time series

In [1]:
```
import pandas as pd
from datetime import datetime
from datetime import timedelta
from dateutil.parser import parse
import numpy as np
```

# 日期和时间数据的类型

## *datetime* 类型

In [2]:
```
now = datetime.now()
now
```

Out[2]:
```
datetime.datetime(2021, 10, 9, 21, 38, 2, 642419)
```

In [3]:
```
now.year, now.month, now.day, now.hour, now.minute, now.second
```

Out[3]:
```
(2021, 10, 9, 21, 38, 2)
```

## *timedelta* 类型

In [4]:
```
delta = datetime(2021, 5, 1, 0, 8, 40) - datetime(2019, 5, 1)
delta
```

Out[4]:
```
datetime.timedelta(days=731, seconds=520)
```

In [5]:
```
start = datetime(2019, 5, 1)
start + timedelta(20, 20)
```

Out[5]:
```
datetime.datetime(2019, 5, 21, 0, 0, 20)
```

# datetime 与 string 的转换

## *pd.to_datetime( list )* : datetime → string

to_datetime 方法可以解析多种不同的日期表示形式

In [6]:
```
datestrs = ['2019/5/1/00:00:00', '2021-5-1 00:00:00']
pd.to_datetime(datestrs)
```

Out[6]:
```
DatetimeIndex(['2019-05-01', '2021-05-01'], dtype='datetime64[ns]', freq=None)
```

## *str(datetime), datetime.strftime('%Y-%m-%d')* : datetime → string

In [7]:
```
now = datetime.now()
now, str(now), now.strftime('%Y-%m-%d')
```

Out[7]:
```
(datetime.datetime(2021, 10, 9, 21, 38, 2, 880786),
 '2021-10-09 21:38:02.880786',
 '2021-10-09')
```

*datetime* 格式定义

| 代码 | 说明 |
|---|---|
| %Y, %y | 4 位数的年, 2 位数的年 |
| %m , %h | 2 位数的月 [01, 12], 英文缩写表示的月 [Jan, Feb, ..., Dec] |
| %d | 2 位数的日 [01, 31] |

| 代码 | 说明 |
|---|---|
| %H , %I | 24 小时制的时 [00, 23], 12 小时制的时 [01, 12] |
| %M, %S | 2 位数的分 [00, 59], 2 位数的秒 [00, 59] |
| %w | 整数表示的星期几 [0, 6] |
| %W | 2 位数的每年的第几周 [00， 53], 以星期一为每周的开始 |
| %U | 2 位数的每年的第几周 [00， 53], 以星期日为每周的开始 |
| %F | %Y-%m-%d |
| %D | %m/%d/%y |

### *datetime.strptime(string, '%Y-%m-%d')* : string → datetime

In [8]:
```
value = '2019-05-01'
datetime.strptime(value, '%Y-%m-%d')
```

Out[8]:
```
datetime.datetime(2019, 5, 1, 0, 0)
```

In [9]:
```
datestrs = ['7/25/1998', '11/24/1997']
[datetime.strptime(x, '%m/%d/%Y') for x in datestrs]
```

Out[9]:
```
[datetime.datetime(1998, 7, 25, 0, 0), datetime.datetime(1997, 11, 24, 0, 0)]
```

### *dateutil.parser.parse( string, dayfirst )* : string → datetime

In [10]:
```
parse('2019-05-01')
```

Out[10]:
```
datetime.datetime(2019, 5, 1, 0, 0)
```

In [11]:
```
parse('May 1, 2019 10:45 PM')
```

Out[11]:
```
datetime.datetime(2019, 5, 1, 22, 45)
```

在国际通用的格式中, 日出现在月的前面很普遍, 传入 dayfirst=True 即可解决这个问题

In [12]:
```
parse('1/5/2019'), parse('1/5/2019', dayfirst=True)
```

Out[12]:
```
(datetime.datetime(2019, 1, 5, 0, 0), datetime.datetime(2019, 5, 1, 0, 0))
```

## 时间序列的索引

In [13]:
```
dates = [datetime(2011, 1, 2), datetime(2011, 1, 5),
         datetime(2011, 1, 7), datetime(2011, 1, 8),
         datetime(2011, 1, 10), datetime(2011, 1, 12)]
ts = pd.Series(np.random.randn(6), index=dates)
ts
```

