

財務金融大數據分析



Tamkang
University
淡江大學

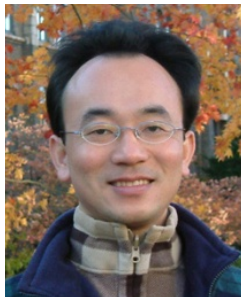
Big Data Analytics in Finance

Python Numpy大數據分析
(Big Data Analytics with Numpy
in Python)

1061BDAF07

MIS EMBA (M2322) (8605)

Thu 12,13,14 (19:20-22:10) (D503)



Min-Yuh Day

戴敏育

Assistant Professor

專任助理教授

Dept. of Information Management, Tamkang University

淡江大學 資訊管理學系

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

2017-11-09



課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2017/09/21	財務金融大數據分析課程介紹 (Course Orientation for Big Data Analytics in Finance)
2	2017/09/28	金融科技商業模式 (Business Models of Fintech)
3	2017/10/05	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	2017/10/12	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	2017/10/19	事件研究法 (Event Study)
6	2017/10/26	財務金融大數據分析個案研究 I (Case Study on Big Data Analytics in Finance I)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
7	2017/11/02	Python 財務大數據分析基礎 (Foundations of Finance Big Data Analytics in Python)
8	2017/11/09	Python Numpy 大數據分析 (Big Data Analytics with Numpy in Python)
9	2017/11/16	Python Pandas 財務大數據分析 (Finance Big Data Analytics with Pandas in Python)
10	2017/11/23	期中報告 (Midterm Project Report)
11	2017/11/30	文字探勘分析技術與自然語言處理 (Text Mining Techniques and Natural Language Processing)
12	2017/12/07	Python Keras 深度學習 (Deep Learning with Keras in Python)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
13	2017/12/14	財務金融大數據分析個案研究 II (Case Study on Big Data Analytics in Finance II)
14	2017/12/21	TensorFlow深度學習 (Deep Learning with TensorFlow)
15	2017/12/28	財務金融大數據深度學習 (Deep Learning for Finance Big Data)
16	2018/01/04	社會網絡分析 (Social Network Analysis)
17	2018/01/11	期末報告 I (Final Project Presentation I)
18	2018/01/18	期末報告 II (Final Project Presentation II)

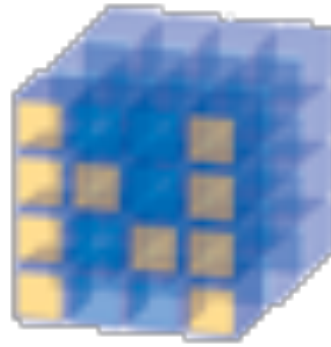
Big Data Analytics

with

Numpy

in Python

NumPy



NumPy
Base

**N-dimensional array
package**

NumPy
is the
fundamental package
for
scientific computing
with **Python.**

Wes McKinney (2012), Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, O'Reilly Media

Data Wrangling with Pandas, NumPy, and IPython

Python for
Data Analysis



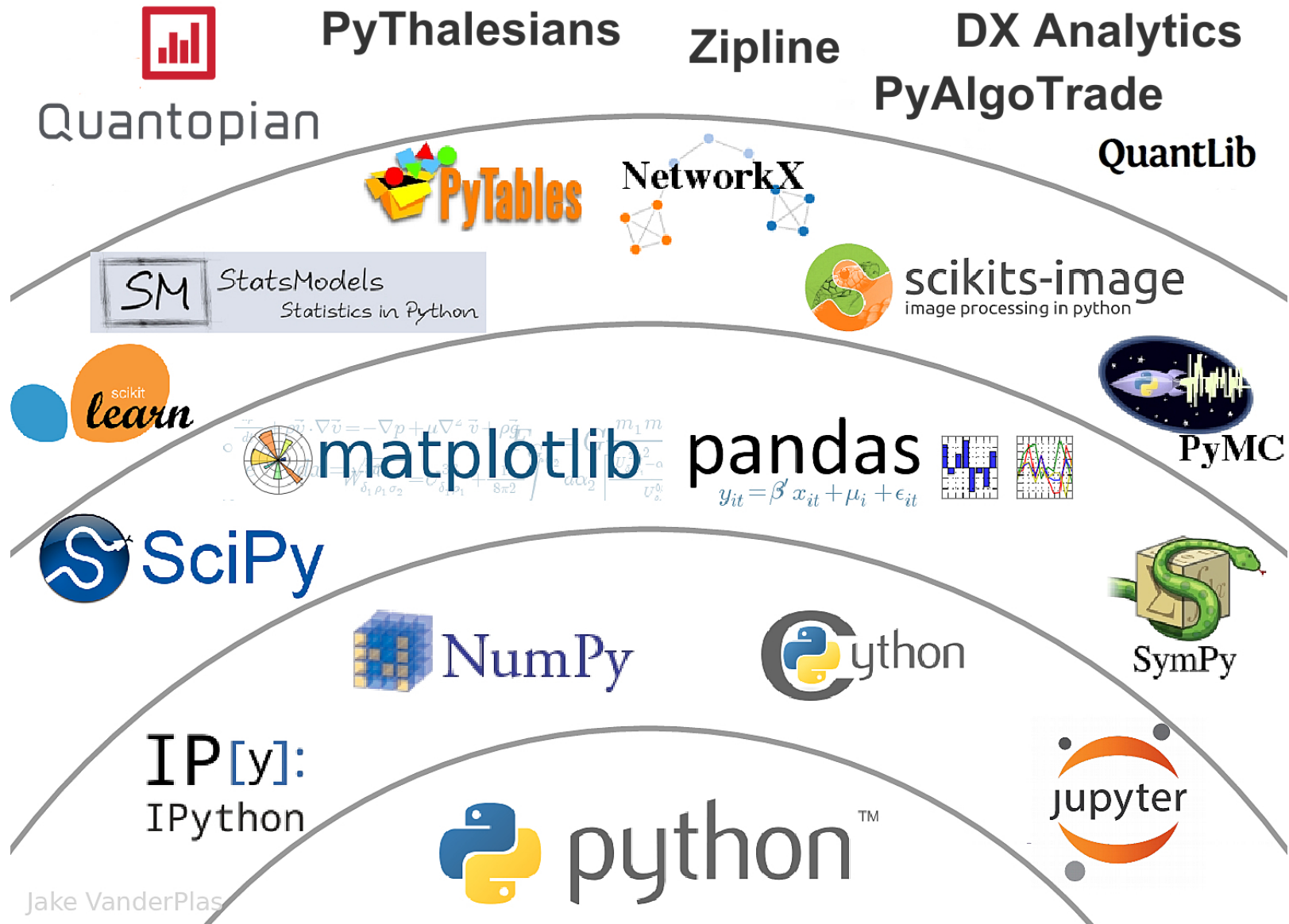
O'REILLY®

Wes McKinney

Copyrighted Material

Front Cover

The Quant Finance PyData Stack



Jake VanderPlas

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

NumPy



NumPy

Scipy.org

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.

About NumPy

[License](#)

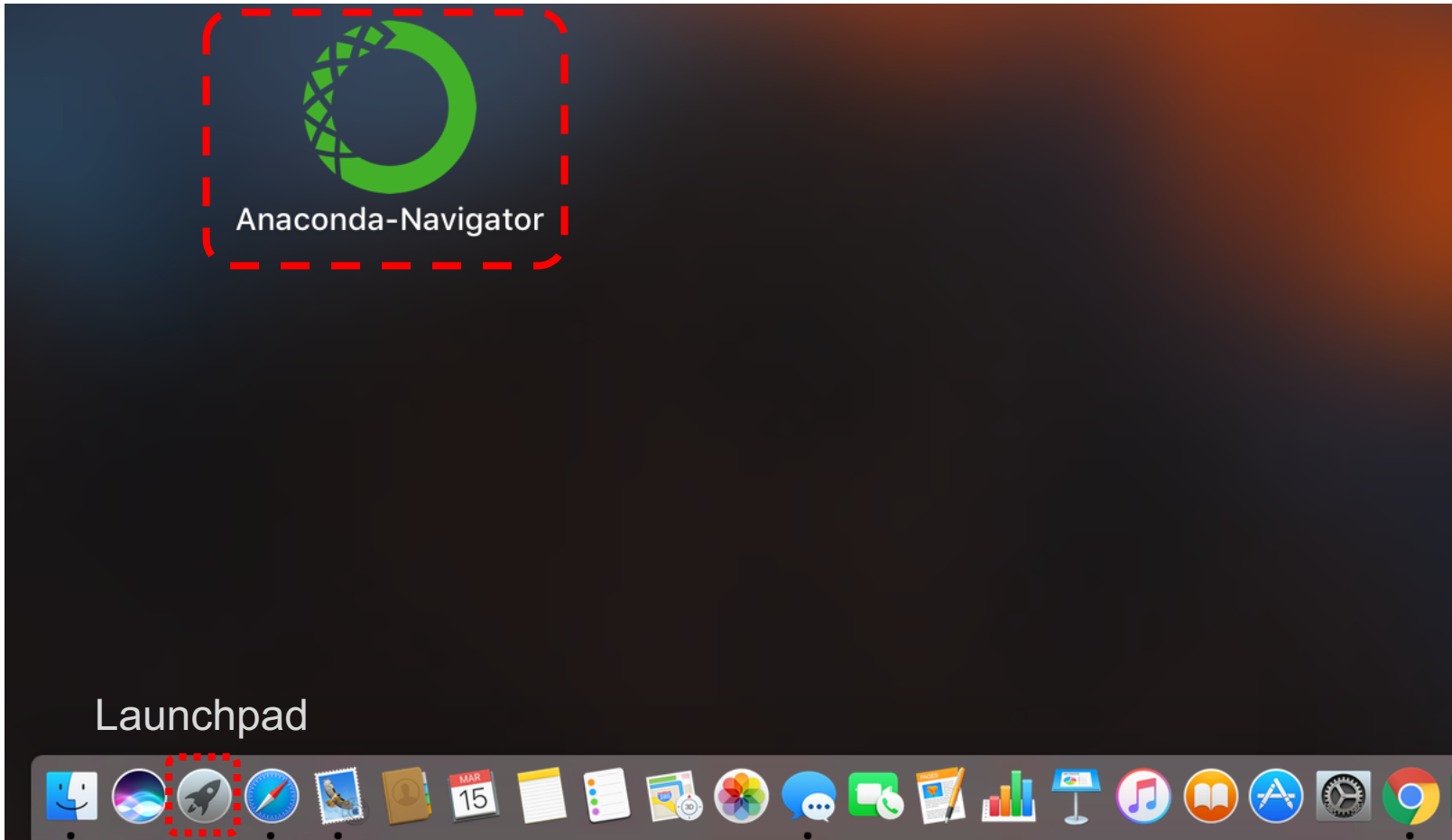
[Old array packages](#)





Python

Anaconda-Navigator



Anaconda-Navigator

The screenshot displays the Anaconda Navigator web interface. At the top, the title bar reads "Anaconda Navigator". The main header features the "ANACONDA NAVIGATOR" logo on the left, an "Upgrade Now" button with an information icon, and a "Sign in to Anaconda Cloud" button on the right. A left-hand navigation sidebar includes links for Home, Environments, Projects (beta), Learning, and Community, along with buttons for Documentation, Developer Blog, and Feedback, and social media icons for Twitter, YouTube, and GitHub.

The main content area shows a grid of application cards under the heading "Applications on root Channels". A central dialog box is overlaid on the interface, containing the following text:

ANACONDA NAVIGATOR

Thanks for installing Anaconda!

Anaconda Navigator helps you easily start important Python applications and manage the packages in your local Anaconda installation. It also connects you to online resources for learning and engaging with the Python, SciPy, and PyData community.

To help us improve Anaconda Navigator, fix bugs, and make it even easier for everyone to use Python, we gather anonymized usage information, just like most web browsers and mobile apps.

To opt out of this, please uncheck below (You can always change this setting in the Preferences menu).





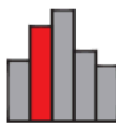

Yes, I'd like to help improve Anaconda.

At the bottom of the dialog, there are two buttons: "Ok" and "Ok, and don't show again". The "Ok, and don't show again" button is highlighted with a red dashed border.

The background application cards include:

- jupyter notebook** (version 4.3.1): Web-based, interactive computing environment. Edit and run human-readable code while describing the data.
- spyder** (version 3.1.2): Python Development Environment. Powerful Python IDE with interactive testing, introspection features.
- anaconda-fusion** (version 1.0.2): Integration between Excel and Anaconda via Notebooks. Run data science functions, interact with results and create advanced visualizations in a code-free app inside Excel.
- glueviz** (version 0.9.1): Multidimensional data visualization across files. Explore relationships within and among related datasets.
- rstudio** (version 1.0.136): A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Jupyter Notebook

 <p>jupyter notebook</p> <p>↗ 4.3.1</p> <p>Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.</p> <p>Launch</p>	 <p>IP[y]:</p> <p>qtconsole</p> <p>4.2.1</p> <p>PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.</p> <p>Launch</p>	 <p>spyder</p> <p>↗ 3.1.2</p> <p>Scientific PYTHON Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features</p> <p>Launch</p>
 <p>anaconda-fusion</p> <p>1.0.2</p> <p>Integration between Excel ® and Anaconda via Notebooks. Run data science functions, interact with results and create advanced visualizations in a code-free app inside Excel</p> <p>Install</p>	 <p>glueviz</p> <p>0.9.1</p> <p>Multidimensional data visualization across files. Explore relationships within and among related datasets.</p> <p>Install</p>	 <p>rstudio</p> <p>1.0.136</p> <p>A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.</p> <p>Install</p>

Jupyter Notebook

New Python 3

The screenshot shows a web browser window with the address bar displaying `localhost:8888/tree/Documents/Data/SCDBA`. The JupyterLab interface includes a top navigation bar with the Jupyter logo and a 'Logout' button. Below this, there are tabs for 'Files', 'Running', and 'Clusters'. A message reads 'Select items to perform actions on them.' To the right of this message are 'Upload', 'New', and a refresh icon. The 'New' dropdown menu is open, showing options: 'Text File', 'Folder', 'Terminal', 'Notebooks', and 'Python 3'. The 'Python 3' option is highlighted and enclosed in a red dashed box. The breadcrumb path below the menu is `Documents / Data / SCDBA`, and the file list below it shows a folder named `..`.

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

The image shows a web browser window displaying a Jupyter Notebook. The browser's address bar shows the URL `localhost:8888/notebooks/Documents/Data/SCDBA/HelloWorld.ipynb`. The notebook's title bar reads "jupyter HelloWorld (autosaved)". A menu bar includes "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". On the right side of the menu bar, there is a "Python 3" dropdown and a "Logout" button. Below the menu bar is a toolbar with various icons. The "Run" icon (a right-pointing arrow) is highlighted with a red dashed box. The main content area of the notebook shows a code cell with the input `In [1]: print("hello, world")` and the output `hello, world`.

Text input and output

```
print("Hello World")
```

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

```
x = 3  
print(x)
```

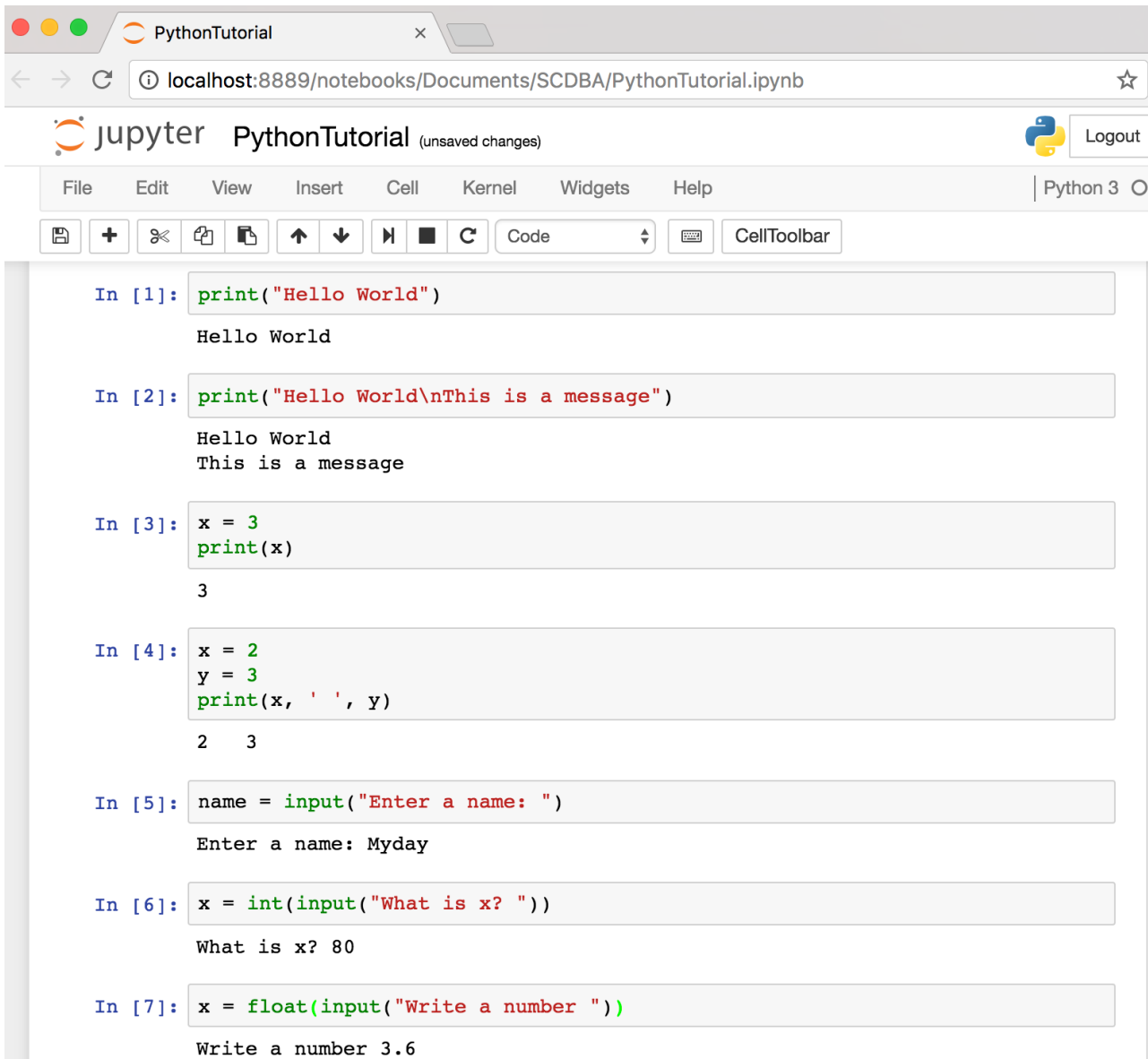
```
x = 2  
y = 3  
print(x, ' ', y)
```

```
name = input("Enter a name: ")
```

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

```
x = float(input("Write a number "))
```

Text input and output



The screenshot shows a Jupyter Notebook window titled "PythonTutorial" with the URL `localhost:8889/notebooks/Documents/SCDBA/PythonTutorial.ipynb`. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations and execution. The notebook contains seven code cells, each with its input and output:

```
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
```


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

```
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

 jupyter PythonTutorial Last Checkpoint: a minute ago (unsaved changes)

 Logout

File Edit View Insert Cell Kernel Widgets Help

Python 3 

          Code   CellToolbar

```
In [1]: 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
```

```
In [ ]:
```

Future value
of a specified
principal amount,
rate of interest, and
a number of years

Future Value (FV)

```
# How much is your $100 worth after 7 years?  
print(100 * 1.1 ** 7)  
# output = 194.87
```

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

```
194.871710000000012
```

Future Value (FV)

```
pv = 100  
r = 0.1  
n = 7
```

```
fv = pv * ((1 + (r)) ** n)  
print(round(fv, 2))
```

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

194.87

Future Value (FV)

```
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

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


if elif else

```
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"
```

A

<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
```

for loops

```
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

```
age = 10
```

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

```
10  
11  
12  
13  
14  
15  
16  
17  
18  
19
```

Functions

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

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

1.8

Lists

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

4

60

70

90

Tuples

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])
```

10

20

30

50

Dictionary

```
k = { 'EN': 'English', 'FR': 'French' }  
print(k['EN'])
```

Dictionary

'EN' → 'English'

'FR' → 'French'

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
```

```
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))
```

File Input / Output

```
with open('myfile.txt', 'w') as file:  
    file.write('Hello World\nThis is Python File Input Output')  
  
with open('myfile.txt', 'r') as file:  
    text = file.read()  
print(text)
```

```
with open('myfile.txt', 'w') as file:  
    file.write('Hello World\nThis 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\nThis is Python File Input Output'
```

File Input / Output

```
with open('myfile.txt', 'a+') as file:  
    file.write('\n' + 'New line')
```

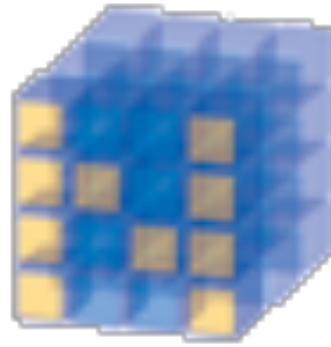
```
with open('myfile.txt', 'r') as file:  
    text = file.read()  
print(text)
```

```
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
```

NumPy



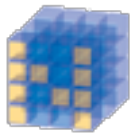
NumPy
Base

**N-dimensional array
package**



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.



NumPy

NumPy ndarray

One-dimensional Array (1-D Array)

0	1			n-1
1	2	3	4	5

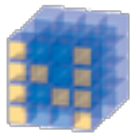
Two-dimensional Array (2-D Array)

	0	1			n-1
0	1	2	3	4	5
1	6	7	8	9	10
	11	12	13	14	15
m-1	16	17	18	19	20



NumPy

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



NumPy

Base
N-dimensional
array package

```
v = range (1, 6)  
print(v)
```

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

```
2 * v
```

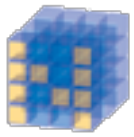
```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

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

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

```
2 * v
```

```
array([ 2,  4,  6,  8, 10])
```

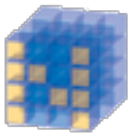
NumPy

NumPy Create Array

```
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])
```



NumPy

NumPy

```
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.]]
[[ 0.22886991  0.68473232]
 [ 0.20683825  0.16589995]]
```

Numpy Quickstart Tutorial

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`

Table Of Contents

- Quickstart tutorial
 - Prerequisites
 - The Basics
 - An example
 - Array Creation
 - Printing Arrays
 - Basic Operations
 - Universal Functions
 - Indexing, Slicing and Iterating
 - Shape Manipulation
 - Changing the shape of an array
 - Stacking together different arrays
 - Splitting one array into several smaller ones
 - Copies and Views
 - No Copy at All
 - View or Shallow Copy
 - Deep Copy
 - Functions and Methods Overview
 - Less Basic
 - Broadcasting rules
 - Fancy indexing and index tricks
 - Indexing with Arrays of

```
import numpy as np
```

```
a = np.arange(15).reshape(3, 5)
```

a.shape

a.ndim

a.dtype.name

```
import numpy as np  
a = np.arange(15).reshape(3, 5)  
a
```

```
array([[ 0,  1,  2,  3,  4],  
       [ 5,  6,  7,  8,  9],  
       [10, 11, 12, 13, 14]])
```

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

```
(3, 5)
```

```
a.ndim
```

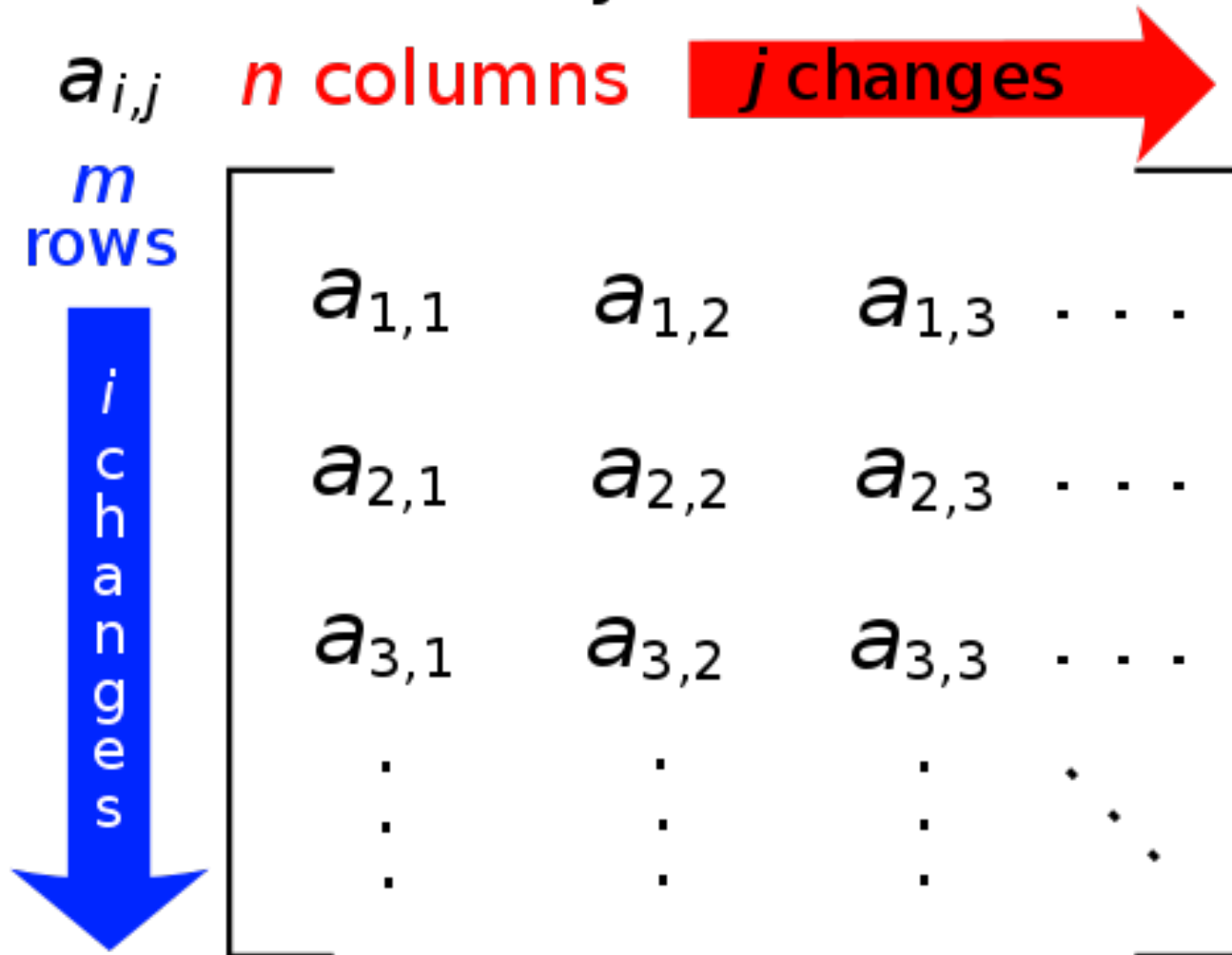
```
2
```

```
a.dtype.name
```

```
'int64'
```

Matrix

m -by- n matrix



NumPy ndarray: Multidimensional Array Object

NumPy ndarray

One-dimensional Array (1-D Array)

0	1			n-1
1	2	3	4	5

Two-dimensional Array (2-D Array)

	0	1			n-1
0	1	2	3	4	5
1	6	7	8	9	10
	11	12	13	14	15
m-1	16	17	18	19	20

