Tamkang University
淡江大學

Tamkang University

Social Computing and Big Data Analytics
社群運算與大數據分析
Finance Big Data Analytics with Pandas in Python
(Python Pandas 財務大數據分析)

1042SCBDA06
MIS MBA (M2226) (8628)
Wed, 8,9, (15:10-17:00) (B309)

Min-Yuh Day
戴敏育
Assistant Professor
專任助理教授
Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系
http://mail. tku.edu.tw/myday/

2016-03-23

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2016/02/17	Course Orientation for Social Computing and Big Data Analytics (社群運算與大數據分析課程介紹)
2	2016/02/24	Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data (資料科學與大數據分析：探索、分析、視覺化與呈現資料)
3	2016/03/02	Fundamental Big Data: MapReduce Paradigm, Hadoop and Spark Ecosystem (大數據基礎：MapReduce典範、Hadoop與Spark生態系統)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
4	2016/03/09	Big Data Processing Platforms with SMACK: Spark, Mesos, Akka, Cassandra and Kafka (大數據處理平台SMACK：Spark, Mesos, Akka, Cassandra, Kafka)
5	2016/03/16	Big Data Analytics with Numpy in Python (Python Numpy 大數據分析)
6	2016/03/23	Finance Big Data Analytics with Pandas in Python (Python Pandas 財務大數據分析)
7	2016/03/30	Text Mining Techniques and Natural Language Processing (文字探勘分析技術與自然語言處理)
8	2016/04/06	Off-campus study (教學行政觀摩日)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
9	2016/04/13	Social Media Marketing Analytics (社群媒體行銷分析)
10	2016/04/20	期中報告 (Midterm Project Report)
11	2016/04/27	Deep Learning with Theano and Keras in Python (Python Theano 和 Keras 深度學習)
12	2016/05/04	Deep Learning with Google TensorFlow (Google TensorFlow 深度學習)
13	2016/05/11	Sentiment Analysis on Social Media with Deep Learning (深度學習社群媒體情感分析)

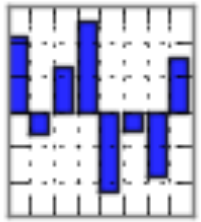
課程大綱 (Syllabus)

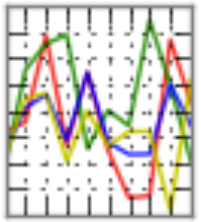
週次 (Week)	日期 (Date)	內容 (Subject/Topics)
14	2016/05/18	Social Network Analysis (社會網絡分析)
15	2016/05/25	Measurements of Social Network (社會網絡量測)
16	2016/06/01	Tools of Social Network Analysis (社會網絡分析工具)
17	2016/06/08	Final Project Presentation I (期末報告 I)
18	2016/06/15	Final Project Presentation II (期末報告 II)

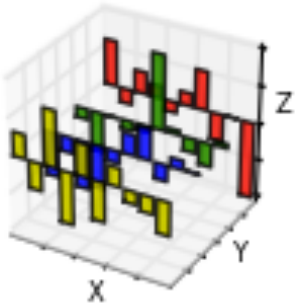
python™

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$





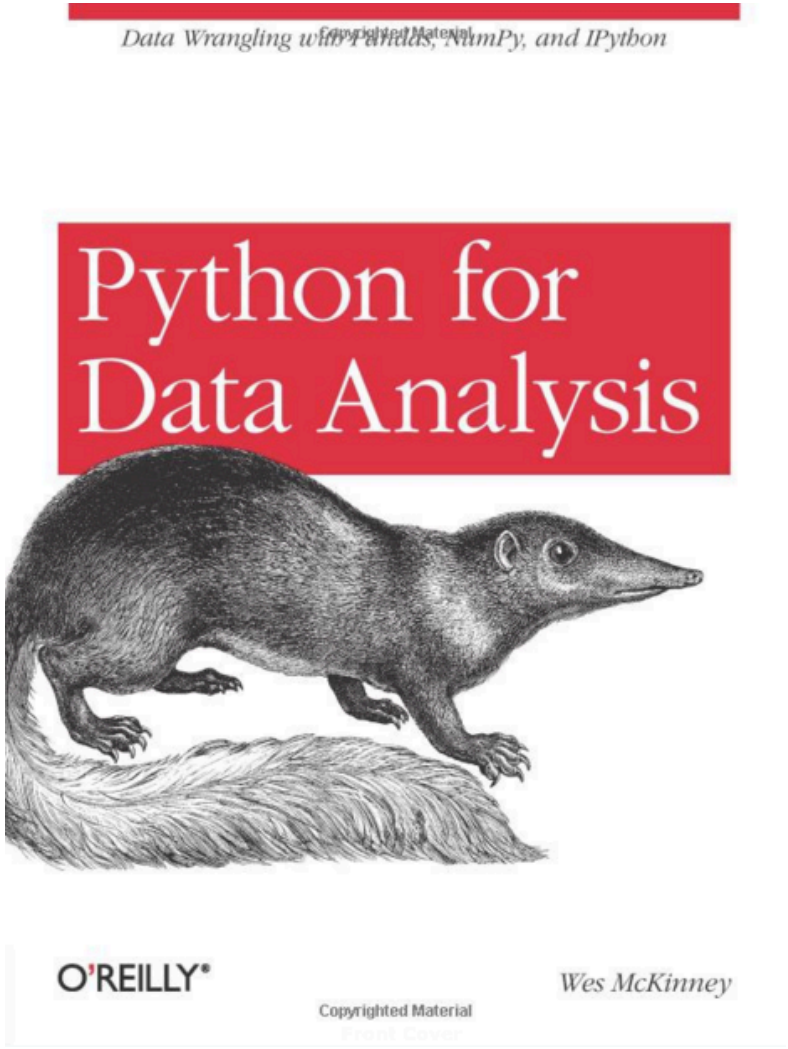


Source: https://www.python.org/community/logos/
http://pandas.pydata.org/

Yahoo Finance Charts
Apple Inc. (AAPL)

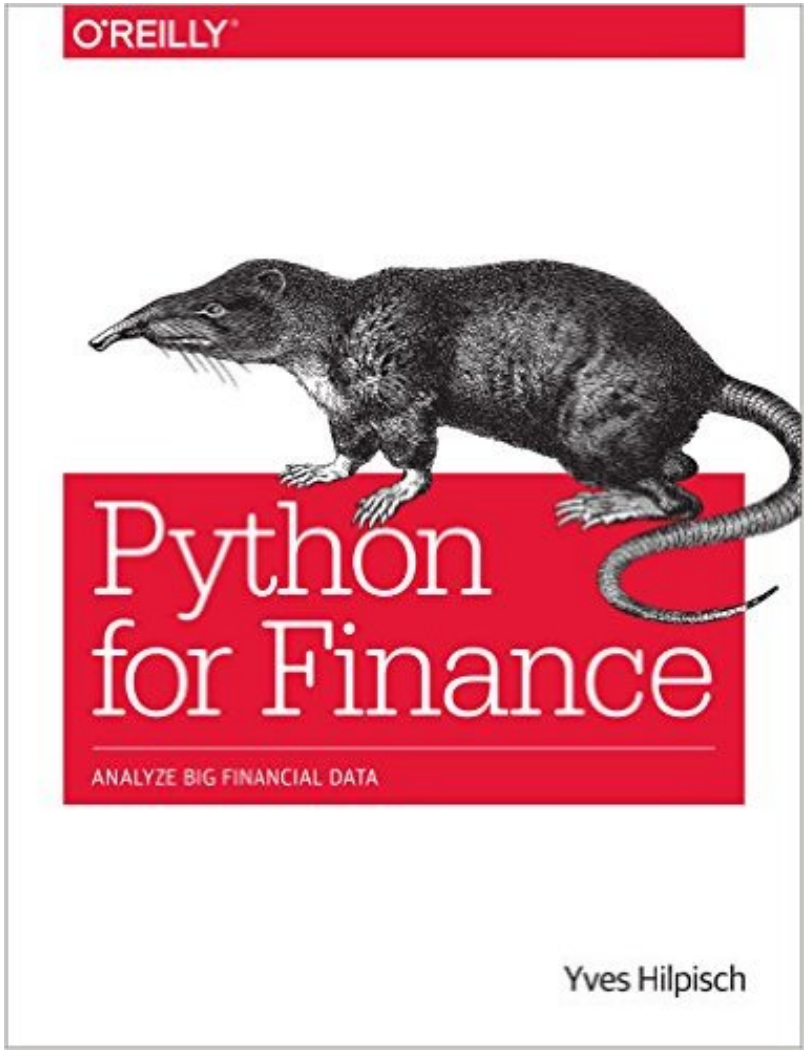
http://finance.yahoo.com/echarts?s=AAPL+Interactive#{"showArea":false,"showLine":false,"showCandle":true,"lineType":"candle","range":"5y","allowChartStacking":true}

Wes McKinney (2012), Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, O'Reilly Media



Source: http://www.amazon.com/Python-Data-Analysis-Wrangling-IPython/dp/1449319793/

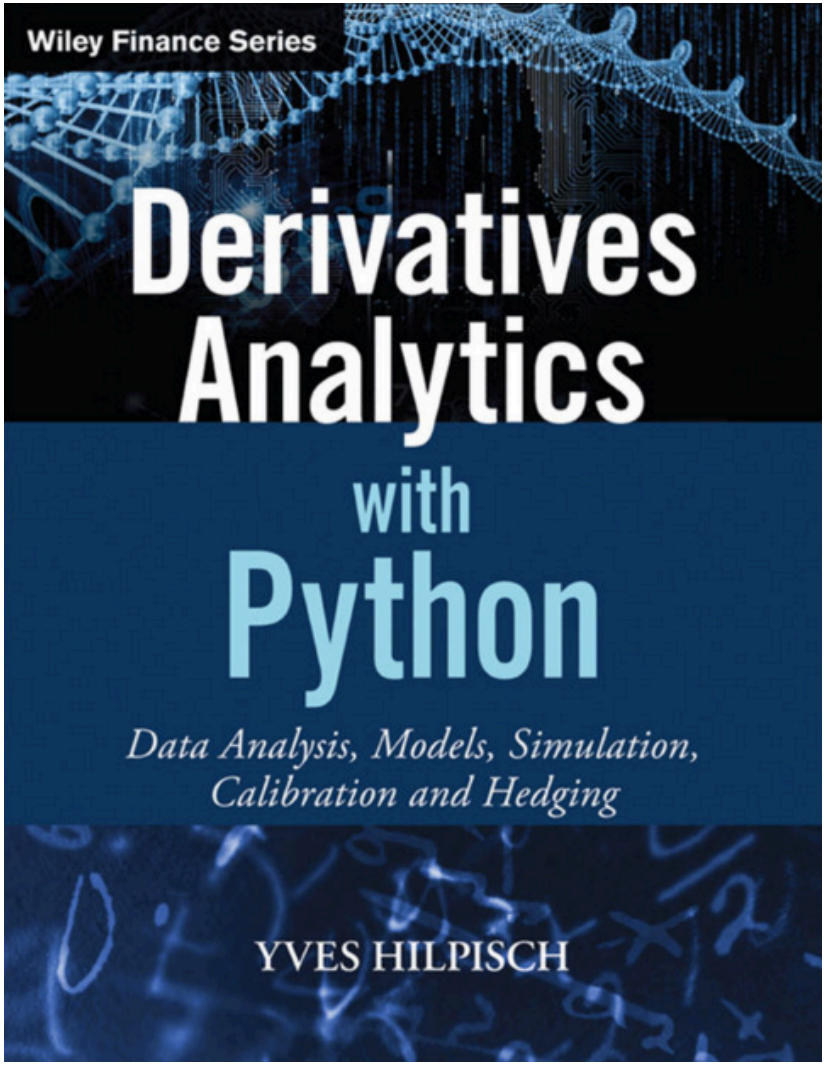
Yves Hilpisch, Python for Finance: Analyze Big Financial Data, O'Reilly, 2014



Source: http://www.amazon.com/Python-Finance-Analyze-Financial-Data/dp/1491945281

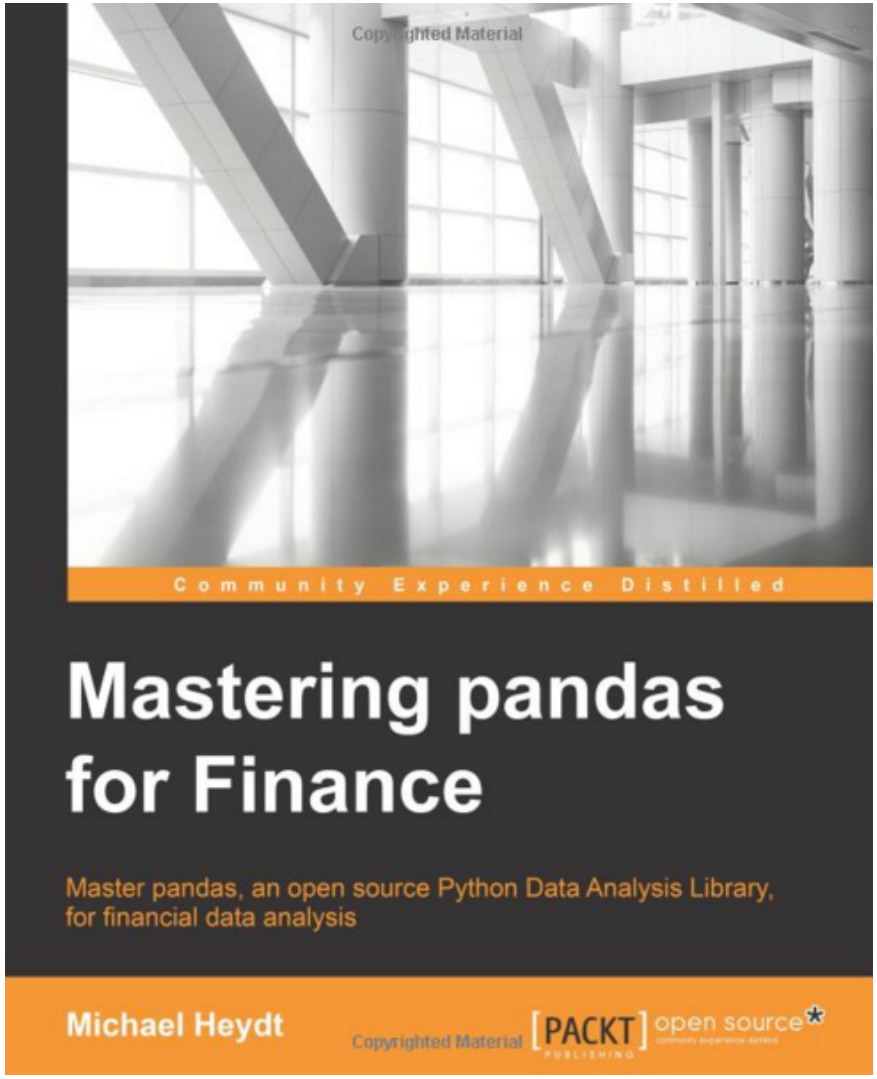
Yves Hilpisch (2015), Derivatives Analytics with Python:

Data Analysis, Models, Simulation, Calibration and Hedging, Wiley



Source: http://www.amazon.com/Derivatives-Analytics-Python-Simulation-Calibration/dp/1119037999/

Michael Heydt ,
Mastering Pandas for Finance,
Packt Publishing, 2015



Source: http://www.amazon.com/Mastering-Pandas-Finance-Michael-Heydt/dp/1783985100

pandas

Python Data Analysis Library

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

> **Note**
>
> We are proud to announce that *pandas* has become a sponsored project of the (NUMFocus organization). This will help ensure the success of development of *pandas* as a world-class open-source project.

0.18.0 Final (March 13, 2016)

This is a major release from 0.17.1 and includes a small number of API changes, several new features, enhancements, and performance improvements along with a large number of bug fixes. We recommend that all users upgrade to this version.

Highlights include:

- pandas >= 0.18.0 will no longer support compatibility with Python version 2.6 GH7718 or version 3.3 GH11273
- Moving and expanding window functions are now methods on Series and DataFrame, similar to `.groupby`, see here

VERSIONS

Release
0.18.0 - March 2016
download // docs // pdf

Development
0.18.1 - April 2016
github // docs

Previous Releases
0.17.1 - download // docs // pdf
0.17.0 - download // docs // pdf
0.16.2 - download // docs // pdf
0.16.1 - download // docs // pdf
0.16.0 - download // docs // pdf
0.15.2 - download // docs // pdf
0.15.1 - download // docs // pdf
0.15.0 - download // docs // pdf
0.14.1 - download // docs // pdf
0.14.0 - download // docs // pdf
0.13.1 - download // docs // pdf
0.13.0 - download // docs // pdf
0.12.0 - download // docs // pdf
0.11.0 - download // docs // pdf

http://pandas.pydata.org/

pandas

Python Data Analysis Library

providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Source: http://pandas.pydata.org/

pandas: powerful Python data analysis toolkit

Table Of Contents

pandas: powerful Python data analysis toolkit

PDF Version

Zipped HTML

Date: March 17, 2016 **Version**: 0.18.0

Binary Installers: http://pypi.python.org/pypi/pandas

Source Repository: http://github.com/pydata/pandas

Issues & Ideas: https://github.com/pydata/pandas/issues

Q&A Support: http://stackoverflow.com/questions/tagged/pandas

Developer Mailing List: http://groups.google.com/group/pydata

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real world** data analysis in Python. Additionally, it has the broader goal of becoming **the most powerful and flexible open source data analysis / manipulation tool available in any language**. It is already well on its way toward this goal.

pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

The two primary data structures of pandas, `Series` (1-dimensional) and `DataFrame` (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users

http://pandas.pydata.org/pandas-docs/version/0.18.0/

pandas:
powerful Python data analysis toolkit

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

Source: http://pandas.pydata.org/pandas-docs/stable/

Series
DataFrame

The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.

Source: http://pandas.pydata.org/pandas-docs/stable/

pandas DataFrame

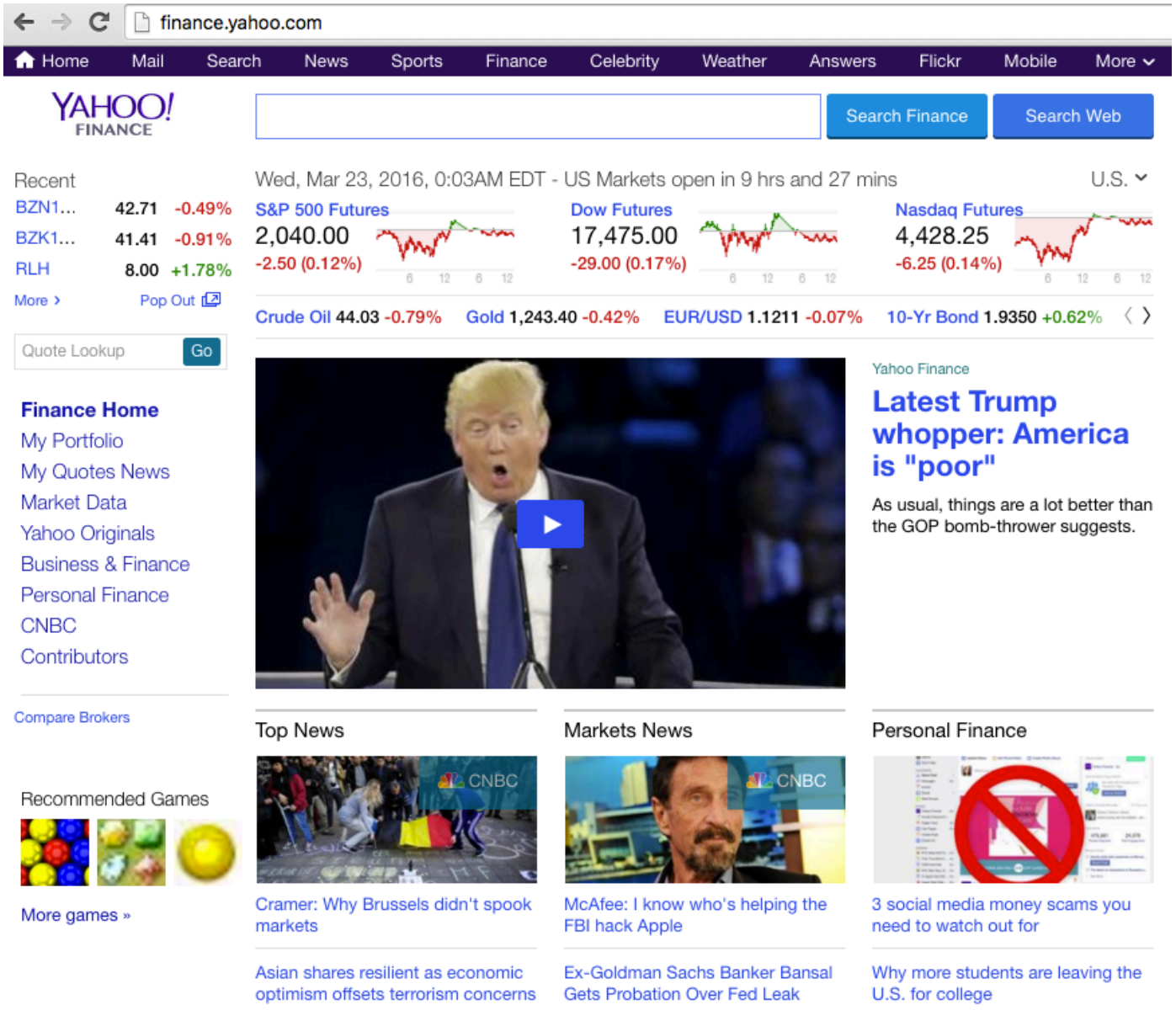
- DataFrame provides everything that R's data.frame provides and much more.
- pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

pandas
Comparison with SAS

pandas	SAS
DataFrame	data set
column	variable
row	observation
groupby	BY-group
NaN	.

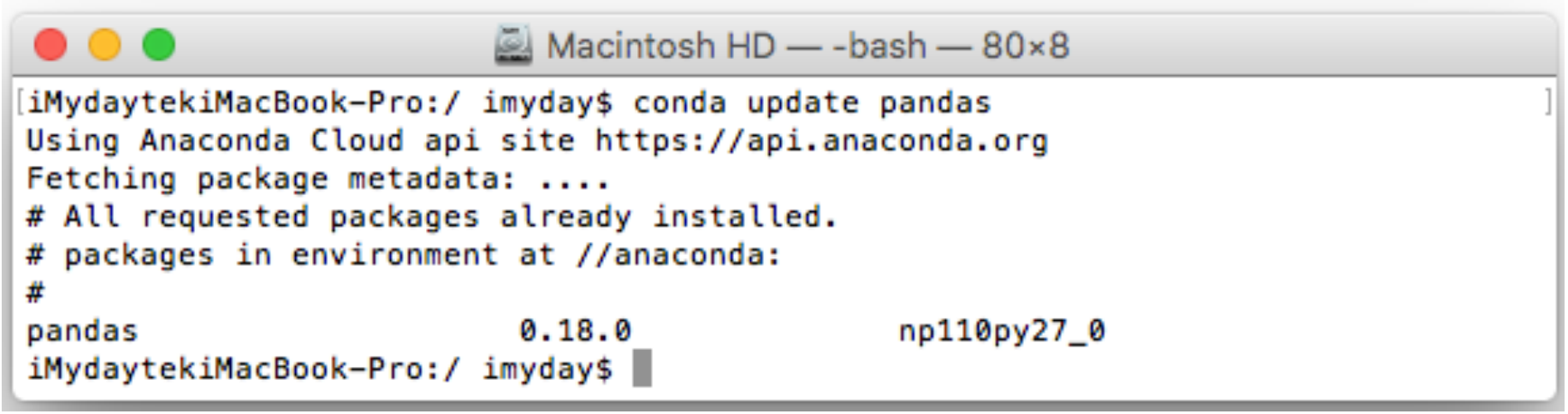
Source: http://pandas.pydata.org/pandas-docs/stable/comparison_with_sas.html

Yahoo Finance



http://finance.yahoo.com/

conda update pandas

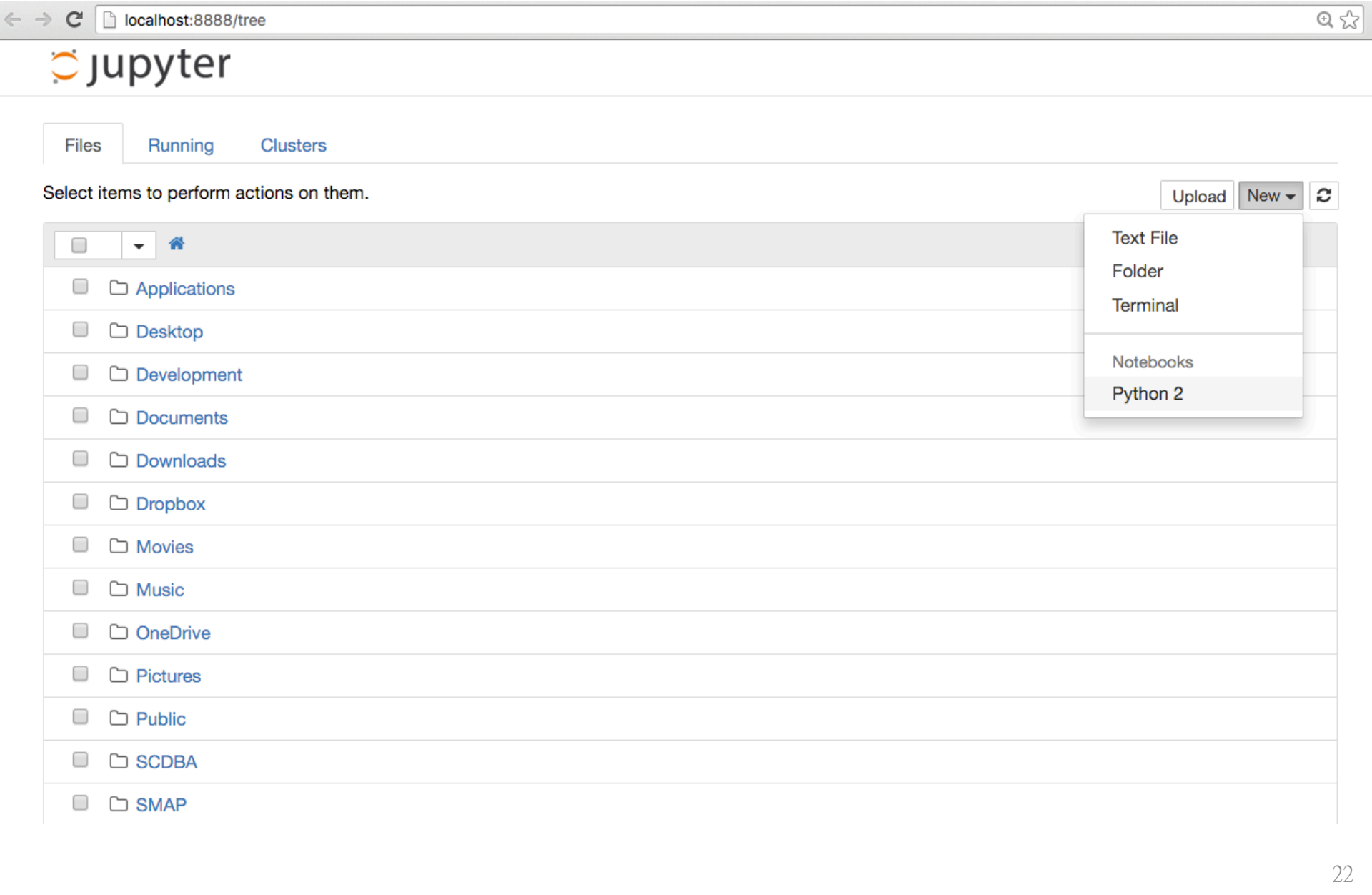


jupyter notebook

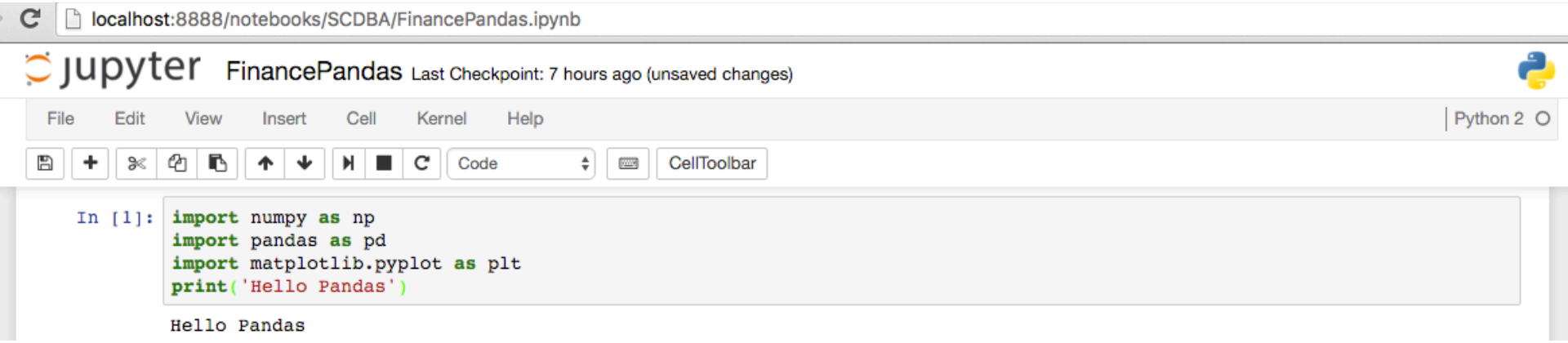
imyday — jupyter-notebook — 90×7

```
[iMydaytekiMacBook-Pro:~ imyday$ jupyter notebook
 [I 05:00:21.870 NotebookApp] Serving notebooks from local directory: /Users/imyday
 [I 05:00:21.870 NotebookApp] 0 active kernels
 [I 05:00:21.870 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/
 [I 05:00:21.870 NotebookApp] Use Control-C to stop this server and shut down all kernels (
twice to skip confirmation).
```

Jupyter New Python 2 Notebook



Jupyter New Python 2 Notebook




```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
print('Hello Pandas')
```

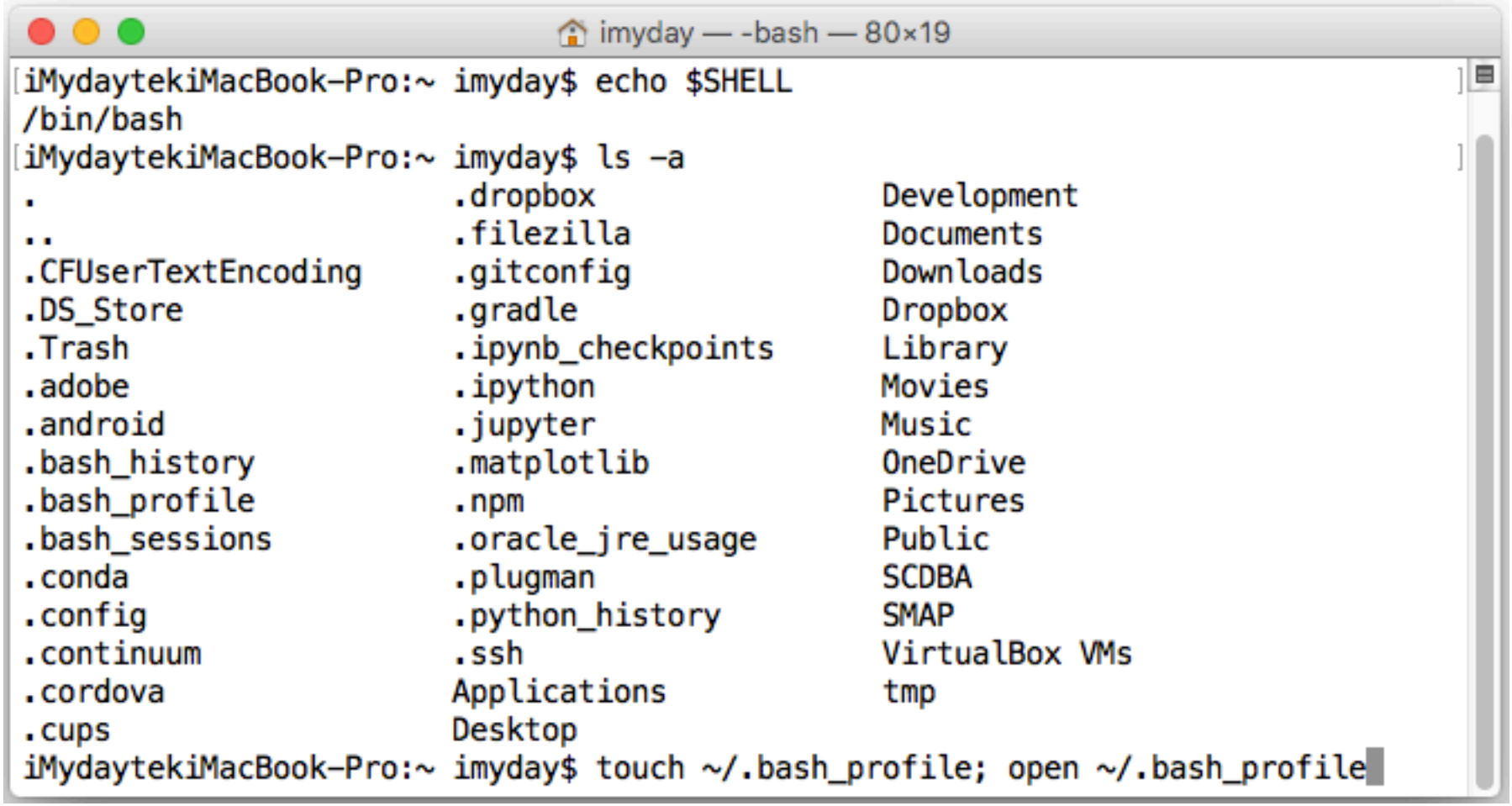
Source: http://pandas.pydata.org/pandas-docs/stable/10min.html

echo $SHELL

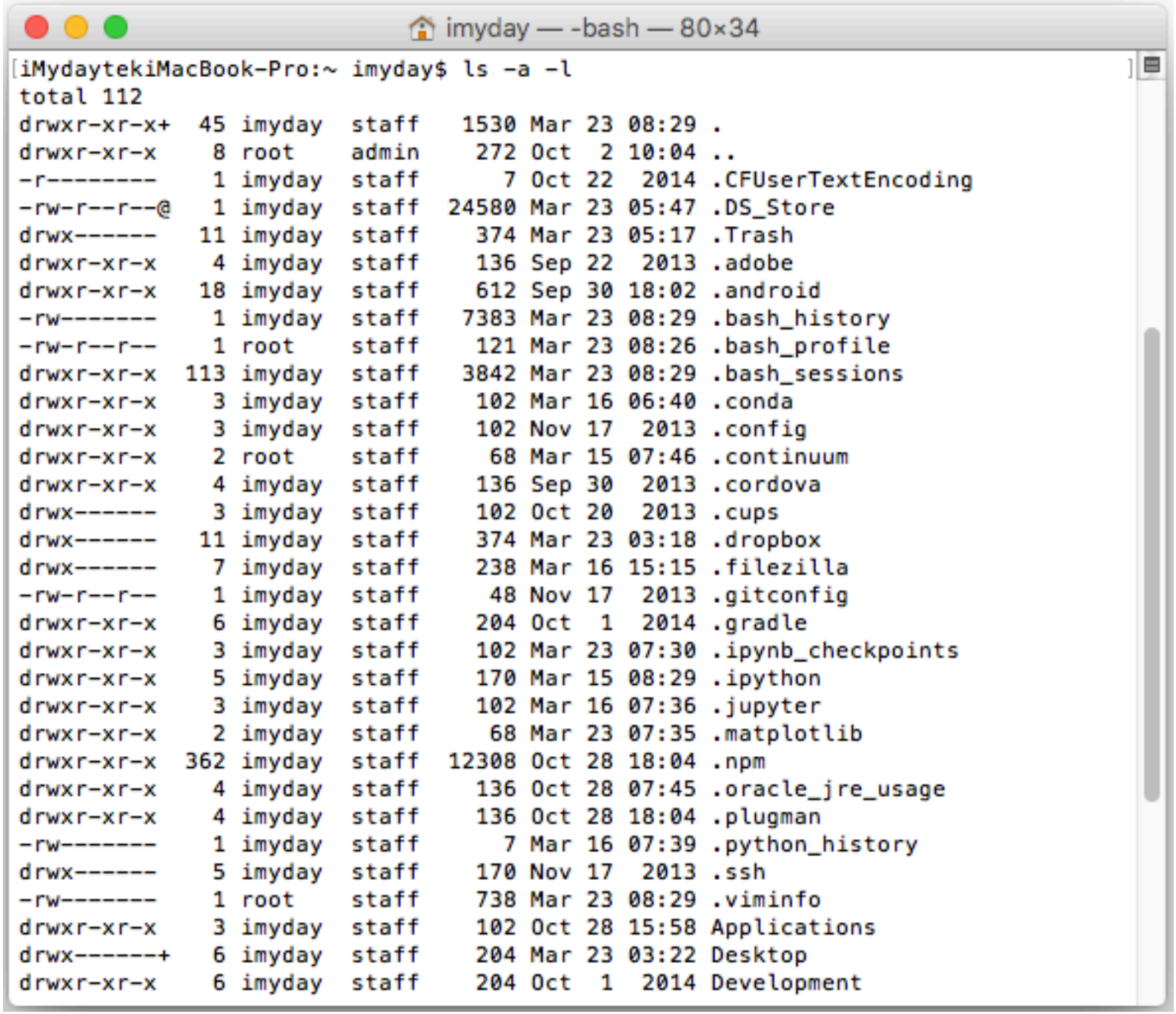
```
iMydaytekiMacBook-Pro:~ imyday$ echo $SHELL
/bin/bash
iMydaytekiMacBook-Pro:~ imyday$
```

```
ls –a
touch ~/.bash_profile; open ~/.bash_profile
```

.bash_profile

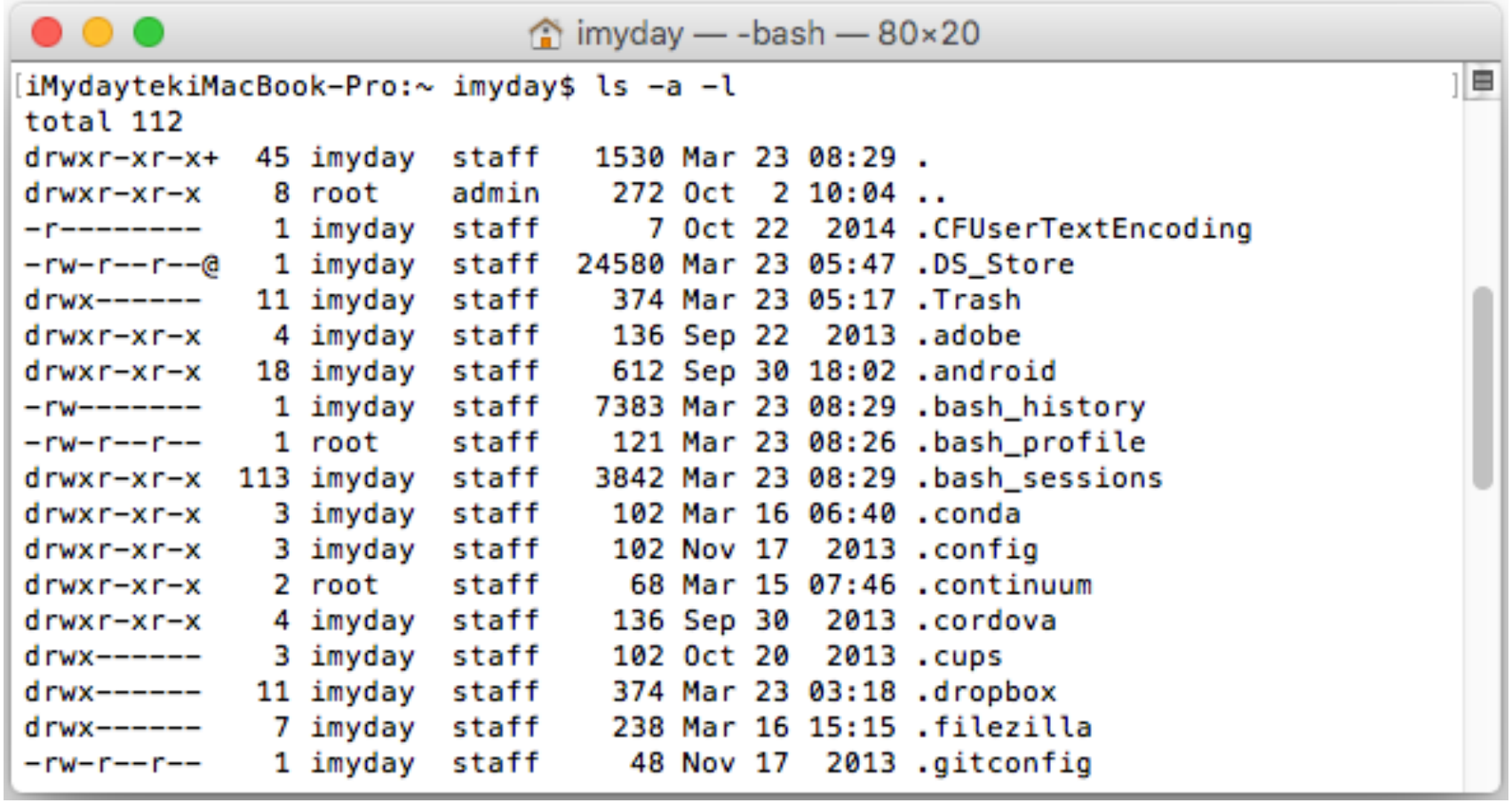


.bash_profile

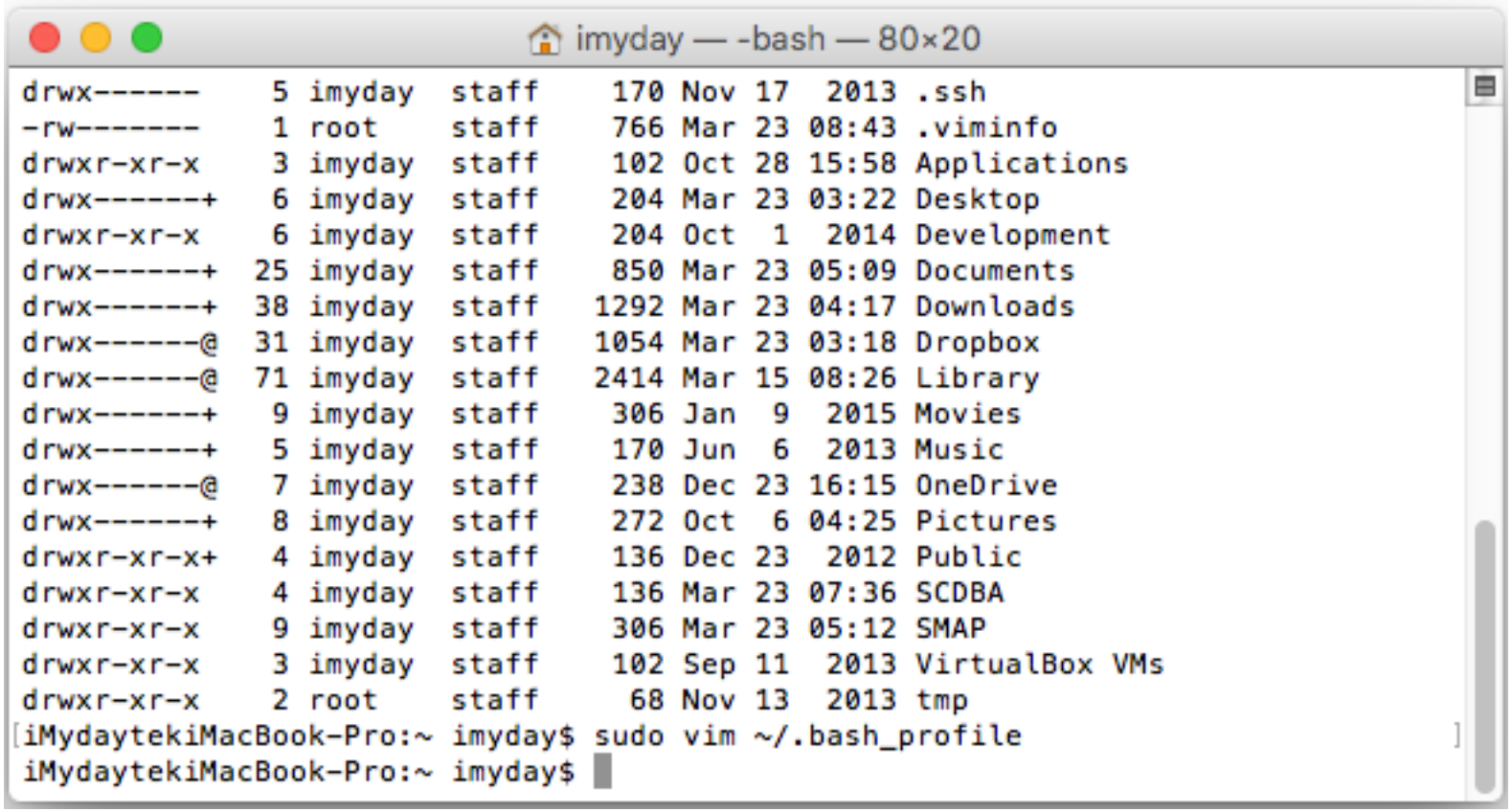

imyday — -bash — 80×34
iMydaytekiMacBook-Pro:~ imyday$ ls -a -l
total 112
drwxr-xr-x+ 45 imyday staff   1530 Mar 23 08:29 .
drwxr-xr-x   8 root   admin   272 Oct 2 10:04 ..
-r--------   1 imyday staff   7 Oct 22 2014 .CFUserTextEncoding
-rw-r--r--@   1 imyday staff 24580 Mar 23 05:47 .DS_Store
drwx------   11 imyday staff   374 Mar 23 05:17 .Trash
drwxr-xr-x   4 imyday staff   136 Sep 22 2013 .adobe
drwxr-xr-x   18 imyday staff   612 Sep 30 18:02 .android
-rw-------   1 imyday staff   7383 Mar 23 08:29 .bash_history
-rw-r--r--   1 root   staff   121 Mar 23 08:26 .bash_profile
drwxr-xr-x 113 imyday staff   3842 Mar 23 08:29 .bash_sessions
drwxr-xr-x   3 imyday staff   102 Mar 16 06:40 .conda
drwxr-xr-x   3 imyday staff   102 Nov 17 2013 .config
drwxr-xr-x   2 root   staff   68 Mar 15 07:46 .continuum
drwxr-xr-x   4 imyday staff   136 Sep 30 2013 .cordova
drwx------   3 imyday staff   102 Oct 20 2013 .cups
drwx------   11 imyday staff   374 Mar 23 03:18 .dropbox
drwx------   7 imyday staff   238 Mar 16 15:15 .filezilla
-rw-r--r--   1 imyday staff   48 Nov 17 2013 .gitconfig
drwxr-xr-x   6 imyday staff   204 Oct 1 2014 .gradle
drwxr-xr-x   3 imyday staff   102 Mar 23 07:30 .ipynb_checkpoints
drwxr-xr-x   5 imyday staff   170 Mar 15 08:29 .ipython
drwxr-xr-x   3 imyday staff   102 Mar 16 07:36 .jupyter
drwxr-xr-x   2 imyday staff   68 Mar 23 07:35 .matplotlib
drwxr-xr-x 362 imyday staff 12308 Oct 28 18:04 .npm
drwxr-xr-x   4 imyday staff   136 Oct 28 07:45 .oracle_jre_usage
drwxr-xr-x   4 imyday staff   136 Oct 28 18:04 .plugman
-rw-------   1 imyday staff   7 Mar 16 07:39 .python_history
drwx------   5 imyday staff   170 Nov 17 2013 .ssh
-rw-------   1 root   staff   738 Mar 23 08:29 .viminfo
drwxr-xr-x   3 imyday staff   102 Oct 28 15:58 Applications
drwx------+   6 imyday staff   204 Mar 23 03:22 Desktop
drwxr-xr-x   6 imyday staff   204 Oct 1 2014 Development

ls –a -l

.bash_profile



$ sudo vim ~/.bash_profile



$ sudo vim ~/.bash_profile

i

<esc>

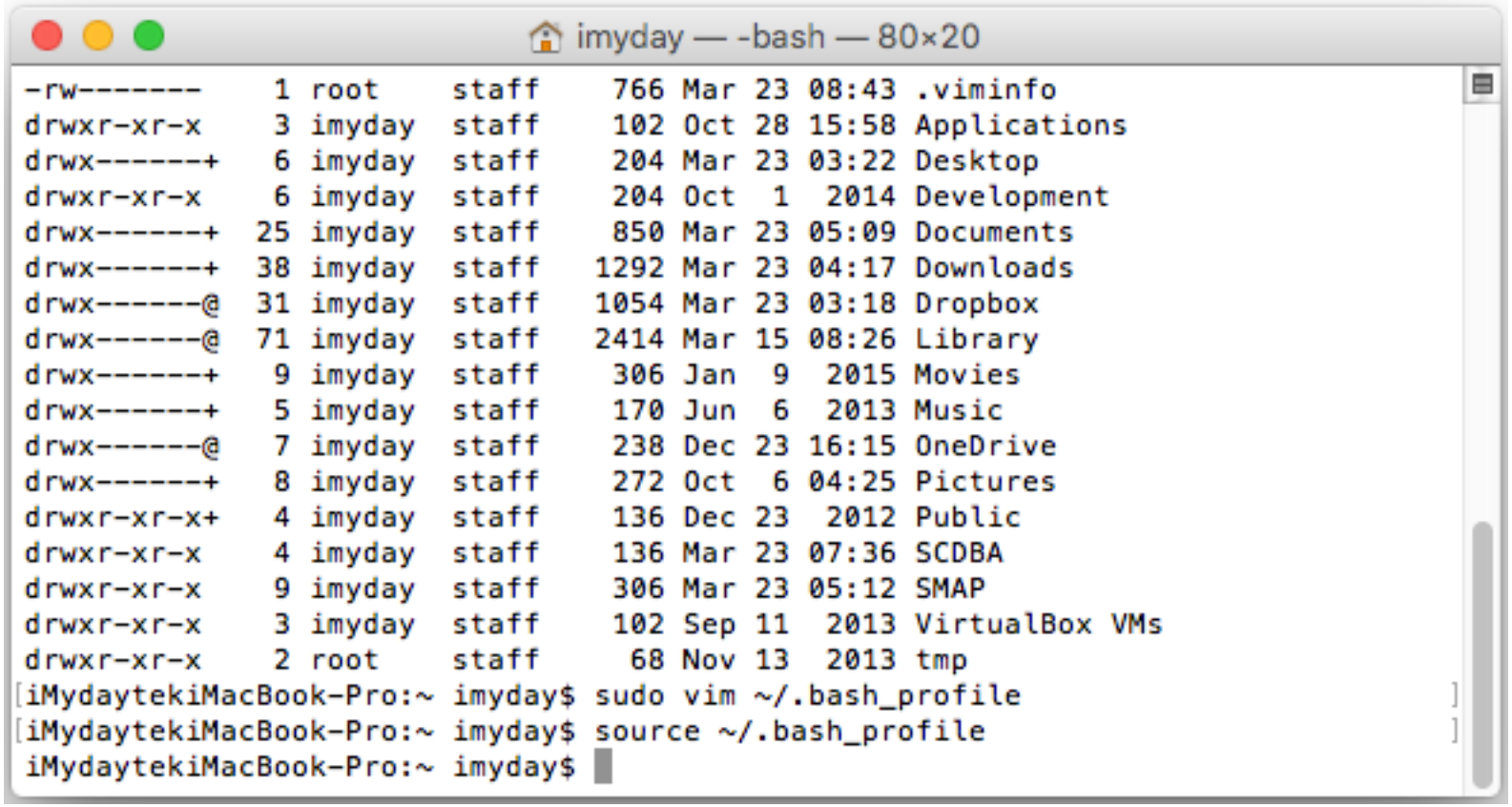
:w

.bash_profile

<esc>

:x

$ source ~/.bash_profile



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
print('Hello Pandas')
```

```
s = pd.Series([1,3,5,np.nan,6,8])
s
```

```
dates = pd.date_range('20160301',
periods=6)
dates
```

Source: http://pandas.pydata.org/pandas-docs/stable/10min.html

localhost:8888/notebooks/SCDBA/FinancePandas.ipynb

jupyter FinancePandas Last Checkpoint: 7 hours ago (autosaved)

File Edit View Insert Cell Kernel Help Python 2

Code CellToolbar

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
print('Hello Pandas')
```

```
Hello Pandas
```

In [2]:
```
s = pd.Series([1,3,5,np.nan,6,8])
s
```

Out[2]:
```
0    1.0
1    3.0
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64
```

In [3]:
```
dates = pd.date_range('20160301', periods=6)
dates
```

Out[3]:
```
DatetimeIndex(['2016-03-01', '2016-03-02', '2016-03-03', '2016-03-04',
               '2016-03-05', '2016-03-06'],
              dtype='datetime64[ns]', freq='D')
```

```
df = pd.DataFrame(np.random.randn(6,4),
index=dates, columns=list('ABCD'))
df
```

In [4]:
```
df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))
df
```

Out[4]:

	A	B	C	D
2016-03-01	-0.825411	0.953499	0.226521	-0.340819
2016-03-02	0.368679	-0.111796	-1.222590	-0.864284
2016-03-03	0.277131	0.459754	0.934857	0.293983
2016-03-04	-0.361468	-0.319544	-1.821719	0.461231
2016-03-05	-0.592124	0.079755	0.996592	-0.390480
2016-03-06	-0.780635	0.961863	-1.149705	0.462091

```
df = pd.DataFrame(np.random.randn(4,6),
index=['student1','student2','student3',
'student4'], columns=list('ABCDEF'))
df
```

```
df = pd.DataFrame(np.random.randn(4,6), index=['student1','student2','student3','student4'], columns=list('ABCDEF'))
df
```

	A	B	C	D	E	F
student1	-1.182365	-0.735508	0.430146	0.835689	-1.418364	-0.085068
student2	0.066564	-1.981319	-0.827381	-1.526019	0.162447	-0.692818
student3	-0.189524	0.289908	1.309277	2.553557	0.952868	-0.151531
student4	-1.526795	1.386375	-0.612484	0.976676	-0.671884	-0.601432

```
df2 = pd.DataFrame({ 'A' : 1.,
'B' : pd.Timestamp('20160323'),
'C' : pd.Series(2.5,index=list(range(4)),dtype='float32'),
'D' : np.array([3] * 4,dtype='int32'),
'E' : pd.Categorical(["test","train","test","train"]),
'F' : 'foo' })
df2
```

In [5]:
```
df2 = pd.DataFrame({ 'A' : 1.,
'B' : pd.Timestamp('20130102'),
'C' : pd.Series(1,index=list(range(4)),dtype='float32'),
'D' : np.array([3] * 4,dtype='int32'),
'E' : pd.Categorical(["test","train","test","train"]),
'F' : 'foo' })
```

In [6]:
```
df2
```

Out[6]:

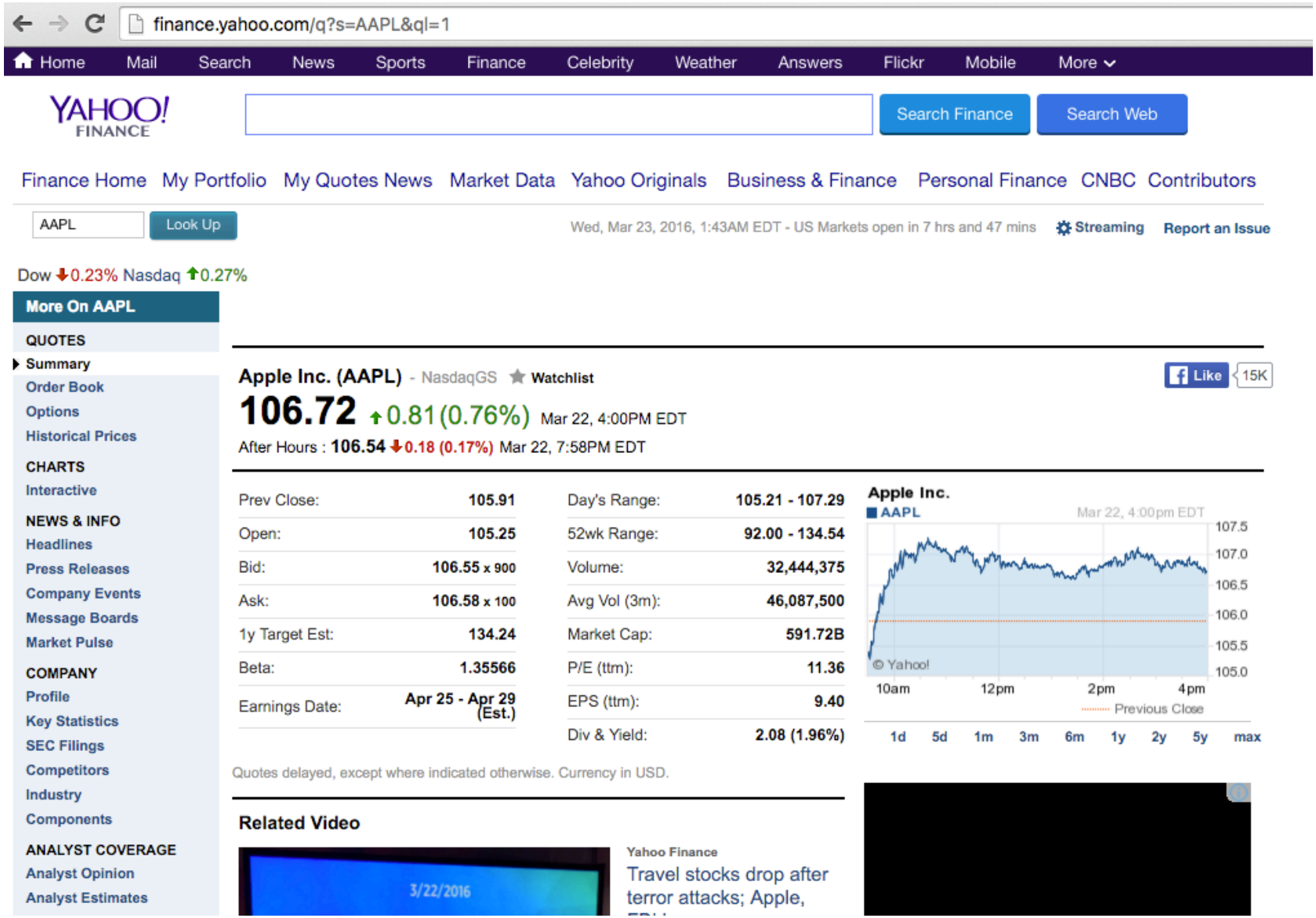
	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	test	foo
1	1.0	2013-01-02	1.0	3	train	foo
2	1.0	2013-01-02	1.0	3	test	foo
3	1.0	2013-01-02	1.0	3	train	foo

df2.dtypes

```
In [7]: df2.dtypes

Out[7]: A           float64
        B    datetime64[ns]
        C           float32
        D             int32
        E          category
        F            object
        dtype: object
```

Apple Inc. (AAPL) -NasdaqGS

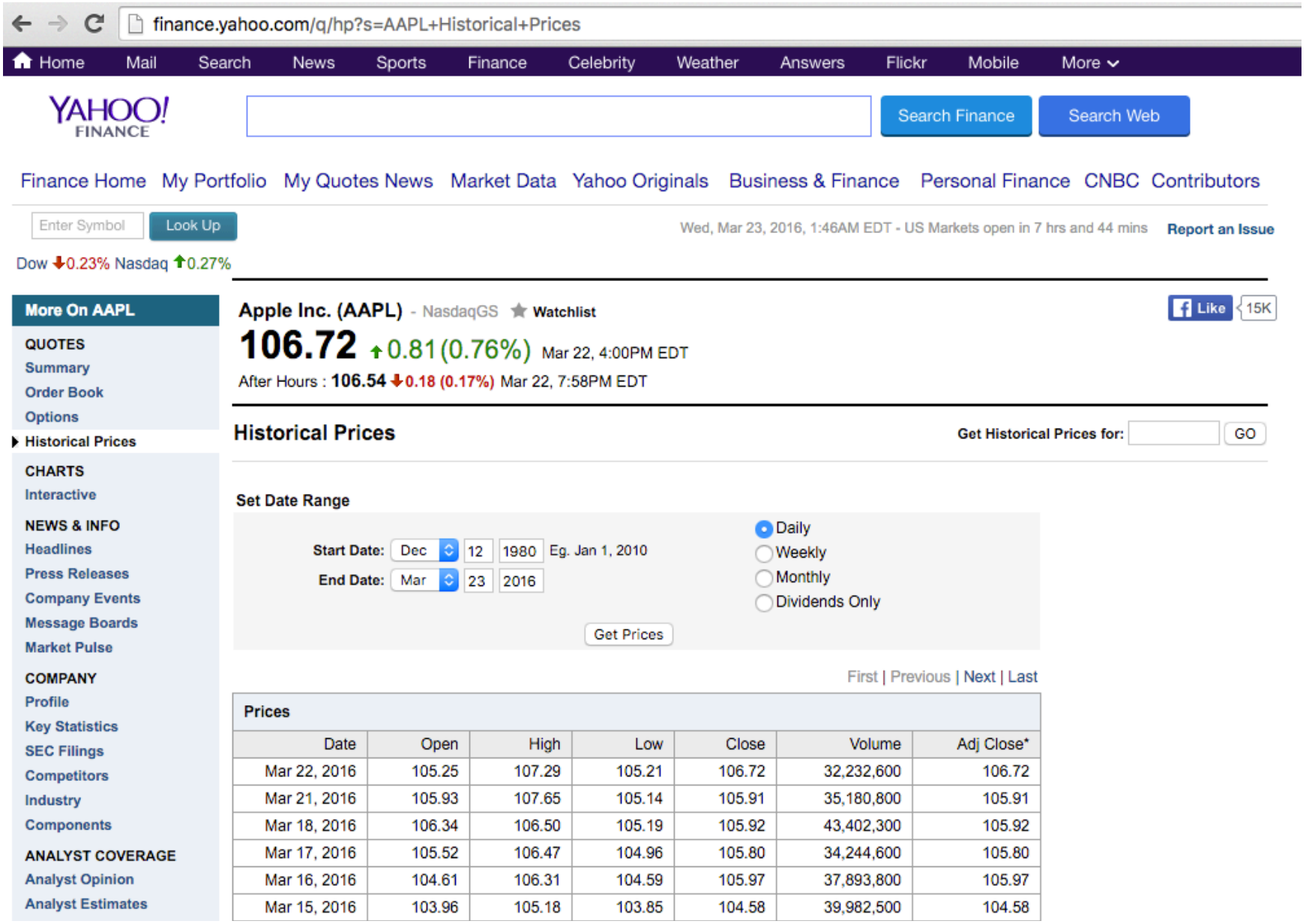


http://finance.yahoo.com/q?s=AAPL&ql=1

Yahoo Finance Charts
Apple Inc. (AAPL)

http://finance.yahoo.com/echarts?s=AAPL+Interactive#{"showArea":false,"showLine":false,"showCandle":true,"lineType":"candle","range":"5y","allowChartStacking":true}

Apple Inc. (AAPL)



http://finance.yahoo.com/q/hp?s=AAPL+Historical+Prices

Yahoo Finance Historical Prices Apple Inc. (AAPL)

First | Previous | Next | Last

Prices						
Date	Open	High	Low	Close	Volume	Adj Close*
Mar 22, 2016	105.25	107.29	105.21	106.72	32,232,600	106.72
Mar 21, 2016	105.93	107.65	105.14	105.91	35,180,800	105.91
Mar 18, 2016	106.34	106.50	105.19	105.92	43,402,300	105.92
Mar 17, 2016	105.52	106.47	104.96	105.80	34,244,600	105.80
Mar 16, 2016	104.61	106.31	104.59	105.97	37,893,800	105.97
Mar 15, 2016	103.96	105.18	103.85	104.58	39,982,500	104.58
Mar 14, 2016	101.91	102.91	101.78	102.52	25,027,400	102.52
Mar 11, 2016	102.24	102.28	101.50	102.26	27,200,800	102.26
Mar 10, 2016	101.41	102.24	100.15	101.17	33,470,400	101.17
Mar 9, 2016	101.31	101.58	100.27	101.12	27,130,700	101.12
Mar 8, 2016	100.78	101.76	100.40	101.03	31,274,200	101.03
Mar 7, 2016	102.39	102.83	100.96	101.87	35,828,900	101.87
Mar 4, 2016	102.37	103.75	101.37	103.01	45,936,500	103.01
Mar 3, 2016	100.58	101.71	100.45	101.50	36,792,200	101.50
Mar 2, 2016	100.51	100.89	99.64	100.75	33,084,900	100.75
Mar 1, 2016	97.65	100.77	97.42	100.53	50,153,900	100.53

http://finance.yahoo.com/q/hp?s=AAPL+Historical+Prices

Yahoo Finance Historical Prices

http://ichart.finance.yahoo.com/table.csv?s=AAPL

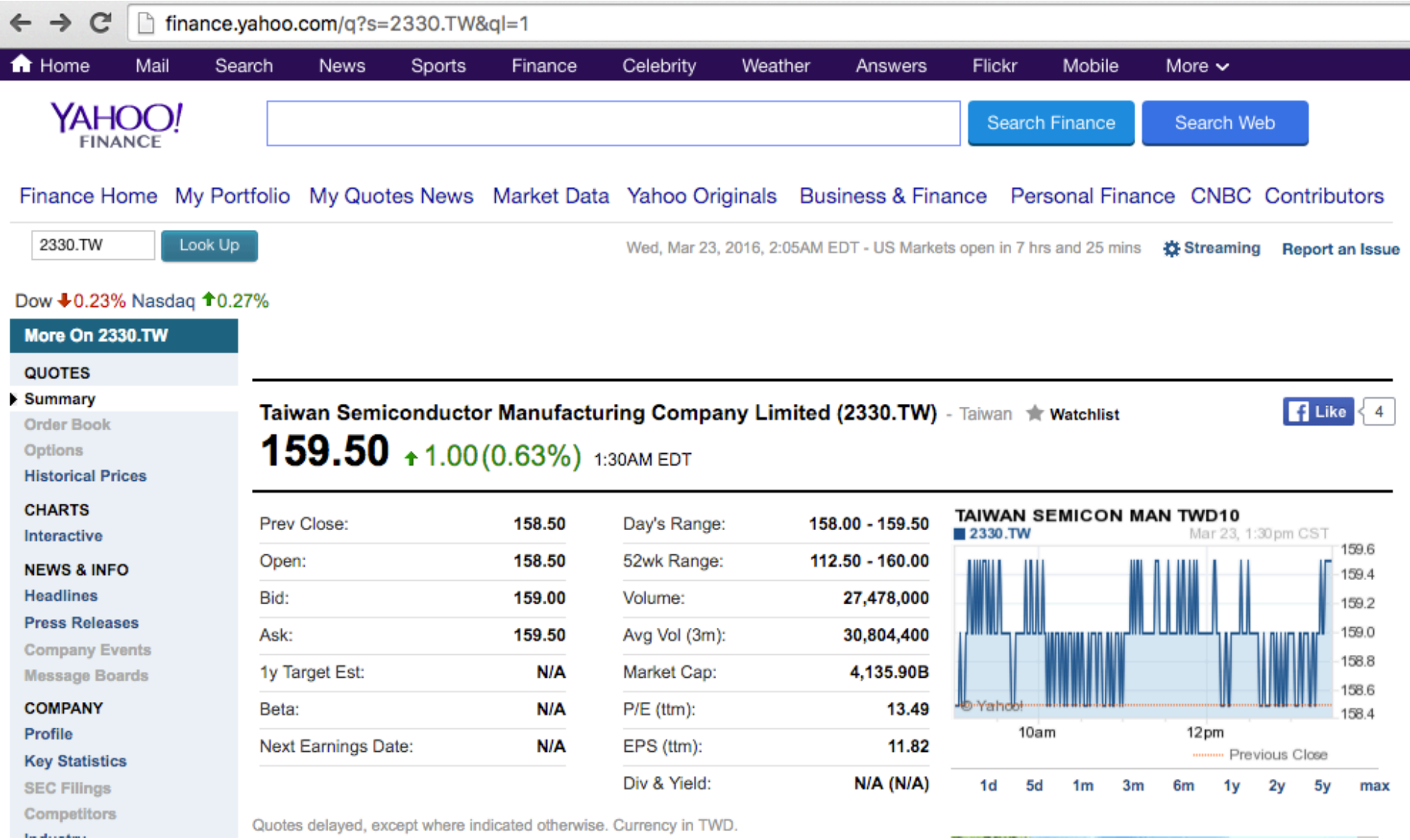
table.csv

```
Date,Open,High,Low,Close,Volume,Adj Close
2016-03-22,105.25,107.290001,105.209999,106.720001,32232600,106.720001
2016-03-21,105.93,107.650002,105.139999,105.910004,35180800,105.910004
2016-03-18,106.339996,106.50,105.190002,105.919998,43402300,105.919998
2016-03-17,105.519997,106.470001,104.959999,105.800003,34244600,105.800003
2016-03-16,104.610001,106.309998,104.589996,105.970001,37893800,105.970001
2016-03-15,103.959999,105.18,103.849998,104.580002,39982500,104.580002
2016-03-14,101.910004,102.910004,101.779999,102.519997,25027400,102.519997
2016-03-11,102.239998,102.279999,101.50,102.260002,27200800,102.260002
2016-03-10,101.410004,102.239998,100.150002,101.169998,33470400,101.169998
2016-03-09,101.309998,101.580002,100.269997,101.120003,27130700,101.120003
2016-03-08,100.779999,101.760002,100.400002,101.029999,31274200,101.029999
2016-03-07,102.389999,102.830002,100.959999,101.870003,35828900,101.870003
2016-03-04,102.370003,103.75,101.370003,103.010002,45936500,103.010002
2016-03-03,100.580002,101.709999,100.449997,101.50,36792200,101.50
2016-03-02,100.510002,100.889999,99.639999,100.75,33084900,100.75
2016-03-01,97.650002,100.769997,97.419998,100.529999,50153900,100.529999
2016-02-29,96.860001,98.230003,96.650002,96.690002,34876600,96.690002
2016-02-26,97.199997,98.019997,96.580002,96.910004,28913200,96.910004
2016-02-25,96.050003,96.760002,95.25,96.760002,27393900,96.760002
2016-02-24,93.980003,96.379997,93.32,96.099998,36155600,96.099998
2016-02-23,96.400002,96.50,94.550003,94.690002,31686700,94.690002
2016-02-22,96.309998,96.900002,95.919998,96.879997,34048200,96.879997
2016-02-19,96.00,96.760002,95.800003,96.040001,34485600,96.040001
2016-02-18,98.839996,98.889999,96.089996,96.260002,38494400,96.260002
2016-02-17,96.669998,98.209999,96.150002,98.120003,44390200,98.120003
```

Yahoo Finance Charts
Alphabet Inc. (GOOG)

http://finance.yahoo.com/echarts?s=GOOG+Interactive#{"showArea":false,"showLine":false,"showCandle":true,"lineType":"candle","range":"5y","allowChartStacking":true}

Taiwan Semiconductor Manufacturing Company Limited (2330.TW)

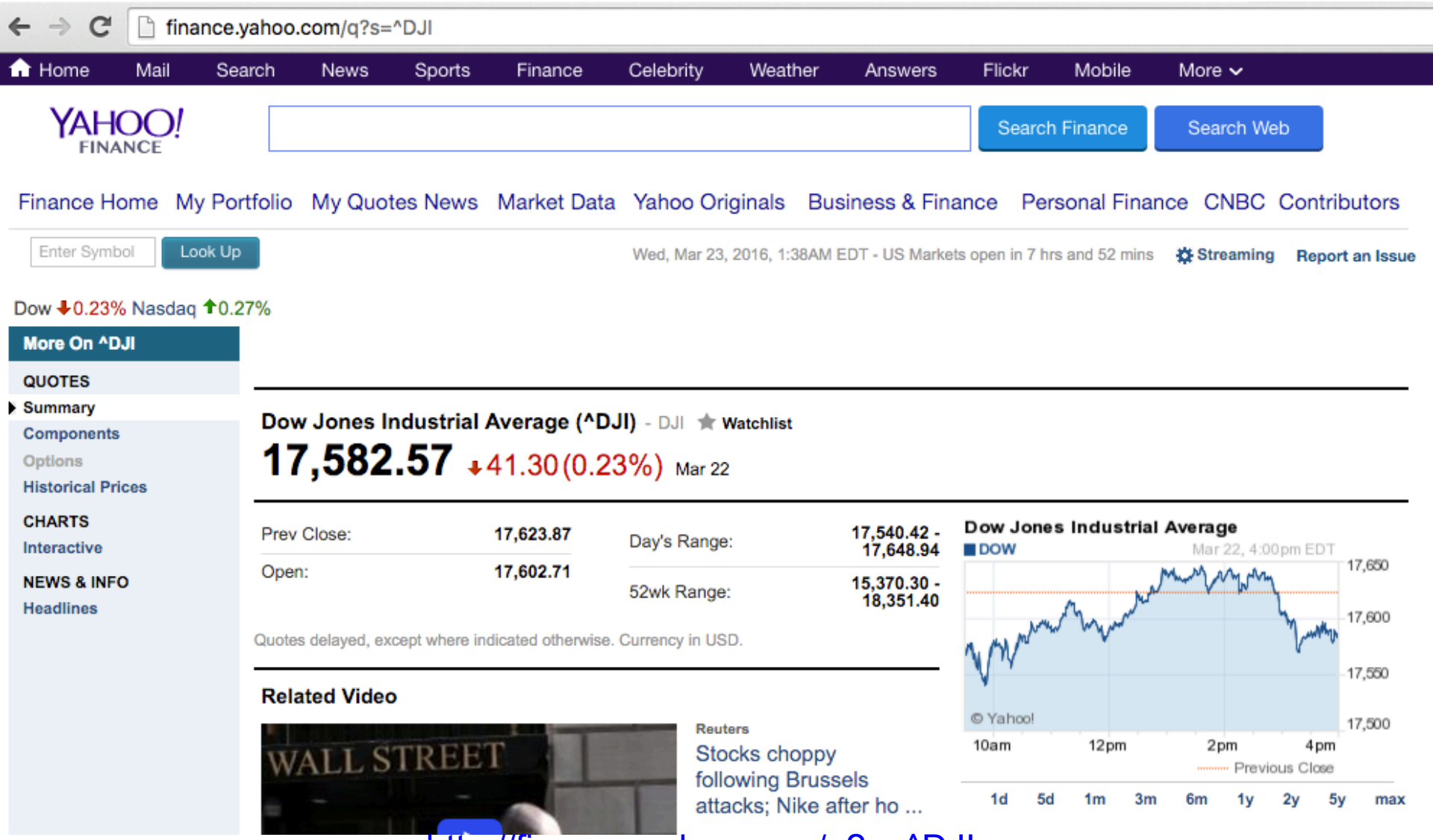


http://finance.yahoo.com/q?s=2330.TW

Yahoo Finance Charts
TSMC (2330.TW)

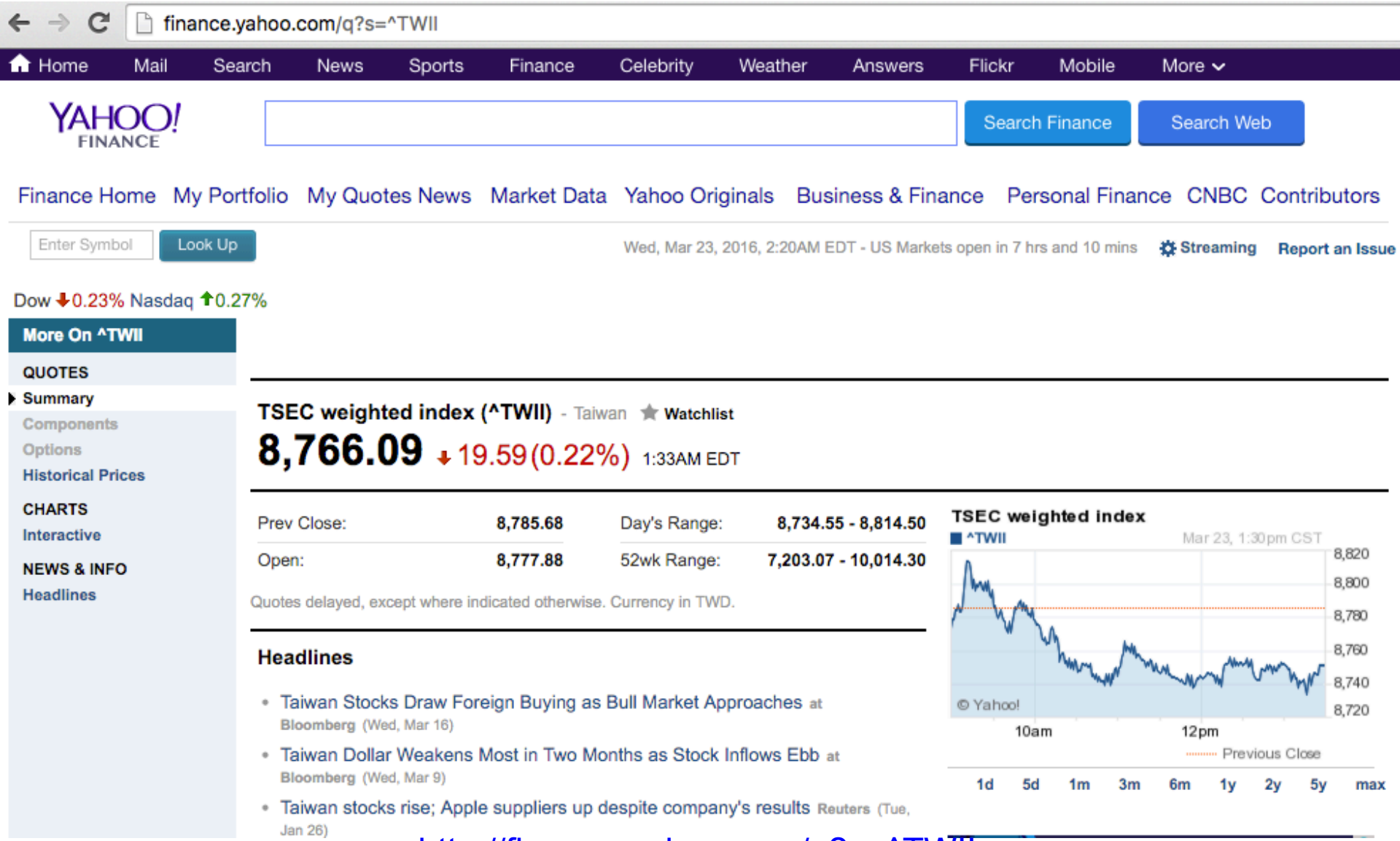
http://finance.yahoo.com/echarts?s=2330.TW+Interactive#{"showArea":false,"showLine":false,"showCandle":true,"lineType":"candle","range":"5y","allowChartStacking":true}

Dow Jones Industrial Average (^DJI)



http://finance.yahoo.com/q?s=^DJI

TSEC weighted index (^TWII) - Taiwan



http://finance.yahoo.com/q?s=^TWII

```
import pandas.io.data as web
#df = web.DataReader('AAPL', 'yahoo')
df = web.DataReader('AAPL', data_source='yahoo',
start='1/1/2010', end='3/22/2016')
#df = web.DataReader('AAPL', data_source='google',
start='1/1/2010', end='3/22/2016')
df.tail()
```

In [9]:
```
import pandas.io.data as web
#df = web.DataReader('AAPL', 'yahoo')
df = web.DataReader('AAPL', data_source='yahoo', start='1/1/2010', end='3/22/2016')
#df = web.DataReader('AAPL', data_source='google', start='1/1/2010', end='3/22/2016')
df.tail()
```

Out[9]:

	Open	High	Low	Close	Volume	Adj Close
Date						
2016-03-16	104.610001	106.309998	104.589996	105.970001	37893800	105.970001
2016-03-17	105.519997	106.470001	104.959999	105.800003	34244600	105.800003
2016-03-18	106.339996	106.500000	105.190002	105.919998	43402300	105.919998
2016-03-21	105.930000	107.650002	105.139999	105.910004	35180800	105.910004
2016-03-22	105.250000	107.290001	105.209999	106.720001	32232600	106.720001

```
df = web.DataReader('GOOG',
data_source='yahoo', start='1/1/1980',
end='3/23/2016')
df.head(10)
```

```
import pandas.io.data as web
```

```
df = web.DataReader('GOOG', data_source='yahoo', start='1/1/1980', end='3/23/2016')
df.head(10)
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2004-08-19	100.000168	104.060182	95.960165	100.340176	44871300	50.119968
2004-08-20	101.010175	109.080187	100.500174	108.310183	22942800	54.100990
2004-08-23	110.750191	113.480193	109.050183	109.400185	18342800	54.645447
2004-08-24	111.240189	111.600192	103.570177	104.870176	15319700	52.382705
2004-08-25	104.960181	108.000187	103.880180	106.000184	9232100	52.947145
2004-08-26	104.950180	107.950188	104.660179	107.910182	7128600	53.901190
2004-08-27	108.100185	108.620186	105.690180	106.150181	6241200	53.022069
2004-08-30	105.280178	105.490184	102.010172	102.010172	5221400	50.954132
2004-08-31	102.300173	103.710180	102.160177	102.370175	4941200	51.133953
2004-09-01	102.700174	102.970180	99.670169	100.250171	9181600	50.075011

df.tail(10)

```
df.tail(10)
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2016-03-09	698.469971	705.679993	694.000000	705.239990	1418700	705.239990
2016-03-10	708.119995	716.440002	703.359985	712.820007	2829400	712.820007
2016-03-11	720.000000	726.919983	717.125000	726.820007	1963900	726.820007
2016-03-14	726.809998	735.500000	725.150024	730.489990	1716900	730.489990
2016-03-15	726.919983	732.289978	724.770020	728.330017	1720100	728.330017
2016-03-16	726.369995	737.469971	724.510010	736.090027	1572300	736.090027
2016-03-17	736.450012	743.070007	736.000000	737.780029	1856800	737.780029
2016-03-18	741.859985	742.000000	731.830017	737.599976	2796400	737.599976
2016-03-21	736.500000	742.500000	733.515991	742.090027	1831800	742.090027
2016-03-22	737.460022	745.000000	737.460022	740.750000	1264400	740.750000

df2.count()

```
df.count()
```

```
Open          2919
High          2919
Low           2919
Close         2919
Volume        2919
Adj Close     2919
dtype: int64
```

```
import pandas.io.data as web
import datetime
start = datetime.datetime(2010, 1, 1)
end = datetime.datetime(2015, 12, 31)
f = web.DataReader(”F", 'yahoo', start, end)
f.ix['2015-12-31']
```

In [10]:
```
import pandas.io.data as web
import datetime
start = datetime.datetime(2010, 1, 1)
end = datetime.datetime(2015, 12, 31)
f = web.DataReader("F", 'yahoo', start, end)
f.ix['2015-12-31']
```

Out[10]:
```
Open          1.414000e+01
High          1.416000e+01
Low           1.404000e+01
Close         1.409000e+01
Volume        1.988100e+07
Adj Close     1.363029e+01
Name: 2015-12-31 00:00:00, dtype: float64
```

```
import pandas.io.data as web
import datetime
start = datetime.datetime(2010, 1, 1)
end = datetime.datetime(2016, 3, 23)
f = web.DataReader("2330.TW", 'yahoo', start, end)
f.ix['2015-12-31']
```


```
import pandas.io.data as web
import datetime
start = datetime.datetime(2010, 1, 1)
end = datetime.datetime(2016, 3, 23)
f = web.DataReader("2330.TW", 'yahoo', start, end)
f.ix['2015-12-31']
```


```
Open              142.0
High              144.0
Low               141.0
Close             143.0
Volume        18336000.0
Adj Close         143.0
Name: 2015-12-31 00:00:00, dtype: float64
```

```
f.to_csv('2330.TW.Yahoo.Finance.Data.csv')
```

2330.TW.Yahoo.Finance.Data.csv

```
Date,Open,High,Low,Close,Volume,Adj Close
2010-01-01,64.5,64.5,64.5,64.5,0,52.8308
2010-01-04,65.0,65.0,64.0,64.9,39407000,53.1584
2010-01-05,65.0,65.1,63.9,64.5,37138000,52.8308
2010-01-06,64.5,64.9,63.7,64.9,49261000,53.1584
2010-01-07,64.9,65.0,64.2,64.2,42134000,52.5851
2010-01-08,63.5,64.3,63.5,64.0,46076000,52.4213
2010-01-11,64.0,64.9,63.5,64.5,36799000,52.8308
2010-01-12,64.4,64.4,63.3,63.6,49853000,52.0936
2010-01-13,63.0,63.1,62.6,62.8,47976000,51.4384
2010-01-14,63.6,63.6,63.0,63.2,36149000,51.766
2010-01-15,62.9,63.5,62.8,63.5,47852000,52.0117
2010-01-18,62.8,63.1,62.8,62.9,30136000,51.5203
2010-01-19,63.0,63.2,62.0,62.5,47202000,51.1926
2010-01-20,62.9,63.2,62.2,63.0,52281000,51.6022
```

```
sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"

# sURL = "http://ichart.finance.yahoo.com/table.csv?s=AAPL"
# sBaseURL = "http://ichart.finance.yahoo.com/table.csv?s="
sURL = "http://ichart.finance.yahoo.com/table.csv?s=" + sSymbol
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=2330.TW")
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=AAPL")
req = requests.get(sURL)

sText = req.text
#print(sText)
#df = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
#df = web.DataReader("2330.TW", 'yahoo')

sPath = "data/"
sPathFilename = sPath + sSymbol + ".csv"
print(sPathFilename)

f = open(sPathFilename, 'w')
f.write(sText)
f.close()
sIOdata = io.StringIO(sText)
df = pd.DataFrame.from_csv(sIOdata)
df.head(5)
```

In [11]:

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

import pandas.io.data as web
import random

import datetime
import time
import timeit

import io
import os

import re
import codecs
import requests
get_ipython().magic('matplotlib inline')

from scipy import stats

#pd.set_option('display.notebook_repr_html', False)
pd.set_option('display.max_columns', 15)
pd.set_option('display.max_rows', 10)
pd.set_option('precision', 3)

def getDateTimeNow():
    strnow = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
    return strnow

print('Hello Pandas')
print(getDateTimeNow())
```

```
Hello Pandas
20160323_145708
```

```
sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"

# sURL = "http://ichart.finance.yahoo.com/table.csv?s=AAPL"
# sBaseURL = "http://ichart.finance.yahoo.com/table.csv?s="
sURL = "http://ichart.finance.yahoo.com/table.csv?s=" + sSymbol
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=2330.TW")
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=AAPL")
req = requests.get(sURL)

sText = req.text
#print(sText)
#df = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
#df = web.DataReader("2330.TW", 'yahoo')

sPath = "data/"
sPathFilename = sPath + sSymbol + ".csv"
print(sPathFilename)

f = open(sPathFilename, 'w')
f.write(sText)
f.close()
sIOdata = io.StringIO(sText)
df = pd.DataFrame.from_csv(sIOdata)
df.head(5)
```

In [13]:
```
starttime = datetime.datetime(2000, 1, 1)
endtime = datetime.datetime(2015, 12, 31)
sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"
#"^TWII"
#"AAPL"
#SHA:000016"
#"600000.SS"
#"2330.TW"
df = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
#df_01 = web.DataReader("2330.TW", 'yahoo')
sSymbol = sSymbol.replace(":","_")
sSymbol = sSymbol.replace("^","_")
sPath = "data/financedata/"
#sPath = "/users/imyday/SCDBA/data/"  #Mac OS X
#sPath = "C:\data\"                   #Windows

sPathFilename = sPath + sSymbol + "_Yahoo_Finance.csv"
print(sPathFilename)
df.to_csv(sPathFilename)
df.head(5)
```

```
data/financedata/AAPL_Yahoo_Finance.csv
```

Out[13]:

	Open	High	Low	Close	Volume	Adj Close
Date						
2000-01-03	104.875	112.500	101.688	111.938	133949200	3.702
2000-01-04	108.250	110.625	101.188	102.500	128094400	3.390

df.tail(5)

```
In [14]: df.tail(5)
```

Out[14]:

	Open	High	Low	Close	Volume	Adj Close
Date						
2015-12-24	109.00	109.00	107.95	108.03	13596700	107.447
2015-12-28	107.59	107.69	106.18	106.82	26704200	106.243
2015-12-29	106.96	109.43	106.86	108.74	30931200	108.153
2015-12-30	108.58	108.70	107.18	107.32	25213800	106.741
2015-12-31	107.01	107.03	104.82	105.26	40912300	104.692

```
sSymbol = "AAPL"

# sURL = "http://ichart.finance.yahoo.com/table.csv?s=AAPL"
sURL = "http://ichart.finance.yahoo.com/table.csv?s=" + sSymbol
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=AAPL")
req = requests.get(sURL)

sText = req.text
#print(sText)

sPath = "data/"
sPathFilename = sPath + sSymbol + ".csv"
print(sPathFilename)

f = open(sPathFilename, 'w')
f.write(sText)
f.close()
sIOdata = io.StringIO(sText)
df = pd.DataFrame.from_csv(sIOdata)
df.head(5)
```

In [15]:
```
sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"

# sURL = "http://ichart.finance.yahoo.com/table.csv?s=AAPL"
# sBaseURL = "http://ichart.finance.yahoo.com/table.csv?s="
sURL = "http://ichart.finance.yahoo.com/table.csv?s=" + sSymbol
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=2330.TW")
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=AAPL")
req = requests.get(sURL)

sText = req.text
#print(sText)
#df = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
#df = web.DataReader("2330.TW", 'yahoo')

sPath = "data/"
sPathFilename = sPath + sSymbol + ".csv"
print(sPathFilename)

f = open(sPathFilename, 'w')
f.write(sText)
f.close()
sIOdata = io.StringIO(sText)
df = pd.DataFrame.from_csv(sIOdata)
df.head(5)
```

```
data/AAPL.csv
```

data/AAPL.csv

Out[15]:

	Open	High	Low	Close	Volume	Adj Close
Date						
2016-03-22	105.25	107.29	105.21	106.72	32232600	106.72
2016-03-21	105.93	107.65	105.14	105.91	35180800	105.91
2016-03-18	106.34	106.50	105.19	105.92	43402300	105.92
2016-03-17	105.52	106.47	104.96	105.80	34244600	105.80
2016-03-16	104.61	106.31	104.59	105.97	37893800	105.97

```
def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
    #GetMarketFinanceData_From_YahooFinance
    #"^TWII"
    #"000001.SS"
    #"AAPL"
    #SHA:000016"
    #"600000.SS"
    #"2330.TW"
    #sSymbol = "^TWII"
    starttime = datetime.datetime(2000, 1, 1)
    endtime = datetime.datetime(2015, 12, 31)
    sPath = sDir
    #sPath = "data/financedata/"
   df_YahooFinance = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
    #df_01 = web.DataReader("2330.TW", 'yahoo')
    sSymbol = sSymbol.replace(":","_")
    sSymbol = sSymbol.replace("^","_")
    sPathFilename = sPath + sSymbol + "_Yahoo_Finance.csv"
    df_YahooFinance.to_csv(sPathFilename)
    #df_YahooFinance.head(5)
    return sPathFilename
    #End def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
```

In [16]:
```
def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
    #GetMarketFinanceData_From_YahooFinance
    #"^TWII"
    #"000001.SS"
    #"AAPL"
    #SHA:000016"
    #"600000.SS"
    #"2330.TW"
    #sSymbol = "^TWII"
    starttime = datetime.datetime(2000, 1, 1)
    endtime = datetime.datetime(2015, 12, 31)
    sPath = sDir
    #sPath = "data/financedata/"
    df_YahooFinance = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
    #df_01 = web.DataReader("2330.TW", 'yahoo')
    sSymbol = sSymbol.replace(":","_")
    sSymbol = sSymbol.replace("^","_")
    sPathFilename = sPath + sSymbol + "_Yahoo_Finance.csv"
    df_YahooFinance.to_csv(sPathFilename)
    #df_YahooFinance.head(5)
    return sPathFilename
    #End def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
```

```
sSymbol = "AAPL”
starttime = datetime.datetime(2000, 1, 1)
endtime = datetime.datetime(2015, 12, 31)
sDir = "data/financedata/"

sPathFilename = getYahooFinanceData(sSymbol, starttime, endtime, sDir)
print(sPathFilename)
```

In [17]:
```
sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"
starttime = datetime.datetime(2000, 1, 1)
endtime = datetime.datetime(2015, 12, 31)
sDir = "data/financedata/"

sPathFilename = getYahooFinanceData(sSymbol, starttime, endtime, sDir)
print(sPathFilename)
```

```
data/financedata/AAPL_Yahoo_Finance.csv
```

References

- Wes McKinney (2012), Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, O'Reilly Media
- Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
- Yves Hilpisch (2015), Derivatives Analytics with Python: Data Analysis, Models, Simulation, Calibration and Hedging, Wiley
- Michael Heydt (2015) , Mastering Pandas for Finance, Packt Publishing
- Michael Heydt (2015), Learning Pandas - Python Data Discovery and Analysis Made Easy, Packt Publishing
- James Ma Weiming (2015), Mastering Python for Finance, Packt Publishing
- Fabio Nelli (2015), Python Data Analytics: Data Analysis and Science using PANDAs, matplotlib and the Python Programming Language, Apress
- Wes McKinney (2013), 10-minute tour of pandas, https://vimeo.com/59324550
- Jason Wirth (2015), A Visual Guide To Pandas, https://www.youtube.com/watch?v=9d5-Ti6onew
- Edward Schofield (2013), Modern scientific computing and big data analytics in Python, PyCon Australia, https://www.youtube.com/watch?v=hqOsfS3dP9w