

大數據分析

(Big Data Analysis)

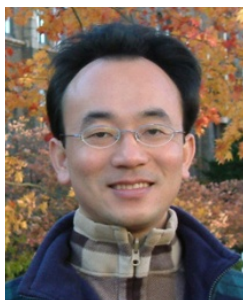
Python Pandas 大數據量化分析

(Quantitative Big Data Analysis with Pandas in Python)

1091BDA04

MBA, IM, NTPU (M5127) (Fall 2020)

Wed 7, ,8, 9 (15:10-18:00) (B8F40)



Min-Yuh Day

戴敏育

Associate Professor

副教授

Institute of Information Management, National Taipei University

國立臺北大學 資訊管理研究所

<https://web.ntpu.edu.tw/~myday>

2020-10-21



課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2020/09/16	大數據分析介紹 (Introduction to Big Data Analysis)
2	2020/09/23	AI人工智慧與大數據分析 (AI and Big Data Analysis)
3	2020/09/30	Python 大數據分析基礎 (Foundations of Big Data Analysis in Python)
4	2020/10/07	數位沙盒第一堂課：數位沙盒服務平台簡介 (Digital Sandbox Lesson 1: Introduction to FintechSpace Digital Sandbox)
5	2020/10/14	數位沙盒第二堂課：工程師操作說明與實作教學 (Digital Sandbox Lesson 2: Hands-on Practices)
6	2020/10/21	Python Pandas 大數據量化分析 (Quantitative Big Data Analysis with Pandas in Python)

課程大綱 (Syllabus)

- | 週次 (Week) | 日期 (Date) | 內容 (Subject/Topics) |
|-----------|------------|--|
| 7 | 2020/10/28 | Python Scikit-Learn 機器學習 I
(Machine Learning with Scikit-Learn In Python I) |
| 8 | 2020/11/04 | 數位沙盒第三堂課：學生小組討論實作與成果發表
(Digital Sandbox Lesson 3: Learning Teams
Hands-on Project Discussion and Project Presentation) |
| 9 | 2020/11/11 | 期中報告 (Midterm Project Report) |
| 10 | 2020/11/18 | Python Scikit-Learn 機器學習 II
(Machine Learning with Scikit-Learn In Python II) |
| 11 | 2020/11/25 | TensorFlow 深度學習金融大數據分析 I
(Deep Learning for Finance Big Data Analysis with TensorFlow I) |
| 12 | 2020/12/02 | 大數據分析個案研究
(Case Study on Big Data Analysis) |

課程大綱 (Syllabus)

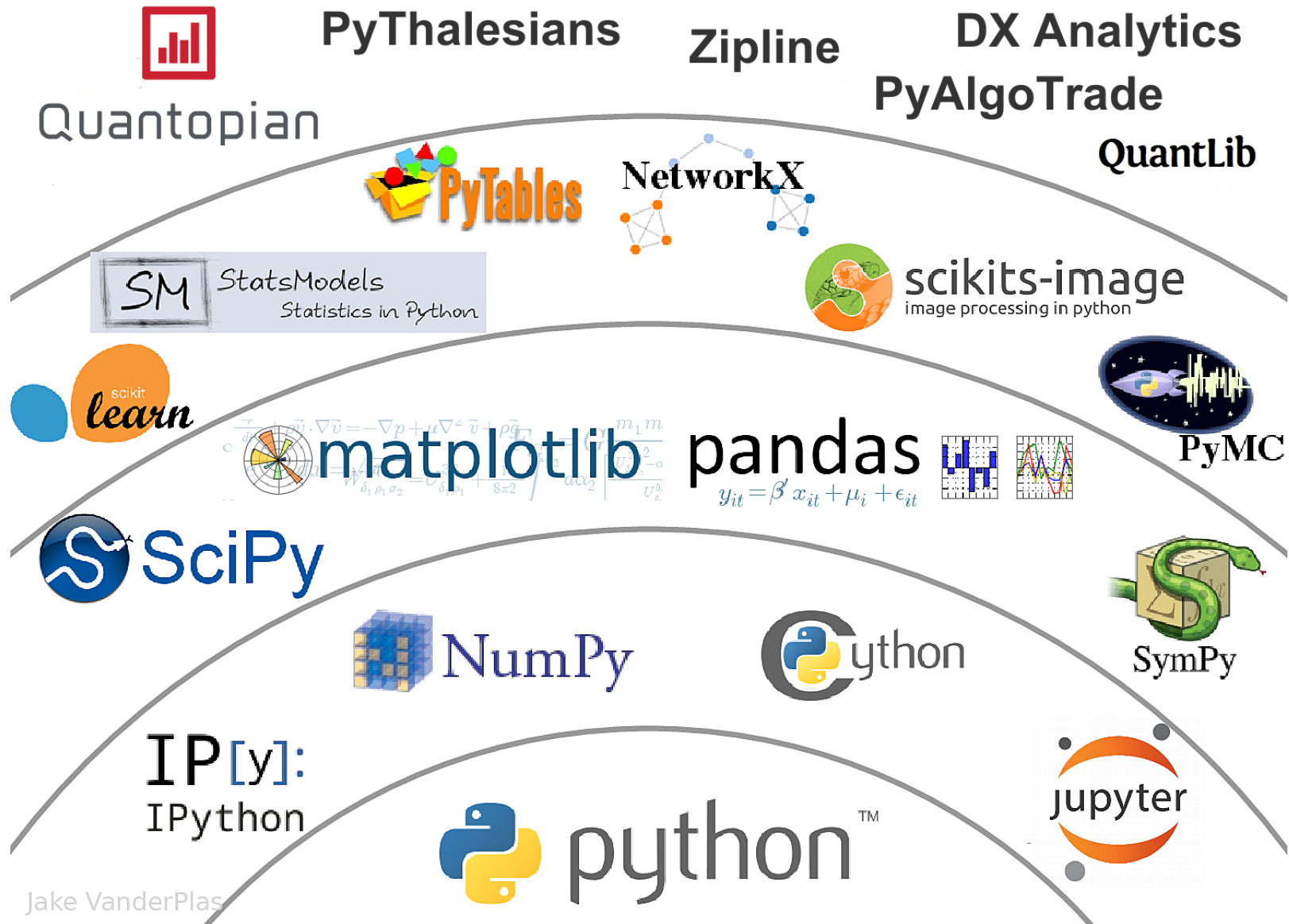
- | 週次 (Week) | 日期 (Date) | 內容 (Subject/Topics) |
|-----------|------------|---|
| 13 | 2020/12/09 | TensorFlow 深度學習金融大數據分析 II
(Deep Learning for Finance Big Data Analysis with TensorFlow II) |
| 14 | 2020/12/16 | TensorFlow 深度學習金融大數據分析 III
(Deep Learning for Finance Big Data Analysis with TensorFlow III) |
| 15 | 2020/12/23 | AI 機器人理財顧問
(Artificial Intelligence for Robo-Advisors) |
| 16 | 2020/12/30 | 金融科技智慧型交談機器人
(Conversational Commerce and Intelligent Chatbots for Fintech) |
| 17 | 2021/01/06 | 期末報告 I (Final Project Report I) |
| 18 | 2021/01/13 | 期末報告 II (Final Project Report I) |

**Quantitative
Big Data Analysis
with
Pandas in Python**

Outline

- **Quantitative Big Data Analysis with Pandas in Python**
 - **Pandas**
 - Data structures and data analysis tools

The Quant Finance PyData Stack

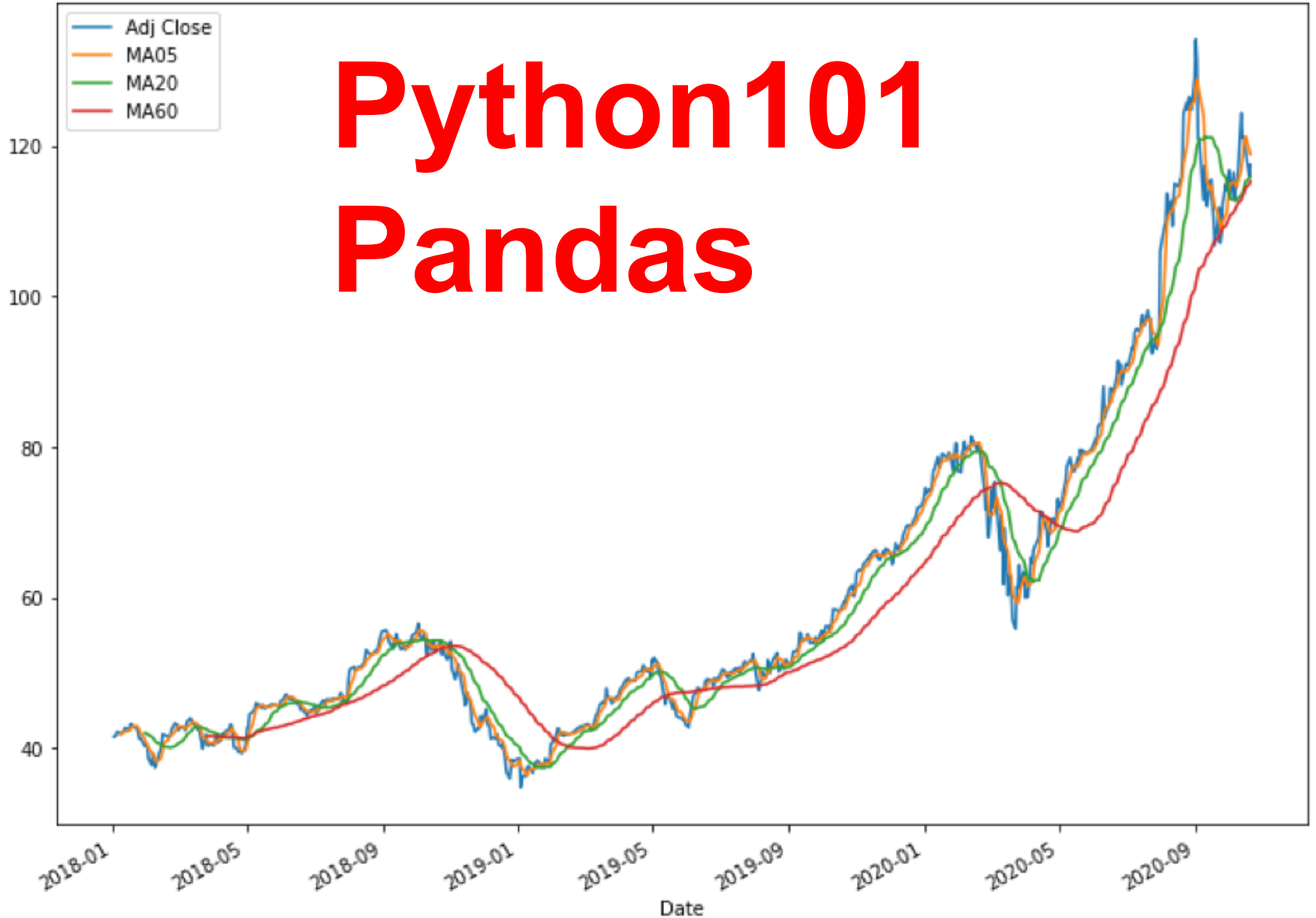


Jake VanderPlas

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

AAPL

Python101 Pandas



<https://tinyurl.com/aintpuython101>

Python in Google Colab (Python101)

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT>



python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

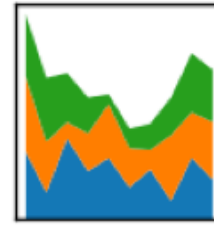
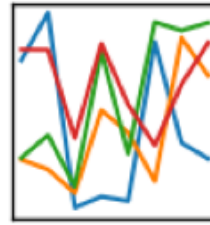
```
1 import pandas as pd
2 import pandas_datareader.data as web
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import datetime as dt
6 %matplotlib inline
7
8 #Read Stock Data from Yahoo Finance
9 end = dt.datetime.now()
10 #start = dt.datetime(end.year-2, end.month, end.day)
11 start = dt.datetime(2018, 1, 1)
12 df = web.DataReader("AAPL", 'yahoo', start, end)
13 df.to_csv('AAPL.csv')
14 print(df.tail())
15 df2 = pd.read_csv('AAPL.csv') #df.from_csv('AAPL.csv')
16 print(df2.tail())
17
18 df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
19 plt.figure(figsize=(12,9))
20 top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
21 bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
22 top.plot(df.index, df['Adj Close'], color='blue') #df.index gives the dates
23 bottom.bar(df.index, df['Volume'])
24
25 # set the labels
26 top.axes.get_xaxis().set_visible(False)
27 top.set_title('AAPL')
28 top.set_ylabel('Adj Close')
29 bottom.set_ylabel('Volume')
30 plt.figure(figsize=(12,9))
31 sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')
32
33 # simple moving averages
34 df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
35 df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
36 df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
37 df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60']})
38 df2.plot(figsize=(12, 9), legend=True, title='AAPL')
39 df2.to_csv('AAPL_MA.csv')
40 fig = plt.gcf()
41 fig.set_size_inches(12, 9)
42 fig.savefig('AAPL_plot.png', dpi=300)
```

Python101 Pandas

<https://tinyurl.com/aintpuython101>

pandas

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



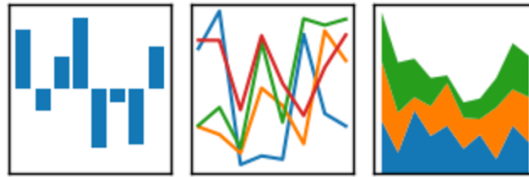
Python

Pandas

pandas

pandas

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



[overview](#) // [get pandas](#) // [documentation](#) // [community](#) // [talks](#)

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

<http://pandas.pydata.org/>

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](#)

pandas

Python Data Analysis Library

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

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

pandas-datareader

🏠 pandas-datareader
latest

Search docs

What's New
Remote Data Access
Caching queries
Other Data Sources
Data Readers



Take the [Python User's Survey](#) and let us know more about your usage of Python.

Sponsored · Ads served ethically

📖 Read the Docs v: latest ▾

pandas-datareader

Up to date remote data access for pandas, works for multiple versions of pandas.

pypi **v0.7.0** build **failing** coverage **79%** docs **passing** health **95%**

⚠ Warning

As of v0.6.0 Yahoo!, Google Options, Google Quotes and EDGAR have been immediately deprecated due to large changes in their API and no stable replacement.

📌 Note

As of v0.6.0 Google finance is still functioning for historical price data, although there are frequent reports of failures. Failure is frequently encountered when bulk downloading historical price data.

Usage

Starting in 0.19.0, pandas no longer supports `pandas.io.data` or `pandas.io.wb`, so you must replace your imports from `pandas.io` with those from `pandas_datareader`:

```
from pandas.io import data, wb # becomes
from pandas_datareader import data, wb
```

Many functions from the data module have been included in the top level API.

<https://pandas-datareader.readthedocs.io/en/latest/>

Quandl

ALTERNATIVE DATA

FINANCIAL DATA

Quandl

DOCS & HELP

BLOG

LOG IN

SIGN UP



Analysis Tools

ALL TOOLS

API

R

PYTHON

EXCEL



Get Financial Data Directly into Python

Get millions of financial and economic datasets from hundreds of publishers directly into Python.

Load Quandl Data Directly Into Python

All the Data You Want

Quandl unifies financial and economic datasets from hundreds of publishers on a single user-friendly platform.

Directly Into Python

<https://www.quandl.com/tools/python>

PyDatastream



Search projects



Help

Donate

Log in

Register

PyDatastream 0.5.1



```
pip install PyDatastream
```



Last released: Nov 17, 2017

Python interface to the Thomson Reuters Dataworks Enterprise (Datastream) API

Navigation

Project description

Release history

Download files

Project description

PyDatastream is a Python interface to the **Thomson Dataworks Enterprise (DWE)** SOAP API (non free), with some convenience functions for retrieving Datastream data specifically. This package requires valid credentials for this API.

For the documentation please refer to README.md inside the package or on the GitHub (<https://github.com/vfilimonov/pydatastream/blob/master/README.md>).

<https://pypi.org/project/PyDatastream/>

pandasSDMX

pandaSDMX: Statistical Data and Metadata eXchange in Python

Table Of Contents

[pandaSDMX: Statistical Data and Metadata eXchange in Python](#)

- [Supported data providers](#)
- [Main features](#)
- [Example](#)
- [Quick install](#)
- [pandaSDMX Links](#)
- [Table of contents](#)

[Indices and tables](#)

This Page

[Show Source](#)

Quick search



Take the [Python User's Survey](#) and let us know more about your usage of Python.

Sponsored · Ads served ethically

pandaSDMX is an Apache 2.0-licensed [Python](#) client to retrieve and acquire statistical data and metadata disseminated in [SDMX 2.1](#), an ISO-standard widely used by institutions such as statistics offices, central banks, and international organisations. pandaSDMX exposes datasets and related structural metadata including dataflows, codelists, and datastructure definitions as [pandas](#) Series or multi-indexed DataFrames. Many other output formats and storage backends are available thanks to [Odo](#).

Supported data providers

pandaSDMX ships with built-in support for the following agencies (others may be configured by the user):

- [Australian Bureau of Statistics \(ABS\)](#)
- [European Central Bank \(ECB\)](#)
- [Eurostat](#)
- [French National Institute for Statistics \(INSEE\)](#)
- [Instituto Nacional de la Estadística y Geografía - INEGI \(Mexico\)](#)
- [International Monetary Fund \(IMF\) - SDMX Central only](#)
- [International Labour Organization \(ILO\)](#)
- [Italian statistics Office \(ISTAT\)](#)
- [Norges Bank \(Norway\)](#)
- [Organisation for Economic Cooperation and Development \(OECD\)](#)
- [United Nations Statistics Division \(UNSD\)](#)
- [UNESCO \(free registration required\)](#)
- [World Bank - World Integrated Trade Solution \(WITS\)](#)

Fred API



ECONOMIC RESEARCH
FEDERAL RESERVE BANK of ST. LOUIS

REGISTER | SIGN IN

FRED® Economic Data Information Services Publications Working Papers Economists About

St. Louis Fed Home

Home > Developer API > FRED API

[API Keys](#) | [Terms of Use](#)

FRED® API

[General Documentation](#) | [API](#) | [Toolkits](#)

The FRED® API is a web service that allows developers to write programs and build applications that retrieve economic data from the [FRED®](#) and [ALFRED®](#) websites hosted by the [Economic Research Division](#) of the [Federal Reserve Bank of St. Louis](#). Requests can be customized according to data source, release, category, series, and other preferences.

General Documentation

- [Overview](#)
- [What is FRED®?](#)
- [What is ALFRED®?](#)
- [FRED® versus ALFRED®](#)
- [Real-Time Periods](#)
- [Errors](#)

API

Categories

- [fred/category](#) – Get a category.
- [fred/category/children](#) – Get the child categories for a specified parent category.
- [fred/category/related](#) – Get the related categories for a category.
- [fred/category/series](#) – Get the series in a category.
- [fred/category/tags](#) – Get the tags for a category.
- [fred/category/related_tags](#) – Get the related tags for a category.

<https://research.stlouisfed.org/docs/api/fred/>

pandas: powerful Python data analysis toolkit

pandas 0.19.2 documentation »

Table Of Contents

- What's New
- Installation
- Contributing to pandas
- Frequently Asked Questions (FAQ)
- Package overview
- 10 Minutes to pandas
- Tutorials
- Cookbook
- Intro to Data Structures
- Essential Basic Functionality
- Working with Text Data
- Options and Settings
- Indexing and Selecting Data
- MultIndex / Advanced Indexing
- Computational tools
- Working with missing data
- Group By: split-apply-combine
- Merge, join, and concatenate
- Reshaping and Pivot Tables
- Time Series / Date functionality
- Time Deltas
- Categorical Data
- Visualization
- Style
- IO Tools (Text, CSV, HDF5, ...)
- Remote Data Access
- Enhancing Performance
- Sparse data structures
- Caveats and Gotchas
- rpy2 / R interface
- pandas Ecosystem
- Comparison with R / R libraries
- Comparison with SQL
- Comparison with SAS
- API Reference

pandas: powerful Python data analysis toolkit

PDF Version

Zipped HTML

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

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

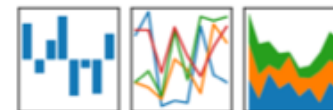
Source Repository: <http://github.com/pydata/pandas>

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



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

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.

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

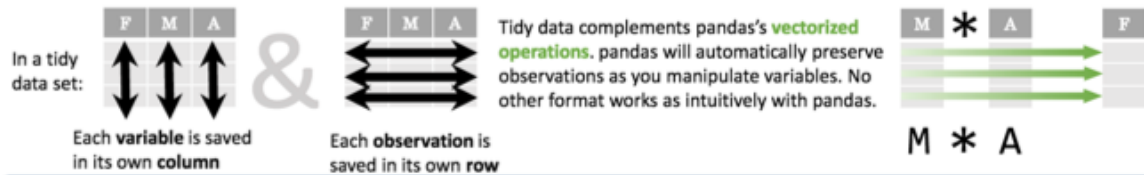
pandas	SAS
DataFrame	data set
column	variable
row	observation
groupby	BY-group
NaN	.

Python Pandas Cheat Sheet

Data Wrangling with pandas Cheat Sheet

<http://pandas.pydata.org>

Tidy Data – A foundation for wrangling in pandas



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.

n	v	a	b	c
d	1	4	7	10
e	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']))
```

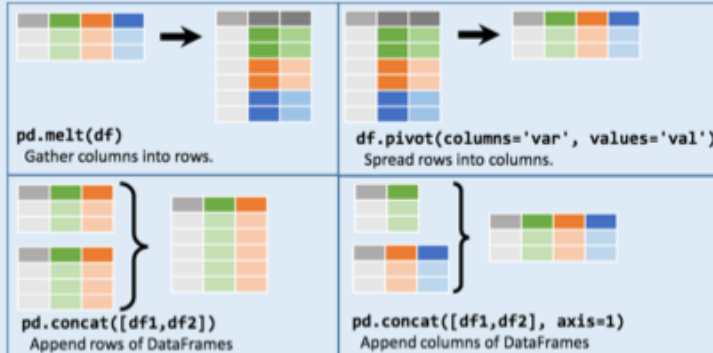
Create DataFrame with a MultiIndex

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



```
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.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.
```

```
df.head(n)
Select first n rows.
```

```
df.tail(n)
Select last n rows.
```

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

regex	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 Lusting, Princeton Consultants

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

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

Pandas DataFrame

```
type(df)
```

```
type(df)
```

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

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

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

```
dates = pd.date_range('20181001',
periods=6)
dates
```

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 print('pandas imported')
```

pandas imported

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

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

```
1 dates = pd.date_range('20181001', periods=6)
2 dates
```

```
DatetimeIndex(['2018-10-01', '2018-10-02', '2018-10-03', '2018-10-04',
               '2018-10-05', '2018-10-06'],
              dtype='datetime64[ns]', freq='D')
```

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

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

	A	B	C	D
2018-10-01	-0.336188	0.584621	-1.061433	-0.036278
2018-10-02	0.903683	-0.839723	-0.270219	-1.099606
2018-10-03	0.920208	-0.240353	-0.818598	-1.105489
2018-10-04	0.221045	-0.314589	0.042071	-1.447280
2018-10-05	0.946862	-1.570305	-1.009180	-0.375659
2018-10-06	-0.225148	0.510691	2.002372	-0.335005

```
df = pd.DataFrame(np.random.randn(3,5),
index=['student1', 'student2', 'student3']
, columns=list('ABCDE'))
df
```

```
1 df = pd.DataFrame(np.random.randn(3,5), index=['student1', 'student2', 'student3'], columns=list('ABCDE'))
2 df
```

	A	B	C	D	E
student1	-0.346884	-1.232934	-0.302072	-1.345084	-0.723880
student2	1.090955	-0.010483	1.280072	-0.253958	-0.030604
student3	0.325660	0.808956	-0.395820	-1.498926	1.603471

```
df2 = pd.DataFrame({ 'A' : 1.,
' B' : pd.Timestamp('20181001'),
' 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
```

```
1 df2 = pd.DataFrame({ 'A' : 1.,
2 ' B' : pd.Timestamp('20181001'),
3 ' C' : pd.Series(2.5,index=list(range(4)),dtype='float32'),
4 ' D' : np.array([3] * 4,dtype='int32'),
5 ' E' : pd.Categorical(["test","train","test","train"]),
6 ' F' : 'foo' })
7 df2
```

	A	B	C	D	E	F
0	1.0	2018-10-01	2.5	3	test	foo
1	1.0	2018-10-01	2.5	3	train	foo
2	1.0	2018-10-01	2.5	3	test	foo
3	1.0	2018-10-01	2.5	3	train	foo

df2.dtypes

```
df2.dtypes
```

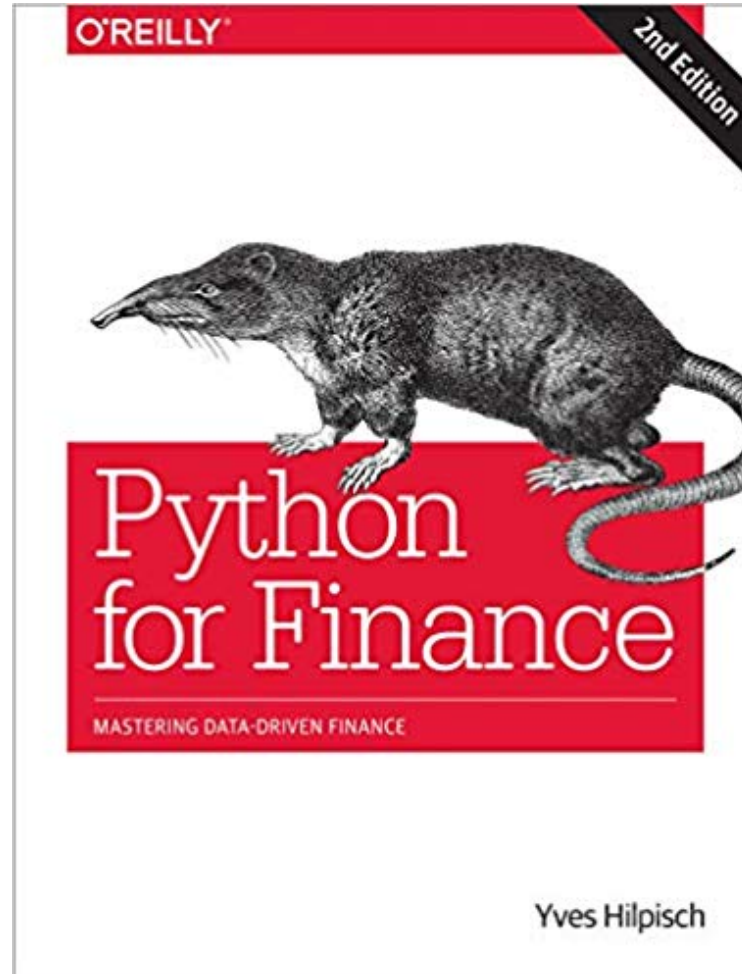
```
A          float64  
B    datetime64[ns]  
C          float32  
D          int32  
E          category  
F          object  
dtype: object
```


Python Pandas for Finance

Yves Hilpisch (2018),

Python for Finance: Mastering Data-Driven Finance,

O'Reilly



<https://github.com/yhilpisch/py4fi2nd>

Source: <https://www.amazon.com/Python-Finance-Mastering-Data-Driven/dp/1492024333>

! pip install pandas_datareader

```
1 ! pip install pandas_datareader
```

```
Collecting pandas_datareader
```

```
  Downloading https://files.pythonhosted.org/packages/cc/5c/ea5b6dcfd0f55c5fb1e37fb45335ec01cceca199b8a79339137f5ed269e0/pandas\_datarea
```

```
100% |██████████████████████████████████████████| 112kB 2.7MB/s
```

```
Collecting lxml (from pandas_datareader)
```

```
  Downloading https://files.pythonhosted.org/packages/03/a4/9eea8035fc7c7670e5eab97f34ff2ef0ddd78a491bf96df5accedb0e63f5/lxml-4.2.5-cp3
```

```
100% |██████████████████████████████████████████| 5.8MB 7.5MB/s
```

```
Requirement already satisfied: pandas>=0.19.2 in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (0.22.0)
```

```
Requirement already satisfied: requests>=2.3.0 in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (2.18.4)
```

```
Requirement already satisfied: wrapt in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (1.10.11)
```

```
Requirement already satisfied: python-dateutil>=2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19.2->pandas_datareader) (2.6.0)
```

```
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19.2->pandas_datareader) (1.14.6)
```

```
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19.2->pandas_datareader) (2018.5)
```

```
Requirement already satisfied: idna<2.7,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (2.6)
```

```
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (3.0.4)
```

```
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (2017.7.27.1)
```

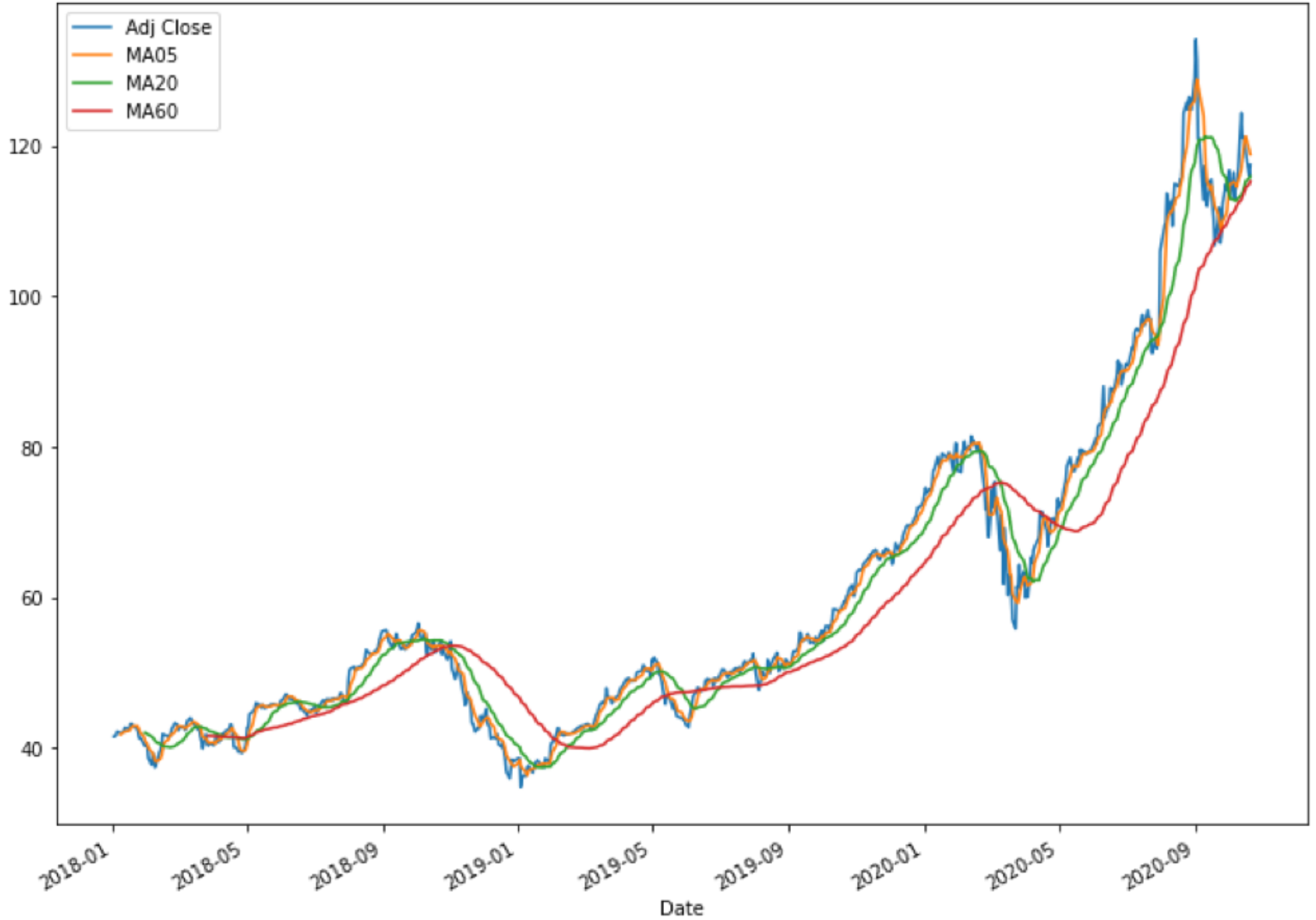
```
Requirement already satisfied: urllib3<1.23,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (1.21.1)
```

```
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2->pandas>=0.19.2->pandas_datareader) (1.11.0)
```

```
Installing collected packages: lxml, pandas-datareader
```

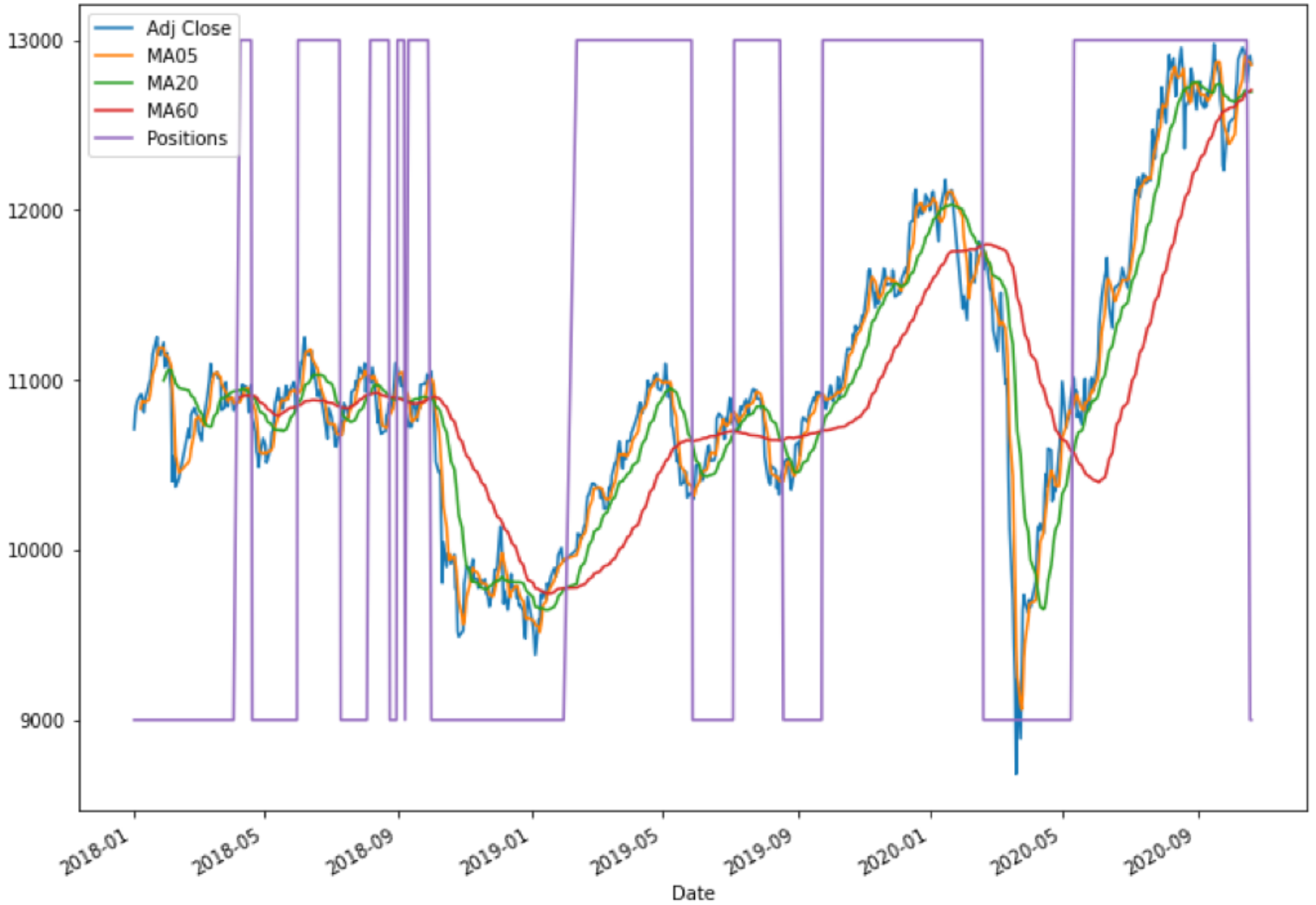
```
Successfully installed lxml-4.2.5 pandas-datareader-0.7.0
```

AAPL



<https://tinyurl.com/aintpupython101>

TWII



Finance Data from Yahoo Finance

```
# !pip install pandas_datareader
import pandas_datareader.data as web
import datetime as dt
#Read Stock Data from Yahoo Finance
start = dt.datetime(2016, 1, 1)
end = dt.datetime(2017, 12, 31)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

```
# !pip install pandas_datareader
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(2016, 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'])
```

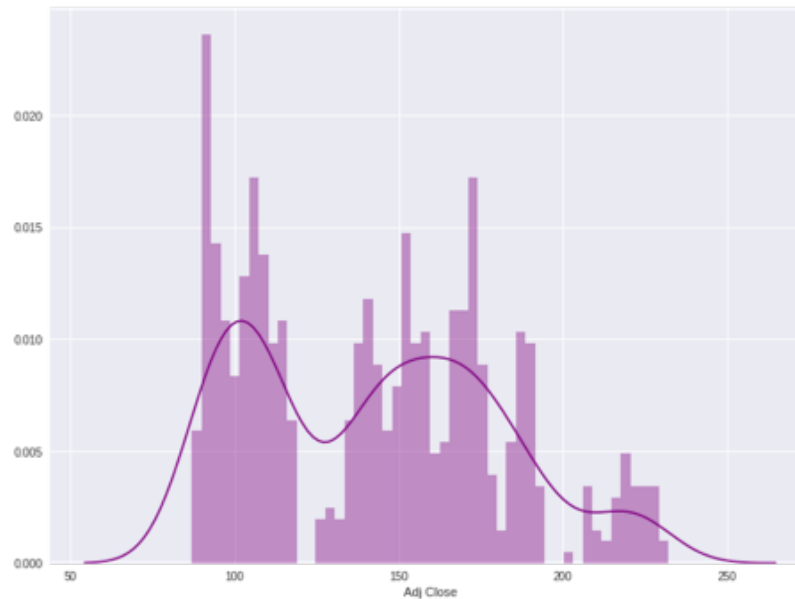


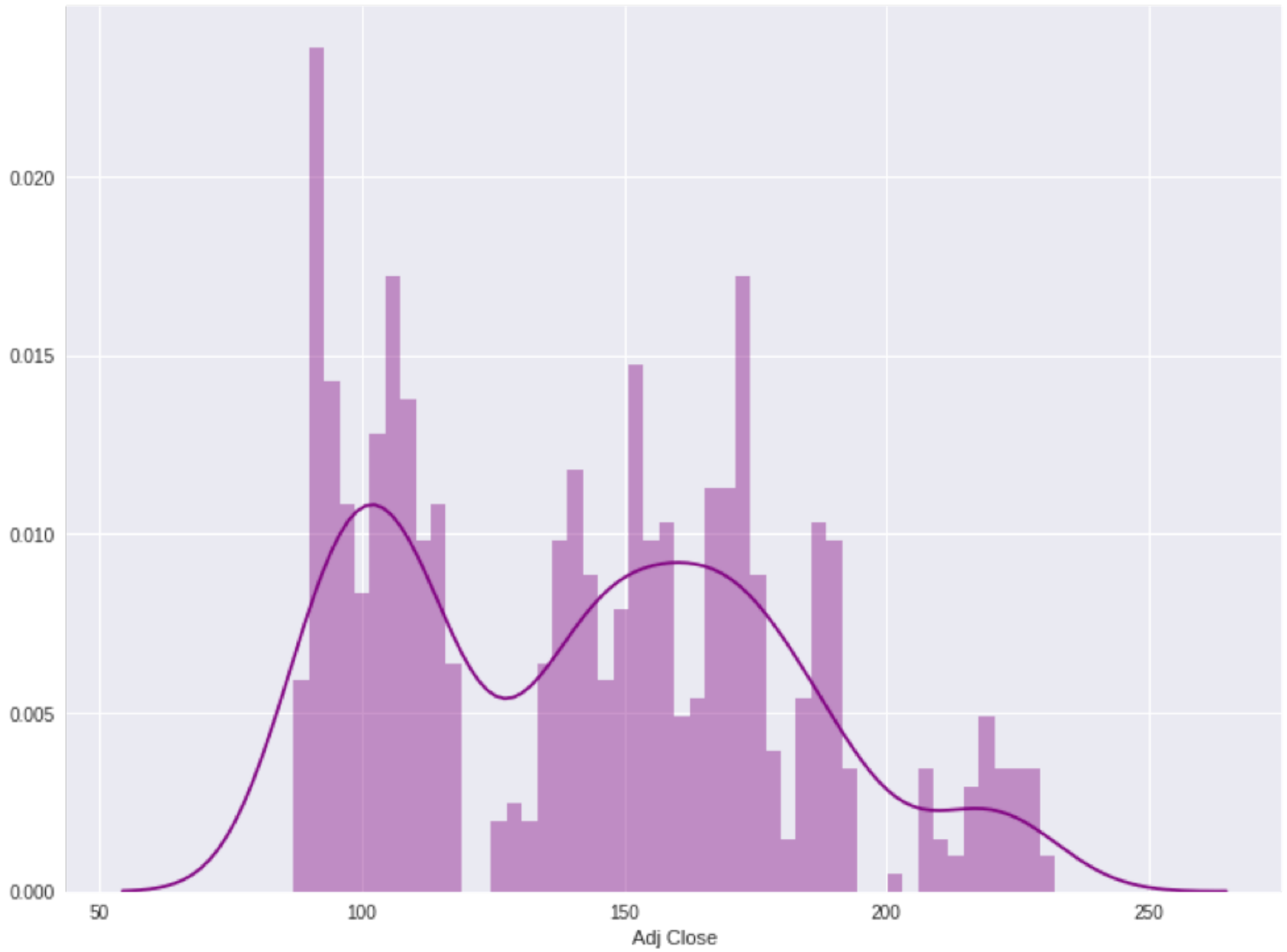
AAPL



```
# 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()
```

AAPL



```

# !pip install pandas_datareader
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(2016, 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()

```

```

1 # !pip install pandas_datareader
2 import pandas as pd
3 import pandas_datareader.data as web
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 import datetime as dt
7 %matplotlib inline
8
9 #Read Stock Data from Yahoo Finance
10 end = dt.datetime.now()
11 #start = dt.datetime(end.year-2, end.month, end.day)
12 start = dt.datetime(2016, 1, 1)
13 df = web.DataReader("AAPL", 'yahoo', start, end)
14 df.to_csv('AAPL.csv')
15 df.from_csv('AAPL.csv')
16 df.tail()
17
18 df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
19 plt.figure(figsize=(12,9))
20 top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
21 bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
22 top.plot(df.index, df['Adj Close'], color='blue') #df.index gives the dates
23 bottom.bar(df.index, df['Volume'])
24
25 # set the labels
26 top.axes.get_xaxis().set_visible(False)
27 top.set_title('AAPL')
28 top.set_ylabel('Adj Close')
29 bottom.set_ylabel('Volume')
30
31 plt.figure(figsize=(12,9))
32 sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')
33
34 # simple moving averages
35 df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
36 df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
37 df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
38 df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60']})
39 df2.plot(figsize=(12, 9), legend=True, title='AAPL')
40 df2.to_csv('AAPL_MA.csv')
41 fig = plt.gcf()
42 fig.set_size_inches(12, 9)
43 fig.savefig('AAPL_plot.png', dpi=300)
44 plt.show()

```


Finance Data from Quandl

```
# ! pip install quandl
import quandl
# quandl.ApiConfig.api_key = "YOURAPIKEY"
df = quandl.get("WIKI/AAPL", start_date="2016-01-01", end_date="2017-12-31")
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

```
1 # ! pip install quandl
2 import quandl
3 # quandl.ApiConfig.api_key = "YOURAPIKEY"
4 df = quandl.get("WIKI/AAPL", start_date="2016-01-01", end_date="2017-12-31")
5 df.to_csv('AAPL.csv')
6 df.from_csv('AAPL.csv')
7 df.tail()
```

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: FutureWarning: from_csv is deprecated. Please use read_csv(...) instead
"""

	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
Date												
2017-12-22	174.68	175.424	174.500	175.01	16052615.0	0.0	1.0	174.68	175.424	174.500	175.01	16052615.0
2017-12-26	170.80	171.470	169.679	170.57	32968167.0	0.0	1.0	170.80	171.470	169.679	170.57	32968167.0
2017-12-27	170.10	170.780	169.710	170.60	21672062.0	0.0	1.0	170.10	170.780	169.710	170.60	21672062.0
2017-12-28	171.00	171.850	170.480	171.08	15997739.0	0.0	1.0	171.00	171.850	170.480	171.08	15997739.0
2017-12-29	170.52	170.590	169.220	169.23	25643711.0	0.0	1.0	170.52	170.590	169.220	169.23	25643711.0

Yahoo Finance Symbols: AAPL Apple Inc. (AAPL)

Home Mail Flickr Tumblr News Sports Finance Celebrity Answers Groups Mobile More ▾

YAHOO!
FINANCE

Search for news, symbols or companies

Search

Finance Home Originals Events Personal Finance Technology Markets Industries **NEW** My Screeners My Portfolio

S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

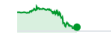
-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,245.40

-1.10 (-0.09%)



Quote Lookup

Search for symbols or companies: YHOO, GOOG, DIS



Symbols similar to 'aapl'

All Markets ▾

All (9)

Stocks (6)

Mutual Funds (0)

ETFs (1)

Indices (2)

Futures (0)

Currencies (0)

Symbol	Company Name	Last Price	Industry / Category	Type	Exchange
AAPL	Apple Inc.	139.84	Electronic Equipment	Stocks	NMS
AAPL.SW	Apple Inc.	140.70	N/A	Stocks	EBS
AAPL.MX	Apple Inc.	2678.68	Electronic Equipment	Stocks	MEX
AAPL34F.SA	Apple Inc.	0.00	N/A	Stocks	SAO
AAPL34.SA	Apple Inc.	43.14	Electronic Equipment	Stocks	SAO

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

Apple Inc. (AAPL) - NasdaqGS



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%) **139.35** -0.49 (-0.35%)

At close: 4:00PM EDT

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](#)



Yahoo Finance Charts: Apple Inc. (AAPL)

YAHOO! FINANCE

Go to Quote Summary Page



S&P 500

2,344.02
-29.45 (-1.24%)



Dow 30

20,668.01
-237.85 (-1.14%)



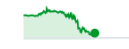
Nasdaq

5,793.83
-107.70 (-1.83%)



Crude Oil

47.50
+0.16 (+0.34%)



Gold

1,245.50
-1.00 (-0.08%)



Apple Inc. (AAPL) 139.84 -1.62 (-1.15%) As of 4:00PM EDT. Market closed.



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

Apple Inc. (AAPL) Historical Data

Home Mail Flickr Tumblr News Sports Finance Celebrity Answers Groups Mobile More



Search for news, symbols or companies

Search

Finance Home Originals Events Personal Finance Technology Markets Industries **NEW** My Screeners My Portfolio

S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,245.60

-0.90 (-0.07%)



Apple Inc. (AAPL)

NasdaqGS - NasdaqGS Delayed Price. Currency in USD

Add to watchlist

139.84 -1.62 (-1.15%) **139.35** -0.49 (-0.35%)

At close: 4:00PM EDT

After hours: 7:59PM EDT

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

Thank you for helping us improve your Yahoo experience

Learn more about your feedback.

Time Period: Mar 22, 2016 - Mar 22, 2017

Show: Historical Prices

Frequency: Daily

Apply

Currency in USD

Download Data

Date	Open	High	Low	Close	Adj Close*	Volume
Mar 21, 2017	142.11	142.80	139.73	139.84	139.84	39,116,800

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

Yahoo Finance Historical Prices

Apple Inc. (AAPL)

Time Period: [Mar 22, 2016 - Mar 22, 2017](#) ▾Show: [Historical Prices](#) ▾Frequency: [Daily](#) ▾

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
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)



Search for news, symbols or companies

Search

Time Period: Dec 12, 1980 - Mar 22, 2017

Show: Historical Prices

Frequency: Daily

Apply

Download Data

1D 5D 3M 6M
 YTD 1Y 5Y MAX
 Start Date: 12/12/1980
 End Date: 3/22/2017
 Done Cancel

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

Yahoo Finance Historical Prices

Apple Inc. (AAPL)



Search for news, symbols or companies

Search

Time Period: Dec 12, 1980 - Mar 22, 2017

Show: Historical Prices

Frequency: Daily

Apply

Download Data

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

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)

YAHOO! FINANCE

Go to Quote Summary Page



S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

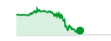
-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,245.60

-0.90 (-0.07%)



Alphabet Inc. (GOOG) 830.46 -17.94 (-2.11%) As of 4:00PM EDT. Market closed.



Dow Jones Industrial Average (^DJI)

YAHOO! FINANCE

Go to Quote Summary Page



S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

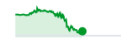
-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,244.90

-1.60 (-0.13%)



Dow Jones Industrial Average (^DJI) 20,668.01 -237.85 (-1.14%) As of 4:36PM EDT. Market closed.



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

TSEC weighted index (^TWII) - Taiwan

YAHOO! FINANCE

Go to Quote Summary Page



S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

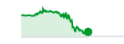
-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,245.10

-1.40 (-0.11%)



TSEC weighted index (^TWII) 9,868.95 -103.54 (-1.04%) As of 10:38AM CST. Taiwan Delayed Price. Market open.



<http://finance.yahoo.com/chart/^TWII>

Taiwan Semiconductor Manufacturing Company Limited (2330.TW)

Home Mail Flickr Tumblr News Sports Finance Celebrity Answers Groups Mobile More



Search for news, symbols or companies

Search

Finance Home Originals Events Personal Finance Technology Markets Industries **NEW** My Screeners My Portfolio

S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,244.90

-1.60 (-0.13%)



Taiwan Semiconductor Manufacturing Company Limited (2330.TW)

Taiwan - Taiwan Delayed Price. Currency in TWD

Add to watchlist

192.50 -2.50 (-1.28%)

As of 9:52AM CST. Market open.

Summary

Conversations

Statistics

Profile

Financials

Options

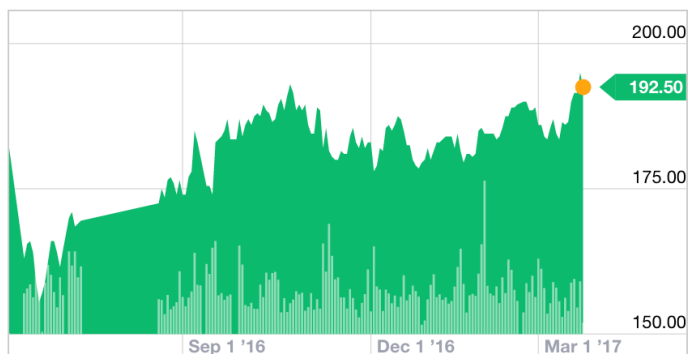
Holders

Historical Data

Analysts

Previous Close	195.00	Market Cap	4.98T
Open	192.50	Beta	N/A
Bid	192.00 x	PE Ratio (TTM)	14.90
Ask	192.50 x	EPS (TTM)	12.89
Day's Range	191.50 - 193.00	Earnings Date	Apr 13, 2017
52 Week Range	154.00 - 193.00	Dividend & Yield	N/A (N/A)
Volume	6,977,000	Ex-Dividend Date	N/A

1D 5D 1M 6M YTD **1Y** 2Y 5Y 10Y MAX [Interactive chart](#)



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

Yahoo Finance Charts

TSMC (2330.TW)

YAHOO! FINANCE

Go to Quote Summary Page



S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

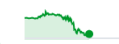
-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,245.00

-1.50 (-0.12%)



Taiwan Semiconductor Manufacturing Company Limited (2330.TW) 192.00 -3.00 (-1.54%)

As of 10:29AM CST. Taiwan Delayed Price. Market open.



<http://finance.yahoo.com/chart/2330.TW>

Yahoo Finance Charts

US Dollar/USDX - Index - Cash (DX-Y.NYB)

yahoo! finance

US Dollar/USDX - Index - Cash (DX-Y.NYB)

97.35 +0.11 (+0.12%)
As of 9:52AM EST. Market open.

[Summary](#) **[Chart](#)** [Conversations](#) [Historical Data](#) [Options](#) [Components](#)



<https://finance.yahoo.com/quote/DX-Y.NYB/chart?p=DX-Y.NYB>

Yahoo Finance Charts

USD/TWD (USDTWD=X)

yahoo! finance

Search for news, symbols or companies

Sign in Mail

Finance Home Watchlists My Portfolio Screeners Premium Markets Industries Videos News Personal Finance Tech

30.4100 -0.0200 (-0.0657%)

As of 3:04PM GMT. Market open.

Summary **Chart** Conversations

Indicators Comparison Date Range 1D 5D 1M 3M 6M YTD 1Y 2Y **5Y** Max Interval 1W Candle Draw Full screen

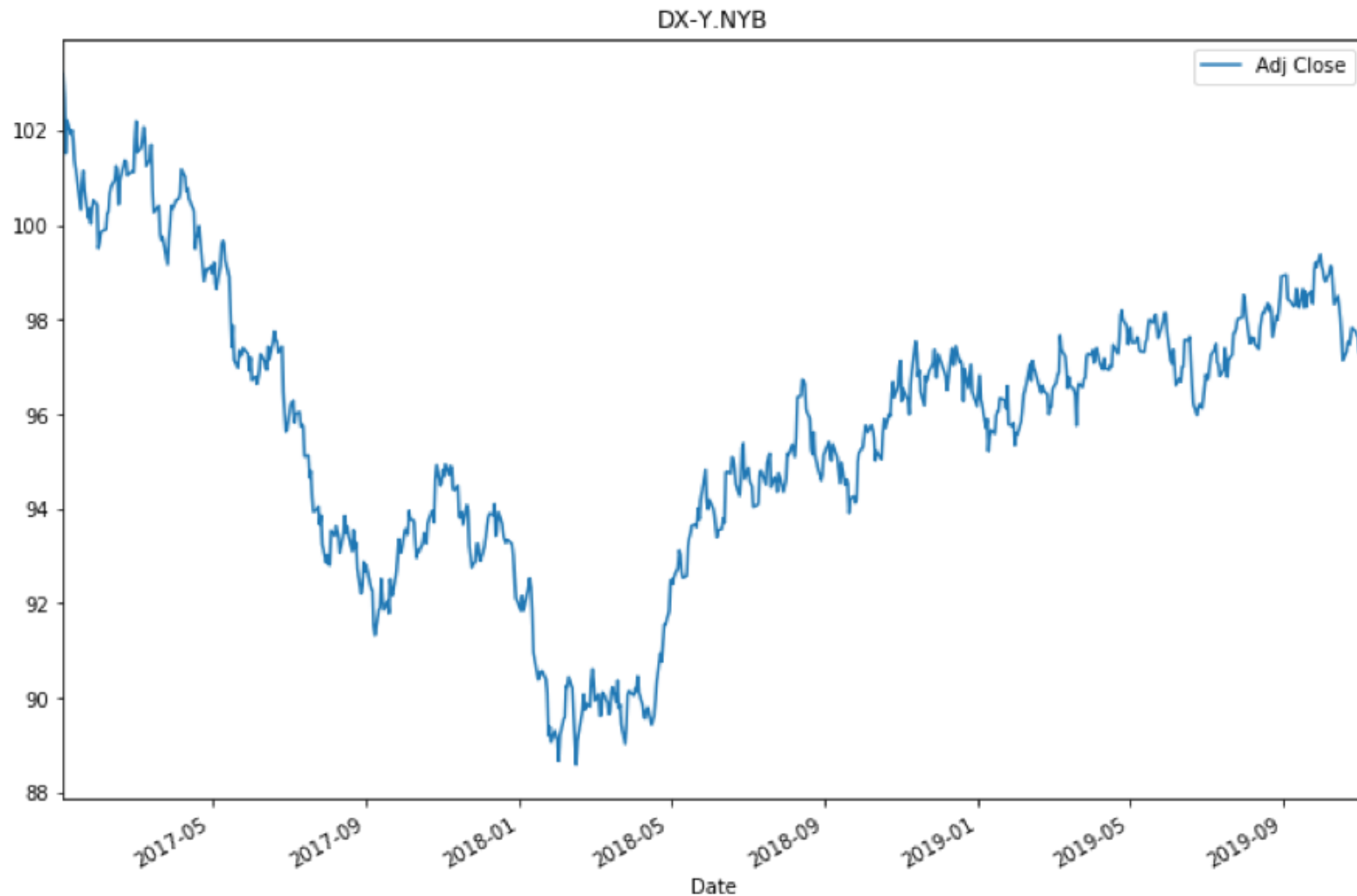


<https://finance.yahoo.com/quote/USDTWD%3DX/chart?p=USDTWD%3DX>

US Dollar/USDX - Index - Cash (DX-Y.NYB)

```
#!/pip install pandas_datareader
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(2017, 1, 1)
df = web.DataReader("DX-Y.NYB", 'yahoo', start, end)
df.to_csv('DX-Y.NYB.csv')
print(df.tail())
df2 = pd.read_csv('DX-Y.NYB.csv')
print(df2.tail())
df['Adj Close'].plot(legend=True, figsize=(12, 8),
title='DX-Y.NYB', label='Adj Close')
```

US Dollar/USDX - Index - Cash (DX-Y.NYB)



```

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 ×

```
1 Date,Open,High,Low,Close,Volume,Adj Close
2 2010-01-01,64.5,64.5,64.5,64.5,0,52.8308
3 2010-01-04,65.0,65.0,64.0,64.9,39407000,53.1584
4 2010-01-05,65.0,65.1,63.9,64.5,37138000,52.8308
5 2010-01-06,64.5,64.9,63.7,64.9,49261000,53.1584
6 2010-01-07,64.9,65.0,64.2,64.2,42134000,52.5851
7 2010-01-08,63.5,64.3,63.5,64.0,46076000,52.4213
8 2010-01-11,64.0,64.9,63.5,64.5,36799000,52.8308
9 2010-01-12,64.4,64.4,63.3,63.6,49853000,52.0936
10 2010-01-13,63.0,63.1,62.6,62.8,47976000,51.4384
11 2010-01-14,63.6,63.6,63.0,63.2,36149000,51.766
12 2010-01-15,62.9,63.5,62.8,63.5,47852000,52.0117
13 2010-01-18,62.8,63.1,62.8,62.9,30136000,51.5203
14 2010-01-19,63.0,63.2,62.0,62.5,47202000,51.1926
15 2010-01-20,62.9,63.2,62.2,63.0,52281000,51.6022
```


df.loc[start:end]

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

import yfinance as yf

```
#yfinance
!pip install yfinance
!pip install pandas_datareader
from pandas_datareader import data as pdr
import yfinance as yf
yf.pdr_override()
df = pdr.get_data_yahoo("AAPL", start="2018-01-01", end="2020-10-20")
print(df.tail())
print(df.head())
```

	Open	High	Low	Close	Adj Close	Volume
Date						
2020-10-13	125.269997	125.389999	119.650002	121.099998	121.099998	262330500
2020-10-14	121.000000	123.029999	119.620003	121.190002	121.190002	151062300
2020-10-15	118.720001	121.199997	118.150002	120.709999	120.709999	112559200
2020-10-16	121.279999	121.550003	118.809998	119.019997	119.019997	115393800
2020-10-19	119.959999	120.419998	115.660004	115.980003	115.980003	120221600

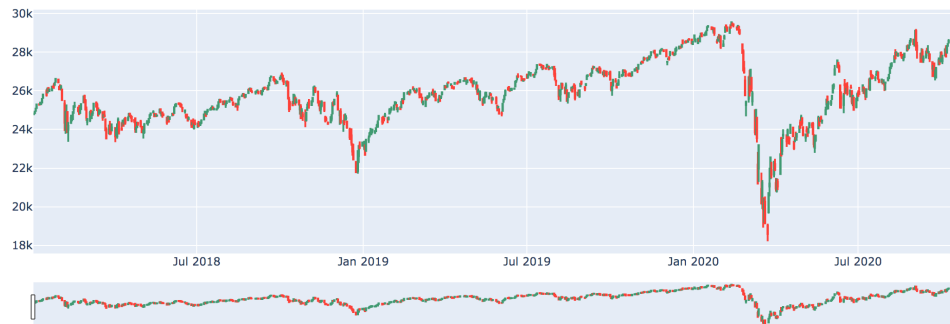
Plotly

```
!pip install pandas_datareader
!pip install plotly
import plotly.graph_objects as go
import pandas as pd
import pandas_datareader.data as web
from datetime import datetime
import datetime as dt

#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2018, 1, 1)
end = dt.datetime.now()
df = web.DataReader("^DJI", 'yahoo', start, end) #^TWII #2330.TW #^DJI #AAPL
df.to_csv('DJI.csv')

df = pd.read_csv('DJI.csv')
fig = go.Figure(data=[go.Candlestick(x=df['Date'],
                                     open=df['Open'],
                                     high=df['High'],
                                     low=df['Low'],
                                     close=df['Close'])])

fig.show()
```



<https://tinyurl.com/aintpupython101>

candlestick_ohlc

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

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



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

TA-Lib websites, products and trademarks are owned by TicTacTec LLC.

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!](#)

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
```

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
```

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
```
















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

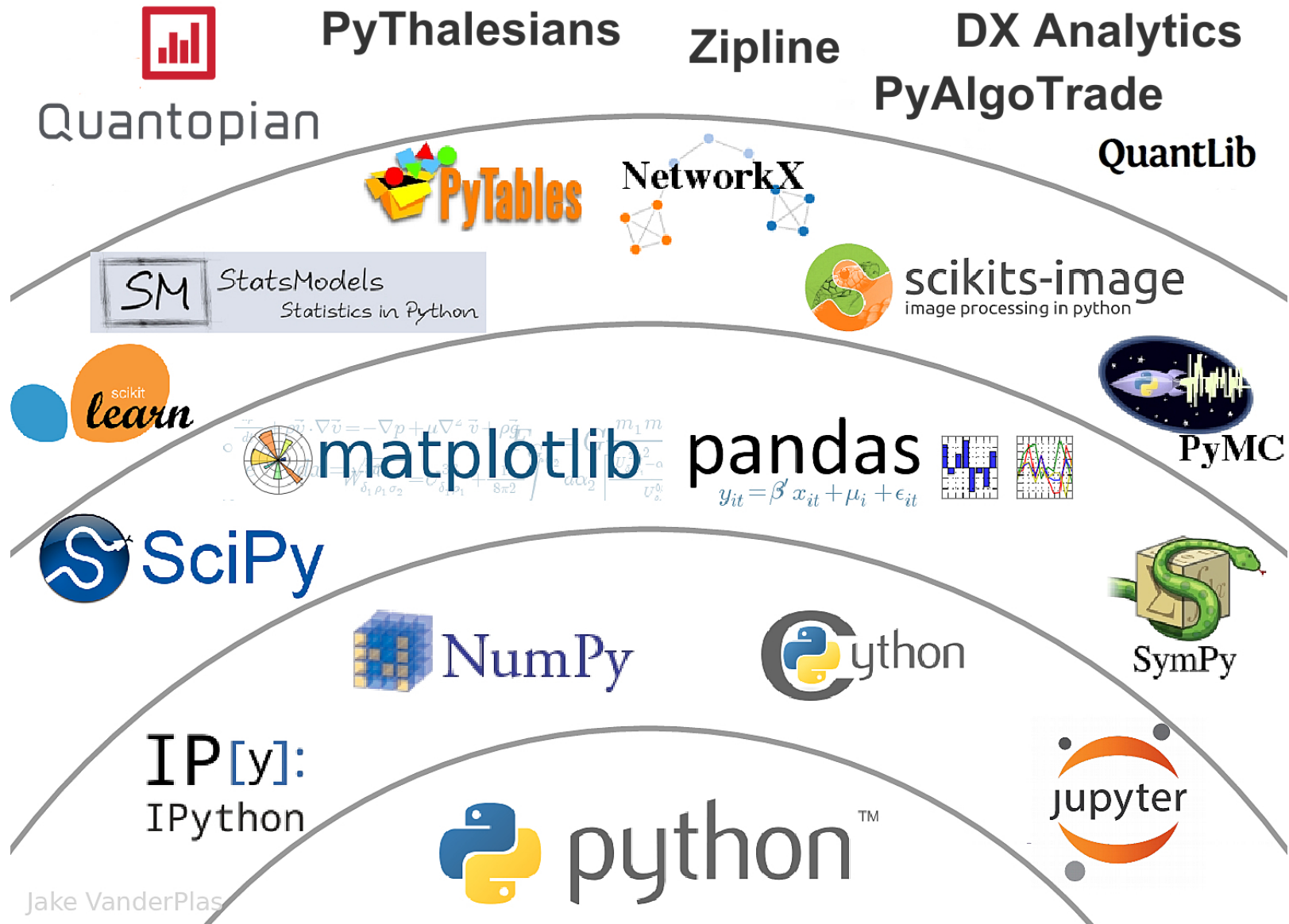
Branch: 2nd-edition ▾ New pull request Find file Clone or download ▾

 **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>

The Quant Finance PyData Stack



Jake VanderPlas

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

Quantopian



Investor Relations

Allocations

Research

Community

Learn

Help

Log In

Sign Up

Leveling Wall Street's Playing Field

Quantopian inspires talented people everywhere to write investment algorithms.
Select authors may license their algorithms to us and get paid based on performance.

Start Coding



<https://www.quantopian.com/>

Summary

- **Quantitative Big Data Analysis with Pandas in Python**
 - **Pandas**
 - Data structures and data analysis tools

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>
- Yves Hilpisch (2018), "Python for Finance: Mastering Data-Driven Finance", 2nd Edition, O'Reilly Media.
<https://github.com/yhilpisch/py4fi2nd>
- Ties de Kok (2017), Learn Python for Research,
<https://github.com/TiesdeKok/LearnPythonforResearch>
- Avinash Jain (2017), Introduction To Python Programming, Udemy,
<https://www.udemy.com/pythonforbeginnersintro/>
- 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/>
- Skikit-learn, <http://scikit-learn.org/>
- Data School (2015), Machine learning in Python with scikit-learn,
<https://www.youtube.com/playlist?list=PL5-da3qGB5ICeMbQuqbbCOQWcS6OYBr5A>
- Jason Brownlee (2016), Your First Machine Learning Project in Python Step-By-Step,
<https://machinelearningmastery.com/machine-learning-in-python-step-by-step/>
- Min-Yuh Day (2020), Python 101, <https://tinyurl.com/aintpupython101>