Artificial Intelligence for Investment Analysis

Python AI 投資分析基礎
(Foundations of AI Investment Analysis in Python)

Min-Yuh Day
Assistant Professor
Dept. of Information Management, Tamkang University

http://mail.tku.edu.tw/myday/
2018-10-25
<table>
<thead>
<tr>
<th>週次 (Week)</th>
<th>日期 (Date)</th>
<th>內容 (Subject/Topics)</th>
</tr>
</thead>
</table>
| 1          | 2018/09/13 | 人工智慧投資分析課程介紹  
(Course Orientation on Artificial Intelligence for Investment Analysis) |
| 2          | 2018/09/20 | AI 金融科技: 金融服務創新應用  
(AI in FinTech: Financial Services Innovation and Application) |
| 3          | 2018/09/27 | 機器人理財顧問與AI交談機器人  
(Robo-Advisors and AI Chatbots) |
| 4          | 2018/10/04 | 投資心理學與行為財務學  
(Investing Psychology and Behavioral Finance) |
| 5          | 2018/10/11 | 財務金融事件研究法 (Event Studies in Finance) |
| 6          | 2018/10/18 | 人工智慧投資分析個案研究 I  
(Case Study on Artificial Intelligence for Investment Analysis I) |
週次 (Week)  日期 (Date)  內容 (Subject/Topics)
7  2018/10/25  Python AI投資分析基礎
   (Foundations of AI Investment Analysis in Python)
8  2018/11/01  Python Pandas量化投資分析
   (Quantitative Investing with Pandas in Python)
9  2018/11/08  Python Scikit-Learn 機器學習
   (Machine Learning with Scikit-Learn In Python)
10 2018/11/15  期中報告 (Midterm Project Report)
11 2018/11/22  TensorFlow 深度學習財務時間序列預測 I
   (Deep Learning for Financial Time Series Forecasting with TensorFlow I)
12 2018/11/29  TensorFlow 深度學習財務時間序列預測 II
   (Deep Learning for Financial Time Series Forecasting with TensorFlow II)
課程大綱 (Syllabus)

週次 (Week) 日期 (Date) 內容 (Subject/Topics)

13 2018/12/06 人工智慧投資分析個案研究 II
   (Case Study on Artificial Intelligence for Investment Analysis II)

14 2018/12/13 TensorFlow 深度學習財務時間序列預測 III
   (Deep Learning for Financial Time Series Forecasting with TensorFlow III)

15 2018/12/20 投資組合最佳化與程式交易
   (Portfolio Optimization and Algorithmic Trading)

16 2018/12/27 自然語言處理 (Natural Language Processing)

17 2019/01/03 期末報告 I (Final Project Presentation I)

18 2019/01/10 期末報告 II (Final Project Presentation II)
Foundations of AI Investment Analysis in Python
Outline

• Foundations of AI Investment Analysis in Python
  – Python
    • Programming language
  – Numpy
    • Scientific computing
  – Pandas
    • Data structures and data analysis tools
The Quant Finance PyData Stack

Quantopian

PyThalesians

Zipline

DX Analytics

PyAlgoTrade

QuantLib

StatsModels

Statistics in Python

scikits-image

image processing in python

Matplotlib

Pandas

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

SciPy

NumPy

Python

IPython

Jupyter

Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#7
Python
Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.

Source: https://www.python.org/doc/essays/blurb/
Welcome to Colaboratory!

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. See our FAQ for more info.

Getting Started

- Overview of Colaboratory
- Loading and saving data: Local files, Drive, Sheets, Google Cloud Storage
- Importing libraries and installing dependencies
- Using Google Cloud BigQuery
- Forms, Charts, Markdown, & Widgets
- TensorFlow with GPU
- Machine Learning Crash Course: Intro to Pandas & First Steps with TensorFlow

Highlighted Features

Seedbank

Looking for Colab notebooks to learn from? Check out Seedbank, a place to discover interactive machine learning examples.

TensorFlow execution

Colaboratory allows you to execute TensorFlow code in your browser with a single click. The example below adds two matrices.

\[
\begin{bmatrix}
1 & 1 & 1
\end{bmatrix} +
\begin{bmatrix}
1 & 2 & 3
\end{bmatrix} =
\begin{bmatrix}
2 & 3 & 4
\end{bmatrix}
\]

https://colab.research.google.com/notebooks/welcome.ipynb
Python in Google Colab

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

```python
# Future Value
pv = 100
r = 0.1
n = 7
fv = pv * ((1 + (r)) ** n)
print(round(fv, 2))
```

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

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

```
# Python Function def
def getfv(pv, r, n):
    fv = pv * ((1 + (r)) ** n)
    return fv

fv = getfv(100, 0.1, 7)
print(round(fv, 2))
```

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

Pass
Numpy

NumPy

Base

N-dimensional array package
Python Pandas

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

http://pandas.pydata.org/
Iris flower data set

setosa  versicolor  virginica

Source: https://en.wikipedia.org/wiki/Iris_flower_data_set
Iris Classification

iris.data


<table>
<thead>
<tr>
<th>Sepal Length</th>
<th>Sepal Width</th>
<th>Petal Length</th>
<th>Petal Width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>3.6</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.4</td>
<td>3.9</td>
<td>1.7</td>
<td>0.4</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.6</td>
<td>3.4</td>
<td>1.4</td>
<td>0.3</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>3.4</td>
<td>1.5</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.4</td>
<td>2.9</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.9</td>
<td>3.1</td>
<td>1.5</td>
<td>0.1</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.4</td>
<td>3.7</td>
<td>1.5</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.8</td>
<td>3.4</td>
<td>1.6</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.8</td>
<td>3.0</td>
<td>1.4</td>
<td>0.1</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.3</td>
<td>3.0</td>
<td>1.1</td>
<td>0.1</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.8</td>
<td>4.0</td>
<td>1.2</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.7</td>
<td>4.4</td>
<td>1.5</td>
<td>0.4</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.4</td>
<td>3.9</td>
<td>1.3</td>
<td>0.4</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.3</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.7</td>
<td>3.8</td>
<td>1.7</td>
<td>0.3</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.1</td>
<td>3.8</td>
<td>1.5</td>
<td>0.3</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.4</td>
<td>3.4</td>
<td>1.7</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.1</td>
<td>3.7</td>
<td>1.5</td>
<td>0.4</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.6</td>
<td>3.6</td>
<td>1.0</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.1</td>
<td>3.3</td>
<td>1.7</td>
<td>0.5</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.8</td>
<td>3.4</td>
<td>1.9</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>3.0</td>
<td>1.6</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>3.4</td>
<td>1.6</td>
<td>0.4</td>
<td>Iris-setosa</td>
</tr>
</tbody>
</table>

**setosa**

**virginica**

**versicolor**
Google Colab
Google Colab
Connect Colaboratory to Google Drive

Colaboratory was connected to Google Drive.

Make Colaboratory the default app for files it can open

OK
Google Colab
Google Colab
Google Colab
Run Jupyter Notebook Python3 GPU Google Colab
Google Colab Python Hello World

print('Hello World')
import seaborn as sns
sns.set(style="ticks", color_codes=True)
iris = sns.load_dataset("iris")
g = sns.pairplot(iris, hue="species").
import seaborn as sns
sns.set(style="ticks", color_codes=True)
iris = sns.load_dataset("iris")
g = sns.pairplot(iris, hue="species")

Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import import scatter_matrix

# Load dataset
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)

print(df.head(10))
print(df.tail(10))
print(df.describe())
print(df.info())
print(df.shape)
print(df.groupby('class').size())

plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()

df.hist()
plt.show()

scatter_matrix(df)
plt.show()

sns.pairplot(df, hue="class", size=2)
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix

imported

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']

df = pd.read_csv(url, names=names)
print(df.head(10))

# Load dataset
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']

df = pd.read_csv(url, names=names)
print(df.head(10))

<table>
<thead>
<tr>
<th>sepal-length</th>
<th>sepal-width</th>
<th>petal-length</th>
<th>petal-width</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>3.6</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.4</td>
<td>3.9</td>
<td>1.7</td>
<td>0.4</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.6</td>
<td>3.4</td>
<td>1.4</td>
<td>0.3</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>3.4</td>
<td>1.5</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.4</td>
<td>2.9</td>
<td>1.4</td>
<td>0.2</td>
<td>Iris-setosa</td>
</tr>
<tr>
<td>4.9</td>
<td>3.1</td>
<td>1.5</td>
<td>0.1</td>
<td>Iris-setosa</td>
</tr>
</tbody>
</table>
```python
print(df.tail(10).)
```

<table>
<thead>
<tr>
<th>sepal-length</th>
<th>sepal-width</th>
<th>petal-length</th>
<th>petal-width</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>140</td>
<td>6.7</td>
<td>3.1</td>
<td>5.6</td>
<td>2.4 Iris-virginica</td>
</tr>
<tr>
<td>141</td>
<td>6.9</td>
<td>3.1</td>
<td>5.1</td>
<td>2.3 Iris-virginica</td>
</tr>
<tr>
<td>142</td>
<td>5.8</td>
<td>2.7</td>
<td>5.1</td>
<td>1.9 Iris-virginica</td>
</tr>
<tr>
<td>143</td>
<td>6.8</td>
<td>3.2</td>
<td>5.9</td>
<td>2.3 Iris-virginica</td>
</tr>
<tr>
<td>144</td>
<td>6.7</td>
<td>3.3</td>
<td>5.7</td>
<td>2.5 Iris-virginica</td>
</tr>
<tr>
<td>145</td>
<td>6.7</td>
<td>3.0</td>
<td>5.2</td>
<td>2.3 Iris-virginica</td>
</tr>
<tr>
<td>146</td>
<td>6.3</td>
<td>2.5</td>
<td>5.0</td>
<td>1.9 Iris-virginica</td>
</tr>
<tr>
<td>147</td>
<td>6.5</td>
<td>3.0</td>
<td>5.2</td>
<td>2.0 Iris-virginica</td>
</tr>
<tr>
<td>148</td>
<td>6.2</td>
<td>3.4</td>
<td>5.4</td>
<td>2.3 Iris-virginica</td>
</tr>
<tr>
<td>149</td>
<td>5.9</td>
<td>3.0</td>
<td>5.1</td>
<td>1.8 Iris-virginica</td>
</tr>
</tbody>
</table>
```python
print(df.describe())
```

<table>
<thead>
<tr>
<th></th>
<th>sepal-length</th>
<th>sepal-width</th>
<th>petal-length</th>
<th>petal-width</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>150.000000</td>
<td>150.000000</td>
<td>150.000000</td>
<td>150.000000</td>
</tr>
<tr>
<td>mean</td>
<td>5.843333</td>
<td>3.054000</td>
<td>3.758667</td>
<td>1.198667</td>
</tr>
<tr>
<td>std</td>
<td>0.828066</td>
<td>0.433594</td>
<td>1.764420</td>
<td>0.763161</td>
</tr>
<tr>
<td>min</td>
<td>4.300000</td>
<td>2.000000</td>
<td>1.000000</td>
<td>0.100000</td>
</tr>
<tr>
<td>25%</td>
<td>5.100000</td>
<td>2.800000</td>
<td>1.600000</td>
<td>0.300000</td>
</tr>
<tr>
<td>50%</td>
<td>5.800000</td>
<td>3.000000</td>
<td>4.350000</td>
<td>1.300000</td>
</tr>
<tr>
<td>75%</td>
<td>6.400000</td>
<td>3.300000</td>
<td>5.100000</td>
<td>1.800000</td>
</tr>
<tr>
<td>max</td>
<td>7.900000</td>
<td>4.400000</td>
<td>6.900000</td>
<td>2.500000</td>
</tr>
</tbody>
</table>
print(df.info())
print(df.shape)

print(df.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal-length    150 non-null float64
sepal-width     150 non-null float64
petal-length    150 non-null float64
petal-width     150 non-null float64
class           150 non-null object
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
None

print(df.shape)

(150, 5)
df.groupby('class').size()

print(df.groupby('class').size())

class
Iris-setosa    50
Iris-versicolor 50
Iris-virginica 50
dtype: int64
plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```python
df.hist()
plt.show()
```
scatter_matrix(df)
plt.show()
sns.pairplot(df, hue="class", size=2)
Anaconda
The Most Popular Python Data Science Platform

Source: https://www.anaconda.com/
Download Anaconda

Don't Miss AnacondaCon Apr 8-11 Austin TX!

Download Anaconda Distribution

Version 5.1 | Release Date: February 15, 2018

Download For: 🍊 🍎 🍍

High-Performance Distribution
Easily install 1,000+ data science packages

Package Management
Manage packages, dependencies and environments with conda

Portal to Data Science
Uncover insights in your data and create interactive visualizations

Anaconda 5.1 For macOS Installer

https://www.anaconda.com/download
Download Anaconda Python 3.6 version

Easily install 1,000+ data science packages

Manage packages, dependencies and environments with conda

Uncover insights in your data and create interactive visualizations

Anaconda 5.1 For macOS Installer

Python 3.6 version

- Download

- 64-Bit Graphical Installer (595 MB)
- 64-Bit Command-Line Installer (511 MB)

Python 2.7 version

- Download

- 64-Bit Graphical Installer (588 MB)
- 64-Bit Command-Line Installer (506 MB)

*How to get Python 3.5 or other Python versions
How to Install ANACONDA

https://www.anaconda.com/download/#macos
Install Anaconda

Python 3.6

Anaconda3-5.1.0-MacOSX-x86_64.pkg

Installer package - 623.6 MB
Install Anaconda 3

Anaconda is the leading open data science platform powered by Python.

Share your notebooks and packages on Anaconda Cloud!
Sign up for free

235 python packages included.
Supported packages: 482

Source: https://docs.anaconda.com/anaconda/packages/pkg-docs
# Name                      Version                   Build  Channel
--|---------------------------|-----------------------------|--------|--------------------------

## Install Anaconda 3

```plain
Install Anaconda 3

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
<th>Build</th>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>libedit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libpng</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libgfortran</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libxslt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libxml2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libxml</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libyaml</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libedit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libpng</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libxml2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libxml</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libyaml</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libz</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

---

235 python packages included.
Verify that conda is installed, check current conda version

• `conda --version`

• Update conda to the current version
  – `conda update conda`

http://conda.pydata.org/docs/using/using.html#verify-that-conda-is-installed-check-current-conda-version
Check current conda version
Check current python version
Check conda environments

• conda --version
• python --version
• conda info --envs

http://conda.pydata.org/docs/using/using.html#verify-that-conda-is-installed-check-current-conda-version
Terminal

>_

Terminal

search: terminal

terminal
conda list

# packages in environment at /Users/imiyan/anaconda:

<table>
<thead>
<tr>
<th>Package</th>
<th>Version</th>
<th>Epoch</th>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>_license</td>
<td>1.1</td>
<td></td>
<td>py36_1</td>
</tr>
<tr>
<td>alabaster</td>
<td>0.7.9</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>anaconda</td>
<td>4.3.1</td>
<td></td>
<td>np111py36_0</td>
</tr>
<tr>
<td>anaconda-client</td>
<td>1.6.0</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>anaconda-navigator</td>
<td>1.5.0</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>anaconda-project</td>
<td>0.4.1</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>appnope</td>
<td>0.1.0</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>appscript</td>
<td>1.0.1</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>astroid</td>
<td>1.4.9</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>astropy</td>
<td>1.3</td>
<td></td>
<td>np111py36_0</td>
</tr>
<tr>
<td>babel</td>
<td>2.3.4</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>backports</td>
<td>1.0</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>beautifulsoup4</td>
<td>4.5.3</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>bitarray</td>
<td>0.8.1</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>blaze</td>
<td>0.10.1</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>bokeh</td>
<td>0.12.4</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>boto</td>
<td>2.45.0</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>bottleneck</td>
<td>1.2.0</td>
<td></td>
<td>np111py36_0</td>
</tr>
<tr>
<td>cffi</td>
<td>1.9.1</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>chardet</td>
<td>2.3.0</td>
<td></td>
<td>py36_0</td>
</tr>
<tr>
<td>chest</td>
<td>0.2.3</td>
<td></td>
<td>py36_0</td>
</tr>
</tbody>
</table>

http://conda.pydata.org/docs/using/using.html#verify-that-conda-is-installed-check-current-conda-version
python --version

[iMyday-MacBook-Pro:~ imyday]$ python --version
Python 3.6.0 :: Anaconda 4.3.1 (x86_64)
conda --version

python --version
conda activate py35
conda info --envs

source activate py35

source deactivate py35
source deactivate py35

conda info --envs

# conda environments:
#
py27          /Users/imyday/anaconda/envs/py27
py35          /Users/imyday/anaconda/envs/py35
root          * /Users/imyday/anaconda

source activate py35

python --version
conda activate py35
conda info --envs

source activate py35

source deactivate py35
source deactivate py35

conda info --envs

# conda environments:
#
py27          /Users/imyday/anaconda/envs/py27
py35          /Users/imyday/anaconda/envs/py35
root          * /Users/imyday/anaconda

source activate py35

python --version
conda activate py35
conda info --envs

source activate py35

source deactivate py35
source deactivate py35

conda info --envs

# conda environments:
#
py27          /Users/imyday/anaconda/envs/py27
py35          /Users/imyday/anaconda/envs/py35
root          * /Users/imyday/anaconda

Python
HelloWorld
Anaconda-Navigator
Jupyter Notebook

Select items to perform actions on them.

The notebook list is empty.
Jupyter Notebook
New Python 3
print("hello, world")
from platform import python_version
print("Python Version:", python_version())
Python
Programming
Python Fiddle

Hello Python Fiddle

http://pythonfiddle.com/
print("Hello World")

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

x = 3
print(x)

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

ame = input("Enter a name: ")

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

x = float(input("Write a number "))
Python in Google Colab

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

```python
1
print("hello, world")

doctest:failure

2
# comment
3
from platform import python_version
4
print("Python Version:", python_version())
```

Python Version: 3.6.6

```python
[3]
1 # https://www.learnpython.org/en/
2 # LearnPython.org interactive Python tutorial
3 print("Hello World")
4 print("Hello World\nThis is a message")
5 x = 3
6 print(x)
7 x = 2
8 y = 3
9 print(x, '', y)
```

Hello World
Hello World
This is a message
3
2
3

```python
[4]
1 # Python Variables
2 x = 2
3 price = 2.5
4 word = 'Hello'
5 word = 'Hello'
6 word = "Hello"
7 word = "'Hello'
8 word = "'Hello"'
```
Text input and output

```python
In [1]: print("Hello World")
Hello World

In [2]: print("Hello World\nThis is a message")
Hello World
This is a message

In [3]: x = 3
   print(x)
3

In [4]: x = 2
   y = 3
   print(x, ' ', y)
2 3

In [5]: name = input("Enter a name: ")
Enter a name: Myday

In [6]: x = int(input("What is x? "))
What is x? 80

In [7]: x = float(input("Write a number "))
Write a number 3.6
```
Variables

x = 2
price = 2.5
word = 'Hello'

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

x = 2
x = x + 1
x = 5

Source: http://pythonprogramminglanguage.com/
Python Basic Operators

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

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

7 + 2 = 9
7 - 2 = 5
7 * 2 = 14
7 / 2 = 3.5
7 // 2 = 3
7 % 2 = 1
7 ** 2 = 49
BMI Calculator in Python

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

height_m = height_cm/100
BMI = (weight_kg/(height_m**2))
print("Your BMI is: "+str(round(BMI,1)))
```

Source: http://code.activestate.com/recipes/580615-bmi-code/
BMI Calculator in Python

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

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

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

Enter your height in cm: 170
Enter your weight in kg: 60
Your BMI is: 20.8
```
Future value of a specified principal amount, rate of interest, and a number of years.

Source: https://www.w3resource.com/python-exercises/python-basic-exercise-39.php
# How much is your $100 worth after 7 years?

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

# output = 194.87

Future Value (FV)

\[ pv = 100 \]
\[ r = 0.1 \]
\[ n = 7 \]

\[ fv = pv \times ((1 + (r)) \times n) \]
\[ \text{print}(\text{round}(fv, 2)) \]

194.87
Future Value (FV)

```python
given:

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

future_value = amount * ((1 + (0.01 * interest)) ** years)

print(round(future_value, 2))
```

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

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

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

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

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

Source: http://pythonprogramminglanguage.com/
for loops

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

1
2
3
4
5
6
7
8
9
10

Source: http://pythonprogramminglanguage.com/
for loops

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

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

Source: [http://pythonprogramminglanguage.com/](http://pythonprogramminglanguage.com/)
while loops

age = 10

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

Source: https://learnpython.trinket.io/learn-python-part-8-loops/#/while-loops/about-while-loops
```python
def convertCMtoM(xcm):
    m = xcm/100
    return m

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

1.8
```
Lists

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

4
60
70
90
A tuple in Python is a collection that cannot be modified. A tuple is defined using parenthesis.

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

Source: http://pythonprogramminglanguage.com/tuples/
k = { 'EN':'English', 'FR':'French' }
print(k['EN'])

Sets

animals = {'cat', 'dog'}

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

True
False
True
3
3
2

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

Source: http://cs231n.github.io/python-numpy-tutorial/
with open('myfile.txt', 'w') as file:
    file.write('Hello World
This is Python File Input Output')

with open('myfile.txt', 'r') as file:
    text = file.read()
print(text)
with open('myfile.txt', 'a+') as file:
    file.write(\n' + 'New line')

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

Hello World
This is Python File Input Output
New line
Big Data Analytics with Numpy in Python
NumPy

NumPy
Base
N-dimensional array package
NumPy is the fundamental package for scientific computing with Python.

Source: http://www.numpy.org/
• NumPy provides a multidimensional array object to store homogenous or heterogeneous data; it also provides optimized functions/methods to operate on this array object.

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

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the BSD license, enabling reuse with few restrictions.

Getting Started

- Getting NumPy
- Installing the SciPy Stack
- NumPy and SciPy documentation page
- NumPy Tutorial
- NumPy for MATLAB© Users
- NumPy functions by category
- NumPy Mailing List

For more information on the SciPy Stack (for which NumPy provides the fundamental array data structure), see scipy.org.

http://www.numpy.org/
NumPy ndarray

One-dimensional Array (1-D Array)

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 & 4 & 5 \\
\end{array}
\]

Two-dimensional Array (2-D Array)

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 & 4 & 5 \\
0 & 1 & 2 & 3 & 4 & 5 \\
1 & 6 & 7 & 8 & 9 & 10 \\
m-1 & 11 & 12 & 13 & 14 & 15 \\
m-1 & 16 & 17 & 18 & 19 & 20 \\
\end{array}
\]
v = list(range(1, 6))
v
2 * v

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

Source: Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
NumPy
Base
N-dimensional array package

```python
1 v = list(range(1, 6))
2 v
[1, 2, 3, 4, 5]

1 2 * v
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]

1 import numpy as np
2 v = np.arange(1, 6)
3 v
array([1, 2, 3, 4, 5])

1 2 * v
array([2, 4, 6, 8, 10])
```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
c = a * b

Source: Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
```python
import numpy as np

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

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

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

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

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

```
[[ 0.  0.]
 [0.  0.]]
[[ 1.  1.]]
[[ 7 7]
 [7 7]]
[[ 1.  0.]
 [0.  1.]]
[[ 0.66258211 0.65552598]
 [0.00429934 0.21695824]]
```
Quickstart tutorial

Prerequisites

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

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

The Basics

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

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

\[
\begin{bmatrix}
1.0, & 0.0, & 0.0 \\
0.0, & 1.0, & 2.0
\end{bmatrix}
\]

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

- `ndarray.ndim`: the number of axes (dimensions) of the array. In the Python world, the number of dimensions is referred to as rank.
- `ndarray.shape`:
import numpy as np
a = np.arange(15).reshape(3, 5)

a.shape
a.ndim
a.dtype.name
Matrix

An $m$-by-$n$ matrix has $m$ rows and $n$ columns. Each element in the matrix is denoted by $a_{i,j}$, where $i$ represents the row number and $j$ represents the column number.

$$
\begin{bmatrix}
  a_{1,1} & a_{1,2} & a_{1,3} & \cdots \\
  a_{2,1} & a_{2,2} & a_{2,3} & \cdots \\
  a_{3,1} & a_{3,2} & a_{3,3} & \cdots \\
  \vdots & \vdots & \vdots & \ddots \\
  \vdots & \vdots & \vdots & \ddots 
\end{bmatrix}
$$

Source: https://simple.wikipedia.org/wiki/Matrix_(mathematics)
NumPy ndarray: Multidimensional Array Object
NumPy ndarray

One-dimensional Array (1-D Array)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Two-dimensional Array (2-D Array)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>m-1</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>
import numpy as np

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

One-dimensional Array
(1-D Array)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

a
```

**Two-dimensional Array**

(2-D Array)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>m-1</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>
import numpy as np

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

a
```python
import numpy as np

# Create a 3x4 array with specified elements
a = np.array([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])

# Print the number of dimensions
print(a.ndim)
# Output: 2

# Print the shape of the array
print(a.shape)
# Output: (3, 4)
```

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

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
NumPy Basics: Arrays and Vectorized Computation
NumPy Array

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0,0</td>
<td>0,1</td>
<td>0,2</td>
</tr>
<tr>
<td>1</td>
<td>1,0</td>
<td>1,1</td>
<td>1,2</td>
</tr>
<tr>
<td>2</td>
<td>2,0</td>
<td>2,1</td>
<td>2,2</td>
</tr>
</tbody>
</table>

# Numpy Array

<table>
<thead>
<tr>
<th>Expression</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>arr[:2, 1:]</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>arr[2]</td>
<td>(3,)</td>
</tr>
<tr>
<td>arr[2, :]</td>
<td>(3,)</td>
</tr>
<tr>
<td>arr[2:, :]</td>
<td>(1, 3)</td>
</tr>
<tr>
<td>arr[:, :2]</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>arr[1, :2]</td>
<td>(2,)</td>
</tr>
<tr>
<td>arr[1:2, :2]</td>
<td>(1, 2)</td>
</tr>
</tbody>
</table>

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

https://github.com/wesm/pydata-book
NumPy Basics: Arrays and

In [ ]:
```python
import numpy as np
np.random.seed(12345)
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 5))
np.set_printoptions(precision=4, suppress=True)
```

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

In [ ]:
```python
%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

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

Pandas
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/ea5b6dcd0f55c5fb1e37fb45335ec01ccee8a7939137f5ed269e0/pandas_datareader-0.17.1-py3-none-any.whl (112kB)
    100% |████████████████████████████████| 112kB 2.7MB/s
Collecting lxml (from pandas_datareader)
  Downloading https://files.pythonhosted.org/packages/03/a4/9eea8035fc7c7670e5eab97f34ff2ef0ddd78a491bf96df5acc6db0e63f5/lxml-4.2.5-cp37-cp37m-manylinux1_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-datareader) (2.7.3)
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from pandas-datareader) (1.14.6)
Requirement already satisfied: idna<=2.7 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2) (2.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2) (1.12.0)
Installing collected packages: lxml, pandas-datareader
Successfully installed lxml-4.2.5 pandas-datareader-0.17.1
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')
```python
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)
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), colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'], color='blue')
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()
```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

# Read Stock Data from Yahoo Finance
end = dt.datetime.now()
start = dt.datetime(end.year-2, end.month, end.day)
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_MAs.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()```
```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](https://www.quandl.com/tools/python)
The Quant Finance PyData Stack

Quantopian

PyThalesians

Zipline

DX Analytics

PyAlgoTrade

QuantLib

StatsModels

Statistics in Python

matplotlib

pandas

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

SciPy

NumPy

Python

Ipython

jupyter

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

• Foundations of AI Investment Analysis in Python
  – Python
    • Programming language
  – Numpy
    • Scientific computing
  – Pandas
    • Data structures and data analysis tools
References

• Alfred Essa (2015), Awesome Data Science: 1.0 Jupyter Notebook Tour, https://www.youtube.com/watch?v=e9cSF3eVQv0
• Ties de Kok (2017), Learn Python for Research, https://github.com/TiesdeKok/LearnPythonforResearch
• Numpy Tutorial, https://docs.scipy.org/doc/numpy-dev/user/quickstart.html
• Wes McKinney (2012), Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, O'Reilly Media
• Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
• Michael Heydt (2015), Mastering Pandas for Finance, Packt Publishing
• Michael Heydt (2015), Learning Pandas - Python Data Discovery and Analysis Made Easy, Packt Publishing
• James Ma Weiming (2015), Mastering Python for Finance, Packt Publishing
• Fabio Nelli (2015), Python Data Analytics: Data Analysis and Science using PANDAs, matplotlib and the Python Programming Language, Apress
• Edward Schofield (2013), Modern scientific computing and big data analytics in Python, PyCon Australia, https://www.youtube.com/watch?v=hqOsfS3dP9w
• Python Programming, https://pythonprogramming.net/
• Python, https://www.python.org/
• Python Programming Language, http://pythonprogramminglanguage.com/
• Numpy, http://www.numpy.org/
• Pandas, http://pandas.pydata.org/
• Skikit-learn, http://scikit-learn.org/
• Data School (2015), Machine learning in Python with scikit-learn, https://www.youtube.com/playlist?list=PL5-da3qGB5IcMeMbQuqbbCOQWCs6OYBr5A