```
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])
```



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

Two-dimensional Array (2-D Array)

	0	1		n-1	
0	1	2	3	4	5
1	6	7	8	9	10
	11	12	13	14	15
m-1	16	17	18	19	20

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

```
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
```

0	1	2	3
10	11	12	13
20	21	22	23

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

```
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)
```

0	1	2	3
10	11	12	13
20	21	22	23

NumPy Basics: Arrays and Vectorized Computation

NumPy Array

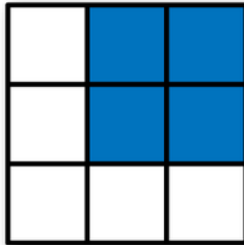
axis 1

		0	1	2
axis 0	0	0, 0	0, 1	0, 2
	1	1, 0	1, 1	1, 2
	2	2, 0	2, 1	2, 2

Numpy Array

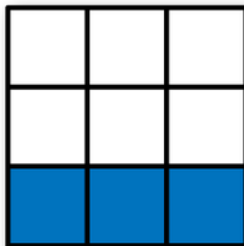
Expression

Shape



`arr[:2, 1:]`

`(2, 2)`



`arr[2]`

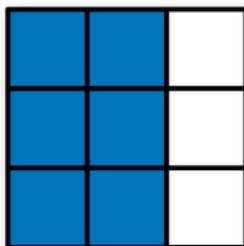
`(3,)`

`arr[2, :]`

`(3,)`

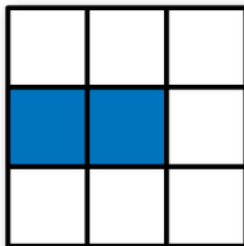
`arr[2:, :]`

`(1, 3)`



`arr[:, :2]`

`(3, 2)`



`arr[1, :2]`

`(2,)`

`arr[1:2, :2]`


`(1, 2)`
















Wes McKinney (2017), "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython", 2nd Edition, O'Reilly Media.

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

52 commits 2 branches 0 releases 6 contributors

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

 **betatim** committed with **wesm** Add requirements (#71) Latest commit ea47998 5 days ago

 datasets	Add Kaggle titanic dataset	5 months ago
 examples	Remove sex column from tips dataset	4 months ago
 .gitignore	Add gitignore	2 years ago
 COPYING	Use MIT license for code examples	a month ago
 README.md	Add launch in Azure Notebooks button (#70)	19 days ago
 appa.ipynb	Make more cells markdown instead of raw	a month ago
 ch02.ipynb	Make more cells markdown instead of raw	a month ago
 ch03.ipynb	Make more cells markdown instead of raw	a month ago
 ch04.ipynb	Convert all notebooks to v4 format	a month ago
 ch05.ipynb	Make more cells markdown instead of raw	a month ago
 ch06.ipynb	Make more cells markdown instead of raw	a month ago
 ch07.ipynb	Convert all notebooks to v4 format	a month ago
 ch08.ipynb	Make more cells markdown instead of raw	a month ago
 ch09.ipynb	Make more cells markdown instead of raw	a month ago
 ch10.ipynb	Make more cells markdown instead of raw	a month ago

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


Wes McKinney (2017), "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython", 2nd Edition, O'Reilly Media.




wesm / pydata-book Watch 640 Star 4,031 Fork 4,348

[Code](#) [Issues 2](#) [Pull requests 0](#) [Projects 0](#) [Insights](#)

Branch: 2nd-edition **pydata-book / ch04.ipynb** Find file Copy path

wesm Convert all notebooks to v4 format c2780a0 on Sep 27

2 contributors 

1857 lines (1856 sloc) | 32.6 KB Raw Blame History   

NumPy Basics: Arrays and

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

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

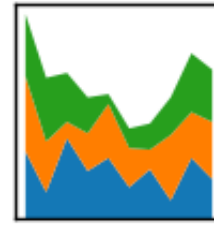
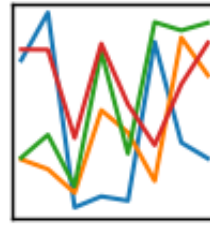
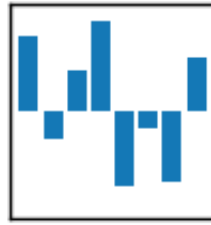
```
In [ ]: %time for _ in range(10): my_arr2 = my_arr * 2
%time for _ in range(10): my_list2 = [x * 2 for x in my_list]
```

The NumPy ndarray: A Multidimensional Array Object

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


pandas

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



Python

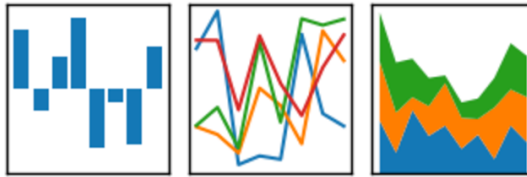
Pandas for

Finance

pandas

pandas

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



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

Python Data Analysis Library

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

pandas is a [NUMFocus](#) sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project.

A Fiscally Sponsored Project of

NUMFOCUS
OPEN CODE = BETTER SCIENCE

0.19.2 Final (December 24, 2016)

This is a minor bug-fix release in the 0.19.x series and includes some small regression fixes, bug fixes and performance improvements.

Highlights include:

- Compatibility with Python 3.6

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

VERSIONS

Release

0.19.2 - December 2016

[download](#) // [docs](#) // [pdf](#)

Development

0.20.0 - 2017

[github](#) // [docs](#)

Previous Releases

0.19.1 - [download](#) // [docs](#) // [pdf](#)

0.19.0 - [download](#) // [docs](#) // [pdf](#)

0.18.1 - [download](#) // [docs](#) // [pdf](#)

0.18.0 - [download](#) // [docs](#) // [pdf](#)

0.17.1 - [download](#) // [docs](#) // [pdf](#)

0.17.0 - [download](#) // [docs](#) // [pdf](#)

0.16.2 - [download](#) // [docs](#) // [pdf](#)

0.16.1 - [download](#) // [docs](#) // [pdf](#)

0.16.0 - [download](#) // [docs](#) // [pdf](#)

0.15.2 - [download](#) // [docs](#) // [pdf](#)

0.15.1 - [download](#) // [docs](#) // [pdf](#)

0.15.0 - [download](#) // [docs](#) // [pdf](#)

0.14.1 - [download](#) // [docs](#) // [pdf](#)

pandas

Python Data Analysis Library

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

Jupyter Notebook New Python 3

The screenshot shows the Jupyter Notebook web interface. The browser address bar displays `localhost:8888/tree/Documents/SCDBA/Pandas`. The Jupyter logo is on the left, and a 'Logout' button is on the right. Below the logo are tabs for 'Files', 'Running', and 'Clusters'. A message says 'Select items to perform actions on them.' To the right are 'Upload' and 'New' buttons. The 'New' button is open, showing a dropdown menu with options: 'Text File', 'Folder', 'Terminal', 'Notebooks', and 'Python 3'. The 'Python 3' option is highlighted with a red dashed box. The breadcrumb path is `Home / Documents / SCDBA / Pandas`. Below the path is a folder icon and `..`. The main area contains the text 'Notebook list empty.'

Creating pd.DataFrame

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

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

df
```

Out[1]:

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

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

Pandas DataFrame

```
type(df)
```

```
type(df)
```

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

conda install pandas-datareader

```
imyday — -bash — 80x24
[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-datarea 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
```

Jupyter Notebook New Python 3

The screenshot shows the Jupyter Notebook web interface. The browser address bar displays `localhost:8888/tree/Documents/SCDBA/Pandas`. The Jupyter logo is on the left, and a 'Logout' button is on the right. Below the logo are tabs for 'Files', 'Running', and 'Clusters'. A message says 'Select items to perform actions on them.' To the right are 'Upload' and 'New' buttons. The 'New' button is open, showing a list of options: 'Text File', 'Folder', 'Terminal', 'Notebooks', and 'Python 3'. The 'Python 3' option is highlighted with a red dashed box. The breadcrumb path is `Home / Documents / SCDBA / Pandas`. The main content area shows a folder icon and the text 'Notebook list empty.'


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

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

```
dates = pd.date_range('20170301',
periods=6)
dates
```

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
print('Hello Pandas')
```

Hello Pandas

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

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

```
In [3]: dates = pd.date_range('20170301', periods=6)
dates
```

```
Out[3]: DatetimeIndex(['2017-03-01', '2017-03-02', '2017-03-03', '2017-03-04',
                        '2017-03-05', '2017-03-06'],
                        dtype='datetime64[ns]', freq='D')
```

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

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

	A	B	C	D
2017-03-01	-1.613934	0.168983	0.686730	1.520077
2017-03-02	0.656124	-1.155197	-1.969431	0.444569
2017-03-03	-0.242393	-0.372972	-0.310283	1.102838
2017-03-04	-0.034627	-0.422640	0.489834	1.498832
2017-03-05	-1.022516	-0.841598	-0.136151	-0.767695
2017-03-06	1.208003	3.114586	-0.451570	0.579479

```
df = pd.DataFrame(np.random.randn(4,6),
index=['student1', 'student2', 'student3',
'student4'], columns=list('ABCDEF'))
df
```

```
df = pd.DataFrame(np.random.randn(4,6), index=['student1', 'student2', 'student3', 'student4'], columns=list('ABCDEF'))
df
```

	A	B	C	D	E	F
student1	-0.420406	0.829262	-0.326521	-0.037699	-1.350555	-0.617676
student2	0.310825	-0.356479	0.149704	-0.685609	-0.744307	-0.488782
student3	-1.295312	0.765656	1.701502	-0.415809	-0.454114	0.397702
student4	0.979525	0.367767	1.869465	0.988012	0.916746	0.165911

```
df2 = pd.DataFrame({ 'A' : 1.,
' B' : pd.Timestamp('20170322'),
' C' : pd.Series(2.5,index=list(range(4)),dtype='float32'),
' D' : np.array([3] * 4,dtype='int32'),
' E' : pd.Categorical(["test","train","test","train"]),
' F' : 'foo' })
```

df2

```
df2 = pd.DataFrame({ 'A' : 1.,
' B' : pd.Timestamp('20170322'),
' C' : pd.Series(2.5,index=list(range(4)),dtype='float32'),
' D' : np.array([3] * 4,dtype='int32'),
' E' : pd.Categorical(["test","train","test","train"]),
' F' : 'foo' })
```

df2

	A	B	C	D	E	F
0	1.0	2017-03-22	2.5	3	test	foo
1	1.0	2017-03-22	2.5	3	train	foo
2	1.0	2017-03-22	2.5	3	test	foo
3	1.0	2017-03-22	2.5	3	train	foo

df2.dtypes

```
df2.dtypes
```

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

Yahoo Finance Symbols: AAPL

Apple Inc. (AAPL)

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

YAHOO!
FINANCE

Search for news, symbols or companies

Search

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

S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

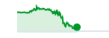
-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,245.40

-1.10 (-0.09%)



Quote Lookup

Search for symbols or companies: YHOO, GOOG, DIS



Symbols similar to 'aapl'

All Markets ▾

All (9)

Stocks (6)

Mutual Funds (0)

ETFs (1)

Indices (2)

Futures (0)

Currencies (0)

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

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

Apple Inc. (AAPL) - NasdaqGS



Search for news, symbols or companies

Search

Apple Inc. (AAPL)

NasdaqGS - NasdaqGS Delayed Price. Currency in USD

[★ Add to watchlist](#)

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

At close: 4:00PM EDT

After hours: 7:59PM EDT

Summary

[Conversations](#)

[Statistics](#)

[Profile](#)

[Financials](#)

[Options](#)

[Holders](#)

[Historical Data](#)

[Analysts](#)

Previous Close	141.46	Market Cap	733.68B
Open	142.11	Beta	1.45
Bid	139.31 x 100	PE Ratio (TTM)	16.79
Ask	139.40 x 1300	EPS (TTM)	8.33
Day's Range	139.73 - 142.80	Earnings Date	Apr 24, 2017 - Apr 28, 2017
52 Week Range	89.47 - 142.80	Dividend & Yield	2.28 (1.63%)
Volume	39,529,912	Ex-Dividend Date	N/A
Avg. Volume	26,889,183	1y Target Est	143.29

1D 5D 1M 6M YTD 1Y 2Y **5Y** 10Y MAX

[Interactive chart](#)



Trade prices are not sourced from all markets

<http://finance.yahoo.com/quote/AAPL?p=AAPL>

Yahoo Finance Charts: Apple Inc. (AAPL)

YAHOO! FINANCE

Go to Quote Summary Page



S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

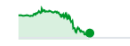
-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,245.50

-1.00 (-0.08%)



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



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

Apple Inc. (AAPL) Historical Data

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



Search for news, symbols or companies

Search

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

S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,245.60

-0.90 (-0.07%)



Apple Inc. (AAPL)

NasdaqGS - NasdaqGS Delayed Price. Currency in USD

Add to watchlist

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

At close: 4:00PM EDT

After hours: 7:59PM EDT

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

Thank you for helping us improve your Yahoo experience

Learn more about your feedback.

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

Show: Historical Prices

Frequency: Daily

Apply

Currency in USD

Download Data

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

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

Yahoo Finance Historical Prices

Apple Inc. (AAPL)

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

Currency in USD



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

<http://finance.yahoo.com/quote/AAPL/history>

Yahoo Finance Historical Prices

Apple Inc. (AAPL)



Search for news, symbols or companies

Search

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

Show: Historical Prices

Frequency: Daily

Apply

Download Data

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

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

Yahoo Finance Historical Prices

Apple Inc. (AAPL)



Search for news, symbols or companies

Search

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

Show: Historical Prices

Frequency: Daily

Apply

Download Data

Currency in USD

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

Yahoo Finance Historical Prices

<http://ichart.finance.yahoo.com/table.csv?s=AAPL>

table.csv

```
Date,Open,High,Low,Close,Volume,Adj Close
2017-03-21,142.110001,142.800003,139.729996,139.839996,39116800,139.839996
2017-03-20,140.399994,141.50,140.229996,141.460007,20213100,141.460007
2017-03-17,141.00,141.00,139.889999,139.990005,43597400,139.990005
2017-03-16,140.720001,141.020004,140.259995,140.690002,19132500,140.690002
2017-03-15,139.410004,140.75,139.029999,140.460007,25566800,140.460007
2017-03-14,139.300003,139.649994,138.839996,138.990005,15189700,138.990005
2017-03-13,138.850006,139.429993,138.820007,139.199997,17042400,139.199997
2017-03-10,139.25,139.360001,138.639999,139.139999,19488000,139.139999
2017-03-09,138.740005,138.789993,137.050003,138.679993,22065200,138.679993
2017-03-08,138.949997,139.800003,138.820007,139.00,18681800,139.00
2017-03-07,139.059998,139.979996,138.789993,139.520004,17267500,139.520004
2017-03-06,139.369995,139.770004,138.600006,139.339996,21155300,139.339996
2017-03-03,138.779999,139.830002,138.589996,139.779999,21108100,139.779999
2017-03-02,140.00,140.279999,138.759995,138.960007,26153300,138.960007
2017-03-01,137.889999,140.149994,137.600006,139.789993,36272400,139.789993
2017-02-28,137.080002,137.440002,136.699997,136.990005,23403500,136.990005
2017-02-27,137.139999,137.440002,136.279999,136.929993,20196400,136.929993
2017-02-24,135.910004,136.660004,135.279999,136.660004,21690900,136.660004
2017-02-23,137.380005,137.479996,136.300003,136.529999,20704100,136.529999
2017-02-22,136.429993,137.119995,136.110001,137.110001,20745300,137.110001
```

Yahoo Finance Charts

Alphabet Inc. (GOOG)

YAHOO! FINANCE

Go to Quote Summary Page



S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,245.60

-0.90 (-0.07%)



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



Dow Jones Industrial Average (^DJI)

YAHOO! FINANCE

Go to Quote Summary Page



S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

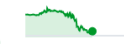
-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,244.90

-1.60 (-0.13%)



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



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

TSEC weighted index (^TWII) - Taiwan

YAHOO! FINANCE

Go to Quote Summary Page



S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

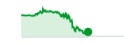
-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,245.10

-1.40 (-0.11%)



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



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

Taiwan Semiconductor Manufacturing Company Limited (2330.TW)

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



Search for news, symbols or companies

Search

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

S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

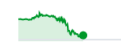
-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,244.90

-1.60 (-0.13%)



Taiwan Semiconductor Manufacturing Company Limited (2330.TW)

Taiwan - Taiwan Delayed Price. Currency in TWD

Add to watchlist

192.50 -2.50 (-1.28%)

As of 9:52AM CST. Market open.

Summary

Conversations

Statistics

Profile

Financials

Options

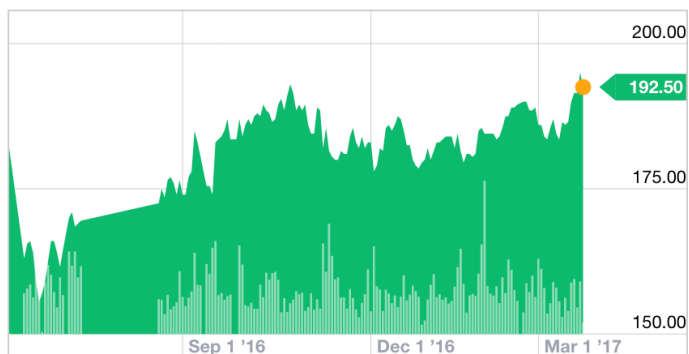
Holders

Historical Data

Analysts

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

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



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

Yahoo Finance Charts

TSMC (2330.TW)

YAHOO! FINANCE

Go to Quote Summary Page



S&P 500

2,344.02

-29.45 (-1.24%)



Dow 30

20,668.01

-237.85 (-1.14%)



Nasdaq

5,793.83

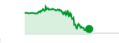
-107.70 (-1.83%)



Crude Oil

47.50

+0.16 (+0.34%)



Gold

1,245.00

-1.50 (-0.12%)



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

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



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

