

Pandas教程

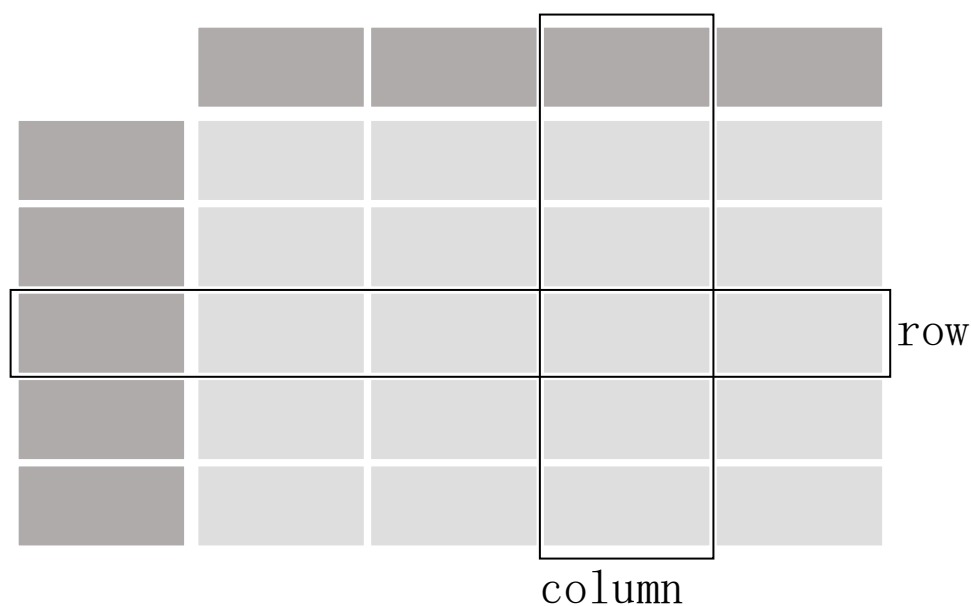
1、安装和导入

- 可以这样导入pandas，习惯上，使用pd作为缩写

```
1  #安装pandas
2  #pip install pandas
3
4  import pandas as pd
```

2、Dataframe

DataFrame



- 小例子，我们输入一些姓名、年龄、性别的信，在一个表格中，运行下面代码：

```
1  df = pd.DataFrame(
2      {
3          "Name": [
4              "小明",
5              "大熊",
6              "张三",
7          ],
8          "Age": [22, 35, 58],
9          "Sex": ["male", "male", "female"],
10     }
11 )
12 print(df)
13 ...
```

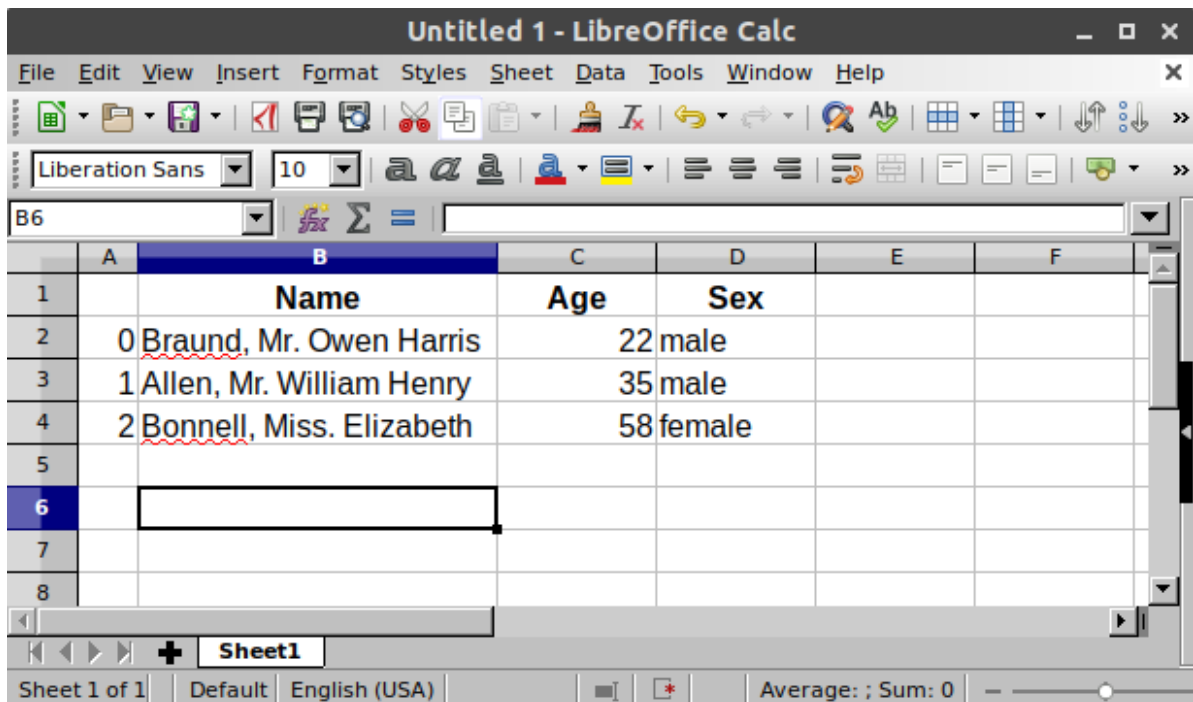
```
14  输出:
15      Name  Age    Sex
16  0  小明   22    male
17  1  大熊   35    male
18  2  张三   58    female
19  ' '
20
```

