データ可視化のためのMatplotlib

外為ユーロ円のテクニカル分析

Yahooサイトから日次の[OHLC, Adj Close]の価格による分析を行います。

In [29]:
```
#!pip install pandas_datareader
import matplotlib.pyplot as plt
import  pandas_datareader as web
eurjpy_day = web.DataReader("EURJPY=X", 'yahoo')
```

In [30]:
```
eurjpy_day.head()
```

Out[30]:

Date	High	Low	Open	Close	Volume	Adj Close
2015-06-23	139.160004	138.289993	138.410004	138.391998	0.0	138.391998
2015-06-24	138.830994	137.692993	138.794998	138.770996	0.0	138.770996
2015-06-25	138.649994	137.804001	138.460999	138.460007	0.0	138.460007
2015-06-28	137.169998	134.445999	134.445999	134.556000	0.0	134.556000
2015-06-29	137.619003	135.912994	137.619003	137.576996	0.0	137.576996

In [31]:
```
eurjpy_day.tail()
```

Out[31]:

Date	High	Low	Open	Close	Volume	Adj Close
2020-06-16	121.214996	120.195999	120.961998	121.000999	0.0	121.000999
2020-06-17	120.528000	119.676003	120.123001	120.176003	0.0	120.176003
2020-06-18	120.190002	119.539001	119.870003	119.914001	0.0	119.914001
2020-06-21	120.391998	119.339996	119.390999	119.403999	0.0	119.403999
2020-06-23	120.890999	119.879997	120.369003	120.791000	0.0	120.791000

移動平均線ゴールデンクロスとデードクロス (トレンドリバーサル)

移動平均線100日,200日

In [56]:
```
sma200 = eurjpy_day["Adj Close"].rolling(200).mean()
sma100 = eurjpy_day["Adj Close"].rolling(100).mean()
print(sma200["2017-01"].head(20),sma100["2017-01"].head(20))
```

```
Date
2017-01-02    118.219550
2017-01-03    118.199085
2017-01-04    118.175885
2017-01-05    118.153895
2017-01-06    118.124995
2017-01-09    118.105320
2017-01-10    118.085880
2017-01-11    118.069155
2017-01-12    118.053480
2017-01-13    118.045595
2017-01-16    118.034940
2017-01-17    118.024225
2017-01-18    118.009470
2017-01-19    118.002195
2017-01-20    117.999045
2017-01-23    117.998560
2017-01-24    117.988570
2017-01-25    117.978815
2017-01-26    117.967090
2017-01-27    117.960645
Name: Adj Close, dtype: float64 Date
2017-01-02    116.60567
2017-01-03    116.70233
2017-01-04    116.79821
2017-01-05    116.89902
2017-01-06    116.98520
2017-01-09    117.08288
2017-01-10    117.17610
2017-01-11    117.26527
2017-01-12    117.35245
2017-01-13    117.43454
2017-01-16    117.50721
2017-01-17    117.57805
2017-01-18    117.63836
2017-01-19    117.70532
2017-01-20    117.77432
2017-01-23    117.83381
2017-01-24    117.89296
2017-01-25    117.97199
2017-01-26    118.04547
2017-01-27    118.11531
Name: Adj Close, dtype: float64
```

In [79]:
```
print(sma200["2018-05"].head(20),sma100["2018-05"].head(20))
```

```
Date
2018-05-01    132.384405
2018-05-02    132.388555
2018-05-03    132.390900
2018-05-06    132.394025
2018-05-07    132.394990
2018-05-08    132.389505
2018-05-09    132.388000
2018-05-10    132.383705
2018-05-13    132.384170
2018-05-14    132.386325
2018-05-15    132.384735
2018-05-16    132.389500
2018-05-17    132.395775
2018-05-20    132.406015
2018-05-21    132.414525
2018-05-22    132.419940
2018-05-23    132.413995
2018-05-24    132.407145
2018-05-27    132.407455
2018-05-28    132.400800
Name: Adj Close, dtype: float64 Date
2018-05-01    133.00467
2018-05-02    132.98234
2018-05-03    132.96693
2018-05-06    132.94803
2018-05-07    132.92156
2018-05-08    132.87796
2018-05-09    132.83424
2018-05-10    132.79458
2018-05-13    132.75834
2018-05-14    132.72235
2018-05-15    132.68385
2018-05-16    132.64043
2018-05-17    132.59954
2018-05-20    132.55174
2018-05-21    132.50576
2018-05-22    132.45706
2018-05-23    132.39127
2018-05-24    132.31143
2018-05-27    132.23072
2018-05-28    132.14821
Name: Adj Close, dtype: float64
```

