计算机与信息工程学院实验报告

••••••••••••••••••••••••••••••••• 密 ••••••••••••••••••••••••••••••••• 封 ••••••••••••••••••••••••••••••••• 线 •••••••••••••••••••••••••••••••••

姓名：\_杨佳森\_学号：\_2112080106\_专业：数据科学与大数据技术年级：\_ 2021 \_

课程： 数据分析和可视化 主讲教师：\_周黎鸣 辅导教师：\_无\_

实验时间： 2023年4\_月26日\_上\_午\_10\_时至12\_时，实验地点\_计算机大楼201\_\_

实验题目： pandas 分析基础与数据预处理

实验目的： 掌握常见的数据读取方式 ；掌握 DataFrame 常见属性与方法、基础时间数据处理方法、分组聚合的 原理与方法、透视表与交叉表的制作 ；掌握数据合并、数据清洗、基本数据标准化、常用的数据转换方法 。

实验环境（硬件和软件）普通 PC 机 和Windows 操作系统、Jupyter Noteboo

实验内容：

**（一）pandas 统计分析**

1、读/写不同数据源的数据

2、DataFrame 的常用操作

3、转换与处理时间序列数据

4、使用分组聚合进行组内计算

5、创建透视表与交叉表

**（二）使用 pandas 进行预处理**

1、合并数据

2、清洗数据

3、标准化数据

4、转换数据

实验步骤：

（一）pandas 统计分析与预处理

1、摘抄 DataFrame 常用操作的代码

2、摘抄处理时间序列数据的代码

3、摘抄分组聚合进行组内计算的代码

3、摘抄数据预处理的代码，包括合并数据、清洗数据、标准化数据、转换 数据

（二）问题讨论

记录实验过程中遇到的最大的 1～2 个问题，说明这些问题如何得到解决

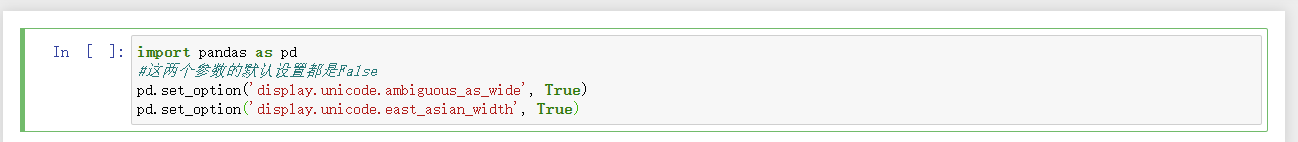
实验数据记录：

**（一）pandas 统计分析**

**1、读/写不同数据源的数据**

数据读取是进行数据预处理、建模与分析的前提。不同的数据源，需要使用不同的函数读取。pandas内置了十几种数据源读取函数和对应的数据写入函数。

用[pandas](https://so.csdn.net/so/search?q=pandas&spm=1001.2101.3001.7020)展示数据输出时列名不能对齐：



常见的数据源有3种，分别是数据库数据、文本文件（包括一般文本文件和CSV文件）、Excel文件。

数据库数据读取：

read\_sql\_table只能够读取数据库的某一个表格，不能实现查询的操作。

pandas.read\_sql\_table(table\_name, con, schema=None, index\_col=None, coerce\_float=True, columns=None)

read\_sql\_query则只能实现查询操作，不能直接读取数据库中的某个表。

pandas.read\_sql\_query(sql, con, index\_col=None, coerce\_float=True)

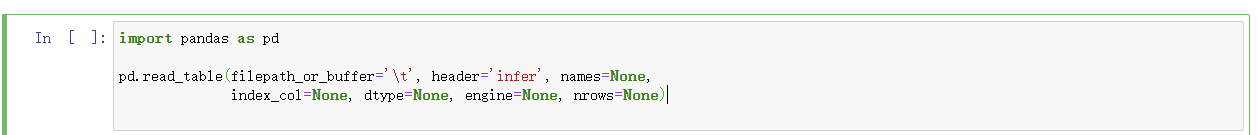
read\_sql是两者的综合，既能够读取数据库中的某一个表，也能够实现查询操作。

pandas.read\_sql(sql, con, index\_col=None, coerce\_float=True, columns=None)

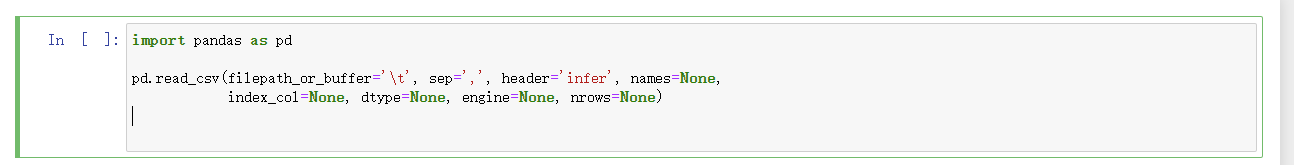
文本文件读取：

csv是一种文本文件，也是一种字符分隔文件。所以可以用读取文本文件的read\_table函数读取，也可以用读取csv字符分割文件的read\_csv函数读取。

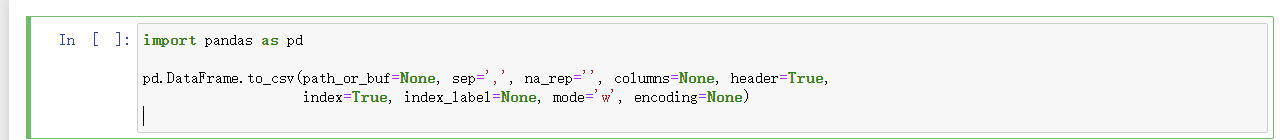
使用read\_table读取文本文件：



使用read\_csv函数来读取csv文件：



文本文件的存储和读取类似，结构化数据可以pandas的to\_csv函数实现：



读写 HTML文件

Pandas.read\_html( io,match=".+",flavor=None,header=None,index\_col=None,skiprows=None,attrs=None,parse\_dates=False,thousands=",",encoding=None,decimal=".",converters=None,na\_values=None,keep\_default\_na=True,displayed\_only=True,)

io：接收网址，文件，字符串。网址不接受https，尝试去掉s后爬取

match：正则表达式，返回与正则表达式匹配的表格。默认".+"

flavor：解析器。默认为(“lxml”,“bs4”)

header：指定列标题所在的行

index\_col：指定行标题对应的列

skiprows：跳过第n行

attrs：传递一个字典，标示表格的属性值

parse\_dates：解析日期

thousands：千位分隔符

encoding：编码方式

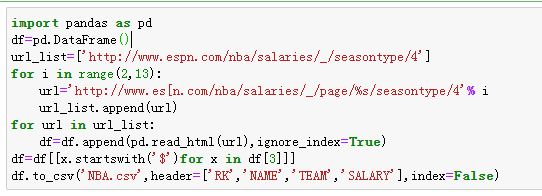
decimal：小数点指示，默认使用"."

converters：转换某些列的函数的字典

na\_values：标示那些为NA的值

keep\_default\_na：保持默认的NA值，与na\_values一起使用

displayed\_only：是否应解析具有"display:none"的元素。默认为True



Excel文件读取：

pandas 提供了 read\_excel 函数来读取“xls”“xlsx”两种 Excel 文件。

pandas.read\_excel(io, sheetname=0, header=0, index\_col=None, names=None, dtype=None)

将文件存储为 Excel 文件，可以使用 to\_excel 方法。其语法格式如下。

DataFrame.to\_excel(excel\_writer=None, sheet\_name=‘None’, na\_rep=”, header=True, index=True, index\_label=None, mode=’w’, encoding=None)

**2、DataFrame 的常用操作**

**Pandas中的Series对象是一个带索引数据构成的一维数组。**



从上面的结果中，不难发现Series对象将一组对象和一组索引绑定在一起，我们可以通过values属性和index属性来获取数据。values属性返回的结果与Numpy数组类似；index属性返回的结果是一个类型为pd.Index的类数组对象。

**Numpy数组通过隐式定义的整数索引来获取数值；**

**Series对象用一种显式定义的索引与数值关联。**

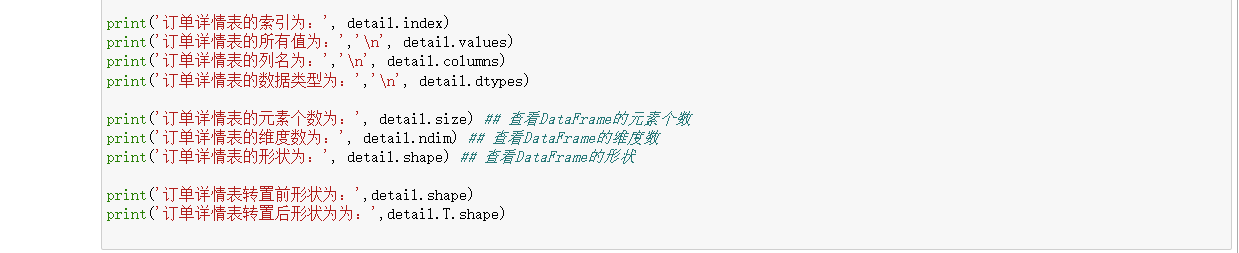
我们可以把Pandas中的Series对象看成一种特殊的Python字典，字典是一种将任意键映射到一组任意值的数据结构，而Series对象其实是一种将类型键映射到一组类型值的数据结构。类型至关重要：就像Numpy数组背后特定类型的经过编译的代码使得它在某些操作上比普通的Python列表更加高效一样，Pandas中的Series对象的类型信息使得它在某些操作上比Python的字典更高效。

DataFrame是非常常用的pandas对象，类似于Execl表格。完成数据读取后，数据就以DataFrame数据结构存储在内存中。

DataFrame基础属性有values、index、columns和dtypes，分别可获取元素、索引、列名和类型。

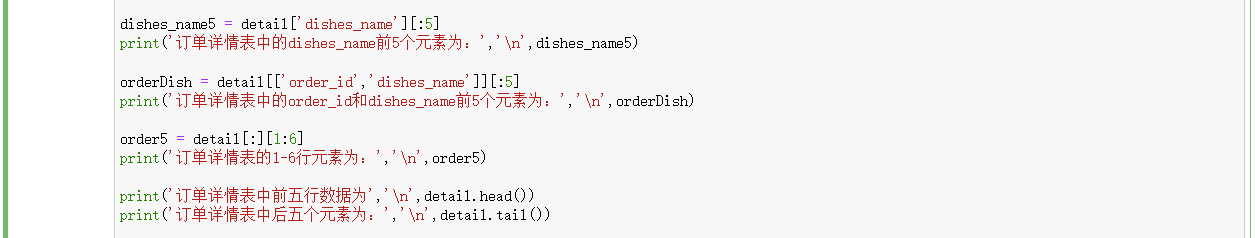
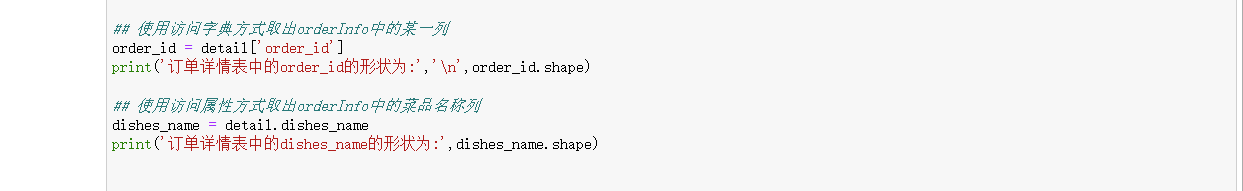
除了上述4个基本属性外，size、ndim和shape属性能够获取DataFrame的元素个数、维度数和数据形状（行列数目）。

另外，T属性能够实现DataFrame的转置（行列转换）。



DataFrame作为一种二维数据表结构，能够像数据库一样实现增删改查操作，如添加一行，删除一行，添加一列，删除一列，修改某一个值，或将在某个区间的值替换等。

基本查看方式：

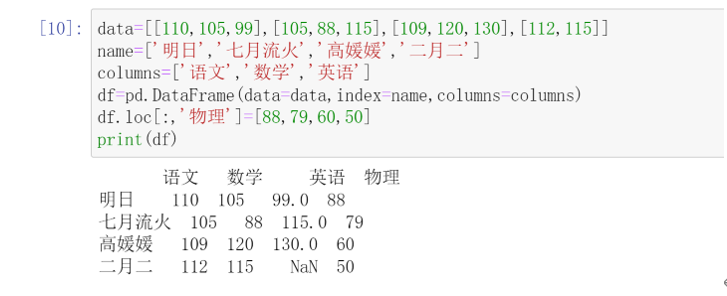


DataFrame的loc和iloc访问方式：

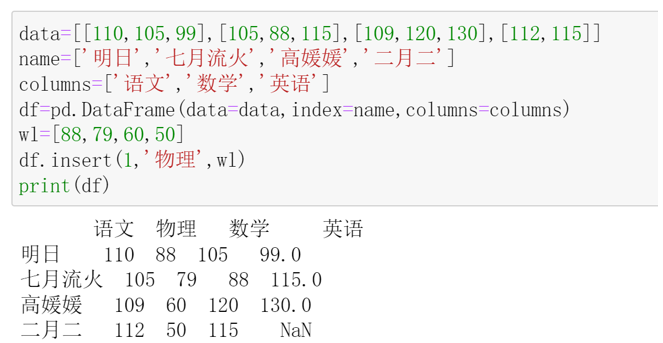
loc方法是针对DataFrame索引名称的切片方法，如果传入的不是索引名称，那么切片操作将无法执行。利用loc方法，能够实现所有单层索引切片操作。iloc和loc的区别在于，iloc接受的必须是行索引和列索引的位置。