要手动将数据存储表中，请创建一个“`DataFrame`”。使用 Python 列表字典时，字典键将用作列标题，每个列表中的值将用作“`DataFrame`”的列。

`DataFrame` 是一种二维数据结构，可以存储不同类型的数据（包括字符、整数、浮点值、分类数据等）列中，它类似于电子表格、SQL 表。- 该表有 3 列，每列都有一个列标签。列标签分别是“姓名”、“年龄”和“性别”。

- `Name` 列由文本数据组成，每个值都是一个字符串，
- `Age` 列是数字，
- `Sex` 列是文本数据。

在电子表格软件中，我们数据的表格表示看起来非常相似：



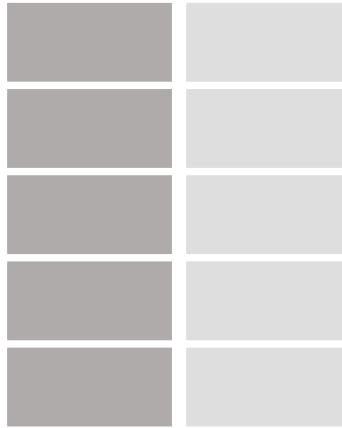
The screenshot shows the LibreOffice Calc interface with a spreadsheet titled 'Untitled 1'. The spreadsheet has columns labeled A through F and rows numbered 1 through 8. The data is organized into three columns: 'Name' (column B), 'Age' (column C), and 'Sex' (column D). The data rows are as follows:

| | A | B | C | D | E | F |
|---|---|--------------------------|-----|--------|---|---|
| 1 | | Name | Age | Sex | | |
| 2 | 0 | Braund, Mr. Owen Harris | 22 | male | | |
| 3 | 1 | Allen, Mr. William Henry | 35 | male | | |
| 4 | 2 | Bonnell, Miss. Elizabeth | 58 | female | | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |

3、Series

`DataFrame` 中的每一列都是一个 `Series`

Series



| | |
|--|--|
| | |
| | |
| | |
| | |
| | |

- 如果只是对处理 `Age` 列中的数据感兴趣，那么可以这样操作

```
1 print(df["Age"])
2
3 '''
4 Out[4]:
5 0    22
6 1    35
7 2    58
8 Name: Age, dtype: int64
9 '''
```

选择pandas `DataFrame` 的单列时，结果是 `Series` 。要选择列，请使用方括号 `[]` 之间的列标签。

也可以创建一个“`Series`”：

```
1 ages = pd.Series([22, 35, 58], name="Age")
2 print(ages)
3 '''
4 In [6]: ages
5 Out[6]:
6 0    22
7 1    35
8 2    58
9 Name: Age, dtype: int64
10 '''
```

Pandas的 `Series` 没有列标签，因为它只是 `DataFrame` 的单列。但是有行标签

4、求统计信息

- 如果想知道最大年龄 的人

我们可以通过选择 `Age` 列并应用 `max()` 在 `DataFrame` 上执行此操作：

```
1 print(df["Age"].max())
2 #输出结果为： 58
```