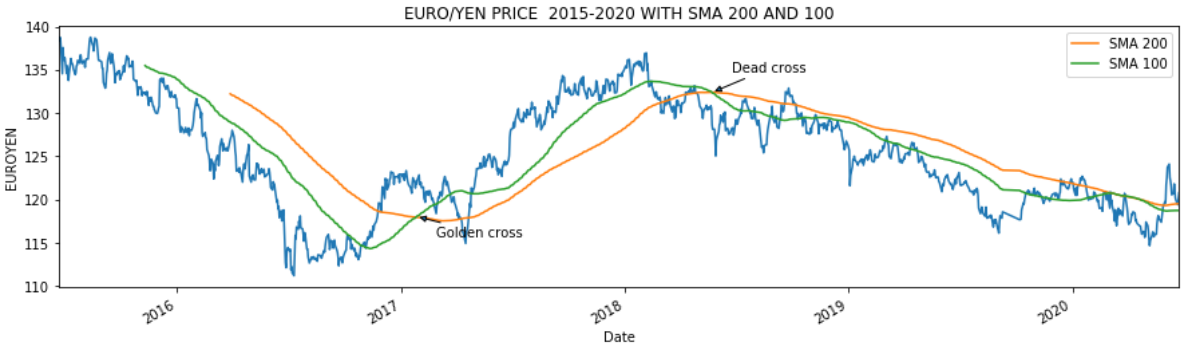
In [72]:
```
import matplotlib.dates as mdates
import datetime as dt

print(eurjpy_day["Adj Close"]["2017-01-26"])
print(mdates.date2num(dt.datetime(2017,1,26)))
```

```
121.79000091552734
736355.0
```

In [ ]:
```

```

In [98]:
```
plt.figure(figsize=(15,4))
plt.title("EURO/YEN PRICE  2015-2020 WITH SMA 200 AND 100")
ax = eurjpy_day["Adj Close"].plot()
plt.ylabel("EUROYEN")
eurjpy_day["Adj Close"].rolling(200).mean().plot()
eurjpy_day["Adj Close"].rolling(100).mean().plot()
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines[1:],["SMA 200","SMA 100"], loc='best')
ax.annotate("Golden cross", (mdates.date2num(dt.datetime(2017,1,26)),sma100["2017-01-26"]),
            xytext=(15, -15), textcoords='offset points',
            arrowprops=dict(arrowstyle='-|>'))
ax.annotate("Dead cross", (mdates.date2num(dt.datetime(2018,5,23)),sma100["2018-05-23"]),
            xytext=(15, 15), textcoords='offset points',
            arrowprops=dict(arrowstyle='-|>'))
```

Out[98]:
```
Text(15, 15, 'Dead cross')
```



In [33]:
```
eurjpy_day.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1283 entries, 2015-06-23 to 2020-06-23
Data columns (total 6 columns):
High         1283 non-null float64
Low          1283 non-null float64
Open         1283 non-null float64
Close        1283 non-null float64
Volume       1283 non-null float64
Adj Close    1283 non-null float64
dtypes: float64(6)
memory usage: 70.2 KB
```

上昇トレンド(bull)と下落トレンド(bear)

In [34]:
```
df_bull=eurjpy_day['2016-08':'2017-12']
df_bull.head()
```

Out[34]:

Date	High	Low	Open	Close	Volume	Adj Close
2016-08-01	114.768997	113.168999	114.250999	114.305000	0.0	114.305000
2016-08-02	113.602997	112.959999	113.382004	113.360001	0.0	113.360001
2016-08-03	113.230003	112.492996	112.959000	113.003998	0.0	113.003998
2016-08-04	112.878998	112.337997	112.709000	112.672997	0.0	112.672997
2016-08-07	113.714996	113.005997	113.124001	113.234001	0.0	113.234001

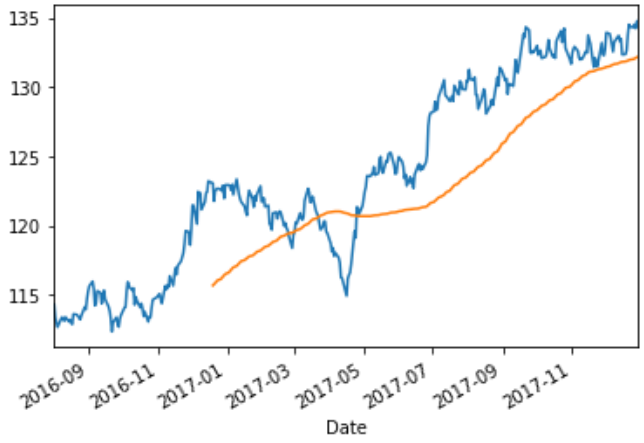
In [35]:
```
df_bear=eurjpy_day['2018':]
df_bear.head()
```

Out[35]:

Date	High	Low	Open	Close	Volume	Adj Close
2018-01-01	135.369995	135.164993	135.240005	135.240005	0.0	135.240005
2018-01-02	135.598007	134.940002	135.445007	135.440002	0.0	135.440002
2018-01-03	135.496002	134.809998	135.406998	135.401001	0.0	135.401001
2018-01-04	136.352997	135.244003	135.253998	135.233994	0.0	135.233994
2018-01-05	136.626007	136.089996	136.093002	136.121994	0.0	136.121994

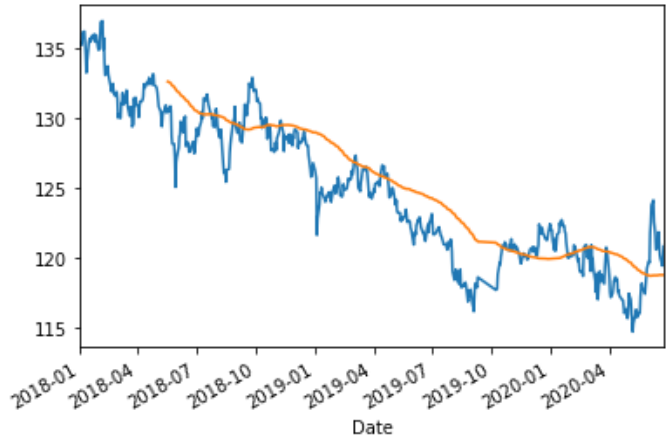
In [36]:
```
df_bull["Adj Close"].plot()
df_bull["Adj Close"].rolling(100).mean().plot()
```

Out[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bdb350e0b8>
```



In [37]:
```
df_bear["Adj Close"].plot()
df_bear["Adj Close"].rolling(100).mean().plot()
#df_bear["Adj Close"].rolling(50).mean().plot()
```

Out[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bdb3317940>
```

