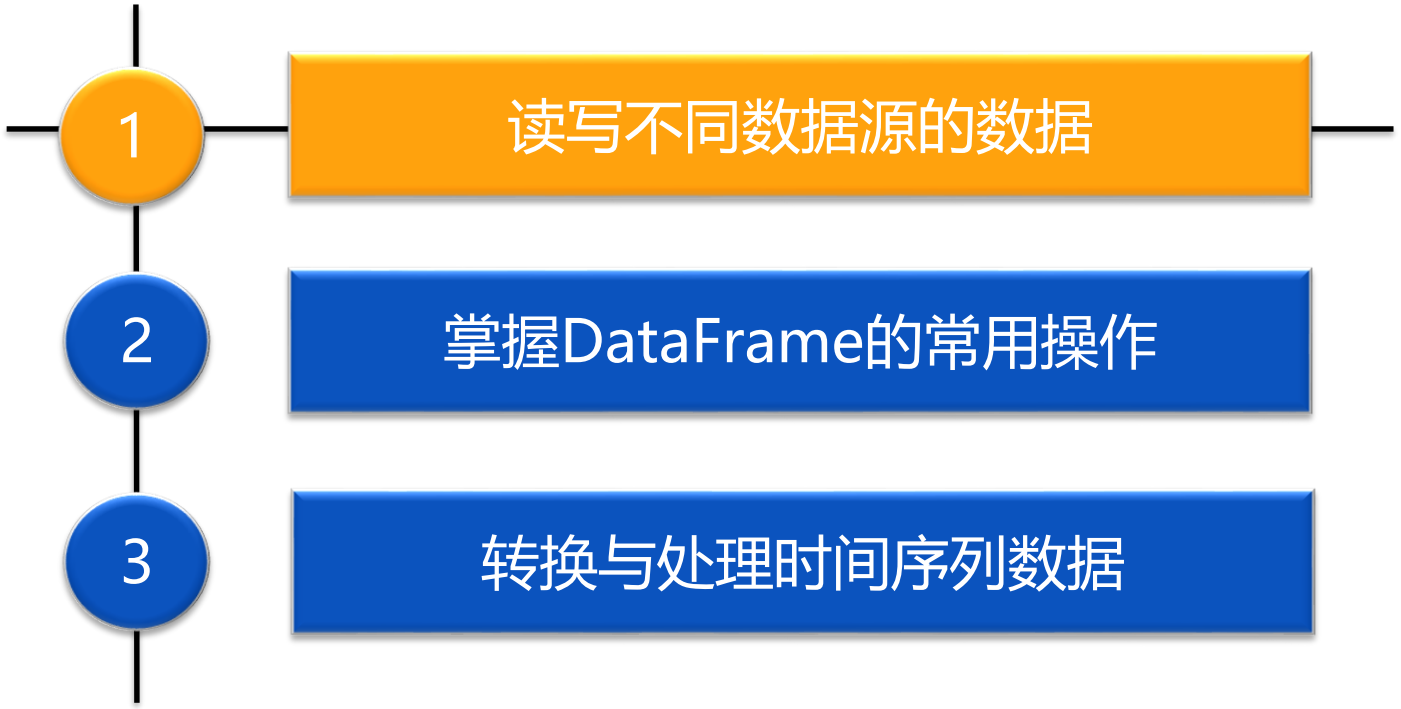
pandas统计分析基础

2019/9/8

目录



读写数据库数据

1.数据库数据读取

- pandas提供了读取与存储关系型数据库数据的函数与方法。除了pandas库外，还需要使用SQLAlchemy库建立对应的数据库连接。SQLAlchemy配合相应数据库的Python连接工具（例如MySQL数据库需要安装mysqlclient或者pymysql库），使用create_engine函数，建立一个数据库连接。

- creat_engine中填入的是一个连接字符串。在使用Python的SQLAlchemy时，MySQL和Oracle数据库连接字符串的格式如下

数据库产品名+连接工具名：//用户名:密码@数据库IP地址:数据库端口号/数据库名称？charset = 数据库数据编码

读写数据库数据

1.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)*

读写数据库数据

1.数据库数据读取

pandas三个数据库数据读取函数的参数几乎完全一致，唯一的区别在于传入的是语句还是表名。

参数名称	说明
sql or table_name	接收string。表示读取的数据的表名或者sql语句。无默认。
con	接收数据库连接。表示数据库连接信息。无默认
index_col	接收int，sequence或者False。表示设定的列作为行名，如果是一个数列则是多重索引。默认为None。
coerce_float	接收boolean。将数据库中的decimal类型的数据转换为pandas中的float64类型的数据。默认为True。
columns	接收list。表示读取数据的列名。默认为None。

