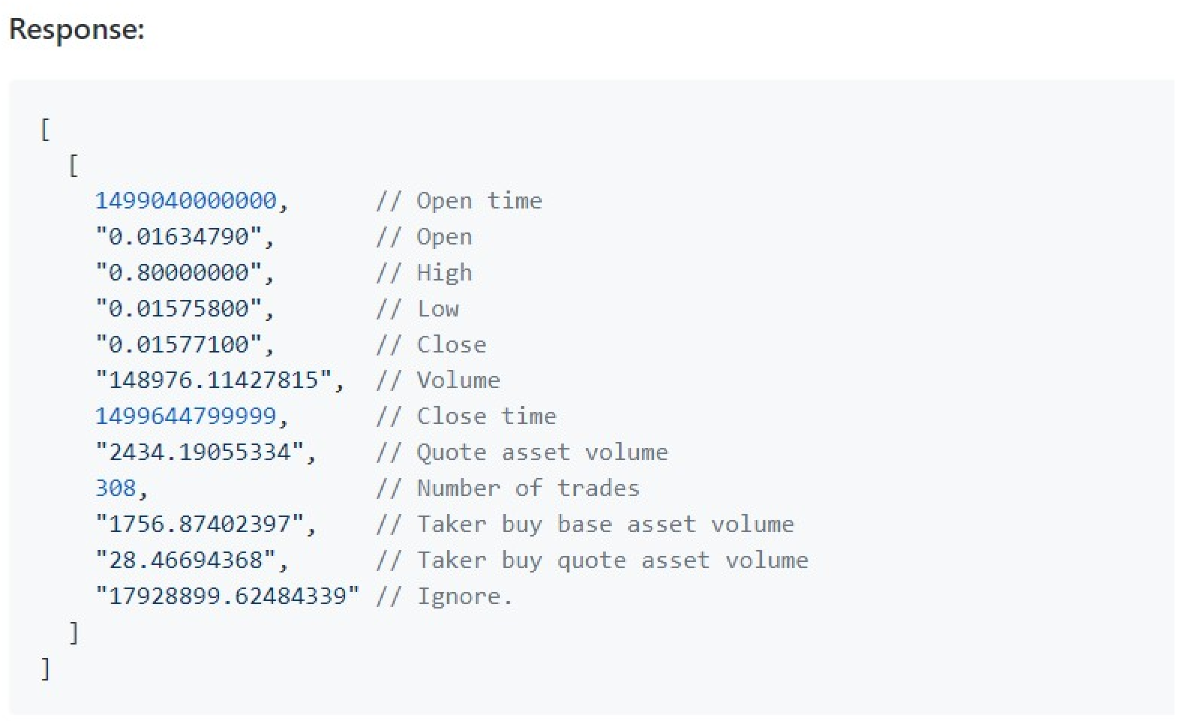
In [1]:
```
import pandas as pd
#import numpy as np
import binance
import ccxt

from pprint import pprint



```

CCXT and Binance REST

In [7]:
```
id = 'binance'; symbol = 'EOS/BTC'; start = '30 minutes ago UTC'; end = 'Now';
key = 'foo'; secret = 'bar'; #when using the private api connection, you need r

#these lines get the latest candlesticks by using ccxt
exchange = getattr(ccxt, id)({})
klines = exchange.fetch_ohlcv(symbol, timeframe, limit=limit)




#while these lines get the latest candlesticks by using directly binances REST
#clientpub = Client(key, secret, {"verify": True, "timeout": 3})
#klines = clientpub.get_historical_klines(symbol, timeframe, start, end)

pprint(klines)


```

```
ETH/BTC
LTC/BTC
BNB/BTC
NEO/BTC
QTUM/ETH
EOS/ETH
SNT/ETH
BNT/ETH
BCH/BTC
GAS/BTC
BNB/ETH
BTC/USDT
ETH/USDT
HSR/BTC
OAX/ETH
DNT/ETH
MCO/ETH
ICN/ETH
MCO/BTC
```

to understand the output, have a look at: https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md (https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md)



