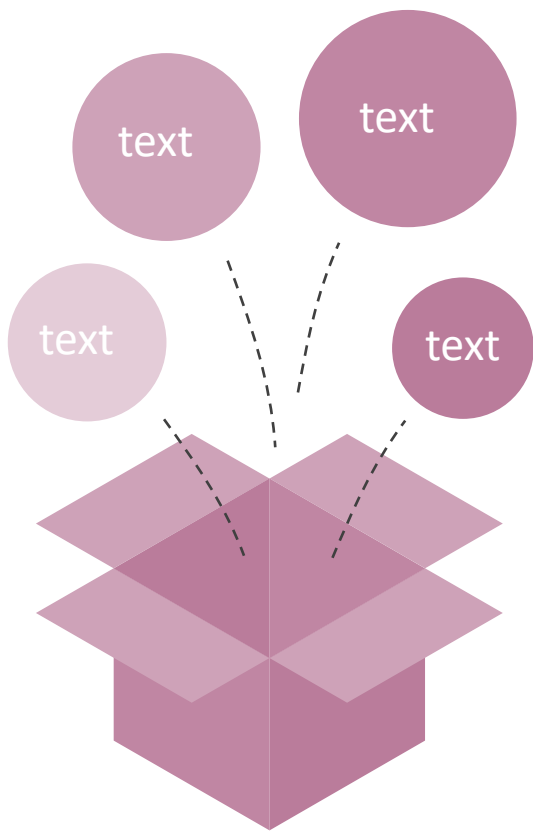


编辑标题

```
import pandas as pd
books = pd.read_excel('/data/course_data/data_analysis/04Books.xlsx', index_col = 'ID')
print(books)
```

05

运行结果



	Name	OnePrice	Count	Price
ID				
1	Product_001	9.82	5	NaN
2	Product_002	11.99	4	NaN
3	Product_003	9.62	6	NaN
4	Product_004	11.08	8	NaN
5	Product_005	7.75	3	NaN
6	Product_006	7.34	4	NaN
7	Product_007	10.97	6	NaN
8	Product_008	11.14	7	NaN
9	Product_009	8.98	2	NaN
10	Product_010	9.18	3	NaN
11	Product_011	8.31	4	NaN
12	Product_012	7.29	9	NaN
13	Product_013	8.36	5	NaN
14	Product_014	9.16	6	NaN
15	Product_015	10.31	3	NaN
16	Product_016	10.26	6	NaN
17	Product_017	11.95	8	NaN
18	Product_018	11.22	2	NaN
19	Product_019	10.95	4	NaN
20	Product_020	8.82	6	NaN



THANK YOU

欢迎各位认真听讲