```

import pandas as pd
import pandas_datareader.data as web
df = web.DataReader('AAPL', data_source='yahoo',
start='1/1/2010', end='3/21/2017')
df.to_csv('AAPL.csv')
df.tail()

```

```

import pandas as pd
import pandas_datareader.data as web
#df = web.DataReader('AAPL', 'yahoo')
df = web.DataReader('AAPL', data_source='yahoo', start='1/1/2010', end='3/21/2017')
#df = web.DataReader('AAPL', data_source='google', start='1/1/2010', end='3/21/2017')
df.to_csv('AAPL.csv')
df.tail()

```

	Open	High	Low	Close	Volume	Adj Close
Date						
2017-03-15	139.410004	140.750000	139.029999	140.460007	25566800	140.460007
2017-03-16	140.720001	141.020004	140.259995	140.690002	19132500	140.690002
2017-03-17	141.000000	141.000000	139.889999	139.990005	43597400	139.990005
2017-03-20	140.399994	141.500000	140.229996	141.460007	20213100	141.460007
2017-03-21	142.110001	142.800003	139.729996	139.839996	39116800	139.839996

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

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

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

df.tail(10)

```
df.tail(10)
```

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

df.count()

```
df.count()
```

```
Open          3169  
High          3169  
Low           3169  
Close         3169  
Volume        3169  
Adj Close     3169  
dtype: int64
```

df.ix['2015-12-31']

```
df.ix['2015-12-31']
```

```
Open          7.695000e+02
High          7.695000e+02
Low           7.583400e+02
Close         7.588800e+02
Volume        1.489600e+06
Adj Close     7.588800e+02
Name: 2015-12-31 00:00:00, dtype: float64
```



```
df.to_csv('2330.TW.Yahoo.Finance.Data.csv')
```

2330.TW.Yahoo.Finance.Data.csv ×

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

Python Pandas for Finance

Python Pandas for Finance

```
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
```

```
#Python for Stocks: 1
#Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/
#Import Packages Required
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
```

Python Pandas for Finance

```
#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2015, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

```
#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2015, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2017-03-15	139.410004	140.750000	139.029999	140.460007	25566800	140.460007
2017-03-16	140.720001	141.020004	140.259995	140.690002	19132500	140.690002
2017-03-17	141.000000	141.000000	139.889999	139.990005	43597400	139.990005
2017-03-20	140.399994	141.500000	140.229996	141.460007	20213100	141.460007
2017-03-21	142.110001	142.800003	139.729996	139.839996	39116800	139.839996

Finance Data from Quandl

```
import quandl
```

```
df = quandl.get("WIKI/AAPL", start_date="2015-01-01", end_date="2017-10-31")  
df.to_csv('AAPL.csv')  
df.from_csv('AAPL.csv')  
df.tail()
```

```
import quandl  
df = quandl.get("WIKI/AAPL", start_date="2015-01-01", end_date="2017-10-31")  
df.to_csv('AAPL.csv')  
df.from_csv('AAPL.csv')  
df.tail()
```

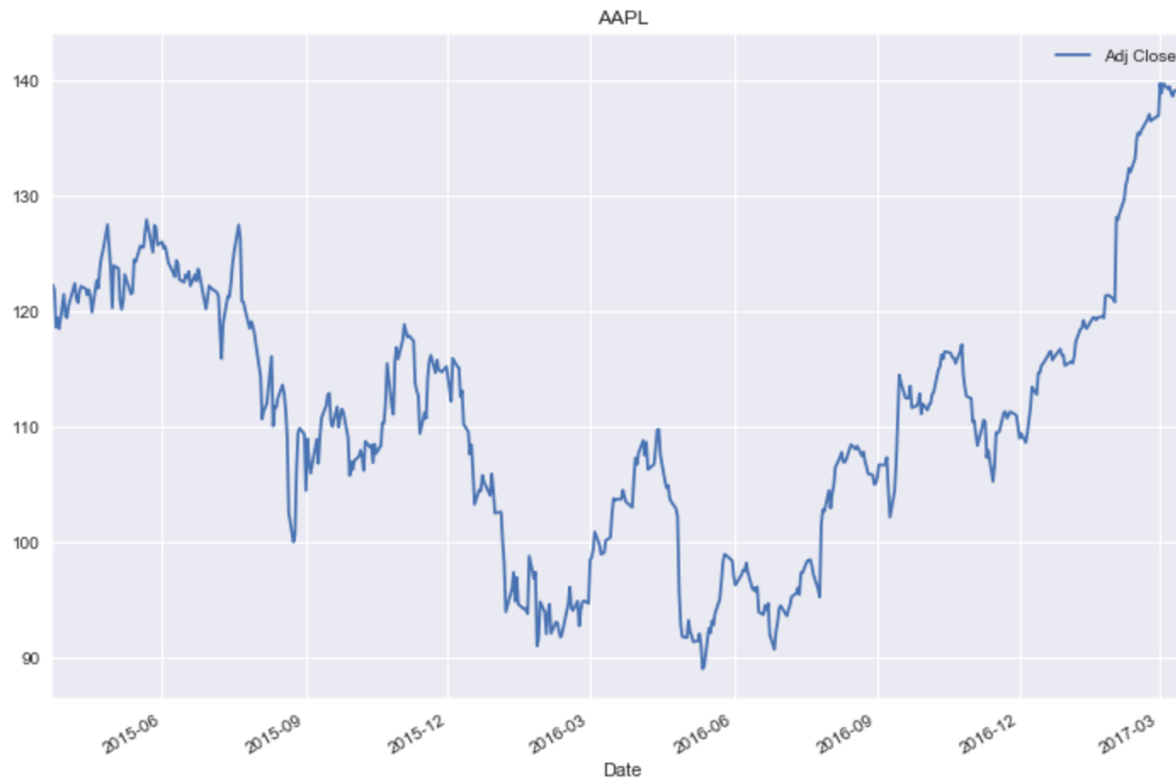
	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
Date												
2017-10-25	156.91	157.5500	155.27	156.405	20126554.0	0.0	1.0	156.91	157.5500	155.27	156.405	20126554.0
2017-10-26	157.23	157.8295	156.78	157.410	16751691.0	0.0	1.0	157.23	157.8295	156.78	157.410	16751691.0
2017-10-27	159.29	163.6000	158.70	163.050	43904150.0	0.0	1.0	159.29	163.6000	158.70	163.050	43904150.0
2017-10-30	163.89	168.0700	163.72	166.720	43923292.0	0.0	1.0	163.89	168.0700	163.72	166.720	43923292.0
2017-10-31	167.90	169.6499	166.94	169.040	35474672.0	0.0	1.0	167.90	169.6499	166.94	169.040	35474672.0

Python Pandas for Finance

```
df['Adj Close'].plot(legend=True,  
figsize=(12, 8), title='AAPL', label='Adj  
Close')
```

```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1150bac88>
```



Python Pandas for Finance

```
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')
```

Python Pandas for Finance

<matplotlib.text.Text at 0x115630860>



Python Pandas for Finance

```
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')
```



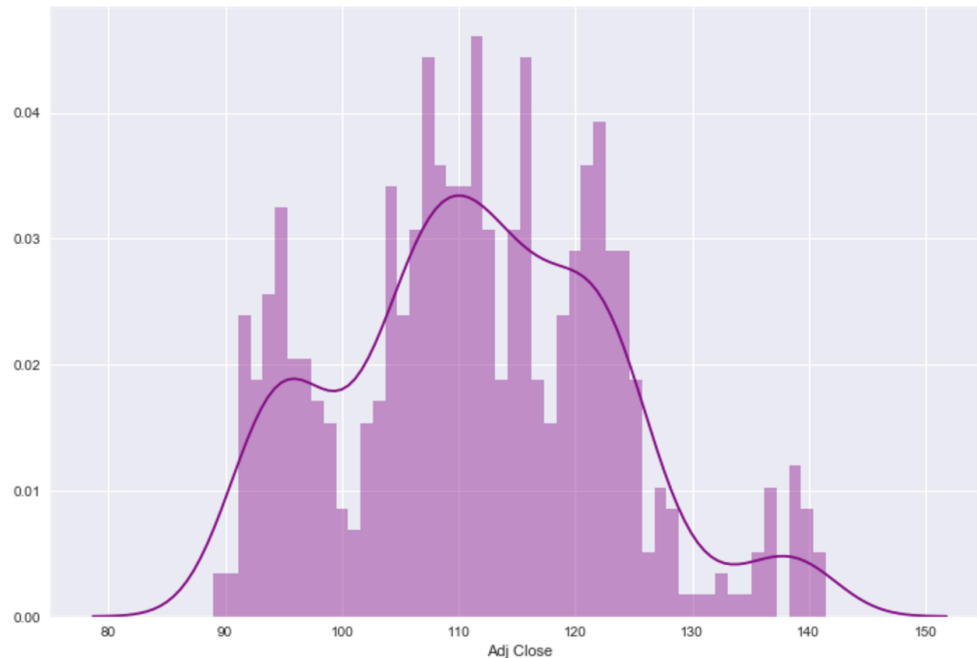
Python Pandas for Finance

```
plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(),
bins=50, color='purple')
```

```
plt.figure(figsize=(12,8))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')
```

```
/Users/imyday/anaconda/lib/python3.6/site-packages/statsmodels/nonparametric/kdetools
g: using a non-integer number instead of an integer will result in an error in the fu
y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x116309780>
```

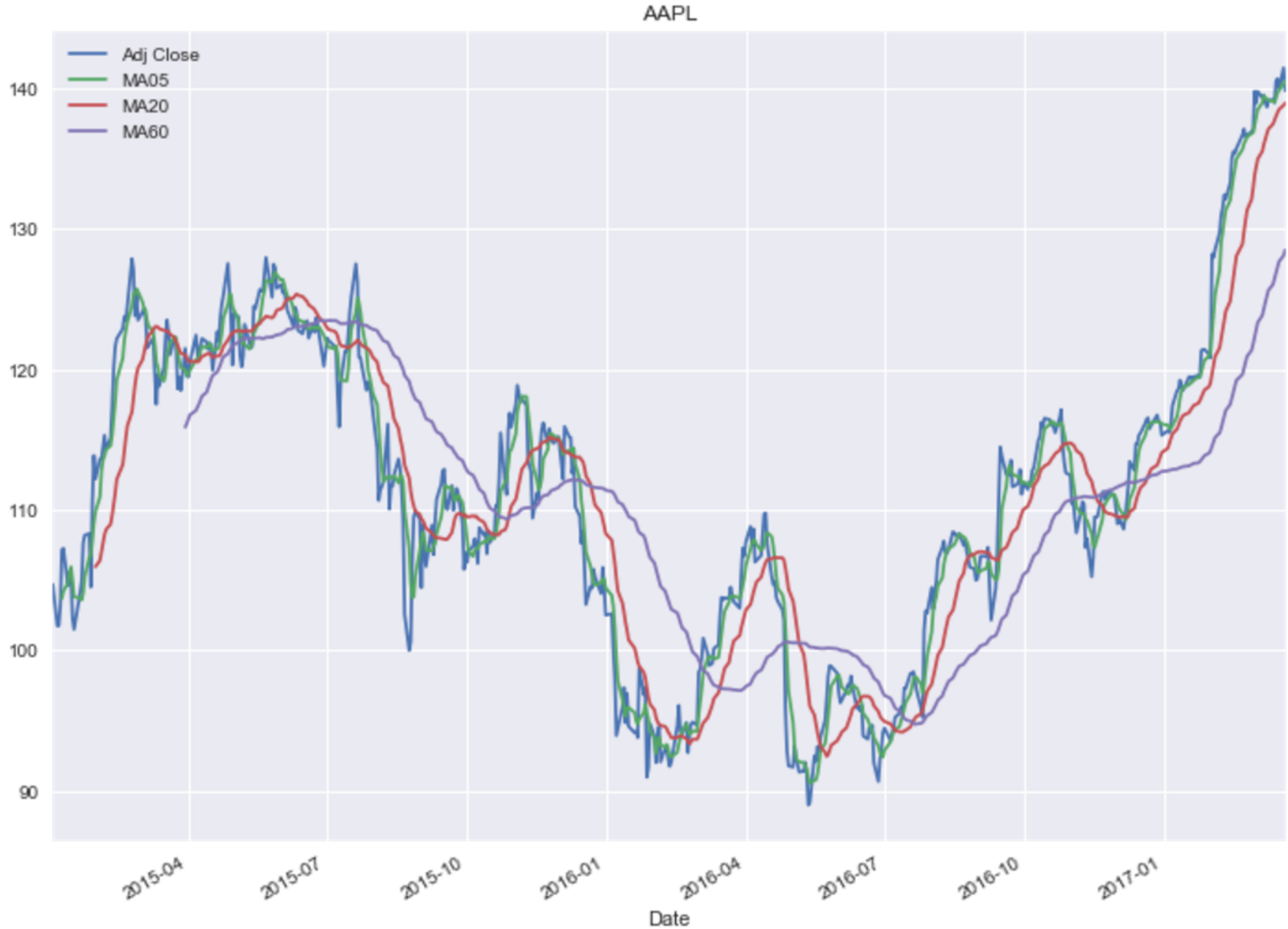


Python Pandas for Finance

```
# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean()
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 Pandas for Finance



Python Pandas for Finance

```
# 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)
```



```

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(2015, 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()

```

Examples: Python Pandas for Finance

```
In [11]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import pandas.io.data as web
import random

import datetime
import time
import timeit

import io
import os

import re
import codecs
import requests
get_ipython().magic('matplotlib inline')

from scipy import stats

#pd.set_option('display.notebook_repr_html', False)
pd.set_option('display.max_columns', 15)
pd.set_option('display.max_rows', 10)
pd.set_option('precision', 3)

def getDateTimeNow():
    strnow = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
    return strnow

print('Hello Pandas')
print(getDateTimeNow())
```

```
Hello Pandas
20160323_145708
```



```

sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"

# sURL = "http://ichart.finance.yahoo.com/table.csv?s=AAPL"
# sBaseURL = "http://ichart.finance.yahoo.com/table.csv?s="
sURL = "http://ichart.finance.yahoo.com/table.csv?s=" + sSymbol
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=2330.TW")
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=AAPL")
req = requests.get(sURL)

sText = req.text
#print(sText)
#df = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
#df = web.DataReader("2330.TW", 'yahoo')

sPath = "data/"
sPathFilename = sPath + sSymbol + ".csv"
print(sPathFilename)

f = open(sPathFilename, 'w')
f.write(sText)
f.close()
sIOdata = io.StringIO(sText)
df = pd.DataFrame.from_csv(sIOdata)
df.head(5)

```

```

In [13]: starttime = datetime.datetime(2000, 1, 1)
          endtime = datetime.datetime(2015, 12, 31)
          sSymbol = "AAPL"
          #sSymbol = "GOOG"
          #sSymbol = "IBM"
          #sSymbol = "MSFT"
          #sSymbol = "^TWII"
          #sSymbol = "000001.SS"
          #sSymbol = "2330.TW"
          #sSymbol = "2317.TW"
          #"^TWII"
          #"AAPL"
          #SHA:000016"
          #"600000.SS"
          #"2330.TW"
          df = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
          #df_01 = web.DataReader("2330.TW", 'yahoo')
          sSymbol = sSymbol.replace(":", "_")
          sSymbol = sSymbol.replace("^", "_")
          sPath = "data/financedata/"
          #sPath = "/users/imyday/SCDBA/data/" #Mac OS X
          #sPath = "C:\data\" #Windows

          sPathFilename = sPath + sSymbol + "_Yahoo_Finance.csv"
          print(sPathFilename)
          df.to_csv(sPathFilename)
          df.head(5)

```

data/financedata/AAPL_Yahoo_Finance.csv

Out[13]:

	Open	High	Low	Close	Volume	Adj Close
Date						
2000-01-03	104.875	112.500	101.688	111.938	133949200	3.702
2000-01-04	108.250	110.625	101.188	102.500	128094400	3.390

df.tail(5)

```
In [14]: df.tail(5)
```

```
Out[14]:
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2015-12-24	109.00	109.00	107.95	108.03	13596700	107.447
2015-12-28	107.59	107.69	106.18	106.82	26704200	106.243
2015-12-29	106.96	109.43	106.86	108.74	30931200	108.153
2015-12-30	108.58	108.70	107.18	107.32	25213800	106.741
2015-12-31	107.01	107.03	104.82	105.26	40912300	104.692

```
sSymbol = "AAPL"

