Time Series

In [1]:
```
import numpy as np
import pandas as pd
np.random.seed(12345)
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 6))
PREVIOUS_MAX_ROWS = pd.options.display.max_rows
pd.options.display.max_rows = 20
np.set_printoptions(precision=4, suppress=True)
```

Date and Time Data Types and Tools

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

Out[2]:
```
datetime.datetime(2024, 9, 9, 22, 5, 0, 303112)
```

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

Out[3]:
```
(2024, 9, 9)
```

In [4]:
```
delta = datetime(2011, 1, 7) - datetime(2008, 6, 24, 8, 15)
delta
```

Out[4]:
```
datetime.timedelta(days=926, seconds=56700)
```

In [5]:
```
delta.days
```

Out[5]:
```
926
```

In [6]:
```
delta.seconds
```

Out[6]:
```
56700
```

In [7]:
```
from datetime import timedelta
start = datetime(2011, 1, 7)
start + timedelta(12)
```

Out[7]:
```
datetime.datetime(2011, 1, 19, 0, 0)
```

In [8]:
```
start - 2 * timedelta(12)
```

Out[8]:
```
datetime.datetime(2010, 12, 14, 0, 0)
```

Converting Between String and Datetime

In [9]:
```
stamp = datetime(2011, 1, 3)
str(stamp)
```

Out[9]:
```
'2011-01-03 00:00:00'
```

In [10]:
```
stamp.strftime('%Y-%m-%d')
```

Out[10]:
```
'2011-01-03'
```

In [11]:
```
value = '2011-01-03'
datetime.strptime(value, '%Y-%m-%d')
```

Out[11]:
```
datetime.datetime(2011, 1, 3, 0, 0)
```

In [12]:
```
datestrs = ['7/6/2011', '8/6/2011']
[datetime.strptime(x, '%m/%d/%Y') for x in datestrs]
```

Out[12]:
```
[datetime.datetime(2011, 7, 6, 0, 0), datetime.datetime(2011, 8, 6, 0, 0)]
```

In [13]:
```
from dateutil.parser import parse
parse('2011-01-03')
```

Out[13]:
```
datetime.datetime(2011, 1, 3, 0, 0)
```

In [14]:
```
parse('Jan 31, 1997 10:45 PM')
```

Out[14]:
```
datetime.datetime(1997, 1, 31, 22, 45)
```

In [15]:
```
parse('6/12/2011', dayfirst=True)
```

Out[15]:
```
datetime.datetime(2011, 12, 6, 0, 0)
```

In [16]:
```
datestrs = ['2011-07-06 12:00:00', '2011-08-06 00:00:00']
pd.to_datetime(datestrs)
```

Out[16]:
```
DatetimeIndex(['2011-07-06 12:00:00', '2011-08-06 00:00:00'], dtype='datetime64[ns]', freq=None)
```

In [17]:
```
idx = pd.to_datetime(datestrs + [None])
idx
```

Out[17]:
```
DatetimeIndex(['2011-07-06 12:00:00', '2011-08-06 00:00:00', 'NaT'], dtype='datetime64[ns]', freq=None)
```

In [18]:
```
idx[2]
```

Out[18]:
```
NaT
```

In [19]:
```
pd.isnull(idx)
```

Out[19]:
```
array([False, False,  True])
```

Time Series Basics

In [20]:
```
from datetime import datetime
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[20]:
```
2011-01-02   -0.204708
2011-01-05    0.478943
2011-01-07   -0.519439
2011-01-08   -0.555730
2011-01-10    1.965781
2011-01-12    1.393406
dtype: float64
```

In [21]:
```
ts.index
```

Out[21]:
```
DatetimeIndex(['2011-01-02', '2011-01-05', '2011-01-07', '2011-01-08',
               '2011-01-10', '2011-01-12'],
              dtype='datetime64[ns]', freq=None)
```

In [22]:
```
ts + ts[::2]
```

Out[22]:
```
2011-01-02   -0.409415
2011-01-05         NaN
2011-01-07   -1.038877
2011-01-08         NaN
2011-01-10    3.931561
2011-01-12         NaN
dtype: float64
```

In [23]:
```
ts.index.dtype
```

Out[23]:
```
dtype('<M8[ns]')
```

In [24]:
```
stamp = ts.index[0]
stamp
```

Out[24]:
```
Timestamp('2011-01-02 00:00:00')
```

Indexing, Selection, Subsetting

In [25]:
```
stamp = ts.index[2]
ts[stamp]
```

Out[25]:
```
-0.5194387150567381
```

In [26]:
```
ts['1/10/2011']
```

Out[26]:
```
1.9657805725027142
```

In [27]:
```
ts['20110110']
```

Out[27]:
```
1.9657805725027142
```

In [28]:
```
longer_ts = pd.Series(np.random.randn(1000),
                      index=pd.date_range('1/1/2000', periods=1000))
longer_ts
```

Out[28]:
```
2000-01-01    0.092908
2000-01-02    0.281746
2000-01-03    0.769023
2000-01-04    1.246435
2000-01-05    1.007189
                ...
2002-09-22    0.930944
2002-09-23   -0.811676
2002-09-24   -1.830156
2002-09-25   -0.138730
2002-09-26    0.334088
Freq: D, Length: 1000, dtype: float64
```

In [29]:
```
longer_ts['2001']
```

Out[29]:
```
2001-01-01    1.599534
2001-01-02    0.474071
2001-01-03    0.151326
2001-01-04   -0.542173
2001-01-05   -0.475496
                ...
2001-12-27    0.057874
2001-12-28   -0.433739
2001-12-29    0.092698
2001-12-30   -1.397820
2001-12-31    1.457823
Freq: D, Length: 365, dtype: float64
```

In [30]:
```
longer_ts['2001-05']
```

Out[30]:
```
2001-05-01   -0.622547
2001-05-02    0.936289
2001-05-03    0.750018
2001-05-04   -0.056715
2001-05-05    2.300675
                ...
2001-05-27    0.235477
2001-05-28    0.111835
2001-05-29   -1.251504
2001-05-30   -2.949343
2001-05-31    0.634634
Freq: D, Length: 31, dtype: float64
```

In [31]:
```
ts[datetime(2011, 1, 7):]
```

Out[31]:
```
2011-01-07   -0.519439
2011-01-08   -0.555730
2011-01-10    1.965781
2011-01-12    1.393406
dtype: float64
```

In [32]:
```
ts
```

Out[32]:
```
2011-01-02   -0.204708
2011-01-05    0.478943
2011-01-07   -0.519439
2011-01-08   -0.555730
2011-01-10    1.965781
2011-01-12    1.393406
dtype: float64
```

In [33]:
```
ts['1/6/2011':'1/11/2011']
```

Out[33]:
```
2011-01-07   -0.519439
2011-01-08   -0.555730
2011-01-10    1.965781
dtype: float64
```

In [34]:
```
ts.truncate(after='1/9/2011')
```

Out[34]:
```
2011-01-02   -0.204708
2011-01-05    0.478943
2011-01-07   -0.519439
2011-01-08   -0.555730
dtype: float64
```

In [35]:
```
dates = pd.date_range('1/1/2000', periods=100, freq='W-WED')
long_df = pd.DataFrame(np.random.randn(100, 4),
                       index=dates,
                       columns=['Colorado', 'Texas',
                                'New York', 'Ohio'])
long_df.loc['5-2001']
```

Out[35]:

	Colorado	Texas	New York	Ohio
2001-05-02	-0.006045	0.490094	-0.277186	-0.707213
2001-05-09	-0.560107	2.735527	0.927335	1.513906
2001-05-16	0.538600	1.273768	0.667876	-0.969206
2001-05-23	1.676091	-0.817649	0.050188	1.951312
2001-05-30	3.260383	0.963301	1.201206	-1.852001

Time Series with Duplicate Indices

In [36]:
```
dates = pd.DatetimeIndex(['1/1/2000', '1/2/2000', '1/2/2000',
                          '1/2/2000', '1/3/2000'])
dup_ts = pd.Series(np.arange(5), index=dates)
dup_ts
```

Out[36]:
```
2000-01-01    0
2000-01-02    1
2000-01-02    2
2000-01-02    3
2000-01-03    4
dtype: int32
```

In [37]:
```
dup_ts.index.is_unique
```

Out[37]:
```
False
```

In [38]:
```
dup_ts['1/3/2000']  # not duplicated
```

Out[38]:
```
4
```

In [39]:
```
dup_ts['1/2/2000']  # duplicated
```

Out[39]:
```
2000-01-02    1
2000-01-02    2
2000-01-02    3
dtype: int32
```

In [40]:
```
grouped = dup_ts.groupby(level=0)
grouped.mean()
```

Out[40]:
```
2000-01-01    0.0
2000-01-02    2.0
2000-01-03    4.0
dtype: float64
```

In [41]:
```
grouped.count()
```

Out[41]:
```
2000-01-01    1
2000-01-02    3
2000-01-03    1
dtype: int64
```

Date Ranges, Frequencies, and Shifting

In [42]:
```
ts
```

Out[42]:
```
2011-01-02   -0.204708
2011-01-05    0.478943
2011-01-07   -0.519439
2011-01-08   -0.555730
2011-01-10    1.965781
2011-01-12    1.393406
dtype: float64
```

In [43]:
```
resampler = ts.resample('D')
```

In [44]:
```
resampler
```

Out[44]:
```
<pandas.core.resample.DatetimeIndexResampler object at 0x000002512678B4100>
```

Generating Date Ranges

In [45]:
```
index = pd.date_range('2012-04-01', '2012-06-01')
index
```

Out[45]:
```
DatetimeIndex(['2012-04-01', '2012-04-02', '2012-04-03', '2012-04-04',
               '2012-04-05', '2012-04-06', '2012-04-07', '2012-04-08',
               '2012-04-09', '2012-04-10', '2012-04-11', '2012-04-12',
               '2012-04-13', '2012-04-14', '2012-04-15', '2012-04-16',
               '2012-04-17', '2012-04-18', '2012-04-19', '2012-04-20',
               '2012-04-21', '2012-04-22', '2012-04-23', '2012-04-24',
               '2012-04-25', '2012-04-26', '2012-04-27', '2012-04-28',
               '2012-04-29', '2012-04-30', '2012-05-01', '2012-05-02',
               '2012-05-03', '2012-05-04', '2012-05-05', '2012-05-06',
               '2012-05-07', '2012-05-08', '2012-05-09', '2012-05-10',
               '2012-05-11', '2012-05-12', '2012-05-13', '2012-05-14',
               '2012-05-15', '2012-05-16', '2012-05-17', '2012-05-18',
               '2012-05-19', '2012-05-20', '2012-05-21', '2012-05-22',
               '2012-05-23', '2012-05-24', '2012-05-25', '2012-05-26',
               '2012-05-27', '2012-05-28', '2012-05-29', '2012-05-30',
               '2012-05-31', '2012-06-01'],
              dtype='datetime64[ns]', freq='D')
```

In [46]:
```
pd.date_range(start='2012-04-01', periods=20)
```

Out[46]:
```
DatetimeIndex(['2012-04-01', '2012-04-02', '2012-04-03', '2012-04-04',
               '2012-04-05', '2012-04-06', '2012-04-07', '2012-04-08',
               '2012-04-09', '2012-04-10', '2012-04-11', '2012-04-12',
               '2012-04-13', '2012-04-14', '2012-04-15', '2012-04-16',
               '2012-04-17', '2012-04-18', '2012-04-19', '2012-04-20'],
              dtype='datetime64[ns]', freq='D')
```

In [47]:
```
pd.date_range(end='2012-06-01', periods=20)
```

Out[47]:
```
DatetimeIndex(['2012-05-13', '2012-05-14', '2012-05-15', '2012-05-16',
               '2012-05-17', '2012-05-18', '2012-05-19', '2012-05-20',
               '2012-05-21', '2012-05-22', '2012-05-23', '2012-05-24',
               '2012-05-25', '2012-05-26', '2012-05-27', '2012-05-28',
               '2012-05-29', '2012-05-30', '2012-05-31', '2012-06-01'],
              dtype='datetime64[ns]', freq='D')
```

In [48]:
```
pd.date_range('2000-01-01', '2000-12-01', freq='BM')
```

Out[48]:
```
DatetimeIndex(['2000-01-31', '2000-02-29', '2000-03-31', '2000-04-28',
               '2000-05-31', '2000-06-30', '2000-07-31', '2000-08-31',
               '2000-09-29', '2000-10-31', '2000-11-30'],
              dtype='datetime64[ns]', freq='BM')
```

In [49]:
```
pd.date_range('2012-05-02 12:56:31', periods=5)
```

Out[49]:
```
DatetimeIndex(['2012-05-02 12:56:31', '2012-05-03 12:56:31',
               '2012-05-04 12:56:31', '2012-05-05 12:56:31',
               '2012-05-06 12:56:31'],
              dtype='datetime64[ns]', freq='D')
```

In [50]:
```
pd.date_range('2012-05-02 12:56:31', periods=5, normalize=True)
```

Out[50]:
```
DatetimeIndex(['2012-05-02', '2012-05-03', '2012-05-04', '2012-05-05',
               '2012-05-06'],
              dtype='datetime64[ns]', freq='D')
```

Frequencies and Date Offsets

In [51]:
```
from pandas.tseries.offsets import Hour, Minute
hour = Hour()
hour
```

Out[51]:
```
<Hour>
```

In [52]:
```
four_hours = Hour(4)
four_hours
```

Out[52]:
```
<4 * Hours>
```

In [53]:
```
pd.date_range('2000-01-01', '2000-01-03 23:59', freq='4h')
```

Out[53]:
```
DatetimeIndex(['2000-01-01 00:00:00', '2000-01-01 04:00:00',
               '2000-01-01 08:00:00', '2000-01-01 12:00:00',
               '2000-01-01 16:00:00', '2000-01-01 20:00:00',
               '2000-01-02 00:00:00', '2000-01-02 04:00:00',
               '2000-01-02 08:00:00', '2000-01-02 12:00:00',
               '2000-01-02 16:00:00', '2000-01-02 20:00:00',
               '2000-01-03 00:00:00', '2000-01-03 04:00:00',
               '2000-01-03 08:00:00', '2000-01-03 12:00:00',
               '2000-01-03 16:00:00', '2000-01-03 20:00:00'],
              dtype='datetime64[ns]', freq='4H')
```

In [54]:
```
Hour(2) + Minute(30)
```

Out[54]:
```
<150 * Minutes>
```

In [55]:
```
pd.date_range('2000-01-01', periods=10, freq='1h30min')
```

Out[55]:
```
DatetimeIndex(['2000-01-01 00:00:00', '2000-01-01 01:30:00',
               '2000-01-01 03:00:00', '2000-01-01 04:30:00',
               '2000-01-01 06:00:00', '2000-01-01 07:30:00',
               '2000-01-01 09:00:00', '2000-01-01 10:30:00',
               '2000-01-01 12:00:00', '2000-01-01 13:30:00'],
              dtype='datetime64[ns]', freq='90T')
```

Week of month dates

In [56]:
```
rng = pd.date_range('2012-01-01', '2012-09-01', freq='WOM-3FRI')
list(rng)
```

Out[56]:
```
[Timestamp('2012-01-20 00:00:00', freq='WOM-3FRI'),
 Timestamp('2012-02-17 00:00:00', freq='WOM-3FRI'),
 Timestamp('2012-03-16 00:00:00', freq='WOM-3FRI'),
 Timestamp('2012-04-20 00:00:00', freq='WOM-3FRI'),
 Timestamp('2012-05-18 00:00:00', freq='WOM-3FRI'),
 Timestamp('2012-06-15 00:00:00', freq='WOM-3FRI'),
 Timestamp('2012-07-20 00:00:00', freq='WOM-3FRI'),
 Timestamp('2012-08-17 00:00:00', freq='WOM-3FRI')]
```

In [57]:
```
rng = pd.date_range('2012-01-01', '2012-09-01', freq='WOM-2SAT')
list(rng)
```

Out[57]:
```
[Timestamp('2012-01-14 00:00:00', freq='WOM-2SAT'),
 Timestamp('2012-02-11 00:00:00', freq='WOM-2SAT'),
 Timestamp('2012-03-10 00:00:00', freq='WOM-2SAT'),
 Timestamp('2012-04-14 00:00:00', freq='WOM-2SAT'),
 Timestamp('2012-05-12 00:00:00', freq='WOM-2SAT'),
 Timestamp('2012-06-09 00:00:00', freq='WOM-2SAT'),
 Timestamp('2012-07-14 00:00:00', freq='WOM-2SAT'),
 Timestamp('2012-08-11 00:00:00', freq='WOM-2SAT')]
```

Shifting (Leading and Lagging) Data

In [58]:
```
ts = pd.Series(np.random.randn(4),
               index=pd.date_range('1/1/2000', periods=4, freq='M'))
ts
```

Out[58]:
```
2000-01-31   -0.066748
2000-02-29    0.838639
2000-03-31   -0.117388
2000-04-30   -0.517795
Freq: M, dtype: float64
```

In [59]:
```
ts.shift(2)
```

Out[59]:
```
2000-01-31         NaN
2000-02-29         NaN
2000-03-31   -0.066748
2000-04-30    0.838639
Freq: M, dtype: float64
```

In [60]:
```
ts.shift(-2)
```

Out[60]:
```
2000-01-31   -0.117388
2000-02-29   -0.517795
2000-03-31         NaN
2000-04-30         NaN
Freq: M, dtype: float64
```

ts / ts.shift(1) - 1

In [61]:
```
ts.shift(2, freq='M')
```

Out[61]:
```
2000-03-31   -0.066748
2000-04-30    0.838639
2000-05-31   -0.117388
2000-06-30   -0.517795
Freq: M, dtype: float64
```

In [62]:
```
ts.shift(3, freq='D')
```

Out[62]:
```
2000-02-03   -0.066748
2000-03-03    0.838639
2000-04-03   -0.117388
2000-05-03   -0.517795
dtype: float64
```

In [63]:
```
ts.shift(1, freq='90T')
```

Out[63]:
```
2000-01-31 01:30:00   -0.066748
2000-02-29 01:30:00    0.838639
2000-03-31 01:30:00   -0.117388
2000-04-30 01:30:00   -0.517795
dtype: float64
```

Shifting dates with offsets

In [64]:
```
from pandas.tseries.offsets import Day, MonthEnd
now = datetime(2011, 11, 17)
now + 3 * Day()
```

Out[64]:
```
Timestamp('2011-11-20 00:00:00')
```

In [65]:
```
now + MonthEnd()
```

Out[65]:
```
Timestamp('2011-11-30 00:00:00')
```

In [66]:
```
now + MonthEnd(2)
```

Out[66]:
```
Timestamp('2011-12-31 00:00:00')
```

In [67]:
```
offset = MonthEnd()
offset.rollforward(now)
```

Out[67]:
```
Timestamp('2011-11-30 00:00:00')
```

In [68]:
```
offset.rollback(now)
```

Out[68]:
```
Timestamp('2011-10-31 00:00:00')
```

In [69]:
```
ts = pd.Series(np.random.randn(20),
               index=pd.date_range('1/15/2000', periods=20, freq='4d'))
ts
```

Out[69]:
```
2000-01-15   -0.116696
2000-01-19    2.389645
2000-01-23   -0.932454
2000-01-27   -0.229331
2000-01-31   -1.140330
2000-02-04    0.439920
2000-02-08   -0.823758
2000-02-12   -0.520930
2000-02-16    0.350282
2000-02-20    0.204395
2000-02-24    0.133445
2000-02-28    0.327905
2000-03-03    0.072153
2000-03-07    0.131678
2000-03-11   -1.297459
2000-03-15    0.997747
2000-03-19    0.870955
2000-03-23   -0.991253
2000-03-27    0.151699
2000-03-31    1.266151
Freq: 4D, dtype: float64
```

In [70]:
```
ts.groupby(offset.rollforward).mean()
```

Out[70]:
```
2000-01-31   -0.005833
2000-02-29    0.015894
2000-03-31    0.150209
dtype: float64
```

In [71]:
```
ts.resample('M').mean()
```

Out[71]:
```
2000-01-31   -0.005833
2000-02-29    0.015894
2000-03-31    0.150209
Freq: M, dtype: float64
```

Time Zone Handling

In [72]:
```
import pytz
pytz.common_timezones[-5:]
```

Out[72]:
```
['US/Eastern', 'US/Hawaii', 'US/Mountain', 'US/Pacific', 'UTC']
```

In [75]:
```
pytz.common_timezones[:20]
```

Out[75]:
```
['Africa/Abidjan',
 'Africa/Accra',
 'Africa/Addis_Ababa',
 'Africa/Algiers',
 'Africa/Asmara',
 'Africa/Bamako',
 'Africa/Bangui',
 'Africa/Banjul',
 'Africa/Bissau',
 'Africa/Blantyre',
 'Africa/Brazzaville',
 'Africa/Bujumbura',
 'Africa/Cairo',
 'Africa/Casablanca',
 'Africa/Ceuta',
 'Africa/Conakry',
 'Africa/Dakar',
 'Africa/Dar_es_Salaam',
 'Africa/Djibouti',
 'Africa/Douala']
```

In [74]:
```
tz = pytz.timezone('America/New_York')
tz
```

Out[74]:
```
<DstTzInfo 'America/New_York' LMT-1 day, 19:04:00 STD>
```

Time Zone Localization and Conversion

In [76]:
```
rng = pd.date_range('3/9/2012 9:30', periods=6, freq='D')
ts = pd.Series(np.random.randn(len(rng)), index=rng)
ts
```

Out[76]:
```
2012-03-09 09:30:00   -0.202469
2012-03-10 09:30:00    0.050718
2012-03-11 09:30:00    0.639869
2012-03-12 09:30:00    0.597594
2012-03-13 09:30:00   -0.797246
2012-03-14 09:30:00    0.472879
Freq: D, dtype: float64
```

In [77]:
```
print(ts.index.tz)
```

