Python Numpy, Pandasによる データ分析入門

日付と時刻

In [17]:
```
from datetime import datetime, date, time
dt = datetime(2020, 6, 16, 15, 30, 21)
print(dt.day,dt.minute)
```

```
16 30
```

In [18]:
```
print(dt.date(),dt.time())
```

```
2020-06-16 15:30:21
```

In [19]:
```
dt.strftime('%m/%d/%Y %H:%M')
```

Out[19]:
```
'06/16/2020 15:30'
```

In [21]:
```
datetime.strptime('2020616', '%Y%m%d')
```

Out[21]:
```
datetime.datetime(2020, 6, 16, 0, 0)
```

In [22]:
```
dt.replace(minute=0, second=0)
```

Out[22]:
```
datetime.datetime(2020, 6, 16, 15, 0)
```

In [24]:
```
dt2 = datetime(2020, 6, 30, 19, 30, 21)
delta = dt2 - dt
print(delta, type(delta))
```

```
14 days, 4:00:00 <class 'datetime.timedelta'>
```

時系列の基本

In [26]:
```
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)
```

In [27]:
```
#Date and Time Data Types and Tools
from datetime import datetime
now = datetime.now()
print(now)
print(now.year, now.month, now.day)
```

```
2020-06-16 14:59:23.489791
2020 6 16
```

In [29]:
```
delta = datetime(2011, 1, 7) - datetime(2008, 6, 24, 8, 15)
print(delta, delta.days, delta.seconds)
```

```
926 days, 15:45:00 926 56700
```

In [30]:
```
from datetime import timedelta
start = datetime(2011, 1, 7)
start + timedelta(12)
start - 2 * timedelta(12)
```

Out[30]:
```
datetime.datetime(2010, 12, 14, 0, 0)
```

In [ ]:
```
#Converting Between String and Datetime
```

In [31]:
```
stamp = datetime(2011, 1, 3)
str(stamp)
stamp.strftime('%Y-%m-%d')
```

Out[31]:
```
'2011-01-03'
```

In [32]:
```
value = '2011-01-03'
datetime.strptime(value, '%Y-%m-%d')
datestrs = ['7/6/2011', '8/6/2011']
[datetime.strptime(x, '%m/%d/%Y') for x in datestrs]
```

Out[32]:
```
[datetime.datetime(2011, 7, 6, 0, 0), datetime.datetime(2011, 8, 6, 0, 0)]
```

In [33]:
```
from dateutil.parser import parse
parse('2011-01-03')
```

Out[33]:
```
datetime.datetime(2011, 1, 3, 0, 0)
```

In [34]:
```
parse('Jan 31, 1997 10:45 PM')
```

Out[34]:
```
datetime.datetime(1997, 1, 31, 22, 45)
```

In [35]:
```
parse('6/12/2011', dayfirst=True)
```

Out[35]:
```
datetime.datetime(2011, 12, 6, 0, 0)
```

In [36]:
```
datestrs = ['2011-07-06 12:00:00', '2011-08-06 00:00:00']
pd.to_datetime(datestrs)
```

Out[36]:
```
DatetimeIndex(['2011-07-06 12:00:00', '2011-08-06 00:00:00'], dtype='datetime64[ns]', freq=None)
```

In [37]:
```
idx = pd.to_datetime(datestrs + [None])
idx
idx[2]
pd.isnull(idx)
```

Out[37]:
```
array([False, False,  True])
```

Time Series Basics

In [39]:
```
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[39]:
```
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 [40]:
```
ts.index
```

Out[40]:
```
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 [41]:
```
ts + ts[::2]
```

Out[41]:
```
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 [42]:
```
ts.index.dtype
```

Out[42]:
```
dtype('<M8[ns]')
```

In [43]:
```
stamp = ts.index[0]
stamp
```

Out[43]:
```
Timestamp('2011-01-02 00:00:00')
```

Indexing, Selection, Subsetting

In [44]:
```
stamp = ts.index[2]
ts[stamp]
```

Out[44]:
```
-0.5194387150567381
```

In [45]:
```
print(ts['1/10/2011'],ts['20110110'])
```

```
1.9657805725027142 1.9657805725027142
```

In [46]:
```
longer_ts = pd.Series(np.random.randn(1000),
                      index=pd.date_range('1/1/2000', periods=1000))
longer_ts
longer_ts['2001']
```

Out[46]:
```
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 [47]:
```
longer_ts['2001-05']
```

Out[47]:
```
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 [48]:
```
ts[datetime(2011, 1, 7):]
```