Response:

```
[
  [
    1499040000000,      // Open time
    "0.01634790",       // Open
    "0.80000000",       // High
    "0.01575800",       // Low
    "0.01577100",       // Close
    "148976.11427815",  // Volume
    1499644799999,      // Close time
    "2434.19055334",    // Quote asset volume
    308,                // Number of trades
    "1756.87402397",    // Taker buy base asset volume
    "28.46694368",      // Taker buy quote asset volume
    "17928899.62484339" // Ignore.
  ]
]
```

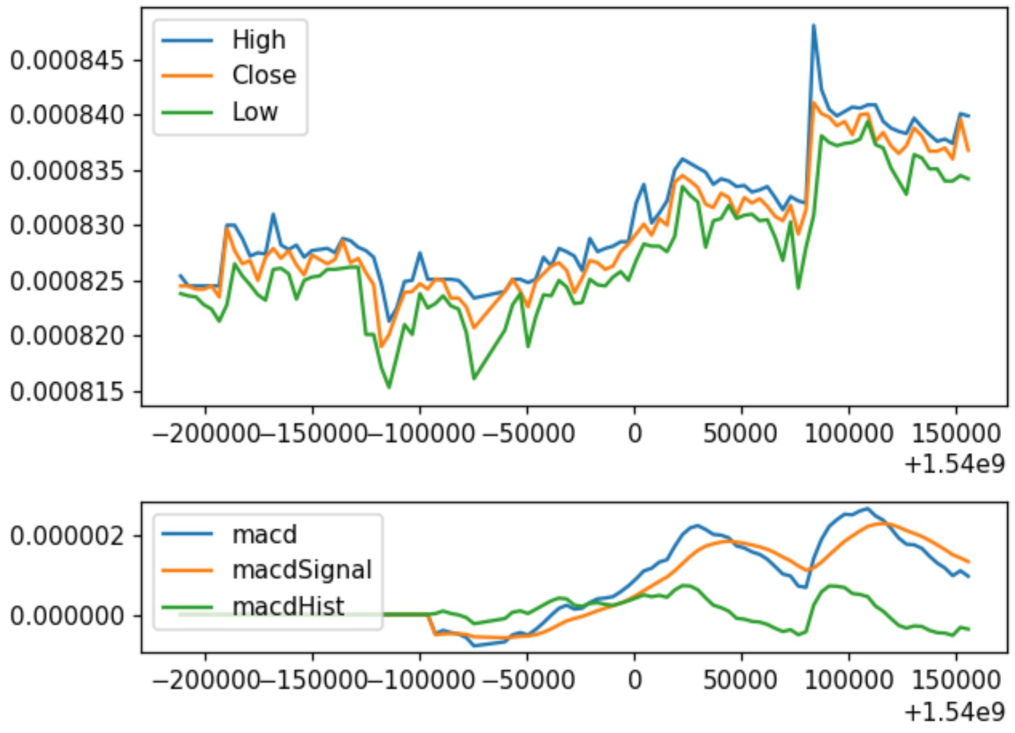
In [3]:

```
#I prefer converting the candles into the feature rich pandas dataframes, which

from collections import deque

candles = deque(maxlen=len(klines))

for k in klines:
    parse = {}
    parse['Opentime'] = float(k[0])/1000
    parse['Open'] = float(k[1])
    parse['High'] = float(k[2])
    parse['Low'] = float(k[3])
    parse['Close'] = float(k[4])
    parse['Volume'] = float(k[5])
    candles.append(parse)

ohlcv = pd.DataFrame(list(candles))
```

```
       Close      High       Low      Open      Opentime     Volume
0   0.000831  0.000831  0.000829  0.000830  1.540364e+09   51392.51
1   0.000830  0.000831  0.000829  0.000831  1.540368e+09   49038.29
2   0.000830  0.000832  0.000829  0.000830  1.540372e+09   64065.18
3   0.000831  0.000832  0.000829  0.000830  1.540375e+09   55238.35
4   0.000832  0.000833  0.000830  0.000831  1.540379e+09   53281.78
5   0.000831  0.000833  0.000830  0.000831  1.540382e+09   64123.61
6   0.000829  0.000832  0.000827  0.000831  1.540386e+09   70697.48
7   0.000830  0.000832  0.000827  0.000830  1.540390e+09   70546.63
8   0.000830  0.000831  0.000828  0.000831  1.540393e+09   43480.18
9   0.000831  0.000831  0.000829  0.000830  1.540397e+09   48428.45
10  0.000831  0.000831  0.000830  0.000831  1.540400e+09   52032.41
11  0.000830  0.000832  0.000829  0.000831  1.540404e+09   44223.21
12  0.000830  0.000832  0.000829  0.000830  1.540408e+09   50223.20
13  0.000833  0.000833  0.000829  0.000830  1.540411e+09   53061.05
14  0.000829  0.000835  0.000827  0.000833  1.540415e+09   71592.90
15  0.000829  0.000830  0.000829  0.000829  1.540418e+09   38047.67
16  0.000828  0.000830  0.000822  0.000829  1.540422e+09  172050.25