读写数据库数据

2.数据库数据存储

数据库数据读取有三个函数，但数据存储则只有一个to_sql方法。

*DataFrame.**to_sql**(name, con, schema=None, if_exists='fail', index=True, index_label=None, dtype=None)*

参数名称	说明
name	接收string。代表数据库表名。无默认。
con	接收数据库连接。无默认。
if_exists	接收fail，replace，append。fail表示如果表名存在则不执行写入操作；replace表示如果存在，将原数据库表删除，再重新创建；append则表示在原数据库表的基础上追加数据。默认为fail。
index	接收boolean。表示是否将行索引作为数据传入数据库。默认True。
index_label	接收string或者sequence。代表是否引用索引名称，如果index参数为True此参数为None则使用默认名称。如果为多重索引必须使用sequence形式。默认为None。
dtype	接收dict。代表写入的数据类型（列名为key，数据格式为values）。默认为None。

读写文本文件

1.文本文件读取

- 文本文件是一种由若干行字符构成的计算机文件，它是一种典型的顺序文件。
- csv是一种逗号分隔的文件格式，因为其分隔符不一定是逗号，又被称为字符分隔文件，文件以纯文本形式存储表格数据（数字和文本）。

读写文本文件

1.文本文件读取

- 使用read_table来读取文本文件。

  *pandas.**read_table**(filepath_or_buffer, sep='\t', header='infer', names=None, index_col=None, dtype=None, engine=None, nrows=None)*

- 使用read_csv函数来读取csv文件。

  *pandas.**read_csv**(filepath_or_buffer, sep='\t', header='infer', names=None, index_col=None, dtype=None, engine=None, nrows=None)*

读写文本文件

1.文本文件读取

read_table和read_csv常用参数及其说明。

参数名称	说明
filepath	接收string。代表文件路径。无默认。
sep	接收string。代表分隔符。read_csv默认为“,” ，read_table默认为制表符“[Tab]”。
header	接收int或sequence。表示将某行数据作为列名。默认为infer，表示自动识别。
names	接收array。表示列名。默认为None。
index_col	接收int、sequence或False。表示索引列的位置，取值为sequence则代表多重索引。默认为None。
dtype	接收dict。代表写入的数据类型（列名为key，数据格式为values）。默认为None。
engine	接收c或者python。代表数据解析引擎。默认为c。
nrows	接收int。表示读取前n行。默认为None。

读写文本文件

1.文本文件读取

- read_table和read_csv函数中的sep参数是指定文本的分隔符的，如果分隔符指定错误，在读取数据的时候，每一行数据将连成一片。
- header参数是用来指定列名的，如果是None则会添加一个默认的列名。
- encoding代表文件的编码格式，常用的编码有utf-8、utf-16、gbk、gb2312、gb18030等。如果编码指定错误数据将无法读取，IPython解释器会报解析错误。

读写文本文件

2.文本文件储存

文本文件的存储和读取类似，结构化数据可以通过pandas中的to_csv函数实现以csv文件格式存储文件。

*DataFrame.**to_csv**(path_or_buf=None, sep=',', na_rep='', columns=None, header=True, index=True,index_label=None,mode='w',encoding=None)*

参数名称	说明	参数名称	说明
path_or_buf	接收string。代表文件路径。无默认。	index	接收boolean，代表是否将行名（索引）写出。默认为True。
sep	接收string。代表分隔符。默认为“,”。	index_labels	接收sequence。表示索引名。默认为None。
na_rep	接收string。代表缺失值。默认为“”。	mode	接收特定string。代表数据写入模式。默认为w。
columns	接收list。代表写出的列名。默认为None。	encoding	接收特定string。代表存储文件的编码格式。默认为None。
header	接收boolean，代表是否将列名写出。默认为True。		

读写Excel文件

1.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)*

参数名称	说明
io	接收string。表示文件路径。无默认。
sheetname	接收string、int。代表excel表内数据的分表位置。默认为0。
header	接收int或sequence。表示将某行数据作为列名。默认为infer，表示自动识别。
names	接收int、sequence或者False。表示索引列的位置，取值为sequence则代表多重索引。默认为None。
index_col	接收int、sequence或者False。表示索引列的位置，取值为sequence则代表多重索引。默认为None。
dtype	接收dict。代表写入的数据类型（列名为key，数据格式为values）。默认为None。

读写Excel文件

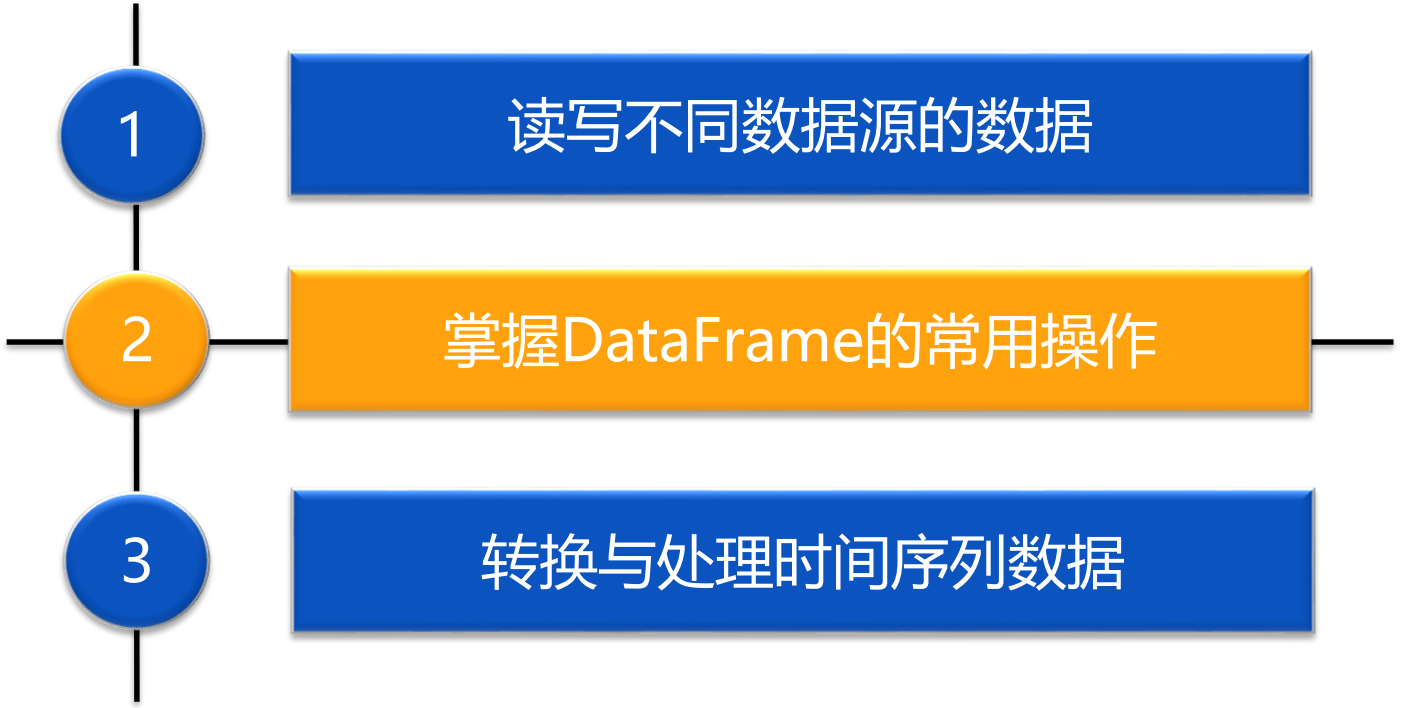
2.Excel文件储存

- 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, sheetname=None', na_rep='', header=True, index=True, index_label=None, mode='w', encoding=None)*

- to_csv方法的常用参数基本一致，区别之处在于指定存储文件的文件路径参数名称为excel_writer，并且没有sep参数，增加了一个sheetnames参数用来指定存储的Excel sheet的名称，默认为sheet1。