# sURL = "http://ichart.finance.yahoo.com/table.csv?s=AAPL"
sURL = "http://ichart.finance.yahoo.com/table.csv?s=" + sSymbol
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=AAPL")
req = requests.get(sURL)

sText = req.text
#print(sText)

sPath = "data/"
sPathFilename = sPath + sSymbol + ".csv"
print(sPathFilename)

f = open(sPathFilename, 'w')
f.write(sText)
f.close()
sIOdata = io.StringIO(sText)
df = pd.DataFrame.from_csv(sIOdata)
df.head(5)
```

```
In [15]: sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"

# sURL = "http://ichart.finance.yahoo.com/table.csv?s=AAPL"
# sBaseURL = "http://ichart.finance.yahoo.com/table.csv?s="
sURL = "http://ichart.finance.yahoo.com/table.csv?s=" + sSymbol
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=2330.TW")
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=AAPL")
req = requests.get(sURL)

sText = req.text
#print(sText)
#df = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
#df = web.DataReader("2330.TW", 'yahoo')

sPath = "data/"
sPathFilename = sPath + sSymbol + ".csv"
print(sPathFilename)

f = open(sPathFilename, 'w')
f.write(sText)
f.close()
sIOdata = io.StringIO(sText)
df = pd.DataFrame.from_csv(sIOdata)
df.head(5)
```

data/AAPL.csv

Out[15]:

	Open	High	Low	Close	Volume	Adj Close
Date						
2016-03-22	105.25	107.29	105.21	106.72	32232600	106.72
2016-03-21	105.93	107.65	105.14	105.91	35180800	105.91
2016-03-18	106.34	106.50	105.19	105.92	43402300	105.92
2016-03-17	105.52	106.47	104.96	105.80	34244600	105.80
2016-03-16	104.61	106.31	104.59	105.97	37893800	105.97

```

def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
    #GetMarketFinanceData_From_YahooFinance
    # "^TWII"
    # "000001.SS"
    # "AAPL"
    # "SHA:000016"
    # "600000.SS"
    # "2330.TW"
    # sSymbol = "^TWII"
    starttime = datetime.datetime(2000, 1, 1)
    endtime = datetime.datetime(2015, 12, 31)
    sPath = sDir
    # sPath = "data/financedata/"
    df_YahooFinance = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
    # df_01 = web.DataReader("2330.TW", 'yahoo')
    sSymbol = sSymbol.replace(":", "_")
    sSymbol = sSymbol.replace("^", "_")
    sPathFilename = sPath + sSymbol + "_Yahoo_Finance.csv"
    df_YahooFinance.to_csv(sPathFilename)
    # df_YahooFinance.head(5)
    return sPathFilename
# End def getYahooFinanceData(sSymbol, starttime, endtime, sDir):

```

```
In [16]: def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
#GetMarketFinanceData_From_YahooFinance
#"^TWII"
#"000001.SS"
#"AAPL"
#"SHA:000016"
#"600000.SS"
#"2330.TW"
#sSymbol = "^TWII"
starttime = datetime.datetime(2000, 1, 1)
endtime = datetime.datetime(2015, 12, 31)
sPath = sDir
#sPath = "data/financedata/"
df_YahooFinance = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
#df_01 = web.DataReader("2330.TW", 'yahoo')
sSymbol = sSymbol.replace(":", "_")
sSymbol = sSymbol.replace("^", "_")
sPathFilename = sPath + sSymbol + "_Yahoo_Finance.csv"
df_YahooFinance.to_csv(sPathFilename)
#df_YahooFinance.head(5)
return sPathFilename
#End def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
```



```
sSymbol = "AAPL"
starttime = datetime.datetime(2000, 1, 1)
endtime = datetime.datetime(2015, 12, 31)
sDir = "data/financedata/"

sPathFilename = getYahooFinanceData(sSymbol, starttime, endtime,
sDir)
print(sPathFilename)
```

```
In [17]: sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"
starttime = datetime.datetime(2000, 1, 1)
endtime = datetime.datetime(2015, 12, 31)
sDir = "data/financedata/"

sPathFilename = getYahooFinanceData(sSymbol, starttime, endtime, sDir)
print(sPathFilename)
```

```
data/financedata/AAPL_Yahoo_Finance.csv
```

```

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(2015, 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()

```

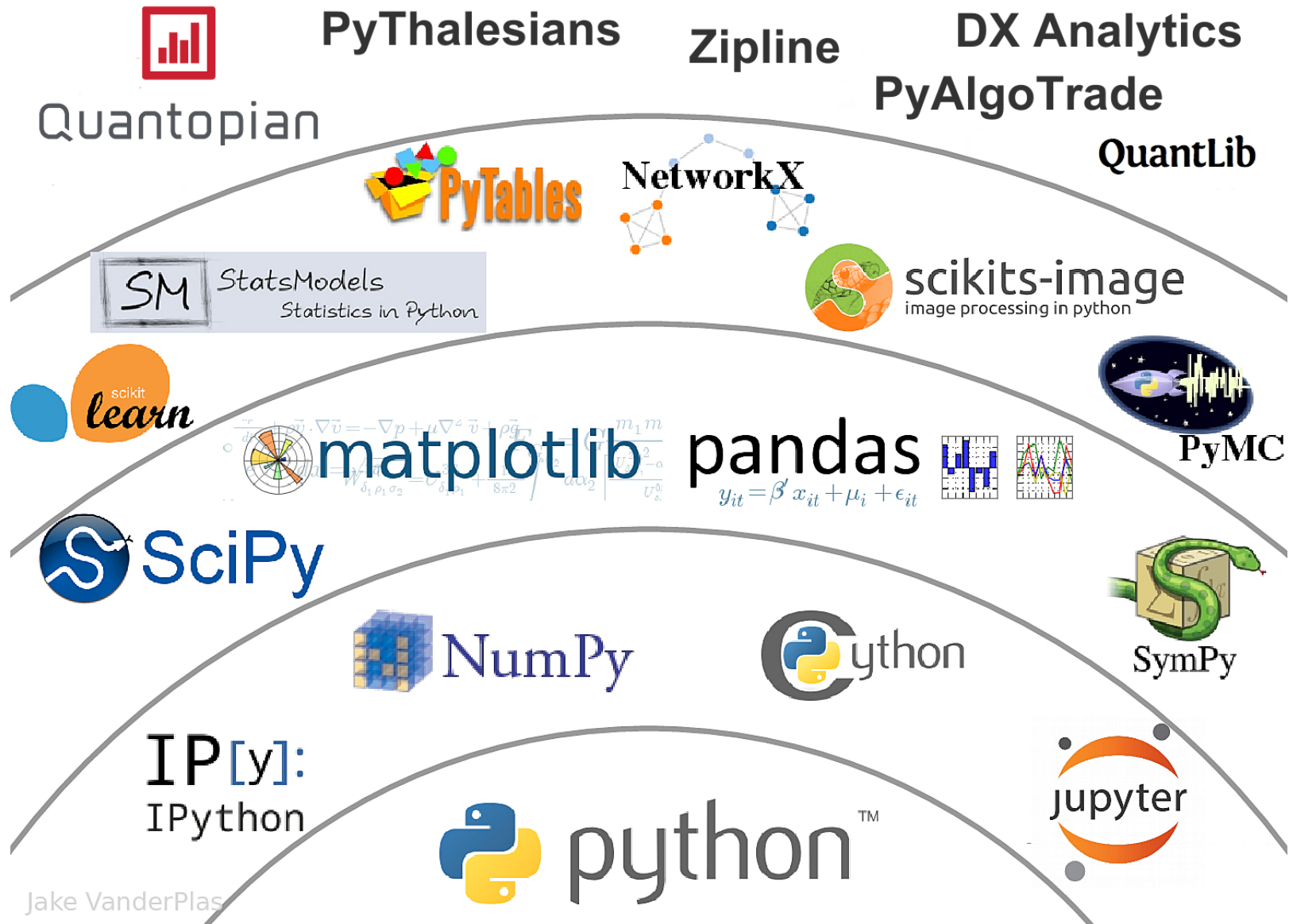
Python Pandas for Finance

```
# 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)
```



The Quant Finance PyData Stack



Jake VanderPlas

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

Quantopian



Investor Relations

Allocations

Research

Community

Learn

Help

Log In

Sign Up

Leveling Wall Street's Playing Field

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

Start Coding



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

References

- 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
- Yves Hilpisch (2015), Derivatives Analytics with Python: Data Analysis, Models, Simulation, Calibration and Hedging, Wiley
- 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
- Wes McKinney (2013), 10-minute tour of pandas, <https://vimeo.com/59324550>
- Jason Wirth (2015), A Visual Guide To Pandas, <https://www.youtube.com/watch?v=9d5-Ti6onew>
- 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/>

References

- Wes McKinney (2017), "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython", 2nd Edition, O'Reilly Media.
<https://github.com/wesm/pydata-book>
- Avinash Jain (2017), Introduction To Python Programming, Udemy,
<https://www.udemy.com/pythonforbeginnersintro/>
- 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>
- Ivan Idris (2015), Numpy Beginner's Guide, Third Edition, Packt Publishing
- Numpy Tutorial, <https://docs.scipy.org/doc/numpy-dev/user/quickstart.html>