使用loc属性在DataFrame对象的最后增加一列

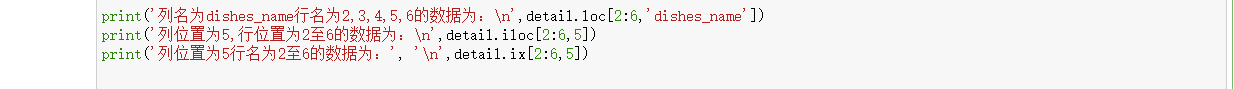


在指定位置插入一列



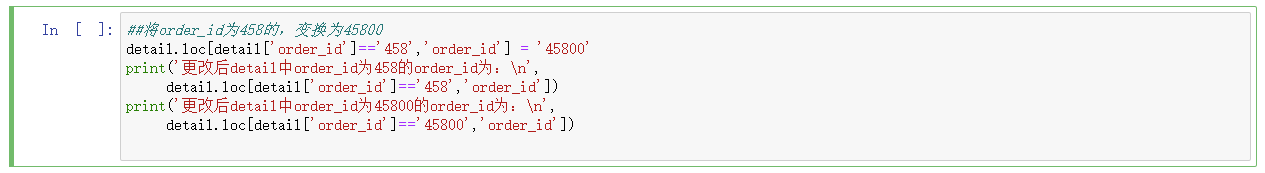
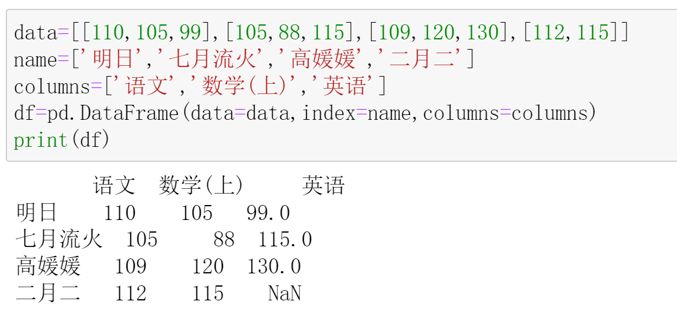
切片方法之ix

ix方法更像是loc和iloc两种切片方法的融合。ix方法在使用时既可以接受索引名称，也可以接受索引位置。当索引名称和位置存在部分重叠时，ix默认优先识别名称。

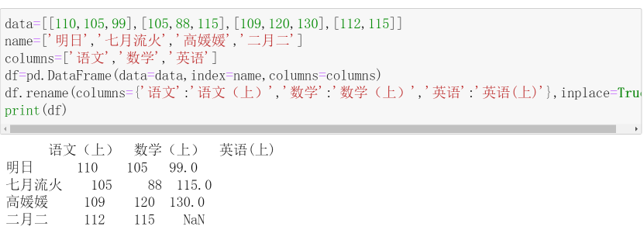


更改DataFrame中的数据：

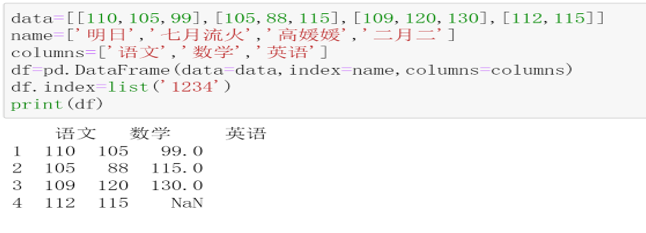
更改DataFrame中的数据的原理是将这部分数据提取出来，重新赋值为新的数据。数据更改是直接对DataFrame原数据更改，操作无法撤销。如果做出更改，则需要对更改条件进行确认或对数据进行备份。



使用DataFrame对象的rename()方法修改标题

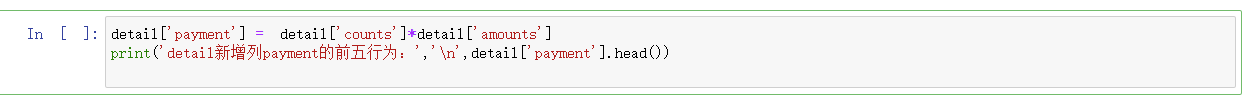
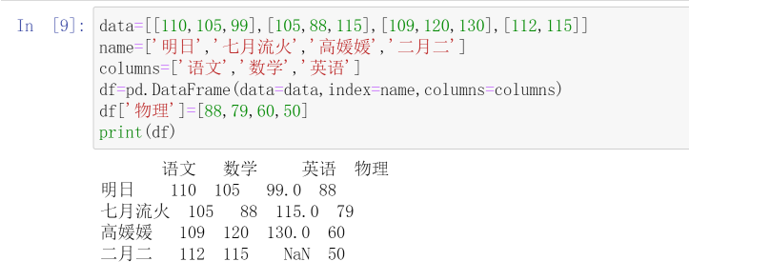


修改行标题



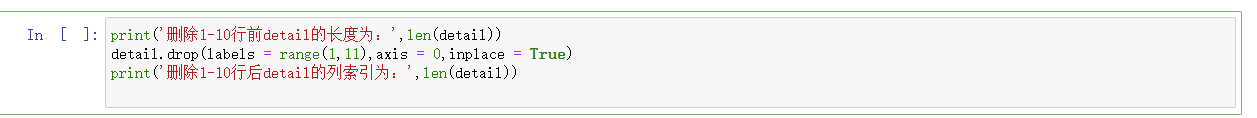
为DataFrame增添数据：

只需新建一个列索引，并对该索引下的数据进行赋值操作。如果新增的一列值是相同的，则直接为其赋值一个常量。



删除某列或某行数据：

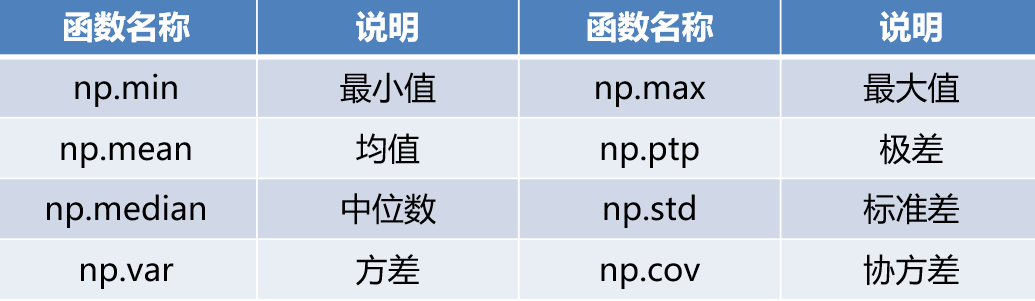
删除某列或某行数据需要用到pandas提供的方法drop。要删除某行数据，只需要将drop方法参数中的“labels”参数换成对应的行索引，将“axis”参数设置为0。



数值型特征的描述性统计——NumPy中的描述性统计函数

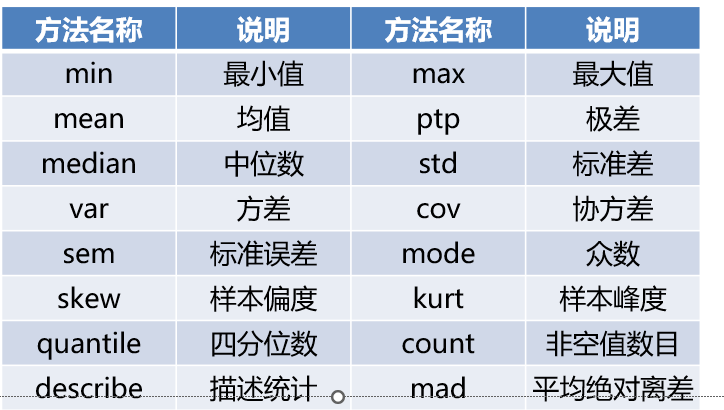
数值型数据的描述性统计主要包括了计算数值型数据的完整情况、最小值、均值、中位数、最大值、四分位数、极差、标准差、方差、协方差和变异系数等。在NumPy库中一些常用的统计学函数如下表所示。

pandas库基于NumPy，自然也可以用这些函数对数据框进行描述性统计。



数值型特征的描述性统计—— pandas描述性统计方法

pandas还提供了一个方法叫作describe，能够一次性得出数据框所有数值型特征的非空值数目、均值、四分位数、标准差。

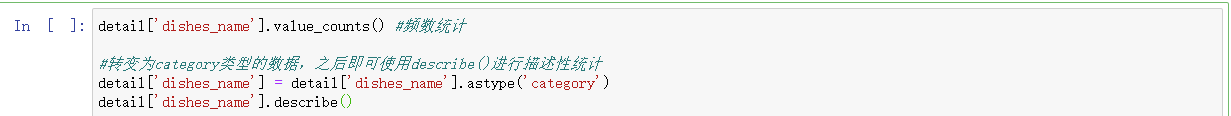


类别型特征的描述性统计

描述类别型特征的分布状况，可以使用频数统计表。pandas库中实现频数统计的方法为value\_counts.

pandas提供了category类，可以使用astype方法将目标特征的数据类型转换为category类别。

describe方法除了支持传统数值型以外，还能够支持对**category类型**的数据进行描述性统计，四个统计量分别为列非空元素的数目，类别的数目，数目最多的类别，数目最多类别的数目。



数据计算

pandas模块的sum()函数可以对每一列数据进行求和

DataFrame.sum(axis=None,skipna=None,level=None,numeric\_only=None,min\_count=0,\*\*kwargs)

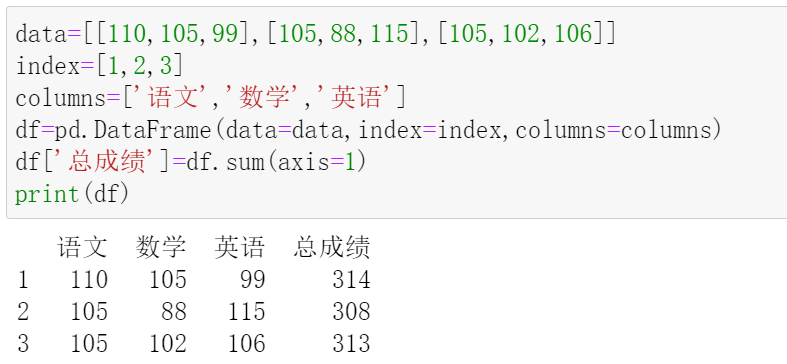
Axis：axis=1表示行，=0表示列

Skipna:布尔型，表示计算结果是否排除NaN/Null

Level:表索引层级

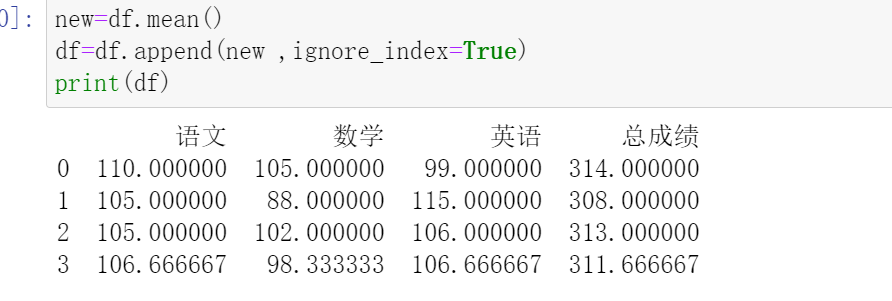
Numeric\_only:仅数字，布尔型

Min\_count:表示执行操作所需数目。



用mean()求各列的平均值

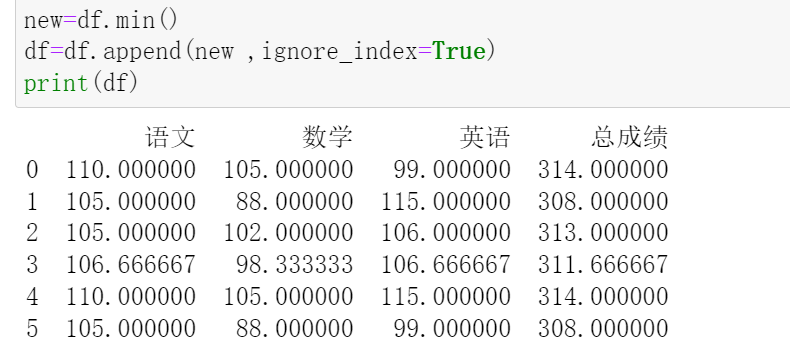
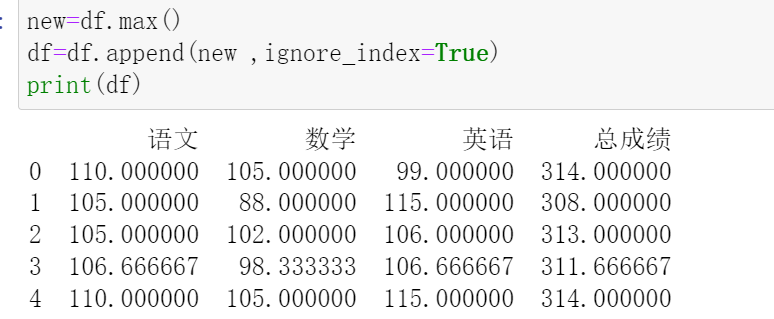
DataFrame.mean(axis=None,skipna=None,level=None,numeric\_only=None,min\_count=0,\*\*kwargs)



max()、min()求极大值、极小值

DataFrame.max(axis=None,skipna=None,level=None,numeric\_only=None,min\_count=0,\*\*kwargs)

