In [73]:

```
import os
import pandas as pd
from decimal import *
import numpy as np
import plotly.graph_objects as go
import plotly.express as px
from plotly import subplots
from datetime import datetime
```

In [74]:

```
from alpha_vantage.timeseries import TimeSeries
```

In [75]:

```
api_key = os.environ["MY_API_KEY"]
```

In [76]:

```
class ScriptData:

    def __init__(self):
        self.stock_data = {}
        self.ts = TimeSeries(key=api_key)

    def __getitem__(self, stock):
        return self.stock_data[stock]

    def __setitem__(self, stock, df):
        self.stock_data[stock] = df

    def __contains__(self, stock):
        return stock in self.stock_data

    #fetch US Stock data in dictionary format point(a)
    def fetch_intraday_data(self, stock):
        df_data, meta_data = self.ts.get_intraday(stock)
#         self.stock_data[f"{stock}"] = df_data
        self.__setitem__(stock, df_data)
        return df_data, stock

    #Converts fetched intraday data (in point a.) as a pandas DataFrame
    def convert_intraday_data(self, stock):
        df = self.__getitem__(stock)
        df = pd.DataFrame(df).transpose().reset_index()
        df.columns = ['timestamp', 'open', 'high', 'low', 'close', 'volume']

        df = df.astype({'timestamp': 'datetime64', 'open': 'float64', 'high':'float64',
        print(df.dtypes)
#         self.stock_data[f"{stock}"]=df
        self.__setitem__(stock, df)

        return df


#function: Moving Average of the ‘close’ column in ‘df’ of specified timeperiod
def indicator1(df, timeperiod):
    ma_df = pd.DataFrame()
    ma_df['timestamp'] = df['timestamp']
    ma_df[f'MA{timeperiod}'] = df['close'].rolling(timeperiod).mean()

    return ma_df
```

In [77]:

```
script_data = ScriptData()
```

In [78]:

```
script_data.fetch_intraday_data('GOOGL')
script_data.convert_intraday_data('GOOGL')
```

```
timestamp    datetime64[ns]
open                float64
high                float64
low                 float64
close               float64
volume                int64
dtype: object
```

Out[78]:

	timestamp	open	high	low	close	volume
0	2022-12-30 20:00:00	88.520	88.570	88.480	88.5500	1828
1	2022-12-30 19:45:00	88.460	88.500	88.450	88.4500	3712
2	2022-12-30 19:30:00	88.480	88.480	88.480	88.4800	244
3	2022-12-30 19:15:00	88.460	88.460	88.450	88.4500	676
4	2022-12-30 19:00:00	88.420	88.420	88.400	88.4000	1984
...	...	...	...	...	...	...
95	2022-12-29 12:00:00	88.345	88.630	88.335	88.5000	578660
96	2022-12-29 11:45:00	88.130	88.395	88.130	88.3484	549434
97	2022-12-29 11:30:00	88.080	88.180	88.005	88.1250	727932
98	2022-12-29 11:15:00	88.310	88.430	87.980	88.0800	586311
99	2022-12-29 11:00:00	88.095	88.345	88.035	88.3100	819905

100 rows × 6 columns

In [79]:

```
'GOOGL' in script_data
```

Out[79]:

```
True
```

In [80]:

```
script_data.fetch_intraday_data('AAPL')
script_data.convert_intraday_data('AAPL')
```

```
timestamp    datetime64[ns]
open                float64
high                float64
low                 float64
close               float64
volume                int64
dtype: object
```

Out[80]:

	timestamp	open	high	low	close	volume
0	2022-12-30 20:00:00	130.010	130.0100	129.97	129.9700	18320
1	2022-12-30 19:45:00	130.000	130.0400	129.98	130.0100	9951
2	2022-12-30 19:30:00	130.040	130.0400	130.00	130.0000	6478
3	2022-12-30 19:15:00	129.990	130.0399	129.99	130.0100	6012
4	2022-12-30 19:00:00	129.980	130.0500	129.97	130.0100	12825
...	...	...	...	...	...	...
95	2022-12-29 12:15:00	130.110	130.1800	129.87	129.9200	1757000
96	2022-12-29 12:00:00	129.902	130.4814	129.87	130.1005	2249460
97	2022-12-29 11:45:00	129.860	130.1400	129.79	129.9100	2700724
98	2022-12-29 11:30:00	129.530	130.0400	129.46	129.8500	2648622

99	2022-12-29 11:15:00	129.960	130.0600	129.44	129.5300	2649580

100 rows × 6 columns

In [81]:

```python
'AAPL' in script_data
```

Out[81]:

```
True
```

In [82]:

```python
'NVDA' in script_data
```

Out[82]:

```
False
```

In [83]:

```python
indicator1(script_data['AAPL'], 5)
```

Out[83]:

	timestamp	MA5
0	2022-12-30 20:00:00	NaN
1	2022-12-30 19:45:00	NaN
2	2022-12-30 19:30:00	NaN
3	2022-12-30 19:15:00	NaN
4	2022-12-30 19:00:00	130.00000
...	...	...
95	2022-12-29 12:15:00	129.78868
96	2022-12-29 12:00:00	129.91178
97	2022-12-29 11:45:00	129.92576
98	2022-12-29 11:30:00	129.95576
99	2022-12-29 11:15:00	129.86210

100 rows × 2 columns

In [84]:

```python
indicator1(script_data['GOOGL'], timeperiod=5)
```

Out[84]:

	timestamp	MA5
0	2022-12-30 20:00:00	NaN
1	2022-12-30 19:45:00	NaN
2	2022-12-30 19:30:00	NaN
3	2022-12-30 19:15:00	NaN
4	2022-12-30 19:00:00	88.46600
...	...	...
95	2022-12-29 12:00:00	88.49502
96	2022-12-29 11:45:00	88.49070
97	2022-12-29 11:30:00	88.44370
98	2022-12-29 11:15:00	88.31968
99	2022-12-29 11:00:00	88.27268

100 rows × 2 columns

In [85]:

```
class Strategy:


    def __init__(self, stock):
        self.stock = stock
        self.df = pd.DataFrame()

    '''Fetch intraday historical data using ScriptData class.
    Compute indicator data on ‘close’ of ‘df’ using indicator1 function'''
    def get_script_data(self):
        self.script_data = ScriptData()
        self.script_data.fetch_intraday_data(self.stock)
        self.script_data.convert_intraday_data(self.stock)
        self.df = self.script_data[self.stock]
        self.timperiod = 5
        self.indicator = indicator1(self.script_data[self.stock], self.timperiod)
        self.df['indicator'] = self.indicator[f'MA{self.timperiod}']

    #Generate a pandas DataFrame
    def get_signals(self):
        df = self.df
        df['position'] =  np.where(df['indicator']> df['close'], 1, 0)
        df['signal'] = df['position'].diff()
        df['trade_signal'] = df['signal'].replace(1.0, 'BUY').replace(-1.0, 'SELL').repl
        df = self.df.dropna()
        return df[['timestamp','trade_signal']]


    #candlestick chart of ‘df and ‘indicator’ using ‘pyalgotrading’, plotly
    def plot(self):
        df = self.df
        fig = px.line(df, y=["close", 'indicator'])
        fig.show()
        fig = go.Figure(data=[go.Candlestick(
                open=df['open'],
                high=df['high'],
                low=df['low'],
                close=df['close'])])
        fig.add_trace(go.Scatter(y=df['indicator'],
                    mode='lines',
                    name='moving_avg'))

        fig.show()
```

In [86]:

```
strategy = Strategy('NVDA')
```

In [87]:

```
strategy.get_script_data()
```

```
timestamp    datetime64[ns]
open                float64
high                float64
low                 float64
close               float64
volume                int64
dtype: object
```

In [88]:

```
strategy.get_signals()
```

Out[88]:

	timestamp	trade_signal
4	2022-12-30 19:00:00	BUY

10	2022-12-30 17:30:00	SELL
12	2022-12-30 17:00:00	BUY
14	2022-12-30 16:30:00	SELL
15	2022-12-30 16:15:00	BUY
24	2022-12-30 14:00:00	SELL
26	2022-12-30 13:30:00	BUY
27	2022-12-30 13:15:00	SELL
30	2022-12-30 12:30:00	BUY
37	2022-12-30 10:45:00	SELL
39	2022-12-30 10:15:00	BUY
40	2022-12-30 10:00:00	SELL
42	2022-12-30 09:30:00	BUY
47	2022-12-30 08:15:00	SELL
59	2022-12-30 05:15:00	BUY
64	2022-12-29 20:00:00	SELL
71	2022-12-29 18:15:00	BUY
72	2022-12-29 18:00:00	SELL
74	2022-12-29 17:30:00	BUY
76	2022-12-29 17:00:00	SELL
77	2022-12-29 16:45:00	BUY
78	2022-12-29 16:30:00	SELL
79	2022-12-29 16:15:00	BUY
81	2022-12-29 15:45:00	SELL
83	2022-12-29 15:15:00	BUY
87	2022-12-29 14:15:00	SELL
91	2022-12-29 13:15:00	BUY
92	2022-12-29 13:00:00	SELL
93	2022-12-29 12:45:00	BUY
96	2022-12-29 12:00:00	SELL
97	2022-12-29 11:45:00	BUY

In [89]:

```
strategy.plot()
```