Out[13]:
```
2011-01-02   -1.248703
2011-01-05   -0.072871
2011-01-07    1.672176
2011-01-08   -0.761810
2011-01-10   -0.044322
2011-01-12   -0.701274
dtype: float64
```

可以传入一个可以被解释为日期的字符串进行切片

In [14]:
```
ts = pd.Series(np.arange(1000),
                       index=pd.date_range('1/1/2000', periods=1000))
ts.head()
```

Out[14]:
```
2000-01-01    0
2000-01-02    1
2000-01-03    2
2000-01-04    3
2000-01-05    4
Freq: D, dtype: int32
```

In [15]:
```
ts['2000-1-1':'2000-1-3']
```

Out[15]:
```
2000-01-01    0
2000-01-02    1
2000-01-03    2
Freq: D, dtype: int32
```

对于较长的时间序列，只需传入“年”或“年月”即可轻松选取数据的切片

In [16]:
```
ts['2001'].head()
```

Out[16]:
```
2001-01-01    366
2001-01-02    367
2001-01-03    368
2001-01-04    369
2001-01-05    370
Freq: D, dtype: int32
```

In [17]:
```
ts['2001-2'].head()
```

Out[17]:
```
2001-02-01    397
2001-02-02    398
2001-02-03    399
2001-02-04    400
2001-02-05    401
Freq: D, dtype: int32
```

传入 datetime 对象也可以进行切片

In [18]:
```
ts[datetime(2001,1,7):datetime(2001,1,8)]
```

Out[18]:
```
2001-01-07    372
2001-01-08    373
Freq: D, dtype: int32
```

## *ts.truncate( before, after )* :

before : 向后切片

In [19]:
```
ts.truncate(before='2001-9').head()
```

Out[19]:
```
2001-09-01    609
2001-09-02    610
2001-09-03    611
2001-09-04    612
2001-09-05    613
Freq: D, dtype: int32
```

after : 向前切片

In [20]:
```
ts.truncate(after='2000-2-4').head()
```

Out[20]:
```
2000-01-01    0
2000-01-02    1
2000-01-03    2
2000-01-04    3
2000-01-05    4
Freq: D, dtype: int32
```

In [21]:
```
ts.truncate(before = '2000-1-1', after='2000-1-4')
```

Out[21]:
```
2000-01-01    0
2000-01-02    1
2000-01-03    2
2000-01-04    3
Freq: D, dtype: int32
```

# 时间的范围、频率和位移

## *pd.date_range( start, end, periods, freq )* :

In [22]:
```
pd.date_range('2019-05-01', '2021-05-01')
```

Out[22]:
```
DatetimeIndex(['2019-05-01', '2019-05-02', '2019-05-03', '2019-05-04',
               '2019-05-05', '2019-05-06', '2019-05-07', '2019-05-08',
               '2019-05-09', '2019-05-10',
               ...
               '2021-04-22', '2021-04-23', '2021-04-24', '2021-04-25',
               '2021-04-26', '2021-04-27', '2021-04-28', '2021-04-29',
               '2021-04-30', '2021-05-01'],
              dtype='datetime64[ns]', length=732, freq='D')
```

In [23]:
```
pd.date_range('2019-05-01', periods=3)
```

Out[23]:
```
DatetimeIndex(['2019-05-01', '2019-05-02', '2019-05-03'], dtype='datetime64[ns]', freq='D')
```

***freq* = *'BM'*** : 'business end of month' 每月最后一个工作日

**时间的基础频率**

| *freq* | **说明** |
|---|---|
| H, T/min, S, L/ms, U | 每小时, 每分钟, 每秒, 每毫秒, 每微秒 |
| D, M, MS | 每日历日, 每月的最后一个日历日, 每月的第一个日历日 |
| B, BM, BMS | 每工作日, 每月的最后一个工作日, 每月的第一个工作日 |
| W-MON, W-TUE, ..., W-SUN | 每周, 从指定的星期几开始算起 |
| WOM-1MON, WOM-2MON, ..., WOM-4SUN | 每月, 从指定的第几周的星期几开始算起 |
| Q-JAN, Q-FEB, ..., Q-DEC | 每季, 从指定的月份的最后一个日历日开始算起 |
| QS-JAN, QS-FEB, ..., QS-DEC | 每季, 从指定的月份的第一个日历日开始算起 |
| BQ-JAN, BQ-FEB, ..., BQ-DEC | 每季, 从指定的月份的最后一个工作日开始算起 |
| BQS-JAN, BQS-FEB, ..., BQS-DEC | 每季, 从指定的月份的第一个工作日开始算起 |
| A-JAN, A-FEB, ..., A-DEC | 每年, 从指定的月份的最后一个日历日开始算起 |
| AS-JAN, AS-FEB, ..., AS-DEC | 每年, 从指定的月份的第一个日历日开始算起 |
| BA-JAN, BA-FEB, ..., BA-DEC | 每年, 从指定的月份的最后一个工作日开始算起 |
| BAS-JAN, BAS-FEB, ..., BAS-DEC | 每年, 从指定的月份的第一个工作日开始算起 |