DataFrame.min(axis=None,skipna=None,level=None,numeric\_only=None,min\_count=0,\*\*kwargs)



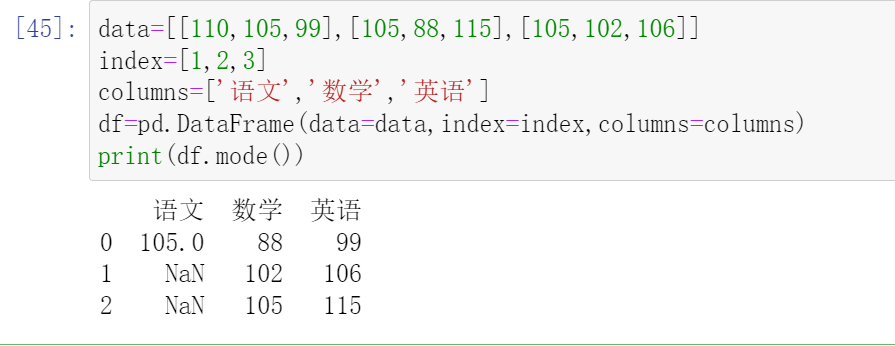
求中位数

DataFrame.median(axis=None,skipna=None,level=None,numeric\_only=None,min\_count=0,\*\*kwargs)



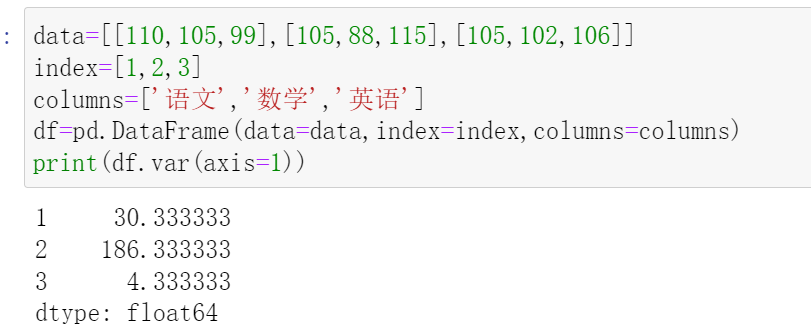
求众数

DataFrame.mode(axis=None,skipna=None,level=None,numeric\_only=None,min\_count=0,\*\*kwargs)



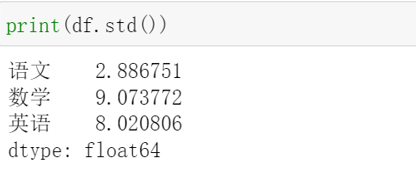
求方差

DataFrame.var(axis=None,skipna=None,level=None,numeric\_only=None,min\_count=0,\*\*kwargs)



标准差

DataFrame.std(axis=None,skipna=None,level=None,numeric\_only=None,min\_count=0,\*\*kwargs)

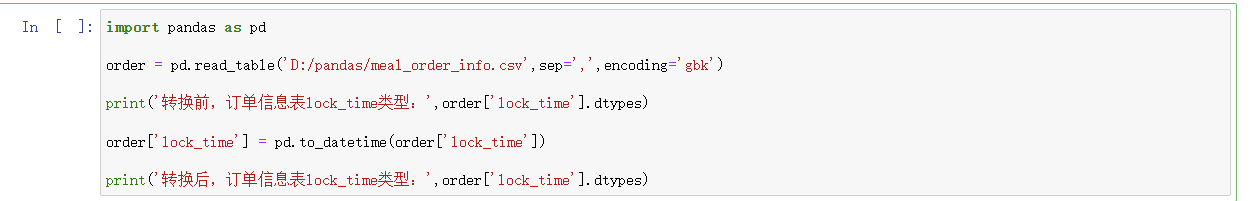


**3、转换与处理时间序列数据**

在多数情况下，对时间类型数据进行分析的前提就是将原本为字符串的时间转换为标准时间类型。pandas继承了NumPy库和datetime库的时间相关模块，提供了6种时间相关的类。



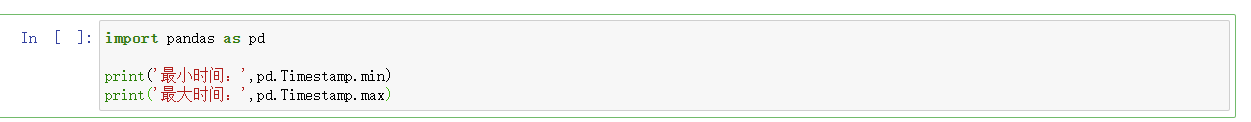
标准时间转换 to\_datetime()函数



**Timestamp类型**

其中Timestamp作为时间类中最基础的，也是最为常用的。在多数情况下，时间相关的字符串都会转换成为Timestamp。pandas提供了to\_datetime函数，能够实现这一目标。

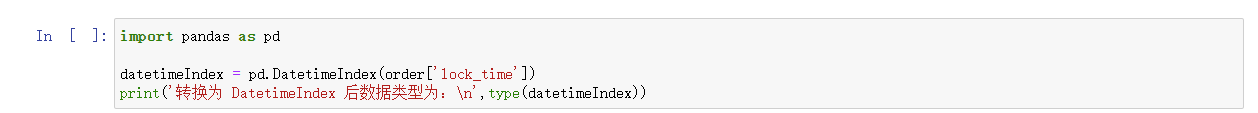
值得注意的是，Timestamp类型时间是有限制的。1677年9月21日-2262年4月11日



**DatetimeIndex与PeriodIndex函数**

除了将数据字原始DataFrame中直接转换为Timestamp格式外，还可以将数据单独提取出来将其转换为DatetimeIndex或者PeriodIndex。

转换为PeriodIndex的时候需要注意，需要通过freq参数指定时间间隔，常用的时间间隔有Y为年，M为月，D为日，H为小时，T为分钟，S为秒。两个函数可以用来转换数据还可以用来创建时间序列数据，其参数非常类似。



DatetimeIndex 和 PeriodIndex 相关参数说明：



to\_datetime（）应用于日期字符串列，并转换为datetime64 [ns]类型。

to\_datetime(arg,errors=“ignore”,dayfirst=False,yearfirst=False,utc=None,box=True,format=None,exact=True,unit=None,inter\_datetime=False,origin=“unix”,cache=False)

arg: 字符串、日期时间、字符串数组。

errors：忽略错误（igonre），引发异常（raise），设置无效解析为 NaT（coerce）。

dayfirst: 默认为 False。如果为 True，解析第一个为天，例如：01/05/2022，解析为 2022-05-01。

yearfirst: 默认为 False。如果为 True，解析第一个为年，例如：22-05-01，解析为 2022-05-01。

utc: 默认为 None，返回协调世界时间（UTC）。

box: 默认为 True，返回 DatetimeIndex；如果为 False，返回值的 ndarray。

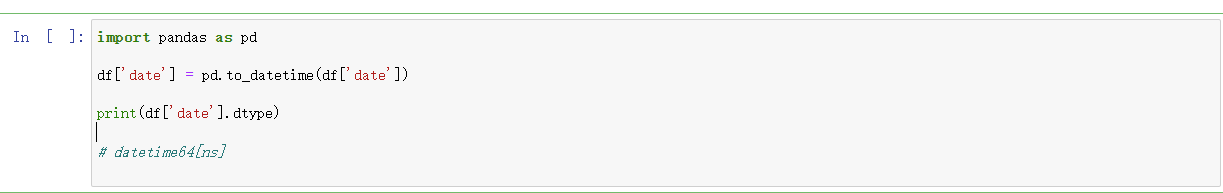
format: 格式化显示时间的格式。字符串，默认值为 None。

exact: 默认为 False。如果为 True，则要求格式完全匹配；如果为 False，则允许格式与目标字符串中的任意位置匹配。

infer\_datetime\_format: 默认为 False。没有格式时，尝试根据第一个日期推断出格式。

origin: 默认为 ‘unix’。定义参考日期。数值将解析为单位数。

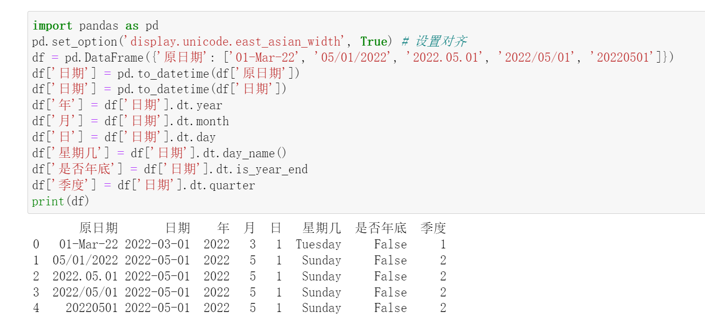
cache: 默认值为 False。设置为 True 启用缓存加速。



dt对象的使用

Series.dt()

返回值:返回与原始系列相同的索引系列。

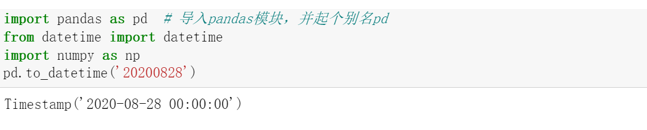


获取日期区间的数据



时间序列

时间戳使用Timestamp（Series派生的子类）对象表示，该对象与datetime具有高度的兼容性，可以直接通过to\_datetime()函数将datetime转换为TimeStamp对象。

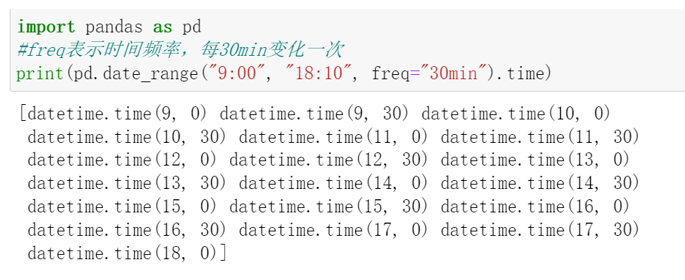


通过 date\_range() 方法可以创建某段连续的时间或者固定间隔的时间时间段。该函数提供了三个参数，分别是：

start：开始时间

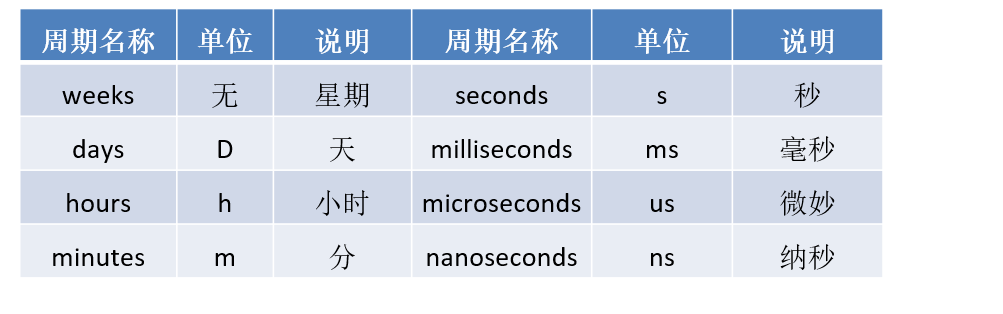
end：结束时间

freq：时间频率，默认为 "D"（天）



**Timedelta类**

Timedelta是时间相关的类中的一个异类，不仅能够使用正数，还能够使用负数表示单位时间，例如1秒，2分钟，3小时等。使用Timedelta类，配合常规的时间相关类能够轻松实现时间的算术运算。目前Timedelta函数中时间周期中没有年和月。所有周期名称，对应单位及其说明如下表所示。



使用Timedelta ，可以很轻松地实现在某个时间上加减一段时间 。

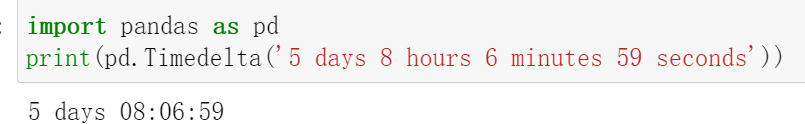
除了使用Timedelta实现时间的平移外，还能够直接对两个时间序列进行相减，从而得出一个Timedelta。



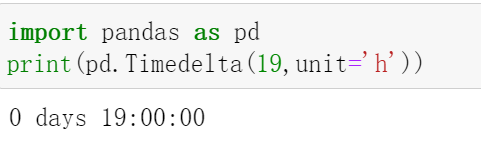
使用 Datetimeindex() 函数设置时间序



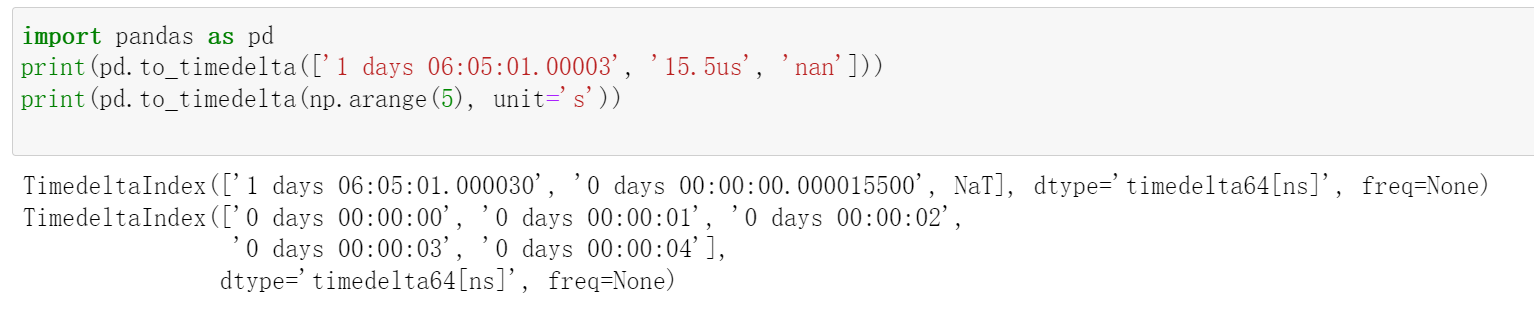
通过传递字符串可以创建 Timedelta 对象



通过传递整数值和unit参数也可以创建一个 Timedelta 对象

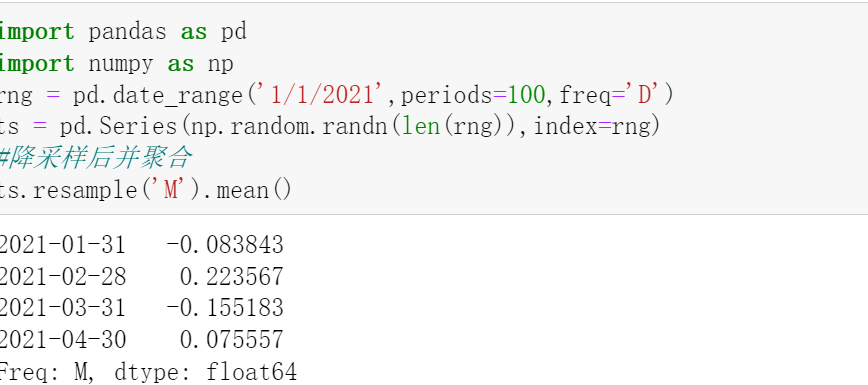


使用pd.to\_timedelta()方法，将具有 timedelta 格式的值 (标量、数组、列表或 Series）转换为 Timedelta 类型。如果输入是 Series，则返回 Series；如果输入是标量，则返回值也为标量，其他情况输出 TimedeltaIndex

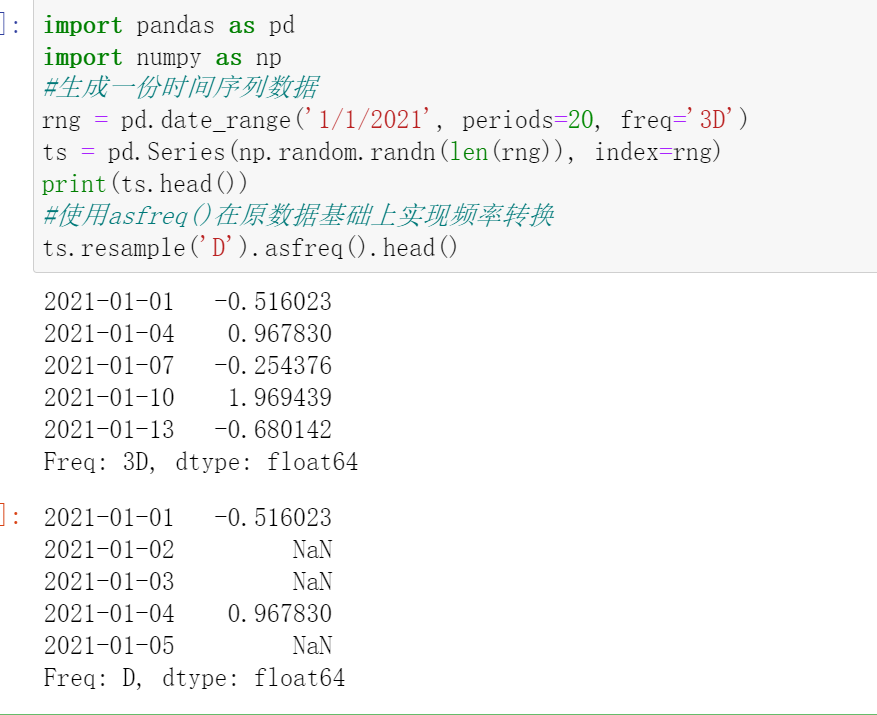


通过 resample() 函数完成数据的降采样

**resample** ( rule, how=None, axis=0, fill\_method=None, closed=None, label=None, convention='start',kind=None, loffset=None, limit=None, base=0 )



升采样是将低频率（时间间隔）转换为高频率



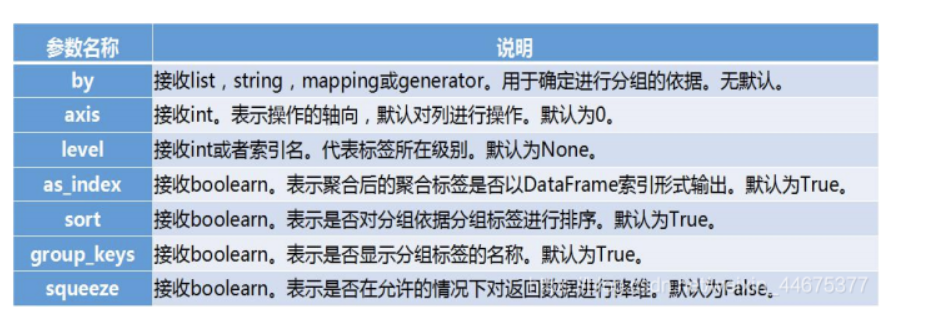
**4、使用分组聚合进行组内计算**

使用 groupby 方法拆分数据

groupby 方法的参数及其说明：

该方法提供的是分组聚合步骤中的拆分功能，能根据索引或字段对数据进行分组。其常用参数与使用格式如下。

[DataFrame](https://so.csdn.net/so/search?q=DataFrame&spm=1001.2101.3001.7020).groupby(by=None, axis=0, level=None, as\_index=True, sort=True, group\_keys=True, squeeze=False, \*\*kwargs)



groupby 方法的参数及其说明——by 参数的特别说明：

如果传入的是一个函数则对索引进行计算并分组。

如果传入的是一个字典或者 Series 则字典或者 Series 的值用来做分组依据。

如果传入一个 NumPy 数组则数据的元素作为分组依据。

如果传入的是字符串或者字符串列表则使用这些字符串所代表的字段作为分组依据。

GroupBy 对象常用的描述性统计方法：

用 groupby 方法分组后的结果并不能直接查看，而是被存在内存中，输出的是内存地址。实际上分组后的数据对象 GroupBy 类似 Series 与DataFrame，是 pandas 提供的一种对象。GroupBy 对象常用的描述性统计方法如下：



使用 agg 方法聚合数据

agg 和 aggregate 函数参数及其说明：

agg，aggregate 方法都支持对每个分组应用某函数，包括 Python 内置函数或自定义函数。

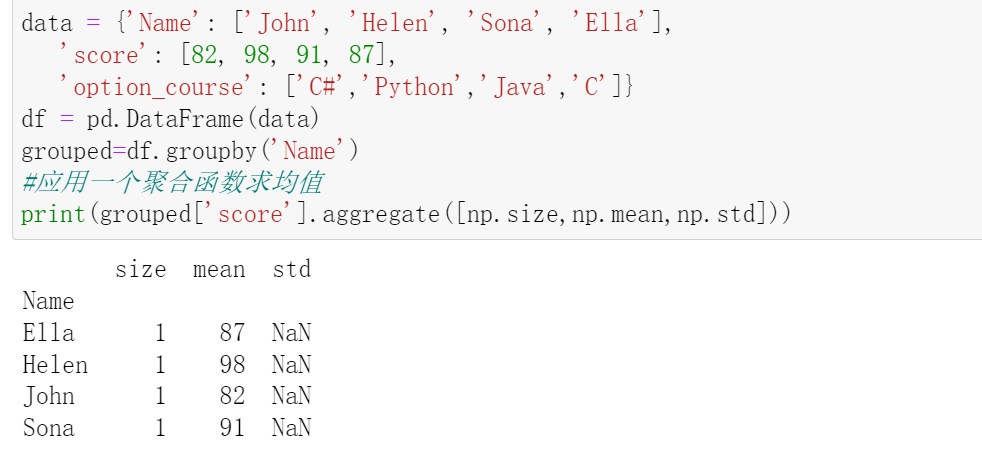
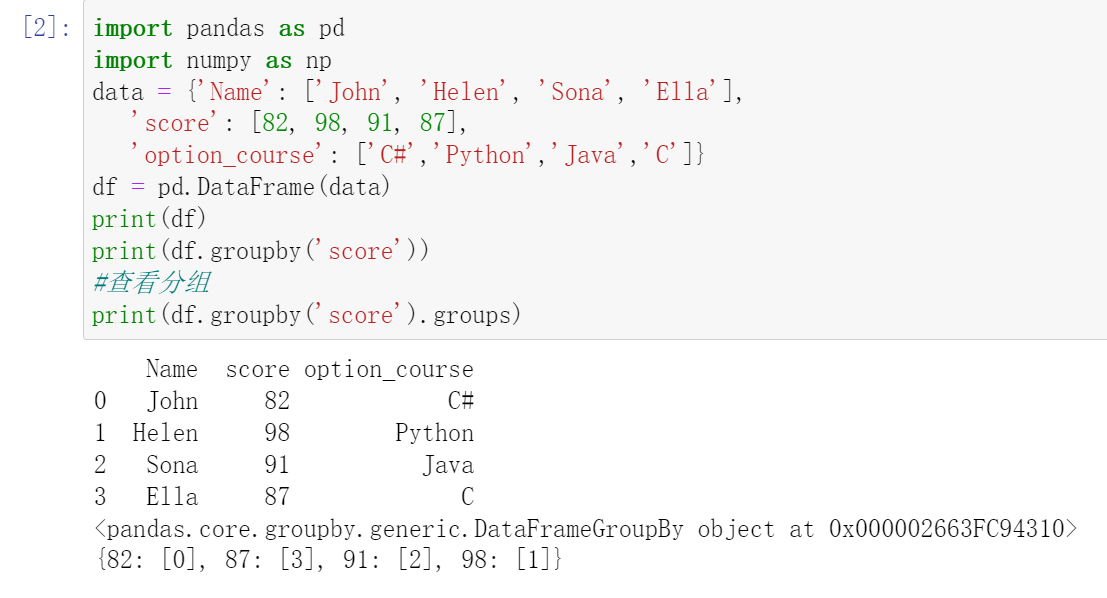
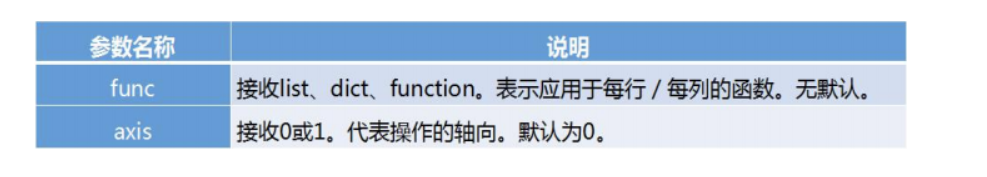
同时这两个方法能够也能够直接对 DataFrame 进行函数应用操作。

在正常使用过程中，agg 函数和 aggregate 函数对 DataFrame 对象操作时功能几乎完全相

同，因此只需要掌握其中一个函数即可。它们的参数说明如下表。

DataFrame.agg(func, axis=0, \*args, \*\*kwargs)

DataFrame.aggregate(func, axis=0, \*args, \*\*kwargs)



对分组的某列或多列使用聚合函数agg()

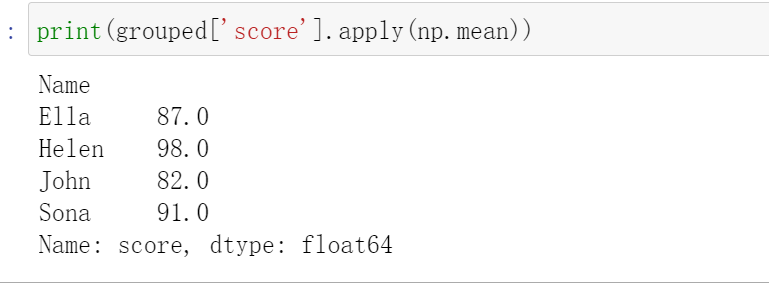


apply方法是一个既能接收返回标量值的函数，又能接收返回数组的函数的聚合方法

apply 方法类似 agg 方法能够将函数应用于每一列。不同之处在于 apply 方法相比 agg方法传入的函数只能够作用于整个 DataFrame 或者 Series，而无法像 agg 一样能够对不同字段，应用不同函数获取不同结果。

使用 apply 方法对 GroupBy 对象进行聚合操作其方法和 agg 方法也相同，只是使用 agg方法能够实现对不同的字段进行应用不同的函数，而 apply 则不行。

DataFrame.apply(func, axis=0, broadcast=False, raw=False, reduce=None, args=(), \*\*kwds)