或者用 `Series` :

```
1 print(ages.max())
2 #输出结果为: 58
```

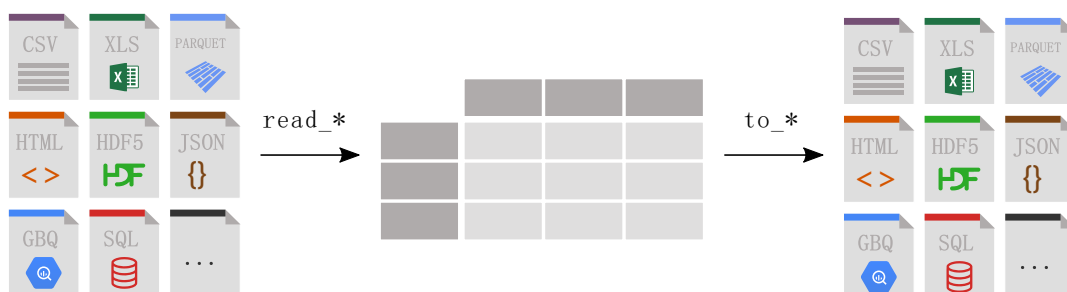
如 `max()` 方法所示, 您可以使用 `DataFrame` 或 `Series` 来处理一些数据, 还有诸如 `mean()` 之类的统计函数, 后面会有介绍。

- 如果对数据表的数值数据的一些基本统计感兴趣, 那么可以这样

```
1 print(df.describe())
2 '''
3          Age
4 count  3.000000
5 mean   38.333333
6 std    18.230012
7 min    22.000000
8 25%    28.500000
9 50%    35.000000
10 75%    46.500000
11 max    58.000000
12 '''
13
```

`describe()` 方法提供了 `DataFrame` 中数值数据的快速概览。由于 `Name` 和 `Sex` 列是文本数据, 因此默认情况下不会被 `describe()` 方法考虑在内。

5、读写表格数据



- 这里我们读入一个泰坦尼克号乘客信息的csv文件,

```
1 titanic = pd.read_csv("data/titanic.csv")
```

pandas 提供了 `read_csv()` 函数来读取存储为 csv 文件的数据到 pandas `DataFrame`。Pandas 支持多种不同的文件格式或数据源 (csv、excel、sql、json、parquet 等), 每种格式都带有前缀 `read_*`。

读入后, 我们查看一下数据:

```
1 print(titanic)
2 '''
3
4      PassengerId  Survived  Pclass
Name ...          Ticket    Fare  Cabin  Embarked
```

```

5      0      1      0      3      Braund, Mr. Owen
   Harris ...      A/5 21171  7.2500  NaN      S
6      1      2      1      1  Cumings, Mrs. John Bradley (Florence Briggs
   Th... ...      PC 17599  71.2833  C85      C
7      2      3      1      3      Heikkinen, Miss.
   Laina ... STON/O2. 3101282  7.9250  NaN      S
8      3      4      1      1      Futrelle, Mrs. Jacques Heath (Lily May
   Peel) ...      113803  53.1000  C123      S
9      4      5      0      3      Allen, Mr. William
   Henry ...      373450  8.0500  NaN      S
10     ..      ...      ...      ...
   ... ...      ...      ...      ...
11     886      887      0      2      Montvila, Rev.
   Juozas ...      211536  13.0000  NaN      S
12     887      888      1      1      Graham, Miss. Margaret
   Edith ...      112053  30.0000  B42      S
13     888      889      0      3      Johnston, Miss. Catherine Helen
   "Carrie" ...      W./C. 6607  23.4500  NaN      S
14     889      890      1      1      Behr, Mr. Karl
   Howell ...      111369  30.0000  C148      C
15     890      891      0      3      Dooley, Mr.
   Patrick ...      370376  7.7500  NaN      Q
16
17 [891 rows x 12 columns]
18 '''
19
20