Out[48]:
```
2011-01-07   -0.519439
2011-01-08   -0.555730
2011-01-10    1.965781
2011-01-12    1.393406
dtype: float64
```

In [49]:
```
ts
ts['1/6/2011':'1/11/2011']
```

Out[49]:
```
2011-01-07   -0.519439
2011-01-08   -0.555730
2011-01-10    1.965781
dtype: float64
```

In [50]:
```
ts.truncate(after='1/9/2011')
```

Out[50]:
```
2011-01-02   -0.204708
2011-01-05    0.478943
2011-01-07   -0.519439
2011-01-08   -0.555730
dtype: float64
```

In [51]:
```
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[51]:

	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 [52]:
```
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[52]:
```
2000-01-01    0
2000-01-02    1
2000-01-02    2
2000-01-02    3
2000-01-03    4
dtype: int32
```

In [53]:
```
dup_ts.index.is_unique
```

Out[53]:
```
False
```

In [54]:
```
dup_ts['1/3/2000']  # not duplicated
dup_ts['1/2/2000']  # duplicated
```

Out[54]:
```
2000-01-02    1
2000-01-02    2
2000-01-02    3
dtype: int32
```

In [55]:
```
grouped = dup_ts.groupby(level=0)
grouped.mean()
```

Out[55]:
```
2000-01-01    0
2000-01-02    2
2000-01-03    4
dtype: int32
```

In [56]:
```
grouped.count()
```

Out[56]:
```
2000-01-01    1
2000-01-02    3
2000-01-03    1
dtype: int64
```

Date Ranges, Frequencies, and Shifting

In [59]:
```
print(ts)
resampler = ts.resample('D')
resampler
```

```
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
```

Out[59]:
```
<pandas.core.resample.DatetimeIndexResampler object at 0x000001C621050A20>
```

Generating Date Ranges

In [60]:
```
index = pd.date_range('2012-04-01', '2012-06-01')
index
```

Out[60]:
```
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 [61]:
```
pd.date_range(start='2012-04-01', periods=20)
pd.date_range(end='2012-06-01', periods=20)
```

Out[61]:
```
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 [62]:
```
pd.date_range('2000-01-01', '2000-12-01', freq='BM')
```

Out[62]:
```
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 [63]:
```
pd.date_range('2012-05-02 12:56:31', periods=5)
```

Out[63]:
```
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 [64]:
```
pd.date_range('2012-05-02 12:56:31', periods=5, normalize=True)
```

Out[64]:
```
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 [65]:
```
from pandas.tseries.offsets import Hour, Minute
hour = Hour()
hour
```

Out[65]:
```
<Hour>
```

In [66]:
```
four_hours = Hour(4)
four_hours
```

Out[66]:
```
<4 * Hours>
```

In [67]:
```
pd.date_range('2000-01-01', '2000-01-03 23:59', freq='4h')
```

Out[67]:
```
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 [68]:
```
Hour(2) + Minute(30)
```

Out[68]:
```
<150 * Minutes>
```

In [69]:
```
pd.date_range('2000-01-01', periods=10, freq='1h30min')
```

Out[69]:
```
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')
```

Converting Timestamps to Periods (and Back)

In [71]:
```
rng = pd.date_range('2000-01-01', periods=3, freq='M')
ts = pd.Series(np.random.randn(3), index=rng)
print(ts)
pts = ts.to_period()
pts
```

```
2000-01-31   -0.517795
2000-02-29   -0.116696
2000-03-31    2.389645
Freq: M, dtype: float64
```

Out[71]:
```
2000-01   -0.517795
2000-02   -0.116696
2000-03    2.389645
Freq: M, dtype: float64
```

In [72]:
```
rng = pd.date_range('1/29/2000', periods=6, freq='D')
ts2 = pd.Series(np.random.randn(6), index=rng)
ts2
ts2.to_period('M')
```

Out[72]:
```
2000-01   -0.932454
2000-01   -0.229331
2000-01   -1.140330
2000-02    0.439920
2000-02   -0.823758
2000-02   -0.520930
Freq: M, dtype: float64
```

In [73]:
```
pts = ts2.to_period()
pts
pts.to_timestamp(how='end')
```

Out[73]:
```
2000-01-29 23:59:59.999999999   -0.932454
2000-01-30 23:59:59.999999999   -0.229331
2000-01-31 23:59:59.999999999   -1.140330
2000-02-01 23:59:59.999999999    0.439920
2000-02-02 23:59:59.999999999   -0.823758
2000-02-03 23:59:59.999999999   -0.520930
Freq: D, dtype: float64
```