In [24]:
```
pd.date_range('2020-01-01', '2021-01-01', freq='M')
```

Out[24]:
```
DatetimeIndex(['2020-01-31', '2020-02-29', '2020-03-31', '2020-04-30',
               '2020-05-31', '2020-06-30', '2020-07-31', '2020-08-31',
               '2020-09-30', '2020-10-31', '2020-11-30', '2020-12-31'],
              dtype='datetime64[ns]', freq='M')
```

In [25]:
```
pd.date_range('2020-01-01', '2021-01-01', freq='4M')
```

Out[25]:
```
DatetimeIndex(['2020-01-31', '2020-05-31', '2020-09-30'], dtype='datetime64[ns]', freq='4M')
```

In [26]:
```
pd.date_range('2020-01-01', '2020-01-02', freq='2h30min')
```

Out[26]:
```
DatetimeIndex(['2020-01-01 00:00:00', '2020-01-01 02:30:00',
               '2020-01-01 05:00:00', '2020-01-01 07:30:00',
               '2020-01-01 10:00:00', '2020-01-01 12:30:00',
               '2020-01-01 15:00:00', '2020-01-01 17:30:00',
               '2020-01-01 20:00:00', '2020-01-01 22:30:00'],
              dtype='datetime64[ns]', freq='150T')
```

## *ts.shift( periods, freq )*：时间序列的平移

In [27]:
```
ts = pd.Series(np.arange(1, 5),
               index=pd.date_range('1/1/2000', periods=4, freq='D'))
ts
```

Out[27]:
```
2000-01-01    1
2000-01-02    2
2000-01-03    3
2000-01-04    4
Freq: D, dtype: int32
```

Series 和 DataFrame 都有一个 shift 方法用于执行单纯的前移或后移操作，保持索引不变

In [28]:
```
ts.shift(-1), ts / ts.shift(1) -1 #计算随时间的变化
```

Out[28]:
```
(2000-01-01    2.0
 2000-01-02    3.0
 2000-01-03    4.0
 2000-01-04    NaN
 Freq: D, dtype: float64,
 2000-01-01         NaN
 2000-01-02    1.000000
 2000-01-03    0.500000
 2000-01-04    0.333333
 Freq: D, dtype: float64)
```

***ts.shift(-1) vs ts.shfit(-1, freq='D')***

In [29]:
```
ts.shift(-1, freq='D') # 时间序列向后平移3个 'D'
```

Out[29]:
```
1999-12-31    1
2000-01-01    2
2000-01-02    3
2000-01-03    4
Freq: D, dtype: int32
```

In [30]:
```
ts.shift(1, freq='3D') # 时间序列向后平移3个 'D'
```

Out[30]:
```
2000-01-04    1
2000-01-05    2
2000-01-06    3
2000-01-07    4
Freq: D, dtype: int32
```

In [31]:
```
ts.shift(1, freq='90T') # 时间序列向后平移90个 'min'
```

Out[31]:
```
2000-01-01 01:30:00    1
2000-01-02 01:30:00    2
2000-01-03 01:30:00    3
2000-01-04 01:30:00    4
Freq: D, dtype: int32
```

## *Hour(), Minute(), Day(), MonthEnd()*：通过偏移量对日期进行平移

> ***Hour(), Minute(), Day(), MonthEnd(), ...***：日期偏移量（ date offset ）对象

In [32]:
```
from pandas.tseries.offsets import Day, MonthEnd
now = datetime.now()
now
```

Out[32]:
```
datetime.datetime(2021, 10, 9, 21, 38, 3, 837907)
```

In [33]:
```
now + 3 * Day(), now + Day(3)
```

Out[33]:
```
(Timestamp('2021-10-12 21:38:03.837907'),
 Timestamp('2021-10-12 21:38:03.837907'))
```