```

- 如果想查看 Pandas DataFrame 的前 8 行。

```

1  print(titanic.head(8))
2  '''
3  Out[4]:
4      PassengerId  Survived  Pclass
   Name ...      Ticket      Fare  Cabin  Embarked
5      0      1      0      3      Braund, Mr. Owen
   Harris ...      A/5 21171  7.2500  NaN      S
6      1      2      1      1  Cumings, Mrs. John Bradley (Florence Briggs
   Th... ...      PC 17599  71.2833  C85      C
7      2      3      1      3      Heikkinen, Miss.
   Laina ... STON/O2. 3101282  7.9250  NaN      S
8      3      4      1      1      Futrelle, Mrs. Jacques Heath (Lily May
   Peel) ...      113803  53.1000  C123      S
9      4      5      0      3      Allen, Mr. William
   Henry ...      373450  8.0500  NaN      S
10     5      6      0      3      Moran, Mr.
   James ...      330877  8.4583  NaN      Q
11     6      7      0      1      McCarthy, Mr.
   Timothy J ...      17463  51.8625  E46      S
12     7      8      0      3      Palsson, Master. Gosta
   Leonard ...      349909  21.0750  NaN      S
13
14 [8 rows x 12 columns]
15 '''
16
17

```

要查看 `DataFrame` 的前 N 行，可以使用 `head()` 方法，并向括号中传入一个参数，比如8，表示查看前8行。

- 如果想看整个 `DataFrame` 的总结，那么可以这么做

```
1 print(titanic.info())
2
3 '''
4 <class 'pandas.core.frame.DataFrame'>
5 RangeIndex: 891 entries, 0 to 890
6 Data columns (total 12 columns):
7 #   Column      Non-Null Count  Dtype
8 ---  -
9 0   PassengerId  891 non-null    int64
10 1   Survived     891 non-null    int64
11 2   Pclass       891 non-null    int64
12 3   Name         891 non-null    object
13 4   Sex          891 non-null    object
14 5   Age         714 non-null    float64
15 6   SibSp       891 non-null    int64
16 7   Parch       891 non-null    int64
17 8   Ticket      891 non-null    object
18 9   Fare        891 non-null    float64
19 10  Cabin       204 non-null    object
20 11  Embarked    889 non-null    object
21 dtypes: float64(2), int64(5), object(5)
22 memory usage: 83.7+ KB
23 '''
24
```

- 方法 `info()` 提供有关 `DataFrame` 的所以信息，因此让我们可以看到更详细的内容：
 - 它是一个 `DataFrame` 。
 - 有 891 个条目，即 891 行。
 - 每行都有一个行标签（也称为“索引”），其值范围从 0 到 890。
 - 该表有 12 列。大多数列的每一行都有一个值（所有 891 个值都是“非空”）。有些列有缺失值和少于 891 个“非空”值。
 - `Name`、`Sex`、`Cabin` 和 `Embarked` 列由文本数据（字符串，又名 `object`）组成。其他列是数字数据，其中一些是整数，其他列是实数（浮点数）。
 - 不同列中的数据类型（字符、整数等）通过列出 `dtypes` 进行汇总。
 - 还提供了用于保存 `DataFrame` 的大致 RAM 量。

6、从DataFrame中选择特定列



- 如只选取乘客的年龄信息，并用一个新的变量来存放它的内容

```
1
2 #选择Age这一列，通过方括号选择
3 ages = titanic["Age"]
4
5 print(ages.head())
6
7 '''
8 Out[5]:
9 0    22.0
10 1    38.0
11 2    26.0
12 3    35.0
13 4    35.0
14 Name: Age, dtype: float64
15 '''
```

要选择单个列，可以使用方括号 `[]` 和感兴趣的列的列名，名字用双引号包围。

前面我们提到，`DataFrame` 中的每一列都是一个 `Series`。当单列被选中时，返回的对象是一个 Pandas 的 `Series`。

可以通过检查输出的类型 `type()` 来验证这一点：

```
1 print(type(titanic["Age"]))
2
3 '''
4 pandas.core.series.Series
5 '''
```

也可以查看 `shape`

```
1 print(titanic["Age"].shape)
2 #(891,)
```

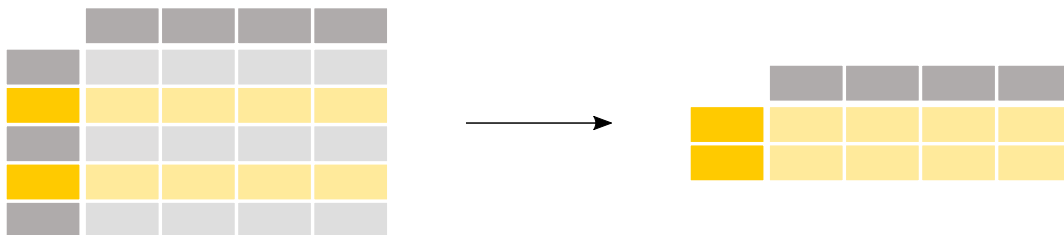
`DataFrame.shape` 包含 行数和列数：(`nrows`, `ncolumns`)。而 `Series` 是一维的，只返回行数。

- 如果想同时查看年龄和性别的数据，那么可以这样

```
1 age_sex = titanic[["Age", "Sex"]]
2
3 print(age_sex.head())
4
5 '''
6 Out[9]:
7      Age  Sex
8 0  22.0  male
9 1  38.0 female
10 2  26.0 female
11 3  35.0 female
12 4  35.0  male
13 '''
14
```

要选择多个列，使用选择括号 `[["Age", "Sex"]]`，这里用了两层括号，表示一个矩阵，类似 numpy 中的二维矩阵。

7、从DataFrame中过滤特定行



- 如果你只对年龄大于35的感兴趣，那么可以这样，在 `[]` 中使用了条件表达式

```
1  above_35 = titanic[titanic["Age"] > 35]
2
3  print(above_35.head())
4  '''
5  Out[13]:
6      PassengerId  Survived  Pclass
7      Name ...   Ticket     Fare  Cabin  Embarked
8      1          2         1       1  Cumings, Mrs. John Bradley (Florence
9      Briggs Th...   ...   PC 17599   71.2833   C85          C
10     6          7         0       1          McCarthy, Mr.
11     Timothy J ...   17463   51.8625   E46          S
12     11         12         1       1          Bonnell, Miss.
13     Elizabeth ...   113783   26.5500   C103          S
14     13         14         0       3          Andersson, Mr.
15     Anders Johan ...   347082   31.2750   NaN          S
16     15         16         1       2          Hewlett, Mrs. (Mary D
17     Kingcome) ...   248706   16.0000   NaN          S
18
19     [5 rows x 12 columns]
20  '''
21
```

选择括号 `titanic["Age"] > 35` 内的满足条件检查的 `Age` 列中大于 35 的值:

```
1  print(titanic["Age"] > 35)
2  '''
3  Out[14]:
4  0      False
5  1       True
6  2      False
7  3      False
8  4      False
9  ...
10 886      False
11 887      False
12 888      False
13 889      False
14 890      False
15 Name: Age, Length: 891, dtype: bool
16  '''
17
```

还可以使用这些条件判断符: (`>`、`==`、`!=`、`<`、`<=` ... `True` 或 `False`), 只会返回满足条件的结果。

我们之前知道，原始的泰坦尼克号 `DataFrame` 由 891 行组成。查看生成的 `DataFrame` `above_35` 的 `shape` 属性来看看满足条件大于35的行数：

```
1 print(above_35.shape)
2 #Out[15]: (217, 12)
```

8、查看非空的数据

```
1 age_no_na = titanic[titanic["Age"].notna()]
2
3 print(age_no_na.head())
4
5
6 ...
7 Out[21]:
8   PassengerId  Survived  Pclass
9   Name ...          Ticket    Fare  Cabin  Embarked
9   0          1           0        3                Braund, Mr. Owen
   Harris ...      A/5 21171   7.2500   NaN        S
10  1          2           1        1 Cumings, Mrs. John Bradley (Florence Briggs
   Th... ...      PC 17599  71.2833   C85        C
11  2          3           1        3                Heikkinen, Miss.
   Laina ... STON/O2. 3101282   7.9250   NaN        S
12  3          4           1        1 Futrelle, Mrs. Jacques Heath (Lily May
   Peel) ...    113803  53.1000  C123        S
13  4          5           0        3                Allen, Mr. William
   Henry ...    373450   8.0500   NaN        S
14
15 [5 rows x 12 columns]
16 ...
17
```

`notna()` 条件函数为每一行不是 `Null` 值的数据返回一个 `True` 值。因此，这可以与选择括号 `[]` 结合使用来过滤数据表。

你可能想知道实际发生了什么变化，因为前 5 行仍然是相同的值。可以通过检查 `shape` 是否已更改：

```
1 print(age_no_na.shape)
2 #(714, 12)
```

9、从DataFrame中选择特定的行和列？



- 若对 35 岁以上乘客的姓名感兴趣。

```

1  adult_names = titanic.loc[titanic["Age"] > 35, "Name"]
2
3  print(adult_names.head())
4  '''
5  Out[24]:
6  1      Cumings, Mrs. John Bradley (Florence Briggs Th...
7  6                                     McCarthy, Mr. Timothy J
8  11                                    Bonnell, Miss. Elizabeth
9  13                                    Andersson, Mr. Anders Johan
10  15                                    Hewlett, Mrs. (Mary D Kingcome)
11  Name: Name, dtype: object
12  '''
13
14

```

在这种情况下，仅使用选择括号 `[]` 是不够的。使用 `loc` / `iloc` 运算符需要在选择括号 `[]` 前面。使用 `loc` / `iloc` 时，逗号前的部分是你要选择的行，逗号后的部分是你要选择的列。

使用列名、行标签或条件表达式时，请在选择括号 `[]` 前使用 `loc` 运算符。对于逗号前后的部分，您可以使用单个标签、标签列表、标签切片、条件表达式或冒号。使用冒号指定您要选择所有行或列。

- 我对第 10 到 25 行和第 3 到 5 列感兴趣。

```

1  print(titanic.iloc[9:25, 2:5])
2
3  '''
4  Out[25]:
5      Pclass      Name      Sex
6  9         2  Nasser, Mrs. Nicholas (Adele Achem)  female
7  10        3   Sandstrom, Miss. Marguerite Rut  female
8  11        1   Bonnell, Miss. Elizabeth  female
9  12        3   Saundercock, Mr. William Henry   male
10  13        3   Andersson, Mr. Anders Johan   male
11  ..      ...
12  20        2      Fynney, Mr. Joseph J   male
13  21        2   Beesley, Mr. Lawrence   male
14  22        3   McGowan, Miss. Anna "Annie"  female
15  23        1   Sloper, Mr. William Thompson   male
16  24        3   Palsson, Miss. Torborg Danira  female
17
18  [16 rows x 3 columns]
19  '''
20
21

```

当根据表中的位置对某些行和/或列特别感兴趣时，请在选择括号 `[]` 前使用 `iloc` 运算符。

小结:

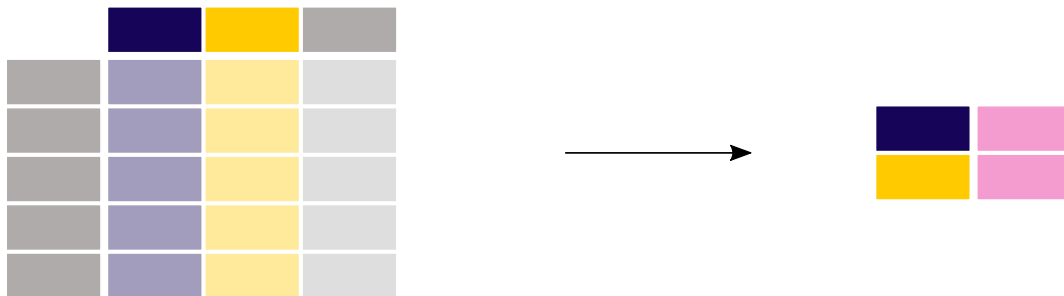
- 选择数据子集时，使用方括号 `[]`。
- 在这些括号内，您可以使用单个列/行标签、列/行标签列表、标签切片、条件表达式或冒号。
- 使用行和列名称时使用 `loc` 选择特定的行或列
- 使用表中的位置时使用 `iloc` 选择特定的行或列，`i` 表示索引 `index`