目录



查看DataFrame的常用属性

基础属性

函数	返回值
values	元素
index	索引
columns	列名
dtypes	类型
size	元素个数
ndim	维度数
shape	数据形状（行列数目）

查改增删DataFrame数据

1.查看访问DataFrame中的数据——数据基本查看方式

- **对单列数据的访问**：DataFrame的单列数据为一个Series。根据DataFrame的定义可以知晓DataFrame是一个带有标签的二维数组，每个标签相当每一列的列名。有以下两种方式来实现对单列数据的访问。
  - 以字典访问某一个key的值的方式使用对应的列名，实现单列数据的访问。
  - 以属性的方式访问，实现单列数据的访问。（不建议使用，易引起混淆）

查改增删DataFrame数据

1.查看访问DataFrame中的数据——数据基本查看方式

- **对某一列的某几行访问**：访问DataFrame中某一列的某几行时，单独一列的DataFrame可以视为一个Series（另一种pandas提供的类，可以看作是只有一列的DataFrame），而访问一个Series基本和访问一个一维的ndarray相同。
- **对多列数据访问**：访问DataFrame多列数据可以将多个列索引名称视为一个列表，同时访问DataFrame多列数据中的多行数据和访问单列数据的多行数据方法基本相同。

查改增删DataFrame数据

1.查看访问DataFrame中的数据——数据基本查看方式

- **对某几行访问**：
  - 如果只是需要访问DataFrame某几行数据的实现方式则和上述的访问多列多行相似，选择所有列，使用“:”代替即可。
  - head和tail也可以得到多行数据，但是用这两种方法得到的数据都是从开始或者末尾获取的连续数据。默认参数为访问5行，只要在方法后方的“()”中填入访问行数即可实现目标行数的查看。

查改增删DataFrame数据

1.查看访问DataFrame中的数据——loc，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。loc方法使用方法如下。

  *DataFrame.**loc**[行索引名称或条件, 列索引名称]*

- iloc和loc区别是iloc接收的必须是行索引和列索引的位置。iloc方法的使用方法如下。

  *DataFrame.**iloc**[行索引位置, 列索引位置]*

查改增删DataFrame数据

1.查看访问DataFrame中的数据——loc，iloc访问方式

- 使用loc方法和iloc实现多列切片，其原理的通俗解释就是将多列的列名或者位置作为一个列表或者数据传入。
- 使用loc，iloc方法可以取出DataFrame中的任意数据。
- 在loc使用的时候内部传入的行索引名称如果为一个区间，则前后均为闭区间；iloc方法使用时内部传入的行索引位置或列索引位置为区间时，则为前闭后开区间。
- loc内部还可以传入表达式，结果会返回满足表达式的所有值。

查改增删DataFrame数据

1.查看访问DataFrame中的数据——loc，iloc访问方式

- 若使用detail.iloc[detail['order_id']=='458',[1,5]]读取数据，则会报错，原因在于此处条件返回的为一个布尔值Series，而iloc可以接收的数据类型并不包括Series。根据Series的构成只要取出该Series的values就可以了。需改为detail.iloc[(detail['order_id']=='458').values,[1,5]])。
- loc更加灵活多变，代码的可读性更高，iloc的代码简洁，但可读性不高。具体在数据分析工作中使用哪一种方法，根据情况而定，大多数时候建议使用loc方法。

查改增删DataFrame数据

1.查看访问DataFrame中的数据——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收索引名称也可以接收索引位置。其使用方法如下。

  *DataFrame.**ix**[行索引的名称或位置或者条件, 列索引名称或位置]*

- 使用ix方法时有个注意事项，第一条，当索引名称和位置存在部分重叠时，ix默认优先识别名称。

查改增删DataFrame数据

1.查看访问DataFrame中的数据——切片方法之ix

控制ix方法需要注意以下几点。

- 使用ix参数时，尽量保持行索引名称和行索引位置重叠，使用时就无须考虑取值时区间的问题。一律为闭区间。
- 使用列索引名称，而非列索引位置。主要用来保证代码可读性。
- 使用列索引位置时，需要注解。同样保证代码可读性。
- 除此之外ix方法还有一个缺点，就是在面对数据量巨大的任务的时候，其效率会低于loc和iloc方法，所以在日常的数据分析工作中建议使用loc和iloc方法来执行切片操作。

查改增删DataFrame数据

2.更新修改DataFrame中的数据

- 更改DataFrame中的数据，原理是将这部分数据提取出来，重新赋值为新的数据。
- 需要注意的是，数据更改直接针对DataFrame原数据更改，操作无法撤销，如果做出更改，需要对更改条件做确认或对数据进行备份。

查改增删DataFrame数据

3.为DataFrame增添数据

- DataFrame添加一列的方法非常简单，只需要新建一个列索引。并对该索引下的数据进行赋值操作即可。
- 新增的一列值是相同的则直接赋值一个常量即可。

查改增删DataFrame数据

4.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，drop方法的用法如下。

- axis为0时表示删除行，axis为1时表示删除列。

$$drop(labels, axis=0, level=None, inplace=False, errors='raise')$$

- 常用参数如下所示。

参数名称	说明
labels	接收string或array。代表删除的行或列的标签。无默认。
axis	接收0或1。代表操作的轴向。默认为0。
levels	接收int或者索引名。代表标签所在级别。默认为None。
inplace	接收boolean。代表操作是否对原数据生效。默认为False。

描述分析DataFrame数据

1.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。

函数名称	说明	函数名称	说明
np.min	最小值	np.max	最大值
np.mean	均值	np.ptp	极差
np.median	中位数	np.std	标准差
np.var	方差	np.cov	协方差

描述分析DataFrame数据

1.数值型特征的描述性统计—— pandas描述性统计方法

- pandas还提供了更加便利的方法来计算均值 ，如detail['amounts'].mean()。
- 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。

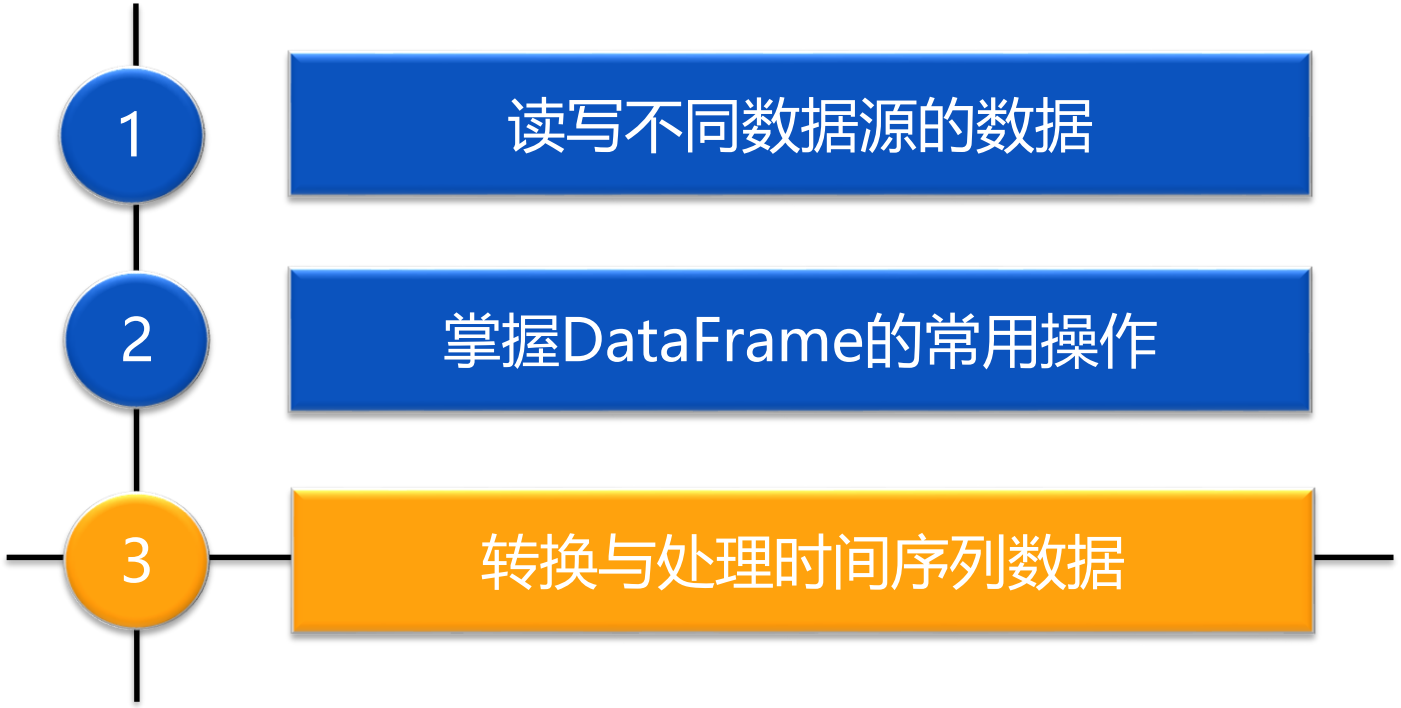
方法名称	说明	方法名称	说明
min	最小值	max	最大值
mean	均值	ptp	极差
median	中位数	std	标准差
var	方差	cov	协方差
sem	标准误差	mode	众数
skew	样本偏度	kurt	样本峰度
quantile	四分位数	count	非空值数目
describe	描述统计	mad	平均绝对离差

