AI in Financial Application

Python AI 智慧金融分析基礎
(Foundations of AI in Finance
Big Data Analytics with Python)

1081AIFA05
EMBA, IMTKU (M2457) (8413) (Fall 2019)
Fri 12,13,14 (19:20-22:10) (D301)

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

http://mail.tku.edu.tw/myday/
2019-11-01
課程大綱 (Syllabus)

週次 (Week) 日期 (Date) 內容 (Subject/Topics)
1 2019/09/13 中秋節 (Mid-Autumn Festival) 放假一天 (Day off)
2 2019/09/20 人工智慧財務金融應用課程介紹 (Course Orientation for AI in Financial Application)
3 2019/09/27 人工智慧投資分析與機器人理財顧問 (Artificial Intelligence for Investment Analysis and Robo-Advisors)
4 2019/10/04 金融科技對話式商務與智慧型交談機器人 (Conversational Commerce and Intelligent Chatbots for Fintech)
5 2019/10/11 國慶日補假 (Bridge Holiday for National Day, Extra Day Off)
6 2019/10/18 財務金融事件研究法 (Event Studies in Finance)
<table>
<thead>
<tr>
<th>週次 (Week)</th>
<th>日期 (Date)</th>
<th>內容 (Subject/Topics)</th>
</tr>
</thead>
</table>
| 7          | 2019/10/25 | 人工智慧財務金融應用個案研究 I  
(Case Study on AI in Financial Application I) |
| 8          | 2019/11/01 | Python AI智慧金融分析基礎  
(Foundations of AI in Finance Big Data Analytics with Python) |
| 9          | 2019/11/08 | Python Pandas 量化投資分析  
(Quantitative Investing with Pandas in Python) |
| 10         | 2019/11/15 | 期中報告 (Midterm Project Report) |
| 11         | 2019/11/22 | Python Scikit-Learn 機器學習財務金融應用  
(Machine Learning in Finance Application with Scikit-Learn In Python) |
| 12         | 2019/11/29 | TensorFlow 深度學習財務金融應用 I  
(Deep Learning for Finance Application with TensorFlow I) |
<table>
<thead>
<tr>
<th>週次 (Week)</th>
<th>日期 (Date)</th>
<th>內容 (Subject/Topics)</th>
</tr>
</thead>
</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) |
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

Quantopian

PyThalesians

Zipline

DX Analytics

PyAlgoTrade

QuantLib

StatsModels (Statistics in Python)

scikit-learn

matplotlib

pandas

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

SciPy

NumPy

SymPy

IPython

Jupyter

Jake VanderPlas

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.

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

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

[Code Snippet]

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

```python
amount = 100
interest = 10 * 0.01 * 10
years = 7
future_value = amount * ((1 + (0.01 * interest)) ** years)
print(round(future_value, 2))
```

```python
# Python Function
def getfv(pv, r, n):
    fv = pv * ((1 + (r)) ** n)
    return fv
fv = getfv(100, 0.1, 7)
print(round(fv, 2))
```

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

Output:
```
194.87
194.87
194.87
```

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


setosa

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
5.0,3.6,1.6,0.4,Iris-setosa

versicolor

virginica
Iris Data Visualization
Connect Google Colab in Google Drive

[Image of Google Drive interface with options for creating new folders, uploading files, and accessing various Google apps like Docs, Sheets, Slides, Forms, Drawings, My Maps, and Sites. There is also an option to connect more apps.]
Google Colab
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

```python
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").
```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](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

# 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))
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>
```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)
```
```python
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 Apple 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
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())

In [1]: print("hello, world")

hello, world

In [2]: from platform import python_version
print("Python Version:", python_version())

Python Version: 3.6.5
Python Programming
Python Fiddle

```
print("Hello Python Fiddle")
```

http://pythonfiddle.com/
```python
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/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

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

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

Python Version: 3.6.6

# 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
```
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

```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
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)))
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
```

Source: http://code.activestate.com/recipes/580615-bmi-code/
Future value of a specified principal amount, rate of interest, and a number of years.

Source: https://www.w3resource.com/python-exercises/python-basic-exercise-39.php
Future Value (FV)

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

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

Source: https://www.w3resource.com/python-exercises/python-basic-exercise-39.php
Future Value (FV)

\[ \text{fv} = \text{pv} \times ((1 + (r))^{n}) \]

\[
\text{print}\left(\text{round}\left(\text{fv}, 2\right)\right)
\]

\[
\begin{align*}
\text{pv} &= 100 \\
\text{r} &= 0.1 \\
\text{n} &= 7 \\
\text{fv} &= \text{pv} \times ((1 + (r))^{n}) \\
\text{print}\left(\text{round}\left(\text{fv}, 2\right)\right) &= 194.87
\end{align*}
\]
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

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

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

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

1
2
3
4
5
6
7
8
9
10

Source: [http://pythonprogramminglanguage.com/](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
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.

\[
x = (10, 20, 30, 40, 50)
\]

```
print(x[0])
print(x[1])
print(x[2])
print(x[-1])
```

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

**Dictionary**

<table>
<thead>
<tr>
<th>'EN'</th>
<th>'English'</th>
</tr>
</thead>
<tbody>
<tr>
<td>'FR'</td>
<td>'French'</td>
</tr>
</tbody>
</table>

English
Sets

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

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

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

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('\n + ' '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

• 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>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>
v = list(range(1, 6))
v
2 * v

import numpy as np
v = np.arange(1, 6)
v
2 * v
NumPy
Base
N-dimensional array package

```python
1 v = list(range(1, 6))
2 v

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

```python
1 2 * v
```

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

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

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

```python
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

c

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

array([ 4, 10, 18])

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

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

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

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

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

# Create an array filled with random values
f = np.random.random((2, 2))
print(f)  # 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.

```
[[ 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
```

https://docs.scipy.org/doc/numpy-dev/user/quickstart.html
import numpy as np
a = np.arange(15).reshape(3, 5)

a.shape
a.ndim
a.dtype.name
A matrix is a rectangular array of numbers, symbols, or expressions, arranged in rows and columns. The size of a matrix is defined by the number of rows and columns it contains. An $m$-by-$n$ matrix has $m$ rows and $n$ columns.

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

- **$a_{i,j}$** represents the element in the $i$-th row and $j$-th column.
- **$m$** rows indicate the number of rows in the matrix.
- **$n$ columns** indicate the number of columns in the matrix.
- **$j$ changes** across the columns.
- **$i$ changes** down the rows.

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

One-dimensional Array
(1-D Array)

0 1  n-1
1 2  3  4  5

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

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

|     | 0   | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  | 20  |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0   | 1   | 2   | 3   | 4   | 5   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 1   | 6   | 7   | 8   | 9   | 10  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| m-1 | 11  | 12  | 13  | 14  | 15  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|     | 16  | 17  | 18  | 19  | 20  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
```python
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]])

print(a.ndim)
2

print(a.shape)
(3, 4)
```

<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>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
</tbody>
</table>
NumPy Basics: Arrays and Vectorized Computation

NumPy Array

```
  0  1  2
0 0,0 0,1 0,2
1 1,0 1,1 1,2
2 2,0 2,1 2,2
```

### Numpy Array

<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.

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

Branch: 2nd-edition

<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

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

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
```
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/ea5b6dcfd0f55c5fb1e37fb45335ec01ceca199b8a79339137f5ed269e0/pandas_datareader-0.7.0.tar.gz (112kB)

Collecting lxml (from pandas_datareader)
  Downloading https://files.pythonhosted.org/packages/03/a4/9ee8035fc7c7670e5eab97f34ff2ef0dd78a491bf96df5accdeb0e63f5/lxml-4.2.5-cp31-cp31m.tar.gz (5.8MB)

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) (2.8.1)
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from pandas-datareader) (1.14.0)
Requirement already satisfied: idna<=2.7,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas-datareader) (2.7)
Requirement already satisfied: charsetcodec>=3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas-datareader) (3.1.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas-datareader) (2018.11.27)
Requirement already satisfied: urllib3<=1.23,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas-datareader) (1.22.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2.8->pandas-datareader) (1.11.0)
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)
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()
```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(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 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()

```

2017-12-22  174.68  175.424  174.500  175.01  16052615.0       0.0         1.0     174.68     175.424     174.500     175.01     16052615.0
2017-12-26  170.80  171.470  169.679  170.57  32998167.0       0.0         1.0     170.80     171.470     169.679     170.57     32998167.0
2017-12-27  170.10  170.780  169.710  170.60  21672062.0       0.0         1.0     170.10     170.780     169.710     170.60     21672062.0
2017-12-28  171.00  171.850  170.480  171.08  15997739.0       0.0         1.0     171.00     171.850     170.480     171.08     15997739.0
2017-12-29  170.52  170.590  169.220  169.23  25643711.0       0.0         1.0     170.52     170.590     169.220     169.23     25643711.0
```
The Quant Finance PyData Stack

Quantopian

PyThalesians

Zipline

DX Analytics

PyAlgoTrade

QuantLib

StatsModels

Statistics in Python

scikit-image

Image processing in Python

Matplotlib

Pandas

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

SciPy

NumPy

SymPy

IPython

Python

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-da3qGB5ICeMbQuqbbCOQWCs6OYBr5A