```
None
```

In [78]:
```
pd.date_range('3/9/2012 9:30', periods=10, freq='D', tz='UTC')
```

Out[78]:
```
DatetimeIndex(['2012-03-09 09:30:00+00:00', '2012-03-10 09:30:00+00:00',
               '2012-03-11 09:30:00+00:00', '2012-03-12 09:30:00+00:00',
               '2012-03-13 09:30:00+00:00', '2012-03-14 09:30:00+00:00',
               '2012-03-15 09:30:00+00:00', '2012-03-16 09:30:00+00:00',
               '2012-03-17 09:30:00+00:00', '2012-03-18 09:30:00+00:00'],
              dtype='datetime64[ns, UTC]', freq='D')
```

In [79]:
```
ts
```

Out[79]:
```
2012-03-09 09:30:00   -0.202469
2012-03-10 09:30:00    0.050718
2012-03-11 09:30:00    0.639869
2012-03-12 09:30:00    0.597594
2012-03-13 09:30:00   -0.797246
2012-03-14 09:30:00    0.472879
Freq: D, dtype: float64
```

In [80]:
```
ts_utc = ts.tz_localize('UTC')
ts_utc
```

Out[80]:
```
2012-03-09 09:30:00+00:00   -0.202469
2012-03-10 09:30:00+00:00    0.050718
2012-03-11 09:30:00+00:00    0.639869
2012-03-12 09:30:00+00:00    0.597594
2012-03-13 09:30:00+00:00   -0.797246
2012-03-14 09:30:00+00:00    0.472879
Freq: D, dtype: float64
```

In [81]:
```
ts_utc.index
```

Out[81]:
```
DatetimeIndex(['2012-03-09 09:30:00+00:00', '2012-03-10 09:30:00+00:00',
               '2012-03-11 09:30:00+00:00', '2012-03-12 09:30:00+00:00',
               '2012-03-13 09:30:00+00:00', '2012-03-14 09:30:00+00:00'],
              dtype='datetime64[ns, UTC]', freq='D')
```

In [82]:
```
ts_utc.tz_convert('America/New_York')
```

Out[82]:
```
2012-03-09 04:30:00-05:00   -0.202469
2012-03-10 04:30:00-05:00    0.050718
2012-03-11 05:30:00-04:00    0.639869
2012-03-12 05:30:00-04:00    0.597594
2012-03-13 05:30:00-04:00   -0.797246
2012-03-14 05:30:00-04:00    0.472879
Freq: D, dtype: float64
```

In [84]:
```
ts_eastern = ts.tz_localize('America/New_York')
ts_eastern.tz_convert('UTC')
```

Out[84]:
```
2012-03-09 14:30:00+00:00   -0.202469
2012-03-10 14:30:00+00:00    0.050718
2012-03-11 13:30:00+00:00    0.639869
2012-03-12 13:30:00+00:00    0.597594
2012-03-13 13:30:00+00:00   -0.797246
2012-03-14 13:30:00+00:00    0.472879
dtype: float64
```

In [85]:
```
ts_eastern.tz_convert('Europe/Berlin')
```

Out[85]:
```
2012-03-09 15:30:00+01:00   -0.202469
2012-03-10 15:30:00+01:00    0.050718
2012-03-11 14:30:00+01:00    0.639869
2012-03-12 14:30:00+01:00    0.597594
2012-03-13 14:30:00+01:00   -0.797246
2012-03-14 14:30:00+01:00    0.472879
dtype: float64
```

In [86]:
```
ts.index.tz_localize('Asia/Shanghai')
```

Out[86]:
```
DatetimeIndex(['2012-03-09 09:30:00+08:00', '2012-03-10 09:30:00+08:00',
               '2012-03-11 09:30:00+08:00', '2012-03-12 09:30:00+08:00',
               '2012-03-13 09:30:00+08:00', '2012-03-14 09:30:00+08:00'],
              dtype='datetime64[ns, Asia/Shanghai]', freq=None)
```

Operations with Time Zone–Aware Timestamp Objects

In [88]:
```
stamp = pd.Timestamp('2011-03-12 04:00')
stamp_utc = stamp.tz_localize('utc')
stamp_utc.tz_convert('America/New_York')
```

Out[88]:
```
Timestamp('2011-03-11 23:00:00-0500', tz='America/New_York')
```

In [89]:
```
stamp_moscow = pd.Timestamp('2011-03-12 04:00', tz='Europe/Moscow')
stamp_moscow
```

Out[89]:
```
Timestamp('2011-03-12 04:00:00+0300', tz='Europe/Moscow')
```

In [90]:
```
stamp_utc.value
```

Out[90]:
```
1299902400000000000
```

In [91]:
```
stamp_utc.tz_convert('America/New_York').value
```

Out[91]:
```
1299902400000000000
```

In [92]:
```
from pandas.tseries.offsets import Hour
stamp = pd.Timestamp('2012-03-12 01:30', tz='US/Eastern')
stamp
```

Out[92]:
```
Timestamp('2012-03-12 01:30:00-0400', tz='US/Eastern')
```

In [93]:
```
stamp + Hour()
```

Out[93]:
```
Timestamp('2012-03-12 02:30:00-0400', tz='US/Eastern')
```

In [94]:
```
stamp = pd.Timestamp('2012-11-04 00:30', tz='US/Eastern')
stamp
```

Out[94]:
```
Timestamp('2012-11-04 00:30:00-0400', tz='US/Eastern')
```

In [95]:
```
stamp + 2 * Hour()
```

Out[95]:
```
Timestamp('2012-11-04 01:30:00-0500', tz='US/Eastern')
```

Operations Between Different Time Zones

In [96]:
```
rng = pd.date_range('3/7/2012 9:30', periods=10, freq='B')
ts = pd.Series(np.random.randn(len(rng)), index=rng)
ts
```

Out[96]:
```
2012-03-07 09:30:00    0.522356
2012-03-08 09:30:00   -0.546348
2012-03-09 09:30:00   -0.733537
2012-03-12 09:30:00    1.302736
2012-03-13 09:30:00    0.022199
2012-03-14 09:30:00    0.364287
2012-03-15 09:30:00   -0.922839
2012-03-16 09:30:00    0.312656
2012-03-19 09:30:00   -1.128497
2012-03-20 09:30:00   -0.333488
Freq: B, dtype: float64
```

In [98]:
```
ts1 = ts[:7].tz_localize('Europe/London')
ts2 = ts1[2:].tz_convert('Europe/Moscow')
result = ts1 + ts2
result.index
```

Out[98]:
```
DatetimeIndex(['2012-03-07 09:30:00+00:00', '2012-03-08 09:30:00+00:00',
               '2012-03-09 09:30:00+00:00', '2012-03-12 09:30:00+00:00',
               '2012-03-13 09:30:00+00:00', '2012-03-14 09:30:00+00:00',
               '2012-03-15 09:30:00+00:00'],
              dtype='datetime64[ns, UTC]', freq=None)
```

Periods and Period Arithmetic

In [99]:
```
p = pd.Period(2007, freq='A-DEC')
p
```

Out[99]:
```
Period('2007', 'A-DEC')
```

In [100…
```
p + 5
```

Out[100]:
```
Period('2012', 'A-DEC')
```

In [101…
```
p - 2
```

Out[101]:
```
Period('2005', 'A-DEC')
```

In [102…
```
pd.Period('2014', freq='A-DEC') - p
```

Out[102]:
```
<7 * YearEnds: month=12>
```

In [103…
```
rng = pd.period_range('2000-01-01', '2000-06-30', freq='M')
rng
```

Out[103]:
```
PeriodIndex(['2000-01', '2000-02', '2000-03', '2000-04', '2000-05', '2000-06'], dt
ype='period[M]')
```

In [104…
```
pd.Series(np.random.randn(6), index=rng)
```

Out[104]:
```
2000-01   -0.514551
2000-02   -0.559782
2000-03   -0.783408
2000-04   -1.797685
2000-05   -0.172670
2000-06    0.680215
Freq: M, dtype: float64
```

In [105…
```
values = ['2001Q3', '2002Q2', '2003Q1']
index = pd.PeriodIndex(values, freq='Q-DEC')
index
```

Out[105]:
```
PeriodIndex(['2001Q3', '2002Q2', '2003Q1'], dtype='period[Q-DEC]')
```

Period Frequency Conversion

In [106…
```
p = pd.Period('2007', freq='A-DEC')
p
```

Out[106]:
```
Period('2007', 'A-DEC')
```

In [107…
```
p.asfreq('M', how='start')
```

Out[107]:
```
Period('2007-01', 'M')
```

In [108…
```
p.asfreq('M', how='end')
```
Out[108]:
```
Period('2007-12', 'M')
```

In [109…
```
p = pd.Period('2007', freq='A-JUN')
p
```
Out[109]:
```
Period('2007', 'A-JUN')
```

In [110…
```
p.asfreq('M', 'start')
```
Out[110]:
```
Period('2006-07', 'M')
```

In [111…
```
p.asfreq('M', 'end')
```
Out[111]:
```
Period('2007-06', 'M')
```

In [112…
```
p = pd.Period('Aug-2007', 'M')
p.asfreq('A-JUN')
```
Out[112]:
```
Period('2008', 'A-JUN')
```

In [113…
```
rng = pd.period_range('2006', '2009', freq='A-DEC')
ts = pd.Series(np.random.randn(len(rng)), index=rng)
ts
```
Out[113]:
```
2006    1.607578
2007    0.200381
2008   -0.834068
2009   -0.302988
Freq: A-DEC, dtype: float64
```

In [114…
```
ts.asfreq('M', how='start')
```
Out[114]:
```
2006-01    1.607578
2007-01    0.200381
2008-01   -0.834068
2009-01   -0.302988
Freq: M, dtype: float64
```

In [115…
```
ts.asfreq('B', how='end')
```
Out[115]:
```
2006-12-29    1.607578
2007-12-31    0.200381
2008-12-31   -0.834068
2009-12-31   -0.302988
Freq: B, dtype: float64
```

In [117…
```
ts.asfreq('D', how='end')
```
Out[117]:
```
2006-12-31    1.607578
2007-12-31    0.200381
2008-12-31   -0.834068
2009-12-31   -0.302988
Freq: D, dtype: float64
```

Quarterly Period Frequencies

In [118…
```
p = pd.Period('2012Q4', freq='Q-JAN')
p
```

Out[118]:
```
Period('2012Q4', 'Q-JAN')
```

In [119…
```
p.asfreq('D', 'start')
```

Out[119]:
```
Period('2011-11-01', 'D')
```

In [120…
```
p.asfreq('D', 'end')
```

Out[120]:
```
Period('2012-01-31', 'D')
```

In [121…
```
p4pm = (p.asfreq('B', 'e') - 1).asfreq('T', 's') + 16 * 60
p4pm
```

Out[121]:
```
Period('2012-01-30 16:00', 'T')
```

In [122…
```
p4pm.to_timestamp()
```

Out[122]:
```
Timestamp('2012-01-30 16:00:00')
```

In [123…
```
rng = pd.period_range('2011Q3', '2012Q4', freq='Q-JAN')
ts = pd.Series(np.arange(len(rng)), index=rng)
ts
```

Out[123]:
```
2011Q3    0
2011Q4    1
2012Q1    2
2012Q2    3
2012Q3    4
2012Q4    5
Freq: Q-JAN, dtype: int32
```

In [124…
```
new_rng = (rng.asfreq('B', 'e') - 1).asfreq('T', 's') + 16 * 60
ts.index = new_rng.to_timestamp()
ts
```

Out[124]:
```
2010-10-28 16:00:00    0
2011-01-28 16:00:00    1
2011-04-28 16:00:00    2
2011-07-28 16:00:00    3
2011-10-28 16:00:00    4
2012-01-30 16:00:00    5
dtype: int32
```

Converting Timestamps to Periods (and Back)

In [125…
```
rng = pd.date_range('2000-01-01', periods=3, freq='M')
ts = pd.Series(np.random.randn(3), index=rng)
ts
```

Out[125]:
```
2000-01-31    1.663261
2000-02-29   -0.996206
2000-03-31    1.521760
Freq: M, dtype: float64
```

In [126…
```
pts = ts.to_period()
pts
```

Out[126]:
```
2000-01    1.663261
2000-02   -0.996206
2000-03    1.521760
Freq: M, dtype: float64
```

In [127…
```
rng = pd.date_range('1/29/2000', periods=6, freq='D')
ts2 = pd.Series(np.random.randn(6), index=rng)
ts2
```

Out[127]:
```
2000-01-29    0.244175
2000-01-30    0.423331
2000-01-31   -0.654040
2000-02-01    2.089154
2000-02-02   -0.060220
2000-02-03   -0.167933
Freq: D, dtype: float64
```

In [128…
```
ts2.to_period('M')
```

Out[128]:
```
2000-01    0.244175
2000-01    0.423331
2000-01   -0.654040
2000-02    2.089154
2000-02   -0.060220
2000-02   -0.167933
Freq: M, dtype: float64
```

In [129…
```
pts = ts2.to_period()
pts
```

Out[129]:
```
2000-01-29    0.244175
2000-01-30    0.423331
2000-01-31   -0.654040
2000-02-01    2.089154
2000-02-02   -0.060220
2000-02-03   -0.167933
Freq: D, dtype: float64
```

In [130…
```
pts.to_timestamp(how='end')
```

Out[130]:
```
2000-01-29 23:59:59.999999999    0.244175
2000-01-30 23:59:59.999999999    0.423331
2000-01-31 23:59:59.999999999   -0.654040
2000-02-01 23:59:59.999999999    2.089154
2000-02-02 23:59:59.999999999   -0.060220
2000-02-03 23:59:59.999999999   -0.167933
Freq: D, dtype: float64
```

In [133…
```
pts.to_timestamp(how='start')
```

Out[133]:
```
2000-01-29    0.244175
2000-01-30    0.423331
2000-01-31   -0.654040
2000-02-01    2.089154
2000-02-02   -0.060220
2000-02-03   -0.167933
Freq: D, dtype: float64
```

Creating a PeriodIndex from Arrays

In [134…
```
data = pd.read_csv('examples/macrodata.csv')
data.head(5)
```

Out[134]:

	year	quarter	realgdp	realcons	realinv	realgovt	realdpi	cpi	m1	tbilrate	unemp
0	1959.0	1.0	2710.349	1707.4	286.898	470.045	1886.9	28.98	139.7	2.82	5.8
1	1959.0	2.0	2778.801	1733.7	310.859	481.301	1919.7	29.15	141.7	3.08	5.1
2	1959.0	3.0	2775.488	1751.8	289.226	491.260	1916.4	29.35	140.5	3.82	5.3
3	1959.0	4.0	2785.204	1753.7	299.356	484.052	1931.3	29.37	140.0	4.33	5.6
4	1960.0	1.0	2847.699	1770.5	331.722	462.199	1955.5	29.54	139.6	3.50	5.2

In [135…
```
data.year
```

Out[135]:
```
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
```

In [136…
```
data.quarter
```

Out[136]:
```
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 [137…
```
index = pd.PeriodIndex(year=data.year, quarter=data.quarter,
                       freq='Q-DEC')
index
```