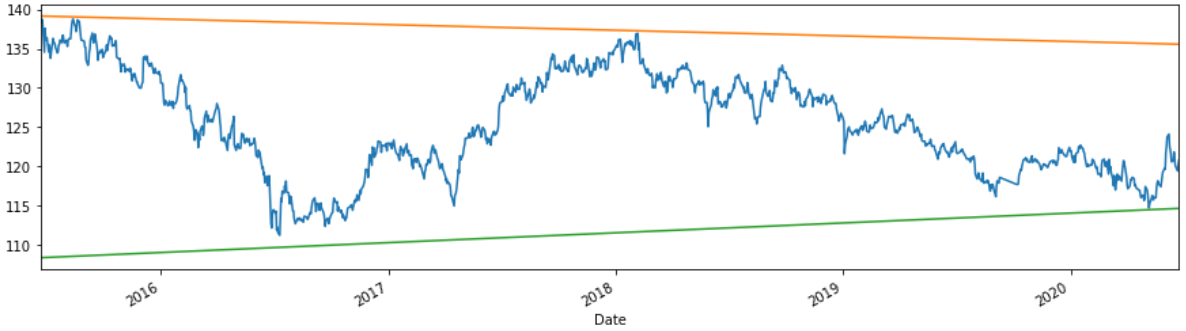


サーポートとレジスタンス（support and resistance）

線形回帰を用いて全期間のサーポートラインとレジスタンスラインを引く

In [39]:
```
from scipy.stats import linregress

data0 = eurjpy_day.copy()

data0['date_id'] = ((data0.index.date - data0.index.date.min())).astype('timedelta64[D]')
data0['date_id'] = data0['date_id'].dt.days + 1

# high trend line  :  レジスタンスライン

data1 = data0.copy()

while len(data1)>3:

    reg = linregress(
                    x=data1['date_id'],
                    y=data1['High'],
                    )
    data1 = data1.loc[data1['High'] > reg[0] * data1['date_id'] + reg[1]]

reg = linregress(
                    x=data1['date_id'],
                    y=data1['High'],
                    )

data0['high_trend'] = reg[0] * data0['date_id'] + reg[1]

# low trend line  : サーポートライン

data1 = data0.copy()

while len(data1)>3:

    reg = linregress(
                    x=data1['date_id'],
                    y=data1['Low'],
                    )
    data1 = data1.loc[data1['Low'] < reg[0] * data1['date_id'] + reg[1]]

reg = linregress(
                    x=data1['date_id'],
                    y=data1['Low'],
                    )

data0['low_trend'] = reg[0] * data0['date_id'] + reg[1]

# plot
plt.figure(figsize=(15,4))
data0['Adj Close'].plot()
data0['high_trend'].plot()
data0['low_trend'].plot()
```

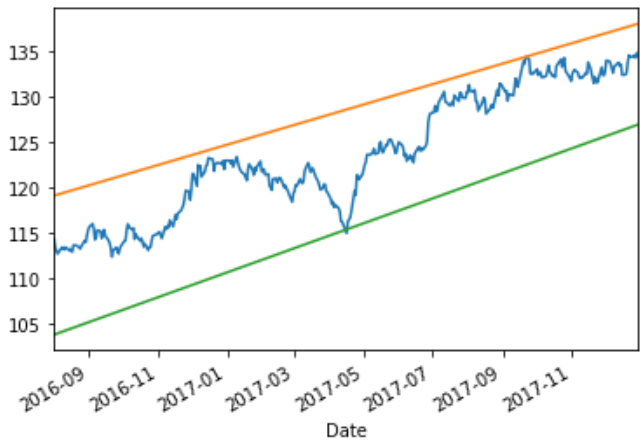
Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x1bdb32a3ef0>



線形回帰を用いて上昇期間のサーポートラインとレジスタンスラインを引く

In [23]:
```
data0 = df_bull.copy()

data0['date_id'] = ((data0.index.date - data0.index.date.min())).astype('timedelta64[D]')
data0['date_id'] = data0['date_id'].dt.days + 1

# high trend line  :  レジスタンスライン

data1 = data0.copy()

while len(data1)>3:

    reg = linregress(
                    x=data1['date_id'],
                    y=data1['High'],
                    )
    data1 = data1.loc[data1['High'] > reg[0] * data1['date_id'] + reg[1]]

reg = linregress(
                    x=data1['date_id'],
                    y=data1['High'],
                    )

data0['high_trend'] = reg[0] * data0['date_id'] + reg[1]

# low trend line  : サーポートライン

data1 = data0.copy()

while len(data1)>3:

    reg = linregress(
                    x=data1['date_id'],
                    y=data1['Low'],
                    )
    data1 = data1.loc[data1['Low'] < reg[0] * data1['date_id'] + reg[1]]

reg = linregress(
                    x=data1['date_id'],
                    y=data1['Low'],
                    )

data0['low_trend'] = reg[0] * data0['date_id'] + reg[1]

# plot

data0['Adj Close'].plot()
data0['high_trend'].plot()
data0['low_trend'].plot()
```