**5、创建透视表与交叉表**

pandas对数据框也可以像excel一样进行[数据透视表](https://so.csdn.net/so/search?q=%E6%95%B0%E6%8D%AE%E9%80%8F%E8%A7%86%E8%A1%A8&spm=1001.2101.3001.7020)整合之类的操作。主要是针对分类数据进行操作，还可以计算数值型数据，去满足复杂的分类数据整理的逻辑。

pandas.pivot\_table(data,values=None,index=None, columns=None,aggfunc=‘mean’,fill\_value=None,margins=False, dropna=True, margins\_name=‘All’)

pivot参数：

data 接收DataFrame。表示创建表的数据。无默认。

values 接收字符串。用于指定想要聚合的数据字段名，默认使用全部数据。默认为None。

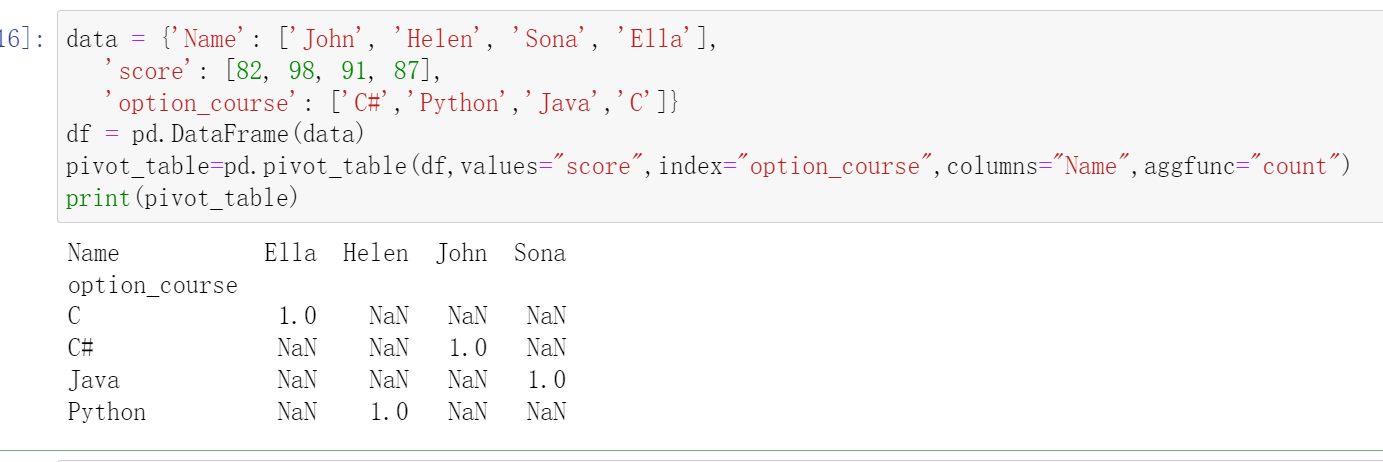
index：新 df 的索引列，用于分组，如果为None，则使用现有索引。

columns：新 df 的列，如果透视后有重复值会报错。

values：用于填充 df 的列。 如果未指定，将使用所有剩余的列，并且结果将具有按层次结构索引的列。

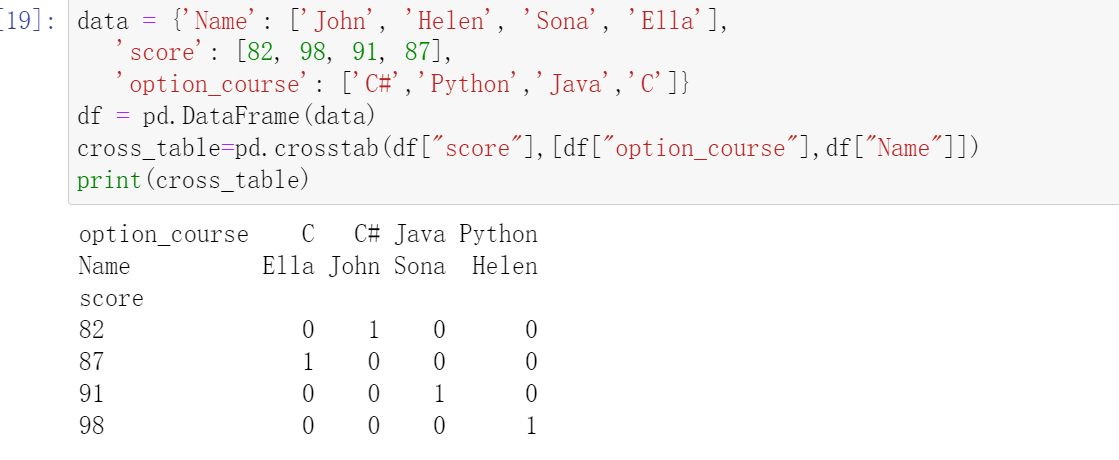
Aggfunc：接收functions。表示聚合函数。默认为mean。接收boolearn。表示汇总（Total）功能的开关，设为True后结果集中会出现名为“ALL”的行margins 和列。默认为False。

Dropna：接收boolearn。表示是否删掉全为NaN的列。默认为True。



pandas.crosstab(index, columns, values=None, rownames=None, colnames=None,aggfunc=None,margins=False,dropna=True,normalize=False)

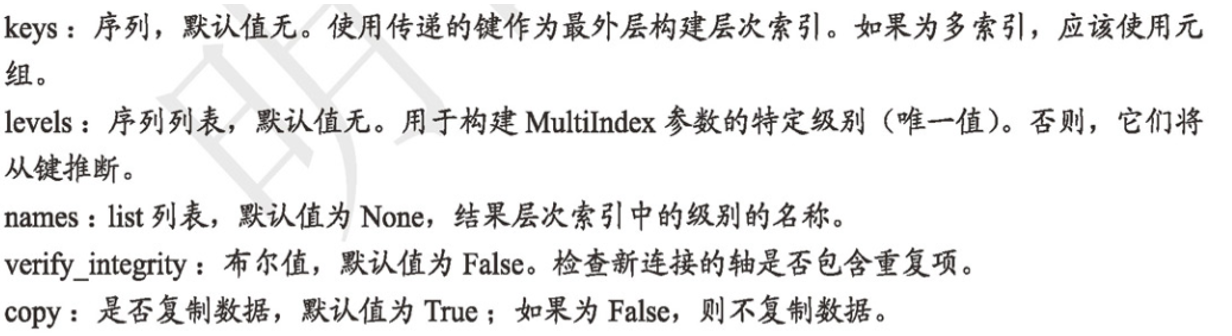
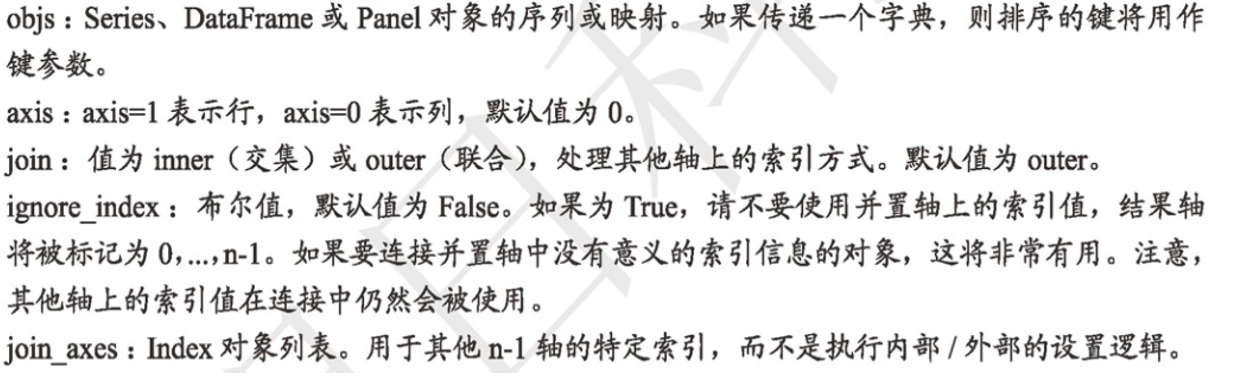
index 接收string或list。表示行索引键。无默认。  
columns 接收string或list。表示列索引键。无默认。  
values 接收array。表示聚合数据。默认为None。  
aggfunc 接收function。表示聚合函数。默认为None。  
rownames 表示行分组键名。无默认。  
colnames 表示列分组键名。无默认。  
dropna 接收boolearn。表示是否删掉全为NaN的。默认为False。  
接收boolearn。默认为True。汇总(Total)功能的开关，设为True  
margins 后结果集中会出现名为“ALL”的行和列。  
normalize 接收boolearn。表示是否对值进行标准化。默认为False。



**（二）使用 pandas 进行预处理**

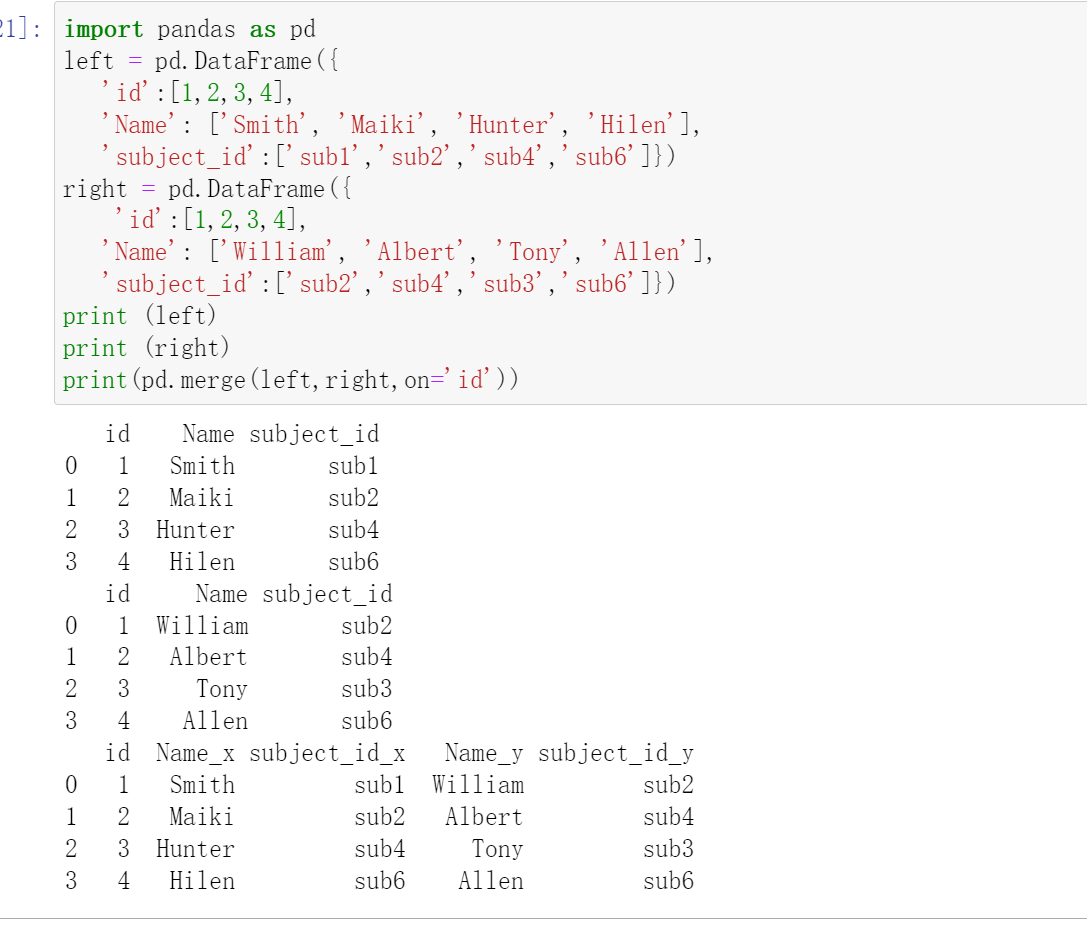
**1、合并数据**

横向堆叠，即将两个表在X轴向拼接在一起，可以使用concat函数完成，concat函数的基本语法pandas.concat(objs, axis=0, join='outer', join\_axes=None, ignore\_index=False, keys=None, levels=None, names=None, verify\_integrity=False, copy=True)



当axis=1的时候，concat做行对齐，然后将不同列名称的两张或多张表合并。当两个表索引不完全一样时，可以使用join参数选择是内连接还是外连接。在内连接的情况下，仅仅返回索引重叠部分。在外连接的情况下，则显示索引的并集部分数据，不足的地方则使用空值填补。

当两张表完全一样时，不论join参数取值是inner或者outer，结果都是将两个表完全按照X轴拼接起来。



使用concat函数时，在默认情况下，即[axis](https://so.csdn.net/so/search?q=axis&spm=1001.2101.3001.7020)=0时，concat做列对齐，将不同行索引的两张或多张表纵向合并。

append方法也可以用于纵向合并两张表。但是append方法实现纵向表堆叠有一个**前提条件**，那就是两张表的列名需要完全一致。append方法的基本语法如下

*pandas.DataFrame.****append****(self, other, ignore\_index=False, verify\_integrity=False)*。