Out[137]:
```
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 [138…
```
data.index = index
data.infl
```

Out[138]:
```
1959Q1    0.00
1959Q2    2.34
1959Q3    2.74
1959Q4    0.27
1960Q1    2.31
          ...
2008Q3   -3.16
2008Q4   -8.79
2009Q1    0.94
2009Q2    3.37
2009Q3    3.56
Freq: Q-DEC, Name: infl, Length: 203, dtype: float64
```

Resampling and Frequency Conversion

In [139…
```
rng = pd.date_range('2000-01-01', periods=100, freq='D')
ts = pd.Series(np.random.randn(len(rng)), index=rng)
ts
```

Out[139]:
```
2000-01-01    0.631634
2000-01-02   -1.594313
2000-01-03   -1.519937
2000-01-04    1.108752
2000-01-05    1.255853
                ...
2000-04-05   -0.423776
2000-04-06    0.789740
2000-04-07    0.937568
2000-04-08   -2.253294
2000-04-09   -1.772919
Freq: D, Length: 100, dtype: float64
```

In [140…
```
ts.resample('M').mean()
```

Out[140]:
```
2000-01-31   -0.165893
2000-02-29    0.078606
2000-03-31    0.223811
2000-04-30   -0.063643
Freq: M, dtype: float64
```

In [141…
```
ts.resample('M', kind='period').mean()
```

Out[141]:
```
2000-01   -0.165893
2000-02    0.078606
2000-03    0.223811
2000-04   -0.063643
Freq: M, dtype: float64
```

In [143…
```
ts_resampled = ts.resample(
    rule='W',                # Reamostragem semanal (W = Week)
    closed='right',          # Fechar o intervalo à direita (inclui o último dia do
    label='right',           # Rotular o intervalo no último dia da semana
    loffset=pd.Timedelta('1D'),  # Deslocar o rótulo em 1 dia para frente
    kind='timestamp',        # O resultado será um timestamp (opção padrão para Ser
    convention='start',      # Se estivermos lidando com períodos, começa no inícic
    base=0,                  # Definir o início da semana sem offset
    origin='epoch',          # A origem para o agrupamento será a época (1970-01-01
    # offset=pd.DateOffset(days=1)  # Adicionar um deslocamento de 1 dia ao início
).sum()                      # Soma os valores de cada intervalo semanal

# Exibir o resultado
print(ts_resampled)
```

```
2000-01-03   -0.962679
2000-01-10   -3.192873
2000-01-17    2.230693
2000-01-24    1.205071
2000-01-31   -4.785641
2000-02-07    2.602033
2000-02-14    4.574027
2000-02-21   -3.981263
2000-02-28   -0.814002
2000-03-06   -0.262772
2000-03-13    2.164324
2000-03-20    0.976878
2000-03-27    3.139698
2000-04-03    0.943960
2000-04-10   -0.335210
dtype: float64
```

```
C:\Users\Usuário\AppData\Local\Temp\ipykernel_9644\1592705933.py:1: FutureWarning:
'base' in .resample() and in Grouper() is deprecated.
The new arguments that you should use are 'offset' or 'origin'.

>>> df.resample(freq="3s", base=2)

becomes:

>>> df.resample(freq="3s", offset="2s")

  ts_resampled = ts.resample(
C:\Users\Usuário\AppData\Local\Temp\ipykernel_9644\1592705933.py:1: FutureWarning:
'loffset' in .resample() and in Grouper() is deprecated.

>>> df.resample(freq="3s", loffset="8H")

becomes:

>>> from pandas.tseries.frequencies import to_offset
>>> df = df.resample(freq="3s").mean()
>>> df.index = df.index.to_timestamp() + to_offset("8H")

  ts_resampled = ts.resample(
```

Downsampling

In [144…

```
rng = pd.date_range('2000-01-01', periods=12, freq='T')
ts = pd.Series(np.arange(12), index=rng)
ts
```

Out[144]:

```
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
2000-01-01 00:07:00     7
2000-01-01 00:08:00     8
2000-01-01 00:09:00     9
2000-01-01 00:10:00    10
2000-01-01 00:11:00    11
Freq: T, dtype: int32
```

In [146…

```
ts.resample('5min').sum()
```

Out[146]:
```
2000-01-01 00:00:00    10
2000-01-01 00:05:00    35
2000-01-01 00:10:00    21
Freq: 5T, dtype: int32
```

In [147…
```
ts.resample('5min', closed='right').sum()
```

Out[147]:
```
1999-12-31 23:55:00     0
2000-01-01 00:00:00    15
2000-01-01 00:05:00    40
2000-01-01 00:10:00    11
Freq: 5T, dtype: int32
```

In [148…
```
ts.resample('5min', closed='right', label='right').sum()
```

Out[148]:
```
2000-01-01 00:00:00     0
2000-01-01 00:05:00    15
2000-01-01 00:10:00    40
2000-01-01 00:15:00    11
Freq: 5T, dtype: int32
```

In [149…
```
ts.resample('5min', closed='right',
            label='right', loffset='-1s').sum()
```

```
C:\Users\Usuário\AppData\Local\Temp\ipykernel_9644\95277004.py:1: FutureWarning:
'loffset' in .resample() and in Grouper() is deprecated.

>>> df.resample(freq="3s", loffset="8H")

becomes:

>>> from pandas.tseries.frequencies import to_offset
>>> df = df.resample(freq="3s").mean()
>>> df.index = df.index.to_timestamp() + to_offset("8H")

  ts.resample('5min', closed='right',
```

Out[149]:
```
1999-12-31 23:59:59     0
2000-01-01 00:04:59    15
2000-01-01 00:09:59    40
2000-01-01 00:14:59    11
Freq: 5T, dtype: int32
```

Open-High-Low-Close (OHLC) resampling

In [150…
```
ts.resample('5min').ohlc()
```

Out[150]:

	open	high	low	close
2000-01-01 00:00:00	0	4	0	4
2000-01-01 00:05:00	5	9	5	9
2000-01-01 00:10:00	10	11	10	11

Upsampling and Interpolation

In [151…
```
frame = pd.DataFrame(np.random.randn(2, 4),
                     index=pd.date_range('1/1/2000', periods=2,
                                         freq='W-WED'),
                     columns=['Colorado', 'Texas', 'New York', 'Ohio'])
frame
```

Out[151]:

	Colorado	Texas	New York	Ohio
2000-01-05	-0.896431	0.677263	0.036503	0.087102
2000-01-12	-0.046662	0.927238	0.482284	-0.867130

In [152…

```
df_daily = frame.resample('D').asfreq()
df_daily
```

Out[152]:

	Colorado	Texas	New York	Ohio
2000-01-05	-0.896431	0.677263	0.036503	0.087102
2000-01-06	NaN	NaN	NaN	NaN
2000-01-07	NaN	NaN	NaN	NaN
2000-01-08	NaN	NaN	NaN	NaN
2000-01-09	NaN	NaN	NaN	NaN
2000-01-10	NaN	NaN	NaN	NaN
2000-01-11	NaN	NaN	NaN	NaN
2000-01-12	-0.046662	0.927238	0.482284	-0.867130

In [153…

```
frame.resample('D').ffill()
```

Out[153]:

	Colorado	Texas	New York	Ohio
2000-01-05	-0.896431	0.677263	0.036503	0.087102
2000-01-06	-0.896431	0.677263	0.036503	0.087102
2000-01-07	-0.896431	0.677263	0.036503	0.087102
2000-01-08	-0.896431	0.677263	0.036503	0.087102
2000-01-09	-0.896431	0.677263	0.036503	0.087102
2000-01-10	-0.896431	0.677263	0.036503	0.087102
2000-01-11	-0.896431	0.677263	0.036503	0.087102
2000-01-12	-0.046662	0.927238	0.482284	-0.867130

In [154…

```
frame.resample('D').ffill(limit=2)
```

Out[154]:

	Colorado	Texas	New York	Ohio
2000-01-05	-0.896431	0.677263	0.036503	0.087102
2000-01-06	-0.896431	0.677263	0.036503	0.087102
2000-01-07	-0.896431	0.677263	0.036503	0.087102
2000-01-08	NaN	NaN	NaN	NaN
2000-01-09	NaN	NaN	NaN	NaN
2000-01-10	NaN	NaN	NaN	NaN
2000-01-11	NaN	NaN	NaN	NaN
2000-01-12	-0.046662	0.927238	0.482284	-0.867130

In [155…
```
frame.resample('W-THU').ffill()
```

Out[155]:

	Colorado	Texas	New York	Ohio
2000-01-06	-0.896431	0.677263	0.036503	0.087102
2000-01-13	-0.046662	0.927238	0.482284	-0.867130

Resampling with Periods

In [156…
```
frame = pd.DataFrame(np.random.randn(24, 4),
                     index=pd.period_range('1-2000', '12-2001',
                                           freq='M'),
                     columns=['Colorado', 'Texas', 'New York', 'Ohio'])
frame[:5]
```

Out[156]:

	Colorado	Texas	New York	Ohio
2000-01	0.493841	-0.155434	1.397286	1.507055
2000-02	-1.179442	0.443171	1.395676	-0.529658
2000-03	0.787358	0.248845	0.743239	1.267746
2000-04	1.302395	-0.272154	-0.051532	-0.467740
2000-05	-1.040816	0.426419	0.312945	-1.115689

In [157…
```
annual_frame = frame.resample('A-DEC').mean()
annual_frame
```

Out[157]:

	Colorado	Texas	New York	Ohio
2000	0.556703	0.016631	0.111873	-0.027445
2001	0.046303	0.163344	0.251503	-0.157276

In [158…
```
# Q-DEC: Quarterly, year ending in December
annual_frame.resample('Q-DEC').ffill()
```

Out[158]:

	Colorado	Texas	New York	Ohio
2000Q1	0.556703	0.016631	0.111873	-0.027445
2000Q2	0.556703	0.016631	0.111873	-0.027445
2000Q3	0.556703	0.016631	0.111873	-0.027445
2000Q4	0.556703	0.016631	0.111873	-0.027445
2001Q1	0.046303	0.163344	0.251503	-0.157276
2001Q2	0.046303	0.163344	0.251503	-0.157276
2001Q3	0.046303	0.163344	0.251503	-0.157276
2001Q4	0.046303	0.163344	0.251503	-0.157276

In [159…
```
annual_frame.resample('Q-DEC', convention='end').ffill()
```

Out[159]:

	Colorado	Texas	New York	Ohio
2000Q4	0.556703	0.016631	0.111873	-0.027445
2001Q1	0.556703	0.016631	0.111873	-0.027445
2001Q2	0.556703	0.016631	0.111873	-0.027445
2001Q3	0.556703	0.016631	0.111873	-0.027445
2001Q4	0.046303	0.163344	0.251503	-0.157276

In [160…

```
annual_frame.resample('Q-MAR').ffill()
```

Out[160]:

	Colorado	Texas	New York	Ohio
2000Q4	0.556703	0.016631	0.111873	-0.027445
2001Q1	0.556703	0.016631	0.111873	-0.027445
2001Q2	0.556703	0.016631	0.111873	-0.027445
2001Q3	0.556703	0.016631	0.111873	-0.027445
2001Q4	0.046303	0.163344	0.251503	-0.157276
2002Q1	0.046303	0.163344	0.251503	-0.157276
2002Q2	0.046303	0.163344	0.251503	-0.157276
2002Q3	0.046303	0.163344	0.251503	-0.157276

Moving Window Functions

In [161…

```
close_px_all = pd.read_csv('examples/stock_px_2.csv',
                           parse_dates=True, index_col=0)
close_px = close_px_all[['AAPL', 'MSFT', 'XOM']]
close_px
```

Out[161]:

	AAPL	MSFT	XOM
2003-01-02	7.40	21.11	29.22
2003-01-03	7.45	21.14	29.24
2003-01-06	7.45	21.52	29.96
2003-01-07	7.43	21.93	28.95
2003-01-08	7.28	21.31	28.83
…	…	…	…
2011-10-10	388.81	26.94	76.28
2011-10-11	400.29	27.00	76.27
2011-10-12	402.19	26.96	77.16
2011-10-13	408.43	27.18	76.37
2011-10-14	422.00	27.27	78.11

2214 rows × 3 columns

In [163…
```
close_px = close_px.resample('B').ffill()
close_px
```

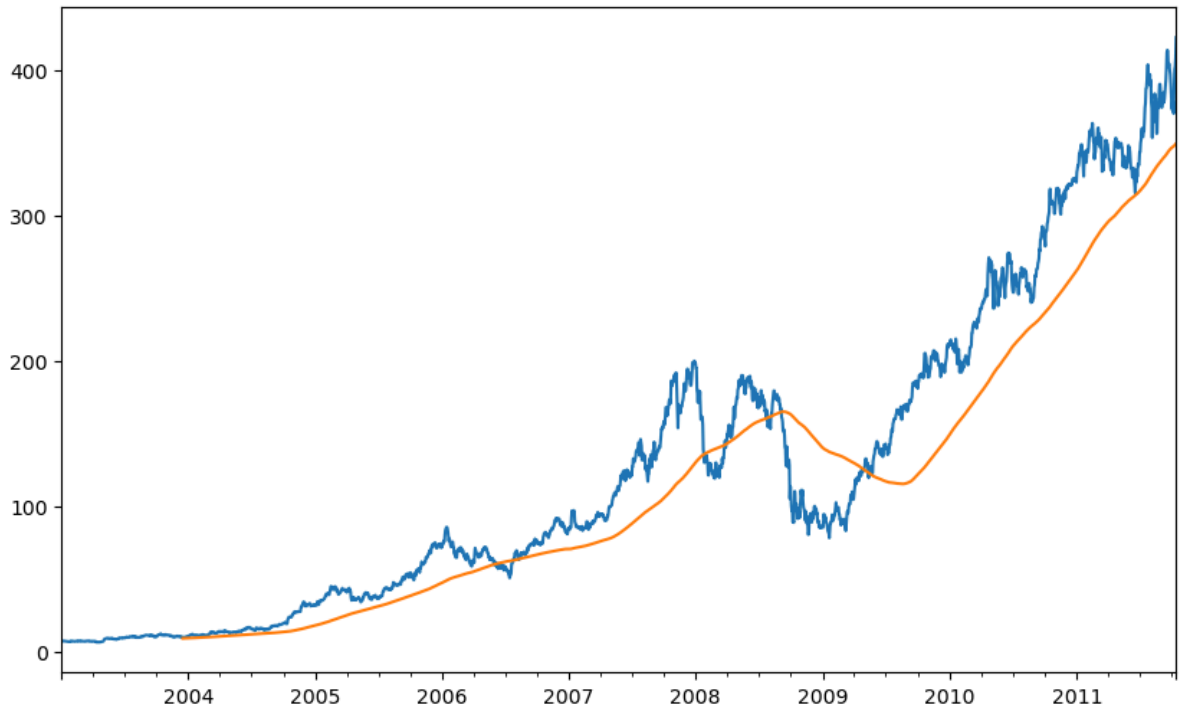
Out[163]:

	AAPL	MSFT	XOM
2003-01-02	7.40	21.11	29.22
2003-01-03	7.45	21.14	29.24
2003-01-06	7.45	21.52	29.96
2003-01-07	7.43	21.93	28.95
2003-01-08	7.28	21.31	28.83
…	…	…	…
2011-10-10	388.81	26.94	76.28
2011-10-11	400.29	27.00	76.27
2011-10-12	402.19	26.96	77.16
2011-10-13	408.43	27.18	76.37
2011-10-14	422.00	27.27	78.11

2292 rows × 3 columns

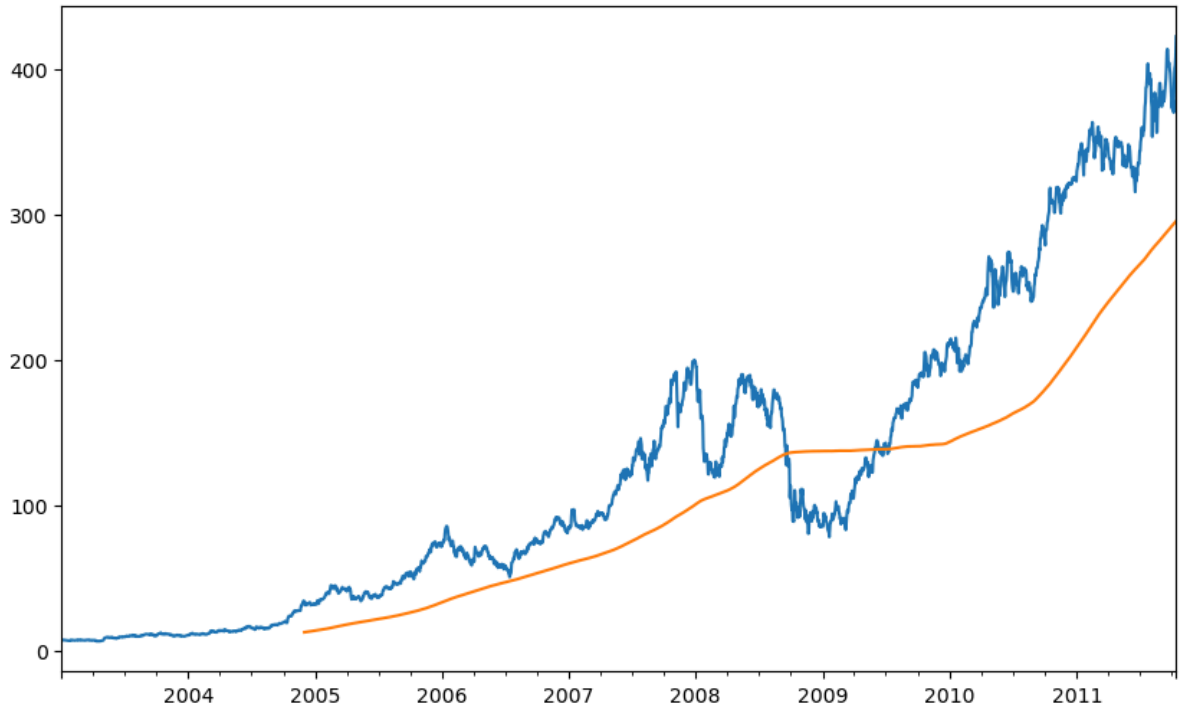
In [164…
```
close_px.AAPL.plot()
close_px.AAPL.rolling(250).mean().plot()
```

Out[164]:
```
<AxesSubplot:>
```



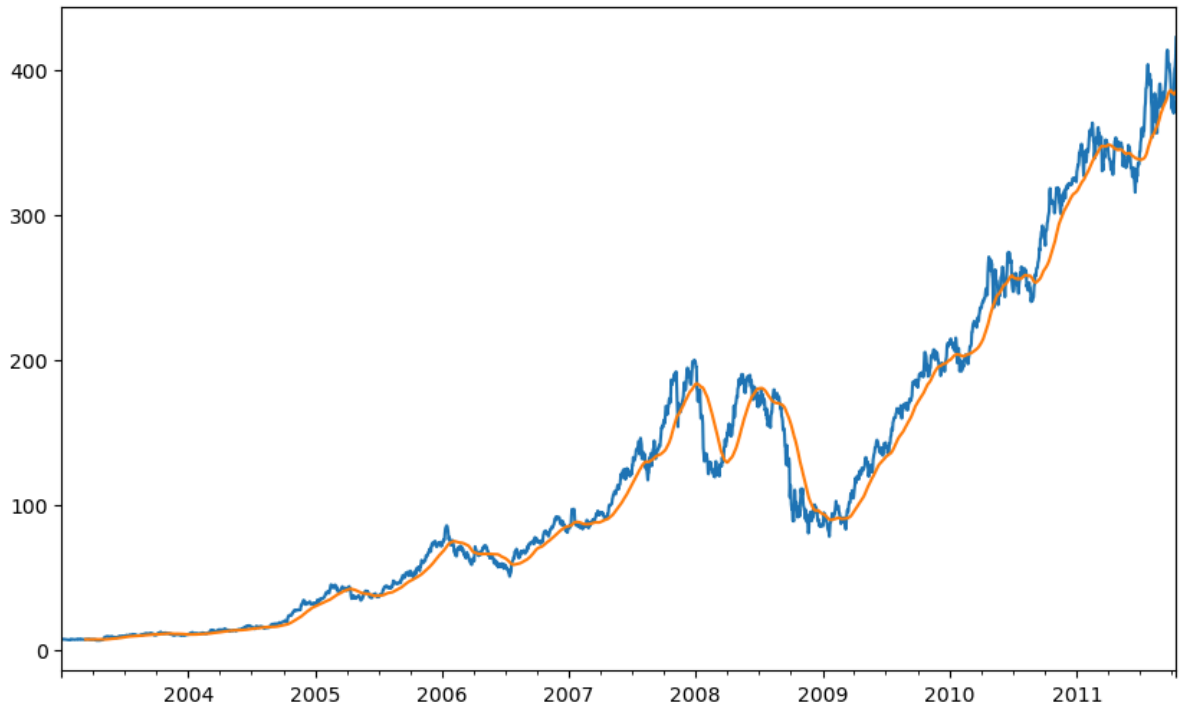
In [166…
```
close_px.AAPL.plot()
close_px.AAPL.rolling(500).mean().plot()
```

Out[166]:
```
<AxesSubplot:>
```



In [167…
```
close_px.AAPL.plot()
close_px.AAPL.rolling(50).mean().plot()
```

Out[167]:
```
<AxesSubplot:>
```

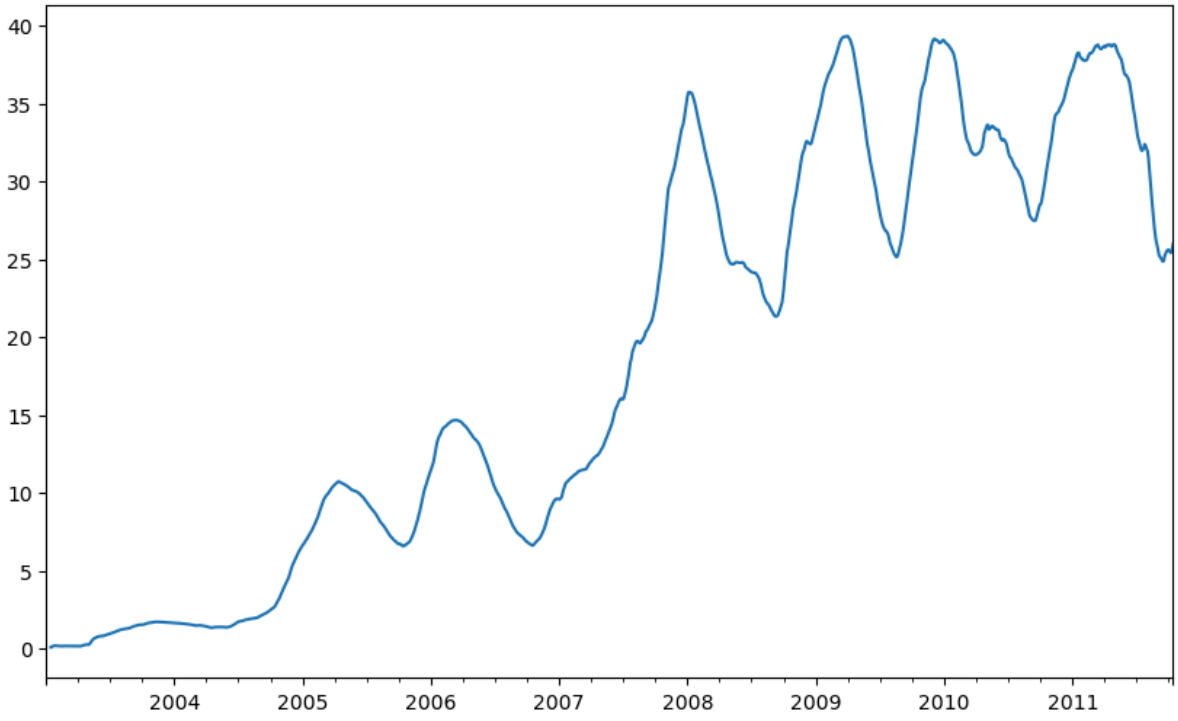


In [168…
```
appl_std250 = close_px.AAPL.rolling(250, min_periods=10).std()
appl_std250[5:12]
```

Out[168]:
```
2003-01-09         NaN
2003-01-10         NaN
2003-01-13         NaN
2003-01-14         NaN
2003-01-15    0.077496
2003-01-16    0.074760
2003-01-17    0.112368
Freq: B, Name: AAPL, dtype: float64
```

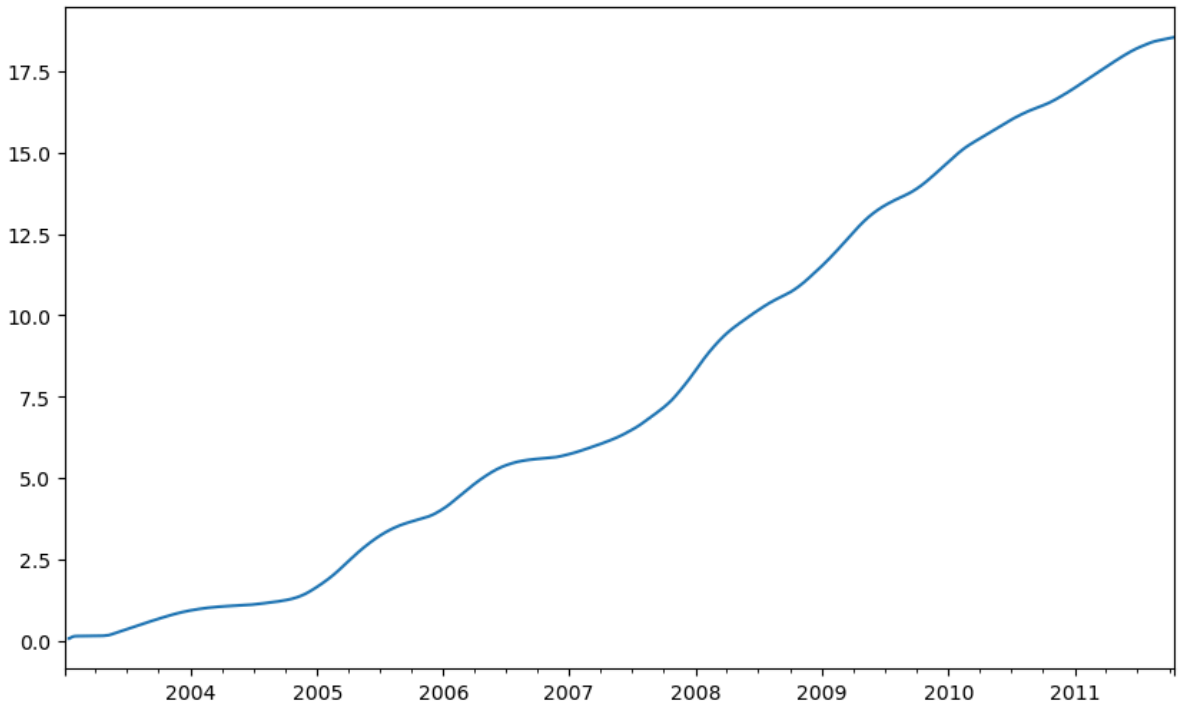
In [169…
```
appl_std250.plot()
```

Out[169]:
```
<AxesSubplot:>
```



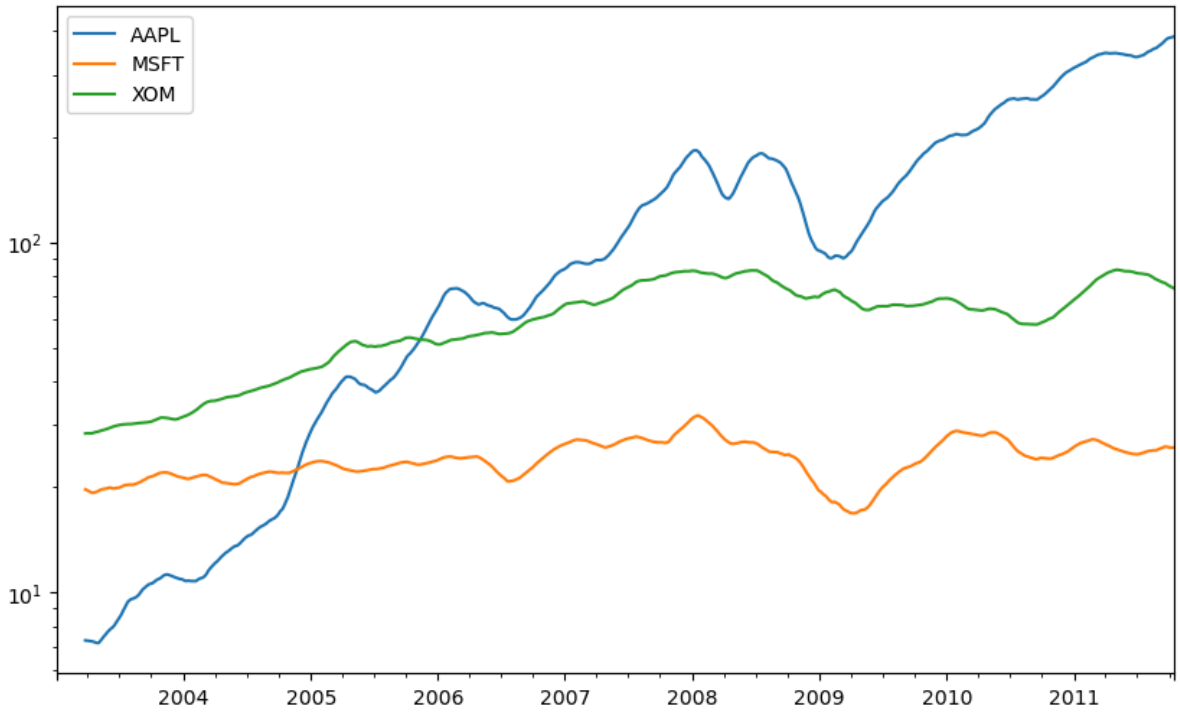
In [174…
```
expanding_mean = appl_std250.expanding().mean()
expanding_mean.plot()
```

Out[174]:
```
<AxesSubplot:>
```



In [172…
```
close_px.rolling(60).mean().plot(logy=True)
```

Out[172]:
```
<AxesSubplot:>
```



In [175…
```
close_px.rolling('20D').mean()
```

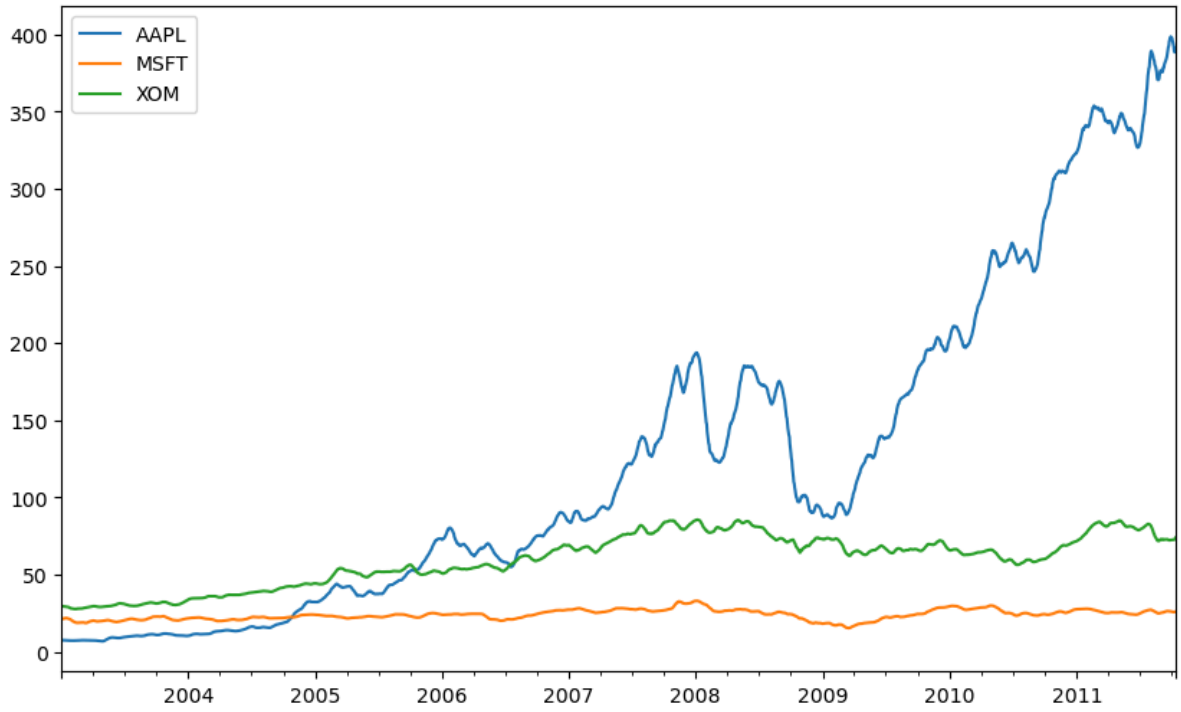
Out[175]:

	AAPL	MSFT	XOM
2003-01-02	7.400000	21.110000	29.220000
2003-01-03	7.425000	21.125000	29.230000
2003-01-06	7.433333	21.256667	29.473333
2003-01-07	7.432500	21.425000	29.342500
2003-01-08	7.402000	21.402000	29.240000
…	…	…	…
2011-10-10	389.351429	25.602143	72.527857
2011-10-11	388.505000	25.674286	72.835000
2011-10-12	388.531429	25.810000	73.400714
2011-10-13	388.826429	25.961429	73.905000
2011-10-14	391.038000	26.048667	74.185333

