

AlgoBulls Python Developer Coding Assignment

June 2023

Project: Design a simple Algorithmic Trading Strategy

Description: You need to code a simple trading strategy in a Jupyter Notebook as per the given requirements:

1. Define a Class **ScriptData** which can fetch US Stock data using [Alpha Vantage](#).
[\[Use this link to get your FREE API Key\]](#).

The class should implement the following methods:

- a. **fetch_intraday_data**: (method arguments: script)
Fetches intraday data for given "script" (Example for script: "GOOGL", "AAPL", "NVDA") and stores as it is.
- b. **convert_intraday_data**: (method arguments: script)
Converts fetched intraday data (in point a.) as a pandas DataFrame (hereafter referred as "*df*") with the following columns:
 - i. timestamp (data type: pandas.Timestamp)
 - ii. open (data type: float)
 - iii. high (data type: float)
 - iv. low (data type: float)
 - v. close (data type: float)
 - vi. volume (data type: int)
- c. **Additional methods for overloading the following operations:**
 - i. getitem
 - ii. setitem
 - iii. contains

Sample code showing how the above class will be used:

```
In [18]: 1 script_data = ScriptData()
```

```
In [19]: 1 script_data.fetch_intraday_data('GOOGL')
          2 script_data.convert_intraday_data('GOOGL')
          3 script_data['GOOGL']
```

```
Out[19]:
```

	timestamp	open	high	low	close	volume
0	2021-11-02 13:00:00	2909.620	2915.120	2898.65	2912.210	126686
1	2021-11-02 14:00:00	2911.155	2916.720	2900.54	2900.540	110407
2	2021-11-02 15:00:00	2901.310	2901.775	2887.56	2896.155	131824
3	2021-11-02 16:00:00	2896.790	2913.000	2890.37	2908.290	365383
4	2021-11-02 17:00:00	2908.650	2908.650	2905.04	2905.050	21430
...
95	2021-11-12 15:00:00	2971.240	2971.610	2963.14	2968.350	65223
96	2021-11-12 16:00:00	2968.140	2977.000	2967.65	2974.240	214453
97	2021-11-12 17:00:00	2973.560	2975.330	2972.51	2974.650	44087
98	2021-11-12 18:00:00	2973.560	2973.560	2973.56	2973.560	953
99	2021-11-12 19:00:00	2972.510	2972.510	2970.00	2970.000	635

100 rows × 6 columns

(The output data may differ for you based on which date you run this code, but the format should be the same)

```
In [20]: 1 script_data.fetch_intraday_data('AAPL')
          2 script_data.convert_intraday_data('AAPL')
          3 script_data['AAPL']
```

```
Out[20]:
```

	timestamp	open	high	low	close	volume
0	2021-11-04 17:00:00	150.7408	150.8606	150.6210	150.7608	1914822
1	2021-11-04 18:00:00	150.7907	150.8107	150.6409	150.6709	57101
2	2021-11-04 19:00:00	150.7008	150.8107	150.7008	150.7707	21546
3	2021-11-04 20:00:00	150.7707	150.7807	150.6709	150.7308	37790
4	2021-11-05 05:00:00	150.9400	151.1200	150.5500	150.7500	18607
...
95	2021-11-12 16:00:00	149.9000	150.4000	149.7500	149.9900	9358141
96	2021-11-12 17:00:00	149.9900	150.0600	149.9500	149.9900	3585851
97	2021-11-12 18:00:00	150.0000	150.0000	149.9700	149.9800	78011
98	2021-11-12 19:00:00	150.0000	150.0000	149.8700	149.9400	24201
99	2021-11-12 20:00:00	149.9600	149.9800	149.7300	149.7300	61456

100 rows × 6 columns

(The output data may differ for you based on which date you run this code, but the format should be the same)

```
In [21]: 1 'GOOGL' in script_data
```

```
Out[21]: True
```

```
In [22]: 1 'AAPL' in script_data
```

```
Out[22]: True
```

```
In [23]: 1 'NVDA' in script_data
```

```
Out[23]: False
```

2. Define a function called **indicator1**. It should take “df” and “timeperiod” (integer) as inputs and give another pandas DataFrame as an output with two columns:
 - a. *timestamp*: Same as ‘timestamp’ column in ‘df’
 - b. *indicator*: Moving Average of the ‘close’ column in ‘df’. The number of elements to be taken for a moving average is defined by ‘timeperiod’. For example, if ‘timeperiod’ is 5, then each row in this column will be an average of total 5 previous values (including current value) of the ‘close’ column.

Some sample code has been given below which shows how the above function will be used:

```
In [17]: 1 indicator1(script_data['GOOGL'], timeperiod=5)
```

```
Out[17]:
```

	timestamp	indicator
0	2021-11-02 13:00:00	NaN
1	2021-11-02 14:00:00	NaN
2	2021-11-02 15:00:00	NaN
3	2021-11-02 16:00:00	NaN
4	2021-11-02 17:00:00	2904.449
...
95	2021-11-12 15:00:00	2959.472
96	2021-11-12 16:00:00	2966.768
97	2021-11-12 17:00:00	2970.364
98	2021-11-12 18:00:00	2972.434
99	2021-11-12 19:00:00	2972.160

100 rows × 2 columns

```
In [18]: 1 indicator1(script_data['AAPL'], timeperiod=5)
```

```
Out[18]:
```

	timestamp	indicator
0	2021-11-04 17:00:00	NaN
1	2021-11-04 18:00:00	NaN
2	2021-11-04 19:00:00	NaN
3	2021-11-04 20:00:00	NaN
4	2021-11-05 05:00:00	150.73664
...
95	2021-11-12 16:00:00	149.84100
96	2021-11-12 17:00:00	149.87900
97	2021-11-12 18:00:00	149.96000
98	2021-11-12 19:00:00	149.96000
99	2021-11-12 20:00:00	149.92600

100 rows × 2 columns

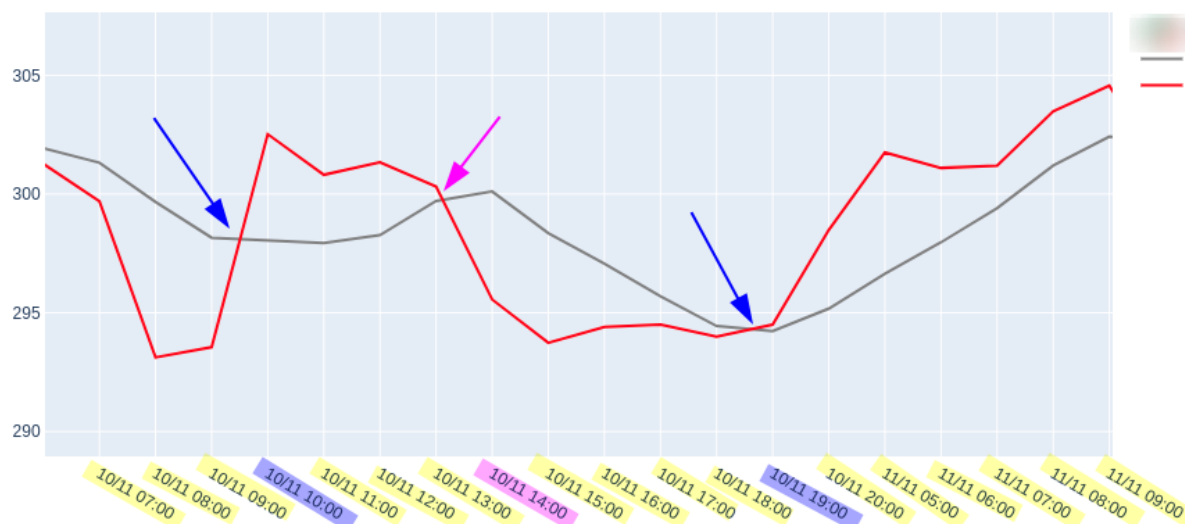
3. Define a class Strategy, which can do the following, given a script name:
 - a. Fetch intraday historical day ('df') using **ScriptData** class.
We'll refer to the 'close' column of 'df' as *close_data*.
 - b. Compute indicator data on 'close' of 'df' using **indicator1** function.
We'll refer to the 'indicator' column of this data as *indicator_data*.
 - c. Generate a pandas DataFrame called 'signals' with 2 columns:
 - i. 'timestamp': Same as 'timestamp' column in 'df'
 - ii. 'signal': This column can have the following values:
 1. BUY (When: If *indicator_data* cuts *close_data* upwards)
 2. SELL (When: If *indicator_data* cuts *close_data* downwards)
 3. NO_SIGNAL (When: If *indicator_data* and *close_data* don't cut each other)

Example of 'Cut Upwards', 'Cut Downwards', 'Do not cut each other':

As an example, for the below graph, if the RED line is *close_data* and GREY line is *indicator_data*, then:

1. The BLUE points represent the instances when *indicator_data* has cut *close_data* 'downwards'
2. The PINK points represent the instances when *indicator_data* has cut *close_data* 'upwards'
3. The YELLOW points represent when *indicator_data* and *close_data* don't cut each other.

So, there will be a SELL signal for BLUE timestamps, 'BUY' signal for PINK timestamp and 'NO_SIGNAL' for yellow timestamps.



- d. Print the 'signals' DataFrame with only those rows where the signal is either 'BUY' or 'SELL'.

Sample code showing how the above class will be used:

```
In [16]: 1 strategy = Strategy('NVDA')
```

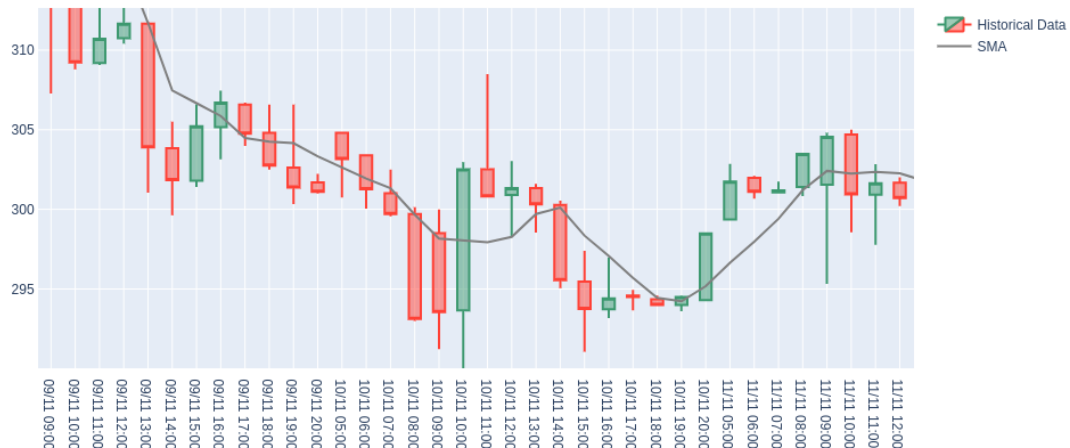
```
In [17]: 1 strategy.get_script_data()
```

```
In [24]: 1 strategy.get_signals()
```

Out[24]:

	timestamp	signal
0	2021-11-05 09:00:00	BUY
1	2021-11-05 11:00:00	SELL
2	2021-11-05 13:00:00	BUY
3	2021-11-05 20:00:00	SELL
4	2021-11-08 08:00:00	BUY
5	2021-11-08 11:00:00	SELL
6	2021-11-08 16:00:00	BUY

4. [OPTIONAL] Plot a candlestick chart of 'df' and 'indicator'. You can use the '[pyalgotrading](#)' package to do so. The chart will look like this.



You may use the following to complete the assignment:

1. Python 3.8+
2. Jupyter Notebook (latest)
3. 3rd party Python packages:
 - a. [alpha_vantage](#)
 - b. pandas
 - c. numpy
 - d. [pyalgotrading](#)

Objective:

1. Please come up with a git repo containing a Jupyter Notebook that can accommodate all the requirements.
2. The Jupyter Notebook should run seamlessly. Just calling the **Kernel -> Restart & Run All** option to do all that is necessary. There should be no errors and no unnecessary code in the notebook.
3. The dependent Python packages must be captured in a *requirements.txt* file which can be installed inside a [virtualenv](#) easily. The Jupyter Notebook should also be run after sourcing the virtualenv.

Duration:

The ideal time is 3 days. If you need extra time, please request the same with appropriate reasoning.

Submission Guidelines:

1. Please ensure the following before submission:
 - a. Ensure that the code is cleaned up as per PEP8 standards
 - b. Add sufficient comments for complex logic
 - c. Upload your code on a private GitHub repo

- d. Include a README in your repo that includes basic documentation on how to run the code
 - e. It is ok to keep pushing any number of intermediate code commits on your repo
2. For submission:
- a. Upload your Jupyter notebook, requirements.txt file and any other resources on the private GitHub repo and share it with the following GitHub ids:
algbulls-dev, akhil-jain-algbulls.
 - b. Send an email to developers@algbulls.com & akhil.jain@algbulls.com, mentioning your name and GitHub ID, requesting a review. Attach this coding assignment pdf to the mail as well.

Asking for clarification/hints:

You can send an email to developers@algbulls.com & akhil.jain@algbulls.com with your query. We will get back to you.