17  0.000828  0.000829  0.000827  0.000828  1.540426e+09   54202.62
18  0.000831  0.000832  0.000827  0.000828  1.540429e+09   63585.73
19  0.000831  0.000832  0.000829  0.000831  1.540433e+09   53110.59
20  0.000830  0.000831  0.000829  0.000831  1.540436e+09   36132.51
21  0.000830  0.000831  0.000828  0.000831  1.540440e+09   36536.04
22  0.000830  0.000831  0.000829  0.000830  1.540444e+09   42496.20
23  0.000831  0.000833  0.000830  0.000830  1.540447e+09   36086.46
24  0.000830  0.000831  0.000830  0.000831  1.540451e+09   32129.21
25  0.000831  0.000832  0.000829  0.000830  1.540454e+09   45709.25
26  0.000831  0.000833  0.000830  0.000832  1.540458e+09   46566.87
27  0.000830  0.000834  0.000830  0.000831  1.540462e+09   43967.50
28  0.000830  0.000832  0.000830  0.000830  1.540465e+09   50598.25
29  0.000830  0.000831  0.000829  0.000830  1.540469e+09   53201.38
..       ...       ...       ...       ...           ...        ...
70  0.000832  0.000832  0.000830  0.000832  1.540616e+09   32766.90
71  0.000832  0.000832  0.000831  0.000832  1.540620e+09   31402.30
72  0.000831  0.000832  0.000830  0.000831  1.540624e+09   26646.71
73  0.000831  0.000831  0.000830  0.000831  1.540627e+09   38048.99
74  0.000831  0.000832  0.000830  0.000831  1.540631e+09   38761.81
75  0.000832  0.000832  0.000831  0.000831  1.540634e+09   31988.46
76  0.000831  0.000833  0.000831  0.000832  1.540638e+09   38852.69
77  0.000831  0.000832  0.000831  0.000831  1.540642e+09   28660.13
78  0.000831  0.000833  0.000830  0.000831  1.540645e+09   33136.60
79  0.000833  0.000833  0.000830  0.000831  1.540649e+09   38622.92
80  0.000835  0.000835  0.000831  0.000833  1.540652e+09   56765.49
81  0.000834  0.000836  0.000833  0.000835  1.540656e+09   45256.79
82  0.000835  0.000836  0.000833  0.000834  1.540660e+09   37450.08
83  0.000835  0.000836  0.000834  0.000836  1.540663e+09   34018.70
84  0.000835  0.000836  0.000834  0.000835  1.540667e+09   41128.30
85  0.000835  0.000836  0.000834  0.000835  1.540670e+09   32563.74
86  0.000832  0.000835  0.000831  0.000835  1.540674e+09   34129.63
87  0.000834  0.000834  0.000830  0.000832  1.540678e+09   43547.09
88  0.000833  0.000834  0.000832  0.000833  1.540681e+09   27858.57
```

