In [ ]:

```
df2.drop_duplicates(['类型', '所在省份'])
```

In [ ]:

```
df2.drop_duplicates(['类型', '所在省份'], keep = False)
```

利用查重标识结果直接删除

df[~df.duplicated()]

In [ ]:

```
df2[~df2.duplicated(['类型', '所在省份'])]
```

8.3 实战：进一步整理PM2.5数据

要求:

PM2.5数据中数值-999表示缺失，请将这些数据替换为np.nan
基于上述处理结果，删除缺失值记录
在数据中查找到PM2.5数值完全相同的记录
在数据中查找到同一年中PM2.5数值完全相同的记录

9 处理日期时间变量

9.1 建立Timestamp类和Period类

Pandas中和时间有关的类

Class名称	功能/用途	创建方法
Timestamp	表示具体的日期时间(Timestamp)	to_datetime, Timestamp
DatetimeIndex	日期时间(Timestamp)的索引	to_datetime, date_range, bdate_range, DatetimeIndex
Period	表示具体的日期时间范围(Peroid)	Period
PeriodIndex	Period的索引	period_range, PeriodIndex

9.1.1 Timestamp对象

In [ ]:

```
from datetime import datetime # 从datetime包中引入datetime
datetime(2012, 5, 1)
```

In [ ]:

```
pd.Timestamp(datetime(2012, 5, 1))
```

In [ ]:

```
pd.Timestamp(datetime(2012, 5, 1, 1, 2, 3))
```

In [ ]:

```
pd.Timestamp('2012-05-01 1:2:3')
```

In [ ]:

```
pd.Timestamp(2012, 5, 1)
```

9.1.2 Peroid对象

可以被看作是简化之后的Timestamp对象

> 由于详细数据的不完整，而表示的是一段时间，而不是一个时点
> 但是实际使用中很可能是按照时点在使用

很多使用方法和Timestamp相同，因此不再详细介绍

In [ ]:

```
pd.Period('2011-01')
```

In [ ]:

```
pd.Period('2012-05', freq='D')
```

9.2 将数据转换为Timestamp类

9.2.1 使用pd.Timestamp()直接转换

In [ ]:

```
pd.Timestamp(bj08['Date (LST)'][0])
```

In [ ]:

```
bj08['Date (LST)'].apply(pd.Timestamp)
```

9.2.2 用to_datetime进行批量转换

pd.to_datetime(

```
    arg : 需要转换为Timestamp类的数值
        integer, float, string, datetime, list, tuple, 1-d array,Series
    errors = 'raise' : {'ignore', 'raise', 'coerce'}
        'raise', 抛出错误
        'coerce', 设定为 NaT
        'ignore', 返回原值
    短日期的解释方式：类似"10/11/12"这样的数据如何解释
        dayfirst = 'False' : 数值是否day在前
        yearfirst = 'False' : 数值是否year在前，该设定优先
    box = True : 是否返回为DatetimeIndex，False时返回ndarray数组
    format = None : 需要转换的字符串格式设定

)
```

In [ ]:

```
pd.to_datetime(datetime(2012, 5, 1, 1, 2, 3))
```

In [ ]:

```
pd.to_datetime('2012-05-01 1:2:3')
```

In [ ]:

```
pd.to_datetime(['2005/11/23', '2010.12.31'])
```

In [ ]:

```
pd.to_datetime(bj08['Date (LST)'], format = "%Y-%m-%d %H:%M")
```

9.2.3 基于所需的变量列合成Timestamp类

In [ ]:

```
pd.to_datetime(bj08[['Year', 'Month', 'Day', 'Hour']])
```

9.3 使用DatetimeIndex类

DatetimeIndex类对象除了拥有Index类对象的所有功能外，还针对日期时间的特点有如下增强：

基于日期时间的各个层级做快速索引操作
快速提取所需的时间层级
按照所指定的时间范围做快速切片

9.3.1 建立DatetimeIndex对象

9.3.1.1 建立索引时自动转换

使用Timestamp对象建立索引，将会自动转换为DatetimeIndex对象

In [ ]:

```
bj08idx = bj08.set_index(pd.to_datetime(bj08['Date (LST)']))
print(type(bj08idx.index))
bj08idx
```

9.3.1.2 使用date_range建立DatetimeIndex对象

这种建立方式主要是和reindx命令配合使用，以快速完成对时间序列中缺失值的填充工作

pd.date_range(

start /end = None : 日期时间范围的起点/终点，均为类日期时间格式的字符串/数据
periods = None : 准备生成的总记录数
freq = 'D' : 生成记录时的时间周期，可以使用字母和数值倍数的组合，如'5H'
name = None : 生成的DatetimeIndex对象的名称

)

pd.bdate_range(

主要参数和pd.date_range几乎完全相同，但默认freq = 'B' (business daily)
另外附加了几个针对工作日/休息日筛选的参数

)

In [ ]:

```
pd.date_range('1/1/2012', periods=5, freq='M')
```

9.3.2 基于索引的快速切片操作

In [ ]:

```
bj08idx["2008-11-1":"2008-11-5"]
```

In [ ]:

```
bj08idx["2008-11"]
```

In [ ]:

```
bj08idx["2008-11-1":"2008-11-5 9:00:00"]
```

9.4 对时间序列做基本处理

9.4.1 序列的分组汇总

9.4.1.1 直接取出索引的相应层级

DatetimeIndex对象可直接引用的Attributes

```
    date : Returns numpy array of python datetime.date objects
    time : Returns numpy array of datetime.time

    year : The year of the datetime
    quarter : The quarter of the date
    month : The month as January=1, December=12
    week : The week ordinal of the year
    weekday : The day of the week with Monday=0, Sunday=6
    weekday_name : The name of day in a week (ex: Friday)
    weekofyear : The week ordinal of the year
    day : The days of the datetime
    dayofweek : The day of the week with Monday=0, Sunday=6
    dayofyear : The ordinal day of the year
    days_in_month : The number of days in the month
    daysinmonth : The number of days in the month
    hour : The hours of the datetime
    minute : The minutes of the datetime
    second : The seconds of the datetime
    microsecond : The microseconds of the datetime
    nanosecond : The nanoseconds of the datetime

    is_leap_year : if the date belongs to a leap year
    is_month_end : if last day of month (defined by frequency)
    is_month_start : if first day of month (defined by frequency)
    is_quarter_end : if last day of quarter (defined by frequency)
    is_quarter_start : if first day of quarter (defined by frequency)
    is_year_end : if last day of year (defined by frequency)
    is_year_start : if first day of year (defined by frequency)
```

In [ ]:

```
bj08idx.index.hour
```

9.4.1.2 直接使用groupby方法进行汇总

In [ ]:

```
bj08idx.groupby(bj08idx.index.month).max()
```

9.4.1.3 使用功能更强的resample函数

df.resample()

使用上比groupby更简单(输入更简洁)
可以将数值和汇总单位进行组合，实现更复杂的汇总计算

Alias	Description
B	business day frequency
C	custom business day frequency
D	calendar day frequency
W	weekly frequency
M	month end frequency
SM	semi-month end frequency (15th and end of month)
BM	business month end frequency
CBM	custom business month end frequency
MS	month start frequency
SMS	semi-month start frequency (1st and 15th)
BMS	business month start frequency
CBMS	custom business month start frequency
Q	quarter end frequency
BQ	business quarter end frequency
QS	quarter start frequency
BQS	business quarter start frequency
A, Y	year end frequency
BA, BY	business year end frequency
AS, YS	year start frequency
BAS, BYS	business year start frequency
BH	business hour frequency
H	hourly frequency
T, min	minutely frequency
S	secondly frequency
L, ms	milliseconds
U, us	microseconds
N	nanoseconds

In [ ]:

```
bj08idx.resample('3D').mean()
```

9.4.2 序列的缺失值处理

时间序列要求记录的时间点连贯无缺失，因此需要：

首先建立针对整个时间范围的完整序列框架
随后针对无数据的时间点进行缺失值处理

df.reindex()是完成该任务的强有力工具

In [ ]:

```
bj09idx = bj09.set_index(pd.to_datetime(bj09['Date (LST)']))
bj09idx = bj09idx[bj09idx.Value > 0]
bj09idx
```

In [ ]:

```
idx = pd.date_range(start = '2009-02-01 00:00:00',
                    end = '2009-12-31 23:00:00', freq='H')
idx
```

In [ ]:

```
bj09idx.reindex(idx)
```

In [ ]:

```
bj09idx[bj09idx.index.duplicated()]
```

In [ ]:

```
bj09idx['2009-03-08']
```

In [ ]:

```
bj09idx[~bj09idx.index.duplicated()].reindex(idx)
```

In [ ]:

```
bj09idx[~bj09idx.index.duplicated()].reindex(idx, method = 'bfill')
```

9.4.3 序列数值平移

df.shift(

periods = 1 : 希望移动的周期数
freq : 时间频度字符串
axis = 0

)

In [ ]:

```
bj08idx.shift(3)
```

9.5 实战：建立时间索引

要求:

自行练习分别使用Date (LST)和年、月、日、时变量建立DatetimeIndex
尝试只使用年、月、日建立Period对象，然后转换为DatetimeIndex
基于DatetimeIndex，进一步完成原先在第7章练习中完成过的任务
- 计算出每年PM2.5的平均值、中位数、最大值、最小值
- 计算出每年PM2.5值大于200、300、500的天数
- 将PM2.5数据整理为以年为行，月为列，单元格为最大值的宽表形式
- 将2009年和2012年的数据分别提取出来，然后合并为一个数据框
- 将数据转换为每年一列的宽表格式

10 数据的图形展示

10.1 配置绘图系统环境