常用参数如下所示：

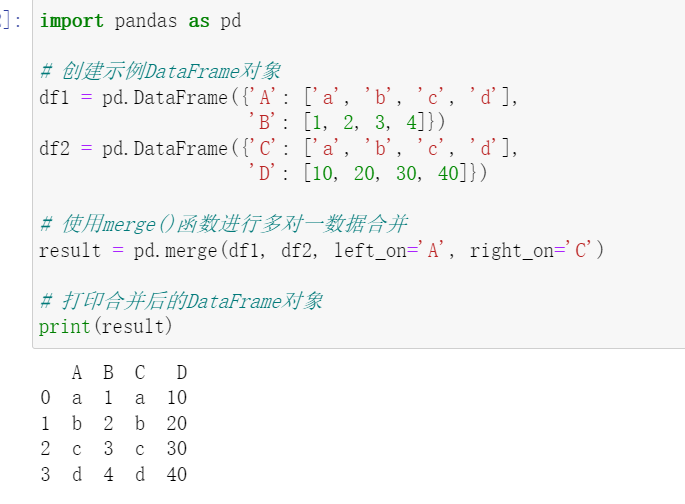
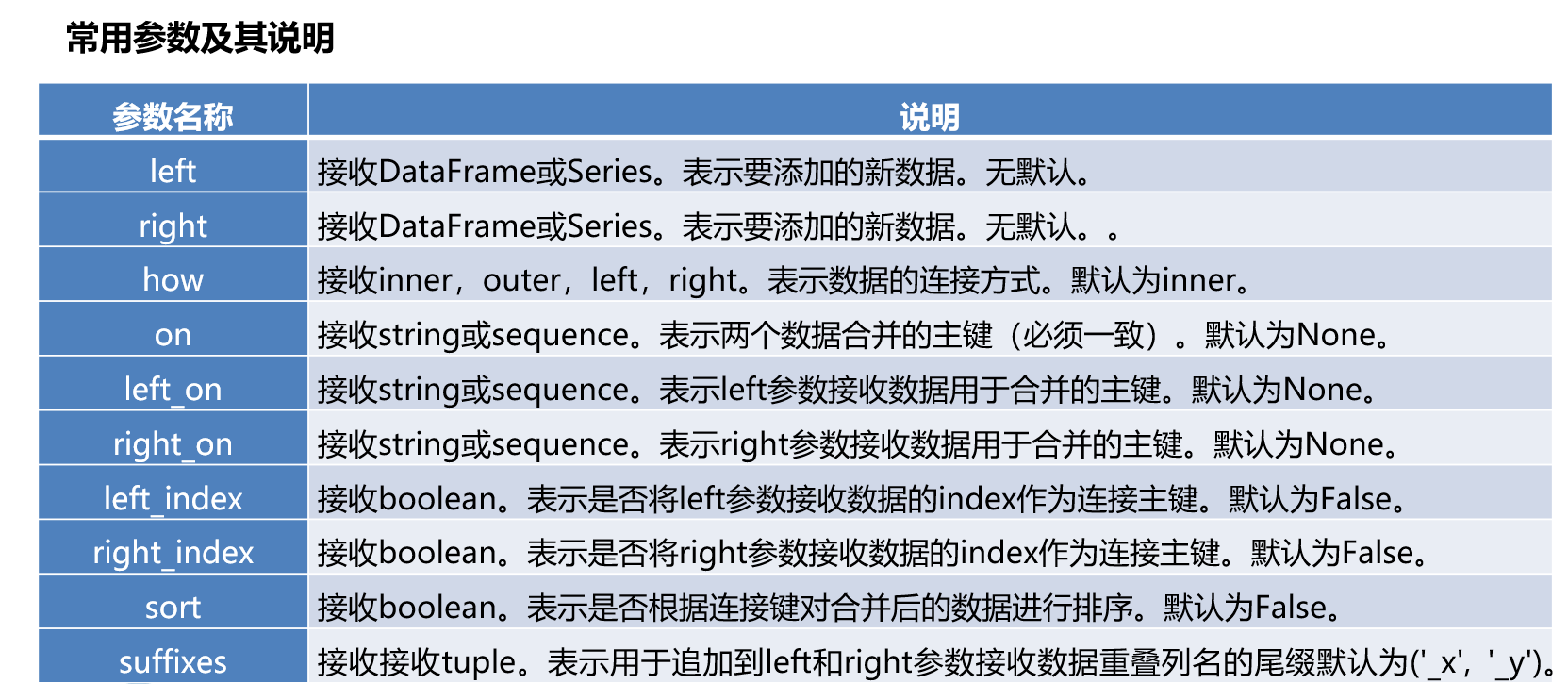


**主键合并**——merge函数

和数据库的join一样，merge函数也有左连接（left）、右连接（right）、内连接（inner）和外连接（outer），但比起数据库SQL语言中的join和merge函数还有其自身独到之处，例如可以在合并过程中对数据集中的数据进行排序等。

*pandas.****merge****(left, right, how='inner', on=None, left\_on=None, right\_on=None, left\_index=False, right\_index=False, sort=False, suffixes=('\_x', '\_y'), copy=True, indicator=False)*

可根据merge函数中的参数说明，并按照需求修改相关参数，就可以多种方法实现主键合并。



**主键合并**——join方法

join方法也可以实现部分主键合并的功能，但是join方法使用时，两个主键的名字必须相同。

*pandas.DataFrame.join(self, other, on=None, how='left', lsuffix='', rsuffix='', sort=False)*

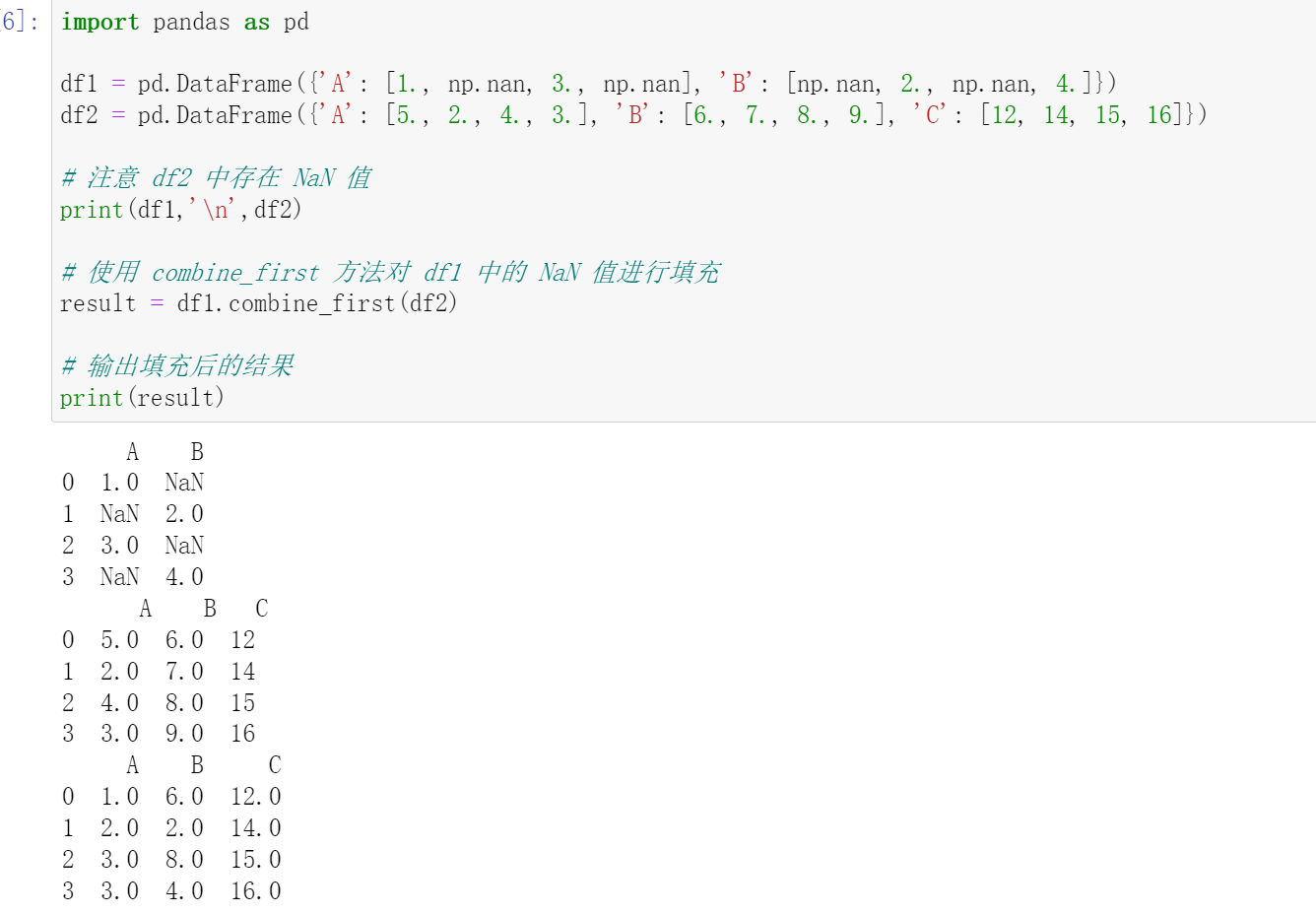
常用参数说明如下：



**使用combine\_first()方法合并两个DataFrame对象时，必须确保它们的行索引和列索引有重叠的部分**

**Pandas.DataFrame.combine\_first(other)**

**Other:接收DataFrame。表示参与重叠合并的另一个DataFrame.**



**2、清洗数据**

记录重复

记录重复，即一个或者多个特征某几个记录的值完全相同

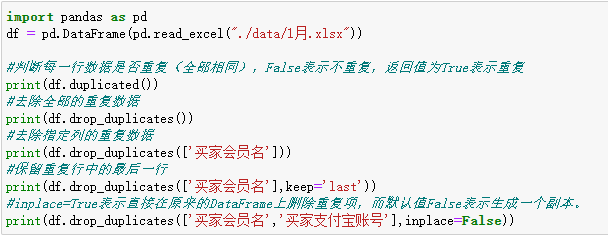
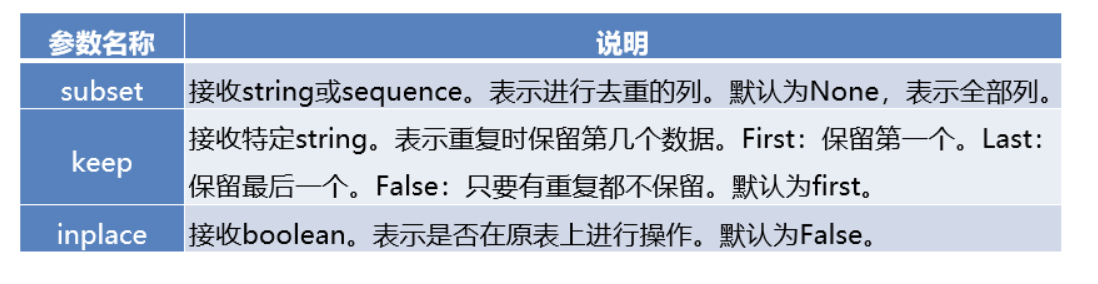
方法一是利用列表（list）去重，自定义去重函数：

方法二是利用集合（set）的元素是唯一的特性去重，如dish\_set = set(dishes)

比较上述两种方法可以发现，方法一代码冗长。方法二代码简单了许多，但会导致数据的排列发生改变。

pandas提供了一个名为drop\_duplicates的去重方法。该方法只对DataFrame或者Series类型有效。这种方法不会改变数据原始排列，并且兼具代码简洁和运行稳定的特点。该方法不仅支持单一特征的数据去重，还能够依据DataFrame的其中一个或者几个特征进行去重操作。

*pandas.DataFrame(Series).****drop\_duplicates****(self, subset=None, keep='first', inplace=False)*



**特征重复**

结合相关的数学和统计学知识，去除连续型特征重复可以利用特征间的相似度将两个相似度为1的特征去除一个。在pandas中相似度的计算方法为corr，使用该方法计算相似度时，默认为“pearson”法 ，可以通过“method”参数调节，目前还支持“spearman”法和“kendall”法。

但是通过相似度矩阵去重存在一个弊端，该方法只能对数值型重复特征去重，类别型特征之间无法通过计算相似系数来衡量相似度。  
除了使用相似度矩阵进行特征去重之外，可以通过DataFrame.equals的方法进行特征去重。

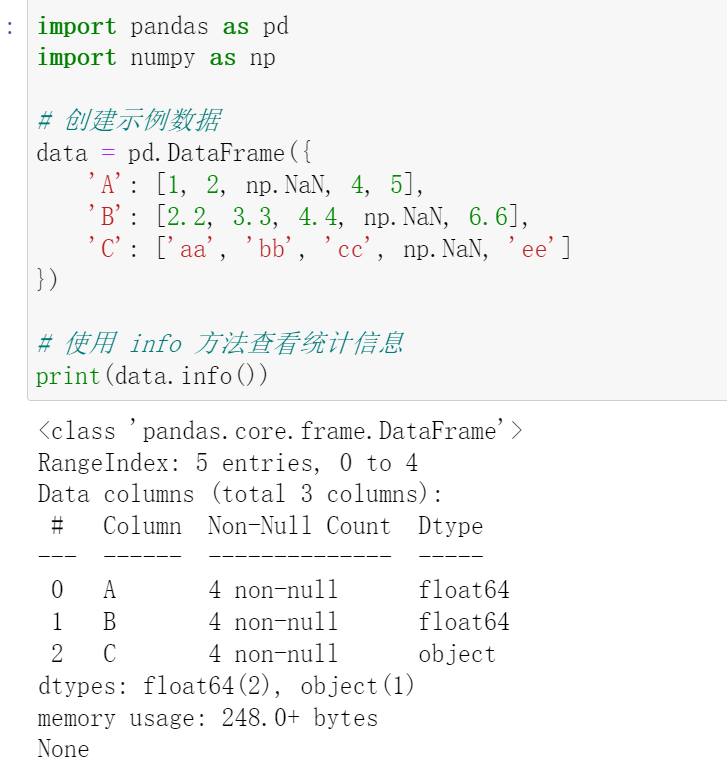
**利用isnull或notnull找到缺失值**

数据中的某个或某些特征的值是不完整的，这些值称为缺失值。

pandas提供了识别缺失值的方法isnull以及识别非缺失值的方法notnull，这两种方法在使用时返回的都是布尔值True和False。

结合sum函数和isnull、notnull函数，可以检测数据中缺失值的分布以及数据中一共含有多少缺失值。

isnull和notnull之间结果正好相反，因此使用其中任意一个都可以判断出数据中缺失值的位置。



①删除法

删除法分为删除观测记录和删除特征两种，它属于利用减少样本量来换取信息完整度的一种方法，是一种最简单的缺失值处理方法。

pandas中提供了简便的删除缺失值的方法dropna，该方法既可以删除观测记录，亦可以删除特征。

pandas.DataFrame.dropna(self, axis=0, how='any', thresh=None, subset=None, inplace=False)