Let's plot it

In [8]:
```
#this is useless for the bot itself, but its helpful for us weak humans to see

%matplotlib notebook
import matplotlib.pyplot as plt
import talib as ta
#import tulipy as ti #some TA is calculated wrong in talib, e.g. the stochastic
#from mpl_finance import candlestick_ohlc


ohlcv['macd'], ohlcv['macdSignal'], ohlcv['macdHist'] = ta.MACD(ohlcv.Close.val
ohlcv.fillna(0,inplace=True)

fig = plt.figure()
ax1 = plt.subplot2grid((8,1), (0,0), rowspan=5, colspan=1)
ax2 = plt.subplot2grid((8,1), (6,0), rowspan=2, colspan=1)#,sharex=ax1)

ax1.plot(ohlcv['Opentime'],ohlcv['High'],label='High')
ax1.plot(ohlcv['Opentime'],ohlcv['Close'],label='Close')
ax1.plot(ohlcv['Opentime'],ohlcv['Low'],label='Low')
ax1.legend(loc='upper left')

#instead of the ax1 lines, we can plot it as candlesticks with the line below,
#but this requires converting the timestamps, so it doesn't work here :P
#candlestick_ohlc(ax1, ohlcv.values, width=1.0, colorup='#77d879', colordown='#

ax2.plot(ohlcv['Opentime'],ohlcv['macd'],label='macd')
ax2.plot(ohlcv['Opentime'],ohlcv['macdSignal'],label='macdSignal')
ax2.plot(ohlcv['Opentime'],ohlcv['macdHist'],label='macdHist')
ax2.legend(loc='upper left')

```



```
Out[8]: <matplotlib.legend.Legend at 0x1abae41e828>
```

In [14]:

```
#we can also use the Rest queries for gathering data for a large amount of mark
import time

timeframe = '1h'; limit=100
exchanges = ['binance','bitmex','bittrex','bitfinex','cryptopia'] #and potentia

#these two loops goes through every market in every exchange listed in 'exchang
for id in exchanges:
    print(id)
    exchange = getattr(ccxt, str(id))({})

    for market in exchange.fetch_markets():
        symbol = market['symbol']
        print(symbol)
        candles = exchange.fetch_ohlcv(symbol, timeframe, limit=10)
        pprint (candles)

        #have to wait a short while to stay within the api quiery limit
        time.sleep (exchange.rateLimit / 10000)
```

```
binance
ETH/BTC
[[1540688400000, 0.031433, 0.03145, 0.031391, 0.031427, 3261.035],
 [1540692000000, 0.031428, 0.031457, 0.031391, 0.03144, 2813.215],
 [1540695600000, 0.031439, 0.0315, 0.031391, 0.031455, 2592.042],
 [1540699200000, 0.03145, 0.031473, 0.03135, 0.031429, 3395.507],
 [1540702800000, 0.031427, 0.031465, 0.0314, 0.031429, 2259.954],
 [1540706400000, 0.031428, 0.031468, 0.031405, 0.031431, 2960.773],
 [1540710000000, 0.031422, 0.031468, 0.031401, 0.031434, 2902.199],
 [1540713600000, 0.031434, 0.031485, 0.031411, 0.031475, 4082.144],
 [1540717200000, 0.031477, 0.03149, 0.031448, 0.031462, 4264.695],
 [1540720800000, 0.031463, 0.031496, 0.03146, 0.031476, 2671.832]]
LTC/BTC
[[1540688400000, 0.008039, 0.008041, 0.00802, 0.00802, 1358.35],
 [1540692000000, 0.008025, 0.00803, 0.008014, 0.008024, 1346.83],
 [1540695600000, 0.008021, 0.008033, 0.008012, 0.008023, 1911.35],
 [1540699200000, 0.008026, 0.008028, 0.008007, 0.008024, 2308.45],
 [1540702800000, 0.008025, 0.008057, 0.008013, 0.008053, 1845.79],
 [1540706400000, 0.008048, 0.008058, 0.008025, 0.008038, 1618.9],
 [1540710000000, 0.008038, 0.008046, 0.008018, 0.008022, 2026.54],
 [1540713600000, 0.008037, 0.008046, 0.008022, 0.008028, 1977.11],
 [1540717200000, 0.008037, 0.008054, 0.008021, 0.008032, 2216.95],
 [1540720800000, 0.008025, 0.008039, 0.008018, 0.008026, 1499.07]]
BNB/BTC
[[1540688400000, 0.0014953, 0.0014989, 0.0014869, 0.0014975, 33736.99],
 [1540692000000, 0.0014974, 0.001499, 0.0014954, 0.0014975, 26781.69],
 [1540695600000, 0.0014974, 0.001499, 0.0014954, 0.001497, 34708.33],
 [1540699200000, 0.0014978, 0.0014995, 0.0014965, 0.0014986, 25730.11],
 [1540702800000, 0.0014978, 0.0014986, 0.0014926, 0.0014958, 21351.76],
 [1540706400000, 0.0014958, 0.0014979, 0.0014919, 0.0014948, 24343.77],
 [1540710000000, 0.0014948, 0.0014956, 0.0014912, 0.0014949, 35929.33],
 [1540713600000, 0.0014945, 0.0014978, 0.0014921, 0.0014928, 42149.98],
 [1540717200000, 0.0014926, 0.0014928, 0.0014912, 0.0014924, 37082.09],
 [1540720800000, 0.0014922, 0.0014925, 0.0014885, 0.001489, 30065.31]]
NEO/BTC
[[1540688400000, 0.002487, 0.002489, 0.002475, 0.002483, 3660.93],
 [1540692000000, 0.002483, 0.00249, 0.002478, 0.002483, 2221.1],
 [1540695600000, 0.002483, 0.002493, 0.002479, 0.002488, 3158.99],
 [1540699200000, 0.002489, 0.00249, 0.002481, 0.002487, 3154.08],
 [1540702800000, 0.002487, 0.002489, 0.002481, 0.002485, 2881.9],
 [1540706400000, 0.002484, 0.002492, 0.002483, 0.002487, 3126.76],
 [1540710000000, 0.002486, 0.002493, 0.002485, 0.002489, 3154.79],
 [1540713600000, 0.00249, 0.002495, 0.002487, 0.002495, 4506.35],
 [1540717200000, 0.002493, 0.002495, 0.002487, 0.002493, 2633.24],
 [1540720800000, 0.002494, 0.002496, 0.002491, 0.002492, 2086.55]]
QTUM/ETH
[[1540688400000, 0.0198, 0.019907, 0.01975, 0.019907, 4777.84],
 [1540692000000, 0.01985, 0.019909, 0.019839, 0.019839, 114.73],
 [1540695600000, 0.019837, 0.019862, 0.019792, 0.019793, 52.07],
 [1540699200000, 0.019793, 0.019806, 0.01977, 0.019806, 44.58],
```

