

# Section 1: Price Trends

In this lesson, stock price trends and the concept of moving averages are discussed.

## We'll cover the following

- Stocks data
- Stock price trend
- Moving average
  - Moving average comparisons
- Comparison

## Stocks data #

The stock data for **Systems Ltd**, **NETSOL**, **PTCL**, and **Avanceon** for the year 2018 will be used for this analysis. Any stock data has six important columns.

- **Time**: The date and time of the day
- **Open**: The price of the stock at the start of the day when the market opens
- **High**: The highest price of the stock on that day
- **Low**: The lowest price of the stock on that day
- **Close**: The price of the stock at the end of the day when the market closes
- **Volume**: The number of stocks traded that day

Let's look up the stocks data for **Systems Ltd**.




```
import pandas as pd

sys = pd.read_csv('Year_2018/SYS.csv')
#The "SYS" file name can be changed to "NETSOL", "AVN" and "PTC".
print(sys)
```



On **line 3**, the stock data for **Systems Ltd** is read in the `DataFrame` as a `sys` variable. The `SYS.csv` file contains this data. Similarly, by changing the `SYS.csv` to `NETSOL.csv`, `PTC.csv`, or `AVN.csv`, their data can also be obtained.

All the files are available for download below.

 Year\_2018.zip  

## Stock price trend #

For this analysis, the column `Close` will be of great importance. As mentioned above, this column contains the closing price of the stock of each day. So, let's visualize this using a line graph how the closing price of **Systems Ltd** stock has varied throughout the year of 2018.

```
import pandas as pd
import matplotlib.pyplot as plt

sys = pd.read_csv('Year_2018/SYS.csv')
#The "SYS" file name can be changed to "NETSOL", "AVN" and "PTC".

sys['Time'] = pd.to_datetime(sys.Time) # correct the format of date

sys = sys.set_index('Time') # Set Time column as row index

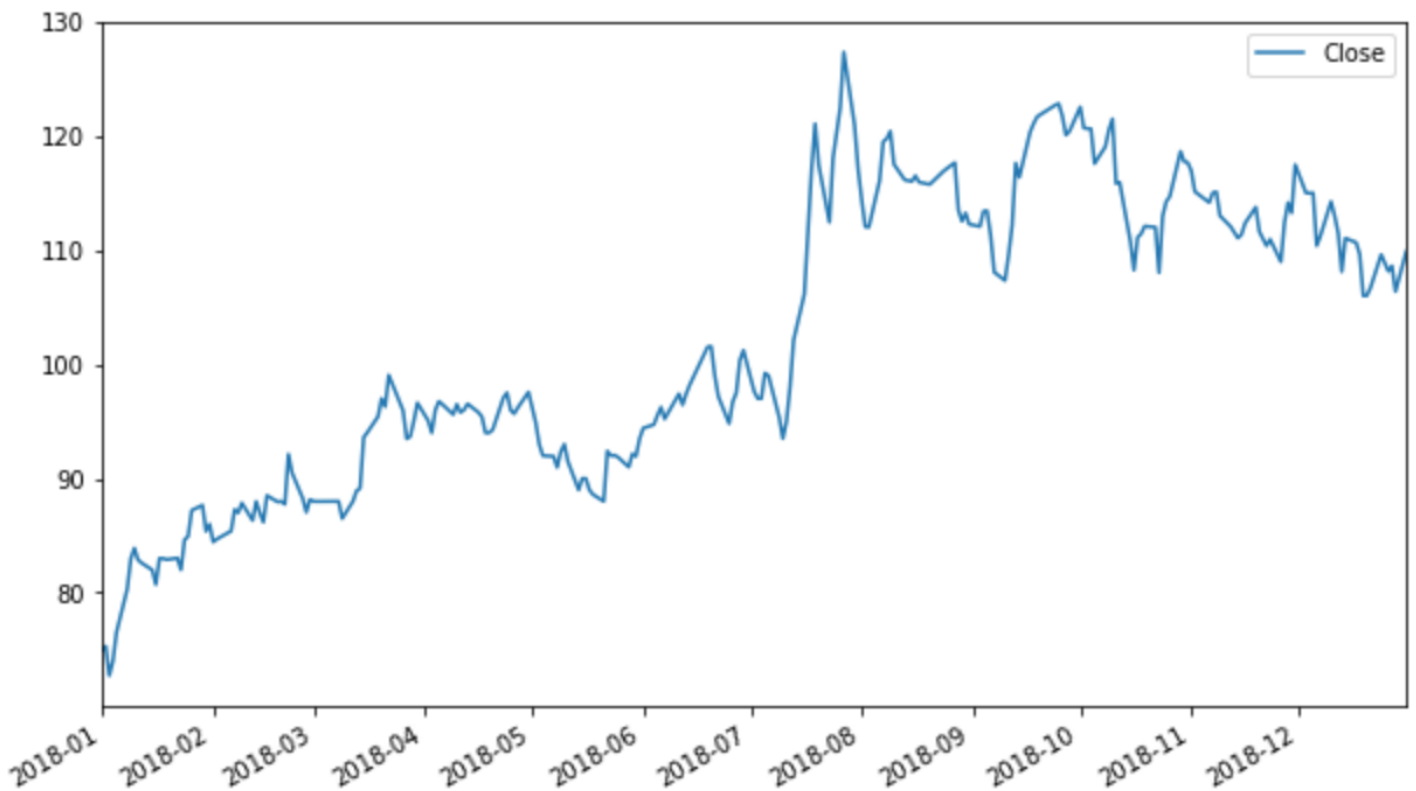
sys['Close'].plot(legend = True, figsize=(15,10))
```



On **line 7**, the date in the `Time` column is assigned to the correct date time format using the `to_datetime` function of *Pandas*.

On **line 9**, the `Time` column is assigned as the row index as every other value of every column is uniquely associated with the `Time` column. Doing this automatically plots the graph according to different time frames.

On **line 11**, the values in the `Close` column are plotted against the index, which in this case is our `Time` column. The `legend` parameter shows the `Close` variable name using the *blue line* in the plot. The `figsize` parameter sets the dimensions of the plot.



The output of the code snippet generates the above graph. The x-axis represents the time frames and the y-axis represents the price.

It can be seen that the stock price of **Systems Ltd** is generally increasing for the year 2018. The stock price at the end of the year is higher than the price at the start of the year. The trends of the other three companies can also be viewed by replacing the `SYS.csv` file with the respective company file names.

The trend graphs of other companies can be viewed in the output of the code by replacing the current company name with one of your choices.

In the graph above, the stock price goes up and down at random days. This creates a lot of noise in the trend. A more precise trend graph can be obtained by the moving average method.

## Moving average #

A moving average method is used to smooth the short term random price changes by filtering unnecessary noise. As its name suggests, it takes a rolling average or mean of all prices in a specifically defined window. A single point of a moving average graph is plotted with information on the past prices; it is also called a

trend indicator.

The working of the moving average can be understood from the following image.

Values	Moving Average
39	
42	
50	44
60	51
71	60
79	70
85	78
81	82
76	81

Moving average with window size 3

```
import pandas as pd
import matplotlib.pyplot as plt

sys = pd.read_csv('Year_2018/SYS.csv') #The "SYS" file name can be changed to "NETSOL", "AVN" and
sys['Time'] = pd.to_datetime(sys.Time) # correct the format of date
sys = sys.set_index('Time') # Set Time column as row index

days = 50 # Moving average window
col_name = "mv_avg for " + str(days) + " days" # New column to store moving average vlues

sys[col_name] = sys['Close'].rolling(days).mean() #Calculating moving average

print(sys)

sys[['Close', 'mv_avg for 50 days']].plot(figsize=(15,10)) # Plotting the closing price with moving
```

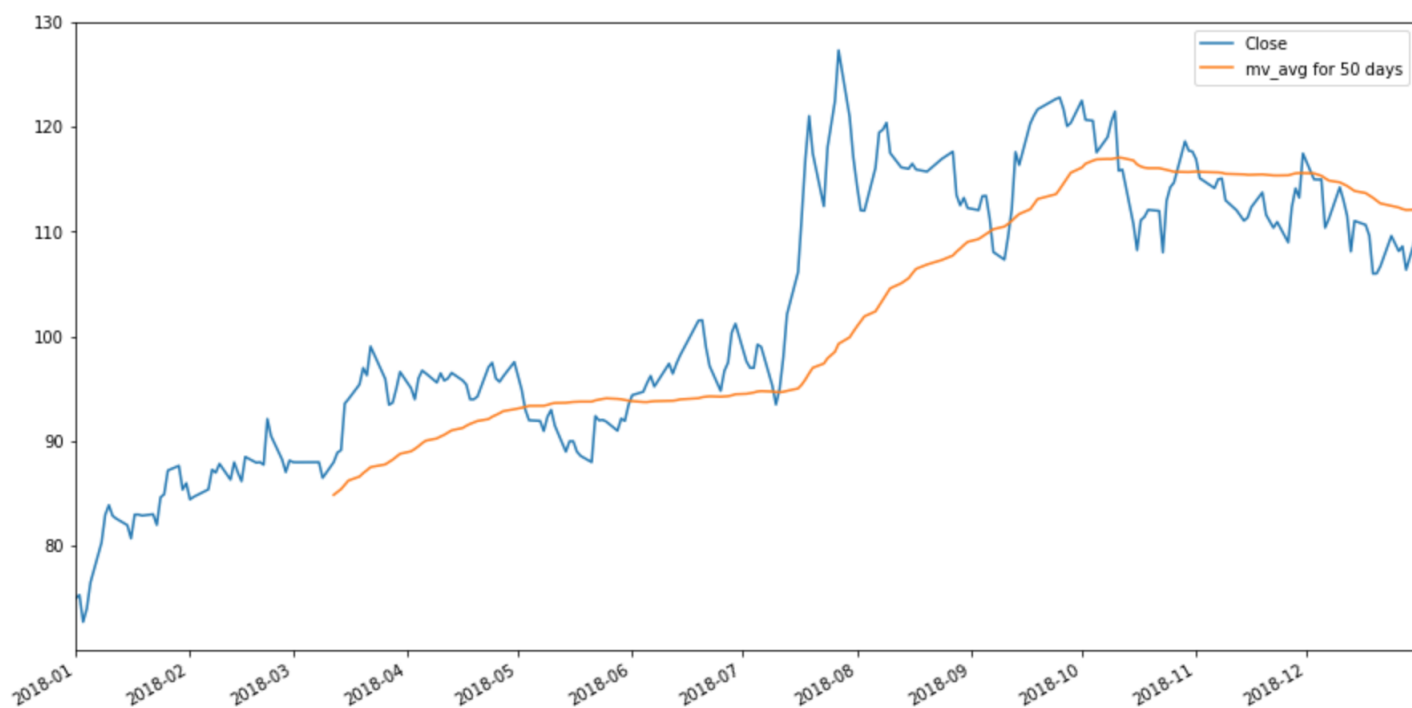
On **\*\* lines 9 & 10\*\*** the window length and the new column name are defined

On **line 8 & 9**, the window length and the new column name are defined, respectively.

On **line 11**, the `rolling` function of *pandas* takes the `days` as a parameter and applies the `mean`, or average, operation on the values of the `Close` column. The `rolling` function automatically does the moving task.

On **line 15**, the `Close` column and the new *moving average* column are plotted together for comparison.

The values of the new *moving average* column can be seen in the **output** by clicking the **right** arrow. Each value of the new column is an average of fifty-row values of the `Close` column, so there won't be any values for the first fifty rows as the first value will occur on the *fifty-first* row.



The output of the code snippet generates the above graph. The x-axis and y-axis represent the same values as before. It can be seen that, in the *moving average* graph, all the fluctuations are gone. A smooth curve represents the trend of stock behavior.

In the *moving average*, the length of the window or the number of days is of significant importance. A lower number of days is highly sensitive to fluctuation changes, while a higher number of days is less sensitive to these changes.

Keep in mind that less sensitivity offers more smooth curves.

# Moving average comparisons #

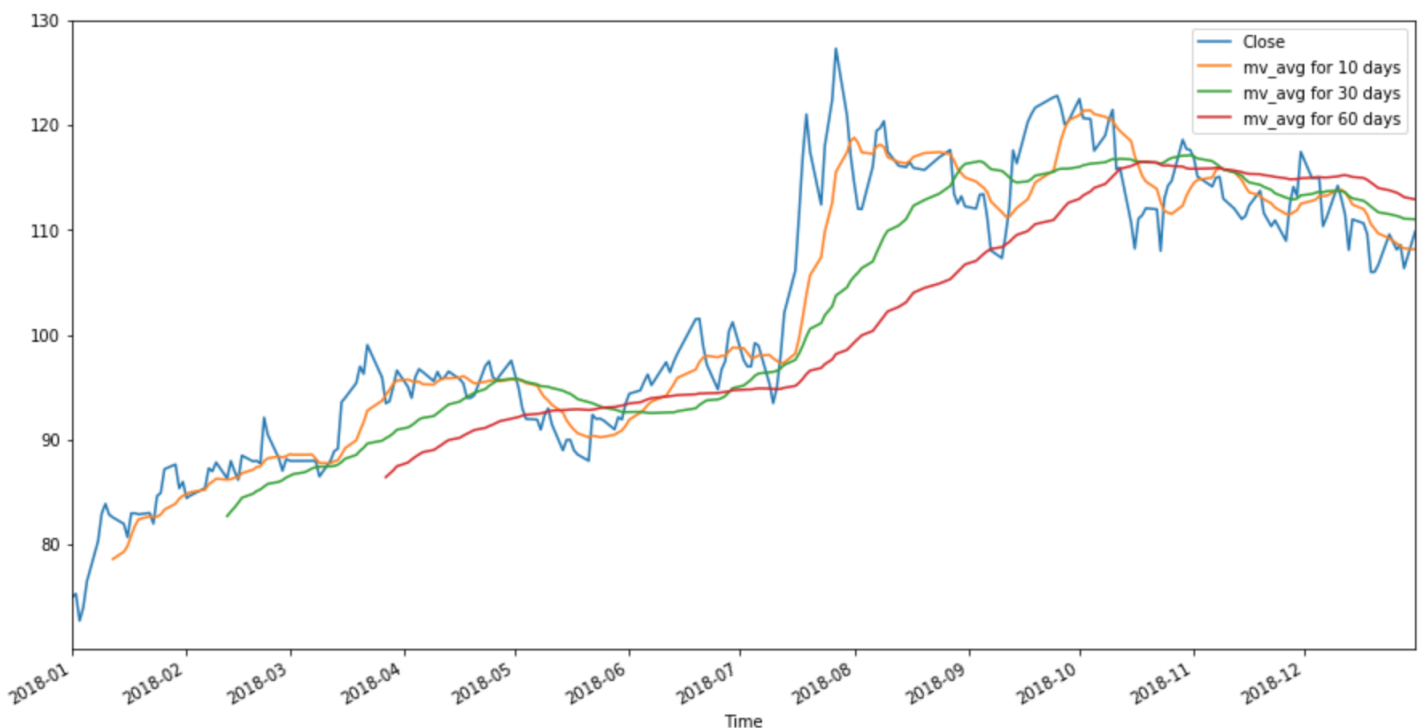
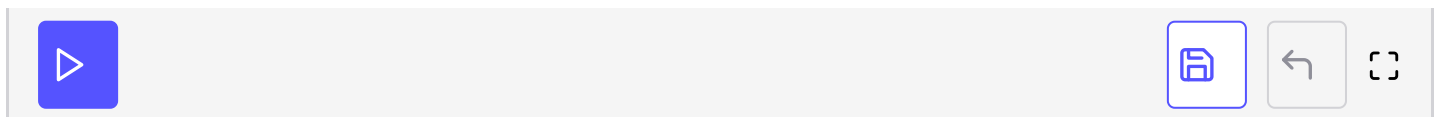
Let's visualize how lower and higher numbers of days affect the moving average curve.

```
import pandas as pd
import matplotlib.pyplot as plt

sys = pd.read_csv('Year_2018/SYS.csv') #The "SYS" file name can be changed to "NETSOL", "AVN" and
sys['Time'] = pd.to_datetime(sys.Time)
sys = sys.set_index('Time')

days = [10, 30, 60] # Multiple number of days
for day in days:
    col_name = "mv_avg for " + str(day) + " days"
    sys[col_name] = sys['Close'].rolling(day).mean()

sys[['Close', 'mv_avg for 10 days', 'mv_avg for 30 days', 'mv_avg for 60 days']].plot(subplots = F
```



The trends for different days are clearly plotted and separated by different colors. We can see that as the number of days increases the *moving average* curve becomes smoother in determining the trend. Still, it should also be noted that all *moving average* curves give a better understanding of the stock trend than the regular plot.

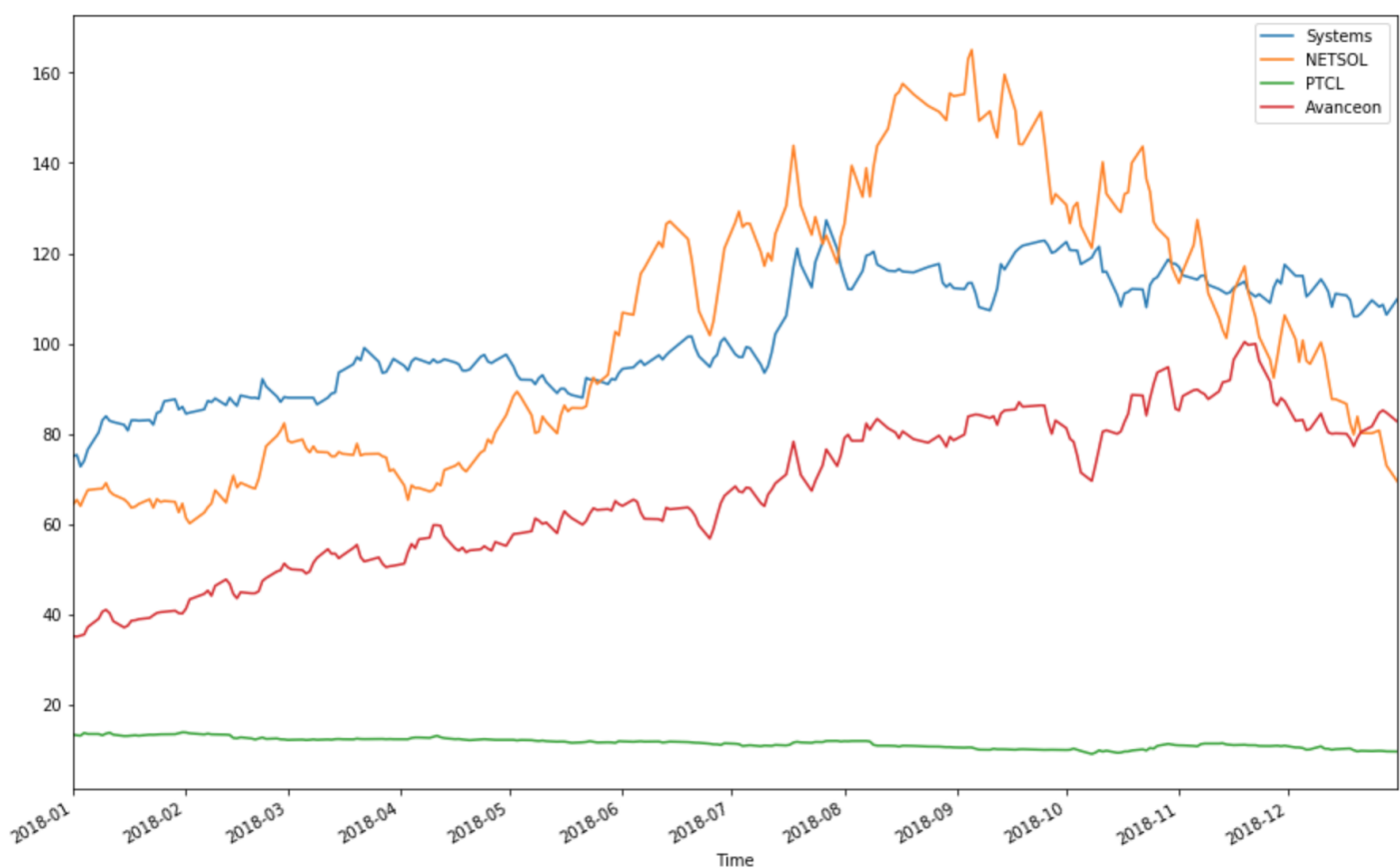
For more information on how moving averages can help in determining stock

trends, refer [here](#).

**NOTE:** All the above information can be generated for any of the other companies as well. Just replace the **SYS.csv** file name with the respective company's file name, as described above.

## Comparison #

We can determine which of the four companies stocks did well for the year 2018 by plotting the closing prices of each company's stock. The following image shows this plot.



It can be seen from the comparisons of the closing price of the four companies that 2018 was not a good year for **NETSOL** and **PTC**. The stocks of both **Systems Ltd** and **Avnceaon** did pretty good and showed steady growth.

In the next section, you will determine and plot the daily returns for these companies.