②替换法

替换法是指用一个特定的值替换缺失值。

特征可分为数值型和类别型，两者出现缺失值时的处理方法也是不同的。

缺失值所在特征为数值型时，通常利用其均值、中位数和众数等描述其集中趋势的统计量来代替缺失值。

缺失值所在特征为类别型时，则选择使用众数来替换缺失值。

pandas库中提供了缺失值替换的方法名为fillna，其基本语法如下。

pandas.DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None)



③插值法

删除法简单易行，但是会引起数据结构变动，样本减少；替换法使用难度较低，但是会影响数据的标准差，导致信息量变动。在面对数据缺失问题时，除了这两种方法之外，还有一种常用的方法—插值法。

常用的插值法有**线性插值**、**多项式插值**和**样条插值**等：

**线性插值**是一种较为简单的插值方法，它针对已知的值求出线性方程，通过求解线性方程得到缺失值。

**多项式插值**是利用已知的值拟合一个多项式，使得现有的数据满足这个多项式，再利用这个多项式求解缺失值，常见的多项式插值法有拉格朗日插值和牛顿插值等。

**样条插值**是以可变样条来作出一条经过一系列点的光滑曲线的插值方法，插值样条由一些多项式组成，每一个多项式都是由相邻两个数据点决定，这样可以保证两个相邻多项式及其导数在连接处连续。

异常值

异常值是指数据中个别值的数值明显偏离其余的数值，有时也称为离群点，检测异常值就是检验数据中是否有录入错误以及是否含有不合理的数据。

异常值的存在对数据分析十分危险，如果计算分析过程的数据有异常值，那么会对结果会产生不良影响，从而导致分析结果产生偏差乃至错误。

常用的异常值检测主要为3σ原则和箱线图分析两种方法。

1、3σ原则又称为拉依达法则。该法则就是先假设一组检测数据只含有随机误差，对原始数据进行计算处理得到标准差，然后按一定的概率确定一个区间，认为误差超过这个区间的就属于异常值。

这种判别处理方法仅适用于对正态或近似正态分布的样本数据进行处理，如下表所示，其中σ代表标准差，μ代表均值，x=μ为图形的对称轴。

数据的数值分布几乎全部集中在区间(μ-3σ,μ+3σ)内，超出这个范围的数据仅占不到0.3%。故根据小概率原理，可以认为超出3σ的部分数据为异常数据。

2、**箱线图分析**

箱型图提供了识别异常值的一个标准，即**异常值**通常被定义为小于QL-1.5IQR或大于QU+1.5IQR的值。

**QL称为下四分位数**，表示全部观察值中有四分之一的数据取值比它小。

**QU称为上四分位数**，表示全部观察值中有四分之一的数据取值比它大。

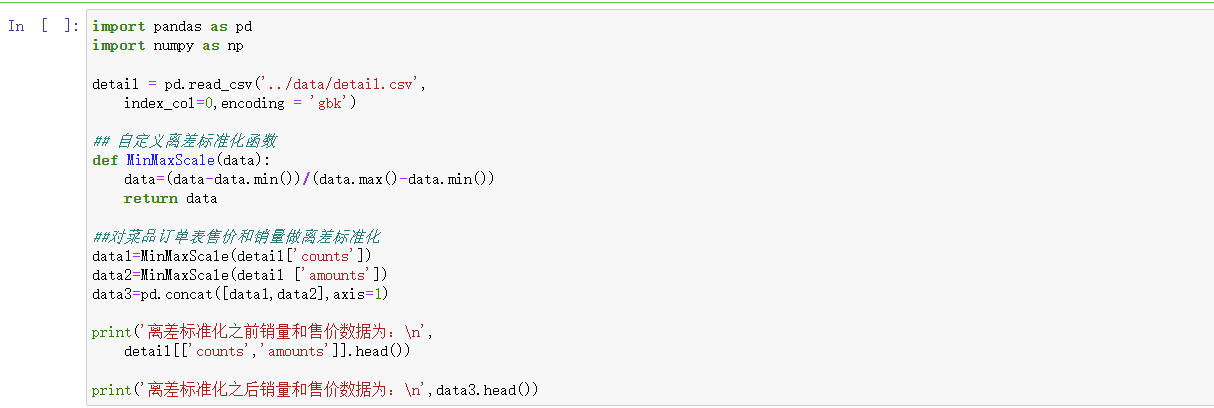
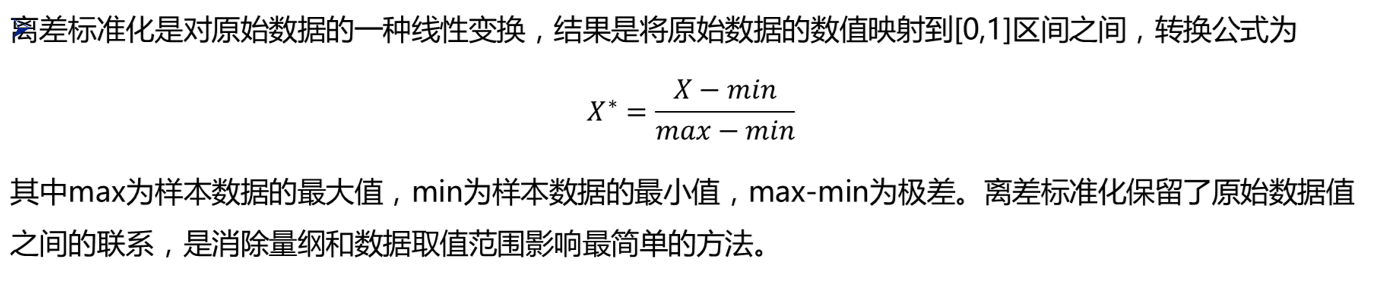
**IQR称为四分位数间距**，是上四分位数QU与下四分位数QL之差，其间包含了全部观察值的一半。

箱线图依据实际数据绘制，真实、直观地表现出了数据分布的本来面貌，且没有对数据做任何限制性要求，其判断异常值的标准以四分位数和四分位数间距为基础。

四分位数给出了数据分布的中心、散布和形状的某种指示，具有一定的鲁棒性，即25%的数据可以变得任意远而不会很大地扰动四分位数，所以异常值通常不能对这个标准施加影响。鉴于此，箱线图识别异常值的结果比较客观，因此在识别异常值方面具有一定的优越性。

**3、标准化数据**

**离差标准化公式**



离差标准化的特点

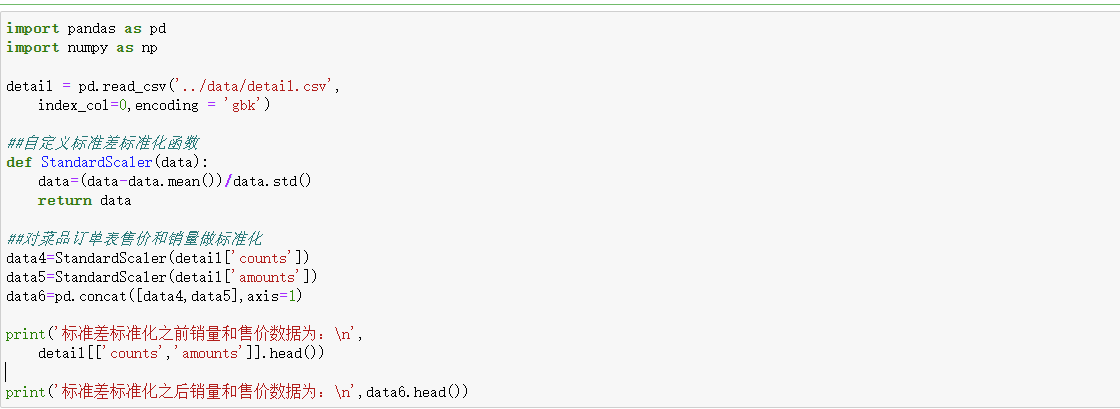
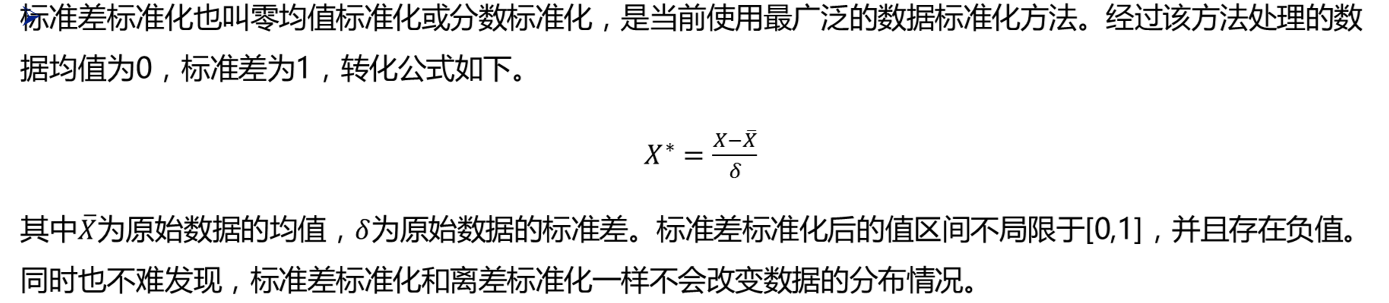
数据的整体分布情况并不会随离差标准化而发生改变，原先取值较大的数据，在做完离差标准化后的值依旧较大。

当数据和最小值相等的时候，通过离差标准化可以发现数据变为0。

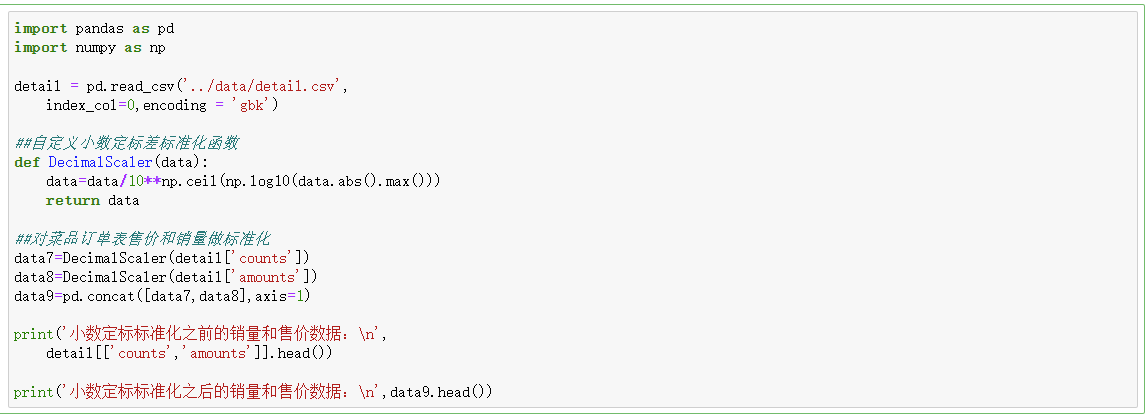
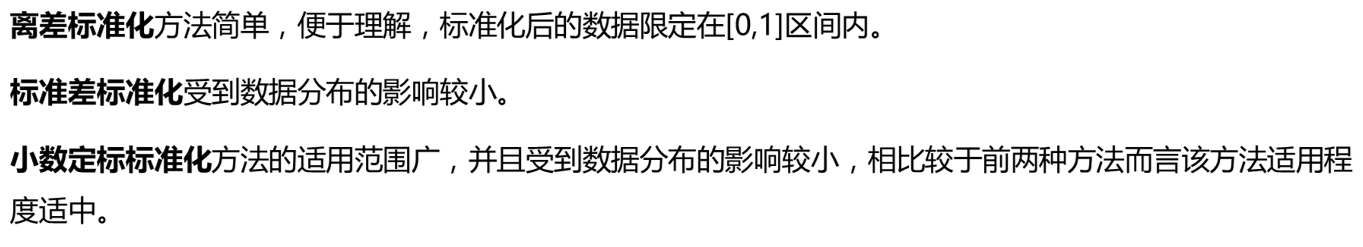
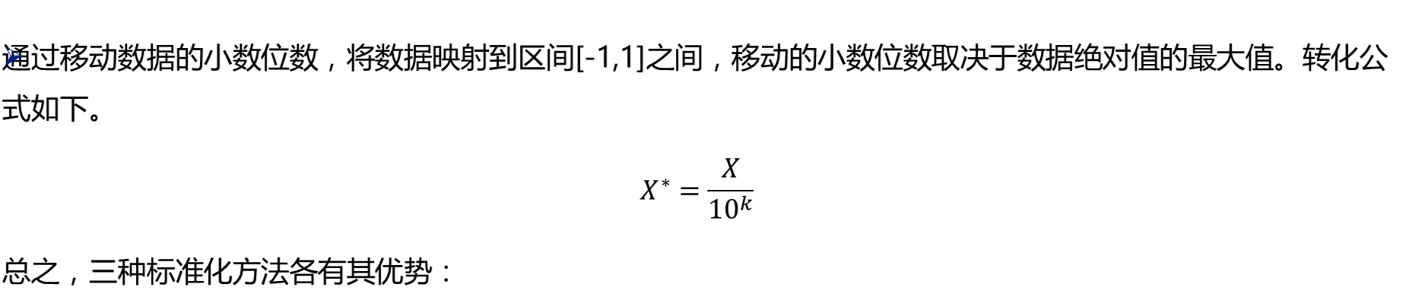
若**数据极差过大**就会出现数据在离差标准化后数据之间的差值非常小的情况。