Websockets

In [2]:
```python
#this exact websocket code is only for binanace, many other exchanges should al
from binance.client import Client
from binance.websockets import BinanceSocketManager

def process_message(msg): #this function is run every time we get an message (i
    print(msg)

interval = '1m'; symbol = 'ETHBTC'
clientpub = Client('','', {"verify": True, "timeout": 3})
bm = BinanceSocketManager(clientpub)
#the lines above configures the websocket, and the one below starts it
conn_key = bm.start_kline_socket(symbol, process_message)
```

In [3]:
```python
#websockets run until they are stopped (binances websockets also stop after 24
bm.stop_socket(conn_key)
```

In [ ]:
```python

```

In [10]:

```
'''
What is more usefull is multiplexing multiple websockets into one connection.
Here the data stream is fed into process_m_message, which splits into parseCand
depending on which data stream the message comes from. So put the technical ana

This makes it very to trade multiple markets or exchanges at the same time!
'''

def process_m_message(msg):
    if msg['stream'] == symbol.lower()+'@depth'+str(depth):
        parseOrderBook(msg)
    elif msg['stream'] == symbol.lower()+'@kline_'+interval:
        parseKlines(msg)
    else:
        print('no message!?!'+str(msg))

def parseCandles(smsg):
    pprint(smsg)
    print('I got lots of candles!')

def parseOrderBook(smsg):
    pprint(smsg)
    print('I got a yummy orderbook')

depth = 20; interval = '1m'; symbol = 'LUNBTC'

clientpub = Client('','', {"verify": True, "timeout": 3})
bm = BinanceSocketManager(clientpub)
conn_key = bm.start_multiplex_socket([symbol.lower()+'@depth'+str(depth), symbo
bm.start()



```

```
I got a yummy orderbook
I got a yummy orderbook
I got lots of candles!
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
I got lots of candles!
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
I got lots of candles!
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
I got lots of candles!
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
I got lots of candles!
I got a yummy orderbook
I got a yummy orderbook
I got lots of candles!
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
I got a yummy orderbook
```

In [11]:
```
bm.stop_socket(conn_key)
```

```
I got a yummy orderbook
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: