財務金融大數據分析
Big Data Analytics in Finance

Tamkang University
淡江大學

Python Pandas 財務大數據分析 (Finance Big Data Analytics with Pandas in Python)

1061BDAF08
MIS EMBA (M2322) (8605)
Thu 12,13,14 (19:20-22:10) (D503)

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/

2017-11-16

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2017/09/21	財務金融大數據分析課程介紹 (Course Orientation for Big Data Analytics in Finance)
2	2017/09/28	金融科技商業模式 (Business Models of Fintech)
3	2017/10/05	人工智慧投資分析與機器人理財顧問 (Artificial Intelligence for Investment Analysis and Robo-Advisors)
4	2017/10/12	金融科技對話式商務與智慧型交談機器人 (Conversational Commerce and Intelligent Chatbots for Fintech)
5	2017/10/19	事件研究法 (Event Study)
6	2017/10/26	財務金融大數據分析個案研究 I (Case Study on Big Data Analytics in Finance I)

課程大綱 (Syllabus)

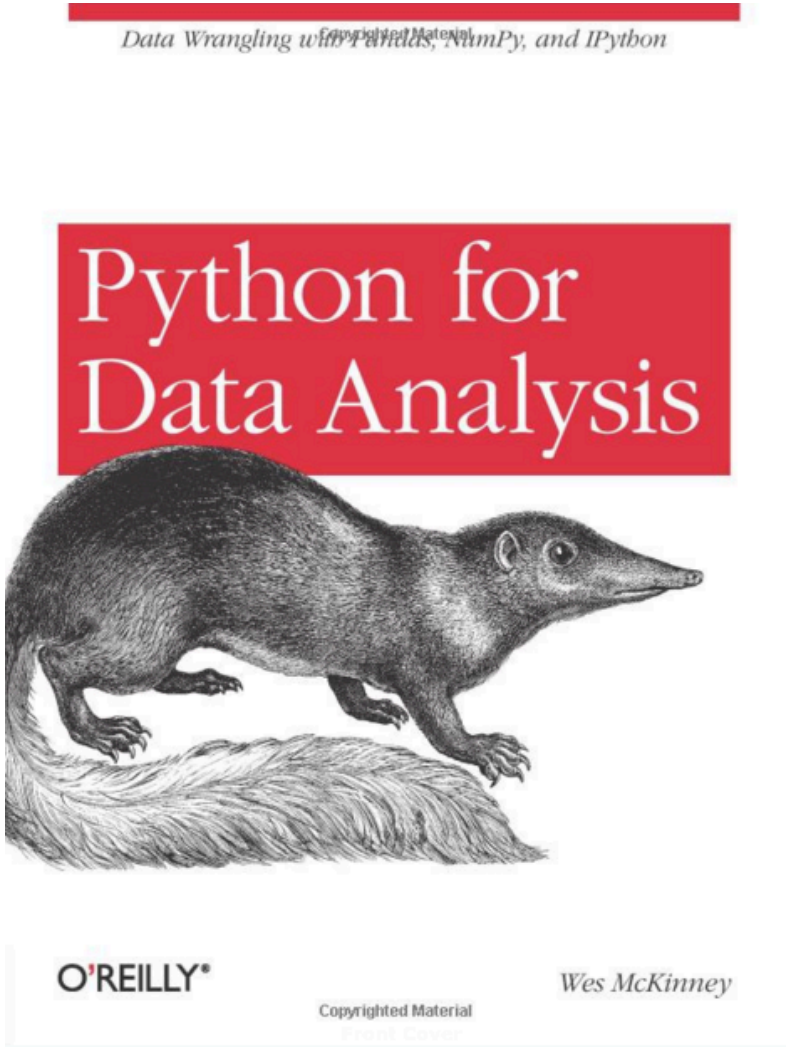
週次 (Week)	日期 (Date)	內容 (Subject/Topics)
7	2017/11/02	Python 財務大數據分析基礎 (Foundations of Finance Big Data Analytics in Python)
8	2017/11/09	Python Numpy大數據分析 (Big Data Analytics with Numpy in Python)
9	2017/11/16	Python Pandas 財務大數據分析 (Finance Big Data Analytics with Pandas in Python)
10	2017/11/23	期中報告 (Midterm Project Report)
11	2017/11/30	文字探勘分析技術與自然語言處理 (Text Mining Techniques and Natural Language Processing)
12	2017/12/07	Python Keras深度學習 (Deep Learning with Keras in Python)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
13	2017/12/14	財務金融大數據分析個案研究 II (Case Study on Big Data Analytics in Finance II)
14	2017/12/21	TensorFlow深度學習 (Deep Learning with TensorFlow)
15	2017/12/28	財務金融大數據深度學習 (Deep Learning for Finance Big Data)
16	2018/01/04	社會網絡分析 (Social Network Analysis)
17	2018/01/11	期末報告 I (Final Project Presentation I)
18	2018/01/18	期末報告 II (Final Project Presentation II)

Finance
Big Data Analytics with Pandas in Python

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

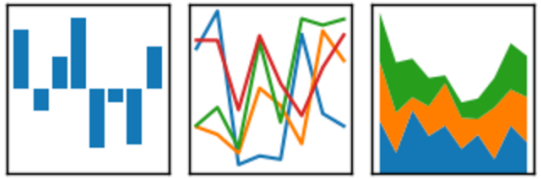


Source: http://www.amazon.com/Python-Data-Analysis-Wrangling-IPython/dp/1449319793/

pandas

pandas

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



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.

pandas is a NUMFocus sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project.

A Fiscally Sponsored Project of
NUMFOCUS
OPEN CODE = BETTER SCIENCE

0.19.2 Final (December 24, 2016)

This is a minor bug-fix release in the 0.19.x series and includes some small regression fixes, bug fixes and performance improvements.

Highlights include:

- Compatibility with Python 3.6

VERSIONS

Release
0.19.2 - December 2016
download // docs // pdf

Development
0.20.0 - 2017
github // docs

Previous Releases
0.19.1 - download // docs // pdf
0.19.0 - download // docs // pdf
0.18.1 - download // docs // pdf
0.18.0 - download // docs // pdf
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

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 Ecosystem

- Statistics and Machine Learning
  - Statsmodels
  - sklearn-pandas
- Visualization
  - Bokeh
  - yhat/ggplot
  - Seaborn
  - Vincent
  - IPython Vega
  - Plotly
  - Pandas-Qt
- IDE
  - IPython
  - quantopian/qgrid
  - Spyder
- API
  - pandas-datareader
  - quandl/Python
  - pydatastream
  - pandaSDMX
  - fredapi
- Domain Specific
  - Geopandas
  - xarray
- Out-of-core
  - Dask
  - Blaze
  - Odo

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

pandas: powerful Python data analysis toolkit

pandas 0.19.2 documentation »

Table Of Contents

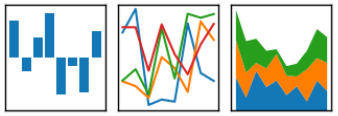
pandas: powerful Python data analysis toolkit

PDF Version

Zipped HTML

pandas

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



Date: Dec 24, 2016 **Version**: 0.19.2

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, `DataFrame` provides everything that R's `data.frame` provides and much more. pandas is built on top of NumPy and is

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

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

- Primary data structures of pandas
  - Series (1-dimensional)
  - 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	.

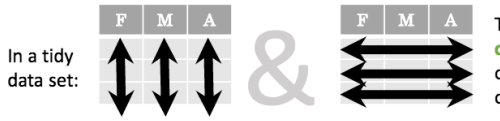
Source: http://pandas.pydata.org/pandas-docs/stable/comparison_with_sas.html

Python Pandas Cheat Sheet

Data Wrangling with pandas Cheat Sheet

http://pandas.pydata.org

Tidy Data – A foundation for wrangling in pandas



Each **variable** is saved in its own **column**

Each **observation** is saved in its own **row**

Tidy data complements pandas's **vectorized operations**. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.

M * A

Syntax – Creating DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
       index = [1, 2, 3])
```

Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```

Specify values for each row.

		a	b	c
n	v			
d	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
        [('d',1),('d',2),('e',2)],
          names=['n','v'])))
```

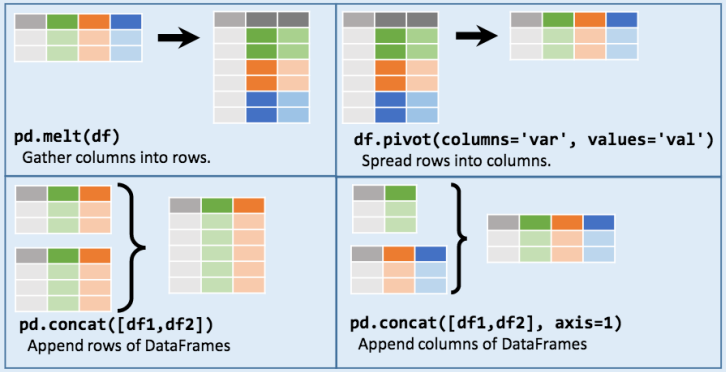
Create DataFrame with a MultiIndex

Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
        .rename(columns={
                'variable' : 'var',
                'value' : 'val'})
        .query('val >= 200')
    )
```

Reshaping Data – Change the layout of a data set



pd.melt(df)
Gather columns into rows.

df.pivot(columns='var', values='val')
Spread rows into columns.

pd.concat([df1,df2])
Append rows of DataFrames

pd.concat([df1,df2], axis=1)
Append columns of DataFrames

df=df.sort_values('mpg')
Order rows by values of a column (low to high).

df=df.sort_values('mpg',ascending=False)
Order rows by values of a column (high to low).

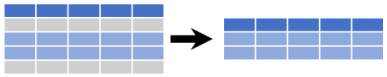
df=df.rename(columns = {'y':'year'})
Rename the columns of a DataFrame

df=df.sort_index()
Sort the index of a DataFrame

df=df.reset_index()
Reset index of DataFrame to row numbers, moving index to columns.

df=df.drop(['Length','Height'], axis=1)
Drop columns from DataFrame

Subset Observations (Rows)



df[df.Length > 7]
Extract rows that meet logical criteria.

df.drop_duplicates()
Remove duplicate rows (only considers columns).

df.head(n)
Select first n rows.

df.tail(n)
Select last n rows.

df.sample(frac=0.5)
Randomly select fraction of rows.

df.sample(n=10)
Randomly select n rows.

df.iloc[10:20]
Select rows by position.

df.nlargest(n, 'value')
Select and order top n entries.

df.nsmallest(n, 'value')
Select and order bottom n entries.

Subset Variables (Columns)

df[['width','length','species']]
Select multiple columns with specific names.

df['width'] *or* **df.width**
Select single column with specific name.

df.filter(regex='*regex*')
Select columns whose name matches regular expression *regex*.

regex (Regular Expressions) Examples	
'\.'	Matches strings containing a period '.'
'Length$'	Matches strings ending with word 'Length'
'^Sepal'	Matches strings beginning with the word 'Sepal'
'^x[1-5]$'	Matches strings beginning with 'x' and ending with 1,2,3,4,5
''^(?!Species$).*'	Matches strings except the string 'Species'

df.loc[:,'x2':'x4']
Select all columns between x2 and x4 (inclusive).

df.iloc[:,[1,2,5]]
Select columns in positions 1, 2 and 5 (first column is 0).

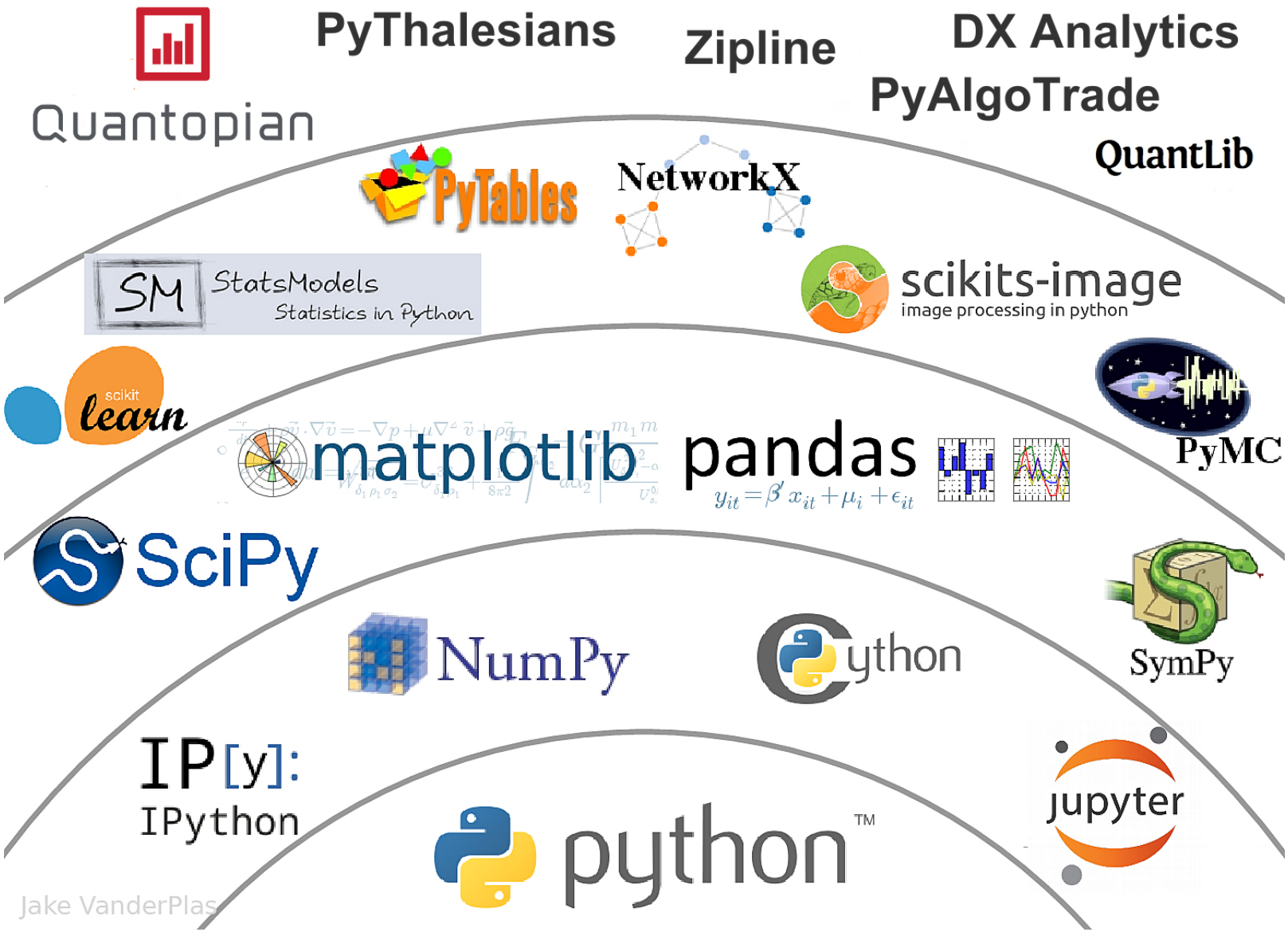
df.loc[df['a'] > 10, ['a','c']]
Select rows meeting logical condition, and only the specific columns .

Logic in Python (and pandas)			
<	Less than	!=	Not equal to
>	Greater than	df.column.isin(*values*)	Group membership
==	Equals	pd.isnull(*obj*)	Is NaN
<=	Less than or equals	pd.notnull(*obj*)	Is not NaN
>=	Greater than or equals	&,|,~,^,df.any(),df.all()	Logical and, or, not, xor, any, all

http://pandas.pydata.org/ This cheat sheet inspired by Rstudio Data Wrangling Cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf) Written by Irv Lustig, Princeton Consultants

Source: https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf

The Quant Finance PyData Stack

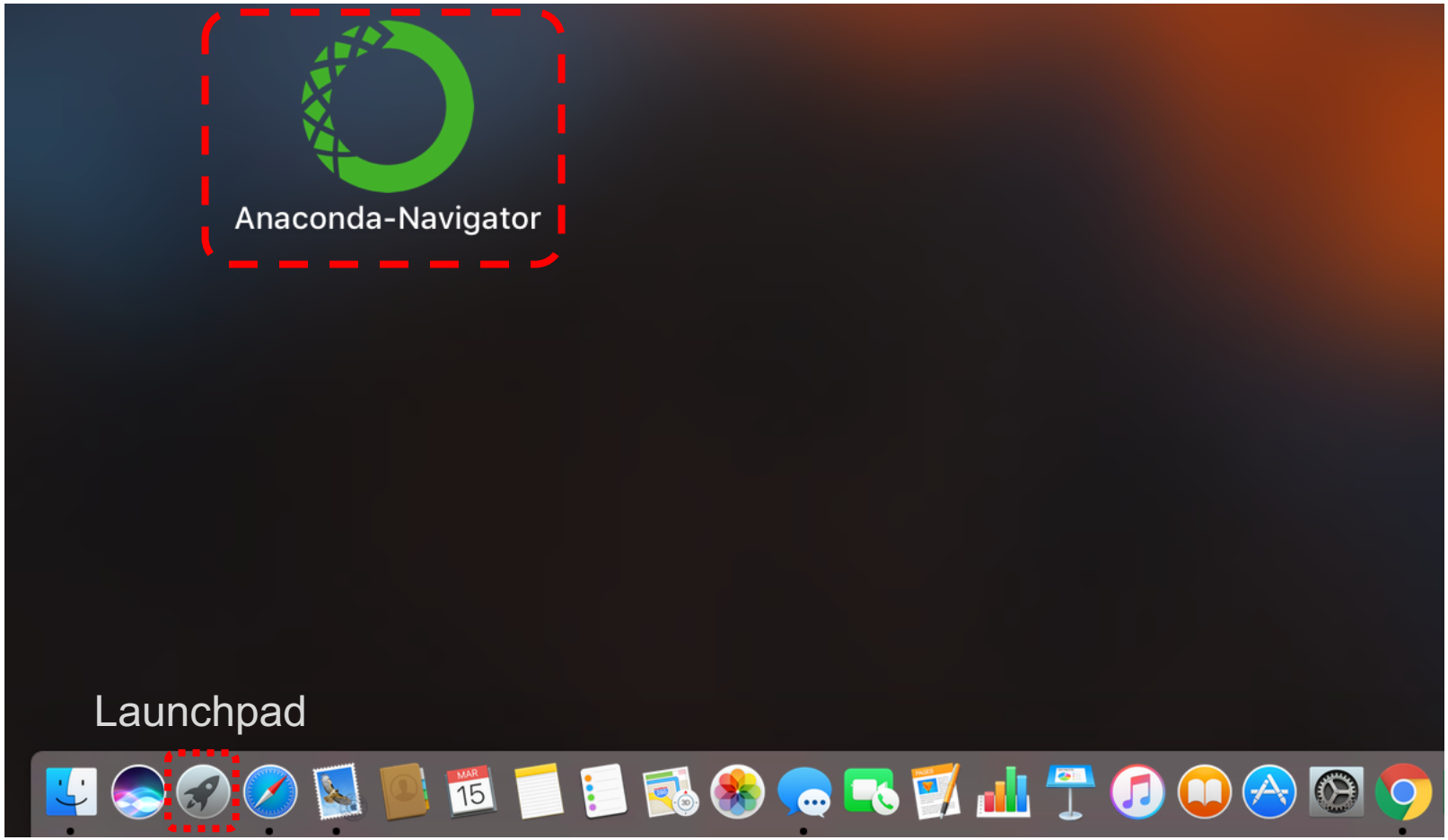


Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#/5

Source: https://www.python.org/community/logos/

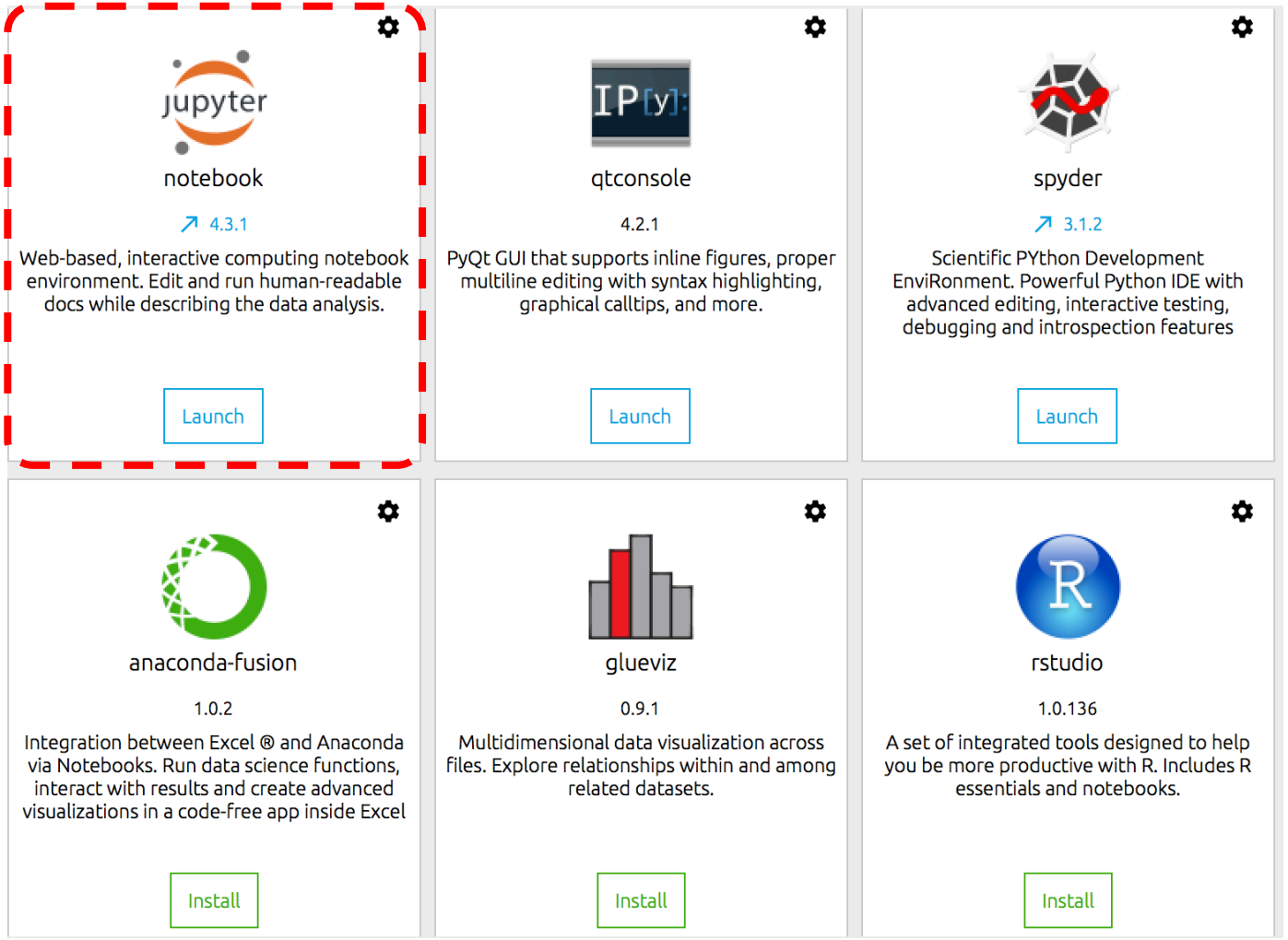
Python

Anaconda-Navigator

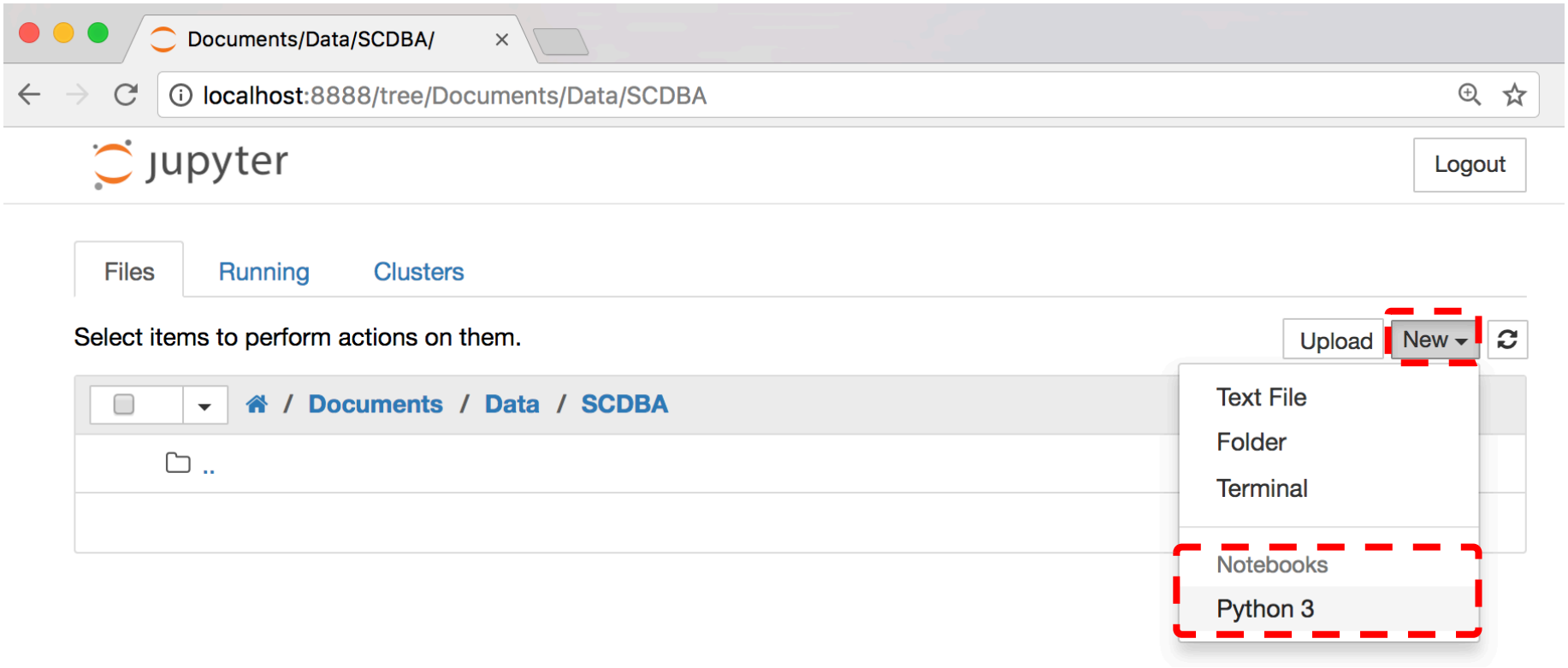


Anaconda-Navigator

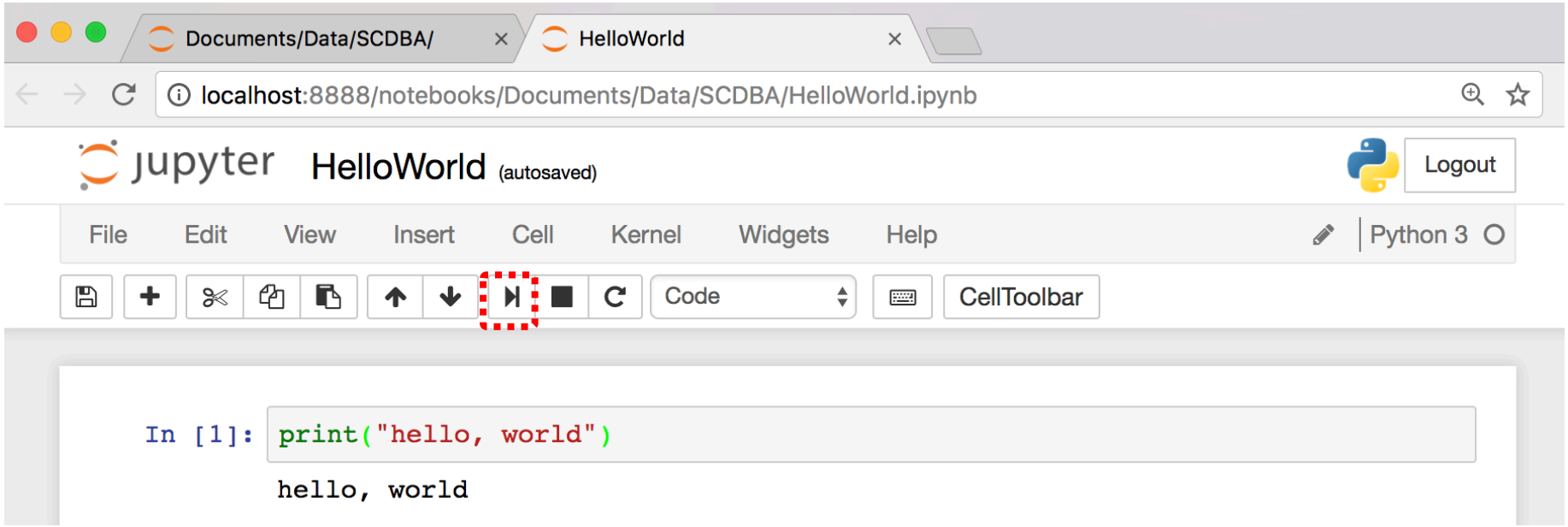
Jupyter Notebook



Jupyter Notebook
New Python 3



print("hello, world")



Text input and output

```
print("Hello World")
```

```
print("Hello World\nThis is a message")
```

```
x = 3
print(x)
```

```
x = 2
y = 3
print(x, ' ', y)
```

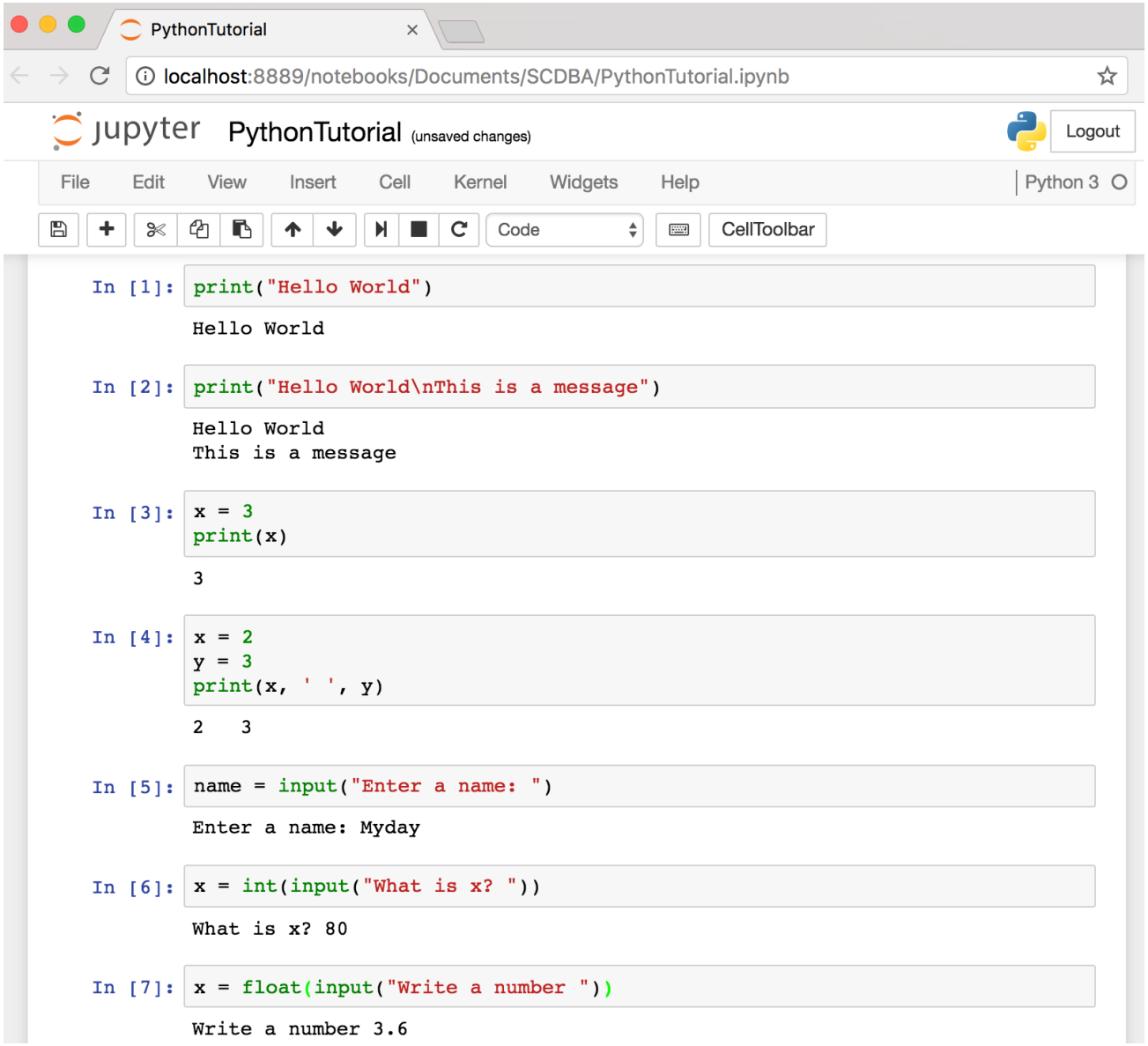
```
name = input("Enter a name: ")
```

```
x = int(input("What is x? "))
```

```
x = float(input("Write a number "))
```

Source: http://pythonprogramminglanguage.com/text-input-and-output/

Text input and output



Source: http://pythonprogramminglanguage.com/text-input-and-output/

Variables

```
x = 2
price = 2.5
word = 'Hello'
```

```
word = 'Hello'
word = "Hello"
word = '''Hello'''
```

```
x = 2
x = x + 1
x = 5
```

Source: http://pythonprogramminglanguage.com/

Python Basic Operators

```
print('7 + 2 =', 7 + 2)
print('7 - 2 =', 7 - 2)
print('7 * 2 =', 7 * 2)
print('7 / 2 =', 7 / 2)
print('7 // 2 =', 7 // 2)
print('7 % 2 =', 7 % 2)
print('7 ** 2 =', 7 ** 2)
```

```
print('7 + 2 =', 7 + 2)
print('7 - 2 =', 7 - 2)
print('7 * 2 =', 7 * 2)
print('7 / 2 =', 7 / 2)
print('7 // 2 =', 7 // 2)
print('7 % 2 =', 7 % 2)
print('7 ** 2 =', 7 ** 2)
```

```
7 + 2 = 9
7 - 2 = 5
7 * 2 = 14
7 / 2 = 3.5
7 // 2 = 3
7 % 2 = 1
7 ** 2 = 49
```

BMI Calculator in Python

```
height_cm = float(input("Enter your height in cm: "))
weight_kg = float(input("Enter your weight in kg: "))

height_m = height_cm/100
BMI = (weight_kg/(height_m**2))

print("Your BMI is: " + str(round(BMI,1)))
```

Source: http://code.activestate.com/recipes/580615-bmi-code/

BMI Calculator in Python

jupyter PythonTutorial Last Checkpoint: a minute ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Python 3

Code CellToolbar

In [1]:
```
height_cm = float(input("Enter your height in cm: "))
weight_kg = float(input("Enter your weight in kg: "))

height_m = height_cm/100
BMI = (weight_kg/(height_m**2))

print("Your BMI is: " + str(round(BMI,1)))
```

```
Enter your height in cm: 170
Enter your weight in kg: 60
Your BMI is: 20.8
```

In [ ]:

Source: http://code.activestate.com/recipes/580615-bmi-code/

Future value of a specified principal amount, rate of interest, and a number of years

Source: https://www.w3resource.com/python-exercises/python-basic-exercise-39.php

Future Value (FV)

```
# How much is your $100 worth after 7 years?
print(100 * 1.1 ** 7)
# output = 194.87
```

```
print(100 * 1.1 ** 7)
```

```
194.87171000000012
```

Source: https://www.w3resource.com/python-exercises/python-basic-exercise-39.php

Future Value (FV)

```
pv = 100
r = 0.1
n = 7

fv = pv * ((1 + (r)) ** n)
print(round(fv, 2))
```

```
pv = 100
r = 0.1
n = 7

fv = pv * ((1 + (r)) ** n)
print(round(fv, 2))
```

```
194.87
```

Future Value (FV)

```
amount = 100
interest = 10 #10% = 0.01 * 10
years = 7

future_value = amount * ((1 + (0.01 * interest)) ** years)
print(round(future_value, 2))
```

```
amount = 100
interest = 10 #10% = 0.01 * 10
years = 7

future_value  = amount * ((1 + (0.01 * interest)) ** years)
print(round(future_value, 2))
```

```
194.87
```

Source: https://www.w3resource.com/python-exercises/python-basic-exercise-39.php

if statements

> greater than
< smaller than
== equals
!= is not

```
score = 80
if score >=60 :
    print("Pass")
else:
    print("Fail")
```

```
Pass
```


```
score = 80
if score >=60 :
    print("Pass")
else:
    print("Fail")
```


Source: http://pythonprogramminglanguage.com/

if elif else

```
score = 90
grade = ""
if score >=90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "E"
print(grade)
# grade = ”A”
```

```
A
```

http://pythontutor.com/visualize.html
https://goo.gl/E6w5ph

Source: http://pythonprogramminglanguage.com/

for loops

```
for i in range(1,11):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

Source: http://pythonprogramminglanguage.com/

for loops

```
for i in range(1,10):
    for j in range(1,10):
        print(i, ' * ' , j , ' = ', i*j)
```

```
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
```

Source: http://pythonprogramminglanguage.com/

while loops

```
age = 10

while age < 20:
    print(age)
    age = age + 1
```

```
10
11
12
13
14
15
16
17
18
19
```

Source: https://learnpython.trinket.io/learn-python-part-8-loops#/while-loops/about-while-loops

Functions

```
def convertCMtoM(xcm):
    m = xcm/100
    return m

cm = 180
m = convertCMtoM(cm)
print(str(m))
```

```
1.8
```

python™

Lists

```
x = [60, 70, 80, 90]
print(len(x))
print(x[0])
print(x[1])
print(x[-1])
```

```
4
60
70
90
```

Tuples

A tuple in Python is a collection that cannot be modified.

A tuple is defined using parenthesis.

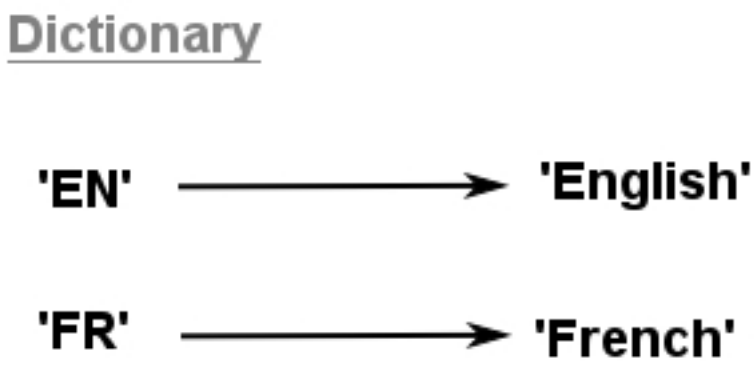
```
x = (10, 20, 30, 40, 50)
print(x[0])
print(x[1])
print(x[2])
print(x[-1])
```

```
10
20
30
50
```

Source: http://pythonprogramminglanguage.com/tuples/

Dictionary

```
k = { 'EN':'English', 'FR':'French' }
print(k['EN'])
```



English

Sets

```
animals = {'cat', 'dog'}
```

```
animals = {'cat', 'dog'}
print('cat' in animals)  # Check if an element is in a set; prints "True"
print('fish' in animals) # prints "False"
animals.add('fish')      # Add an element to a set
print('fish' in animals) # Prints "True"
print(len(animals))      # Number of elements in a set; prints "3"
animals.add('cat')       # Adding an element that is already in the set does nothing
print(len(animals))      # Prints "3"
animals.remove('cat')    # Remove an element from a set
print(len(animals))      # Prints "2"
```

```
True
False
True
3
3
2
```

```
animals = {'cat', 'dog'}
print('cat' in animals)
print('fish' in animals)
animals.add('fish')
print('fish' in animals)
print(len(animals))
animals.add('cat')
print(len(animals))
animals.remove('cat')
print(len(animals))
```

Source: http://cs231n.github.io/python-numpy-tutorial/

File Input / Output

```
with open('myfile.txt', 'w') as file:
    file.write('Hello World\nThis is Python File Input Output')

with open('myfile.txt', 'r') as file:
    text = file.read()
print(text)
```

```
with open('myfile.txt', 'w') as file:
    file.write('Hello World\nThis is Python File Input Output')
```

```
with open('myfile.txt', 'r') as file:
    text = file.read()
print(text)
```

```
Hello World
This is Python File Input Output
```

```
text
```

```
'Hello World\nThis is Python File Input Output'
```

Source: https://github.com/TiesdeKok/LearnPythonforResearch/blob/master/0_python_basics.ipynb

File Input / Output

```
with open('myfile.txt', 'a+') as file:
    file.write('\n' + 'New line')

with open('myfile.txt', 'r') as file:
    text = file.read()
print(text)
```

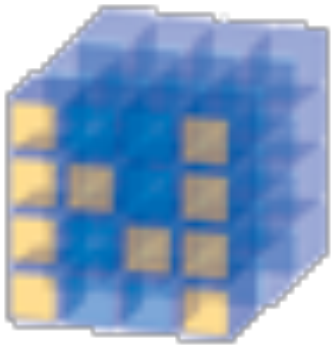
```
with open('myfile.txt', 'a+') as file:
    file.write('\n' + 'New line')
```

```
with open('myfile.txt', 'r') as file:
    text = file.read()
print(text)
```

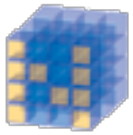
```
Hello World
This is Python File Input Output
New line
```

Source: https://github.com/TiesdeKok/LearnPythonforResearch/blob/master/0_python_basics.ipynb

Numpy



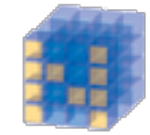
NumPy
Base
N-dimensional array
package



NumPy

NumPy

- NumPy provides a **multidimensional array** object to store homogenous or heterogeneous data;
  it also provides optimized functions/methods to operate on this array object.



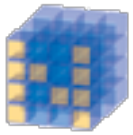
NumPy ndarray

One-dimensional Array (1-D Array)

0	1			n-1
1	**2**	**3**	**4**	**5**

Two-dimensional Array (2-D Array)

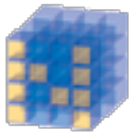
	0	1			n-1
0	**1**	**2**	**3**	**4**	**5**
1	**6**	**7**	**8**	**9**	**10**
	11	**12**	**13**	**14**	**15**
m-1	**16**	**17**	**18**	**19**	**20**



NumPy

NumPy

```
v = range(1, 6)
print(v)
2 * v
import numpy as np
v = np.arange(1, 6)
v
2 * v
```



NumPy

Base
N-dimensional
array package

```
v = range (1, 6)
print(v)
```

```
[1, 2, 3, 4, 5]
```

```
2 * v
```

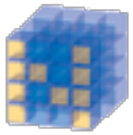
```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

```
import numpy as np
v = np.arange(1, 6)
v
```

```
array([1, 2, 3, 4, 5])
```

```
2 * v
```

```
array([ 2,  4,  6,  8, 10])
```



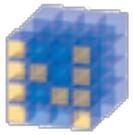
NumPy Create Array

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
c = a * b
c
```

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
c = a * b
c
```

```
array([ 4, 10, 18])
```

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



NumPy

NumPy

```python
import numpy as np

a = np.zeros((2,2))  # Create an array of all zeros
print(a)             # Prints "[[ 0.  0.]
                     #          [ 0.  0.]]"

b = np.ones((1,2))   # Create an array of all ones
print(b)             # Prints "[[ 1.  1.]]"

c = np.full((2,2), 7) # Create a constant array
print(c)               # Prints "[[ 7.  7.]
                       #          [ 7.  7.]]"

d = np.eye(2)        # Create a 2x2 identity matrix
print(d)             # Prints "[[ 1.  0.]
                     #          [ 0.  1.]]"

e = np.random.random((2,2)) # Create an array filled with random values
print(e)                    # Might print "[[ 0.91940167  0.08143941]
                            #               [ 0.68744134  0.87236687]]"
```

```
[[ 0.  0.]
 [ 0.  0.]]
[[ 1.  1.]]
[[ 7.  7.]
 [ 7.  7.]]
[[ 1.  0.]
 [ 0.  1.]]
[[ 0.22886991  0.68473232]
 [ 0.20683825  0.16589995]]
```

Source: http://cs231n.github.io/python-numpy-tutorial/

Numpy Quickstart Tutorial

SciPy.org Sponsored By ENTHOUGHT

Scipy.org | Docs | NumPy v1.13.dev0 Manual | NumPy User Guide

index | next | previous

Quickstart tutorial

Prerequisites

Before reading this tutorial you should know a bit of Python. If you would like to refresh your memory, take a look at the Python tutorial.

If you wish to work the examples in this tutorial, you must also have some software installed on your computer. Please see http://scipy.org/install.html for instructions.

The Basics

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

For example, the coordinates of a point in 3D space `[1, 2, 1]` is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

```
[[ 1., 0., 0.],
 [ 0., 1., 2.]]
```

NumPy's array class is called `ndarray`. It is also known by the alias `array`. Note that `numpy.array` is not the same as the Standard Python Library class `array.array`, which only handles one-dimensional arrays and offers less functionality. The more important attributes of an `ndarray` object are:

ndarray.ndim

the number of axes (dimensions) of the array. In the Python world, the number of dimensions is referred to as *rank*.

ndarray.shape

Table Of Contents

https://docs.scipy.org/doc/numpy-dev/user/quickstart.html

```
import numpy as np
a = np.arange(15).reshape(3, 5)
```

a.shape
a.ndim
a.dtype.name

```
import numpy as np
a = np.arange(15).reshape(3, 5)
a
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

```
print(a.shape)
```

```
(3, 5)
```

```
a.ndim
```

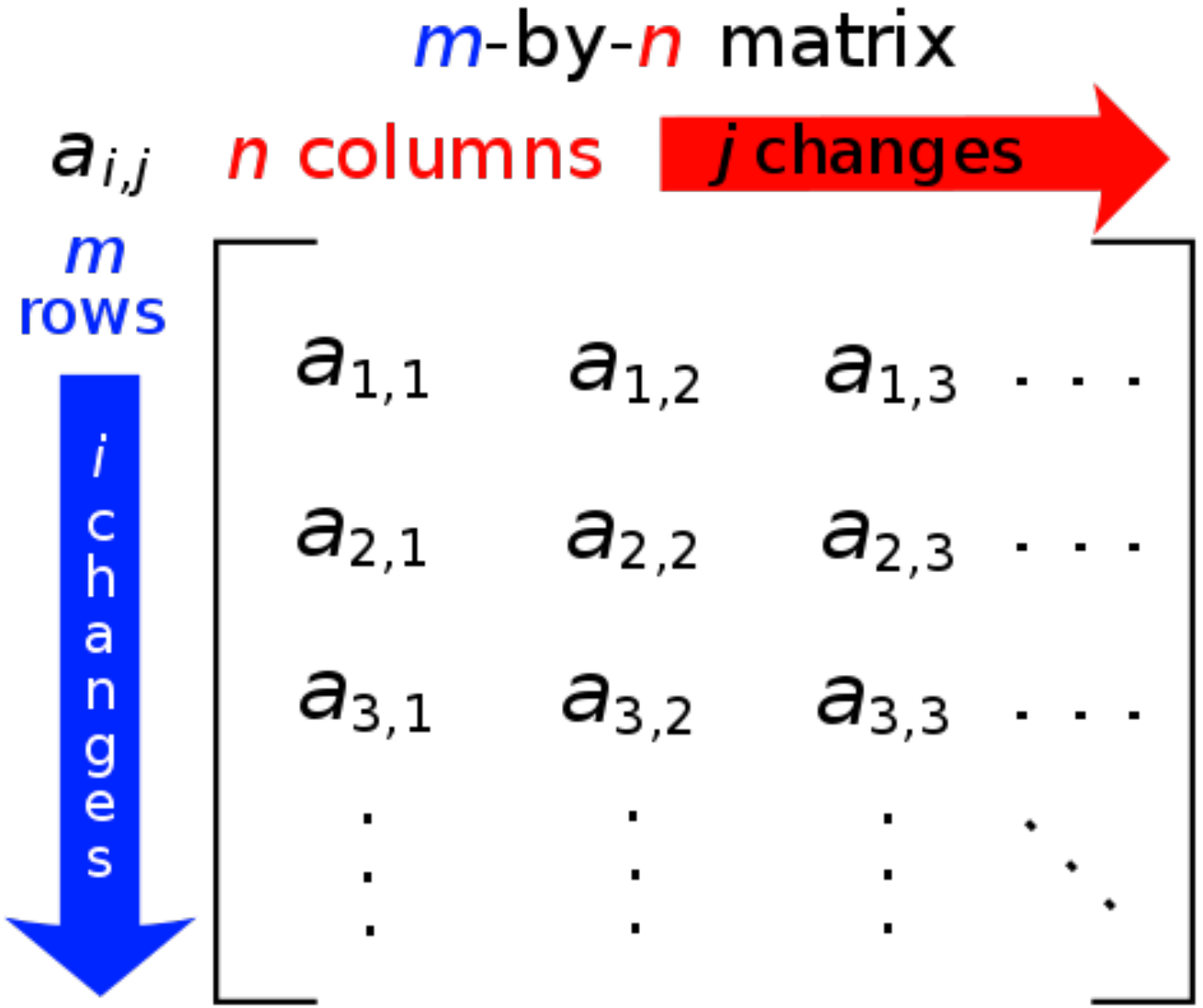
```
2
```

```
a.dtype.name
```

```
'int64'
```

Source: https://docs.scipy.org/doc/numpy-dev/user/quickstart.html

Matrix



Source: https://simple.wikipedia.org/wiki/Matrix_(mathematics)

NumPy ndarray: Multidimensional Array Object

NumPy ndarray

One-dimensional Array (1-D Array)

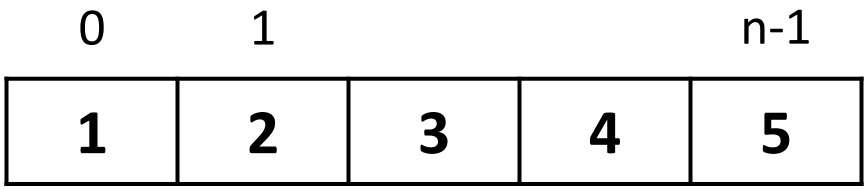
0	1			n-1
1	**2**	**3**	**4**	**5**

Two-dimensional Array (2-D Array)

	0	1			n-1
0	**1**	**2**	**3**	**4**	**5**
1	**6**	**7**	**8**	**9**	**10**
	11	**12**	**13**	**14**	**15**
m-1	**16**	**17**	**18**	**19**	**20**

```
import numpy as np
a = np.array([1,2,3,4,5])
```

One-dimensional Array (1-D Array)




```
a = np.array([1,2,3,4,5])
a
```

```
array([1, 2, 3, 4, 5])
```

```
a = np.array([[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15],[16,17,18,19,20]])
```

Two-dimensional Array (2-D Array)

	0	1			n-1
0	**1**	**2**	**3**	**4**	**5**
1	**6**	**7**	**8**	**9**	**10**
	11	**12**	**13**	**14**	**15**
m-1	**16**	**17**	**18**	**19**	**20**

```
a = np.array([[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15],[16,17,18,19,20]])
a
```

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20]])
```

```
import numpy as np
a = np.array([[0, 1, 2, 3],
[10, 11, 12, 13],
[20, 21, 22, 23]])
a
```

0	1	2	3
10	11	12	13
20	21	22	23

a = np.array([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])

```
a = np.array([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])
a
```

```
array([[ 0,  1,  2,  3],
       [10, 11, 12, 13],
       [20, 21, 22, 23]])
```

```
print(a.ndim)
```

```
2
```

```
print(a.shape)
```

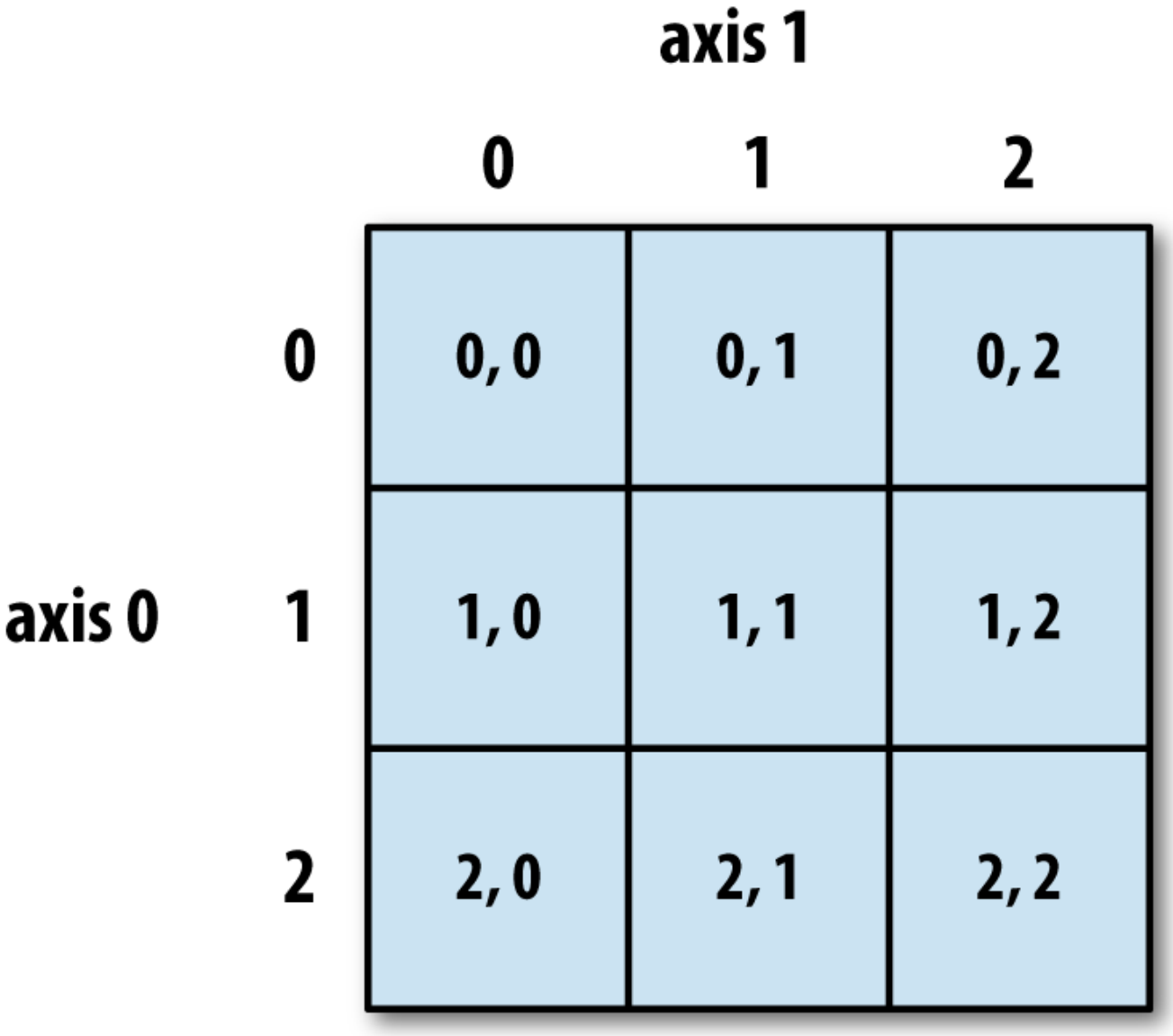
```
(3, 4)
```

0	1	2	3
10	11	12	13
20	21	22	23

NumPy Basics: Arrays and Vectorized Computation

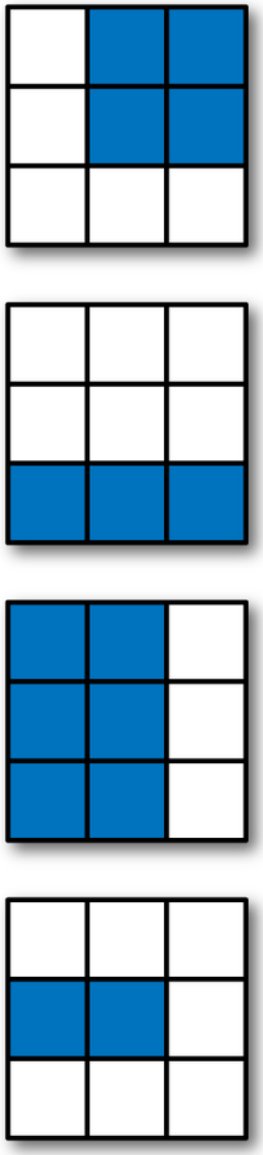
Source: https://www.safaribooksonline.com/library/view/python-for-data/9781449323592/ch04.html

NumPy Array



Source: https://www.safaribooksonline.com/library/view/python-for-data/9781449323592/ch04.html

Numpy Array



Expression	Shape
`arr[:2, 1:]`	`(2, 2)`
`arr[2]`	`(3,)`
`arr[2, :]`	`(3,)`
`arr[2:, :]`	`(1, 3)`
`arr[:, :2]`	`(3, 2)`
`arr[1, :2]`	`(2,)`
`arr[1:2, :2]`	`(1, 2)`

Source: https://www.safaribooksonline.com/library/view/python-for-data/9781449323592/ch04.html

Wes McKinney (2017), "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython", 2nd Edition, O'Reilly Media.

Materials and IPython notebooks for "Python for Data Analysis" by Wes McKinney, published by O'Reilly Media

52 commits | **2** branches | **0** releases | **6** contributors

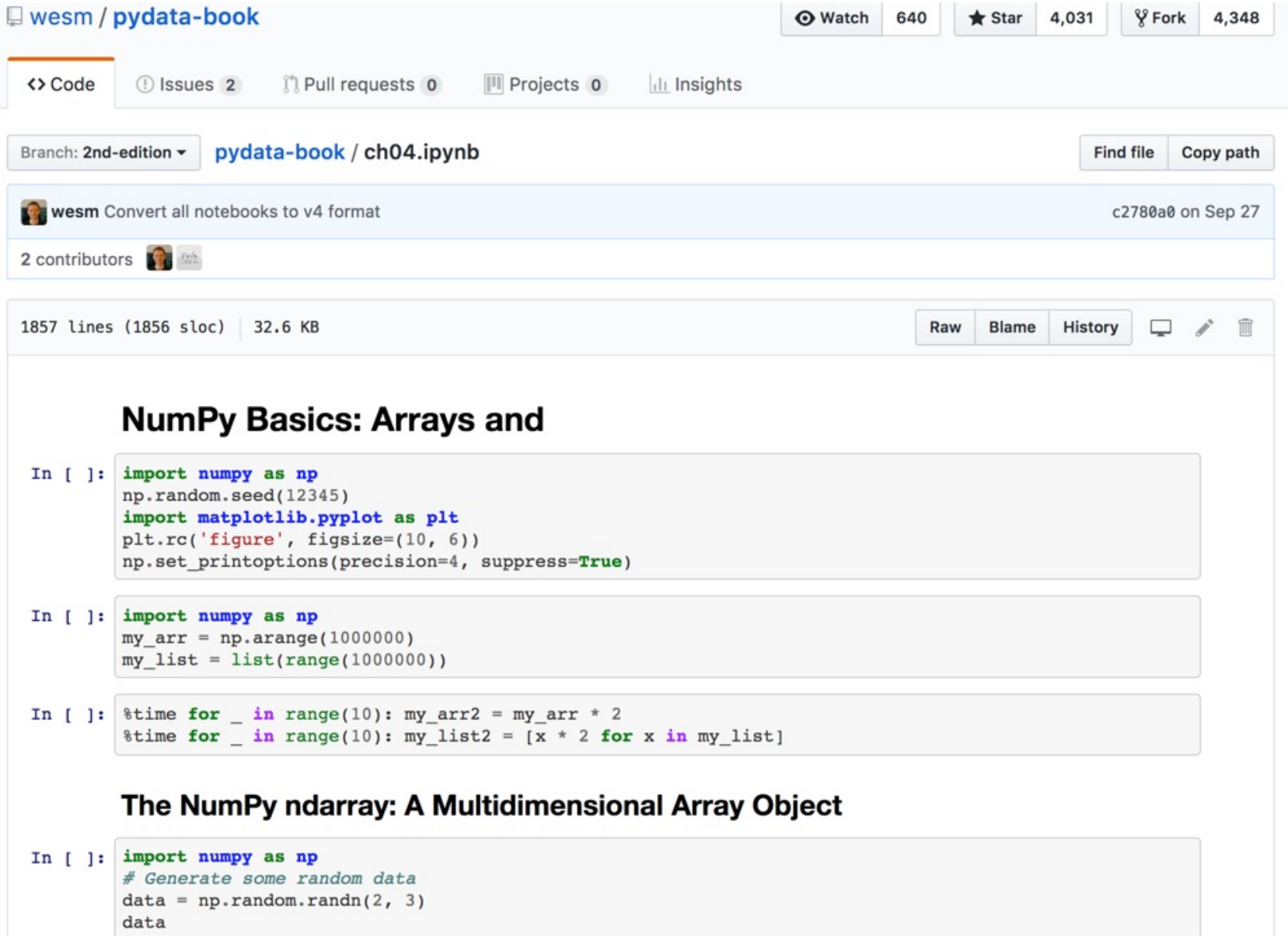
Branch: **2nd-edition** | New pull request | Find file | Clone or download

betatim committed with **wesm** Add requirements (#71) — Latest commit ea47998 5 days ago

datasets	Add Kaggle titanic dataset	5 months ago
examples	Remove sex column from tips dataset	4 months ago
.gitignore	Add gitignore	2 years ago
COPYING	Use MIT license for code examples	a month ago
README.md	Add launch in Azure Notebooks button (#70)	19 days ago
appa.ipynb	Make more cells markdown instead of raw	a month ago
ch02.ipynb	Make more cells markdown instead of raw	a month ago
ch03.ipynb	Make more cells markdown instead of raw	a month ago
ch04.ipynb	Convert all notebooks to v4 format	a month ago
ch05.ipynb	Make more cells markdown instead of raw	a month ago
ch06.ipynb	Make more cells markdown instead of raw	a month ago
ch07.ipynb	Convert all notebooks to v4 format	a month ago
ch08.ipynb	Make more cells markdown instead of raw	a month ago
ch09.ipynb	Make more cells markdown instead of raw	a month ago
ch10.ipynb	Make more cells markdown instead of raw	a month ago

https://github.com/wesm/pydata-book

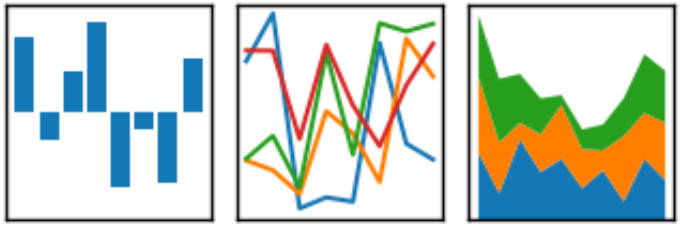
Wes McKinney (2017), "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython", 2nd Edition, O'Reilly Media.



Source: https://github.com/wesm/pydata-book/blob/2nd-edition/ch04.ipynb

pandas

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

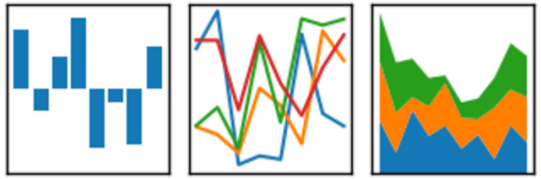


Python Pandas for Finance

pandas

pandas

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



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.

pandas is a NUMFocus sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project.

A Fiscally Sponsored Project of
NUMFOCUS
OPEN CODE = BETTER SCIENCE

0.19.2 Final (December 24, 2016)

This is a minor bug-fix release in the 0.19.x series and includes some small regression fixes, bug fixes and performance improvements.

Highlights include:

- Compatibility with Python 3.6

VERSIONS

Release
0.19.2 - December 2016
download // docs // pdf

Development
0.20.0 - 2017
github // docs

Previous Releases
0.19.1 - download // docs // pdf
0.19.0 - download // docs // pdf
0.18.1 - download // docs // pdf
0.18.0 - download // docs // pdf
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

http://pandas.pydata.org/

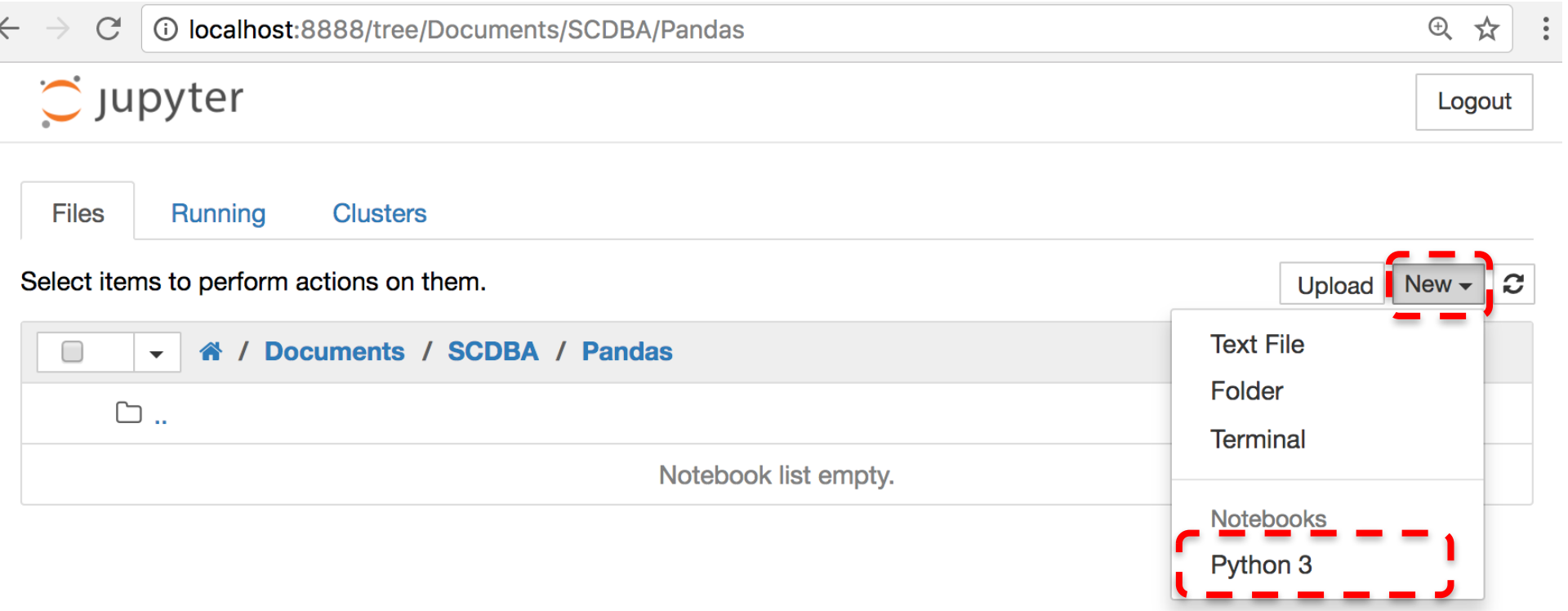
pandas

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/

Jupyter Notebook New Python 3



Creating pd.DataFrame

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

In [1]:
```
import numpy as np
import pandas as pd
df = pd.DataFrame({"a": [4, 5, 6],
                   "b": [7, 8, 9],
                   "c": [10, 11, 12]},
                  index = [1, 2, 3])
df
```

Out[1]:

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame({"a": [4, 5, 6],
                   "b": [7, 8, 9],
                   "c": [10, 11, 12]},
                  index = [1, 2, 3])
```

Source: https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf

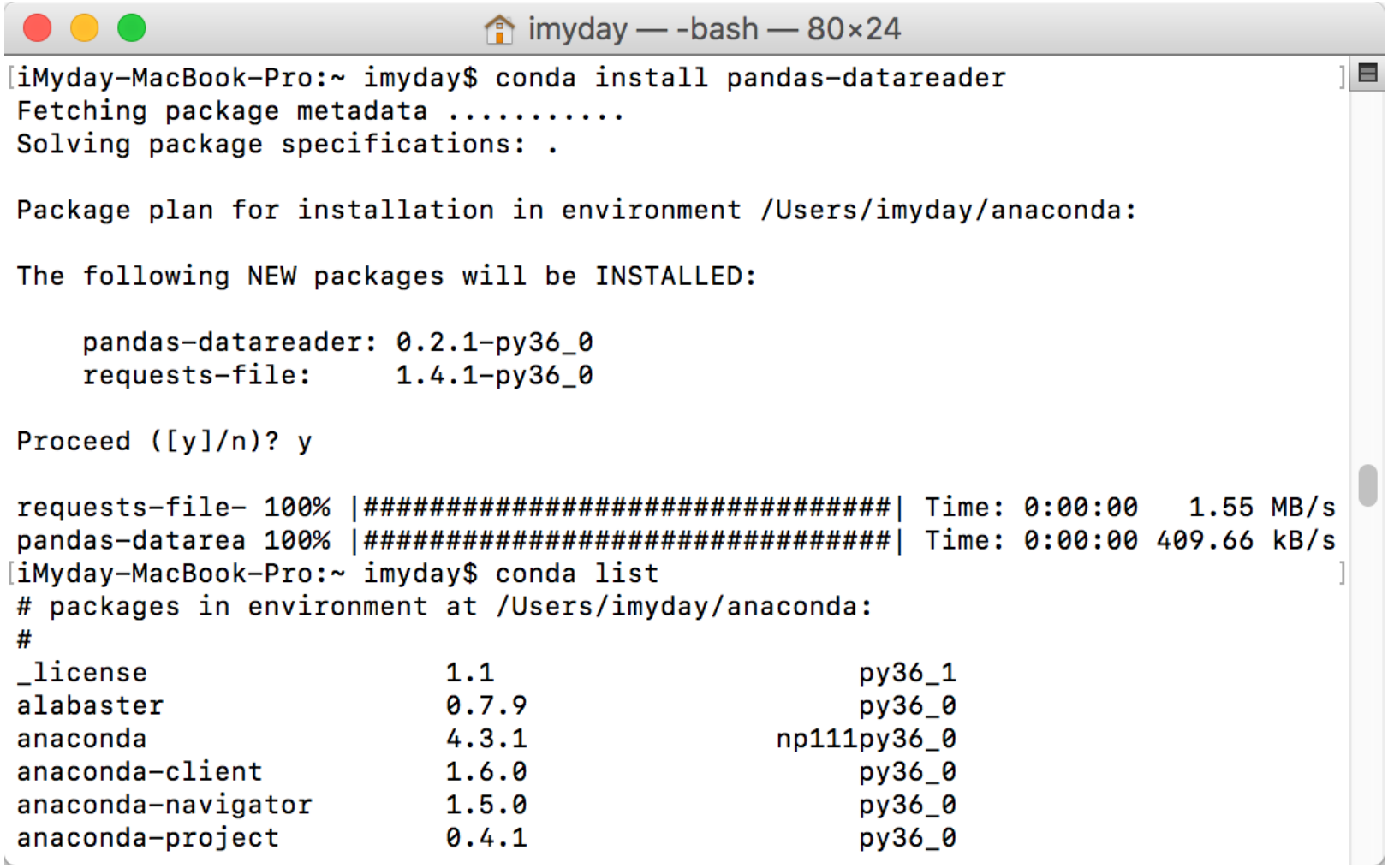
Pandas DataFrame

```
type(df)
```

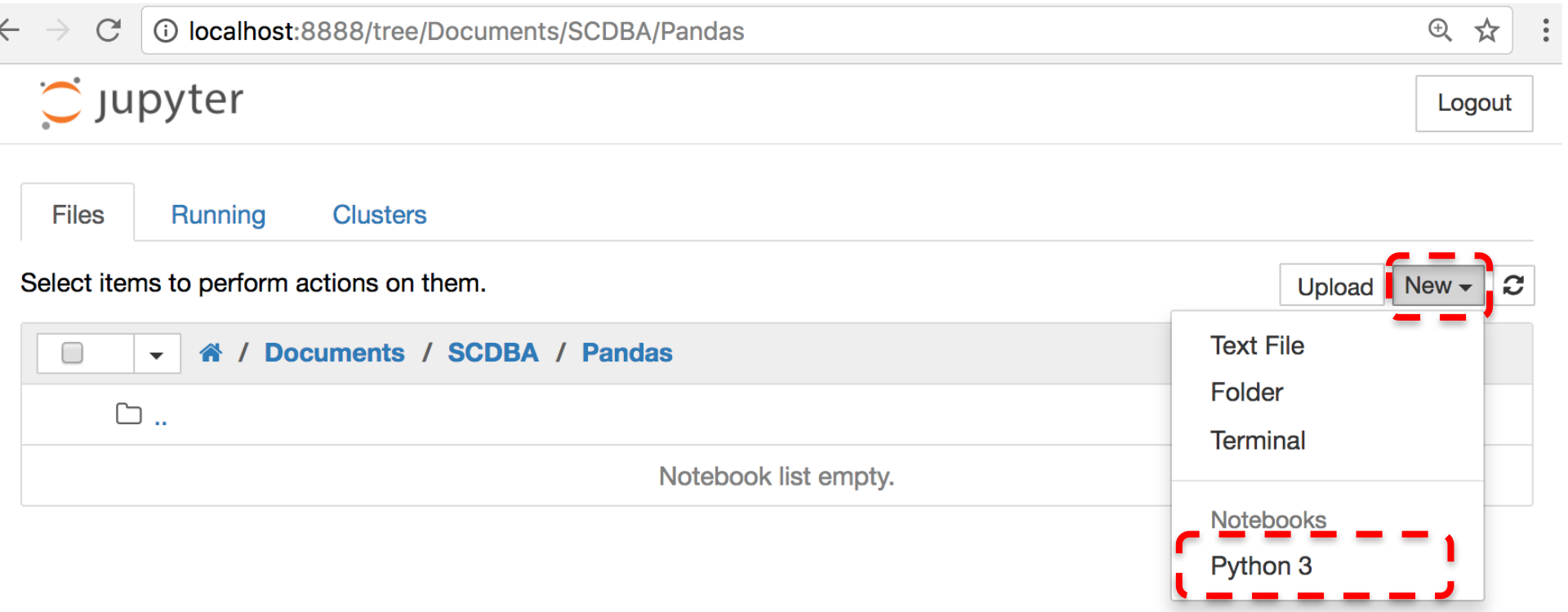
```
type(df)
```

```
pandas.core.frame.DataFrame
```

conda install pandas-datareader



Jupyter Notebook New Python 3



```
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('20170301',
periods=6)
dates
```

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

localhost:8888/notebooks/Documents/SCDBA/python/PythonPandasFinance.ipynb

jupyter PythonPandasFinance (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Python 3

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('20170301', periods=6)
dates
```

Out[3]:
```
DatetimeIndex(['2017-03-01', '2017-03-02', '2017-03-03', '2017-03-04',
               '2017-03-05', '2017-03-06'],
              dtype='datetime64[ns]', freq='D')
```

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

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

	A	B	C	D
2017-03-01	-1.613934	0.168983	0.686730	1.520077
2017-03-02	0.656124	-1.155197	-1.969431	0.444569
2017-03-03	-0.242393	-0.372972	-0.310283	1.102838
2017-03-04	-0.034627	-0.422640	0.489834	1.498832
2017-03-05	-1.022516	-0.841598	-0.136151	-0.767695
2017-03-06	1.208003	3.114586	-0.451570	0.579479

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

	A	B	C	D	E	F
student1	-0.420406	0.829262	-0.326521	-0.037699	-1.350555	-0.617676
student2	0.310825	-0.356479	0.149704	-0.685609	-0.744307	-0.488782
student3	-1.295312	0.765656	1.701502	-0.415809	-0.454114	0.397702
student4	0.979525	0.367767	1.869465	0.988012	0.916746	0.165911

```
df2 = pd.DataFrame({ 'A' : 1.,
'B' : pd.Timestamp('20170322'),
'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
```

```
df2 = pd.DataFrame({ 'A' : 1.,
'B' : pd.Timestamp('20170322'),
'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
```

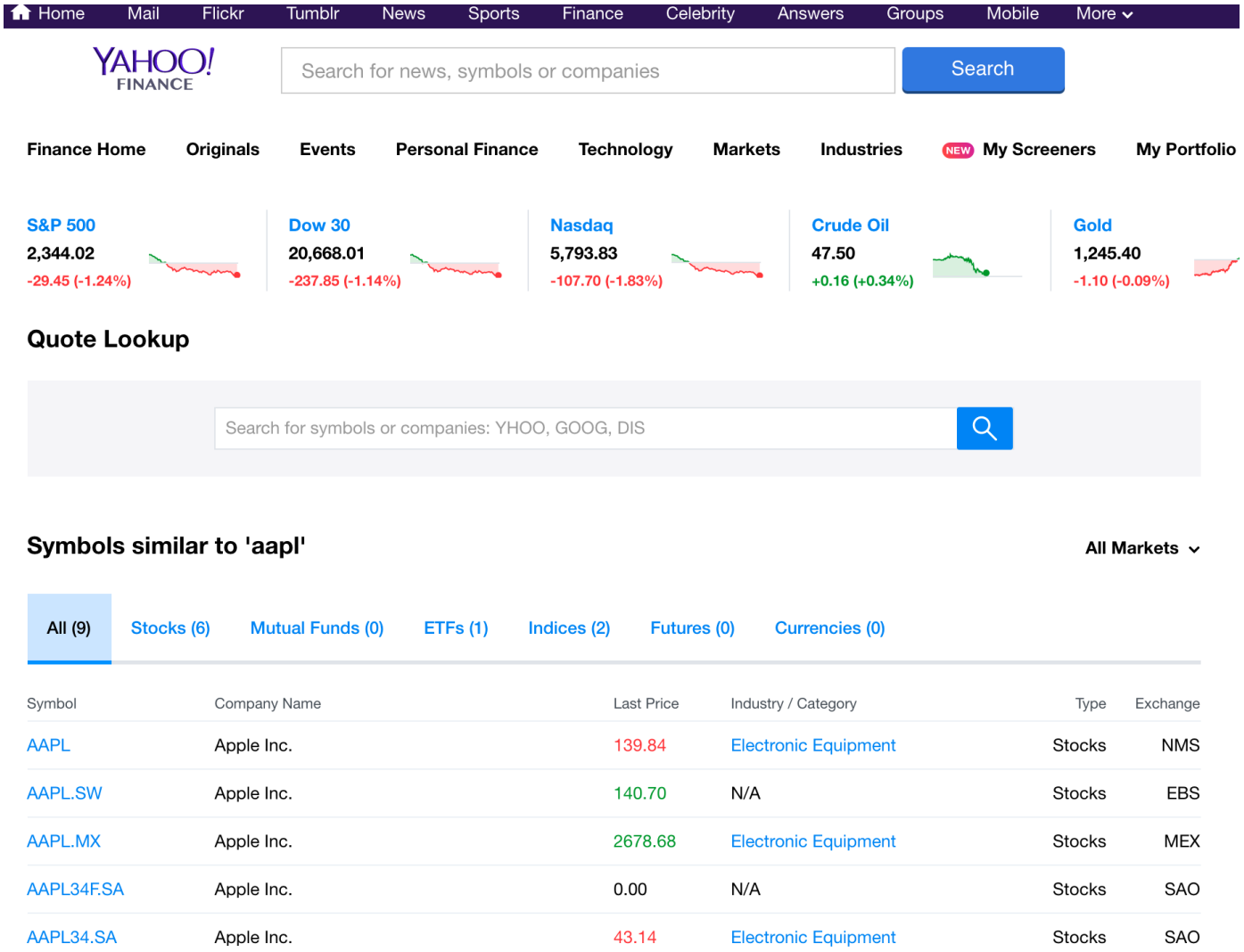
	A	B	C	D	E	F
0	1.0	2017-03-22	2.5	3	test	foo
1	1.0	2017-03-22	2.5	3	train	foo
2	1.0	2017-03-22	2.5	3	test	foo
3	1.0	2017-03-22	2.5	3	train	foo

df2.dtypes

```
df2.dtypes
```

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

Yahoo Finance Symbols: AAPL Apple Inc. (AAPL)



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

Apple Inc. (AAPL) -NasdaqGS

YAHOO! FINANCE

Search for news, symbols or companies | Search

Apple Inc. (AAPL)

NasdaqGS - NasdaqGS Delayed Price. Currency in USD

Add to watchlist

139.84 -1.62 (-1.15%)
At close: 4:00PM EDT

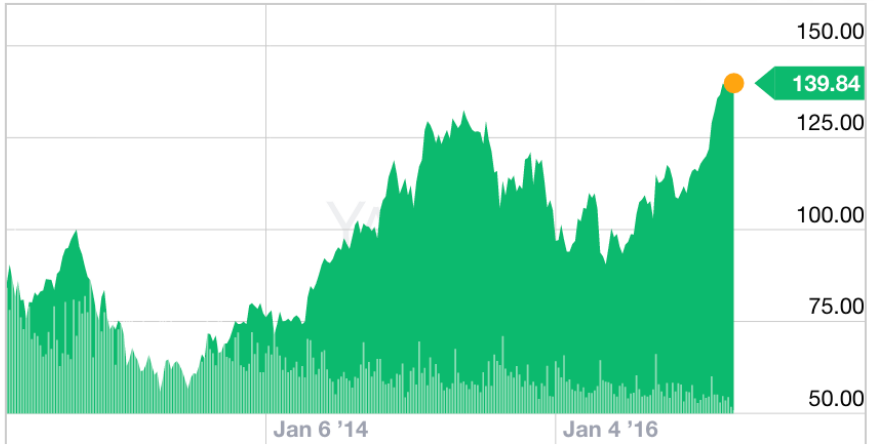
139.35 -0.49 (-0.35%)
After hours: 7:59PM EDT

Summary | Conversations | Statistics | Profile | Financials | Options | Holders | Historical Data | Analysts

Previous Close	141.46	Market Cap	733.68B
Open	142.11	Beta	1.45
Bid	139.31 x 100	PE Ratio (TTM)	16.79
Ask	139.40 x 1300	EPS (TTM)	8.33
Day's Range	139.73 - 142.80	Earnings Date	Apr 24, 2017 - Apr 28, 2017
52 Week Range	89.47 - 142.80	Dividend & Yield	2.28 (1.63%)
Volume	39,529,912	Ex-Dividend Date	N/A
Avg. Volume	26,889,183	1y Target Est	143.29

Trade prices are not sourced from all markets

1D 5D 1M 6M YTD 1Y 2Y **5Y** 10Y MAX | Interactive chart

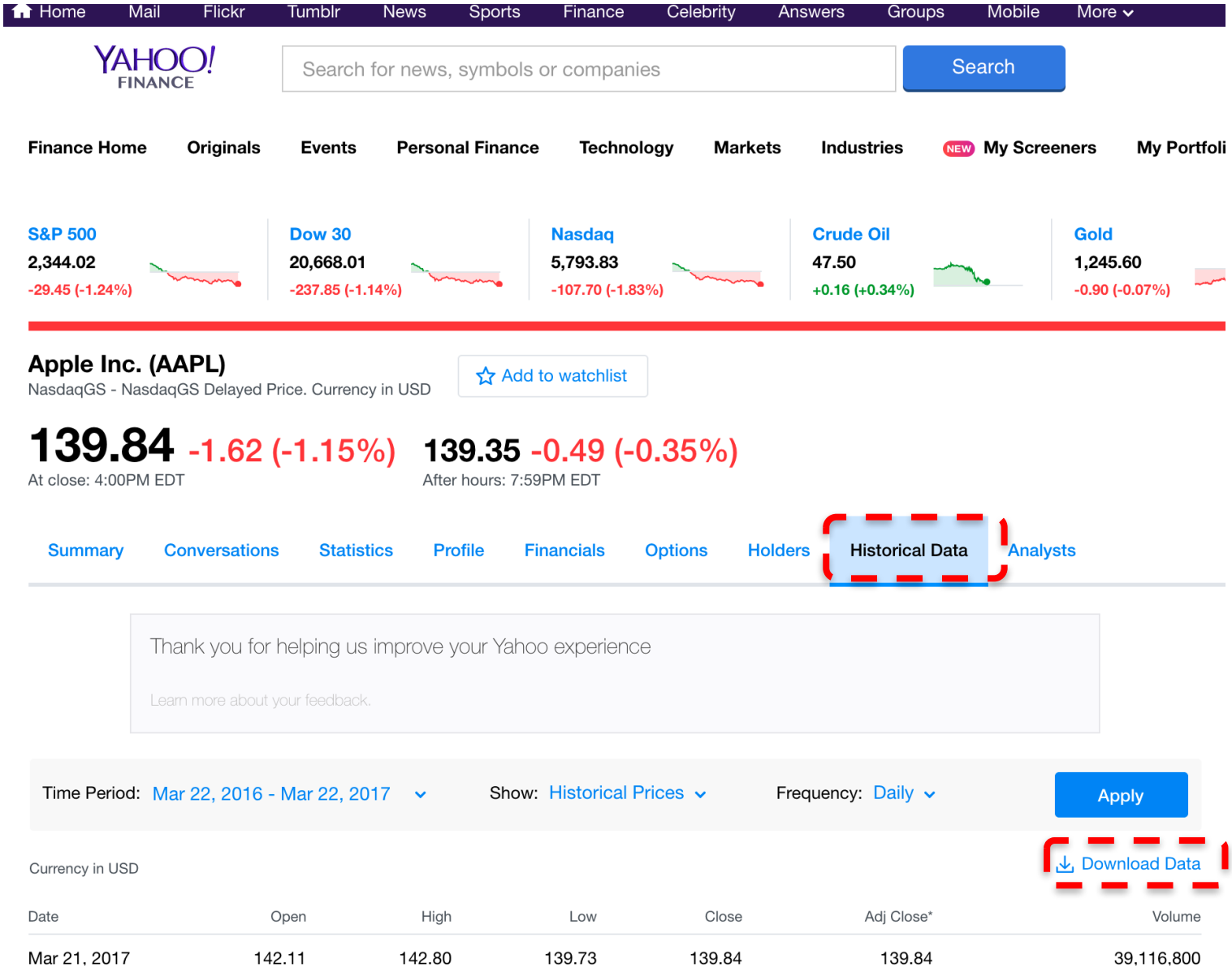


http://finance.yahoo.com/quote/AAPL?p=AAPL

Yahoo Finance Charts: Apple Inc. (AAPL)

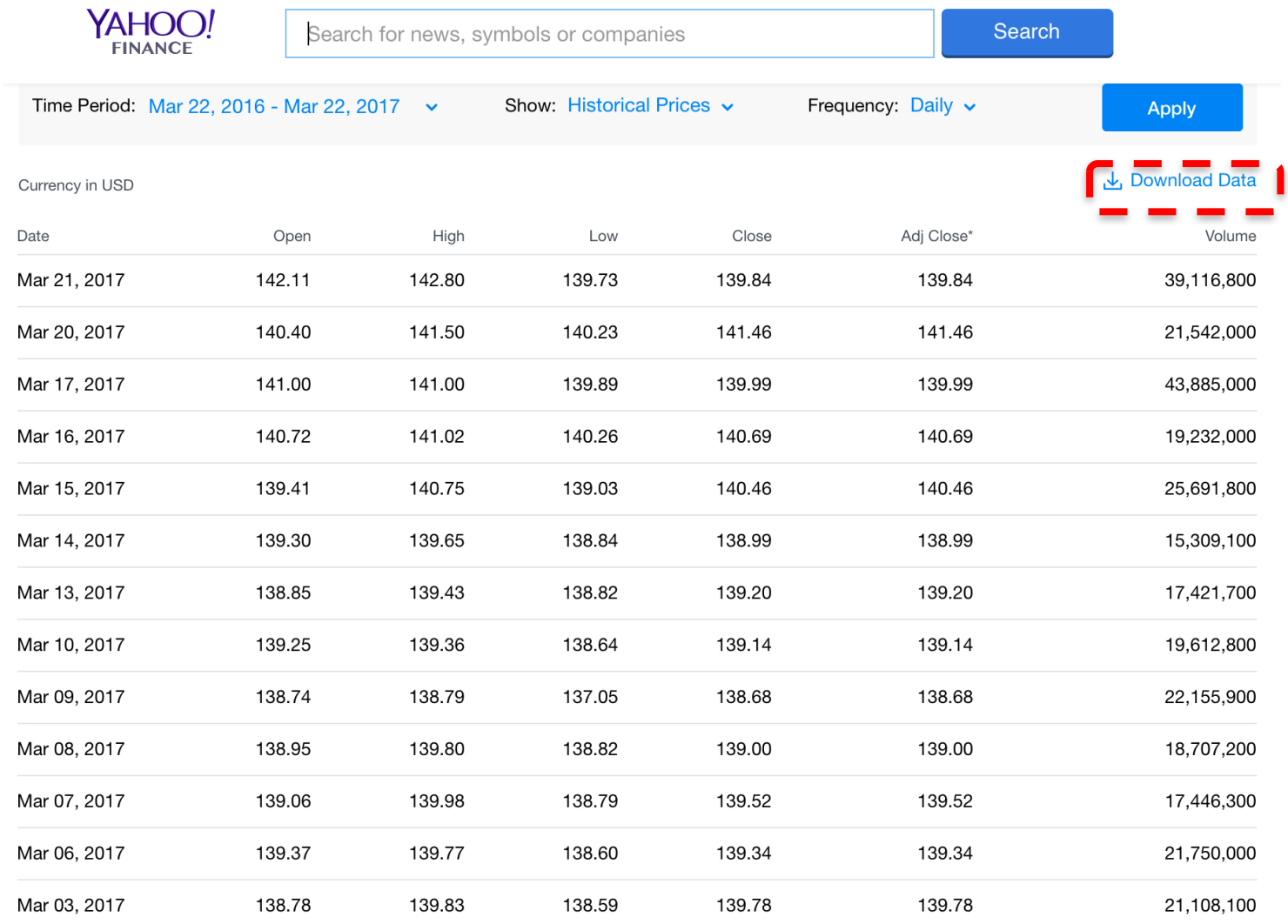
http://finance.yahoo.com/chart/AAPL

Apple Inc. (AAPL) Historical Data



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

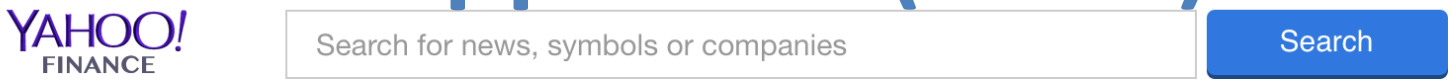
Yahoo Finance Historical Prices Apple Inc. (AAPL)



Date	Open	High	Low	Close	Adj Close*	Volume
Mar 21, 2017	142.11	142.80	139.73	139.84	139.84	39,116,800
Mar 20, 2017	140.40	141.50	140.23	141.46	141.46	21,542,000
Mar 17, 2017	141.00	141.00	139.89	139.99	139.99	43,885,000
Mar 16, 2017	140.72	141.02	140.26	140.69	140.69	19,232,000
Mar 15, 2017	139.41	140.75	139.03	140.46	140.46	25,691,800
Mar 14, 2017	139.30	139.65	138.84	138.99	138.99	15,309,100
Mar 13, 2017	138.85	139.43	138.82	139.20	139.20	17,421,700
Mar 10, 2017	139.25	139.36	138.64	139.14	139.14	19,612,800
Mar 09, 2017	138.74	138.79	137.05	138.68	138.68	22,155,900
Mar 08, 2017	138.95	139.80	138.82	139.00	139.00	18,707,200
Mar 07, 2017	139.06	139.98	138.79	139.52	139.52	17,446,300
Mar 06, 2017	139.37	139.77	138.60	139.34	139.34	21,750,000
Mar 03, 2017	138.78	139.83	138.59	139.78	139.78	21,108,100

http://finance.yahoo.com/quote/AAPL/history

Yahoo Finance Historical Prices
Apple Inc. (AAPL)



Time Period: Dec 12, 1980 - Mar 22, 2017 Show: Historical Prices Frequency: Daily Apply

1D 5D 3M 6M
YTD 1Y 5Y MAX

Start Date 12/12/1980 End Date 3/22/2017

Done Cancel

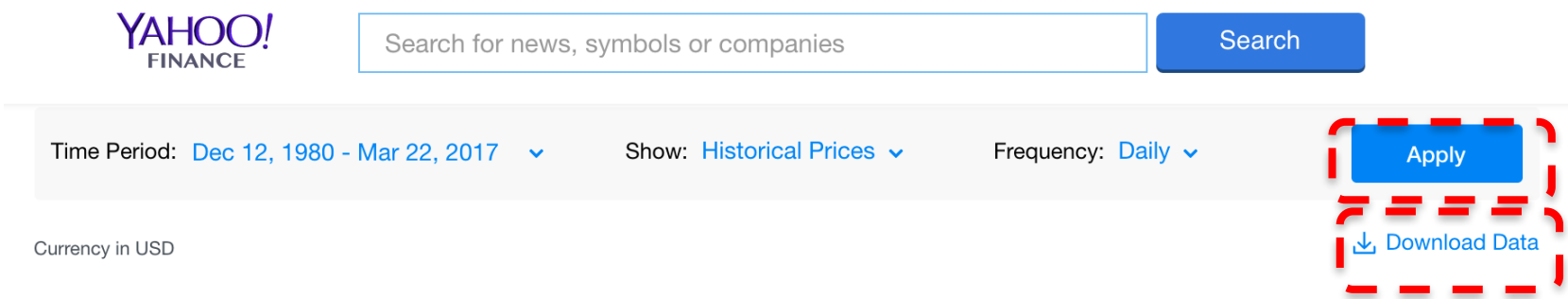
Currency i

Download Data

Date		High	Low	Close	Adj Close*	Volume
Mar 21, 2		42.80	139.73	139.84	139.84	39,116,800
Mar 20, 2		41.50	140.23	141.46	141.46	21,542,000
Mar 17, 2		41.00	139.89	139.99	139.99	43,885,000
Mar 16, 2017	140.72	141.02	140.26	140.69	140.69	19,232,000
Mar 15, 2017	139.41	140.75	139.03	140.46	140.46	25,691,800
Mar 14, 2017	139.30	139.65	138.84	138.99	138.99	15,309,100
Mar 13, 2017	138.85	139.43	138.82	139.20	139.20	17,421,700
Mar 10, 2017	139.25	139.36	138.64	139.14	139.14	19,612,800
Mar 09, 2017	138.74	138.79	137.05	138.68	138.68	22,155,900
Mar 08, 2017	138.95	139.80	138.82	139.00	139.00	18,707,200

http://finance.yahoo.com/quote/AAPL/history?period1=345398400&period2=1490112000&interval=1d&filter=history&frequency=1d

Yahoo Finance Historical Prices Apple Inc. (AAPL)



Currency in USD

Date	Open	High	Low	Close	Adj Close*	Volume
Mar 21, 2017	142.11	142.80	139.73	139.84	139.84	39,116,800
Mar 20, 2017	140.40	141.50	140.23	141.46	141.46	21,542,000
Mar 17, 2017	141.00	141.00	139.89	139.99	139.99	43,885,000
Mar 16, 2017	140.72	141.02	140.26	140.69	140.69	19,232,000
Mar 15, 2017	139.41	140.75	139.03	140.46	140.46	25,691,800
Mar 14, 2017	139.30	139.65	138.84	138.99	138.99	15,309,100
Mar 13, 2017	138.85	139.43	138.82	139.20	139.20	17,421,700
Mar 10, 2017	139.25	139.36	138.64	139.14	139.14	19,612,800
Mar 09, 2017	138.74	138.79	137.05	138.68	138.68	22,155,900
Mar 08, 2017	138.95	139.80	138.82	139.00	139.00	18,707,200

http://finance.yahoo.com/quote/AAPL/history?period1=345398400&period2=1490112000&interval=1d&filter=history&frequency=1d

Yahoo Finance Historical Prices

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

table.csv

```
Date,Open,High,Low,Close,Volume,Adj Close
2017-03-21,142.110001,142.800003,139.729996,139.839996,39116800,139.839996
2017-03-20,140.399994,141.50,140.229996,141.460007,20213100,141.460007
2017-03-17,141.00,141.00,139.889999,139.990005,43597400,139.990005
2017-03-16,140.720001,141.020004,140.259995,140.690002,19132500,140.690002
2017-03-15,139.410004,140.75,139.029999,140.460007,25566800,140.460007
2017-03-14,139.300003,139.649994,138.839996,138.990005,15189700,138.990005
2017-03-13,138.850006,139.429993,138.820007,139.199997,17042400,139.199997
2017-03-10,139.25,139.360001,138.639999,139.139999,19488000,139.139999
2017-03-09,138.740005,138.789993,137.050003,138.679993,22065200,138.679993
2017-03-08,138.949997,139.800003,138.820007,139.00,18681800,139.00
2017-03-07,139.059998,139.979996,138.789993,139.520004,17267500,139.520004
2017-03-06,139.369995,139.770004,138.600006,139.339996,21155300,139.339996
2017-03-03,138.779999,139.830002,138.589996,139.779999,21108100,139.779999
2017-03-02,140.00,140.279999,138.759995,138.960007,26153300,138.960007
2017-03-01,137.889999,140.149994,137.600006,139.789993,36272400,139.789993
2017-02-28,137.080002,137.440002,136.699997,136.990005,23403500,136.990005
2017-02-27,137.139999,137.440002,136.279999,136.929993,20196400,136.929993
2017-02-24,135.910004,136.660004,135.279999,136.660004,21690900,136.660004
2017-02-23,137.380005,137.479996,136.300003,136.529999,20704100,136.529999
2017-02-22,136.429993,137.119995,136.110001,137.110001,20745300,137.110001
```

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}

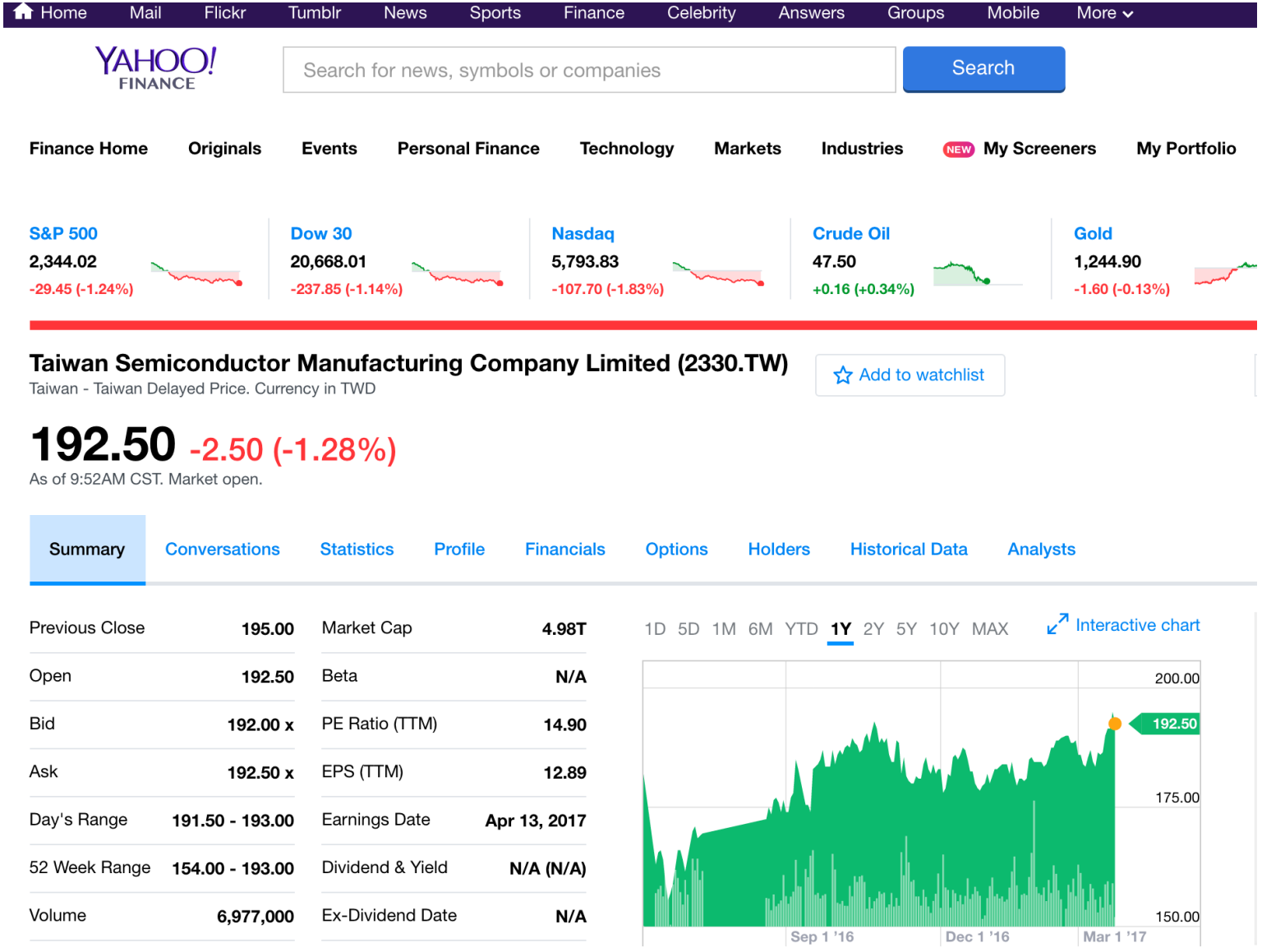
Dow Jones Industrial Average (^DJI)

http://finance.yahoo.com/chart/^DJI

TSEC weighted index (^TWII) - Taiwan

http://finance.yahoo.com/chart/^DJI

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/chart/2330.TW

```
import pandas as pd
import pandas_datareader.data as web
df = web.DataReader('AAPL', data_source='yahoo',
start='1/1/2010', end='3/21/2017')
df.to_csv('AAPL.csv')
df.tail()
```

```
import pandas as pd
import pandas_datareader.data as web
#df = web.DataReader('AAPL', 'yahoo')
df = web.DataReader('AAPL', data_source='yahoo', start='1/1/2010', end='3/21/2017')
#df = web.DataReader('AAPL', data_source='google', start='1/1/2010', end='3/21/2017')
df.to_csv('AAPL.csv')
df.tail()
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2017-03-15	139.410004	140.750000	139.029999	140.460007	25566800	140.460007
2017-03-16	140.720001	141.020004	140.259995	140.690002	19132500	140.690002
2017-03-17	141.000000	141.000000	139.889999	139.990005	43597400	139.990005
2017-03-20	140.399994	141.500000	140.229996	141.460007	20213100	141.460007
2017-03-21	142.110001	142.800003	139.729996	139.839996	39116800	139.839996

```
df = web.DataReader('GOOG',
data_source='yahoo', start='1/1/1980',
end='3/21/2017')
df.head(10)
```

```
df = web.DataReader('GOOG', data_source='yahoo', start='1/1/1980', end='3/21/2017')
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						
2017-03-08	833.510010	838.150024	831.789978	835.369995	988700	835.369995
2017-03-09	836.000000	842.000000	834.210022	838.679993	1259900	838.679993
2017-03-10	843.280029	844.909973	839.500000	843.250000	1701100	843.250000
2017-03-13	844.000000	848.684998	843.250000	845.539978	1149500	845.539978
2017-03-14	843.640015	847.239990	840.799988	845.619995	779900	845.619995
2017-03-15	847.590027	848.630005	840.770020	847.200012	1379600	847.200012
2017-03-16	849.030029	850.849976	846.130005	848.780029	970400	848.780029
2017-03-17	851.609985	853.400024	847.109985	852.119995	1712300	852.119995
2017-03-20	850.010010	850.219971	845.150024	848.400024	1190300	848.400024
2017-03-21	851.400024	853.500000	829.020020	830.460022	2442900	830.460022

df.count()

```
df.count()
```

```
Open          3169
High          3169
Low           3169
Close         3169
Volume        3169
Adj Close     3169
dtype: int64
```

df.ix['2015-12-31']

```
df.ix['2015-12-31']
```

```
Open          7.695000e+02
High          7.695000e+02
Low           7.583400e+02
Close         7.588800e+02
Volume        1.489600e+06
Adj Close     7.588800e+02
Name: 2015-12-31 00:00:00, dtype: float64
```

```
df.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
```

```
import fix_yahoo_finance as yf
data = yf.download("^TWII", start="2017-07-01", end="2017-11-15")
data.to_csv('TWII_201707_201711.csv')
data.tail()
```

```
import fix_yahoo_finance as yf
data = yf.download("^TWII", start="2017-07-01", end="2017-11-15")
data.to_csv('TWII_201707_201711.csv')
data.tail()
```

```
[**********************100%**********************]  1 of 1 downloaded
```

	Open	High	Low	Close	Adj Close	Volume
Date						
2017-11-08	10839.440430	10844.740234	10806.009766	10818.990234	10818.990234	2438000
2017-11-09	10802.950195	10831.379883	10721.870117	10743.269531	10743.269531	2917800
2017-11-10	10713.669922	10742.610352	10659.290039	10732.669922	10732.669922	2277000
2017-11-13	10728.219727	10749.389648	10683.919922	10683.919922	10683.919922	2765100
2017-11-14	10716.589844	10735.080078	10654.580078	10687.179688	10687.179688	2596600

df.loc[start:end]

```
df = df.loc['2017-10-01':'2017-11-15']
```

Python Pandas for Finance

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline
```

```
#Python for Stocks: 1
#Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/
#Import Packages Required
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2015, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

```
#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2015, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2017-03-15	139.410004	140.750000	139.029999	140.460007	25566800	140.460007
2017-03-16	140.720001	141.020004	140.259995	140.690002	19132500	140.690002
2017-03-17	141.000000	141.000000	139.889999	139.990005	43597400	139.990005
2017-03-20	140.399994	141.500000	140.229996	141.460007	20213100	141.460007
2017-03-21	142.110001	142.800003	139.729996	139.839996	39116800	139.839996

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Finance Data from Quandl

```
import quandl
# quandl.ApiConfig.api_key = "YOURAPIKEY"
df = quandl.get("WIKI/AAPL", start_date="2015-01-01", end_date="2017-10-31")
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

```
import quandl
df = quandl.get("WIKI/AAPL", start_date="2015-01-01", end_date="2017-10-31")
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
Date												
2017-10-25	156.91	157.5500	155.27	156.405	20126554.0	0.0	1.0	156.91	157.5500	155.27	156.405	20126554.0
2017-10-26	157.23	157.8295	156.78	157.410	16751691.0	0.0	1.0	157.23	157.8295	156.78	157.410	16751691.0
2017-10-27	159.29	163.6000	158.70	163.050	43904150.0	0.0	1.0	159.29	163.6000	158.70	163.050	43904150.0
2017-10-30	163.89	168.0700	163.72	166.720	43923292.0	0.0	1.0	163.89	168.0700	163.72	166.720	43923292.0
2017-10-31	167.90	169.6499	166.94	169.040	35474672.0	0.0	1.0	167.90	169.6499	166.94	169.040	35474672.0

Source: https://www.quandl.com/tools/python

Python Pandas for Finance

```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
```

```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
<matplotlib.axes._subplots.AxesSubplot at 0x1150bac88>
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0),
rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0),
rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'],
color='blue') #df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
<matplotlib.text.Text at 0x115630860>
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'], color='blue') #df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')
```

<matplotlib.text.Text at 0x115630860>

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

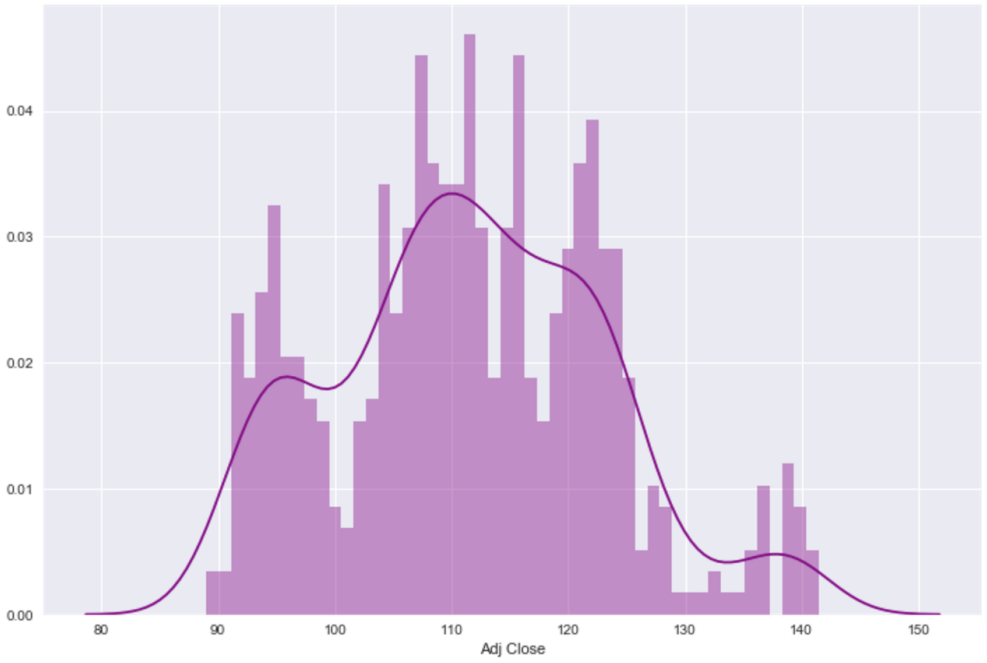
Python Pandas for Finance

```
plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(),
bins=50, color='purple')
```

```
plt.figure(figsize=(12,8))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')
```

```
/Users/imyday/anaconda/lib/python3.6/site-packages/statsmodels/nonparametric/kdetools
g: using a non-integer number instead of an integer will result in an error in the fu
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x116309780>
```



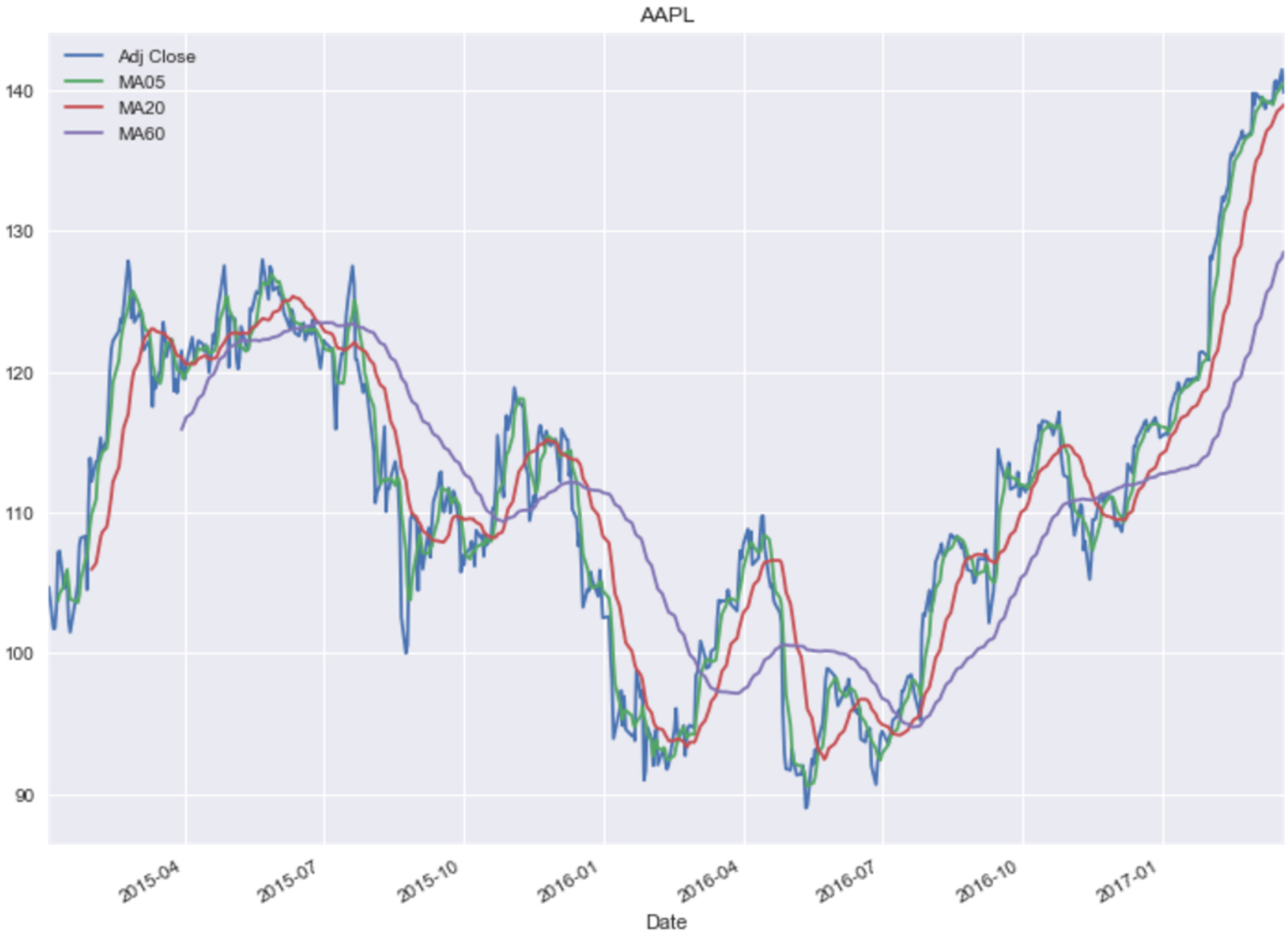
Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean()
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days

df2 = pd.DataFrame({'Adj Close': df['Adj
Close'],'MA05': df['MA05'],'MA20':
df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True,
title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

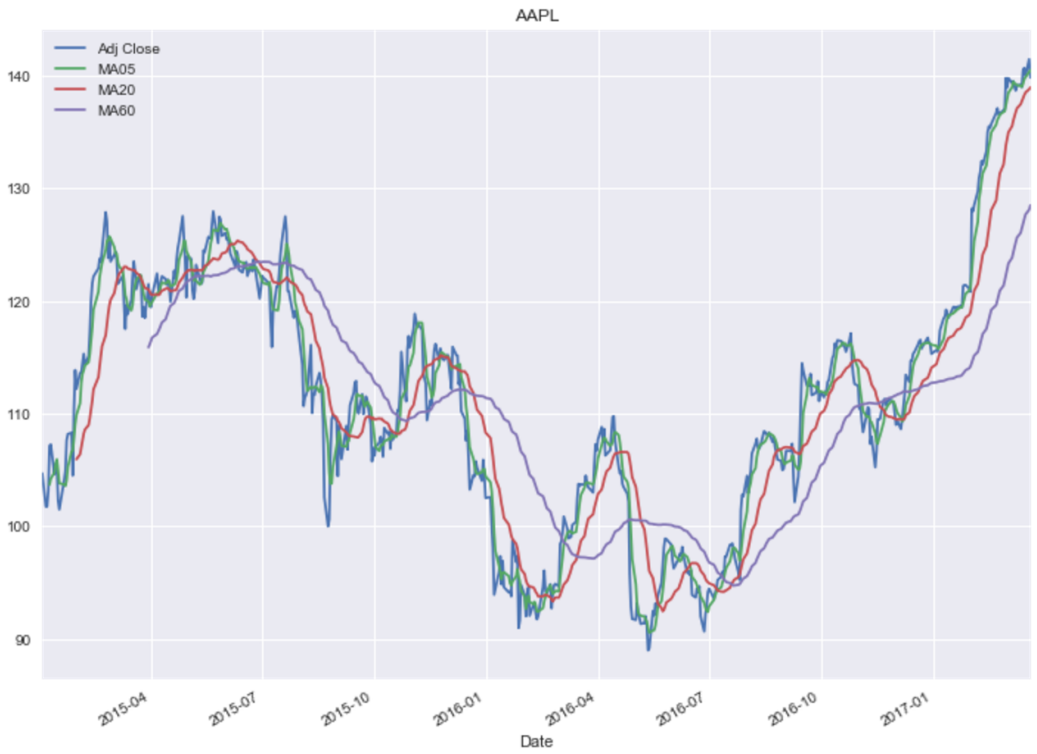
Python Pandas for Finance



Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days

df2 = pd.DataFrame({'Adj Close': df['Adj Close'],'MA05': df['MA05'],'MA20': df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
```

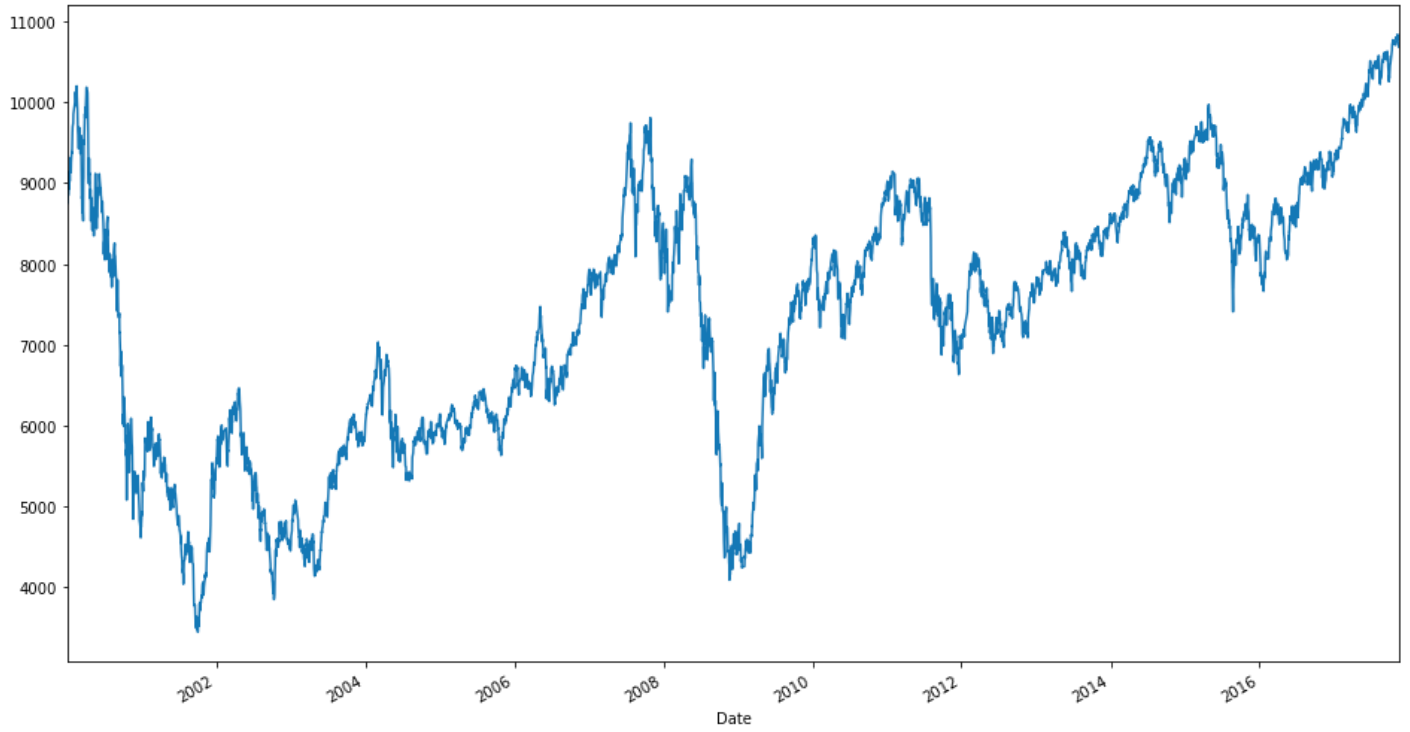


Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

```
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2015, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()

df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'], color='blue') #df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')

plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')

# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
df2 = pd.DataFrame({'Adj Close': df['Adj Close'],'MA05': df['MA05'],'MA20': df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

```
import matplotlib.pyplot as plt
%matplotlib inline
import fix_yahoo_finance as yf
df = yf.download("^TWII", start="2000-01-01", end="2017-11-15")
df.to_csv('YF_TWII_2000_2017.csv')
print(df.head())
fig = plt.figure(figsize=(16,9))
df["Adj Close"].plot()
fig.show()
```



candlestick_ohlc

```
import matplotlib.pyplot as plt

from matplotlib.finance
import candlestick_ohlc
```

Source: https://matplotlib.org/examples/pylab_examples/finance_demo.html

```
import matplotlib.pyplot as plt
from matplotlib.finance import candlestick_ohlc
```

Source: https://ntguardian.wordpress.com/2016/09/19/introduction-stock-market-data-python-1/

daily_to_weekly

```
#Convert Daily Data to Weekly Data
def daily_to_weekly(df):
    #dfWeekly = daily_to_weekly(df)
    #df.sort_index(axis=0, level=None, ascending=True, inplace=True)
    Open = df.Open.resample('W-Fri').first() #W #W-MON #W-Fri
    High = df.High.resample('W-Fri').max()
    Low = df.Low.resample('W-Fri').min()
    Close = df.Close.resample('W-Fri').last()
    Volume = df.Volume.resample('W-Fri').sum()
    Adj_Close = df["Adj Close"].resample('W-Fri').last()
    dfWeekly = pd.concat([Open, High, Low, Close, Volume, Adj_Close], axis=1)
    dfWeekly = dfWeekly[pd.notnull(dfWeekly['Adj Close'])]
    return dfWeekly
```

daily_to_monthly

```
#Convert Daily Data to Monthly Data
def daily_to_monthly(df):
    #dfMonthly = daily_to_monthly(df)
    Open = df.Open.resample('M').first()
    High = df.High.resample('M').max()
    Low = df.Low.resample('M').min()
    Close = df.Close.resample('M').last()
    Volume = df.Volume.resample('M').sum()
    Adj_Close = df["Adj Close"].resample('M').last()
   dfMonthly = pd.concat([Open, High, Low, Close, Volume, Adj_Close], axis=1)
    dfMonthly = dfMonthly[pd.notnull(dfMonthly['Adj Close'])]
    return dfMonthly
```

TA-Lib:
Technical Analysis Library

TA-Lib : Technical Analysis Library

Home

Products
Downloads
Purchase
Support

Function List

Source Code
Community Forum
Useful Links

About Us

Multi-Platform Tools for Market Analysis ...

TA-Lib is widely used by trading software developers requiring to perform technical analysis of financial market data.

- Includes 200 indicators such as ADX, MACD, RSI, Stochastic, Bollinger Bands etc... (more info)
- Candlestick pattern recognition
- Open-source API for C/C++, Java, Perl, Python and 100% Managed .NET

Free Open-Source Library

TA-Lib is available under a BSD License allowing it to be integrated in your own open-source or commercial application. (more info)

Commercial Application

TA-Lib is also available as an easy to install Excel Add-Ins. Try it for free!

http://ta-lib.org/

Stochastic Oscillator (KD)

```
#Stochastic oscillator %D
def KDJ(df, n, m1, m2):
    #KDJ(df, 9, 3, 3)
    KDJ_n = n
    KDJ_m1 = m1
    KDJ_m2 = m2

    df['Low_n'] = pd.rolling_min(df['Low'], KDJ_n)
    df['Low_n'].fillna(value=pd.expanding_min(df['Low']), inplace=True)
    df['High_n'] = pd.rolling_max(df['High'], KDJ_n)
    df['High_n'].fillna(value=pd.expanding_max(df['High']), inplace=True)

    df['RSV'] = (df['Close'] - df['Low_n']) / (df['High_n'] - df['Low_n']) * 100

    df['KDJ_K'] = pd.ewma(df['RSV'], KDJ_m1)
    df['KDJ_D'] = pd.ewma(df['KDJ_K'], KDJ_m2)
    df['KDJ_J'] = 3 * df['KDJ_K'] - 2 * df['KDJ_D']
    return df
```

Source: https://www.quantopian.com/posts/technical-analysis-indicators-without-talib-code

Bollinger Bands

```
#Bollinger Bands
def BBANDS20(df, n):
    MA = pd.Series(pd.rolling_mean(df['Close'], n))
    MSD = pd.Series(pd.rolling_std(df['Close'], n))
    b1 = 4 * MSD / MA
    B1 = pd.Series(b1, name = 'BollingerB_' + str(n))
    df = df.join(B1)
    b2 = (df['Close'] - MA + 2 * MSD) / (4 * MSD)
    B2 = pd.Series(b2, name = 'Bollinger%b_' + str(n))
    df = df.join(B2)
    return df
```

Source: https://www.quantopian.com/posts/technical-analysis-indicators-without-talib-code

Bollinger Bands

```
#BB Bollinger Bands BB_20
def BB_20(df):
    df['BB_MA20'] = pd.stats.moments.rolling_mean(df["Adj Close"], 20)
    df['BB_SD20'] = pd.stats.moments.rolling_std(df['Adj Close'],20)
    df['BB_UpperBand'] = df['BB_MA20'] + (df['BB_SD20']*2) # Default 2*SD
    df['BB_LowerBand'] = df['BB_MA20'] - (df['BB_SD20']*2)
    df['BB_PB'] = (df['Adj Close'] - df['BB_LowerBand']) / (df['BB_UpperBand'] -
df['BB_LowerBand'])
    df['BB_BW'] = (df['BB_UpperBand'] - df['BB_LowerBand']) / df['BB_MA20']
    df['BB_UpperBand_1SD'] = df['BB_MA20'] + (df['BB_SD20'])
    df['BB_LowerBand_1SD'] = df['BB_MA20'] - (df['BB_SD20'])
    #BB_PB: Bollinger Band Percent b (PB)
    #BB_BW: Bollinger Band Band Width (BW)
    return df
```

Source: https://www.quantopian.com/posts/technical-analysis-indicators-without-talib-code

Examples: Python Pandas for Finance

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)
```

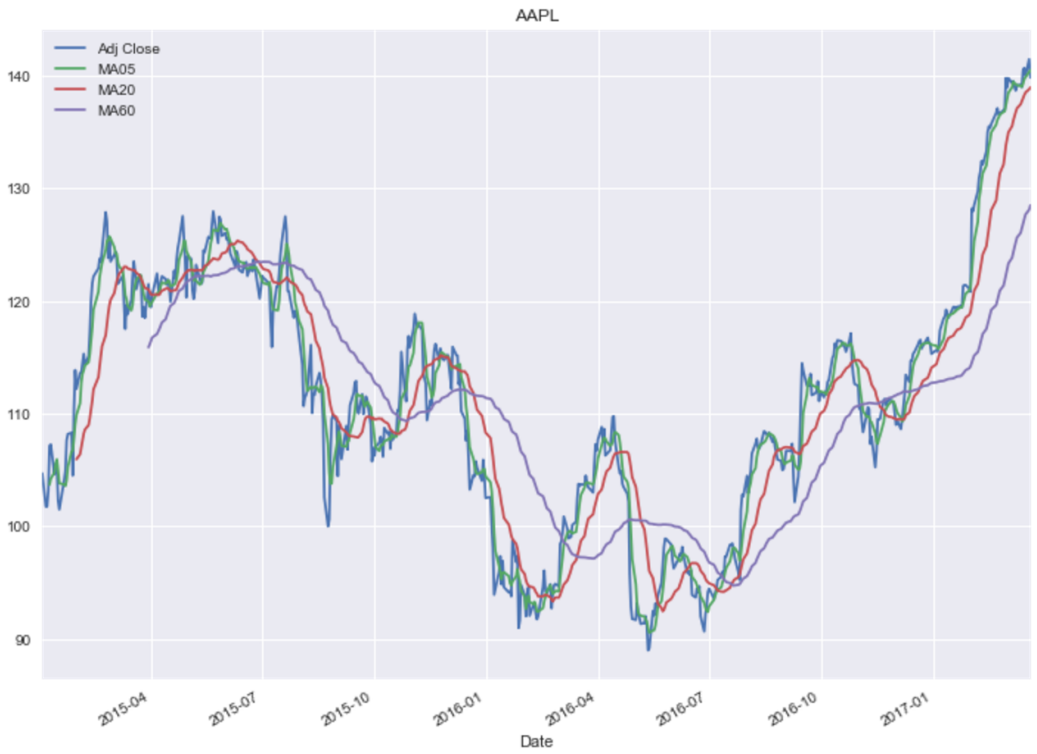
```
data/financedata/AAPL_Yahoo_Finance.csv
```

```
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2015, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()

df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'], color='blue') #df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')

plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')

# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
df2 = pd.DataFrame({'Adj Close': df['Adj Close'],'MA05': df['MA05'],'MA20': df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

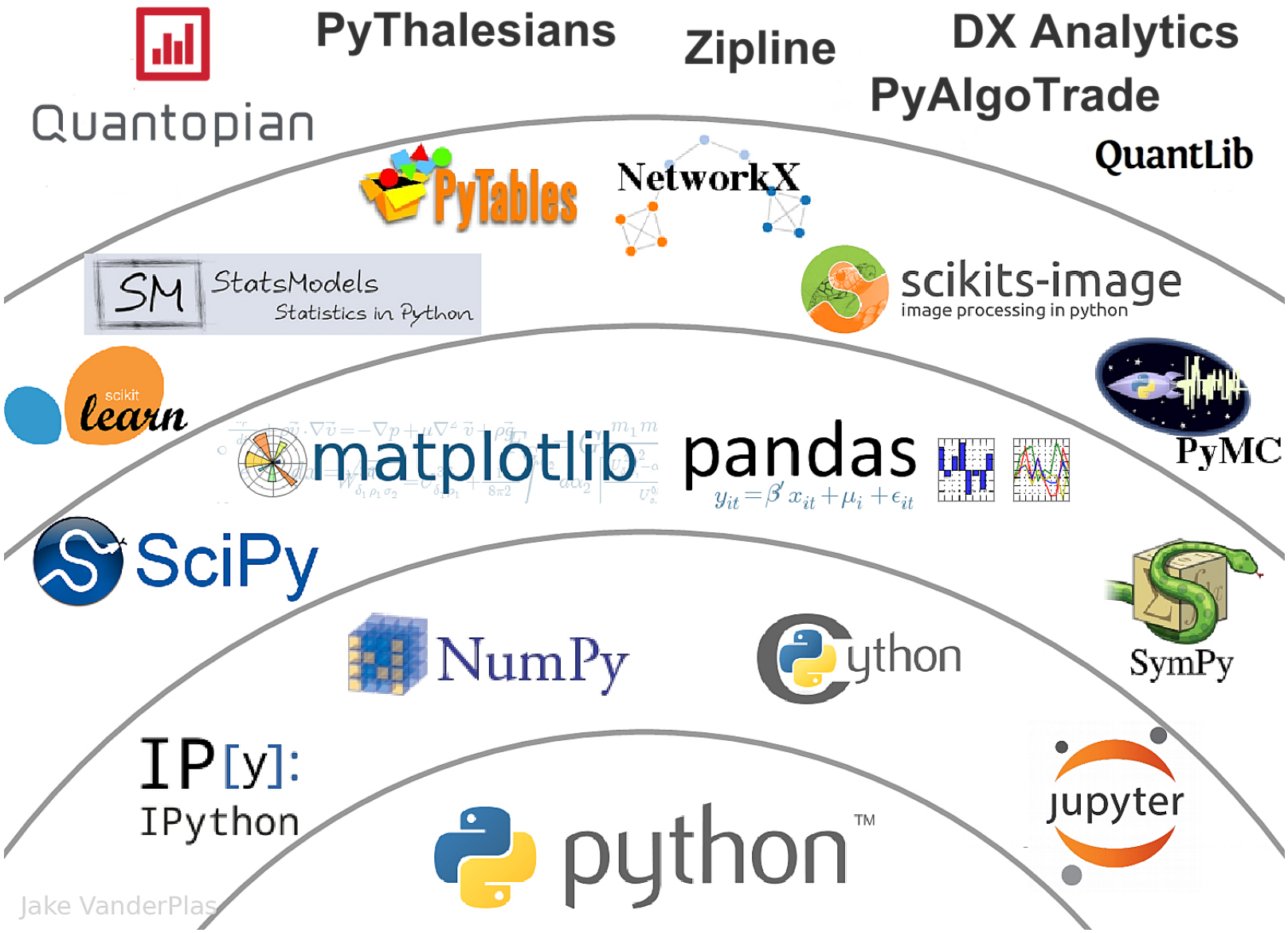
Python Pandas for Finance

```
# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days

df2 = pd.DataFrame({'Adj Close': df['Adj Close'],'MA05': df['MA05'],'MA20': df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
```



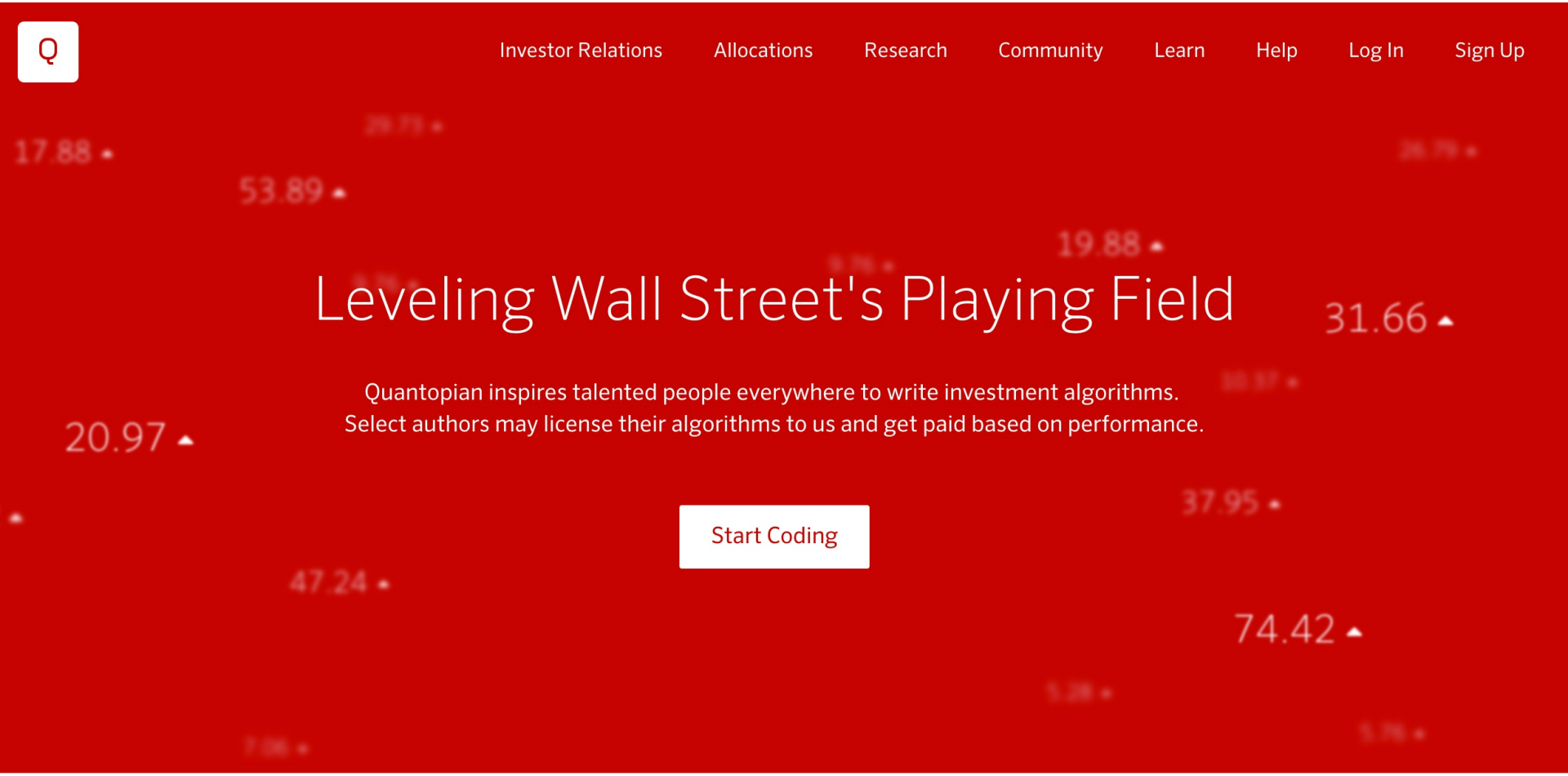
Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

The Quant Finance PyData Stack



Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#/5

Quantopian



https://www.quantopian.com/

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
- Python Programming, https://pythonprogramming.net/
- Python, https://www.python.org/
- Python Programming Language, http://pythonprogramminglanguage.com/
- Numpy, http://www.numpy.org/
- Pandas, http://pandas.pydata.org/

References

- Wes McKinney (2017), "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython", 2nd Edition, O'Reilly Media. https://github.com/wesm/pydata-book
- Avinash Jain (2017), Introduction To Python Programming, Udemy, https://www.udemy.com/pythonforbeginnersintro/
- Alfred Essa (2015), Awesome Data Science: 1.0 Jupyter Notebook Tour, https://www.youtube.com/watch?v=e9cSF3eVQv0
- Ties de Kok (2017), Learn Python for Research, https://github.com/TiesdeKok/LearnPythonforResearch
- Ivan Idris (2015), Numpy Beginner's Guide, Third Edition, Packt Publishing
- Numpy Tutorial, https://docs.scipy.org/doc/numpy-dev/user/quickstart.html