同时，还可以看出离差标准化的**缺点**：若数据集中某个数值很大，则离差标准化的值就会接近于0，并且相互之间差别不大。若将来遇到超过目前属性[min,max]取值范围的时候，会引起系统出错，这时便需要重新确定min和max。

**标准差标准化的公式及特点**



**小数定标标准化公式及对比**



**4、转换数据**

数据分析的预处理工作除数据清理、数据合并以及数据标准化之外，还包括数据变换的过程

数据即使经过了清洗、合并和标准化，依旧不能直接拿来做分析建模。

为了能够将数据分析工作继续往前推进，需要对数据做一些合理的转换，使之符合分析要求。

**哑变量处理**

get\_dummies()函数可以将分类变量转换为哑变量/指示变量，结果生成一个行数等于数据的元素个数，列数等于变量数，列名为各变量名的[DataFrame](https://so.csdn.net/so/search?q=DataFrame&spm=1001.2101.3001.7020),对应分类变量的位置取值为1，其余位置取值为0。

pandas.get\_dummies(data, prefix=None, prefix\_sep=’\_’, dummy\_na=False, columns=None, sparse=False, drop\_first=False, dtype=None)

data 接收array、DataFrame或者Series。表示需要哑变量处理的数据。无默认。

prefix 接收string、string的列表或者string的dict。表示哑变量化后列名的前缀。默认为None。

prefix\_sep 接收string。表示前缀的连接符。默认为‘\_’。

dummy\_na 接收boolean。表示是否为Nan值添加一列。默认为False。

columns 接收类似list的数据。表示DataFrame中需要编码的列名。默认为None，表示对所有object和category类型进行编码。

sparse 接收boolean。表示虚拟列是否是稀疏的。默认为False。

drop\_first 接收boolean。表示是否通过从k个分类级别中删除第一级来获得k-1个分类级别。默认为False。



哑变量的特点

对于一个类别型特征，若其取值有m个，则经过哑变量处理后就变成了m个二元特征，并且这些特征互斥，每次只有一个激活，这使得数据变得稀疏。

对类别型特征进行哑变量处理主要解决了部分算法模型无法处理类别型数据的问题，这在一定程度上起到了扩充特征的作用。由于数据变成了稀疏矩阵的形式，因此也加速了算法模型的运算速度。

**离散化**

某些模型算法，特别是某些分类算法如ID3决策树算法和Apriori算法等，要求数据是离散的，此时就需要将连续型特征（数值型）变换成离散型特征（类别型）。

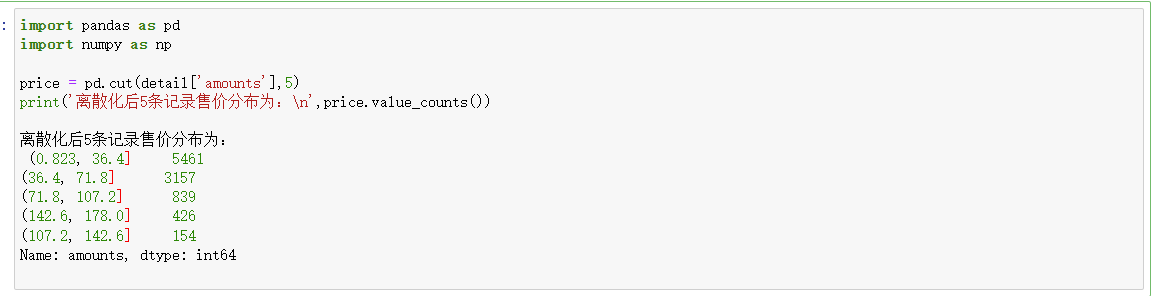
连续特征的离散化就是在数据的取值范围内设定若干个离散的划分点，将取值范围划分为一些离散化的区间，最后用不同的符号或整数值代表落在每个子区间中的数据值。

因此离散化涉及两个子任务，即确定分类数以及如何将连续型数据映射到这些类别型数据上。其原理如图。

等宽法：

将数据的值域分为具有相等宽度的区间，区间的个数由数据本身的特点或由用户决定。使用cut函数可以进行连续型数据的等宽离散化。

pandas.cut(x,bins,right=True,labels=None,retbins=False,percision=3,include\_lowest=False)

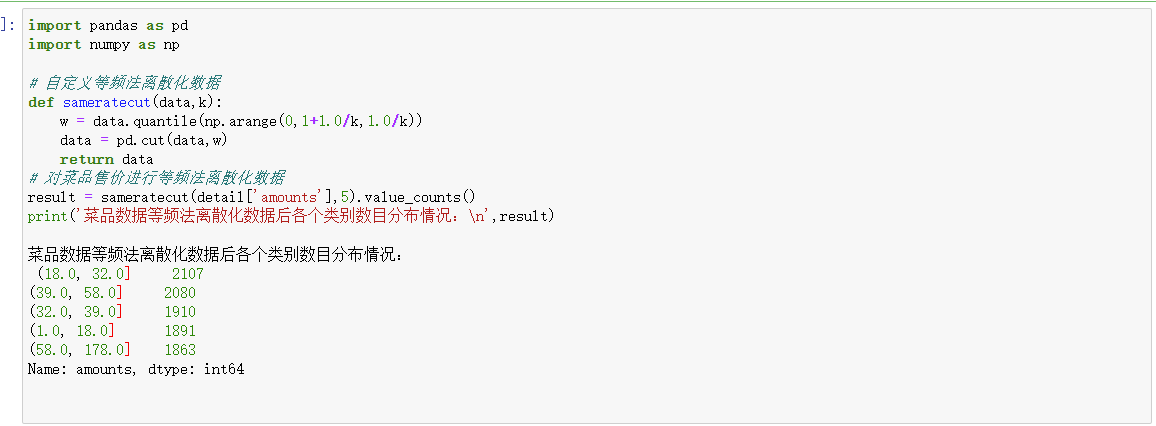


使用等宽法虽然可以将其归到一定的区间中，但是当数据分布不均匀的时候，各个类中的数目也会变得非常不均匀，有些区间包含许多的数据，而有的区间数据很少，后期会对建立的模型有所影响。

**等频法**

cut函数虽然不能够直接实现等频离散化，但是可以通过定义将相同数量的记录放进每个区间。

等频法离散化的方法相比较于等宽法离散化而言，避免了类分布不均匀的问题，但同时却也有可能将数值非常接近的两个值分到不同的区间以满足每个区间中固定的数据个数。



**在定义等频法是关键的方法是使用到了quantile（），才使得每个区间中的数目相对平均。**

**聚类分析法**

一维聚类的方法包括两个步骤：

首先将连续型数据用聚类算法进行聚类

然后处理聚类得到的簇，为合并到一个簇的连续型数据做同一种标记

(聚类分析的离散化方法需要用户指定簇的个数，用来决定产生的区间数。)



数据转换

1、一行数据转换为多列数据

Pandas的DataFrame对象中的str.split()内置方法可以实现分割字符串。

Series.str.split(pat=None,n=-1,expand=False)

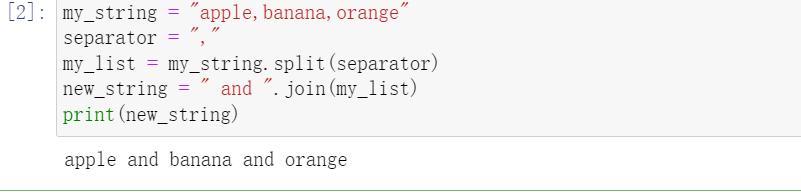
Pat:字符串、符号或正则表达式，字符串分割的依据，默认以空格分隔字符串

N:整型，分割次数，默认为-1,0或-1都将返回所有拆分

Expand:布尔型，分割后的结果是否转换为DataFrame



Join()方法与split()方法结合



2、行列转换

在Pandas处理数据过程中，有时需要对数据进行行列转换或重排，主要使用stack()、unstack（）和pivot方法

DataFrame.stack(level=-1,dropna=True)

Level:索引层级，定义为一个索引或标签，或索引或标签列表，默认值为-1

Dropna:布尔型，默认为True

返回值:DataFrame或Series对象

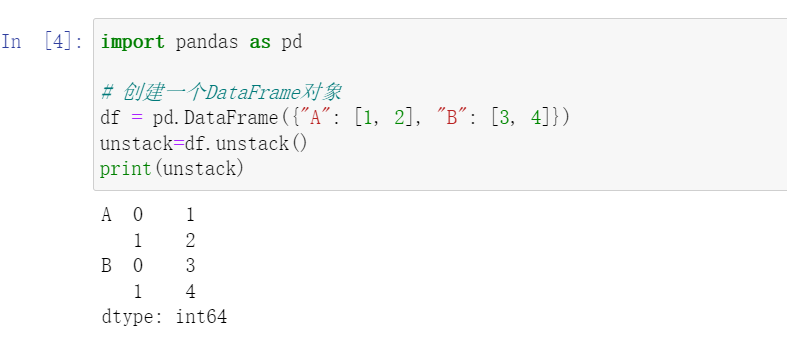


DataFrame.unstack(level=-1,fill\_value=None)

Level:索引层级，定义为一个索引或标签，或索引或标签列表，默认值为-1

fill\_value:整型、字符或字典，如果unstack()产生丢失值，则这个值替换为NaN

返回值:DataFrame或Series对象



DataFrame.pivot(index=None,columns=None,values=None)

Index:字符串或对象，可选参数。列用于创建新DataFrame数据的索引、如果没有则使用现有索引

Columns:字符串或对象，列用于填充新DataFrame的 列

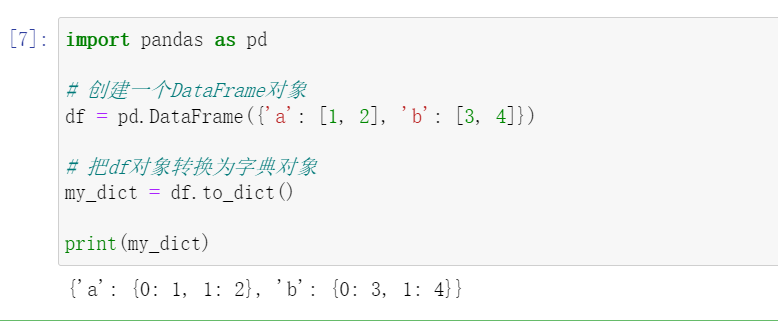
Values: 列用于填充新DataFrame的值，如果未指定，则将使用所有剩余的列，结果将具有分层索引列

返回值:DataFrame或Series对象



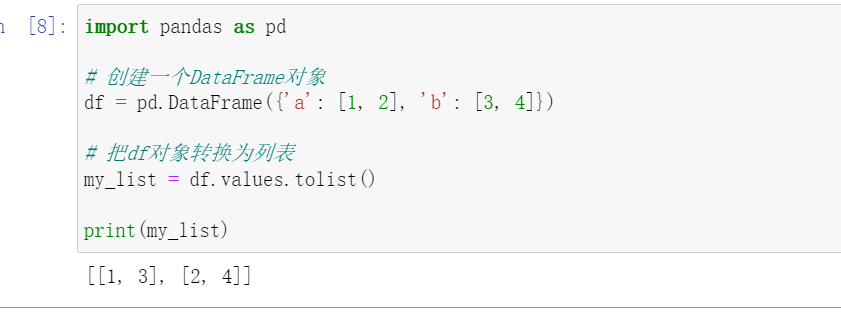
DataFrame转换为字典

DataFrame转换为字典主要使用to\_dict()方法，以索引为字典的键（key）,已列为字典的值(value)



DataFrame转换为列表

DataFrame转换为列表主要使用DataFrame的tolist()



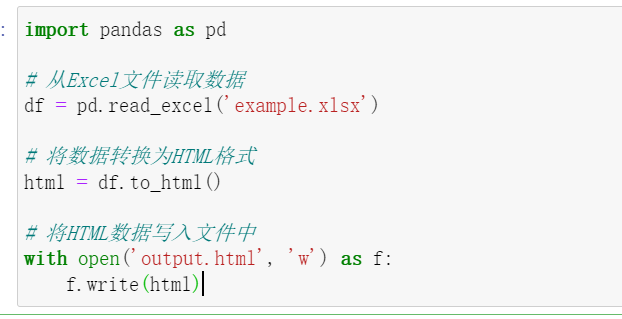
DataFrame转换为元组

首先通过循环语句按行读取DataFrame数据，然后使用元组函数tuple()将其转换为元组



Excel转换为HTML网页格式

要将Excel文件转换为HTML格式，可以使用Python中的pandas库。pandas提供了read\_excel()函数用于读取Excel文件，还提供了to\_html()函数将读取的Excel数据转换为HTML表格格式。



问题讨论：

1. 在print(pd.concat({'A':df1},{'B':df2},axis=1)) 时报错

TypeError: [concat](https://so.csdn.net/so/search?q=concat&spm=1001.2101.3001.7020)() got multiple values for argument 'axis'

原因：pd.concat({'A':df1},{'B':df2},[axis](https://so.csdn.net/so/search?q=axis&spm=1001.2101.3001.7020)=1)，这里面确实是多个axis了，有2个花括号。

改成：pd.concat({'A':df1,'B':df2},axis=1) 就ok 了。

2、报错No module named 'dateutil'

"Unable to import required dependencies:\n" + "\n".join(missing\_dependencies)ImportError: Unable to import required dependencies:dateutil: No module named 'dateutil'

原因：[pandas](https://so.csdn.net/so/search?q=pandas&spm=1001.2101.3001.7020)库是人工从别的环境拷贝的，没有把库：datautil 也拷贝进去。