AI in Finance
Big Data Analytics

Foundations of AI in Finance
Big Data Analytics with Python

1081AIFBDA06
TLVXM2A (M2449) (8497) (Fall 2019)
(MBA, DBETKU) (3 Credits, Required) [Full English Course]
(Master’s Program in Digital Business and Economics)
Tue, 2, 3, 4, (9:10-12:00) (B1012)

Min-Yuh Day, Ph.D.
Associate Professor
Department of Information Management
Tamkang University

http://mail.tku.edu.tw/myday

2019-10-22
<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2019/09/10</td>
<td>Course Orientation on AI in Finance Big Data Analytics</td>
</tr>
<tr>
<td>2</td>
<td>2019/09/17</td>
<td>AI in FinTech: Financial Services Innovation and Application</td>
</tr>
<tr>
<td>3</td>
<td>2019/09/24</td>
<td>ABC: AI, Big Data, Cloud Computing</td>
</tr>
<tr>
<td>4</td>
<td>2019/10/01</td>
<td>Business Models of Fintech</td>
</tr>
<tr>
<td>5</td>
<td>2019/10/08</td>
<td>Event Studies in Finance</td>
</tr>
<tr>
<td>6</td>
<td>2019/10/15</td>
<td>Case Study on AI in Finance Big Data Analytics I</td>
</tr>
<tr>
<td>7</td>
<td>2019/10/22</td>
<td>Foundations of AI in Finance Big Data Analytics with Python</td>
</tr>
<tr>
<td>8</td>
<td>2019/10/29</td>
<td>Case Study on Financial Industry Practice I</td>
</tr>
<tr>
<td>9</td>
<td>2019/11/05</td>
<td>Quantitative Investing with Pandas in Python</td>
</tr>
<tr>
<td>Week</td>
<td>Date</td>
<td>Subject/Topics</td>
</tr>
<tr>
<td>------</td>
<td>--------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>10</td>
<td>2019/11/12</td>
<td>Midterm Project Report</td>
</tr>
<tr>
<td>11</td>
<td>2019/11/19</td>
<td>Machine Learning in Finance Application with Scikit-Learn In Python</td>
</tr>
<tr>
<td>13</td>
<td>2019/12/03</td>
<td>Case Study on AI in Finance Big Data Analytics II</td>
</tr>
<tr>
<td>14</td>
<td>2019/12/10</td>
<td>Deep Learning for Financial Time Series Forecasting with TensorFlow II</td>
</tr>
<tr>
<td>15</td>
<td>2019/12/17</td>
<td>Case Study on Financial Industry Practice II</td>
</tr>
<tr>
<td>17</td>
<td>2019/12/31</td>
<td>Final Project Presentation I</td>
</tr>
<tr>
<td>18</td>
<td>2020/01/07</td>
<td>Final Project Presentation II</td>
</tr>
</tbody>
</table>
Foundations of AI in Finance
Big Data Analytics with Python
Outline

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

PyTables
StatsModels
SciPy
NumPy
IPython
Python

PyThalesians
SM
Zipline
NetworkX
scikits-image
matplotlib
pandas
y_{it} = \beta x_{it} + \mu_i + \epsilon_{it}

Quantopian
PyAlgoTrade
QuantLib

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

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

# Future Value

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

194.87

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

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

194.87

```
# 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))
```

194.87

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

Pass
Numpy

NumPy
Base
N-dimensional array package
Python
matplotlib

Source: https://matplotlib.org/
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


5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
4.8,3.4,1.9,0.2,Iris-setosa
5.0,3.0,1.6,0.2,Iris-setosa
5.0,3.4,1.6,0.4,Iris-setosa

setosa
virginica
versicolor
Iris Data Visualization

Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html
Connect Google Colab in Google Drive
Google Colab
Google Colab

Colaboratory
offered by https://colab.research.google.com
A data analysis tool that combines code, output, and descriptive text into one collaborative document.
Connect Colaboratory to Google Drive

Colaboratory was connected to Google Drive.

Make Colaboratory the default app for files it can open
Google Colab
Google Colab
Run Jupyter Notebook
Python3 GPU
Google Colab
Google Colab Python Hello World
print('Hello World')
Data Visualization in Google Colab

https://colab.research.google.com/drive/1KRqtEud2Hg4dM2au9bfVQKrxWnWN3O9-

Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html
```python
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 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

# Import Libraries
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
print('imported')

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>
The code snippet demonstrates printing the last 10 rows of a DataFrame containing flower measurements:

```python
print(df.tail(10))
```

The table below shows the last 10 rows of the dataframe:

<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>
```python
print(df.info())
print(df.shape)
```

```
<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()
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 Distribution

Version 5.1 | Release Date: February 15, 2018

Download For: Windows 🍎 Linux

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
Python
HelloWorld
Anaconda-Navigator
Anaconda Navigator

Applications on base (root)

- **lab**
  - Version: 0.31.5
  - Description: An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

- **jupyterlab**
  - Version: 5.4.0
  - Description: Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

- **qtconsole**
  - Version: 4.3.1
  - Description: PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

- **spyder**
  - Version: 3.2.6
  - Description: Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

- **vscode**
  - Version: 1.22.2
  - Description: Streamlined code editor with support for development operations like debugging, task running and version control.

- **glueviz**
  - Version: 0.12.4
  - Description: Multidimensional data visualization across files. Explore relationships within and among related datasets.
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
Hello Python Fiddle

Python Fiddle

print("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)

name = 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/1FEG6DnGvwfUdeo4zJ1zTunjMqf2RkCrT

```python
# comment
from platform import python_version
print("Python Version:", python_version())
```

Python Version: 3.6.6

```python
# https://www.learnpython.org/en/
# LearnPython.org interactive Python tutorial
print("Hello World")
print("Hello World\nThis is a message")
x = 3
print(x)
x = 2
y = 3
print(x, ' ', y)
```

Hello World
Hello World
This is a message
3
2
3

```python
# Python Variables
x = 2
price = 2.5
word = 'Hello'
word = 'Hello'
word = "Hello"
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

\[
x = 2 \\
\text{price} = 2.5 \\
\text{word} = 'Hello'
\]

\[
\text{word} = 'Hello' \\
\text{word} = "Hello" \\
\text{word} = '''Hello'''
\]

\[
x = 2 \\
x = x + 1 \\
x = 5
\]
Python Basic Operators

```python
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
# BMI Calculator in Python

import math

# Input height and weight in cm and kg
height_cm = float(input("Enter your height in cm: 
weight_kg = float(input("Enter your weight in kg: 

# Convert height from cm to m
height_m = height_cm/100

# Calculate BMI
BMI = (weight_kg/(height_m**2))

# Output BMI
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
Future Value (FV)

# How much is your $100 worth after 7 years?

```python
print(100 * 1.1 ** 7)
# output = 194.87
```

Future Value (FV)

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

\[ fv = pv \times ((1 + (r)) \times n) \]

**print** (round(fv, 2))

194.87
Future Value (FV)

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

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

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

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

194.87

if statements

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

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

Pass

Source: http://pythonprogramminglanguage.com/
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"

Source: http://pythonprogramminglanguage.com/
http://pythontutor.com/visualize.html
https://goo.gl/E6w5ph
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
**while** loops

```python
age = 10

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

10
11
12
13
14
15
16
17
18
19

Source: [https://learnpython.trinket.io/learn-python-part-8-loops/#/while-loops/about-while-loops](https://learnpython.trinket.io/learn-python-part-8-loops/#/while-loops/about-while-loops)
Functions

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

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

1.8
x = [60, 70, 80, 90]
print(len(x))
print(x[0])
print(x[1])
print(x[-1])
A **tuple** in Python is a collection that cannot be modified. A tuple is defined using *parenthesis*.

```
x = (10, 20, 30, 40, 50)
print(x[0])  # 10
print(x[1])  # 20
print(x[2])  # 30
print(x[-1])  # 50
```
k = { 'EN':'English', 'FR':'French' }
print(k[ 'EN' ])

Dictionary

'EN' → 'English'

'FR' → 'French'

English

Source: http://pythonprogramminglanguage.com/dictionary/
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
```

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)

Hello World
This is Python File Input Output

text

'Hello World
This is Python File Input Output'
with open('myfile.txt', 'a+') as file:
    file.write('
' + 'New line')

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

<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>
</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>2</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>m-1</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>
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
v = list(range(1, 6))

v

[1, 2, 3, 4, 5]

2 * v

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

import numpy as np

v = np.arange(1, 6)

v

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

2 * v

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

c = array([[4, 10, 18]])

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

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

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

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

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

$m$-by-$n$ matrix

$a_{i,j}$

$n$ columns

$j$ changes

$m$ rows

$i$ changes

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

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>
</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>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>m-1</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)

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

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

array([1, 2, 3, 4, 5])
Two-dimensional Array
(2-D Array)

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

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

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

```
a
```

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

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

```
2
```

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

```
(3, 4)
```
NumPy Basics: Arrays and Vectorized Computation

NumPy Array

axis 1

<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>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>arr[:, 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.

<table>
<thead>
<tr>
<th>Branch: 2nd-edition</th>
<th>New pull request</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>betatim committed with wesm</th>
<th>Add requirements (#71)</th>
<th>Latest commit ea47998 5 days ago</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasets</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

In [ ]:
```python
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)
```

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/ea5b6d5c0f55c5f6be37fb45335ec0fcceca198b8a79339137f5ed269e0/pandas_datareader-0.7.0.tar.gz (112kB)
    |████████████████████████████████| 112kB 2.7MB/s

Collecting lxml (from pandas_datareader)
  Downloading https://files.pythonhosted.org/packages/03/a4/9ee8035f7c7e670e5ea97f34ff2ef0edd78a491bf96df5acc4eb0e63f5/lxml-4.2.5-cp31-none-any.whl (5.8MB 7.5MB/s)

- Collecting pandas>=0.19.2 in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (0.22.0)
- Collecting requests>=2.3.0 in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (2.18.4)
- Collecting wrapt in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (1.10.11)
- Collecting python-dateutil>=2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19.2->pandas_datareader) (2018.5)
- Collecting numpy>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (1.14.6)
- Collecting pytz>=2011k in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19.2->pandas_datareader) (2018.7)
- Collecting idna<=2.7,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (2.7)
- Collecting chartered<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (3.0.1)
- Collecting certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (2018.4.17)
- Collecting urllib3<1.23,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (1.21.1)
- Collecting 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')
```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)
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), colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), 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

# 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), colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), 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

fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
# ! 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()
The Quant Finance PyData Stack

PyThalesians   Zipline   DX Analytics
PyAlgoTrade   QuantLib

Quantopian

PyTables

StatsModels
Statistics in Python

NetworkX

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#5
Summary

- Foundations of AI in Finance
- Big Data Analytics with Python
  - Python
    - Programming language
  - Numpy
    - Scientific computing
  - Pandas
    - Data structures and data analysis tools
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-da3qGB5lCeMbQuqbbCOQWC6OYBr5A