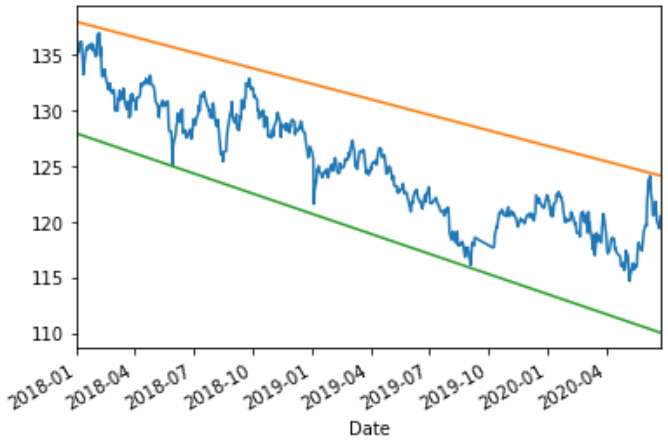
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x1bdb20c1898>



線形回帰を用いて下落期間のサーポートラインとレジスタンスラインを引く

In [24]:
```
data0 = df_bear.copy()

data0['date_id'] = ((data0.index.date - data0.index.date.min())).astype('timedelta64[D]')
data0['date_id'] = data0['date_id'].dt.days + 1

# high trend line  :  レジスタンスライン

data1 = data0.copy()

while len(data1)>3:

    reg = linregress(
                    x=data1['date_id'],
                    y=data1['High'],
                    )
    data1 = data1.loc[data1['High'] > reg[0] * data1['date_id'] + reg[1]]

reg = linregress(
                    x=data1['date_id'],
                    y=data1['High'],
                    )

data0['high_trend'] = reg[0] * data0['date_id'] + reg[1]

# low trend line  : サーポートライン

data1 = data0.copy()

while len(data1)>3:

    reg = linregress(
                    x=data1['date_id'],
                    y=data1['Low'],
                    )
    data1 = data1.loc[data1['Low'] < reg[0] * data1['date_id'] + reg[1]]

reg = linregress(
                    x=data1['date_id'],
                    y=data1['Low'],
                    )

data0['low_trend'] = reg[0] * data0['date_id'] + reg[1]

# plot

data0['Adj Close'].plot()
data0['high_trend'].plot()
data0['low_trend'].plot()
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x1bdb2132898>



日経平均値225の分析

In [82]:
```
n225= web.DataReader("^N225", 'yahoo')
#tick = yf.Ticker('^N225')
#nikkei225 = tick.history(period="max", rounding=True)
#hist = hist['2020-06-19']
```

In [ ]:
```
#hist = nikkei225[-1000:] # Last 1000 days
#h = hist.Close.tolist()
```

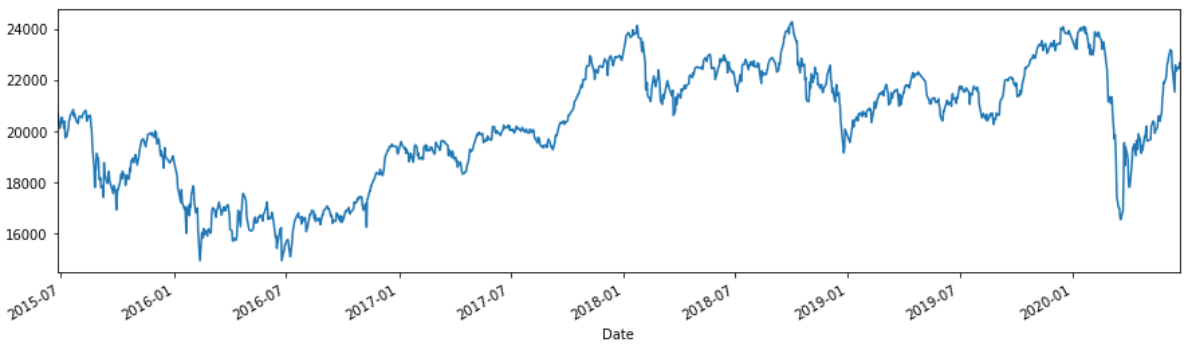
In [83]:
```
print(n225.head(),n225.tail())
```

```
                 High          Low         Open        Close    Volume  ¥
Date
2015-06-25  20866.580078  20758.599609  20777.689453  20771.400391  128300.0
2015-06-26  20785.759766  20650.000000  20758.429688  20706.150391  140100.0
2015-06-29  20361.599609  20093.160156  20305.970703  20109.949219  165300.0
2015-06-30  20243.179688  20118.259766  20174.609375  20235.730469  168100.0
2015-07-01  20346.740234  20225.269531  20291.050781  20329.320312  123600.0

              Adj Close
Date
2015-06-25  20771.400391
2015-06-26  20706.150391
2015-06-29  20109.949219
2015-06-30  20235.730469
2015-07-01  20329.320312                  High          Low         Open        Close    V
olume  ¥
Date
2020-06-17  22536.380859  22318.070312  22517.140625  22455.759766   72500.0
2020-06-18  22432.250000  22125.349609  22363.880859  22355.460938   65200.0
2020-06-19  22523.660156  22352.160156  22515.750000  22478.789062   97000.0
2020-06-22  22575.740234  22311.939453  22353.689453  22437.269531   54600.0
2020-06-23  22693.890625  22257.140625  22636.060547  22656.230469       0.0

              Adj Close
Date
2020-06-17  22455.759766
2020-06-18  22355.460938
2020-06-19  22478.789062
2020-06-22  22437.269531
2020-06-23  22656.230469
```

In [88]:
```
n225_sma200 = n225["Adj Close"].rolling(200).mean()
n225_sma100 = n225["Adj Close"].rolling(100).mean()
#print(sma200["2017-01"].head(20), sma100["2017-01"].head(20))
```

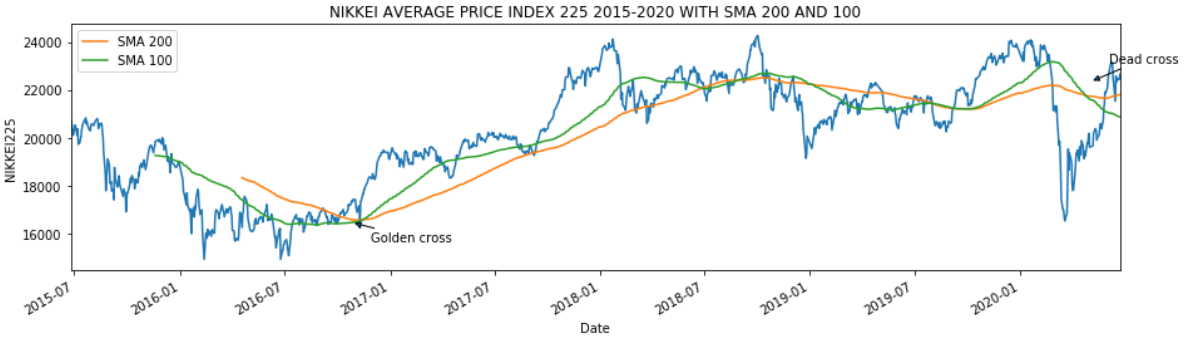
In [85]:
```
plt.figure(figsize=(15,4))
n225["Adj Close"].plot()
```

Out[85]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bdb8db72e8>
```



In [99]:
```
plt.figure(figsize=(15,4))
plt.title("NIKKEI AVERAGE PRICE INDEX 225 2015-2020 WITH SMA 200 AND 100")
ax1 = n225["Adj Close"].plot()
plt.ylabel("NIKKEI225")
n225_sma200.plot()
n225_sma100.plot()
lines, labels = ax1.get_legend_handles_labels()
ax1.legend(lines[1:],["SMA 200","SMA 100"], loc='best')
ax1.annotate("Golden cross", (mdates.date2num(dt.datetime(2016,10,26)),n225_sma100["2016-10-26"]),
             xytext=(15, -15), textcoords='offset points',
             arrowprops=dict(arrowstyle='-|>'))
ax1.annotate("Dead cross", (mdates.date2num(dt.datetime(2020,5,3)),n225_sma100["2020-04-03"]),
             xytext=(15, 15), textcoords='offset points',
             arrowprops=dict(arrowstyle='-|>'))
```

Out[99]: Text(15, 15, 'Dead cross')



In [100]:
```
data0 = n225.copy()

data0['date_id'] = ((data0.index.date - data0.index.date.min())).astype('timedelta64[D]')
data0['date_id'] = data0['date_id'].dt.days + 1

# high trend line  :  レジスタンスライン

data1 = data0.copy()

while len(data1)>3:

    reg = linregress(
                    x=data1['date_id'],
                    y=data1['High'],
                    )
    data1 = data1.loc[data1['High'] > reg[0] * data1['date_id'] + reg[1]]

reg = linregress(
                    x=data1['date_id'],
                    y=data1['High'],
                    )

data0['high_trend'] = reg[0] * data0['date_id'] + reg[1]

# low trend line  : サーポートライン

data1 = data0.copy()

while len(data1)>3:

    reg = linregress(
                    x=data1['date_id'],
                    y=data1['Low'],
                    )
    data1 = data1.loc[data1['Low'] < reg[0] * data1['date_id'] + reg[1]]

reg = linregress(
                    x=data1['date_id'],
                    y=data1['Low'],
                    )

data0['low_trend'] = reg[0] * data0['date_id'] + reg[1]

# plot

data0['Adj Close'].plot()
data0['high_trend'].plot()
data0['low_trend'].plot()
```

Out[100]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bdba3dd9e8>
```

In [ ]:

外為クロス円のデータ分析: 月次

In [18]:
```
#https://www.mizuhobank.co.jp/market/historical.html
import pandas as pd

fx_historical = pd.read_csv("m_quote.csv")
fx_historical
```

Out[18]:

	Unnamed: 0	USD	GBP	EUR	CAD	CHF	SEK	DKK	NOK	AUD	...	HUF	CZK	PLN	RUB
0	2002/4/30	131.15	189.01	115.97	82.83	79.13	12.73	15.61	15.20	70.24	...	*****	*****	*****	NaN
1	2002/5/31	126.44	184.56	115.88	81.58	79.60	12.57	15.59	15.43	69.58	...	*****	*****	*****	NaN
2	2002/6/28	123.53	183.00	117.83	80.64	80.09	12.94	15.86	15.92	70.29	...	*****	*****	*****	NaN
3	2002/7/31	118.05	183.64	117.23	76.52	80.16	12.67	15.79	15.83	65.42	...	*****	*****	*****	NaN
4	2002/8/30	119.08	183.14	116.45	75.85	79.60	12.59	15.69	15.68	64.47	...	*****	*****	*****	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
213	2020/1/31	109.39	142.93	121.35	83.60	112.82	11.50	16.24	12.21	74.95	...	0.36	4.82	28.57	1.77
214	2020/2/28	109.99	142.71	119.99	82.82	112.68	11.36	16.06	11.85	73.41	...	0.36	4.79	28.06	1.72
215	2020/3/31	107.54	133.35	119.14	77.33	112.37	10.98	15.95	10.61	67.06	...	0.35	4.51	26.93	1.46
216	2020/4/30	107.96	133.97	117.37	76.74	111.26	10.76	15.73	10.35	67.82	...	0.33	4.3	25.83	1.44
217	2020/5/29	107.35	131.83	116.99	76.89	110.64	11.04	15.69	10.65	70.03	...	0.33	4.29	25.83	1.48

218 rows × 37 columns

In [ ]:
```

```