In [34]:
```
now + 1 * MonthEnd()
```

Out[34]:
```
Timestamp('2021-10-31 21:38:03.837907')
```

### *offset.rollforward( datetime ), offset.rollback( datetime )*：控制将日期向前或向后“滚动”

In [35]:
```
offset = MonthEnd()
offset.rollforward(now), offset.rollback(now)
```

Out[35]:
```
(Timestamp('2021-10-31 21:38:03.837907'),
 Timestamp('2021-09-30 21:38:03.837907'))
```

In [36]:
```
ts = pd.Series(np.ones(20), index=pd.date_range('1/15/2000', periods=20, freq='4d'))
ts
```

Out[36]:
```
2000-01-15    1.0
2000-01-19    1.0
2000-01-23    1.0
2000-01-27    1.0
2000-01-31    1.0
2000-02-04    1.0
2000-02-08    1.0
2000-02-12    1.0
2000-02-16    1.0
2000-02-20    1.0
2000-02-24    1.0
2000-02-28    1.0
2000-03-03    1.0
2000-03-07    1.0
2000-03-11    1.0
2000-03-15    1.0
2000-03-19    1.0
2000-03-23    1.0
2000-03-27    1.0
2000-03-31    1.0
Freq: 4D, dtype: float64
```

### *ts.groupby( offset.rollforward )*

In [37]:
```
offset = MonthEnd()
ts.groupby(MonthEnd().rollforward).sum()
```

Out[37]:
```
2000-01-31    5.0
2000-02-29    7.0
2000-03-31    8.0
dtype: float64
```

### *ts.resample( freq )*

In [38]:
```
ts.resample('M').mean()
```

Out[38]:
```
2000-01-31    1.0
2000-02-29    1.0
2000-03-31    1.0
Freq: M, dtype: float64
```

# 时期 (period) 及其计算

### *pd.Period( value, freq )*：时期的创建

In [39]:
```
p = pd.Period(2021, freq='A-MAY')
p
```

Out[39]:
```
Period('2021', 'A-MAY')
```

In [40]:
```
p + 3
```

Out[40]:
```
Period('2024', 'A-MAY')
```

In [41]:
```
p - pd.Period(2019, freq='A-MAY')
```

Out[41]:
```
<2 * YearEnds: month=5>
```

## *pd.period_range( start, end, freq )*：时期范围的创建

In [42]:
```
rng = pd.period_range('2000-01-01', '2000-06-30', freq='M')
rng
```

Out[42]:
```
PeriodIndex(['2000-01', '2000-02', '2000-03', '2000-04', '2000-05', '2000-06'], dtype='period[M]')
```

In [43]:
```
pd.Series(np.arange(6), index=rng)
```

Out[43]:
```
2000-01    0
2000-02    1
2000-03    2
2000-04    3
2000-05    4
2000-06    5
Freq: M, dtype: int32
```

## *pd.PeriodIndex( values, freq, year, quarter )*：季度时期范围的创建

In [44]:
```
values = ['2001Q3', '2002Q2', '2003Q1']
pd.PeriodIndex(values, freq='Q-DEC')
```

Out[44]:
```
PeriodIndex(['2001Q3', '2002Q2', '2003Q1'], dtype='period[Q-DEC]')
```

通过数组创建 PeriodIndex

In [45]:
```
data = pd.read_csv('pydata-book-2nd-edition/examples/macrodata.csv')
data.year, data.quarter
```

Out[45]:
```
(0      1959.0
 1      1959.0
 2      1959.0
 3      1959.0
 4      1960.0
         ...
 198    2008.0
 199    2008.0
 200    2009.0
 201    2009.0
 202    2009.0
 Name: year, Length: 203, dtype: float64,
 0      1.0
 1      2.0
 2      3.0
 3      4.0
 4      1.0
       ...
 198    3.0
 199    4.0
 200    1.0
 201    2.0
 202    3.0
 Name: quarter, Length: 203, dtype: float64)
```

In [46]:
```
index = pd.PeriodIndex(year=data.year, quarter=data.quarter,
                       freq='Q-DEC')
index
```