Resampling and Frequency Conversion

In [74]:
```
rng = pd.date_range('2000-01-01', periods=100, freq='D')
ts = pd.Series(np.random.randn(len(rng)), index=rng)
ts
ts.resample('M').mean()
ts.resample('M', kind='period').mean()
```

Out[74]:
```
2000-01   -0.000618
2000-02   -0.007856
2000-03    0.172339
2000-04   -0.437396
Freq: M, dtype: float64
```

Downsampling

In [75]:
```
rng = pd.date_range('2000-01-01', periods=12, freq='T')
ts = pd.Series(np.arange(12), index=rng)
ts
```

Out[75]:
```
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 [76]:
```
ts.resample('5min', closed='right').sum()
```

Out[76]:
```
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 [77]:
```
ts.resample('5min', closed='right').sum()
```

Out[77]:
```
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 [78]:
```
ts.resample('5min', closed='right', label='right').sum()
```

Out[78]:
```
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 [79]:
```
ts.resample('5min', closed='right',
            label='right', loffset='-1s').sum()
```

Out[79]:
```
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 [80]:
```
ts.resample('5min').ohlc()
```

Out[80]:

	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 [81]:
```
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[81]:

	Colorado	Texas	New York	Ohio
2000-01-05	-1.333576	-0.309119	0.028558	1.129605
2000-01-12	-0.374173	-0.011401	0.272924	-0.601544

In [82]:
```
df_daily = frame.resample('D').asfreq()
df_daily
```

Out[82]:

	Colorado	Texas	New York	Ohio
2000-01-05	-1.333576	-0.309119	0.028558	1.129605
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	-0.374173	-0.011401	0.272924	-0.601544

In [83]:
```
frame.resample('D').ffill()
```

Out[83]:

	Colorado	Texas	New York	Ohio
2000-01-05	-1.333576	-0.309119	0.028558	1.129605
2000-01-06	-1.333576	-0.309119	0.028558	1.129605
2000-01-07	-1.333576	-0.309119	0.028558	1.129605
2000-01-08	-1.333576	-0.309119	0.028558	1.129605
2000-01-09	-1.333576	-0.309119	0.028558	1.129605
2000-01-10	-1.333576	-0.309119	0.028558	1.129605
2000-01-11	-1.333576	-0.309119	0.028558	1.129605
2000-01-12	-0.374173	-0.011401	0.272924	-0.601544

In [84]:
```
frame.resample('D').ffill(limit=2)
```

Out[84]:

	Colorado	Texas	New York	Ohio
2000-01-05	-1.333576	-0.309119	0.028558	1.129605
2000-01-06	-1.333576	-0.309119	0.028558	1.129605
2000-01-07	-1.333576	-0.309119	0.028558	1.129605
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	-0.374173	-0.011401	0.272924	-0.601544

In [85]:
```
frame.resample('W-THU').ffill()
```

Out[85]:

	Colorado	Texas	New York	Ohio
2000-01-06	-1.333576	-0.309119	0.028558	1.129605
2000-01-13	-0.374173	-0.011401	0.272924	-0.601544

Resampling with Periods

In [86]:
```
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]
annual_frame = frame.resample('A-DEC').mean()
annual_frame
```

Out[86]:

	Colorado	Texas	New York	Ohio
2000	0.865663	0.044803	-0.290540	0.035150
2001	0.016631	0.111873	-0.027445	0.487329

In [87]:
```
# Q-DEC: Quarterly, year ending in December
annual_frame.resample('Q-DEC').ffill()
annual_frame.resample('Q-DEC', convention='end').ffill()
```

Out[87]:

	Colorado	Texas	New York	Ohio
2000Q4	0.865663	0.044803	-0.290540	0.035150
2001Q1	0.865663	0.044803	-0.290540	0.035150
2001Q2	0.865663	0.044803	-0.290540	0.035150
2001Q3	0.865663	0.044803	-0.290540	0.035150
2001Q4	0.016631	0.111873	-0.027445	0.487329

In [88]:
```
annual_frame.resample('Q-MAR').ffill()
```

Out[88]:

	Colorado	Texas	New York	Ohio
2000Q4	0.865663	0.044803	-0.290540	0.035150
2001Q1	0.865663	0.044803	-0.290540	0.035150
2001Q2	0.865663	0.044803	-0.290540	0.035150
2001Q3	0.865663	0.044803	-0.290540	0.035150
2001Q4	0.016631	0.111873	-0.027445	0.487329
2002Q1	0.016631	0.111873	-0.027445	0.487329
2002Q2	0.016631	0.111873	-0.027445	0.487329
2002Q3	0.016631	0.111873	-0.027445	0.487329