描述分析DataFrame数据

2.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categories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。

目录



转换字符串时间为标准时间

pandas时间相关的类

- 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。

类名称	说明
Timestamp	最基础的时间类。表示某个时间点。在绝大多数的场景中的时间数据都是Timestamp形式的时间。
Period	表示单个时间跨度，或者某个时间段，例如某一天，某一小时等。
Timedelta	表示不同单位的时间，例如1天，1.5小时，3分钟，4秒等，而非具体的某个时间段。
DatetimeIndex	一组Timestamp构成的Index，可以用来作为Series或者DataFrame的索引。
PeriodtimeIndex	一组Period构成的Index，可以用来作为Series或者DataFrame的索引。
TimedeltaIndex	一组Timedelta构成的Index，可以用来作为Series或者DataFrame的索引。

转换字符串时间为标准时间

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。

转换字符串时间为标准时间

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。

转换字符串时间为标准时间

DatetimeIndex与PeriodIndex函数及其参数说明

- DatetimeIndex和PeriodIndex两者区别在日常使用的过程中相对较小，其中DatetimeIndex是用来指代一系列时间点的一种数据结构，而PeriodIndex则是用来指代一系列时间段的数据结构。

参数名称	说明
data	接收array。表示DatetimeIndex的值。无默认。
freq	接收string。表示时间的间隔频率。无默认。
start	接收string。表示生成规则时间数据的起始点。无默认。
periods	表示需要生成的周期数目。无默认。
end	接收string。表示生成规则时间数据的终结点。无默认。
tz	接收timezone。表示数据的时区。默认为None。
name	接收int，string。默认为空。指定DatetimeIndex的名字。

提取时间序列数据信息

Timestamp类常用属性

- 在多数涉及时间相关的数据处理，统计分析的过程中，需要提取时间中的年份，月份等数据。使用对应的Timestamp类属性就能够实现这一目的。
- 结合Python列表推导式，可以实现对DataFrame某一列时间信息数据的提取。

属性名称	说明	属性名称	说明
year	年	week	一年中第几周
month	月	quarter	季节
day	日	weekofyear	一年中第几周
hour	小时	dayofyear	一年中的第几天
minute	分钟	dayofweek	一周第几天
second	秒	weekday	一周第几天
date	日期	weekday_name	星期名称
time	时间	is_leap_year	是否闰年

提取时间序列数据信息

在DatetimeIndex和PeriodIndex中提取信息

- 在DatetimeIndex和PeriodIndex中提取对应信息可以以类属性方式实现。
- 值得注意的是PeriodIndex相比于DatetimeIndex少了weekday_name属性，所以不能够用该属性提取星期名称数据。若想要提取信息名称可以通过提取weekday属性，而后将0-6四个标签分别赋值为Monday至Sunday。

加减时间数据

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。

周期名称	单位	说明	周期名称	单位	说明
weeks	无	星期	seconds	s	秒
days	D	天	milliseconds	ms	毫秒
hours	h	小时	microseconds	us	微妙
minutes	m	分	nanoseconds	ns	纳秒

加减时间数据

Timedelta类

- 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。

Thank you!