2292 rows × 3 columns

In [176…
```
close_px.rolling('20D').mean().plot()
```

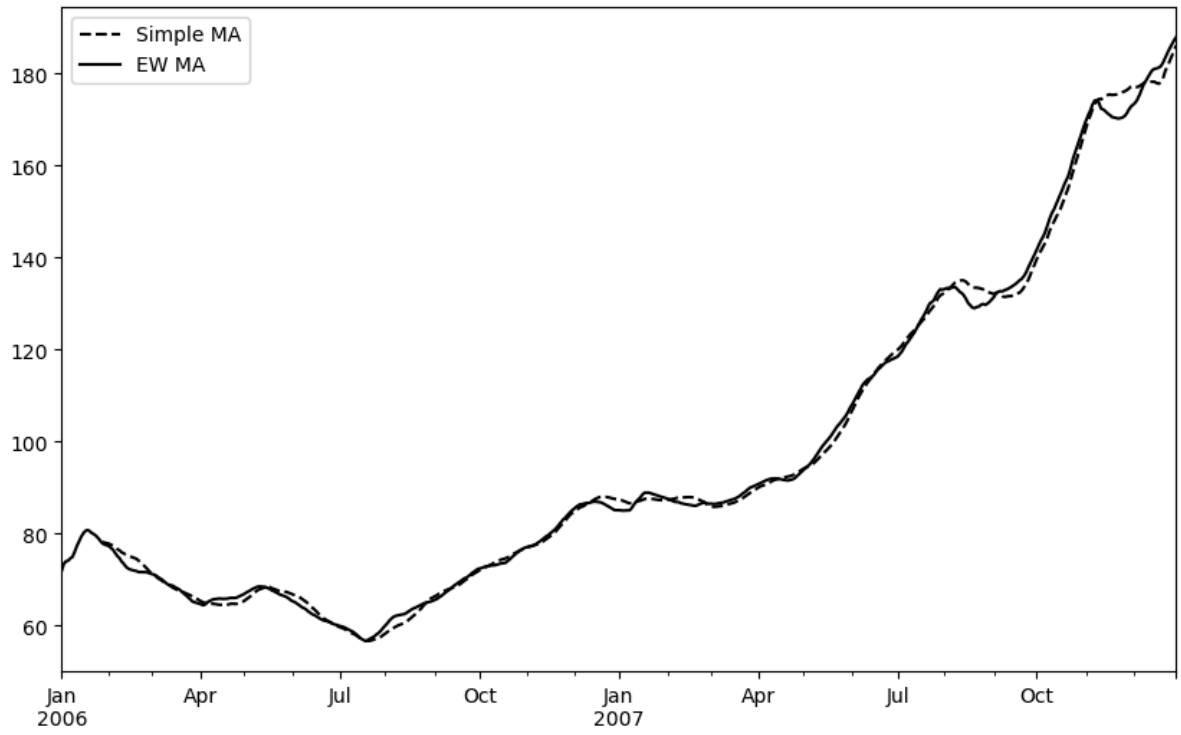
Out[176]:
```
<AxesSubplot:>
```



Exponentially Weighted Functions

In [177…
```
aapl_px = close_px.AAPL['2006':'2007']
ma60 = aapl_px.rolling(30, min_periods=20).mean()
ewma60 = aapl_px.ewm(span=30).mean()
ma60.plot(style='k--', label='Simple MA')
ewma60.plot(style='k-', label='EW MA')
plt.legend()
```

Out[177]:
```
<matplotlib.legend.Legend at 0x25128fdeb20>
```

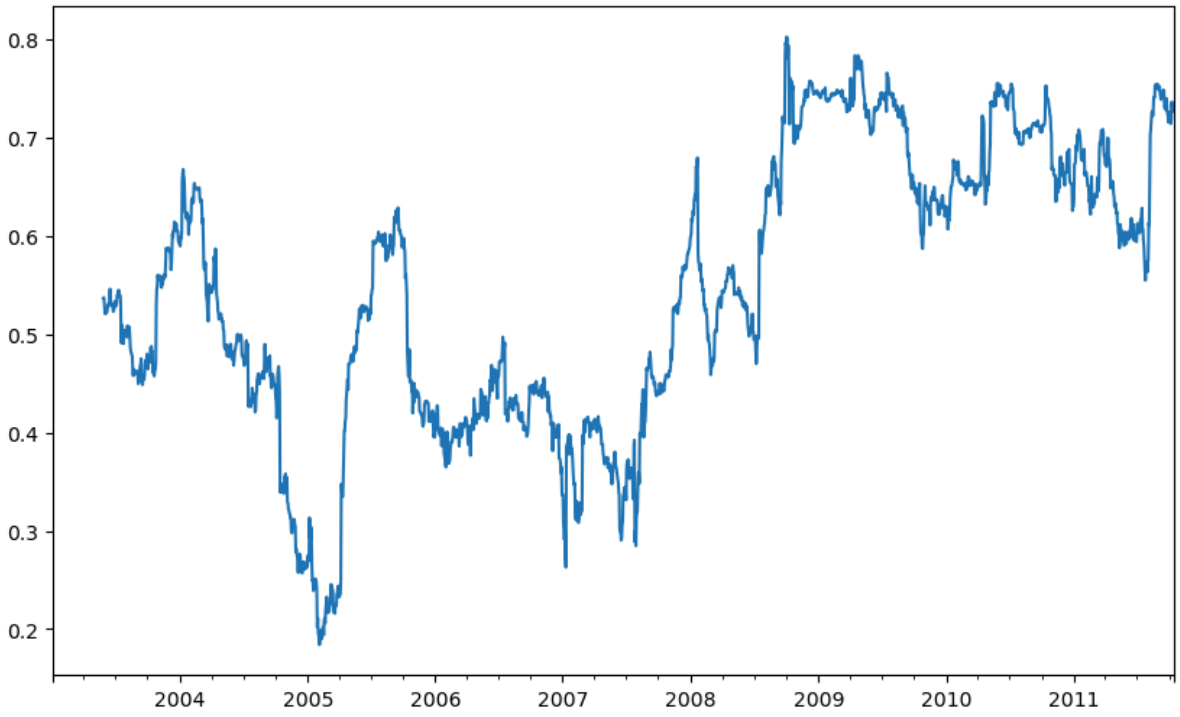


Binary Moving Window Functions

In [178…
```
spx_px = close_px_all['SPX']
spx_rets = spx_px.pct_change()
returns = close_px.pct_change()
```

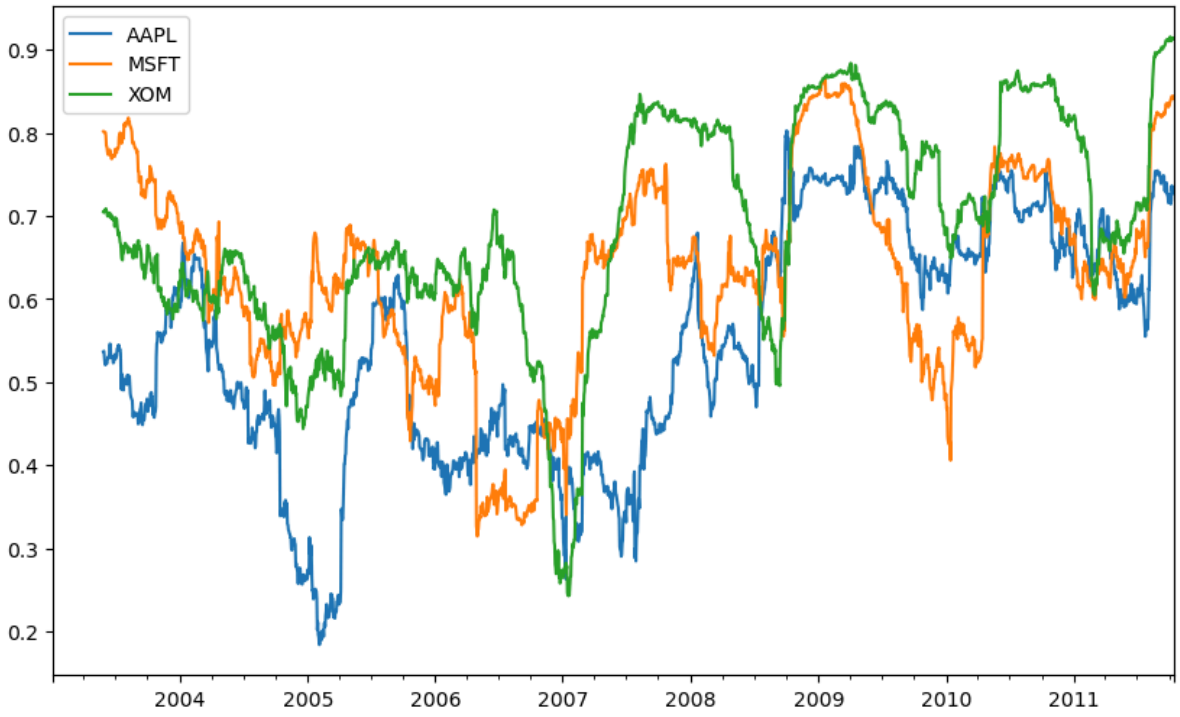
In [179…
```
corr = returns.AAPL.rolling(125, min_periods=100).corr(spx_rets)
corr.plot()
```

Out[179]: <AxesSubplot:>



In [180…
```
corr = returns.rolling(125, min_periods=100).corr(spx_rets)
corr.plot()
```

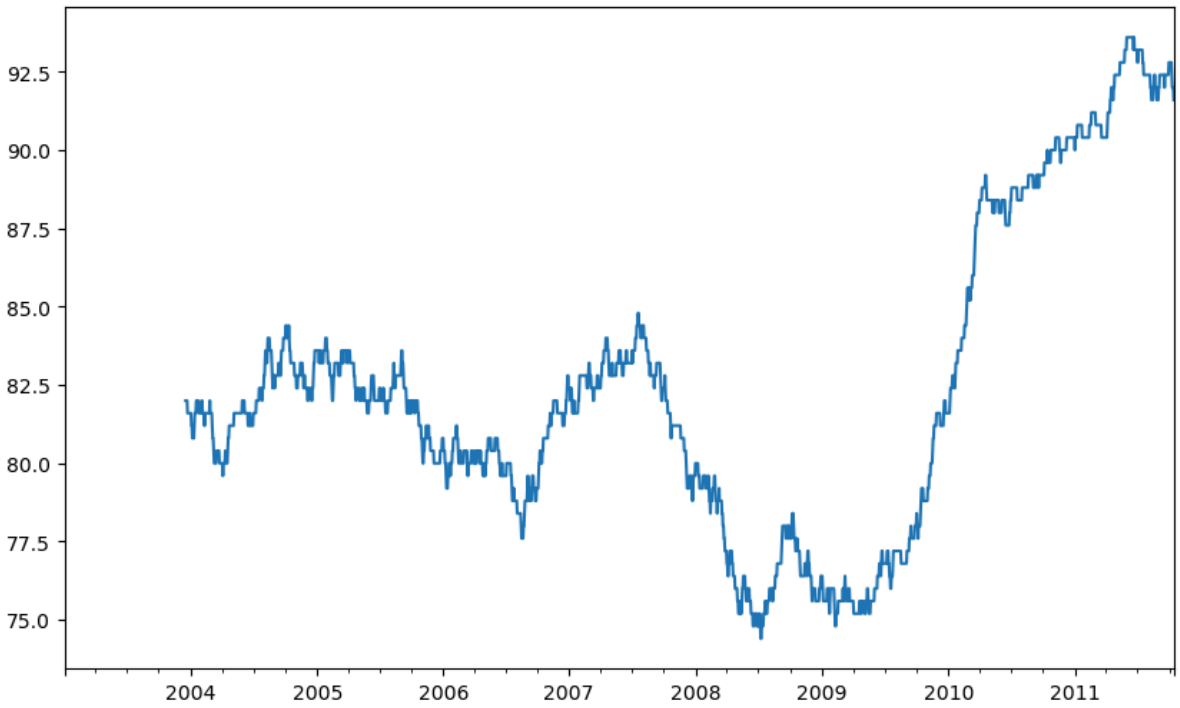
Out[180]: <AxesSubplot:>



User-Defined Moving Window Functions

In [181…
```
from scipy.stats import percentileofscore
score_at_2percent = lambda x: percentileofscore(x, 0.02)
result = returns.AAPL.rolling(250).apply(score_at_2percent)
result.plot()
```

Out[181]:
```
<AxesSubplot:>
```



In [182…
```
pd.options.display.max_rows = PREVIOUS_MAX_ROWS
```

Conclusion