In [387]:

```
import pandas as pd
import numpy as np

from alpha_vantage.timeseries import TimeSeries

from pyalgotrading.utils import func
from pyalgotrading.constants import PlotType


class ScriptData:


    def __init__(self):
        self._dict = {}
        self.api_key = 'H3UKOVC7728S4FJF'       # key of alpha vantage api
        self.output_format ='pandas'            #output format
        self.interval = '5min'                  # interval for market data

    def __getitem__(self, script):
        if self.__contains__(script):
            return self._dict[script]
        return "NO Key Found"


    def __setitem__(self, script, val):
            self._dict[script] = val

    def __contains__(self, script):

        if script in self._dict :
            return True
        else
            return False


    def fetch_intraday_data(self, script):
        ts = TimeSeries(key= self.api_key, output_format=self.output_format)
        #calling alpha vantage api

        data, meta_data = ts.get_intraday(symbol=script,interval=self.interval, outputsiz
        self.__setitem__(script, data)

        ## set the data in _dict (key = script)


    def convert_intraday_data(self, script):
        data = self.__getitem__(script)
        #getting data from _dict

        df = pd.DataFrame(data).reset_index(level=0)
        # creating panda dataframe and reset the index

        df.rename(columns = {'date':'timestamp',
                             '1. open':'open',
                             '2. high':'high',
                             '3. low':'low',
                             '4. close':'close',
                             '5. volume':'volume'},
                              inplace = True )
        #renamed the column names

        self.__setitem__(script, df)
        # update the df in _dict
```

In [388]:

```
script_data = ScriptData()
```

In [389]:

```
script_data.fetch_intraday_data("GOOGL")
script_data.convert_intraday_data("GOOGL")
script_data['GOOGL']
```

Out[389]:

	timestamp	open	high	low	close	volume
0	2023-02-14 20:00:00	94.18	94.1800	94.11	94.12	9174.0
1	2023-02-14 19:55:00	94.23	94.2300	94.18	94.18	2882.0
2	2023-02-14 19:50:00	94.24	94.2400	93.50	93.50	15924.0
3	2023-02-14 19:45:00	94.26	94.2899	94.20	94.20	5486.0
4	2023-02-14 19:40:00	94.26	94.3200	94.25	94.32	5043.0
...	...	...	...	...	...	...
3853	2023-01-17 04:25:00	91.82	91.8200	91.60	91.72	3784.0
3854	2023-01-17 04:20:00	91.86	91.8800	91.86	91.88	1980.0
3855	2023-01-17 04:15:00	91.90	91.9700	91.87	91.87	3610.0
3856	2023-01-17 04:10:00	91.82	91.9300	91.80	91.90	2016.0
3857	2023-01-17 04:05:00	92.02	92.1000	91.71	91.79	6551.0

3858 rows × 6 columns

In [ ]:

In [390]:

```
script_data.fetch_intraday_data("AAPL")
script_data.convert_intraday_data("AAPL")
script_data['AAPL']
```

Out[390]:

	timestamp	open	high	low	close	volume
0	2023-02-14 20:00:00	152.7900	152.8500	152.7600	152.8400	9429.0
1	2023-02-14 19:55:00	152.7500	152.7700	152.7300	152.7700	3596.0
2	2023-02-14 19:50:00	152.7500	152.7600	152.7000	152.7100	4770.0
3	2023-02-14 19:45:00	152.8400	152.8700	152.7500	152.7600	7379.0
4	2023-02-14 19:40:00	152.8700	152.8700	152.8400	152.8400	1795.0
...	...	...	...	...	...	...
3961	2023-01-17 04:25:00	134.1557	134.1856	133.9460	133.9959	4045.0
3962	2023-01-17 04:20:00	134.2256	134.2555	134.1756	134.2555	2752.0
3963	2023-01-17 04:15:00	134.3654	134.3654	134.2755	134.2755	4748.0
3964	2023-01-17 04:10:00	134.3154	134.3454	134.2955	134.3454	3404.0
3965	2023-01-17 04:05:00	134.3254	135.4537	134.2256	134.2755	11471.0

3966 rows × 6 columns

In [391]:

```
'GOOGL' in script_data
```

Out[391]:

```
True
```

In [392]:

```
'AAPL' in script_data
```

Out[392]:

```
True
```

In [393]:

```
'NVDA' in script_data
```

Out[393]:

```
False
```

In [394]:

```
def indicator1(df, timeperiod):

    indicator_df = pd.DataFrame({
                        "timestamp":df["timestamp"],
                        "indicator":df['close'].rolling(window=timeperiod).mean()
                         })

    """ creating new dataframe with 2 colums and calculating
        moving average of close column
    """
    return indicator_df
```

In [395]:

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

Out[395]:

	timestamp	indicator
0	2023-02-14 20:00:00	NaN
1	2023-02-14 19:55:00	NaN
2	2023-02-14 19:50:00	NaN
3	2023-02-14 19:45:00	NaN
4	2023-02-14 19:40:00	94.064
...	...	...
3853	2023-01-17 04:25:00	91.700
3854	2023-01-17 04:20:00	91.730
3855	2023-01-17 04:15:00	91.772
3856	2023-01-17 04:10:00	91.820
3857	2023-01-17 04:05:00	91.832

3858 rows × 2 columns

In [396]:

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

Out[396]:

	timestamp	indicator
0	2023-02-14 20:00:00	NaN
1	2023-02-14 19:55:00	NaN
2	2023-02-14 19:50:00	NaN
3	2023-02-14 19:45:00	NaN
4	2023-02-14 19:40:00	152.78400
...	...	...
3961	2023-01-17 04:25:00	134.11174
3962	2023-01-17 04:20:00	134.11374
3963	2023-01-17 04:15:00	134.13770
3964	2023-01-17 04:10:00	134.17964
3965	2023-01-17 04:05:00	134.22956

3966 rows × 2 columns

In [416]:

```python
class Strategy:
    def __init__(self, script):
        self.script = script
        self.timeperiod = 5


    def get_script_data(self):
        self.script_data = ScriptData()
        self.script_data.fetch_intraday_data(self.script)
        self.script_data.convert_intraday_data(self.script)
        #script_data[self.script]

    def get_signals(self):
        indicator_data = indicator1(
                             self.script_data[self.script],
                             self.timeperiod )

        self.temp_df = pd.merge(self.script_data[self.script],
                           indicator_data, on='timestamp')
        #merging df and indicator_df o timestamp and creating temp_df

        del indicator_data

        self.temp_df['Diff'] = self.temp_df.close - self.temp_df.indicator
        """ Difference of close_data and indicator data """

        self.temp_df['signal'] = np.select([((self.temp_df.Diff < 0) & (self.temp_df.Diff
                                   ((self.temp_df.Diff > 0) & (self.temp_df.Diff.shift(
                                   ['BUY', 'SELL'], 'None')
        """
             diff < 0 and next index of diff > 0 then BUY
             diff > 0 and next index of diff  < 0 then SELL
         """

        signals = self.temp_df.loc[self.temp_df['signal'].isin(['BUY','SELL'])]

        """ Checking signal = BUY or SELL and then accessing only those rows """

        signals = pd.DataFrame({'timestamp':signals['timestamp'],
                      'signal':signals['signal']}).reset_index(drop=True)

        """ created new df --> signals with 2 column -> timestamp and signal"""

        return signals




    def plot_candlestick(self):

        """ Function for plotting the candlestic chart"""
        func.plot_candlestick_chart(self.temp_df.head(100),
            PlotType.JAPANESE,
            caption='',
            hide_missing_dates=False,
            show=True,
            indicators=([{
                'name':'SMA',
                'data':self.temp_df['indicator'].head(100)}]),
            plot_indicators_separately=False,
            plot_height=500, plot_width=1000 )
```

In [417]:

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

In [418]:

```
strategy.get_script_data()
```

In [419]:

```
strategy.get_signals()
```

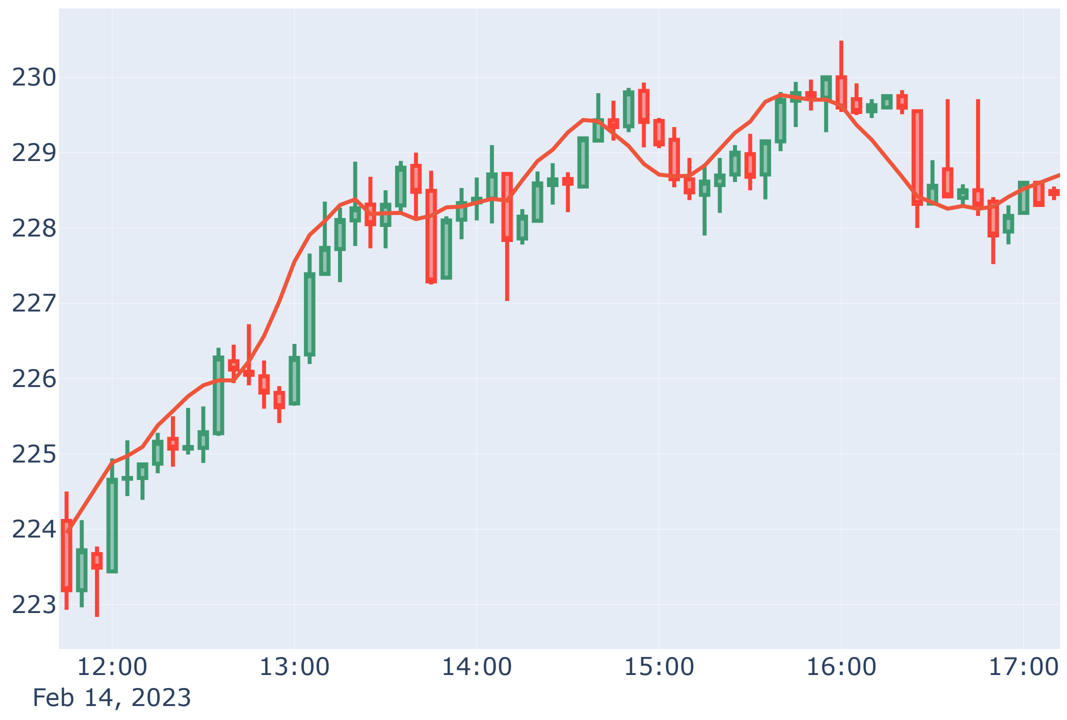
Out[419]:

	timestamp	signal
0	2023-02-14 19:25:00	BUY
1	2023-02-14 19:05:00	SELL
2	2023-02-14 17:50:00	BUY
3	2023-02-14 17:25:00	SELL
4	2023-02-14 17:20:00	BUY
...	...	...
1078	2023-01-17 06:00:00	SELL
1079	2023-01-17 05:55:00	BUY
1080	2023-01-17 04:55:00	SELL
1081	2023-01-17 04:40:00	BUY
1082	2023-01-17 04:20:00	SELL

1083 rows × 2 columns

In [420]:

```
strategy.plot_candlestick()
```



In [ ]: