02B-Pandas

January 20, 2018

1 Getting to know your data with Pandas

1.1 Pandas

Pandas is the Python Data Analysis Library.

Pandas is an extremely versatile tool for manipulating datasets.

It also produces high quality plots with matplotlib, and integrates nicely with other libraries that expect NumPy arrays.

The most important tool provided by Pandas is the **data frame.**

A data frame is a table in which each row and column is given a label.

Pandas DataFrames are documented at:
http://pandas.pydata.org/pandas-docs/dev/generated/pandas.DataFrame.html

1.2 Getting started

```
In [1]: import pandas as pd
        import pandas_datareader.data as web
        # Note that you might need to install pandas_datareader using the command
        #    conda install -c https://conda.anaconda.org/anaconda pandas-datareader
        from pandas import Series, DataFrame

        import matplotlib.pyplot as plt
        import matplotlib as mpl
        import seaborn as sns

        from datetime import datetime

        #pd.__version__

        %matplotlib inline
```

1.3 Fetching, storing and retrieving your data

For demonstration purposes, we'll use a library built-in to Pandas that fetches data from standard online sources, such as Yahoo! Finance.

More information on what types of data you can fetch is at: http://pandas.pydata.org/pandas-docs/stable/remote_data.html

```
In [2]: stocks = 'YELP'
        data_source = 'yahoo'
        start = datetime(2015,1,1)
        end = datetime(2015,12,31)

        yahoo_stocks = web.DataReader(stocks, data_source, start, end)

In [3]: yahoo_stocks.head()
```

Out[3]:

Date	Open	High	Low	Close	Adj Close	Volume
2015-01-02	55.459999	55.599998	54.240002	55.150002	55.150002	1664500
2015-01-05	54.540001	54.950001	52.330002	52.529999	52.529999	2023000
2015-01-06	52.549999	53.930000	50.750000	52.439999	52.439999	3762800
2015-01-07	53.320000	53.750000	51.759998	52.209999	52.209999	1548200
2015-01-08	52.590000	54.139999	51.759998	53.830002	53.830002	2015300

```
In [4]: type(yahoo_stocks)

Out[4]: pandas.core.frame.DataFrame

In [5]: yahoo_stocks.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 252 entries, 2015-01-02 to 2015-12-31
Data columns (total 6 columns):
Open         252 non-null float64
High         252 non-null float64
Low          252 non-null float64
Close        252 non-null float64
Adj Close    252 non-null float64
Volume       252 non-null int64
dtypes: float64(5), int64(1)
memory usage: 13.8 KB
```

1.3.1 Reading data from a .csv file

```
In [6]: yahoo_stocks.to_csv('yahoo_data.csv')
        print(open('yahoo_data.csv').read())

Date,Open,High,Low,Close,Adj Close,Volume
2015-01-02,55.459998999999996,55.599998,54.240002000000004,55.150002,55.150002,1664500
2015-01-05,54.540001000000004,54.950001,52.330002,52.529999,52.529999,2023000
2015-01-06,52.549999,53.93,50.75,52.439999,52.439999,3762800
2015-01-07,53.32,53.75,51.759997999999996,52.209998999999996,52.209998999999996,1548200
2015-01-08,52.59,54.139998999999996,51.759997999999996,53.830002,53.830002,2015300
2015-01-09,55.959998999999996,56.990002000000004,54.720001,56.07,56.07,6222600
2015-01-12,56.0,56.060001,53.43,54.02,54.02,2405100
```

2015-01-13,54.470001,54.799999,52.52,53.18,53.18,1952100
2015-01-14,52.799999,53.68,51.459998999999996,52.200001,52.200001,1854600
2015-01-15,53.0,53.610001000000004,50.029999,50.119999,50.119999,2647800
2015-01-16,50.18,51.490002000000004,50.029999,51.389998999999996,51.389998999999996,2183300
2015-01-20,51.650002,51.779999,50.689999,51.41,51.41,1227600
2015-01-21,51.200001,53.5,51.200001,53.41,53.41,3248100
2015-01-22,53.869999,55.279999,53.119999,54.799999,54.799999,2295400
2015-01-23,54.66,55.639998999999996,54.299999,55.189999,55.189999,1629000
2015-01-26,55.119999,55.790001000000004,54.830002,55.41,55.41,1450300
2015-01-27,56.060001,56.16,54.57,55.630001,55.630001,2410400
2015-01-28,56.150002,56.150002,52.919998,53.0,53.0,2013100
2015-01-29,52.849998,53.310001,51.41,52.93,52.93,1844100
2015-01-30,52.59,53.419998,52.049999,52.470001,52.470001,1875400
2015-02-02,52.939999,53.5,51.209998999999996,53.470001,53.470001,2105500
2015-02-03,53.830002,55.93,53.41,55.779999,55.779999,2876400
2015-02-04,55.529999,57.07,55.25,56.740002000000004,56.740002000000004,2498600
2015-02-05,57.599998,57.700001,56.080002,57.470001,57.470001,4657300
2015-02-06,47.700001,48.169998,44.860001000000004,45.110001000000004,45.110001000000004,251374
2015-02-09,44.91,45.040001000000004,42.099998,42.169998,42.169998,13079300
2015-02-10,43.830002,45.549999,43.310001,44.66,44.66,11267700
2015-02-11,45.389998999999996,46.43,44.810001,46.18,46.18,6359400
2015-02-12,46.450001,47.84,45.950001,47.630001,47.630001,4375000
2015-02-13,48.509997999999996,49.049999,47.220001,47.529999,47.529999,4713100
2015-02-17,47.439999,48.619999,47.029999,48.189999,48.189999,2390100
2015-02-18,47.939999,48.689999,47.200001,47.540001000000004,47.540001000000004,2529500
2015-02-19,47.16,47.790001000000004,46.869999,47.34,47.34,1642200
2015-02-20,47.400002,47.919998,47.099998,47.790001000000004,47.790001000000004,1688500
2015-02-23,47.549999,47.740002000000004,46.529999,47.349998,47.349998,2086000
2015-02-24,46.98,47.73,46.619999,47.290001000000004,47.290001000000004,1506500
2015-02-25,46.939999,47.450001,46.5,47.150002,47.150002,1924600
2015-02-26,48.630001,48.810001,47.560001,47.75,47.75,3059400
2015-02-27,48.32,48.439999,47.049999,48.0,48.0,2118400
2015-03-02,48.02,48.459998999999996,47.189999,47.860001000000004,47.860001000000004,1933700
2015-03-03,47.75,48.98,47.330002,48.580002,48.580002,2352100
2015-03-04,48.68,48.869999,47.310001,47.790001000000004,47.790001000000004,2218800
2015-03-05,47.689999,48.700001,47.41,47.939999,47.939999,1696900
2015-03-06,47.75,48.580002,46.919998,47.049999,47.049999,1994800
2015-03-09,46.959998999999996,46.959998999999996,45.34,45.82,45.82,2554800
2015-03-10,45.040001000000004,45.900002,44.25,45.23,45.23,2361400
2015-03-11,45.080002,46.73,44.73,45.73,45.73,2227300
2015-03-12,45.900002,46.82,45.540001000000004,46.799999,46.799999,1657400
2015-03-13,46.759997999999996,47.5,46.130001,46.450001,46.450001,2259400
2015-03-16,46.349998,46.75,45.599998,46.709998999999996,46.709998999999996,1607800
2015-03-17,46.599998,47.59,46.310001,47.220001,47.220001,1723000
2015-03-18,47.040001000000004,47.57,46.599998,46.82,46.82,2514800
2015-03-19,46.669998,47.240002000000004,44.34,45.18,45.18,9280900
2015-03-20,45.32,46.400002,44.860001000000004,44.939999,44.939999,4240400
2015-03-23,44.860001000000004,47.150002,44.740002000000004,47.029999,47.029999,3658100

2015-03-24,46.970001,47.360001000000004,46.529999,47.040001000000004,47.040001000000004,226660
2015-03-25,47.040001000000004,47.240002000000004,45.73,45.759997999999996,45.759997999999996,25
2015-03-26,45.650002,46.599998,45.310001,45.709998999999996,45.709998999999996,1692000
2015-03-27,45.82,47.349998,45.759997999999996,47.16,47.16,1885200
2015-03-30,47.099998,48.240002000000004,46.709998999999996,47.389998999999996,47.3899989999999
2015-03-31,47.029999,47.919998,46.790001000000004,47.349998,47.349998,1922500
2015-04-01,47.25,47.369999,45.09,45.5,45.5,3670000
2015-04-02,45.400002,47.5,45.32,47.139998999999996,47.139998999999996,2594600
2015-04-06,46.25,47.82,46.080002,47.310001,47.310001,1560500
2015-04-07,47.279999,48.169998,46.959998999999996,46.990002000000004,46.990002000000004,143480
2015-04-08,46.82,47.779999,46.369999,47.43,47.43,1856500
2015-04-09,47.400002,48.099998,46.779999,47.0,47.0,1184400
2015-04-10,47.049999,47.720001,46.549999,47.650002,47.650002,1407400
2015-04-13,47.720001,48.34,47.27,47.400002,47.400002,1485900
2015-04-14,47.360001000000004,47.950001,46.709998999999996,47.599998,47.599998,1692500
2015-04-15,47.619999,50.200001,47.580002,49.57,49.57,5395300
2015-04-16,49.330002,50.0,48.529999,49.369999,49.369999,2694500
2015-04-17,48.91,49.009997999999996,47.939999,48.299999,48.299999,2068500
2015-04-20,48.450001,48.490002000000004,47.830002,48.25,48.25,1112200
2015-04-21,48.419998,49.75,48.150002,49.310001,49.310001,1686900
2015-04-22,49.959998999999996,51.150002,49.790001000000004,50.450001,50.450001,2499300
2015-04-23,50.509997999999996,50.75,49.700001,50.32,50.32,1457100
2015-04-24,50.830002,51.220001,50.27,50.610001000000004,50.610001000000004,1433500
2015-04-27,50.98,52.509997999999996,50.880001,51.02,51.02,2388300
2015-04-28,51.299999,52.25,51.060001,51.220001,51.220001,2433700
2015-04-29,51.0,51.73,50.380001,51.279999,51.279999,6480400
2015-04-30,41.25,42.290001000000004,38.75,39.389998999999996,39.389998999999996,25307400
2015-05-01,39.310001,39.889998999999996,38.540001000000004,39.759997999999996,39.7599979999999
2015-05-04,39.689999,39.709998999999996,38.68,39.610001000000004,39.610001000000004,5171100
2015-05-05,39.540001000000004,39.990002000000004,38.689999,38.880001,38.880001,2500500
2015-05-06,38.799999,39.09,37.91,38.220001,38.220001,2689200
2015-05-07,38.220001,48.73,38.220001,47.009997999999996,47.009997999999996,33831600
2015-05-08,47.25,50.990002000000004,47.200001,49.93,49.93,24155600
2015-05-11,49.360001000000004,50.290001000000004,47.880001,48.619999,48.619999,10430300
2015-05-12,48.220001,49.860001000000004,48.150002,48.830002,48.830002,7587400
2015-05-13,48.830002,49.349998,47.25,47.84,47.84,5577100
2015-05-14,47.849998,48.470001,47.259997999999996,47.349998,47.349998,3539400
2015-05-15,47.360001000000004,47.57,46.68,46.889998999999996,46.889998999999996,3567700
2015-05-18,45.700001,46.619999,45.5,46.560001,46.560001,2890700
2015-05-19,46.52,48.560001,46.0,46.459998999999996,46.459998999999996,4677600
2015-05-20,46.549999,46.93,45.75,46.43,46.43,1605000
2015-05-21,46.099998,46.880001,45.849998,46.139998999999996,46.139998999999996,1929800
2015-05-22,46.0,46.830002,46.0,46.48,46.48,1373800
2015-05-26,46.130001,46.810001,45.360001000000004,45.549999,45.549999,2142300
2015-05-27,45.849998,46.68,44.830002,45.52,45.52,2544100
2015-05-28,45.099998,47.919998,44.849998,47.75,47.75,4838900
2015-05-29,47.16,48.689999,46.75,47.91,47.91,3600400
2015-06-01,47.709998999999996,48.27,46.869999,47.41,47.41,2296200

2015-06-02,47.450001,48.900002,47.310001,48.580002,48.580002,2060700
2015-06-03,48.66,48.799999,47.279999,47.560001,47.560001,2200200
2015-06-04,47.330002,47.860001000000004,46.790001000000004,47.150002,47.150002,3160400
2015-06-05,47.529999,48.5,47.099998,48.220001,48.220001,2715200
2015-06-08,47.900002,48.040001000000004,45.619999,45.669998,45.669998,3707000
2015-06-09,45.970001,46.27,45.23,45.439999,45.439999,2635900
2015-06-10,45.810001,45.810001,44.400002,44.459998999999996,44.459998999999996,3719400
2015-06-11,44.610001000000004,44.860001000000004,43.41,43.540001000000004,43.540001000000004,5
2015-06-12,43.380001,44.299999,43.25,44.040001000000004,44.040001000000004,3401900
2015-06-15,43.490002000000004,44.389998999999996,43.490002000000004,44.0,44.0,2341400
2015-06-16,43.939999,46.220001,43.830002,44.700001,44.700001,5072500
2015-06-17,44.939999,45.189999,44.18,44.599998,44.599998,1711700
2015-06-18,44.509997999999996,45.830002,44.360001000000004,45.439999,45.439999,2045600
2015-06-19,45.060001,45.459998999999996,44.75,45.119999,45.119999,2302200
2015-06-22,45.349998,46.02,44.830002,45.279999,45.279999,2115400
2015-06-23,45.509997999999996,46.400002,45.080002,46.02,46.02,1833600
2015-06-24,46.0,46.25,45.189999,45.25,45.25,1427000
2015-06-25,45.540001000000004,45.849998,44.84,44.880001,44.880001,1368800
2015-06-26,44.959998999999996,45.240002000000004,44.209998999999996,44.509997999999996,44.5099
2015-06-29,43.669998,44.32,42.32,42.470001,42.470001,2402100
2015-06-30,42.970001,43.349998,42.540001000000004,43.029999,43.029999,1834600
2015-07-01,43.360001000000004,43.490002000000004,42.049999,42.439999,42.439999,1696000
2015-07-02,42.369999,42.369999,36.099998,38.18,38.18,13264600
2015-07-06,37.709998999999996,38.34,36.650002,37.490002000000004,37.490002000000004,6536900
2015-07-07,37.310001,37.360001000000004,35.099998,36.299999,36.299999,6272400
2015-07-08,35.639998999999996,36.25,35.139998999999996,35.400002,35.400002,2428400
2015-07-09,35.919998,36.060001,34.630001,34.75,34.75,2778900
2015-07-10,35.099998,35.43,34.650002,34.73,34.73,2223300
2015-07-13,34.84,35.700001,34.650002,35.43,35.43,4069800
2015-07-14,35.43,36.040001000000004,34.73,35.990002000000004,35.990002000000004,2972400
2015-07-15,35.799999,36.080002,34.849998,35.0,35.0,2763000
2015-07-16,35.25,35.549999,34.549999,35.110001000000004,35.110001000000004,2186800
2015-07-17,35.459998999999996,35.900002,34.669998,34.959998999999996,34.959998999999996,301420
2015-07-20,33.939999,35.349998,33.349998,34.549999,34.549999,4215400
2015-07-21,34.549999,35.650002,34.5,35.57,35.57,2824500
2015-07-22,34.950001,35.290001000000004,33.889998999999996,34.439999,34.439999,2852400
2015-07-23,34.439999,35.450001,34.34,35.040001000000004,35.040001000000004,2188300
2015-07-24,35.290001000000004,35.299999,34.0,34.560001,34.560001,2094600
2015-07-27,33.98,34.32,33.529999,33.73,33.73,3470300
2015-07-28,33.060001,33.799999,32.360001000000004,33.509997999999996,33.509997999999996,852920
2015-07-29,24.940001000000002,25.5,23.66,25.059998999999998,25.059998999999998,34598900
2015-07-30,24.91,26.440001000000002,24.809998999999998,26.030001000000002,26.030001000000002,78
2015-07-31,25.690001000000002,27.48,25.690001000000002,26.4,26.4,7889400
2015-08-03,26.6,27.700001,25.559998999999998,25.92,25.92,6636900
2015-08-04,25.879998999999998,26.32,25.4,26.18,26.18,3176900
2015-08-05,26.290001,27.07,26.049999,26.129998999999998,26.129998999999998,2830200
2015-08-06,26.02,26.23,25.040001,25.379998999999998,25.379998999999998,2389000
2015-08-07,25.280001000000002,25.75,24.99,25.35,25.35,2167200

```
2015-08-10,25.6,26.459999,25.530001000000002,26.0,26.0,2139100
2015-08-11,25.65,25.969998999999998,24.91,25.35,25.35,2356000
2015-08-12,24.93,25.629998999999998,24.57,25.469998999999998,25.469998999999998,2341800
2015-08-13,25.52,25.75,24.51,24.540001,24.540001,2263600
2015-08-14,25.26,26.139999,24.91,25.77,25.77,4242700
2015-08-17,25.73,27.040001,25.5,26.639999,26.639999,4077300
2015-08-18,26.610001,26.66,25.01,25.41,25.41,4037700
2015-08-19,25.129998999999998,25.68,24.709999,25.25,25.25,2257700
2015-08-20,24.719998999999998,25.040001,24.0,24.01,24.01,2930700
2015-08-21,23.709999,23.719998999999998,22.67,23.09,23.09,5473700
2015-08-24,21.66,24.23,20.5,22.98,22.98,4882000
2015-08-25,23.870001000000002,24.15,22.719998999999998,22.75,22.75,2860500
2015-08-26,23.34,23.51,22.51,23.379998999999998,23.379998999999998,3354500
2015-08-27,24.040001,24.530001000000002,23.16,23.98,23.98,4050900
2015-08-28,23.639999,24.33,23.639999,23.959999,23.959999,2625100
2015-08-31,23.91,24.58,23.700001,24.280001000000002,24.280001000000002,2257200
2015-09-01,23.59,24.389999,23.16,23.4,23.4,3158300
2015-09-02,23.700001,24.290001,23.24,24.280001000000002,24.280001000000002,2358900
2015-09-03,24.299999,24.48,23.879998999999998,24.17,24.17,1614900
2015-09-04,23.700001,24.66,23.6,24.209999,24.209999,1894400
2015-09-08,24.610001,24.610001,23.940001000000002,23.98,23.98,3224400
2015-09-09,24.08,25.33,24.07,24.68,24.68,2660400
2015-09-10,24.73,25.5,24.6,25.120001000000002,25.120001000000002,1786900
2015-09-11,25.0,25.370001000000002,24.629998999999998,24.709999,24.709999,1702900
2015-09-14,24.77,24.940001000000002,24.17,24.469998999999998,24.469998999999998,1678700
2015-09-15,24.41,25.200001,24.299999,24.940001000000002,24.940001000000002,1316500
2015-09-16,24.809998999999998,25.93,24.73,25.290001,25.290001,3299800
2015-09-17,25.120001000000002,25.68,24.9,25.129998999999998,25.129998999999998,2011400
2015-09-18,24.74,25.440001000000002,24.549999,24.75,24.75,2556500
2015-09-21,24.860001,24.959999,24.1,24.23,24.23,2279600
2015-09-22,23.92,24.49,23.209999,23.35,23.35,2394800
2015-09-23,23.5,23.5,22.93,23.1,23.1,1655100
2015-09-24,22.469998999999998,22.67,22.0,22.58,22.58,2778600
2015-09-25,22.68,22.99,21.959999,22.17,22.17,1701300
2015-09-28,22.15,22.32,21.370001000000002,21.379998999999998,21.379998999999998,1974700
2015-09-29,21.67,21.969998999999998,21.0,21.139999,21.139999,2474400
2015-09-30,21.49,22.110001,21.219998999999998,21.66,21.66,1933600
2015-10-01,21.65,21.870001000000002,20.75,20.870001000000002,20.870001000000002,2769100
2015-10-02,20.75,22.48,20.6,22.379998999999998,22.379998999999998,2248900
2015-10-05,22.379998999999998,22.709999,22.059998999999998,22.68,22.68,1809300
2015-10-06,22.719998999999998,23.190001000000002,22.51,22.68,22.68,1704300
2015-10-07,22.879998999999998,23.530001000000002,22.459999,23.5,23.5,1527900
2015-10-08,23.459999,23.870001000000002,22.879998999999998,23.610001,23.610001,1299600
2015-10-09,23.610001,25.02,23.35,24.879998999999998,24.879998999999998,3320600
2015-10-12,24.690001000000002,24.82,22.219998999999998,22.74,22.74,4755000
2015-10-13,22.809998999999998,23.299999,22.469998999999998,22.49,22.49,1696300
2015-10-14,22.48,22.879998999999998,21.870001000000002,21.98,21.98,2223100
2015-10-15,22.200001,22.84,22.059998999999998,22.610001,22.610001,2083700
```

```
2015-10-16,22.709999,22.799999,22.040001,22.65,22.65,1720600
2015-10-19,22.51,22.84,22.309998999999998,22.6,22.6,1402900
2015-10-20,22.58,23.120001000000002,22.129998999999998,22.52,22.52,1663800
2015-10-21,22.52,22.75,22.190001000000002,22.389999,22.389999,1340800
2015-10-22,22.51,23.0,22.07,22.52,22.52,2341200
2015-10-23,22.799999,22.889999,22.110001,22.559998999999998,22.559998999999998,3188700
2015-10-26,22.6,24.799999,22.32,24.43,24.43,6890700
2015-10-27,24.299999,24.299999,22.01,22.9,22.9,6616200
2015-10-28,21.639999,22.809998999999998,21.43,22.07,22.07,9403700
2015-10-29,23.26,24.200001,22.41,22.950001,22.950001,9628800
2015-10-30,23.110001,23.15,22.0,22.25,22.25,5010000
2015-11-02,22.26,23.85,22.209999,23.799999,23.799999,6459600
2015-11-03,23.67,24.450001,23.6,24.120001000000002,24.120001000000002,2482100
2015-11-04,24.030001000000002,24.6,23.879998999999998,24.42,24.42,1698500
2015-11-05,24.5,25.559998999999998,24.32,25.049999,25.049999,2794300
2015-11-06,25.049999,25.530001000000002,24.5,25.5,25.5,2095400
2015-11-09,25.34,25.65,24.629998999999998,24.959999,24.959999,1567200
2015-11-10,24.85,25.360001,24.59,25.16,25.16,1267900
2015-11-11,25.1,25.309998999999998,24.540001,24.98,24.98,1364400
2015-11-12,25.18,27.219998999999998,24.9,25.9,25.9,5213300
2015-11-13,26.719998999999998,27.49,26.120001000000002,27.1,27.1,4976000
2015-11-16,27.0,27.59,26.469998999999998,27.440001000000002,27.440001000000002,3066700
2015-11-17,27.34,27.610001,26.860001,27.540001,27.540001,2018000
2015-11-18,27.540001,28.83,27.309998999999998,28.23,28.23,3091600
2015-11-19,28.190001000000002,28.690001000000002,27.91,28.059998999999998,28.059998999999998,14
2015-11-20,28.1,31.25,28.049999,31.209999,31.209999,6697500
2015-11-23,30.58,30.809998999999998,29.15,29.860001,29.860001,4029900
2015-11-24,29.459999,30.629998999999998,29.450001,30.01,30.01,2584500
2015-11-25,29.790001,30.540001,29.709999,30.51,30.51,1287100
2015-11-27,30.5,30.6,29.610001,30.18,30.18,1058900
2015-11-30,30.110001,30.719998999999998,29.77,30.129998999999998,30.129998999999998,2015600
2015-12-01,30.110001,30.459999,29.799999,30.309998999999998,30.309998999999998,1886000
2015-12-02,30.299999,32.470001,30.290001,31.389999,31.389999,4650300
2015-12-03,31.389999,32.240002000000004,30.48,30.629998999999998,30.629998999999998,2698900
2015-12-04,30.530001000000002,30.860001,29.32,30.450001,30.450001,2313800
2015-12-07,30.379998999999998,30.639999,29.629998999999998,30.040001,30.040001,1362300
2015-12-08,29.809998999999998,31.379998999999998,29.5,30.92,30.92,1830200
2015-12-09,30.98,31.139999,29.26,30.0,30.0,2238500
2015-12-10,30.110001,31.299999,29.99,30.83,30.83,1251800
2015-12-11,30.690001000000002,30.75,29.6,29.65,29.65,1415000
2015-12-14,29.6,29.889999,28.85,29.58,29.58,2328600
2015-12-15,29.68,30.0,26.459999,26.870001000000002,26.870001000000002,5759200
2015-12-16,26.889999,28.24,26.26,28.030001000000002,28.030001000000002,2992100
2015-12-17,28.139999,28.32,27.190001000000002,27.42,27.42,1483900
2015-12-18,27.309998999999998,27.91,26.9,27.17,27.17,1299800
2015-12-21,27.17,27.360001,26.030001000000002,26.25,26.25,1947600
2015-12-22,26.25,28.700001,26.15,27.93,27.93,2952700
2015-12-23,27.950001,28.42,27.440001000000002,28.15,28.15,1001000
```

```
2015-12-24,28.27,28.59,27.9,28.4,28.4,587400
2015-12-28,28.120001000000002,28.379998999999998,27.77,27.879998999999998,27.879998999999998,10
2015-12-29,27.950001,28.540001,27.74,28.48,28.48,1103900
2015-12-30,28.58,28.780001000000002,28.17,28.25,28.25,1068000
2015-12-31,28.1,28.969998999999998,28.02,28.799999,28.799999,1301500
```

```
In [7]: df = pd.read_csv('yahoo_data.csv')
        df
```

Out[7]:

	Date	Open	High	Low	Close	Adj Close	\
0	2015-01-02	55.459999	55.599998	54.240002	55.150002	55.150002	
1	2015-01-05	54.540001	54.950001	52.330002	52.529999	52.529999	
2	2015-01-06	52.549999	53.930000	50.750000	52.439999	52.439999	
3	2015-01-07	53.320000	53.750000	51.759998	52.209999	52.209999	
4	2015-01-08	52.590000	54.139999	51.759998	53.830002	53.830002	
5	2015-01-09	55.959999	56.990002	54.720001	56.070000	56.070000	
6	2015-01-12	56.000000	56.060001	53.430000	54.020000	54.020000	
7	2015-01-13	54.470001	54.799999	52.520000	53.180000	53.180000	
8	2015-01-14	52.799999	53.680000	51.459999	52.200001	52.200001	
9	2015-01-15	53.000000	53.610001	50.029999	50.119999	50.119999	
10	2015-01-16	50.180000	51.490002	50.029999	51.389999	51.389999	
11	2015-01-20	51.650002	51.779999	50.689999	51.410000	51.410000	
12	2015-01-21	51.200001	53.500000	51.200001	53.410000	53.410000	
13	2015-01-22	53.869999	55.279999	53.119999	54.799999	54.799999	
14	2015-01-23	54.660000	55.639999	54.299999	55.189999	55.189999	
15	2015-01-26	55.119999	55.790001	54.830002	55.410000	55.410000	
16	2015-01-27	56.060001	56.160000	54.570000	55.630001	55.630001	
17	2015-01-28	56.150002	56.150002	52.919998	53.000000	53.000000	
18	2015-01-29	52.849998	53.310001	51.410000	52.930000	52.930000	
19	2015-01-30	52.590000	53.419998	52.049999	52.470001	52.470001	
20	2015-02-02	52.939999	53.500000	51.209999	53.470001	53.470001	
21	2015-02-03	53.830002	55.930000	53.410000	55.779999	55.779999	
22	2015-02-04	55.529999	57.070000	55.250000	56.740002	56.740002	
23	2015-02-05	57.599998	57.700001	56.080002	57.470001	57.470001	
24	2015-02-06	47.700001	48.169998	44.860001	45.110001	45.110001	
25	2015-02-09	44.910000	45.040001	42.099998	42.169998	42.169998	
26	2015-02-10	43.830002	45.549999	43.310001	44.660000	44.660000	
27	2015-02-11	45.389999	46.430000	44.810001	46.180000	46.180000	
28	2015-02-12	46.450001	47.840000	45.950001	47.630001	47.630001	
29	2015-02-13	48.509998	49.049999	47.220001	47.529999	47.529999	
..	...	...	...	...	...	...	
222	2015-11-18	27.540001	28.830000	27.309999	28.230000	28.230000	
223	2015-11-19	28.190001	28.690001	27.910000	28.059999	28.059999	
224	2015-11-20	28.100000	31.250000	28.049999	31.209999	31.209999	
225	2015-11-23	30.580000	30.809999	29.150000	29.860001	29.860001	
226	2015-11-24	29.459999	30.629999	29.450001	30.010000	30.010000	

227	2015-11-25	29.790001	30.540001	29.709999	30.510000	30.510000
228	2015-11-27	30.500000	30.600000	29.610001	30.180000	30.180000
229	2015-11-30	30.110001	30.719999	29.770000	30.129999	30.129999
230	2015-12-01	30.110001	30.459999	29.799999	30.309999	30.309999
231	2015-12-02	30.299999	32.470001	30.290001	31.389999	31.389999
232	2015-12-03	31.389999	32.240002	30.480000	30.629999	30.629999
233	2015-12-04	30.530001	30.860001	29.320000	30.450001	30.450001
234	2015-12-07	30.379999	30.639999	29.629999	30.040001	30.040001
235	2015-12-08	29.809999	31.379999	29.500000	30.920000	30.920000
236	2015-12-09	30.980000	31.139999	29.260000	30.000000	30.000000
237	2015-12-10	30.110001	31.299999	29.990000	30.830000	30.830000
238	2015-12-11	30.690001	30.750000	29.600000	29.650000	29.650000
239	2015-12-14	29.600000	29.889999	28.850000	29.580000	29.580000
240	2015-12-15	29.680000	30.000000	26.459999	26.870001	26.870001
241	2015-12-16	26.889999	28.240000	26.260000	28.030001	28.030001
242	2015-12-17	28.139999	28.320000	27.190001	27.420000	27.420000
243	2015-12-18	27.309999	27.910000	26.900000	27.170000	27.170000
244	2015-12-21	27.170000	27.360001	26.030001	26.250000	26.250000
245	2015-12-22	26.250000	28.700001	26.150000	27.930000	27.930000
246	2015-12-23	27.950001	28.420000	27.440001	28.150000	28.150000
247	2015-12-24	28.270000	28.590000	27.900000	28.400000	28.400000
248	2015-12-28	28.120001	28.379999	27.770000	27.879999	27.879999
249	2015-12-29	27.950001	28.540001	27.740000	28.480000	28.480000
250	2015-12-30	28.580000	28.780001	28.170000	28.250000	28.250000
251	2015-12-31	28.100000	28.969999	28.020000	28.799999	28.799999

	Volume
0	1664500
1	2023000
2	3762800
3	1548200
4	2015300
5	6222600
6	2405100
7	1952100
8	1854600
9	2647800
10	2183300
11	1227600
12	3248100
13	2295400
14	1629000
15	1450300
16	2410400
17	2013100
18	1844100
19	1875400
20	2105500

```
21     2876400
22     2498600
23     4657300
24    25137400
25    13079300
26    11267700
27     6359400
28     4375000
29     4713100
..         ...
222    3091600
223    1487500
224    6697500
225    4029900
226    2584500
227    1287100
228    1058900
229    2015600
230    1886000
231    4650300
232    2698900
233    2313800
234    1362300
235    1830200
236    2238500
237    1251800
238    1415000
239    2328600
240    5759200
241    2992100
242    1483900
243    1299800
244    1947600
245    2952700
246    1001000
247     587400
248    1004500
249    1103900
250    1068000
251    1301500

[252 rows x 7 columns]
```

The number of rows in the DataFrame:

```
In [8]: len(df)

Out[8]: 252
```

1.4 Working with data columns

The columns or "features" in your data

```
In [9]: df.columns

Out[9]: Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')

In [10]: type(df.columns)

Out[10]: pandas.core.indexes.base.Index
```

Selecting a single column from your data

```
In [11]: df['Open']

Out[11]: 0      55.459999
         1      54.540001
         2      52.549999
         3      53.320000
         4      52.590000
         5      55.959999
         6      56.000000
         7      54.470001
         8      52.799999
         9      53.000000
         10     50.180000
         11     51.650002
         12     51.200001
         13     53.869999
         14     54.660000
         15     55.119999
         16     56.060001
         17     56.150002
         18     52.849998
         19     52.590000
         20     52.939999
         21     53.830002
         22     55.529999
         23     57.599998
         24     47.700001
         25     44.910000
         26     43.830002
         27     45.389999
         28     46.450001
         29     48.509998
                  ...
         222    27.540001
         223    28.190001
```

```
224    28.100000
225    30.580000
226    29.459999
227    29.790001
228    30.500000
229    30.110001
230    30.110001
231    30.299999
232    31.389999
233    30.530001
234    30.379999
235    29.809999
236    30.980000
237    30.110001
238    30.690001
239    29.600000
240    29.680000
241    26.889999
242    28.139999
243    27.309999
244    27.170000
245    26.250000
246    27.950001
247    28.270000
248    28.120001
249    27.950001
250    28.580000
251    28.100000
Name: Open, Length: 252, dtype: float64
```

1.4.1 Pandas types

Data frames are their own type.

The columns of a data frame are `Series` objects (like a list in many ways)

```
In [12]: type(df['Open'])

Out[12]: pandas.core.series.Series
```

Here are other ways of selecting columns from your data

```
In [13]: df.Open

Out[13]: 0      55.459999
         1      54.540001
         2      52.549999
         3      53.320000
         4      52.590000
         5      55.959999
```

```
6       56.000000
7       54.470001
8       52.799999
9       53.000000
10      50.180000
11      51.650002
12      51.200001
13      53.869999
14      54.660000
15      55.119999
16      56.060001
17      56.150002
18      52.849998
19      52.590000
20      52.939999
21      53.830002
22      55.529999
23      57.599998
24      47.700001
25      44.910000
26      43.830002
27      45.389999
28      46.450001
29      48.509998
          ...
222     27.540001
223     28.190001
224     28.100000
225     30.580000
226     29.459999
227     29.790001
228     30.500000
229     30.110001
230     30.110001
231     30.299999
232     31.389999
233     30.530001
234     30.379999
235     29.809999
236     30.980000
237     30.110001
238     30.690001
239     29.600000
240     29.680000
241     26.889999
242     28.139999
243     27.309999
244     27.170000
```

```
245    26.250000
246    27.950001
247    28.270000
248    28.120001
249    27.950001
250    28.580000
251    28.100000
Name: Open, Length: 252, dtype: float64
```

```
In [14]: df[['Open','Close']].head()
```

```
Out[14]:         Open      Close
0  55.459999  55.150002
1  54.540001  52.529999
2  52.549999  52.439999
3  53.320000  52.209999
4  52.590000  53.830002
```

```
In [15]: df.Date.head(10)
```

```
Out[15]: 0    2015-01-02
1    2015-01-05
2    2015-01-06
3    2015-01-07
4    2015-01-08
5    2015-01-09
6    2015-01-12
7    2015-01-13
8    2015-01-14
9    2015-01-15
Name: Date, dtype: object
```

```
In [16]: df.Date.tail(10)
```

```
Out[16]: 242    2015-12-17
243    2015-12-18
244    2015-12-21
245    2015-12-22
246    2015-12-23
247    2015-12-24
248    2015-12-28
249    2015-12-29
250    2015-12-30
251    2015-12-31
Name: Date, dtype: object
```

How can we access the column "Adj Close"?

```
In [17]: # df.Adj Close
```

```
In [18]: df['Adj Close'].head()

Out[18]: 0    55.150002
         1    52.529999
         2    52.439999
         3    52.209999
         4    53.830002
         Name: Adj Close, dtype: float64
```

Changing the column names:

```
In [19]: new_column_names = [x.lower().replace(' ','_') for x in df.columns]
         df.columns = new_column_names
         df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252 entries, 0 to 251
Data columns (total 7 columns):
date         252 non-null object
open         252 non-null float64
high         252 non-null float64
low          252 non-null float64
close        252 non-null float64
adj_close    252 non-null float64
volume       252 non-null int64
dtypes: float64(5), int64(1), object(1)
memory usage: 13.9+ KB
```

Now **all** columns can be accessed using the **dot** notation:

```
In [20]: df.adj_close.head()

Out[20]: 0    55.150002
         1    52.529999
         2    52.439999
         3    52.209999
         4    53.830002
         Name: adj_close, dtype: float64
```

1.5 Data Frame methods

A DataFrame object has many useful methods.

```
In [21]: df.mean()

Out[21]: open         3.728766e+01
         high         3.805464e+01
         low          3.656373e+01
```

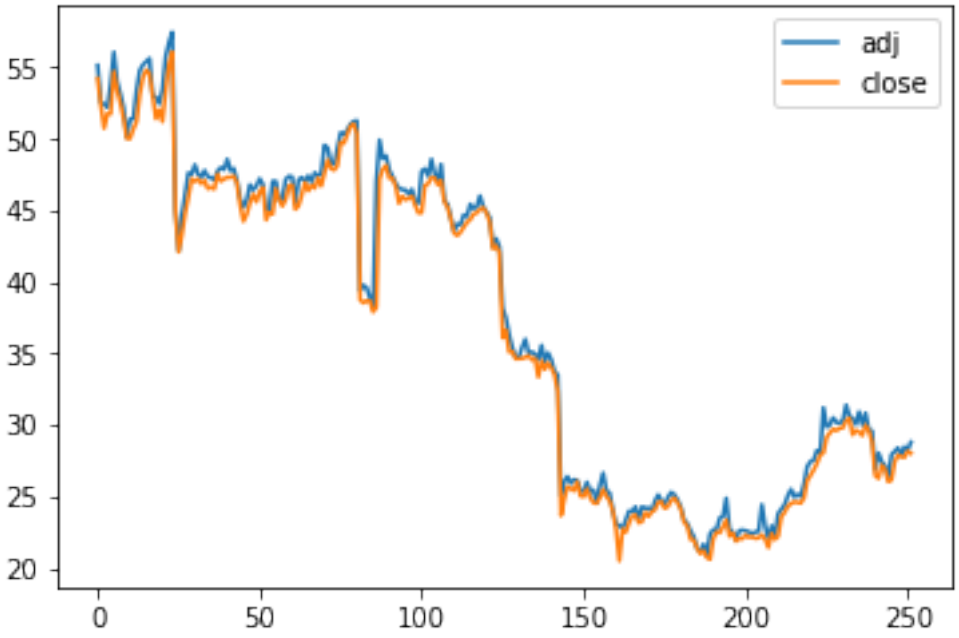
```
close        3.729917e+01
adj_close    3.729917e+01
volume       3.490978e+06
dtype: float64
```

(Notice that `mean` automatically ignores the `date` column.)

```
In [22]: df.std()

Out[22]: open         1.128093e+01
         high         1.138111e+01
         low          1.113097e+01
         close        1.125233e+01
         adj_close    1.125233e+01
         volume       4.145025e+06
         dtype: float64

In [23]: df.median()

Out[23]: open         3.796500e+01
         high         3.871500e+01
         low          3.637500e+01
         close        3.783500e+01
         adj_close    3.783500e+01
         volume       2.354050e+06
         dtype: float64

In [24]: df.open.mean()

Out[24]: 37.28765869841269

In [25]: df.high.mean()

Out[25]: 38.05464295238094
```

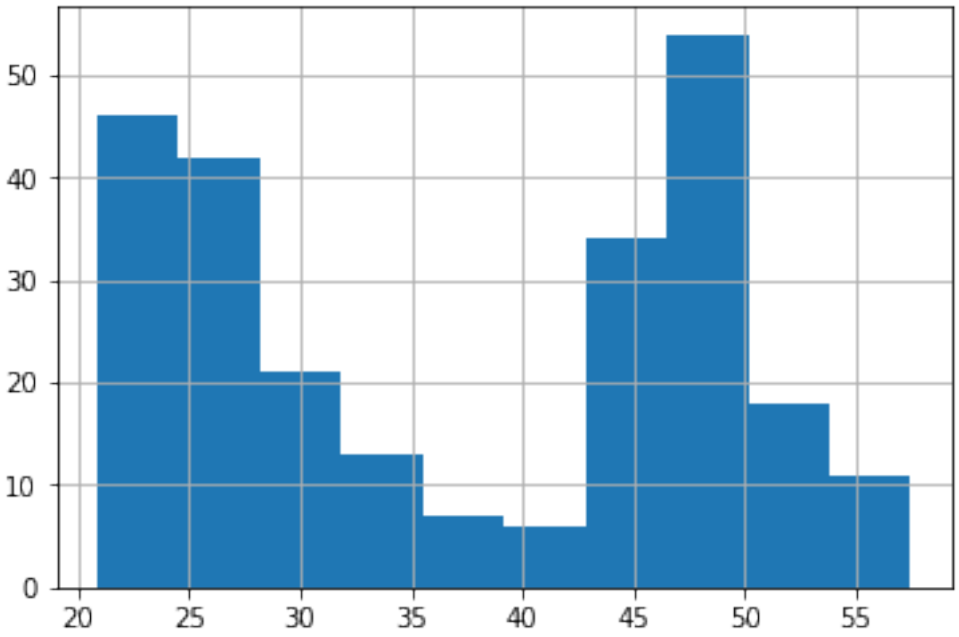
1.5.1 Plotting methods

```
In [26]: df.adj_close.plot(label='adj')
         df.low.plot(label='close')
         plt.legend(loc='best')

Out[26]: <matplotlib.legend.Legend at 0x10be384e0>
```




```
In [27]: df.adj_close.hist()
```

```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x10f0959b0>
```



1.5.2 Bulk Operations

Methods like **sum( )** and **std( )** work on entire columns.

We can run our own functions across all values in a column (or row) using **apply( )**.

```
In [28]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 252 entries, 0 to 251
Data columns (total 7 columns):
date         252 non-null object
open         252 non-null float64
high         252 non-null float64
low          252 non-null float64
close        252 non-null float64
adj_close    252 non-null float64
volume       252 non-null int64
dtypes: float64(5), int64(1), object(1)
memory usage: 13.9+ KB
```

```
In [29]: df.date.head()
```

```
Out[29]: 0    2015-01-02
         1    2015-01-05
         2    2015-01-06
         3    2015-01-07
         4    2015-01-08
         Name: date, dtype: object
```

The **values** property of the column returns a list of values for the column. Inspecting the first value reveals that these are strings with a particular format.

```
In [30]: first_date = df.date.values[0]
         first_date
```

```
Out[30]: '2015-01-02'
```

```
In [31]: datetime.strptime(first_date, "%Y-%m-%d")
```

```
Out[31]: datetime.datetime(2015, 1, 2, 0, 0)
```

```
In [32]: df.date = df.date.apply(lambda d: datetime.strptime(d, "%Y-%m-%d"))
         df.date.head()
```

```
Out[32]: 0   2015-01-02
         1   2015-01-05
         2   2015-01-06
         3   2015-01-07
         4   2015-01-08
         Name: date, dtype: datetime64[ns]
```

Each row in a DataFrame is associated with an index, which is a label that uniquely identifies a row.

The row indices so far have been auto-generated by pandas, and are simply integers starting from 0.

From now on we will use dates instead of integers for indices -- the benefits of this will show later.

Overwriting the index is as easy as assigning to the **`index`** property of the DataFrame.

```
In [33]: df.index = df.date
         df.head()
```

```
Out[33]:                  date       open       high        low      close  adj_close  \
         date
         2015-01-02 2015-01-02  55.459999  55.599998  54.240002  55.150002  55.150002
         2015-01-05 2015-01-05  54.540001  54.950001  52.330002  52.529999  52.529999
         2015-01-06 2015-01-06  52.549999  53.930000  50.750000  52.439999  52.439999
         2015-01-07 2015-01-07  53.320000  53.750000  51.759998  52.209999  52.209999
         2015-01-08 2015-01-08  52.590000  54.139999  51.759998  53.830002  53.830002

                      volume
         date
         2015-01-02  1664500
         2015-01-05  2023000
         2015-01-06  3762800
         2015-01-07  1548200
         2015-01-08  2015300
```

Now that we have made an index based on date, we can drop the original `date` column.

```
In [34]: df = df.drop(['date'],axis=1)
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 252 entries, 2015-01-02 to 2015-12-31
Data columns (total 6 columns):
open         252 non-null float64
high         252 non-null float64
low          252 non-null float64
close        252 non-null float64
adj_close    252 non-null float64
volume       252 non-null int64
dtypes: float64(5), int64(1)
memory usage: 13.8 KB
```

Now if we extract a column, it will also be indexed by date

```
In [35]: df.open.head()
```

```
Out[35]: date
         2015-01-02    55.459999
         2015-01-05    54.540001
         2015-01-06    52.549999
         2015-01-07    53.320000
         2015-01-08    52.590000
         Name: open, dtype: float64
```

1.5.3 Accessing rows of the DataFrame

So far we've seen how to access a column of the DataFrame. To access a row we use a different notation.

To access a row by its index value, use the `.loc()` method.

```
In [36]: df.loc[datetime(2015,1,23,0,0)]
```

```
Out[36]: open         5.466000e+01
         high         5.564000e+01
         low          5.430000e+01
         close        5.519000e+01
         adj_close    5.519000e+01
         volume       1.629000e+06
         Name: 2015-01-23 00:00:00, dtype: float64
```

To access a row by its sequence number (ie, like an array index), use `.iloc()` ('Integer Location')

```
In [37]: df.iloc[0]
```

```
Out[37]: open         5.546000e+01
         high         5.560000e+01
         low          5.424000e+01
         close        5.515000e+01
         adj_close    5.515000e+01
         volume       1.664500e+06
         Name: 2015-01-02 00:00:00, dtype: float64
```

To iterate over the rows, use `.iterrows()`

```
In [38]: num_positive_days = 0
         for idx, row in df.iterrows():
             if row.close > row.open:
                 num_positive_days += 1

         print("The total number of positive-gain days is {}.".format(num_positive_days))
```

```
The total number of positive-gain days is 126.
```

1.6 Filtering

It is very easy to select interesting rows from the data.

All these operations below return a new DataFrame, which itself can be treated the same way as all DataFrames we have seen so far.

```
In [39]: tmp_high = df.high > 55
         tmp_high.head()
```

```
Out[39]: date
         2015-01-02     True
         2015-01-05    False
         2015-01-06    False
         2015-01-07    False
         2015-01-08    False
         Name: high, dtype: bool
```

Summing a Boolean array is the same as counting the number of **`True`** values.

```
In [40]: sum(tmp_high)
```

```
Out[40]: 11
```

Now, let's select only the rows of **`df1`** that correspond to **`tmp_high`**

```
In [41]: df[tmp_high]
```

```
Out[41]:                 open       high        low      close  adj_close   volume
         date
         2015-01-02  55.459999  55.599998  54.240002  55.150002  55.150002  1664500
         2015-01-09  55.959999  56.990002  54.720001  56.070000  56.070000  6222600
         2015-01-12  56.000000  56.060001  53.430000  54.020000  54.020000  2405100
         2015-01-22  53.869999  55.279999  53.119999  54.799999  54.799999  2295400
         2015-01-23  54.660000  55.639999  54.299999  55.189999  55.189999  1629000
         2015-01-26  55.119999  55.790001  54.830002  55.410000  55.410000  1450300
         2015-01-27  56.060001  56.160000  54.570000  55.630001  55.630001  2410400
         2015-01-28  56.150002  56.150002  52.919998  53.000000  53.000000  2013100
         2015-02-03  53.830002  55.930000  53.410000  55.779999  55.779999  2876400
         2015-02-04  55.529999  57.070000  55.250000  56.740002  56.740002  2498600
         2015-02-05  57.599998  57.700001  56.080002  57.470001  57.470001  4657300
```

Putting it all together, we have the following commonly-used patterns:

```
In [42]: positive_days = df[df.close > df.open]
         positive_days.head()
```

```
Out[42]:                 open       high        low      close  adj_close   volume
         date
         2015-01-08  52.590000  54.139999  51.759998  53.830002  53.830002  2015300
         2015-01-09  55.959999  56.990002  54.720001  56.070000  56.070000  6222600
         2015-01-16  50.180000  51.490002  50.029999  51.389999  51.389999  2183300
         2015-01-21  51.200001  53.500000  51.200001  53.410000  53.410000  3248100
         2015-01-22  53.869999  55.279999  53.119999  54.799999  54.799999  2295400
```

```
In [43]: very_positive_days = df[df.close-df.open > 4]
         very_positive_days.head()
```

```
Out[43]:                 open   high        low      close  adj_close    volume
         date
         2015-05-07  38.220001  48.73  38.220001  47.009998  47.009998  33831600
```

1.7 Creating new columns

To create a new column, simply assign values to it. Think of the columns as a dictionary:

```
In [44]: df['profit'] = (df.open < df.close)
         df.head()
```

```
Out[44]:                 open       high        low      close  adj_close   volume  \
         date
         2015-01-02  55.459999  55.599998  54.240002  55.150002  55.150002  1664500
         2015-01-05  54.540001  54.950001  52.330002  52.529999  52.529999  2023000
         2015-01-06  52.549999  53.930000  50.750000  52.439999  52.439999  3762800
         2015-01-07  53.320000  53.750000  51.759998  52.209999  52.209999  1548200
         2015-01-08  52.590000  54.139999  51.759998  53.830002  53.830002  2015300

                     profit
         date
         2015-01-02   False
         2015-01-05   False
         2015-01-06   False
         2015-01-07   False
         2015-01-08    True
```

```
In [45]: for idx, row in df.iterrows():
             if row.close > row.open:
                 df.loc[idx,'gain']='negative'
             elif (row.open - row.close) < 1:
                 df.loc[idx,'gain']='small_gain'
             elif (row.open - row.close) < 6:
                 df.loc[idx,'gain']='medium_gain'
             else:
                 df.loc[idx,'gain']='large_gain'
         df.head()
```

```
Out[45]:                 open       high        low      close  adj_close   volume  \
         date
         2015-01-02  55.459999  55.599998  54.240002  55.150002  55.150002  1664500
         2015-01-05  54.540001  54.950001  52.330002  52.529999  52.529999  2023000
         2015-01-06  52.549999  53.930000  50.750000  52.439999  52.439999  3762800
         2015-01-07  53.320000  53.750000  51.759998  52.209999  52.209999  1548200
         2015-01-08  52.590000  54.139999  51.759998  53.830002  53.830002  2015300
```

```
              profit         gain
date
2015-01-02   False   small_gain
2015-01-05   False  medium_gain
2015-01-06   False   small_gain
2015-01-07   False  medium_gain
2015-01-08    True     negative
```

Here is another, more "functional", way to accomplish the same thing. Define a function that classifies rows, and `apply` it to each row.

```
In [46]: def namerow(row):
             if row.close > row.open:
                 return 'negative'
             elif (row.open - row.close) < 1:
                 return 'small_gain'
             elif (row.open - row.close) < 6:
                 return 'medium_gain'
             else:
                 return 'large_gain'

         df['test_column'] = df.apply(namerow, axis = 1)

In [47]: df.head()

Out[47]:                 open       high        low      close  adj_close   volume  \
         date
         2015-01-02  55.459999  55.599998  54.240002  55.150002  55.150002  1664500
         2015-01-05  54.540001  54.950001  52.330002  52.529999  52.529999  2023000
         2015-01-06  52.549999  53.930000  50.750000  52.439999  52.439999  3762800
         2015-01-07  53.320000  53.750000  51.759998  52.209999  52.209999  1548200
         2015-01-08  52.590000  54.139999  51.759998  53.830002  53.830002  2015300

                     profit         gain  test_column
         date
         2015-01-02   False   small_gain   small_gain
         2015-01-05   False  medium_gain  medium_gain
         2015-01-06   False   small_gain   small_gain
         2015-01-07   False  medium_gain  medium_gain
         2015-01-08    True     negative     negative
```

OK, point made, let's get rid of that extraneous `test_column`:

```
In [48]: df.drop('test_column', axis = 1)

Out[48]:                 open       high        low      close  adj_close    volume  \
         date
         2015-01-02  55.459999  55.599998  54.240002  55.150002  55.150002   1664500
         2015-01-05  54.540001  54.950001  52.330002  52.529999  52.529999   2023000
```

2015-01-06	52.549999	53.930000	50.750000	52.439999	52.439999	3762800
2015-01-07	53.320000	53.750000	51.759998	52.209999	52.209999	1548200
2015-01-08	52.590000	54.139999	51.759998	53.830002	53.830002	2015300
2015-01-09	55.959999	56.990002	54.720001	56.070000	56.070000	6222600
2015-01-12	56.000000	56.060001	53.430000	54.020000	54.020000	2405100
2015-01-13	54.470001	54.799999	52.520000	53.180000	53.180000	1952100
2015-01-14	52.799999	53.680000	51.459999	52.200001	52.200001	1854600
2015-01-15	53.000000	53.610001	50.029999	50.119999	50.119999	2647800
2015-01-16	50.180000	51.490002	50.029999	51.389999	51.389999	2183300
2015-01-20	51.650002	51.779999	50.689999	51.410000	51.410000	1227600
2015-01-21	51.200001	53.500000	51.200001	53.410000	53.410000	3248100
2015-01-22	53.869999	55.279999	53.119999	54.799999	54.799999	2295400
2015-01-23	54.660000	55.639999	54.299999	55.189999	55.189999	1629000
2015-01-26	55.119999	55.790001	54.830002	55.410000	55.410000	1450300
2015-01-27	56.060001	56.160000	54.570000	55.630001	55.630001	2410400
2015-01-28	56.150002	56.150002	52.919998	53.000000	53.000000	2013100
2015-01-29	52.849998	53.310001	51.410000	52.930000	52.930000	1844100
2015-01-30	52.590000	53.419998	52.049999	52.470001	52.470001	1875400
2015-02-02	52.939999	53.500000	51.209999	53.470001	53.470001	2105500
2015-02-03	53.830002	55.930000	53.410000	55.779999	55.779999	2876400
2015-02-04	55.529999	57.070000	55.250000	56.740002	56.740002	2498600
2015-02-05	57.599998	57.700001	56.080002	57.470001	57.470001	4657300
2015-02-06	47.700001	48.169998	44.860001	45.110001	45.110001	25137400
2015-02-09	44.910000	45.040001	42.099998	42.169998	42.169998	13079300
2015-02-10	43.830002	45.549999	43.310001	44.660000	44.660000	11267700
2015-02-11	45.389999	46.430000	44.810001	46.180000	46.180000	6359400
2015-02-12	46.450001	47.840000	45.950001	47.630001	47.630001	4375000
2015-02-13	48.509998	49.049999	47.220001	47.529999	47.529999	4713100
...	...	...	...	...	...	...
2015-11-18	27.540001	28.830000	27.309999	28.230000	28.230000	3091600
2015-11-19	28.190001	28.690001	27.910000	28.059999	28.059999	1487500
2015-11-20	28.100000	31.250000	28.049999	31.209999	31.209999	6697500
2015-11-23	30.580000	30.809999	29.150000	29.860001	29.860001	4029900
2015-11-24	29.459999	30.629999	29.450001	30.010000	30.010000	2584500
2015-11-25	29.790001	30.540001	29.709999	30.510000	30.510000	1287100
2015-11-27	30.500000	30.600000	29.610001	30.180000	30.180000	1058900
2015-11-30	30.110001	30.719999	29.770000	30.129999	30.129999	2015600
2015-12-01	30.110001	30.459999	29.799999	30.309999	30.309999	1886000
2015-12-02	30.299999	32.470001	30.290001	31.389999	31.389999	4650300
2015-12-03	31.389999	32.240002	30.480000	30.629999	30.629999	2698900
2015-12-04	30.530001	30.860001	29.320000	30.450001	30.450001	2313800
2015-12-07	30.379999	30.639999	29.629999	30.040001	30.040001	1362300
2015-12-08	29.809999	31.379999	29.500000	30.920000	30.920000	1830200
2015-12-09	30.980000	31.139999	29.260000	30.000000	30.000000	2238500
2015-12-10	30.110001	31.299999	29.990000	30.830000	30.830000	1251800
2015-12-11	30.690001	30.750000	29.600000	29.650000	29.650000	1415000
2015-12-14	29.600000	29.889999	28.850000	29.580000	29.580000	2328600
2015-12-15	29.680000	30.000000	26.459999	26.870001	26.870001	5759200

```
2015-12-16  26.889999  28.240000  26.260000  28.030001  28.030001   2992100
2015-12-17  28.139999  28.320000  27.190001  27.420000  27.420000   1483900
2015-12-18  27.309999  27.910000  26.900000  27.170000  27.170000   1299800
2015-12-21  27.170000  27.360001  26.030001  26.250000  26.250000   1947600
2015-12-22  26.250000  28.700001  26.150000  27.930000  27.930000   2952700
2015-12-23  27.950001  28.420000  27.440001  28.150000  28.150000   1001000
2015-12-24  28.270000  28.590000  27.900000  28.400000  28.400000    587400
2015-12-28  28.120001  28.379999  27.770000  27.879999  27.879999   1004500
2015-12-29  27.950001  28.540001  27.740000  28.480000  28.480000   1103900
2015-12-30  28.580000  28.780001  28.170000  28.250000  28.250000   1068000
2015-12-31  28.100000  28.969999  28.020000  28.799999  28.799999   1301500
```

date	profit	gain
2015-01-02	False	small_gain
2015-01-05	False	medium_gain
2015-01-06	False	small_gain
2015-01-07	False	medium_gain
2015-01-08	True	negative
2015-01-09	True	negative
2015-01-12	False	medium_gain
2015-01-13	False	medium_gain
2015-01-14	False	small_gain
2015-01-15	False	medium_gain
2015-01-16	True	negative
2015-01-20	False	small_gain
2015-01-21	True	negative
2015-01-22	True	negative
2015-01-23	True	negative
2015-01-26	True	negative
2015-01-27	False	small_gain
2015-01-28	False	medium_gain
2015-01-29	True	negative
2015-01-30	False	small_gain
2015-02-02	True	negative
2015-02-03	True	negative
2015-02-04	True	negative
2015-02-05	False	small_gain
2015-02-06	False	medium_gain
2015-02-09	False	medium_gain
2015-02-10	True	negative
2015-02-11	True	negative
2015-02-12	True	negative
2015-02-13	False	small_gain
...	...	...
2015-11-18	True	negative
2015-11-19	False	small_gain
2015-11-20	True	negative

```
2015-11-23   False   small_gain
2015-11-24    True     negative
2015-11-25    True     negative
2015-11-27   False   small_gain
2015-11-30    True     negative
2015-12-01    True     negative
2015-12-02    True     negative
2015-12-03   False   small_gain
2015-12-04   False   small_gain
2015-12-07   False   small_gain
2015-12-08    True     negative
2015-12-09   False   small_gain
2015-12-10    True     negative
2015-12-11   False  medium_gain
2015-12-14   False   small_gain
2015-12-15   False  medium_gain
2015-12-16    True     negative
2015-12-17   False   small_gain
2015-12-18   False   small_gain
2015-12-21   False   small_gain
2015-12-22    True     negative
2015-12-23    True     negative
2015-12-24    True     negative
2015-12-28   False   small_gain
2015-12-29    True     negative
2015-12-30   False   small_gain
2015-12-31    True     negative

[252 rows x 8 columns]
```

1.8 Grouping

An **extremely** powerful DataFrame method is `groupby()`.

This is entirely analagous to `GROUP BY` in SQL.*

It will group the rows of a DataFrame by the values in one (or more) columns, and let you iterate through each group.

*It's ok if you don't know what SQL's `GROUPBY` is.

Here we will look at the average gain among the categories of gains (negative, small, medium and large) we defined above and stored in column `gain`.

```
In [49]: gain_groups = df.groupby('gain')
```

Essentially, **`gain_groups`** behaves like a dictionary * whose keys are the unique values found in the `gain` column, and * whose values are DataFrames that contain only the rows having the corresponding unique values.

```
In [50]: for gain, gain_data in gain_groups:
             print(gain)
```

```
            print(gain_data.head())
            print('============================')

medium_gain
                open       high        low      close  adj_close   volume  \
date
2015-01-05  54.540001  54.950001  52.330002  52.529999  52.529999  2023000
2015-01-07  53.320000  53.750000  51.759998  52.209999  52.209999  1548200
2015-01-12  56.000000  56.060001  53.430000  54.020000  54.020000  2405100
2015-01-13  54.470001  54.799999  52.520000  53.180000  53.180000  1952100
2015-01-15  53.000000  53.610001  50.029999  50.119999  50.119999  2647800

            profit         gain  test_column
date
2015-01-05   False  medium_gain  medium_gain
2015-01-07   False  medium_gain  medium_gain
2015-01-12   False  medium_gain  medium_gain
2015-01-13   False  medium_gain  medium_gain
2015-01-15   False  medium_gain  medium_gain
============================
negative
                open       high        low      close  adj_close   volume  \
date
2015-01-08  52.590000  54.139999  51.759998  53.830002  53.830002  2015300
2015-01-09  55.959999  56.990002  54.720001  56.070000  56.070000  6222600
2015-01-16  50.180000  51.490002  50.029999  51.389999  51.389999  2183300
2015-01-21  51.200001  53.500000  51.200001  53.410000  53.410000  3248100
2015-01-22  53.869999  55.279999  53.119999  54.799999  54.799999  2295400

            profit      gain test_column
date
2015-01-08    True  negative    negative
2015-01-09    True  negative    negative
2015-01-16    True  negative    negative
2015-01-21    True  negative    negative
2015-01-22    True  negative    negative
============================
small_gain
                open       high        low      close  adj_close   volume  \
date
2015-01-02  55.459999  55.599998  54.240002  55.150002  55.150002  1664500
2015-01-06  52.549999  53.930000  50.750000  52.439999  52.439999  3762800
2015-01-14  52.799999  53.680000  51.459999  52.200001  52.200001  1854600
2015-01-20  51.650002  51.779999  50.689999  51.410000  51.410000  1227600
2015-01-27  56.060001  56.160000  54.570000  55.630001  55.630001  2410400

            profit        gain test_column
date
```

```
2015-01-02   False  small_gain  small_gain
2015-01-06   False  small_gain  small_gain
2015-01-14   False  small_gain  small_gain
2015-01-20   False  small_gain  small_gain
2015-01-27   False  small_gain  small_gain
============================
```

```
In [51]: for gain, gain_data in df.groupby("gain"):
             print('The average closing value for the {} group is {}'.format(gain,
                                                                    gain_data.close.mean()))
```

```
The average closing value for the medium_gain group is 41.008888629629624
The average closing value for the negative group is 37.213571404761886
The average closing value for the small_gain group is 36.39636363636365
```

1.9 Other Pandas Classes

A DataFrame is essentially an annotated 2-D array.
Pandas also has annotated versions of 1-D and 3-D arrays.
A 1-D array in Pandas is called a **Series**.
A 3-D array in Pandas is called a **Panel**.
To use these, read the documentation!

1.10 Comparing multiple stocks

As a last task, we will use the experience we obtained so far -- and learn some new things -- in order to compare the performance of different stocks we obtained from Yahoo finance.

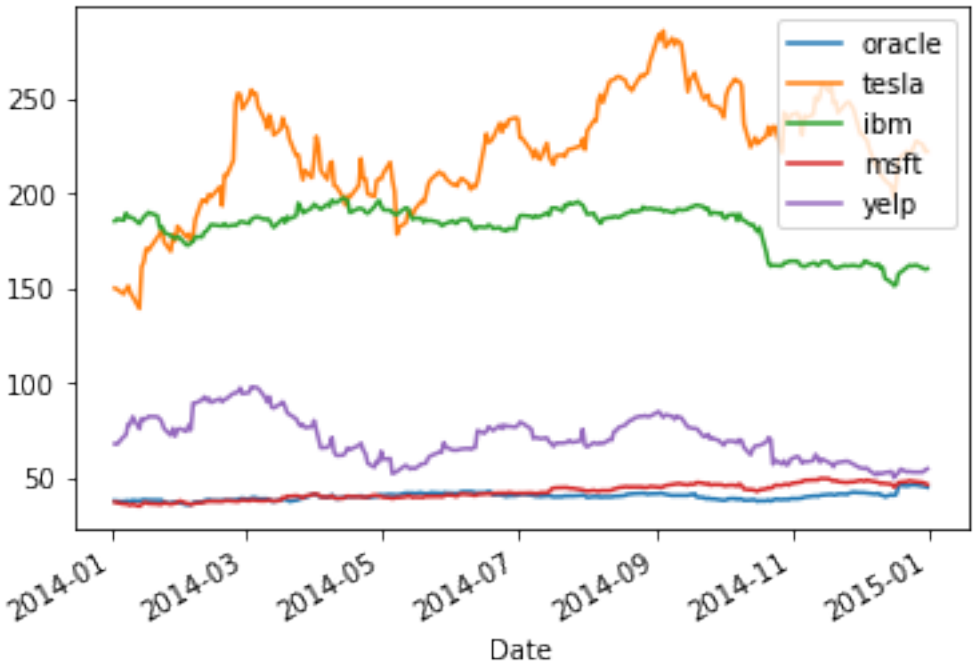
```
In [52]: stocks = ['ORCL', 'TSLA', 'IBM','YELP', 'MSFT']
         attr = 'Close'
         df = web.DataReader(stocks,
                             data_source,
                             start=datetime(2014, 1, 1),
                             end=datetime(2014, 12, 31))[attr]
         df.head()
```

Out[52]:

Date	IBM	MSFT	ORCL	TSLA	YELP
2014-12-31	160.440002	46.450001	44.970001	222.410004	54.730000
2014-12-30	160.050003	47.020000	45.340000	222.229996	54.240002
2014-12-29	160.509995	47.450001	45.610001	225.710007	53.009998
2014-12-26	162.339996	47.880001	46.099998	227.820007	52.939999
2014-12-24	161.820007	48.139999	46.230000	222.259995	53.000000

```
In [53]: df.ORCL.plot(label = 'oracle')
         df.TSLA.plot(label = 'tesla')
         df.IBM.plot(label = 'ibm')
```

```
df.MSFT.plot(label = 'msft')
df.YELP.plot(label = 'yelp')
_ = plt.legend(loc='best')
```



Next, we will calculate returns over a period of length T, defined as:

$$r(t) = \frac{f(t) - f(t-T)}{f(t-T)}$$

The returns can be computed with a simple DataFrame method **`pct_change()`**. Note that for the first T timesteps, this value is not defined (of course):

```
In [54]: rets = df.pct_change(30)
         rets.iloc[25:35]
```

Out[54]:

Date	IBM	MSFT	ORCL	TSLA	YELP
2014-11-24	NaN	NaN	NaN	NaN	NaN
2014-11-21	NaN	NaN	NaN	NaN	NaN
2014-11-20	NaN	NaN	NaN	NaN	NaN
2014-11-19	NaN	NaN	NaN	NaN	NaN
2014-11-18	NaN	NaN	NaN	NaN	NaN
2014-11-17	0.023186	0.064801	-0.084723	0.141945	0.059748
2014-11-14	0.025679	0.054445	-0.099250	0.164019	0.102876
2014-11-13	0.014205	0.045522	-0.107213	0.115148	0.096020
2014-11-12	-0.002587	0.018797	-0.129284	0.093407	0.144314
2014-11-11	0.009146	0.015164	-0.124594	0.129668	0.175660

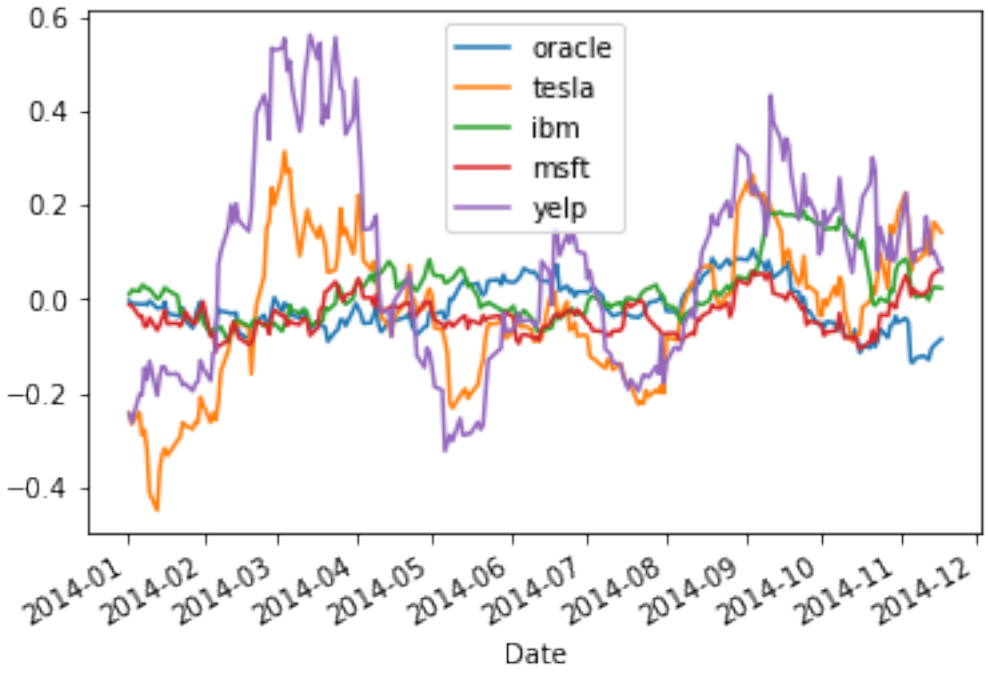
We could also compute any given row of the table directly using the formula, for row 30:

```
In [55]: (df.iloc[30] - df.iloc[0])/df.iloc[0]
```

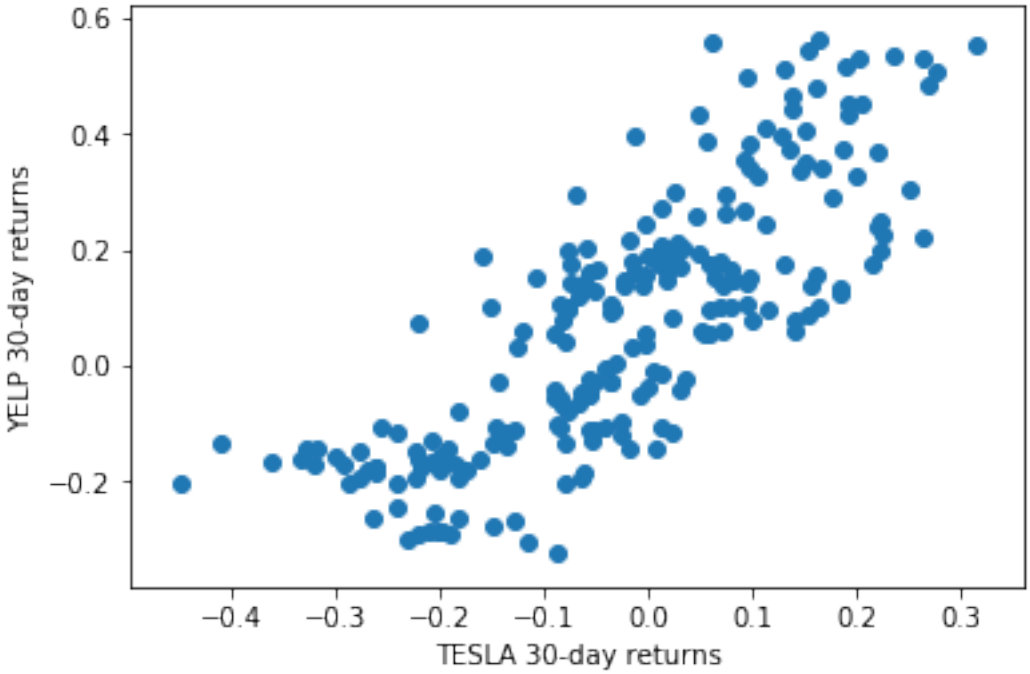
```
Out[55]: IBM     0.023186
         MSFT    0.064801
         ORCL   -0.084723
         TSLA    0.141945
         YELP    0.059748
         dtype: float64
```

Now we'll plot the timeseries of the returns of the different stocks.
Notice that the `NaN` values are gracefully dropped by the plotting function.

```
In [56]: rets.ORCL.plot(label = 'oracle')
         rets.TSLA.plot(label = 'tesla')
         rets.IBM.plot(label = 'ibm')
         rets.MSFT.plot(label = 'msft')
         rets.YELP.plot(label = 'yelp')
         _ = plt.legend(loc='best')
```




```
In [57]: plt.scatter(rets.TSLA, rets.YELP)
         plt.xlabel('TESLA 30-day returns')
         _ = plt.ylabel('YELP 30-day returns')
```



There appears to be some (fairly strong) correlation between the movement of TSLA and YELP stocks. Let's measure this.

The correlation coefficient between variables X and Y is defined as follows:

$$\text{Corr}(X,Y) = \frac{E\left[(X-\mu_X)(Y-\mu_Y)\right]}{\sigma_X \sigma_Y}$$

Pandas provides a DataFrame method to compute the correlation coefficient of all pairs of columns: `corr()`.

```
In [58]: rets.corr()
```

Out[58]:

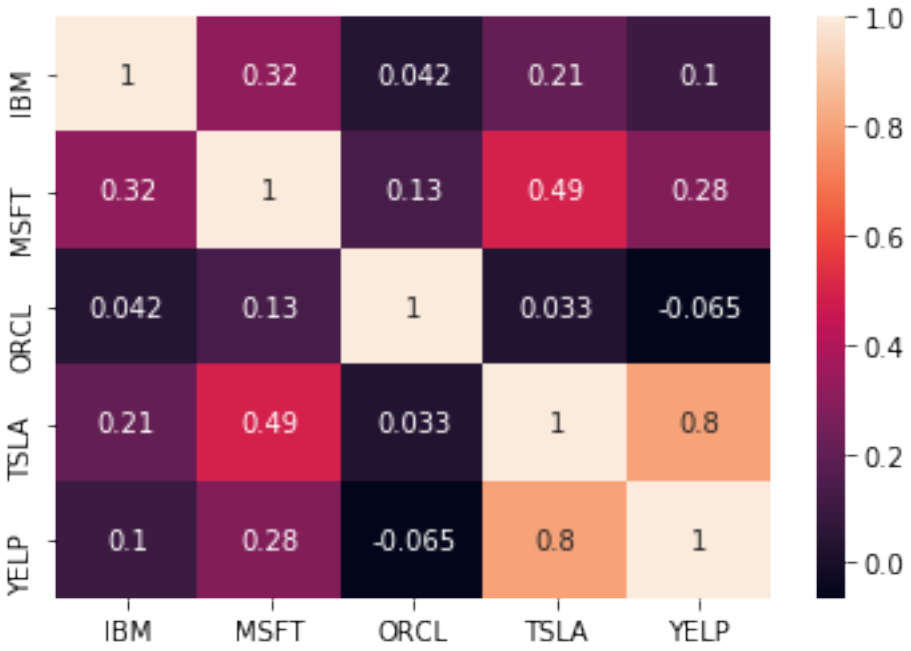
	IBM	MSFT	ORCL	TSLA	YELP
IBM	1.000000	0.321583	0.042213	0.208735	0.103837
MSFT	0.321583	1.000000	0.130515	0.492674	0.282827
ORCL	0.042213	0.130515	1.000000	0.032724	-0.065211
TSLA	0.208735	0.492674	0.032724	1.000000	0.800936
YELP	0.103837	0.282827	-0.065211	0.800936	1.000000

It takes a bit of time to examine that table and draw conclusions.

To speed that process up it helps to visualize the table.

We will learn more about visualization later, but for now this is a simple example.

```
In [59]: _ = sns.heatmap(rets.corr(), annot=True)
         # Here underscore is a "throwaway" name.
```



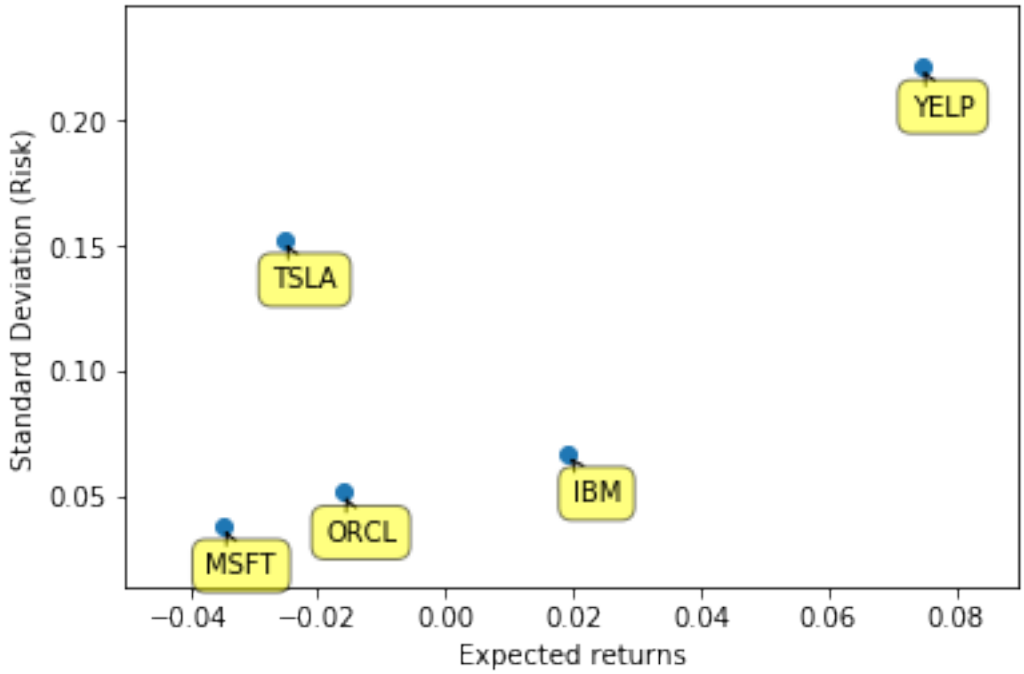
Finally, it is important to know that the plotting performed by Pandas is just a layer on top of `matplotlib` (i.e., the `plt` package).

So Panda's plots can (and should) be replaced or improved by using additional functions from `matplotlib`.

For example, suppose we want to know both the returns as well as the standard deviation of the returns of a stock (i.e., its risk).

Here is visualization of the result of such an analysis, and we construct the plot using only functions from `matplotlib`.

```
In [60]: plt.xlabel('Expected returns')
         plt.ylabel('Standard Deviation (Risk)')
         for label, x, y in zip(rets.columns, rets.mean(), rets.std()):
             plt.annotate(
                 label,
                 xy = (x, y), xytext = (20, -20),
                 textcoords = 'offset points', ha = 'right', va = 'bottom',
                 bbox = dict(boxstyle = 'round,pad=0.5', fc = 'yellow', alpha = 0.5),
                 arrowprops = dict(arrowstyle = '->', connectionstyle = 'arc3,rad=0'))
         _ = plt.scatter(rets.mean(), rets.std())
```



To understand what these functions are doing, (especially the `annotate` function), you will need to consult the online documentation for matplotlib (just search the web).