Out[46]:
```
PeriodIndex(['1959Q1', '1959Q2', '1959Q3', '1959Q4', '1960Q1', '1960Q2',
             '1960Q3', '1960Q4', '1961Q1', '1961Q2',
             ...
             '2007Q2', '2007Q3', '2007Q4', '2008Q1', '2008Q2', '2008Q3',
             '2008Q4', '2009Q1', '2009Q2', '2009Q3'],
            dtype='period[Q-DEC]', length=203)
```

In [47]:
```
data.index = index
data
```

Out[47]:

| | year | quarter | realgdp | realcons | realinv | realgovt | realdpi | cpi | m1 | tbilrate | unemp | pop | infl | realint |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1959Q1** | 1959.0 | 1.0 | 2710.349 | 1707.4 | 286.898 | 470.045 | 1886.9 | 28.980 | 139.7 | 2.82 | 5.8 | 177.146 | 0.00 | 0.00 |
| **1959Q2** | 1959.0 | 2.0 | 2778.801 | 1733.7 | 310.859 | 481.301 | 1919.7 | 29.150 | 141.7 | 3.08 | 5.1 | 177.830 | 2.34 | 0.74 |
| **1959Q3** | 1959.0 | 3.0 | 2775.488 | 1751.8 | 289.226 | 491.260 | 1916.4 | 29.350 | 140.5 | 3.82 | 5.3 | 178.657 | 2.74 | 1.09 |
| **1959Q4** | 1959.0 | 4.0 | 2785.204 | 1753.7 | 299.356 | 484.052 | 1931.3 | 29.370 | 140.0 | 4.33 | 5.6 | 179.386 | 0.27 | 4.06 |
| **1960Q1** | 1960.0 | 1.0 | 2847.699 | 1770.5 | 331.722 | 462.199 | 1955.5 | 29.540 | 139.6 | 3.50 | 5.2 | 180.007 | 2.31 | 1.19 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **2008Q3** | 2008.0 | 3.0 | 13324.600 | 9267.7 | 1990.693 | 991.551 | 9838.3 | 216.889 | 1474.7 | 1.17 | 6.0 | 305.270 | -3.16 | 4.33 |
| **2008Q4** | 2008.0 | 4.0 | 13141.920 | 9195.3 | 1857.661 | 1007.273 | 9920.4 | 212.174 | 1576.5 | 0.12 | 6.9 | 305.952 | -8.79 | 8.91 |
| **2009Q1** | 2009.0 | 1.0 | 12925.410 | 9209.2 | 1558.494 | 996.287 | 9926.4 | 212.671 | 1592.8 | 0.22 | 8.1 | 306.547 | 0.94 | -0.71 |
| **2009Q2** | 2009.0 | 2.0 | 12901.504 | 9189.0 | 1456.678 | 1023.528 | 10077.5 | 214.469 | 1653.6 | 0.18 | 9.2 | 307.226 | 3.37 | -3.19 |
| **2009Q3** | 2009.0 | 3.0 | 12990.341 | 9256.0 | 1486.398 | 1044.088 | 10040.6 | 216.385 | 1673.9 | 0.12 | 9.6 | 308.013 | 3.56 | -3.44 |

203 rows × 14 columns

## *period.asfreq( freq, how )*：时期的频率转换

### 年 → 月

In [48]:
```
p = pd.Period('2021', freq='A-MAY')
p, p.asfreq('M', how='start'), p.asfreq('M', how='end')  # 2020.6 开始  # 2021.5 结束
```

Out[48]:
```
(Period('2021', 'A-MAY'), Period('2020-06', 'M'), Period('2021-05', 'M'))
```

### 月 → 年

In [49]:
```
p = pd.Period('May-2021', 'M')
p, p.asfreq('A-DEC')
```

Out[49]:
```
(Period('2021-05', 'M'), Period('2021', 'A-DEC'))
```

### 年 → 日

In [50]:
```
rng = pd.period_range('2006', '2009', freq='A-DEC')
ts = pd.Series(np.random.randn(len(rng)), index=rng)
ts, ts.asfreq('M', how='start'),  ts.asfreq('D', how='start')
```

Out[50]:
```
(2006   -0.954789
 2007    2.045184
 2008   -0.367045
 2009   -0.672529
 Freq: A-DEC, dtype: float64,
 2006-01   -0.954789
 2007-01    2.045184
 2008-01   -0.367045
 2009-01   -0.672529
 Freq: M, dtype: float64,
 2006-01-01   -0.954789
 2007-01-01    2.045184
 2008-01-01   -0.367045
 2009-01-01   -0.672529
 Freq: D, dtype: float64)
```

