AI in Financial Application

Python Pandas 量化投資分析
(Quantitative Investing with Pandas in Python)

Min-Yuh Day
戴敏育
Associate Professor
副教授

Dept. of Information Management, Tamkang University

http://mail.tku.edu.tw/myday/
2019-11-08
<table>
<thead>
<tr>
<th>週次 (Week)</th>
<th>日期 (Date)</th>
<th>內容 (Subject/Topics)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2019/09/13</td>
<td>中秋節 (Mid-Autumn Festival) 放假一天 (Day off)</td>
</tr>
<tr>
<td>2</td>
<td>2019/09/20</td>
<td>人工智能財務金融應用課程介紹 (Course Orientation for AI in Financial Application)</td>
</tr>
<tr>
<td>3</td>
<td>2019/09/27</td>
<td>人工智慧投資分析與機器人理財顧問 (Artificial Intelligence for Investment Analysis and Robo-Advisors)</td>
</tr>
<tr>
<td>4</td>
<td>2019/10/04</td>
<td>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)</td>
</tr>
<tr>
<td>5</td>
<td>2019/10/11</td>
<td>國慶日補假 (Bridge Holiday for National Day, Extra Day Off)</td>
</tr>
<tr>
<td>6</td>
<td>2019/10/18</td>
<td>財務金融事件研究法 (Event Studies in Finance)</td>
</tr>
<tr>
<td>週次 (Week)</td>
<td>日期 (Date)</td>
<td>內容 (Subject/Topics)</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>7  2019/10/25</td>
<td>人工智慧財務金融應用個案研究 I</td>
<td>(Case Study on AI in Financial Application I)</td>
</tr>
<tr>
<td>8  2019/11/01</td>
<td>Python AI智慧金融分析基礎</td>
<td>(Foundations of AI in Finance Big Data Analytics with Python)</td>
</tr>
<tr>
<td>9  2019/11/08</td>
<td>Python Pandas 量化投資分析</td>
<td>(Quantitative Investing with Pandas in Python)</td>
</tr>
<tr>
<td>10  2019/11/15</td>
<td>期中報告 (Midterm Project Report)</td>
<td></td>
</tr>
<tr>
<td>11  2019/11/22</td>
<td>Python Scikit-Learn 機器學習財務金融應用</td>
<td>(Machine Learning in Finance Application with Scikit-Learn In Python)</td>
</tr>
<tr>
<td>週次 (Week)</td>
<td>日期 (Date)</td>
<td>內容 (Subject/Topics)</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>----------------------</td>
</tr>
</tbody>
</table>
| 13         | 2019/12/06 | 人工智慧財務金融應用個案研究 II  
(Case Study on AI in Financial Application II) |
| 14         | 2019/12/13 | TensorFlow 深度學習財務金融應用 II  
(Deep Learning for Finance Application with TensorFlow II) |
| 15         | 2019/12/20 | TensorFlow 深度學習財務金融應用 III  
(Deep Learning for Finance Application with TensorFlow III) |
| 16         | 2019/12/27 | 社會網絡分析財務金融應用  
(Social Network Analysis for Finance Application) |
| 17         | 2020/01/03 | 期末報告 I (Final Project Presentation I) |
| 18         | 2020/01/10 | 期末報告 II (Final Project Presentation II) |
Quantitative Investing with Pandas in Python
The Quant Finance PyData Stack

Quantopian

PyThalesians

Zipline

DX Analytics

PyAlgoTrade

QuantLib

StatsModels: Statistics in Python

scikit-learn: Machine Learning in Python

matplotlib: A toolkit for making publication-quality graphics in Python

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

SciPy: Open-source scientific computing library for Python

NumPy: The fundamental package for scientific computing with Python

SymPy: A Python library for symbolic mathematics

IPython: An enhanced interactive Python CLI and notebook

Jupyter: A web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text

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

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

```python
# !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(2016, 1, 1)
df = web.DataReader('AAPL', 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.head()

plt.figure(figsize=(12, 8), title='AAPL', label='Adj Close')

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
plt.xticks_rotation(45)
plt.subplots_adjust(right=0.8)

top.set_xlabel('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
MA5 = df['Adj Close'].rolling(5).mean() # 5 days
MA20 = df['Adj Close'].rolling(20).mean() # 20 days
MA60 = df['Adj Close'].rolling(60).mean() # 60 days

df = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA5': df['MA5'], 'MA20': df['MA20'], 'MA60': df['MA60']})
df.plot(figsize=(12, 9), legend=True, title='AAPL')
df.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
```
Wes McKinney (2017), Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, O'Reilly

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

Source: https://www.amazon.com/Python-Data-Analysis-Wrangling-IPython-dp-1491957662/dp/1491957662

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

<table>
<thead>
<tr>
<th>branch</th>
<th>description</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd-edition</td>
<td>Add Kaggle titanic dataset</td>
<td>5 months ago</td>
</tr>
<tr>
<td>examples</td>
<td>Remove sex column from tips dataset</td>
<td>4 months ago</td>
</tr>
<tr>
<td>.gitignore</td>
<td>Add gitignore</td>
<td>2 years ago</td>
</tr>
<tr>
<td>COPYING</td>
<td>Use MIT license for code examples</td>
<td>a month ago</td>
</tr>
<tr>
<td>README.md</td>
<td>Add launch in Azure Notebooks button (#70)</td>
<td>19 days ago</td>
</tr>
<tr>
<td>appa.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch02.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch03.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch04.ipynb</td>
<td>Convert all notebooks to v4 format</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch05.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch06.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch07.ipynb</td>
<td>Convert all notebooks to v4 format</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch08.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch09.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch10.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
</tbody>
</table>

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

NumPy Basics: Arrays and

```python
In [ ]:
import numpy as np
np.random.seed(12345)
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 6))
np.set_printoptions(precision=4, suppress=True)
```

```python
In [ ]:
import numpy as np
my_arr = np.arange(1000000)
my_list = list(range(1000000))
```

```python
In [ ]:
%time for _ in range(10): my_arr2 = my_arr * 2
%time for _ in range(10): my_list2 = [x * 2 for x in my_list]
```

The NumPy ndarray: A Multidimensional Array Object

```python
In [ ]:
import numpy as np
# Generate some random data
data = np.random.randn(2, 3)
data
```
Getting Started with pandas

In [ ]:
```
import pandas as pd
```

In [ ]:
```
from pandas import Series, DataFrame
```

In [ ]:
```
import numpy as np
np.random.seed(12345)
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 6))
PREVIOUS_MAX_ROWS = pd.options.display.max_rows
pd.options.display.max_rows = 20
np.set_printoptions(precision=4, suppress=True)
```

Introduction to pandas Data Structures

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
**Yves Hilpisch (2018), “Python for Finance: Mastering Data-Driven Finance”, O'Reilly**

- **Source:** [https://github.com/yhilpisch/py4fi2nd/tree/master/code](https://github.com/yhilpisch/py4fi2nd/tree/master/code)

### py4fi2nd / code /

<table>
<thead>
<tr>
<th>Folder</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_dates</td>
<td>Initial commit.</td>
<td>11 months ago</td>
</tr>
<tr>
<td>b_bsm</td>
<td>Initial commit.</td>
<td>11 months ago</td>
</tr>
<tr>
<td>ch01</td>
<td>Initial commit.</td>
<td>11 months ago</td>
</tr>
<tr>
<td>ch02</td>
<td>Initial commit.</td>
<td>11 months ago</td>
</tr>
<tr>
<td>ch03</td>
<td>Initial commit.</td>
<td>11 months ago</td>
</tr>
<tr>
<td>ch04</td>
<td>Initial commit.</td>
<td>11 months ago</td>
</tr>
<tr>
<td>ch05</td>
<td>Initial commit.</td>
<td>11 months ago</td>
</tr>
<tr>
<td>ch06</td>
<td>Initial commit.</td>
<td>11 months ago</td>
</tr>
<tr>
<td>ch07</td>
<td>Initial commit.</td>
<td>11 months ago</td>
</tr>
<tr>
<td>ch08</td>
<td>Initial commit.</td>
<td>11 months ago</td>
</tr>
<tr>
<td>ch09</td>
<td>Initial commit.</td>
<td>11 months ago</td>
</tr>
<tr>
<td>ch10</td>
<td>Initial commit.</td>
<td>11 months ago</td>
</tr>
<tr>
<td>ch11</td>
<td>Initial commit.</td>
<td>11 months ago</td>
</tr>
</tbody>
</table>

**Latest commit:** 3de58ff on Dec 4, 2018

**Source:** [https://github.com/yhilpisch/py4fi2nd/tree/master/code](https://github.com/yhilpisch/py4fi2nd/tree/master/code)
Python for Finance (2nd ed.)

Mastering Data-Driven Finance

© Dr. Yves J. Hilpisch | The Python Quants GmbH

Source: https://github.com/yhilpisch/py4fi2nd/blob/master/code/ch08/08_financial_time_series.ipynb
Python Pandas
Python Data Analysis Library

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

*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/
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

# pandas-datareader

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

⚠️ **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.

ℹ️ **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`:

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

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

[Read the Docs](https://pandas-datareader.readthedocs.io/en/latest/)
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

PyDatastream 0.5.1

pip install PyDatastream

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

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: Statistical Data and Metadata eXchange in Python

pandasSDMX 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. pandasSDMX 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

pandasSDMX 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 Estadistica y Geografia - 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)

https://pandasdmx.readthedocs.io/en/latest/
Fred API

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

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 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.
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

<table>
<thead>
<tr>
<th>pandas</th>
<th>SAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataFrame</td>
<td>data set</td>
</tr>
<tr>
<td>column</td>
<td>variable</td>
</tr>
<tr>
<td>row</td>
<td>observation</td>
</tr>
<tr>
<td>groupby</td>
<td>BY-group</td>
</tr>
<tr>
<td>NaN</td>
<td>.</td>
</tr>
</tbody>
</table>

Creating `pd.DataFrame`

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
</tr>
</tbody>
</table>

```python
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

```python
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

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

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

dates = pd.date_range('20181001', periods=6)
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')
```
```python
df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))
df
```

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-10-01</td>
<td>-0.336188</td>
<td>0.584621</td>
<td>-1.061433</td>
<td>-0.036278</td>
</tr>
<tr>
<td>2018-10-02</td>
<td>0.903683</td>
<td>-0.839723</td>
<td>-0.270219</td>
<td>-1.099606</td>
</tr>
<tr>
<td>2018-10-03</td>
<td>0.920208</td>
<td>-0.240353</td>
<td>-0.818598</td>
<td>-1.105489</td>
</tr>
<tr>
<td>2018-10-04</td>
<td>0.221045</td>
<td>-0.314589</td>
<td>0.042071</td>
<td>-1.447280</td>
</tr>
<tr>
<td>2018-10-05</td>
<td>0.946862</td>
<td>-1.570305</td>
<td>-1.009180</td>
<td>-0.375659</td>
</tr>
<tr>
<td>2018-10-06</td>
<td>-0.225148</td>
<td>0.510691</td>
<td>2.002372</td>
<td>-0.335005</td>
</tr>
</tbody>
</table>
```python
df = pd.DataFrame(np.random.randn(3,5),
                 index=[student1, student2, student3],
                 columns=list('ABCDE'))
df
```

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>student1</td>
<td>-0.346884</td>
<td>-1.232934</td>
<td>-0.302072</td>
<td>-1.345084</td>
<td>-0.723880</td>
</tr>
<tr>
<td>student2</td>
<td>1.090955</td>
<td>-0.010483</td>
<td>1.280072</td>
<td>-0.253958</td>
<td>-0.030604</td>
</tr>
<tr>
<td>student3</td>
<td>0.325660</td>
<td>0.808956</td>
<td>-0.395820</td>
<td>-1.498926</td>
<td>1.603471</td>
</tr>
</tbody>
</table>
df2 = pd.DataFrame({ 'A' : 1.0, '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

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0</td>
<td>2018-10-01</td>
<td>2.5</td>
<td>3</td>
<td>test</td>
<td>foo</td>
</tr>
<tr>
<td>1</td>
<td>1.0</td>
<td>2018-10-01</td>
<td>2.5</td>
<td>3</td>
<td>train</td>
<td>foo</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>2018-10-01</td>
<td>2.5</td>
<td>3</td>
<td>test</td>
<td>foo</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>2018-10-01</td>
<td>2.5</td>
<td>3</td>
<td>train</td>
<td>foo</td>
</tr>
</tbody>
</table>
df2.dtypes

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>float64</td>
</tr>
<tr>
<td>B</td>
<td>datetime64[ns]</td>
</tr>
<tr>
<td>C</td>
<td>float32</td>
</tr>
<tr>
<td>D</td>
<td>int32</td>
</tr>
<tr>
<td>E</td>
<td>category</td>
</tr>
<tr>
<td>F</td>
<td>object</td>
</tr>
</tbody>
</table>

dtype: object
Python Pandas for Finance

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

Collecting pandas_datareader
  Downloading https://files.pythonhosted.org/packages/cc/5c/ea5b6dcf0df55c5fb8e37fb45335ec01ccee1a99b8a79339137f5ed269e0/pandas_datareader-0.7.0.tar.gz (112kB)
  100% |████████████████████████████████| 112kB 2.7MB/s 
Collecting lxml (from pandas_datareader)
  Downloading https://files.pythonhosted.org/packages/03/a4/9ee8035fc7c7670e5eab97f34ff2e0dd78a491bf96df5accedb0e63f5/lxml-4.2.5-cp37-cp37m-macosx_10_9_x86_64.whl (5.8MB)
  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) (2.8.1)
Requirement already satisfied: numpy>=1.19.0 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) (2.6.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (2017.11.5)
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)
Installing collected packages: lxml, pandas-datareader
Successfully installed lxml-4.2.5 pandas-datareader-0.7.0
conda install pandas-datareader

[Imyday-MacBook-Pro:~ imyday$ conda install pandas-datareader
Fetching package metadata .......... 
Solving package specifications: .

Package plan for installation in environment /Users/imyday/anaconda:

The following NEW packages will be INSTALLED:

    pandas-datareader: 0.2.1-py36_0
    requests-file: 1.4.1-py36_0

Proceed ([y]/[n])? y

requests-file 100% |#################################| Time: 0:00:00 1.55 MB/s
pandas-datareader 100% |#################################| Time: 0:00:00 409.66 kB/s
[Imyday-MacBook-Pro:~ imyday$ conda list
# packages in environment at /Users/imyday/anaconda:
#
  _license  1.1          py36_1
  alabaster 0.7.9        py36_0
  anaconda  4.3.1        np111py36_0
  anaconda-client 1.6.0   py36_0
  anaconda-navigator 1.5.0 py36_0
  anaconda-project 0.4.1  py36_0
# !pip install pandas_datareader
import pandas_datareader.data as web
import datetime as dt
#Read Stock Data from Yahoo Finance
end = dt.datetime(2017, 12, 31)
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()
# !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()
#!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()
# !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(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
da05 = df['Adj Close'].rolling(5).mean() #5 days
da20 = df['Adj Close'].rolling(20).mean() #20 days
ma60 = df['Adj Close'].rolling(60).mean() #60 days

da05.plot(figsize=(12, 9), legend=True, title='AAPL')
da05.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
Finance Data from Quandl

```python
# ! 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()
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-12-22</td>
<td>174.68</td>
<td>175.424</td>
<td>174.500</td>
<td>175.01</td>
<td>16052615.0</td>
<td>0.0</td>
<td>1.0</td>
<td>174.68</td>
<td>175.424</td>
<td>174.500</td>
<td>175.01</td>
<td>16052615.0</td>
</tr>
<tr>
<td>2017-12-26</td>
<td>170.80</td>
<td>171.470</td>
<td>169.679</td>
<td>170.57</td>
<td>32968167.0</td>
<td>0.0</td>
<td>1.0</td>
<td>170.80</td>
<td>171.470</td>
<td>169.679</td>
<td>170.57</td>
<td>32968167.0</td>
</tr>
<tr>
<td>2017-12-27</td>
<td>170.10</td>
<td>170.780</td>
<td>169.710</td>
<td>170.60</td>
<td>21672062.0</td>
<td>0.0</td>
<td>1.0</td>
<td>170.10</td>
<td>170.780</td>
<td>169.710</td>
<td>170.60</td>
<td>21672062.0</td>
</tr>
<tr>
<td>2017-12-28</td>
<td>171.00</td>
<td>171.850</td>
<td>170.480</td>
<td>171.08</td>
<td>15997739.0</td>
<td>0.0</td>
<td>1.0</td>
<td>171.00</td>
<td>171.850</td>
<td>170.480</td>
<td>171.08</td>
<td>15997739.0</td>
</tr>
<tr>
<td>2017-12-29</td>
<td>170.52</td>
<td>170.590</td>
<td>169.220</td>
<td>169.23</td>
<td>25643711.0</td>
<td>0.0</td>
<td>1.0</td>
<td>170.52</td>
<td>170.590</td>
<td>169.220</td>
<td>169.23</td>
<td>25643711.0</td>
</tr>
</tbody>
</table>

Source: https://www.quandl.com/tools/python
Yahoo Finance Symbols: AAPL Apple Inc. (AAPL)

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%)

Symbols similar to 'aapl'

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Company Name</th>
<th>Last Price</th>
<th>Industry / Category</th>
<th>Type</th>
<th>Exchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAPL</td>
<td>Apple Inc.</td>
<td>139.84</td>
<td>Electronic Equipment</td>
<td>Stocks</td>
<td>NMS</td>
</tr>
<tr>
<td>AAPL,SW</td>
<td>Apple Inc.</td>
<td>140.70</td>
<td>N/A</td>
<td>Stocks</td>
<td>EBS</td>
</tr>
<tr>
<td>AAPL.MX</td>
<td>Apple Inc.</td>
<td>2678.68</td>
<td>Electronic Equipment</td>
<td>Stocks</td>
<td>MEX</td>
</tr>
<tr>
<td>AAPL34F.SA</td>
<td>Apple Inc.</td>
<td>0.00</td>
<td>N/A</td>
<td>Stocks</td>
<td>SAO</td>
</tr>
<tr>
<td>AAPL34.SA</td>
<td>Apple Inc.</td>
<td>43.14</td>
<td>Electronic Equipment</td>
<td>Stocks</td>
<td>SAO</td>
</tr>
</tbody>
</table>

http://finance.yahoo.com/q?s=AAPL
Apple Inc. (AAPL) - NasdaqGS

139.84 -1.62 (-1.15%)  139.35 -0.49 (-0.35%)
At close: 4:00PM EDT
After hours: 7:59PM EDT

<table>
<thead>
<tr>
<th>Summary</th>
<th>Conversations</th>
<th>Statistics</th>
<th>Profile</th>
<th>Financials</th>
<th>Options</th>
<th>Holders</th>
<th>Historical Data</th>
<th>Analysts</th>
</tr>
</thead>
</table>

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

Apple Inc. (AAPL) Historical Data

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

139.84 -1.62 (-1.15%) 139.35 -0.49 (-0.35%)
At close: 4:00PM EDT  After hours: 7:59PM EDT

Thank you for helping us improve your Yahoo experience
Learn more about your feedback.


Currency in USD

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Adj Close*</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mar 21, 2017</td>
<td>142.11</td>
<td>142.80</td>
<td>139.73</td>
<td>139.84</td>
<td>139.84</td>
<td>39,116,800</td>
</tr>
</tbody>
</table>

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

http://finance.yahoo.com/quote/AAPL/history
<table>
<thead>
<tr>
<th>Date</th>
<th>Start Date</th>
<th>End Date</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Adj Close</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mar 21, 2017</td>
<td>12/12/1980</td>
<td>3/22/2017</td>
<td>142.80</td>
<td>139.73</td>
<td>139.84</td>
<td>139.84</td>
<td>39,116,800</td>
</tr>
<tr>
<td>Mar 20, 2017</td>
<td>12/12/1980</td>
<td>3/22/2017</td>
<td>141.50</td>
<td>140.23</td>
<td>141.46</td>
<td>141.46</td>
<td>21,542,000</td>
</tr>
<tr>
<td>Mar 17, 2017</td>
<td>12/12/1980</td>
<td>3/22/2017</td>
<td>141.00</td>
<td>139.89</td>
<td>139.99</td>
<td>139.99</td>
<td>43,885,000</td>
</tr>
<tr>
<td>Mar 16, 2017</td>
<td>12/12/1980</td>
<td>3/22/2017</td>
<td>140.72</td>
<td>141.02</td>
<td>140.69</td>
<td>140.69</td>
<td>19,232,000</td>
</tr>
<tr>
<td>Mar 15, 2017</td>
<td>12/12/1980</td>
<td>3/22/2017</td>
<td>139.41</td>
<td>140.75</td>
<td>139.03</td>
<td>140.46</td>
<td>25,691,800</td>
</tr>
<tr>
<td>Mar 13, 2017</td>
<td>12/12/1980</td>
<td>3/22/2017</td>
<td>138.85</td>
<td>139.43</td>
<td>138.82</td>
<td>139.20</td>
<td>17,421,700</td>
</tr>
<tr>
<td>Mar 10, 2017</td>
<td>12/12/1980</td>
<td>3/22/2017</td>
<td>139.25</td>
<td>139.36</td>
<td>138.64</td>
<td>139.14</td>
<td>19,612,800</td>
</tr>
<tr>
<td>Mar 09, 2017</td>
<td>12/12/1980</td>
<td>3/22/2017</td>
<td>138.74</td>
<td>138.79</td>
<td>137.05</td>
<td>138.68</td>
<td>22,155,900</td>
</tr>
<tr>
<td>Mar 08, 2017</td>
<td>12/12/1980</td>
<td>3/22/2017</td>
<td>138.95</td>
<td>139.80</td>
<td>138.82</td>
<td>139.00</td>
<td>18,707,200</td>
</tr>
</tbody>
</table>
### Yahoo Finance Historical Prices

#### Apple Inc. (AAPL)

**Currency in USD**

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

[Read More](http://finance.yahoo.com/quote/AAPL/history?period1=345398400&period2=1490112000&interval=1d&filter=history&frequency=1d)
<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Volume</th>
<th>Adj Close</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-03-21</td>
<td>142.110001</td>
<td>142.800003</td>
<td>139.729996</td>
<td>139.839996</td>
<td>39116800</td>
<td>139.839996</td>
</tr>
<tr>
<td>2017-03-20</td>
<td>140.399994</td>
<td>141.50</td>
<td>140.229996</td>
<td>141.460007</td>
<td>20213100</td>
<td>141.460007</td>
</tr>
<tr>
<td>2017-03-17</td>
<td>141.00</td>
<td>141.00</td>
<td>139.889999</td>
<td>139.990005</td>
<td>43597400</td>
<td>139.990005</td>
</tr>
<tr>
<td>2017-03-16</td>
<td>140.720001</td>
<td>141.020004</td>
<td>140.259995</td>
<td>140.690002</td>
<td>19132500</td>
<td>140.690002</td>
</tr>
<tr>
<td>2017-03-15</td>
<td>139.410004</td>
<td>140.75</td>
<td>139.029999</td>
<td>140.460007</td>
<td>25566800</td>
<td>140.460007</td>
</tr>
<tr>
<td>2017-03-14</td>
<td>139.300003</td>
<td>139.649994</td>
<td>138.839996</td>
<td>138.990005</td>
<td>15189700</td>
<td>138.990005</td>
</tr>
<tr>
<td>2017-03-13</td>
<td>138.850006</td>
<td>139.429993</td>
<td>138.820007</td>
<td>139.199997</td>
<td>17042400</td>
<td>139.199997</td>
</tr>
<tr>
<td>2017-03-10</td>
<td>139.25</td>
<td>139.360001</td>
<td>138.639999</td>
<td>139.199999</td>
<td>19488000</td>
<td>139.199999</td>
</tr>
<tr>
<td>2017-03-09</td>
<td>138.740005</td>
<td>138.789993</td>
<td>137.050003</td>
<td>138.679993</td>
<td>22065200</td>
<td>138.679993</td>
</tr>
<tr>
<td>2017-03-08</td>
<td>138.949997</td>
<td>139.800003</td>
<td>138.820007</td>
<td>139.00</td>
<td>18681800</td>
<td>139.00</td>
</tr>
<tr>
<td>2017-03-07</td>
<td>139.059998</td>
<td>139.979996</td>
<td>138.789993</td>
<td>139.520004</td>
<td>17267500</td>
<td>139.520004</td>
</tr>
<tr>
<td>2017-03-06</td>
<td>139.369995</td>
<td>139.770004</td>
<td>138.600006</td>
<td>139.339996</td>
<td>21155300</td>
<td>139.339996</td>
</tr>
<tr>
<td>2017-03-03</td>
<td>138.779999</td>
<td>139.830002</td>
<td>138.589996</td>
<td>139.779999</td>
<td>21108100</td>
<td>139.779999</td>
</tr>
<tr>
<td>2017-03-02</td>
<td>140.00</td>
<td>140.279999</td>
<td>138.759995</td>
<td>138.960007</td>
<td>26153300</td>
<td>138.960007</td>
</tr>
<tr>
<td>2017-03-01</td>
<td>137.889999</td>
<td>140.149994</td>
<td>137.600006</td>
<td>139.789993</td>
<td>36272400</td>
<td>139.789993</td>
</tr>
<tr>
<td>2017-02-28</td>
<td>137.080002</td>
<td>137.440002</td>
<td>136.699997</td>
<td>136.990005</td>
<td>23403500</td>
<td>136.990005</td>
</tr>
<tr>
<td>2017-02-27</td>
<td>137.139999</td>
<td>137.440002</td>
<td>136.279999</td>
<td>136.929993</td>
<td>20196400</td>
<td>136.929993</td>
</tr>
<tr>
<td>2017-02-24</td>
<td>135.910004</td>
<td>136.660004</td>
<td>135.279999</td>
<td>136.660004</td>
<td>21690900</td>
<td>136.660004</td>
</tr>
<tr>
<td>2017-02-23</td>
<td>137.380005</td>
<td>137.479996</td>
<td>136.300003</td>
<td>136.529999</td>
<td>20704100</td>
<td>136.529999</td>
</tr>
<tr>
<td>2017-02-22</td>
<td>136.429993</td>
<td>137.119995</td>
<td>136.110001</td>
<td>137.110001</td>
<td>20745300</td>
<td>137.110001</td>
</tr>
</tbody>
</table>
Yahoo Finance Charts

Alphabet Inc. (GOOG)
TSEC weighted index (^TWII) - Taiwan

http://finance.yahoo.com/chart/^TWII
Taiwan Semiconductor Manufacturing Company Limited (2330.TW)

192.50 -2.50 (-1.28%)
As of 9:52AM CST. Market open.

Summary

| 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 |

http://finance.yahoo.com/q?s=2330.TW
Yahoo Finance Charts
TSMC (2330.TW)

Taiwan Semiconductor Manufacturing Company Limited (2330.TW) 192.00 -3.00 (-1.54%)
As of 10:29AM CST. Taiwan Delayed Price. Market open.

Open 192.50
Close 192.00
Low 191.50
High 193.00
Vol 9.33M
% Chg 301.44%

http://finance.yahoo.com/chart/2330.TW
Yahoo Finance Charts
US Dollar/USDX - Index - Cash (DX-Y.NYB)

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

Yahoo Finance Charts
USD/TWD (USDTWD=X)

30.4100  -0.0200 (-0.0657%)
As of 3:04PM GMT. Market open.

https://finance.yahoo.com/quote/USDTWD%3DX/chart?p=USDTWD%3DX
US Dollar/USDX - Index - Cash (DX-Y.NYB)

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

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Volume</th>
<th>Adj Close</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004-08-19</td>
<td>100.000168</td>
<td>104.060182</td>
<td>95.960165</td>
<td>100.340176</td>
<td>44871300</td>
<td>50.119968</td>
</tr>
<tr>
<td>2004-08-20</td>
<td>101.010175</td>
<td>109.080187</td>
<td>100.500174</td>
<td>108.310183</td>
<td>22942800</td>
<td>54.100990</td>
</tr>
<tr>
<td>2004-08-23</td>
<td>110.750191</td>
<td>113.480193</td>
<td>109.050183</td>
<td>109.400185</td>
<td>18342800</td>
<td>54.645447</td>
</tr>
<tr>
<td>2004-08-24</td>
<td>111.240189</td>
<td>111.600192</td>
<td>103.570177</td>
<td>104.870176</td>
<td>15319700</td>
<td>52.382705</td>
</tr>
<tr>
<td>2004-08-25</td>
<td>104.960181</td>
<td>108.000187</td>
<td>103.880180</td>
<td>106.000184</td>
<td>9232100</td>
<td>52.947145</td>
</tr>
<tr>
<td>2004-08-26</td>
<td>104.950180</td>
<td>107.950188</td>
<td>104.660179</td>
<td>107.910182</td>
<td>7128600</td>
<td>53.901190</td>
</tr>
<tr>
<td>2004-08-27</td>
<td>108.100185</td>
<td>108.620186</td>
<td>105.690180</td>
<td>106.150181</td>
<td>6241200</td>
<td>53.022069</td>
</tr>
<tr>
<td>2004-08-30</td>
<td>105.280178</td>
<td>105.490184</td>
<td>102.010172</td>
<td>102.010172</td>
<td>5221400</td>
<td>50.954132</td>
</tr>
<tr>
<td>2004-08-31</td>
<td>102.300173</td>
<td>103.710180</td>
<td>102.160177</td>
<td>102.370175</td>
<td>4941200</td>
<td>51.133953</td>
</tr>
<tr>
<td>2004-09-01</td>
<td>102.700174</td>
<td>102.970180</td>
<td>99.670169</td>
<td>100.250171</td>
<td>9181600</td>
<td>50.075011</td>
</tr>
</tbody>
</table>
df.tail(10)

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Volume</th>
<th>Adj Close</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-03-08</td>
<td>833.510010</td>
<td>838.150024</td>
<td>831.789978</td>
<td>835.369995</td>
<td>988700</td>
<td>835.369995</td>
</tr>
<tr>
<td>2017-03-09</td>
<td>836.000000</td>
<td>842.000000</td>
<td>834.210022</td>
<td>838.679993</td>
<td>1259900</td>
<td>838.679993</td>
</tr>
<tr>
<td>2017-03-10</td>
<td>843.280029</td>
<td>844.909973</td>
<td>839.500000</td>
<td>843.250000</td>
<td>1701100</td>
<td>843.250000</td>
</tr>
<tr>
<td>2017-03-13</td>
<td>844.000000</td>
<td>848.684998</td>
<td>843.250000</td>
<td>845.539978</td>
<td>1149500</td>
<td>845.539978</td>
</tr>
<tr>
<td>2017-03-14</td>
<td>843.640015</td>
<td>847.239990</td>
<td>840.799988</td>
<td>845.619995</td>
<td>779900</td>
<td>845.619995</td>
</tr>
<tr>
<td>2017-03-15</td>
<td>847.590027</td>
<td>848.630005</td>
<td>840.770020</td>
<td>847.200012</td>
<td>1379600</td>
<td>847.200012</td>
</tr>
<tr>
<td>2017-03-16</td>
<td>849.030029</td>
<td>850.849976</td>
<td>846.130005</td>
<td>848.780029</td>
<td>970400</td>
<td>848.780029</td>
</tr>
<tr>
<td>2017-03-17</td>
<td>851.609985</td>
<td>853.400024</td>
<td>847.109985</td>
<td>852.119995</td>
<td>1712300</td>
<td>852.119995</td>
</tr>
<tr>
<td>2017-03-20</td>
<td>850.010010</td>
<td>850.219971</td>
<td>845.150024</td>
<td>848.400024</td>
<td>1190300</td>
<td>848.400024</td>
</tr>
<tr>
<td>2017-03-21</td>
<td>851.400024</td>
<td>853.500000</td>
<td>829.020020</td>
<td>830.460022</td>
<td>2442900</td>
<td>830.460022</td>
</tr>
</tbody>
</table>
df.count()

<table>
<thead>
<tr>
<th>Column</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>3169</td>
</tr>
<tr>
<td>High</td>
<td>3169</td>
</tr>
<tr>
<td>Low</td>
<td>3169</td>
</tr>
<tr>
<td>Close</td>
<td>3169</td>
</tr>
<tr>
<td>Volume</td>
<td>3169</td>
</tr>
<tr>
<td>Adj Close</td>
<td>3169</td>
</tr>
</tbody>
</table>

*dtype: int64*
```python
df.ix['2015-12-31']
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>7.695000e+02</td>
</tr>
<tr>
<td>High</td>
<td>7.695000e+02</td>
</tr>
<tr>
<td>Low</td>
<td>7.583400e+02</td>
</tr>
<tr>
<td>Close</td>
<td>7.588800e+02</td>
</tr>
<tr>
<td>Volume</td>
<td>1.489600e+06</td>
</tr>
<tr>
<td>Adj Close</td>
<td>7.588800e+02</td>
</tr>
</tbody>
</table>

Name: 2015-12-31 00:00:00, dtype: float64
df.to_csv('2330.TW.Yahoo.Finance.Data.csv')

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Volume</th>
<th>Adj Close</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010-01-01</td>
<td>64.5</td>
<td>64.5</td>
<td>64.5</td>
<td>64.5</td>
<td>0.0</td>
<td>52.8308</td>
</tr>
<tr>
<td>2010-01-04</td>
<td>65.0</td>
<td>65.0</td>
<td>64.0</td>
<td>64.9</td>
<td>39407000</td>
<td>53.1584</td>
</tr>
<tr>
<td>2010-01-05</td>
<td>65.0</td>
<td>65.1</td>
<td>63.9</td>
<td>64.5</td>
<td>37138000</td>
<td>52.8308</td>
</tr>
<tr>
<td>2010-01-06</td>
<td>64.5</td>
<td>64.9</td>
<td>63.7</td>
<td>64.9</td>
<td>49261000</td>
<td>53.1584</td>
</tr>
<tr>
<td>2010-01-07</td>
<td>64.9</td>
<td>65.0</td>
<td>64.2</td>
<td>64.2</td>
<td>42134000</td>
<td>52.5851</td>
</tr>
<tr>
<td>2010-01-08</td>
<td>63.5</td>
<td>64.3</td>
<td>63.5</td>
<td>64.0</td>
<td>46076000</td>
<td>52.4213</td>
</tr>
<tr>
<td>2010-01-11</td>
<td>64.0</td>
<td>64.9</td>
<td>63.5</td>
<td>64.5</td>
<td>36799000</td>
<td>52.8308</td>
</tr>
<tr>
<td>2010-01-12</td>
<td>64.4</td>
<td>64.4</td>
<td>63.3</td>
<td>63.6</td>
<td>49853000</td>
<td>52.0936</td>
</tr>
<tr>
<td>2010-01-13</td>
<td>63.0</td>
<td>63.1</td>
<td>62.6</td>
<td>62.8</td>
<td>47976000</td>
<td>51.4384</td>
</tr>
<tr>
<td>2010-01-14</td>
<td>63.6</td>
<td>63.6</td>
<td>63.0</td>
<td>63.2</td>
<td>36149000</td>
<td>51.766</td>
</tr>
<tr>
<td>2010-01-15</td>
<td>62.9</td>
<td>63.5</td>
<td>62.8</td>
<td>63.5</td>
<td>47852000</td>
<td>52.0117</td>
</tr>
<tr>
<td>2010-01-18</td>
<td>62.8</td>
<td>63.1</td>
<td>62.8</td>
<td>62.9</td>
<td>30136000</td>
<td>51.5203</td>
</tr>
<tr>
<td>2010-01-19</td>
<td>63.0</td>
<td>63.2</td>
<td>62.0</td>
<td>62.5</td>
<td>47202000</td>
<td>51.1926</td>
</tr>
<tr>
<td>2010-01-20</td>
<td>62.9</td>
<td>63.2</td>
<td>62.2</td>
<td>63.0</td>
<td>52281000</td>
<td>51.6022</td>
</tr>
</tbody>
</table>
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()
df.loc[start:end]

def = df.loc['2017-10-01':'2017-11-15']
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()
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
# Convert Daily Data to Weekly Data

```python
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
```
# 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

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 %D

def KDJ(df, n, m1, m2):
    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

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

```python
#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
```

The Quant Finance PyData Stack

Quantopian

PyThalesians

Zipline

DX Analytics

PyAlgoTrade

QuantLib

StatsModels

NetworkX

scikits-image

PyMC

matplotlib

pandas

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

SciPy

NumPy

SymPy

IPython

Jupyter
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/
References

• Ties de Kok (2017), Learn Python for Research, https://github.com/TiesdeKok/LearnPythonforResearch
• 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