Moving Window Functions

In [102]:
```
df = pd.read_csv('EURJPY1440_200612.csv',sep='¥t',header=None)
df.columns = ["Date", "Open", "High", "Low","Close","Volume"]
df.head()
```

Out[102]:

	Date	Open	High	Low	Close	Volume
0	2007-01-01 00:00	157.083	157.185	156.770	157.118	265185
1	2007-01-02 00:00	157.135	157.892	156.837	157.747	264717
2	2007-01-03 00:00	157.750	158.032	157.041	157.163	264115
3	2007-01-04 00:00	157.160	157.310	155.455	155.528	261338
4	2007-01-05 00:00	155.532	155.545	154.012	154.195	253289

In [103]:
```
df['Close'].head()
```

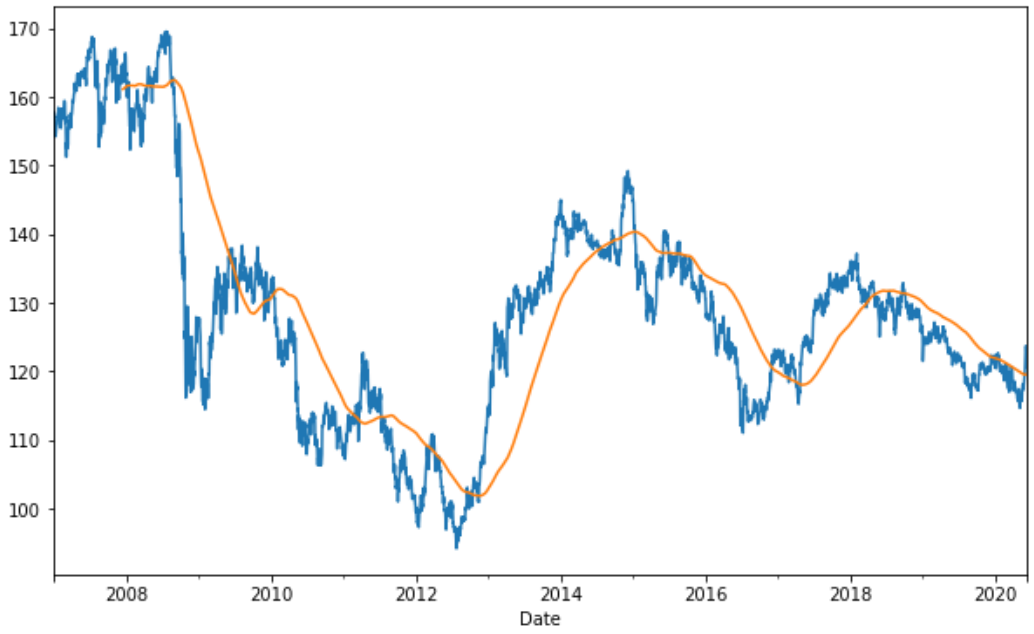
Out[103]:
```
0    157.118
1    157.747
2    157.163
3    155.528
4    154.195
Name: Close, dtype: float64
```

In [112]:
```
df['Date'] = pd.to_datetime(df['Date'])
df=df.set_index(['Date'])
```

In [114]:
```
close_df=df['Close'].resample('B').ffill()
```

In [117]:
```
close_df.plot()
close_df.rolling(250).mean().plot()
```

Out[117]: <matplotlib.axes._subplots.AxesSubplot at 0x1c61fcac668>



In [118]:
```
plt.figure()
```

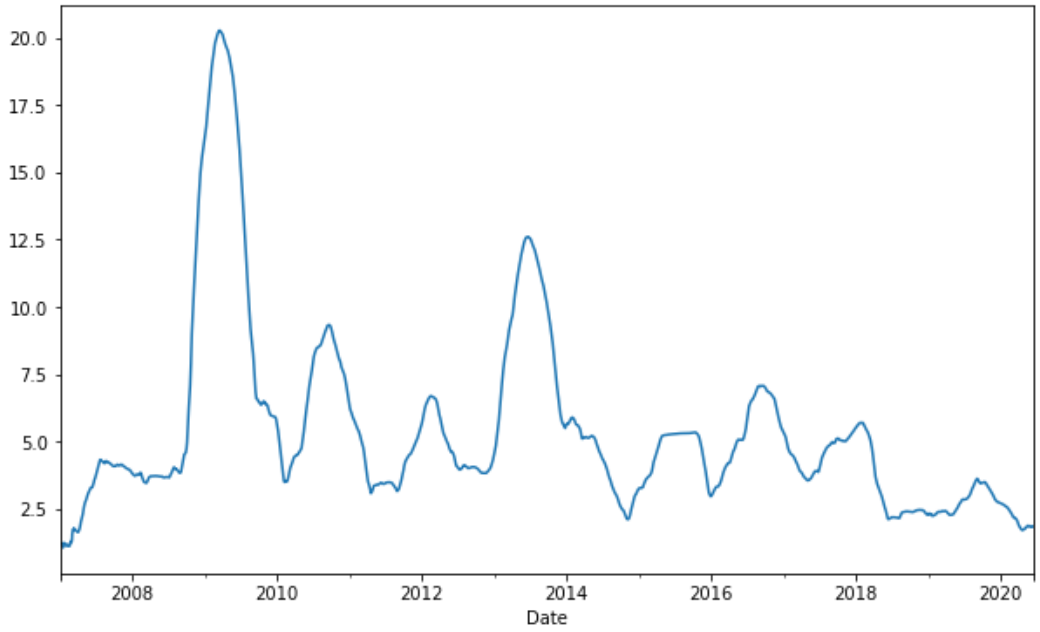
Out[118]: <Figure size 720x432 with 0 Axes>

<Figure size 720x432 with 0 Axes>

In [119]:
```
eurjpy_std250 = close_df.rolling(250, min_periods=10).std()
print(eurjpy_std250[5:12])
eurjpy_std250.plot()
```

```
Date
2007-01-08         NaN
2007-01-09         NaN
2007-01-10         NaN
2007-01-11         NaN
2007-01-12    1.225111
2007-01-15    1.162601
2007-01-16    1.113042
Freq: B, Name: Close, dtype: float64
```

Out[119]: <matplotlib.axes._subplots.AxesSubplot at 0x1c625dd9cc0>



In [120]:
```
expanding_mean = eurjpy_std250.expanding().mean()
```

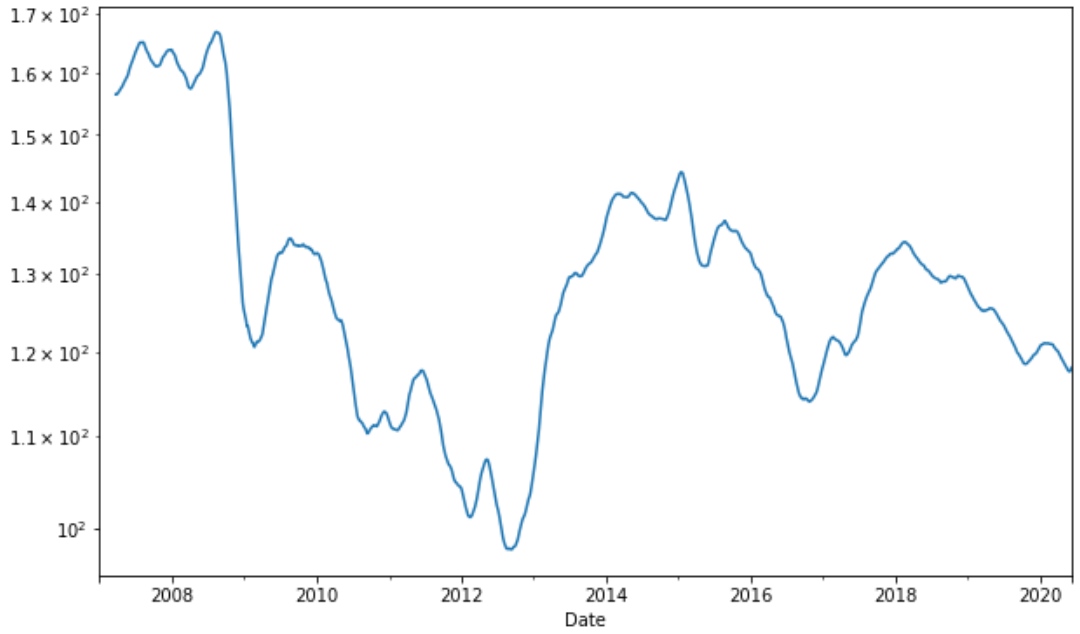
In [121]:
```
plt.figure()
```

Out[121]: <Figure size 720x432 with 0 Axes>

<Figure size 720x432 with 0 Axes>

In [122]:
```
close_df.rolling(60).mean().plot(logy=True)
```

Out[122]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c62175c080>
```



In [123]:
```
close_df.rolling('20D').mean()
```

Out[123]:
```
Date
2007-01-01    157.118000
2007-01-02    157.432500
2007-01-03    157.342667
2007-01-04    156.889000
2007-01-05    156.350200
                 ...
2020-06-05    119.516400
2020-06-08    120.021571
2020-06-09    120.308857
2020-06-10    120.577714
2020-06-11    120.808143
Freq: B, Name: Close, Length: 3509, dtype: float64
```

In [ ]:
```
```

Plotly

In [1]:
```
more EURJPY1440_200612.csv
```

In [1]:
```
import plotly.graph_objects as go

import pandas as pd
from datetime import datetime
```

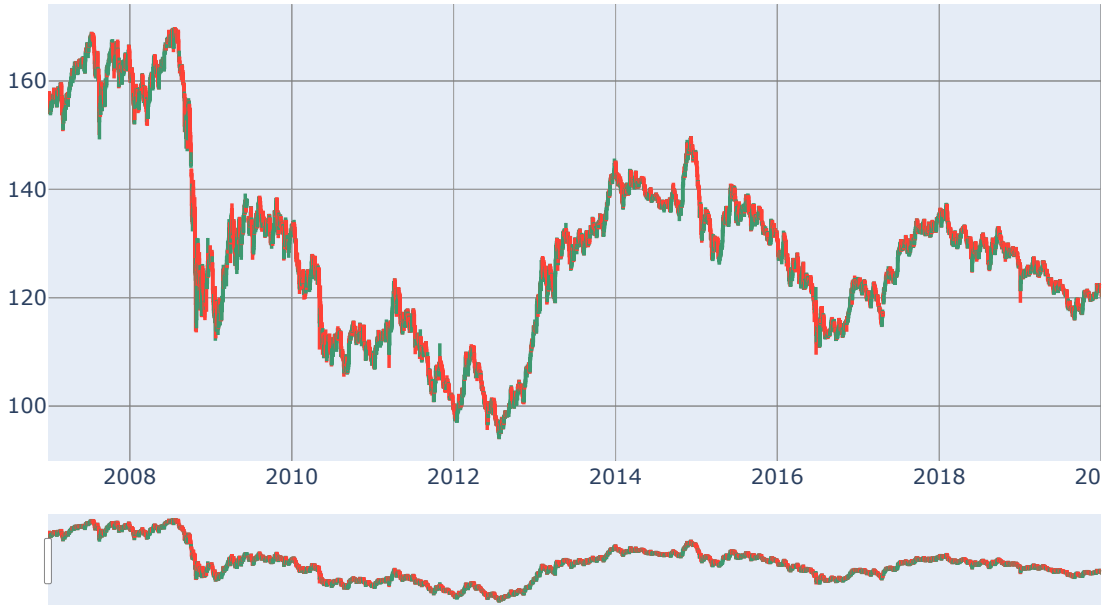
In [6]:
```
df = pd.read_csv('EURJPY1440_200612.csv',sep='¥t',header=None)
df.columns = ["Date", "Open", "High", "Low","Close","Volume"]
df.head()
```

Out[6]:

	Date	Open	High	Low	Close	Volume
0	2007-01-01 00:00	157.083	157.185	156.770	157.118	265185
1	2007-01-02 00:00	157.135	157.892	156.837	157.747	264717
2	2007-01-03 00:00	157.750	158.032	157.041	157.163	264115
3	2007-01-04 00:00	157.160	157.310	155.455	155.528	261338
4	2007-01-05 00:00	155.532	155.545	154.012	154.195	253289

In [7]:
```
fig = go.Figure(data=[go.Candlestick(x=df['Date'],
                open=df['Open'],
                high=df['High'],
                low=df['Low'],
                close=df['Close'])])

