Pandas Basics

Pandas is a very widely used library used to store and manipulate tables of data in Python.

This notebook will cover the first steps with pandas of loading data into a pandas table (called a DataFrame) and extracting subsets of that data.

https://pandas.pydata.org/

1. Loading Data From a File

In [1]:
```
# import the pandas library
import pandas as pd
```

In [2]:
```
# load a csv file from 'https://s3.eu-west-1.amazonaws.com/neueda.conygre.com/pydata/
df = pd.read_csv('https://s3.eu-west-1.amazonaws.com/neueda.conygre.com/pydata/AAPL.c
```

In [3]:
```
# Display the top and bottom rows
display(df.head(20))
df.tail()

# Note how only the results of the LAST line in the cell is displayed

# Note the index column is a number, now go back and change the read_csv call to make
```

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	
2000-01-03	104.87	112.50	101.69	111.94	4783900.0	0.0	1.0	3.369314	3.614454	3.267146	3.5
2000-01-04	108.25	110.62	101.19	102.50	4574800.0	0.0	1.0	3.477908	3.554053	3.251081	3.
2000-01-05	103.75	110.56	103.00	104.00	6949300.0	0.0	1.0	3.333330	3.552125	3.309234	3.
2000-01-06	106.12	107.00	95.00	95.00	6856900.0	0.0	1.0	3.409475	3.437748	3.052206	3.
2000-01-07	96.50	101.00	95.50	99.50	4113700.0	0.0	1.0	3.100399	3.244977	3.068270	3.
2000-01-10	102.00	102.25	94.75	97.75	4509500.0	0.0	1.0	3.277105	3.285138	3.044174	3.
2000-01-11	95.94	99.37	90.50	92.75	3942400.0	0.0	1.0	3.082407	3.192607	2.907628	2.
2000-01-12	95.00	95.50	86.50	87.19	8714900.0	0.0	1.0	3.052206	3.068270	2.779114	2.
2000-01-13	94.48	98.75	92.50	96.75	9220400.0	0.0	1.0	3.035499	3.172688	2.971885	3.
2000-01-14	100.00	102.25	99.37	100.44	3485500.0	0.0	1.0	3.212848	3.285138	3.192607	3.
2000-01-18	101.00	106.00	100.44	103.94	4099800.0	0.0	1.0	3.244977	3.405619	3.226985	3.
2000-01-19	105.62	108.75	103.37	106.56	5336100.0	0.0	1.0	3.393411	3.493973	3.321121	3.
2000-01-20	115.50	121.50	113.50	113.50	16349400.0	0.0	1.0	3.710840	3.903611	3.646583	3.6
2000-01-21	114.25	114.25	110.19	111.31	4427900.0	0.0	1.0	3.670679	3.670679	3.540238	3.
2000-01-24	108.44	112.75	105.12	106.25	3936400.0	0.0	1.0	3.484013	3.622487	3.377346	3.
2000-01-25	105.00	113.12	102.37	112.25	4438800.0	0.0	1.0	3.373491	3.634374	3.288993	3.6
2000-01-26	110.00	114.19	109.75	110.19	3278200.0	0.0	1.0	3.534133	3.668752	3.526101	3.5
2000-01-27	108.81	113.00	107.00	110.00	3037000.0	0.0	1.0	3.495900	3.630519	3.437748	3.
2000-01-28	108.19	110.87	100.62	101.62	3779900.0	0.0	1.0	3.475981	3.562085	3.232768	3.
2000-01-31	101.00	103.87	94.50	103.75	6265000.0	0.0	1.0	3.244977	3.337186	3.036142	3.

Out[3]:

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close
2018-03-21	175.04	175.09	171.26	171.270	35247358.0	0.0	1.0	175.04	175.09	171.26	171.270
2018-03-22	170.00	172.68	168.60	168.845	41051076.0	0.0	1.0	170.00	172.68	168.60	168.845
2018-03-23	168.39	169.92	164.94	164.940	40248954.0	0.0	1.0	168.39	169.92	164.94	164.940
2018-03-26	168.07	173.10	166.44	172.770	36272617.0	0.0	1.0	168.07	173.10	166.44	172.770
2018-03-27	173.68	175.15	166.92	168.340	38962839.0	0.0	1.0	173.68	175.15	166.92	168.340

In [4]:
```
# List only the column names
df.columns
```

Out[4]:
```
Index(['Open', 'High', 'Low', 'Close', 'Volume', 'Ex-Dividend', 'Split Ratio',
       'Adj. Open', 'Adj. High', 'Adj. Low', 'Adj. Close', 'Adj. Volume'],
      dtype='object')
```

2. Select By Column(s)

Here we will look at the syntax to select just a subset of the columns in the table

In [5]:
```
# select just the Close column
df['Close']
```

Out[5]:
```
Date
2000-01-03    111.940
2000-01-04    102.500
2000-01-05    104.000
2000-01-06     95.000
2000-01-07     99.500
               ...
2018-03-21    171.270
2018-03-22    168.845
2018-03-23    164.940
2018-03-26    172.770
2018-03-27    168.340
Name: Close, Length: 4585, dtype: float64
```

In [6]:
```
# Select the Volume and Close columns
df[ ['Volume', 'Close'] ]
```

Out[6]:

Date	Volume	Close
2000-01-03	4783900.0	111.940
2000-01-04	4574800.0	102.500
2000-01-05	6949300.0	104.000
2000-01-06	6856900.0	95.000
2000-01-07	4113700.0	99.500
...	...	...
2018-03-21	35247358.0	171.270
2018-03-22	41051076.0	168.845
2018-03-23	40248954.0	164.940
2018-03-26	36272617.0	172.770
2018-03-27	38962839.0	168.340

4585 rows × 2 columns

3. Select By Row(s)

Here we will look at the syntax to select a subset of the rows in the table

2 Options -

1. Use `.loc[]` or `.iloc` (preferred)
2. **This has been deprecated, but you may see it used!**: Use `[]` i.e. same as select by Column. This allows pandas to auto-detect if you are referring to a column name or a sequence of rows. Shorter syntax **BUT** unclear as to whether you are referring to a ROW or a COLUMN

In [7]:
```
# Option 1 - For clarity!
df.loc['2018']

# Option 2 - This has been deprecated and should not be used!!
df['2018']
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: FutureWarning: In
dexing a DataFrame with a datetimelike index using a single string to slice the rows,
like `frame[string]`, is deprecated and will be removed in a future version. Use `fra
me.loc[string]` instead.
  """
```

Out[7]:

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Lo
2018-01-02	170.160	172.3000	169.2600	172.260	25048048.0	0.0	1.0	170.160	172.3000	169.260
2018-01-03	172.530	174.5500	171.9600	172.230	28819653.0	0.0	1.0	172.530	174.5500	171.960
2018-01-04	172.540	173.4700	172.0800	173.030	22211345.0	0.0	1.0	172.540	173.4700	172.080
2018-01-05	173.440	175.3700	173.0500	175.000	23016177.0	0.0	1.0	173.440	175.3700	173.050
2018-01-08	174.350	175.6100	173.9300	174.350	20134092.0	0.0	1.0	174.350	175.6100	173.930
2018-01-09	174.550	175.0600	173.4100	174.330	21262614.0	0.0	1.0	174.550	175.0600	173.410
2018-01-10	173.160	174.3000	173.0000	174.290	23589129.0	0.0	1.0	173.160	174.3000	173.000
2018-01-11	174.590	175.4886	174.4900	175.280	17523256.0	0.0	1.0	174.590	175.4886	174.490
2018-01-12	176.180	177.3600	175.6500	177.090	25039531.0	0.0	1.0	176.180	177.3600	175.650
2018-01-16	177.900	179.3900	176.1400	176.190	29159005.0	0.0	1.0	177.900	179.3900	176.140
2018-01-17	176.150	179.2500	175.0700	179.100	32752734.0	0.0	1.0	176.150	179.2500	175.070
2018-01-18	179.370	180.1000	178.2500	179.260	30234512.0	0.0	1.0	179.370	180.1000	178.250
2018-01-19	178.610	179.5800	177.4100	178.460	30827809.0	0.0	1.0	178.610	179.5800	177.410
2018-01-22	177.300	177.7800	176.6016	177.000	26023683.0	0.0	1.0	177.300	177.7800	176.601
2018-01-23	177.300	179.4400	176.8200	177.040	31702531.0	0.0	1.0	177.300	179.4400	176.820
2018-01-24	177.250	177.3000	173.2000	174.220	50562257.0	0.0	1.0	177.250	177.3000	173.200
2018-01-25	174.505	174.9500	170.5300	171.110	39661804.0	0.0	1.0	174.505	174.9500	170.530
2018-01-26	172.000	172.0000	170.0600	171.510	37121805.0	0.0	1.0	172.000	172.0000	170.060
2018-01-29	170.160	170.1600	167.0700	167.960	48434424.0	0.0	1.0	170.160	170.1600	167.070
2018-01-30	165.525	167.3700	164.7000	166.970	45137026.0	0.0	1.0	165.525	167.3700	164.700
2018-01-31	166.870	168.4417	166.5000	167.430	30984099.0	0.0	1.0	166.870	168.4417	166.500
2018-02-01	167.165	168.6200	166.7600	167.780	38099665.0	0.0	1.0	167.165	168.6200	166.760
2018-02-02	166.000	166.8000	160.1000	160.370	85436075.0	0.0	1.0	166.000	166.8000	160.100
2018-02-05	159.100	163.8800	156.0000	157.490	66090446.0	0.0	1.0	159.100	163.8800	156.000

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Lo
2018-02-06	154.830	163.7200	154.0000	163.030	66625484.0	0.0	1.0	154.830	163.7200	154.000
2018-02-07	163.085	163.4000	159.0685	159.540	50852130.0	0.0	1.0	163.085	163.4000	159.068
2018-02-08	160.290	161.0000	155.0300	155.320	49594129.0	0.0	1.0	160.290	161.0000	155.030
2018-02-09	157.070	157.8900	150.2400	155.970	66723743.0	0.0	1.0	157.070	157.8900	150.240
2018-02-12	158.500	163.8900	157.5100	162.710	60560145.0	0.0	1.0	158.500	163.8900	157.510
2018-02-13	161.950	164.7500	161.6500	164.340	32104756.0	0.0	1.0	161.950	164.7500	161.650
2018-02-14	163.045	167.5400	162.8800	167.370	39669178.0	0.0	1.0	163.045	167.5400	162.880
2018-02-15	169.790	173.0900	169.0000	172.990	50609595.0	0.0	1.0	169.790	173.0900	169.000
2018-02-16	172.360	174.8200	171.7700	172.430	39638793.0	0.0	1.0	172.360	174.8200	171.770
2018-02-20	172.050	174.2600	171.4200	171.850	33531012.0	0.0	1.0	172.050	174.2600	171.420
2018-02-21	172.830	174.1200	171.0100	171.070	35833514.0	0.0	1.0	172.830	174.1200	171.010
2018-02-22	171.800	173.9500	171.7100	172.600	30504116.0	0.0	1.0	171.800	173.9500	171.710
2018-02-23	173.670	175.6500	173.5400	175.555	33329232.0	0.0	1.0	173.670	175.6500	173.540
2018-02-26	176.350	179.3900	176.2100	178.970	36886432.0	0.0	1.0	176.350	179.3900	176.210
2018-02-27	179.100	180.4800	178.1600	178.390	38685165.0	0.0	1.0	179.100	180.4800	178.160
2018-02-28	179.260	180.6150	178.0500	178.120	33604574.0	0.0	1.0	179.260	180.6150	178.050
2018-03-01	178.540	179.7750	172.6600	175.000	48801970.0	0.0	1.0	178.540	179.7750	172.660
2018-03-02	172.800	176.3000	172.4500	176.210	38453950.0	0.0	1.0	172.800	176.3000	172.450
2018-03-05	175.210	177.7400	174.5200	176.820	28401366.0	0.0	1.0	175.210	177.7400	174.520
2018-03-06	177.910	178.2500	176.1300	176.670	23788506.0	0.0	1.0	177.910	178.2500	176.130

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Lo
2018-03-07	174.940	175.8500	174.2700	175.030	31703462.0	0.0	1.0	174.940	175.8500	174.270
2018-03-08	175.480	177.1200	175.0700	176.940	23163767.0	0.0	1.0	175.480	177.1200	175.070
2018-03-09	177.960	180.0000	177.3900	179.980	31385134.0	0.0	1.0	177.960	180.0000	177.390
2018-03-12	180.290	182.3900	180.2100	181.720	32055405.0	0.0	1.0	180.290	182.3900	180.210
2018-03-13	182.590	183.5000	179.2400	179.970	31168404.0	0.0	1.0	182.590	183.5000	179.240
2018-03-14	180.320	180.5200	177.8100	178.440	29075469.0	0.0	1.0	180.320	180.5200	177.810
2018-03-15	178.500	180.2400	178.0701	178.650	22584565.0	0.0	1.0	178.500	180.2400	178.070
2018-03-16	178.650	179.1200	177.6200	178.020	36836456.0	0.0	1.0	178.650	179.1200	177.620
2018-03-19	177.320	177.4700	173.6600	175.300	32804695.0	0.0	1.0	177.320	177.4700	173.660
2018-03-20	175.240	176.8000	174.9400	175.240	19314039.0	0.0	1.0	175.240	176.8000	174.940
2018-03-21	175.040	175.0900	171.2600	171.270	35247358.0	0.0	1.0	175.040	175.0900	171.260
2018-03-22	170.000	172.6800	168.6000	168.845	41051076.0	0.0	1.0	170.000	172.6800	168.600
2018-03-23	168.390	169.9200	164.9400	164.940	40248954.0	0.0	1.0	168.390	169.9200	164.940
2018-03-26	168.070	173.1000	166.4400	172.770	36272617.0	0.0	1.0	168.070	173.1000	166.440
2018-03-27	173.680	175.1500	166.9200	168.340	38962839.0	0.0	1.0	173.680	175.1500	166.920

In [8]:

```
# Note how with a Date as the index, we can do some clever date selections

# select all in 'Mar 2000'
df.loc['Mar 2000']

# select all between 'Apr 2003' and 'Feb 2004'
df.loc['Apr 2003':'Feb 2004']
```

Out[8]:

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	C
2003-04-01	14.20	14.31	14.0700	14.16	2756100.0	0.0	1.0	0.912449	0.919517	0.904096	0.90
2003-04-02	14.36	14.69	14.2675	14.60	3060200.0	0.0	1.0	0.922730	0.943935	0.916786	0.93
2003-04-03	14.55	14.70	14.3500	14.46	2602000.0	0.0	1.0	0.934939	0.944577	0.922088	0.92
2003-04-04	14.52	14.67	14.3900	14.41	2607500.0	0.0	1.0	0.933011	0.942650	0.924658	0.92
2003-04-07	14.85	14.95	14.4100	14.49	3515400.0	0.0	1.0	0.954216	0.960642	0.925943	0.93
...	...	...	...	...	...	...	...	...	...	...	
2004-02-23	22.45	22.46	21.8899	22.19	3465700.0	0.0	1.0	1.442569	1.443212	1.406579	1.42
2004-02-24	22.14	22.74	22.0000	22.36	4626000.0	0.0	1.0	1.422649	1.461203	1.413653	1.43
2004-02-25	22.22	22.90	22.2100	22.81	4933500.0	0.0	1.0	1.427790	1.471485	1.427147	1.46
2004-02-26	22.84	23.18	22.8000	23.04	3543000.0	0.0	1.0	1.467629	1.489477	1.465059	1.48
2004-02-27	22.96	24.02	22.9500	23.92	8372100.0	0.0	1.0	1.475340	1.543452	1.474697	1.53

230 rows × 12 columns

4. Select a "slice" of rows

In the cells above, when selecting a "range" we used **:**. This syntax is common when selecting "slices" from python structures.

- Use [], loc or iloc
- Same syntax as for slicing Python lists, strings etc.

`myDf[ firstRow : lastRow : step]`

(step is usually left out, and defaults to 1)

In [9]:

```
# The same syntax applies when using a DataFrame!

# Slicing Data e.g. select every row from row 10 to row 15
df[10:15]

# e.g. every second row from row 20 to row 50
df[20:50:2]
```

Out[9]:

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	
2000-02-01	104.00	105.00	100.00	100.25	2839600.0	0.0	1.0	3.341362	3.373491	3.212848	3.2
2000-02-03	100.31	104.25	100.25	103.31	4242800.0	0.0	1.0	3.222808	3.349395	3.220881	3.3
2000-02-07	108.00	114.25	105.94	114.06	3938100.0	0.0	1.0	3.469876	3.670679	3.403692	3.6
2000-02-09	114.12	117.12	112.44	112.62	2672900.0	0.0	1.0	3.666503	3.762888	3.612527	3.6
2000-02-11	113.62	114.12	108.25	108.75	1895100.0	0.0	1.0	3.650438	3.666503	3.477908	3.4
2000-02-15	115.25	119.94	115.19	119.00	4337000.0	0.0	1.0	3.702808	3.853490	3.700880	3.8
2000-02-17	115.19	115.50	113.12	114.87	2584800.0	0.0	1.0	3.700880	3.710840	3.634374	3.6
2000-02-22	110.12	116.94	106.69	113.81	3770500.0	0.0	1.0	3.537989	3.757105	3.427788	3.6
2000-02-24	117.31	119.12	111.75	115.20	3361000.0	0.0	1.0	3.768993	3.827145	3.590358	3.7
2000-02-28	110.12	115.00	108.37	113.25	2931500.0	0.0	1.0	3.537989	3.694776	3.481764	3.6
2000-03-01	118.56	132.06	118.50	130.31	9616100.0	0.0	1.0	3.809153	4.242888	3.807225	4.1
2000-03-03	124.87	128.23	120.00	128.00	2887200.0	0.0	1.0	4.011884	4.119836	3.855418	4.1
2000-03-07	126.44	127.44	121.12	122.87	2437600.0	0.0	1.0	4.062326	4.094454	3.891402	3.9
2000-03-09	120.87	125.00	118.25	122.25	2470700.0	0.0	1.0	3.883370	4.016061	3.799193	3.9
2000-03-13	122.12	126.50	119.50	121.31	2713900.0	0.0	1.0	3.923531	4.064253	3.839354	3.8

5. Filter by a condition

This is different from slicing, we are now selecting parts of the DataFrame for which some condition is True or False

At first the syntax may look complicated, but it's the same pattern for all filters

In [10]:

```
# select rows where 'Open' is greater than 21
df[ df['Open'] > 21 ]

# Can chain together multiple "conditions" with '&' for AND, '|' for OR

# Note the round brackets surrounding each filter

# select rows where 'Open' is greater than 160 AND 'Adj. Low' is greater and 5

df[ (df['Open'] > 160) & (df['Adj. Low'] > 5) ]
```

Out[10]:

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj
2007-10-08	163.49	167.91	162.97	167.9100	29854600.0	0.0	1.0	21.010744	21.578775	20.94
2007-10-09	170.20	171.11	166.68	167.8600	39438800.0	0.0	1.0	21.873072	21.990020	21.42
2007-10-10	167.55	167.88	165.60	166.7900	23842500.0	0.0	1.0	21.532510	21.574920	21.28
2007-10-11	169.49	171.88	153.21	162.2300	58714000.0	0.0	1.0	21.781827	22.088976	19.68
2007-10-12	163.01	167.28	161.80	167.2537	35292000.0	0.0	1.0	20.949057	21.497811	20.79
...	...	...	...	...	...	...	...	...	...	
2018-03-21	175.04	175.09	171.26	171.2700	35247358.0	0.0	1.0	175.040000	175.090000	171.26
2018-03-22	170.00	172.68	168.60	168.8450	41051076.0	0.0	1.0	170.000000	172.680000	168.60
2018-03-23	168.39	169.92	164.94	164.9400	40248954.0	0.0	1.0	168.390000	169.920000	164.94
2018-03-26	168.07	173.10	166.44	172.7700	36272617.0	0.0	1.0	168.070000	173.100000	166.44
2018-03-27	173.68	175.15	166.92	168.3400	38962839.0	0.0	1.0	173.680000	175.150000	166.92

1498 rows × 12 columns

6. The ".query" syntax

There is a second syntax recently added to pandas which allows for filtering DataFrames with a more readable syntax

Sometimes unusual characters in column names can cause problems with this

In [11]:
```
# Standard Syntax From Section 5:
df[ df['Open'] > 21 ]


# Exactly the Same using .query Syntax:
df.query('Open > 21')
```

Out[11]:

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj.
2000-01-03	104.87	112.50	101.69	111.940	4783900.0	0.0	1.0	3.369314	3.614454	3.267
2000-01-04	108.25	110.62	101.19	102.500	4574800.0	0.0	1.0	3.477908	3.554053	3.251
2000-01-05	103.75	110.56	103.00	104.000	6949300.0	0.0	1.0	3.333330	3.552125	3.309
2000-01-06	106.12	107.00	95.00	95.000	6856900.0	0.0	1.0	3.409475	3.437748	3.052
2000-01-07	96.50	101.00	95.50	99.500	4113700.0	0.0	1.0	3.100399	3.244977	3.068
...	...	...	...	...	...	...	...	...	...	
2018-03-21	175.04	175.09	171.26	171.270	35247358.0	0.0	1.0	175.040000	175.090000	171.260
2018-03-22	170.00	172.68	168.60	168.845	41051076.0	0.0	1.0	170.000000	172.680000	168.600
2018-03-23	168.39	169.92	164.94	164.940	40248954.0	0.0	1.0	168.390000	169.920000	164.940
2018-03-26	168.07	173.10	166.44	172.770	36272617.0	0.0	1.0	168.070000	173.100000	166.440
2018-03-27	173.68	175.15	166.92	168.340	38962839.0	0.0	1.0	173.680000	175.150000	166.920

4048 rows × 12 columns

In [12]:

```
# Standard Syntax From Section 5:
display( df[ (df['Open'] > 160) & (df['Close'] - df['Open'] > 20) ] )

# Exactly the Same using .query Syntax:
df.query('(Open > 160) and (Close - Open > 20)')
```

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj
2012-04-17	578.9400	610.00	571.91	609.7000	36626000.0	0.0	1.0	74.401859	78.393502	73.49
2012-05-21	534.5000	561.54	534.05	561.2800	22539500.0	0.0	1.0	68.690700	72.165717	68.63
2012-10-22	612.4200	635.38	610.76	634.0300	19526100.0	0.0	1.0	79.040508	82.003785	78.82
2012-11-19	540.7097	567.50	539.88	565.7300	29404200.0	0.0	1.0	70.116809	73.590855	70.00
2012-12-31	510.5300	535.40	509.00	532.1729	23553300.0	0.0	1.0	66.203240	69.428271	66.00
2014-04-28	572.8000	595.75	572.55	594.0900	23910200.0	0.0	1.0	76.591114	79.659840	76.55

Out[12]:

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj
2012-04-17	578.9400	610.00	571.91	609.7000	36626000.0	0.0	1.0	74.401859	78.393502	73.49
2012-05-21	534.5000	561.54	534.05	561.2800	22539500.0	0.0	1.0	68.690700	72.165717	68.63
2012-10-22	612.4200	635.38	610.76	634.0300	19526100.0	0.0	1.0	79.040508	82.003785	78.82
2012-11-19	540.7097	567.50	539.88	565.7300	29404200.0	0.0	1.0	70.116809	73.590855	70.0(
2012-12-31	510.5300	535.40	509.00	532.1729	23553300.0	0.0	1.0	66.203240	69.428271	66.00
2014-04-28	572.8000	595.75	572.55	594.0900	23910200.0	0.0	1.0	76.591114	79.659840	76.55

Selecting from a DataFrame - Summary

Whatever subset of the data is required can be selected in pandas with a small amount of code!

7. Creating New Columns

Create a new column simply by referring to it (just like any Python variable).

Mathematical operations operate on columns element-wise.

In [13]:
```
# create a column called 'CloseMinusOpen' containing for each row the Close value min

df['CloseMinusOpen'] = df['Close'] - df['Open']

df.head()
```

Out[13]:

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	
2000-01-03	104.87	112.50	101.69	111.94	4783900.0	0.0	1.0	3.369314	3.614454	3.267146	3.59
2000-01-04	108.25	110.62	101.19	102.50	4574800.0	0.0	1.0	3.477908	3.554053	3.251081	3.2(
2000-01-05	103.75	110.56	103.00	104.00	6949300.0	0.0	1.0	3.333330	3.552125	3.309234	3.3(
2000-01-06	106.12	107.00	95.00	95.00	6856900.0	0.0	1.0	3.409475	3.437748	3.052206	3.05
2000-01-07	96.50	101.00	95.50	99.50	4113700.0	0.0	1.0	3.100399	3.244977	3.068270	3.19

8. Aggregating Operations

These are operations that combine elements in a column into a single value, e.g. get the sum of all elements in the row## Aggregating Operations

In [14]:
```
# find the minimum value of the Close column
df['Close'].min()
```

Out[14]:
```
13.12
```

In [15]:
```
# find the minimum value of all columns
df.min()
```

Out[15]:
```
Open              1.299000e+01
High              1.319000e+01
Low               1.272000e+01
Close             1.312000e+01
Volume            7.025000e+05
Ex-Dividend       0.000000e+00
Split Ratio       1.000000e+00
Adj. Open         8.346980e-01
Adj. High         8.475494e-01
Adj. Low          8.173486e-01
Adj. Close        8.430514e-01
Adj. Volume       9.835000e+06
CloseMinusOpen   -3.011770e+01
dtype: float64
```

In [16]:
```
# describe and transpose
df.describe().transpose()
```

Out[16]:

	count	mean	std	min	25%	50%	
Open	4585.0	1.694568e+02	1.677684e+02	1.299000e+01	4.679000e+01	1.119500e+02	1.
High	4585.0	1.713002e+02	1.691125e+02	1.319000e+01	4.760000e+01	1.133700e+02	1.
Low	4585.0	1.674036e+02	1.661504e+02	1.272000e+01	4.621000e+01	1.102700e+02	1.
Close	4585.0	1.693867e+02	1.676526e+02	1.312000e+01	4.689000e+01	1.120100e+02	1.
Volume	4585.0	2.272219e+07	1.836345e+07	7.025000e+05	8.559200e+06	1.861380e+07	3
Ex-Dividend	4585.0	6.645583e-03	1.259664e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.
Split Ratio	4585.0	1.001745e+00	9.103108e-02	1.000000e+00	1.000000e+00	1.000000e+00	1.
Adj. Open	4585.0	4.321587e+01	4.740539e+01	8.346980e-01	3.601282e+00	2.151452e+01	7
Adj. High	4585.0	4.361546e+01	4.776136e+01	8.475494e-01	3.710840e+00	2.180239e+01	7
Adj. Low	4585.0	4.278409e+01	4.703417e+01	8.173486e-01	3.523852e+00	2.118295e+01	7
Adj. Close	4585.0	4.320920e+01	4.740618e+01	8.430514e-01	3.630519e+00	2.157235e+01	7
Adj. Volume	4585.0	1.231199e+08	9.964739e+07	9.835000e+06	5.633116e+07	9.645333e+07	1.
CloseMinusOpen	4585.0	-7.006489e-02	3.459621e+00	-3.011770e+01	-9.700000e-01	1.000000e-02	1.

9. Finding Unique Values

For this example we will use some different data - so load a new DataFrame

In [17]:
```
# read a different csv file into a new variable : 'https://s3.eu-west-1.amazonaws.com
df_faang = pd.read_csv('https://s3.eu-west-1.amazonaws.com/neueda.conygre.com/pydata/

df_faang.head()
```

Out[17]:

	Symbol	Name	Sector	Price	Price/Earnings	Earnings/Share	52 Week High	52 Week Low
0	FB	Facebook, Inc.	Information Technology	171.58	27.90	5.39	195.32	132.44
1	AMZN	Amazon.com Inc	Consumer Discretionary	1350.50	296.16	6.16	1498.00	812.50
2	AAPL	Apple Inc.	Information Technology	155.15	16.86	9.20	180.10	131.12
3	NFLX	Netflix Inc.	Information Technology	250.10	200.08	1.25	286.81	138.26
4	GOOGL	Alphabet Inc Class A	Information Technology	1007.71	31.48	22.27	1198.00	824.30

In [18]:
```
print(df_faang.head())

# How MANY unique sectors are listed?
print(df_faang['Sector'].unique())

# List the unique sectors?
df_faang['Sector'].nunique()
```

```
  Symbol                   Name                  Sector    Price  \
0     FB          Facebook, Inc.  Information Technology   171.58
1   AMZN          Amazon.com Inc  Consumer Discretionary  1350.50
2   AAPL              Apple Inc.  Information Technology   155.15
3   NFLX            Netflix Inc.  Information Technology   250.10
4  GOOGL  Alphabet Inc Class A    Information Technology  1007.71

   Price/Earnings  Earnings/Share  52 Week High  52 Week Low
0           27.90            5.39        195.32       132.44
1          296.16            6.16       1498.00       812.50
2           16.86            9.20        180.10       131.12
3          200.08            1.25        286.81       138.26
4           31.48           22.27       1198.00       824.30
['Information Technology' 'Consumer Discretionary']
2
```

Out[18]:

10. Basic Plotting

Note that we're temporarily back to using our original DataFrame 'df'

In [19]:
```
# First ensure we're sorted correctly - use the AAPL dataset, sort by date (ascending
df = df.sort_values(by='Date')
df.head()
```

Out[19]:

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	
2000-01-03	104.87	112.50	101.69	111.94	4783900.0	0.0	1.0	3.369314	3.614454	3.267146	3.59
2000-01-04	108.25	110.62	101.19	102.50	4574800.0	0.0	1.0	3.477908	3.554053	3.251081	3.2
2000-01-05	103.75	110.56	103.00	104.00	6949300.0	0.0	1.0	3.333330	3.552125	3.309234	3.3
2000-01-06	106.12	107.00	95.00	95.00	6856900.0	0.0	1.0	3.409475	3.437748	3.052206	3.05
2000-01-07	96.50	101.00	95.50	99.50	4113700.0	0.0	1.0	3.100399	3.244977	3.068270	3.19

There are more advanced plotting libraries like seaborn and plotly. However, for simple plots we can access Matplotlib directly THROUGH PANDAS!

In [20]:
```
# Tell jupyter to display plots in the browser
%matplotlib inline

# Plot a line chart for AAPL of columns Low and High (use figsize=(18,6) for a bigger
df[ ['Low', 'High'] ].plot(figsize=(18,6))

# Plot a line chart for AAPL of columns Low and High for 2017 (use figsize=(18,6) for
df.loc['2017', ['Low', 'High']].plot(figsize=(18, 6))
```

Out[20]:
```
<AxesSubplot:xlabel='Date'>
```

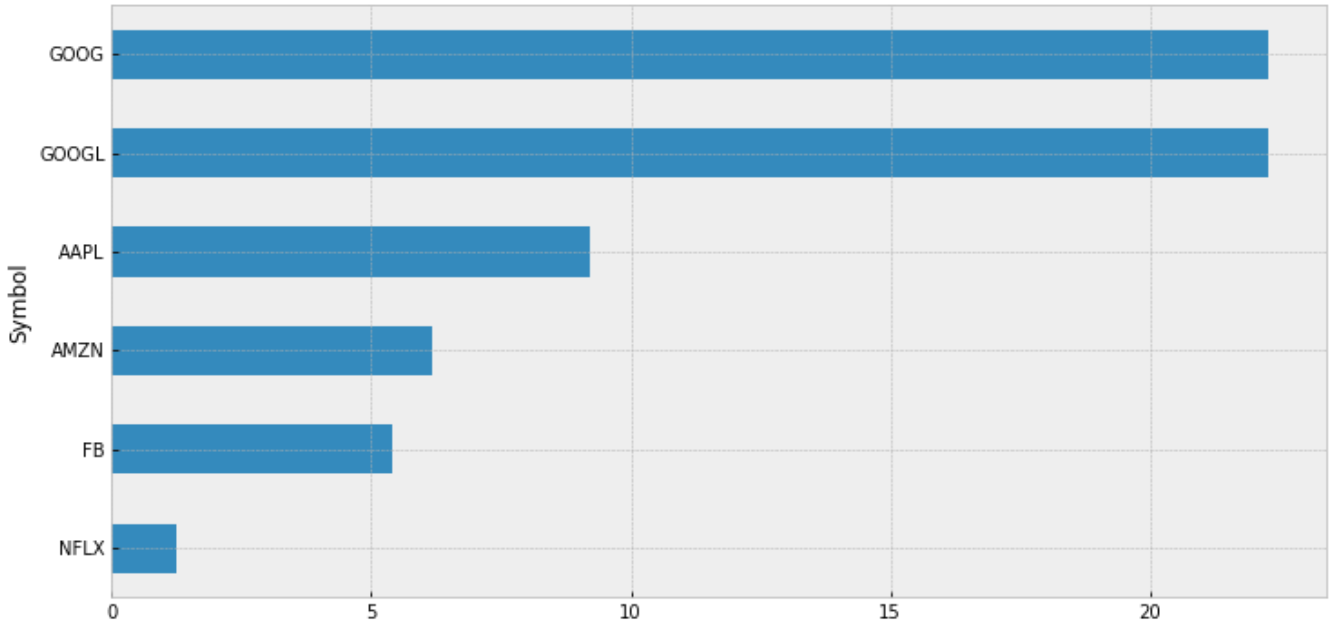
we can use this for other types of plots e.g. bar, horizontal bar (barh)

In [21]:
```
# Use the FAANG dataframe
```

```
# plot faang 'Earnings/Share' as a horizontal bar plot

# could use a better style
import matplotlib
matplotlib.style.use('bmh')

# maybe set the Symbol as the index?

# maybe also sort first?

# experiment with bar, horizontal bar, sort_values
df_faang.set_index('Symbol').sort_values('Earnings/Share')['Earnings/Share'].plot(kin
```

Out[21]:

```
<AxesSubplot:ylabel='Symbol'>
```



In [22]:

```
# Save the plot to a file
import matplotlib.pyplot as plt
plt.savefig('simple_plot.png')
```

```
<Figure size 432x288 with 0 Axes>
```

In [23]:

```
# Save a DataFrame to a csv e.g. Low and High columns from 2004
df.loc['2004', ['High', 'Low']].to_csv('appl_2018.png')
```