10、计算更多统计信息



- 泰坦尼克号乘客的平均年龄是多少？

```
1 print(titanic["Age"].mean())
2 #29.69911764705882
```



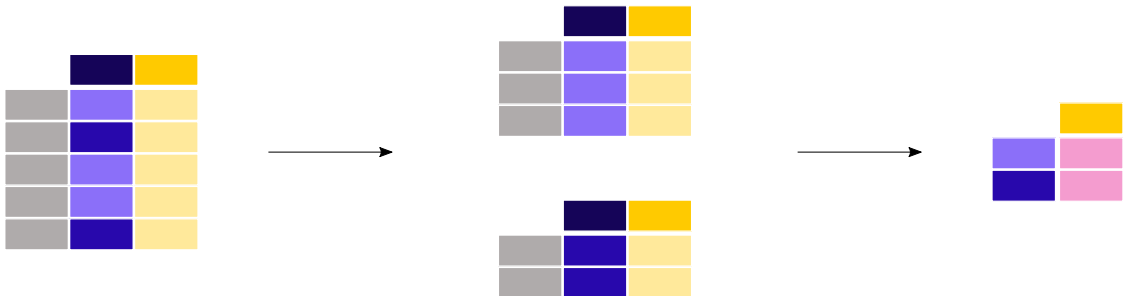
- 泰坦尼克号乘客的年龄和票价的中位数是多少？

```
1 print(titanic[["Age", "Fare"]].median())
2 '''
3 Age      28.0000
4 Fare     14.4542
5 dtype: float64
6 '''
7
```

前面提到过一个 `describe()` 方法

```
1 print(titanic[["Age", "Fare"]].describe())
2
3 '''
4 Out[6]:
5      Age      Fare
6 count  714.000000  891.000000
7 mean    29.699118   32.204208
8 std     14.526497   49.693429
9 min      0.420000    0.000000
10 25%     20.125000    7.910400
11 50%     28.000000   14.454200
12 75%     38.000000   31.000000
13 max     80.000000  512.329200
14 '''
```

11、按类别分组的汇总统计信息



- 男性和女性泰坦尼克号乘客的平均年龄是多少？

```

1  print(titanic[["Sex", "Age"]].groupby("Sex").mean())
2
3  '''
4  Out[8]:
5
6          Age
7  Sex
8  female  27.915709
9  male    30.726645
10 '''

```

由于我们的兴趣是每个性别的平均年龄，因此首先对这两列进行子选择：`titanic[["Sex", "Age"]]`。接下来，将 `groupby()` 方法应用于 `Sex` 列为每个类别创建一个组。计算并返回 **每个性别** 的平均年龄。

在前面的示例中，我们首先明确选择了 2 列。如也可以将 `mean` 方法应用于包含数字列的每一列：

```

1  print(titanic.groupby("Sex").mean())
2  '''
3  Out[9]:
4
5          PassengerId  Survived  Pclass     Age  SibSp  Parch    Fare
6  Sex
7  female    431.028662    0.742038    2.159236  27.915709  0.694268  0.649682  44.479818
8  male      454.147314    0.188908    2.389948  30.726645  0.429809  0.235702  25.523893
9  '''

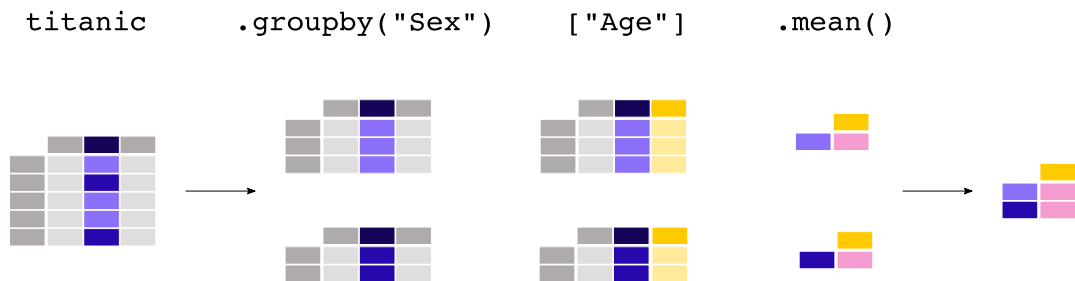
```

获得 `Pclass` 的平均值没有多大意义。如果我们只对每个性别的平均年龄感兴趣，则分组数据也支持列的选择，和上面一样，使用一个 `[]` 来选择相应的列：

```

1 print(titanic.groupby("Sex")["Age"].mean())
2 '''
3 Out[10]:
4 Sex
5 female    27.915709
6 male      30.726645
7 Name: Age, dtype: float64
8 '''

```



本文内容主要来源于官方文档的翻译，更多内容，请参考：https://pandas.pydata.org/docs/getting_started/index.html

(完)