## 按季度计算的时期

In [51]:
```
p = pd.Period('2021Q1', freq='Q-DEC') # 季度以DEC为一年的分界，2021第一季度为2021.1-2021.3
p, p.asfreq('D', 'start'), p.asfreq('D', 'end')
```

Out[51]:
```
(Period('2021Q1', 'Q-DEC'),
 Period('2021-01-01', 'D'),
 Period('2021-03-31', 'D'))
```

In [52]:
```
p = pd.Period('2021Q1', freq='Q-JUN') # 季度以JUN为一年的分界，2021第一季度为2020.7-2020.9
p, p.asfreq('D', 'start'), p.asfreq('D', 'end')
```

Out[52]:
```
(Period('2021Q1', 'Q-JUN'),
 Period('2020-07-01', 'D'),
 Period('2020-09-30', 'D'))
```

In [53]:
```
p = pd.Period('2021Q4', freq='Q-JUN') # 季度以JUN为一年的分界，2021第四季度为2021.4-2021.6
p, p.asfreq('D', 'start'), p.asfreq('D', 'end')
```

Out[53]:
```
(Period('2021Q4', 'Q-JUN'),
 Period('2021-04-01', 'D'),
 Period('2021-06-30', 'D'))
```

获取该季度倒数第二个工作日下午4点的时间戳：

In [54]:
```
from pandas.tseries.offsets import Hour
p4pm = (p.asfreq('B', 'end') - 1).asfreq('T', 'start') + Hour(16)
p4pm, p4pm.to_timestamp()
```

Out[54]:
```
(Period('2021-06-29 16:00', 'T'), Timestamp('2021-06-29 16:00:00'))
```

季度型范围的算术运算

In [55]:
```
rng = pd.period_range('2019Q3', '2021Q4', freq='Q-DEC')
ts = pd.Series(np.arange(len(rng)), index=rng)
ts
```

Out[55]:
```
2019Q3    0
2019Q4    1
2020Q1    2
2020Q2    3
2020Q3    4
2020Q4    5
2021Q1    6
2021Q2    7
2021Q3    8
2021Q4    9
Freq: Q-DEC, dtype: int32
```

In [56]:
```
new_rng = rng.asfreq('D', 'end').asfreq('T', 'start') + 16 * 60
ts.index = new_rng.to_timestamp()
ts
```

Out[56]:
```
2019-09-30 16:00:00    0
2019-12-31 16:00:00    1
2020-03-31 16:00:00    2
2020-06-30 16:00:00    3
2020-09-30 16:00:00    4
2020-12-31 16:00:00    5
2021-03-31 16:00:00    6
2021-06-30 16:00:00    7
2021-09-30 16:00:00    8
2021-12-31 16:00:00    9
Freq: Q-DEC, dtype: int32
```

### *ts.to_period( freq, copy )* : 时间戳 → 时期

In [57]:
```
rng = pd.date_range('2021-01-01', periods=3, freq='M')
ts = pd.Series(np.random.randn(3), index=rng)
ts, ts.to_period()
```

Out[57]:
```
(2021-01-31    0.799601
 2021-02-28   -0.747543
 2021-03-31   -0.333042
 Freq: M, dtype: float64,
 2021-01    0.799601
 2021-02   -0.747543
 2021-03   -0.333042
 Freq: M, dtype: float64)
```

In [58]:
```
rng = pd.date_range('1/29/2021', periods=6, freq='D')
ts2 = pd.Series(np.random.randn(6), index=rng)
ts2, ts2.to_period('M')
```

Out[58]:
```
(2021-01-29   -0.917420
 2021-01-30   -1.667312
 2021-01-31    0.370422
 2021-02-01   -1.841920
 2021-02-02    0.067595
 2021-02-03    0.539212
 Freq: D, dtype: float64,
 2021-01   -0.917420
 2021-01   -1.667312
 2021-01    0.370422
 2021-02   -1.841920
 2021-02    0.067595
 2021-02    0.539212
 Freq: M, dtype: float64)
```

### *ts.to_timestamp( freq, how, copy )* : 时期 → 时间戳

In [59]:
```
rng = pd.period_range('1/29/2021', periods=2, freq='M')
ts3 = pd.Series(np.random.randn(2), index=rng)
ts3, ts3.to_timestamp()
```

Out[59]:
```
(2021-01   -2.029517
 2021-02   -0.154121
 Freq: M, dtype: float64,
 2021-01-01   -2.029517
 2021-02-01   -0.154121
 dtype: float64)
```

## 重采样和频率转换

### *ts.resample( freq, axis, closed, label, kind, fill_method, limit, convention )*

In [60]:
```
rng = pd.date_range('2000-01-01', periods=100, freq='D')
ts = pd.Series(np.ones(len(rng)), index=rng)
ts
```

Out[60]:
```
2000-01-01    1.0
2000-01-02    1.0
2000-01-03    1.0
2000-01-04    1.0
2000-01-05    1.0
             ...
2000-04-05    1.0
2000-04-06    1.0
2000-04-07    1.0
2000-04-08    1.0
2000-04-09    1.0
Freq: D, Length: 100, dtype: float64
```

In [61]:
```
ts.resample('M').sum()
```

Out[61]:
```
2000-01-31    31.0
2000-02-29    29.0
2000-03-31    31.0
2000-04-30     9.0
Freq: M, dtype: float64
```

***kind*** : 聚合到周期 ( 'period' ) 或 时间戳 ( 'timestamp' )

In [62]:
```
ts.resample('M', kind='period').sum()
```

Out[62]:
```
2000-01    31.0
2000-02    29.0
2000-03    31.0
2000-04     9.0
Freq: M, dtype: float64
```

***closed*** : 降采样中, 设置时间闭合的一端, 'right' (start, end] 或 'left' [start, end)

In [63]:
```
rng2 = pd.date_range('2000-01-01', periods=7, freq='T')
ts2 = pd.Series(np.arange(7), index=rng2)
ts2
```

Out[63]:
```
2000-01-01 00:00:00    0
2000-01-01 00:01:00    1
2000-01-01 00:02:00    2
2000-01-01 00:03:00    3
2000-01-01 00:04:00    4
2000-01-01 00:05:00    5
2000-01-01 00:06:00    6
Freq: T, dtype: int32
```

In [64]:
```
ts2.resample('5T', closed='right').sum() # ( , ]
```

Out[64]:
```
1999-12-31 23:55:00     0
2000-01-01 00:00:00    15
2000-01-01 00:05:00     6
Freq: 5T, dtype: int32
```

In [65]:
```
ts2.resample('5T', closed='left').sum() # [ , )
```

Out[65]:
```
2000-01-01 00:00:00    10
2000-01-01 00:05:00    11
Freq: 5T, dtype: int32
```

***label*** : 降采样中, 设置聚合值的标签, 'right' 或 'left'

In [66]:
```
ts2.resample('5T', closed='right', label='right').sum()
```

Out[66]:
```
2000-01-01 00:00:00     0
2000-01-01 00:05:00    15
2000-01-01 00:10:00     6
Freq: 5T, dtype: int32
```

***convention*** : 升采样中, 设置低频周期对应的高频时间戳, 'start' 或 'end'

In [67]:
```
frame = pd.DataFrame(np.random.randn(2, 4),
                     index=pd.date_range('1/1/2000', periods=2,
                                         freq='M'),
                     columns=['Colorado', 'Texas', 'New York', 'Ohio'])
frame
```

Out[67]:

| | Colorado | Texas | New York | Ohio |
|---|---|---|---|---|
| **2000-01-31** | 1.025930 | -0.974227 | 0.842722 | 1.017311 |
| **2000-02-29** | 1.171868 | -1.525096 | -0.159692 | 0.148648 |

In [68]:
```
df_month = frame.resample('M', kind='period').sum()
df_month
```

Out[68]:

| | Colorado | Texas | New York | Ohio |
|---|---|---|---|---|
| **2000-01** | 1.025930 | -0.974227 | 0.842722 | 1.017311 |
| **2000-02** | 1.171868 | -1.525096 | -0.159692 | 0.148648 |

In [69]:
```
df_daily = df_month.resample('D', convention='start').asfreq()
df_daily.head()
```

Out[69]:

| | Colorado | Texas | New York | Ohio |
|---|---|---|---|---|
| **2000-01-01** | 1.02593 | -0.974227 | 0.842722 | 1.017311 |
| **2000-01-02** | NaN | NaN | NaN | NaN |
| **2000-01-03** | NaN | NaN | NaN | NaN |
| **2000-01-04** | NaN | NaN | NaN | NaN |
| **2000-01-05** | NaN | NaN | NaN | NaN |