fig.show()
```



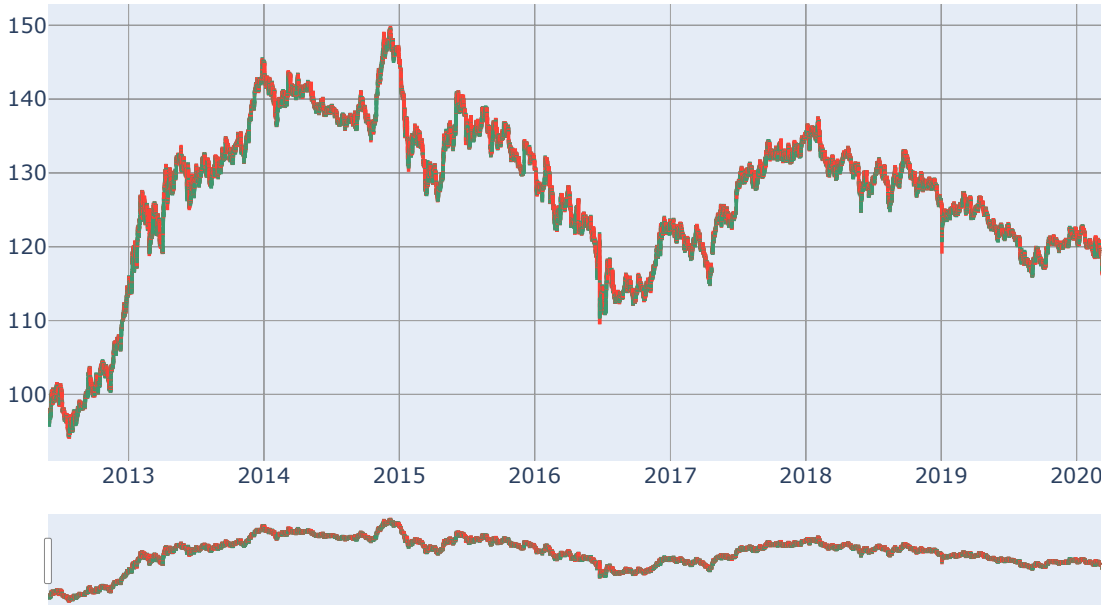
In [8]:
```
df = pd.read_csv('EURJPY60_200612.csv',sep='¥t',header=None)
df.columns = ["Date", "Open", "High", "Low","Close","Volume"]
df.head()
```

Out[8]:

	Date	Open	High	Low	Close	Volume
0	2012-05-31 00:00	97.716	97.740	97.460	97.502	12273
1	2012-05-31 01:00	97.503	97.571	97.394	97.513	9539
2	2012-05-31 02:00	97.508	97.508	97.355	97.396	7039
3	2012-05-31 03:00	97.393	97.482	97.384	97.408	6175
4	2012-05-31 04:00	97.408	97.561	97.391	97.553	7014

In [9]:
```
fig = go.Figure(data=[go.Candlestick(x=df['Date'],
                open=df['Open'],
                high=df['High'],
                low=df['Low'],
                close=df['Close'])])

fig.show()
```



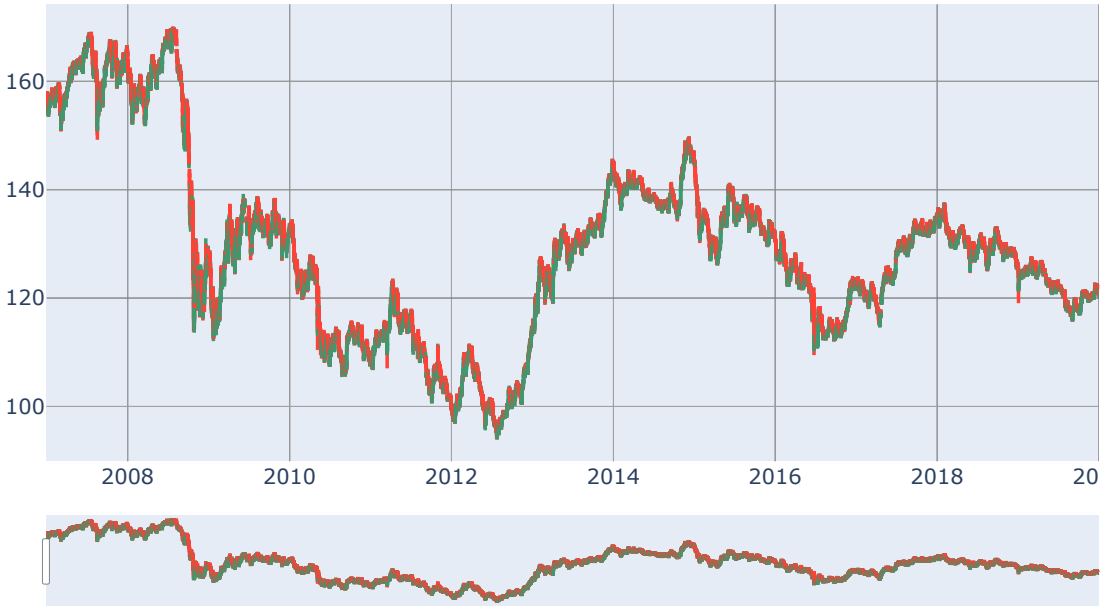
In [2]:
```
df = pd.read_csv('EURJPY240.csv',sep='¥t',header=None)
df.columns = ["Date", "Open", "High", "Low","Close","Volume"]
df.head()
```

Out[2]:

	Date	Open	High	Low	Close	Volume
0	2007-01-01 00:00	157.083	157.092	157.050	157.067	21973
1	2007-01-01 04:00	157.070	157.110	157.022	157.039	22556
2	2007-01-01 08:00	157.030	157.051	156.770	156.790	22345
3	2007-01-01 12:00	156.800	156.806	156.770	156.793	21583
4	2007-01-01 16:00	156.793	157.113	156.776	157.040	147041

In [3]:
```
fig = go.Figure(data=[go.Candlestick(x=df['Date'],
                open=df['Open'],
                high=df['High'],
                low=df['Low'],
                close=df['Close'])])

fig.show()
```



In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21621 entries, 0 to 21620
Data columns (total 6 columns):
Date      21621 non-null object
Open      21621 non-null float64
High      21621 non-null float64
Low       21621 non-null float64
Close     21621 non-null float64
Volume    21621 non-null int64
dtypes: float64(4), int64(1), object(1)
memory usage: 1013.6+ KB
```

In [5]:
```
df['Date'] = pd.to_datetime(df['Date'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21621 entries, 0 to 21620
Data columns (total 6 columns):
Date      21621 non-null datetime64[ns]
Open      21621 non-null float64
High      21621 non-null float64
Low       21621 non-null float64
Close     21621 non-null float64
Volume    21621 non-null int64
dtypes: datetime64[ns](1), float64(4), int64(1)
memory usage: 1013.6 KB
```

In [6]:
```
df=df.set_index(['Date'])
```

In [7]:
```
df.tail()
```

Out[7]:

Date	Open	High	Low	Close	Volume
2020-06-11 04:00:00	121.497	121.674	121.275	121.519	71470
2020-06-11 08:00:00	121.517	121.841	121.467	121.600	69396
2020-06-11 12:00:00	121.600	121.633	121.211	121.381	96168
2020-06-11 16:00:00	121.379	121.433	120.692	120.784	75445
2020-06-11 20:00:00	120.785	120.815	120.429	120.563	28688

In [8]:
```
df2020=df['2020']
df2020.head()
#df.set_index('Date', inplace=True)
#df.index
#df.head()
```

Out[8]:

Date	Open	High	Low	Close	Volume
2020-01-01 20:00:00	121.851	121.984	121.742	121.981	15749
2020-01-02 00:00:00	121.978	122.021	121.880	121.902	24399
2020-01-02 04:00:00	121.901	121.968	121.790	121.953	21817
2020-01-02 08:00:00	121.954	122.017	121.657	121.663	40072
2020-01-02 12:00:00	121.664	121.783	120.981	121.276	151337

In [11]:
```
fig = go.Figure(data=[go.Candlestick(x=df2020.index,
                open=df2020['Open'],
                high=df2020['High'],
                low=df2020['Low'],
                close=df2020['Close'])])

fig.show()
```

In [12]:
```
df2020MayJune=df['2020-05':]
df2020MayJune.head()
```

Out[12]:

Date	Open	High	Low	Close	Volume
2020-05-01 00:00:00	117.423	117.514	117.135	117.238	28299
2020-05-01 04:00:00	117.238	117.462	117.148	117.263	28217
2020-05-01 08:00:00	117.262	117.367	117.035	117.076	39171
2020-05-01 12:00:00	117.077	117.765	117.018	117.424	57081
2020-05-01 16:00:00	117.424	117.488	117.246	117.318	30363

In [13]:
```
fig = go.Figure(data=[go.Candlestick(x=df2020MayJune.index,
                open=df2020MayJune['Open'],
                high=df2020MayJune['High'],
                low=df2020MayJune['Low'],
                close=df2020MayJune['Close'])])

fig.show()
```

In [14]:
```
df2020June=df['2020-06-01':]
fig = go.Figure(data=[go.Candlestick(x=df2020June.index,
                open=df2020June['Open'],
                high=df2020June['High'],
                low=df2020June['Low'],
                close=df2020June['Close'])])

fig.show()
```

In [ ]: