

人工智慧財務金融應用



Tamkang
Universit
淡江大學

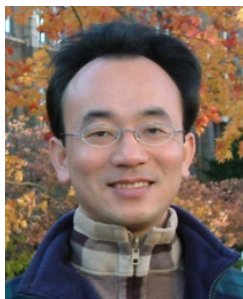
AI in Financial Application

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

戴敏育

Associate Professor

副教授

Dept. of Information Management, Tamkang University

淡江大學 資訊管理學系

<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)

課程大綱 (Syllabus)

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

課程大綱 (Syllabus)

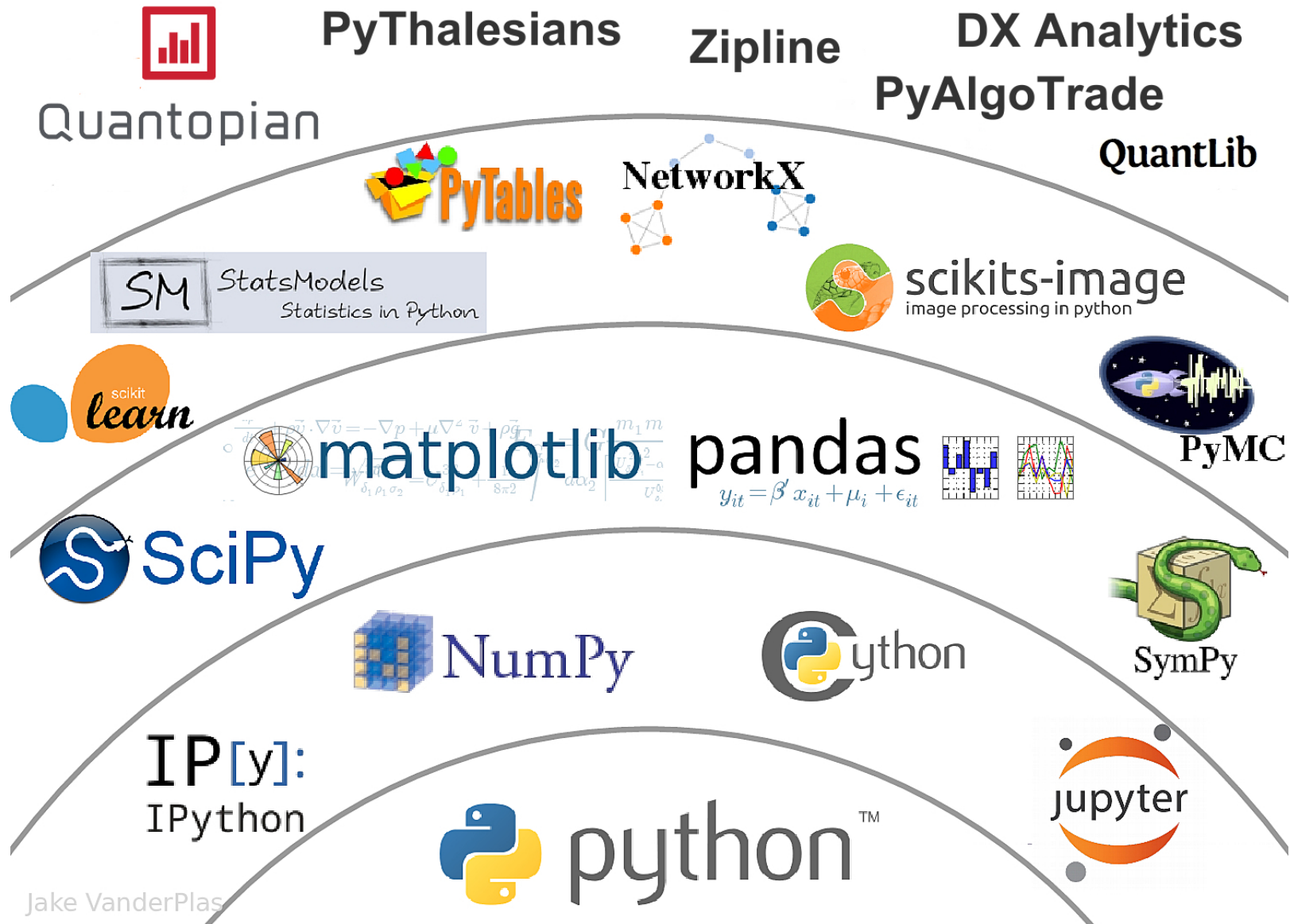
週次 (Week)	日期 (Date)	內容 (Subject/Topics)
13	2019/12/06	人工智慧財務金融應用個案研究 II (Case Study on AI in Financial Application II)
14	2019/12/13	TensorFlow 深度學習財務金融應用 II (Deep Learning for Finance Application with TensorFlow II)
15	2019/12/20	TensorFlow 深度學習財務金融應用 III (Deep Learning for Finance Application with 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



AAPL





Python

Python is an
interpreted,
object-oriented,
high-level
programming language
with
dynamic semantics.

Google Colab

The screenshot shows the Google Colaboratory interface in a web browser. The browser's address bar displays the URL <https://colab.research.google.com/notebooks/welcome.ipynb>. The page header includes the Colab logo, the text "Hello, Colaboratory", and a menu with options: File, Edit, View, Insert, Runtime, Tools, and Help. On the right side of the header, there is a "SHARE" button and a user profile icon. Below the header, a toolbar contains icons for "CODE", "TEXT", "CELL" (with up and down arrows), and "COPY TO DRIVE". To the right of the toolbar are "CONNECT" and "EDITING" buttons. A left sidebar contains a "Table of contents" section with links to "Getting Started", "Highlighted Features", "TensorFlow execution", "GitHub", "Visualization", "Forms", "Examples", and "Local runtime support". The main content area features a large "Welcome to Colaboratory!" message with the Colab logo and a brief description of the environment. Below this, there is a "Getting Started" section with a list of links: "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", and "Machine Learning Crash Course: Intro to Pandas & First Steps with TensorFlow". A "Highlighted Features" section is partially visible, starting with a "Seedbank" subsection that mentions a place to discover machine learning examples, and a "TensorFlow execution" subsection that explains how to run TensorFlow code in the browser, accompanied by a matrix addition example.

Table of contents

- Getting Started
- Highlighted Features
 - TensorFlow execution
- GitHub
- Visualization
- Forms
- Examples
- Local runtime support

SECTION

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.

$$\begin{bmatrix} 1. & 1. & 1. \end{bmatrix} + \begin{bmatrix} 1. & 2. & 3. \end{bmatrix} = \begin{bmatrix} 2. & 3. & 4. \end{bmatrix}$$

Python in Google Colab

<https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkC>

python101.ipynb - Colaboratory rT

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT?authuser=2#scrollTo=wsh36fLxDKC3

python101.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL

CONNECTED EDITING

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

194.87

```
[11] 1 amount = 100
2 interest = 10 #10% = 0.01 * 10
3 years = 7
4
5 future_value = amount * ((1 + (0.01 * interest)) ** years)
6 print(round(future_value, 2))
```

194.87

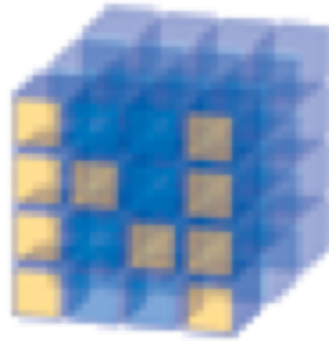
```
[12] 1 # Python Function def
2 def getfv(pv, r, n):
3     fv = pv * ((1 + (r)) ** n)
4     return fv
5 fv = getfv(100, 0.1, 7)
6 print(round(fv, 2))
```

194.87

```
[13] 1 # Python if else
2 score = 80
3 if score >= 60 :
4     print("Pass")
5 else:
6     print("Fail").
```

Pass

NumPy



NumPy
Base

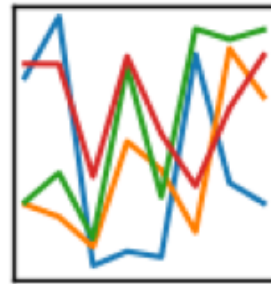
N-dimensional array
package

Python
matplotlib
matplotlib

Python Pandas

pandas

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



Iris flower data set

setosa



versicolor



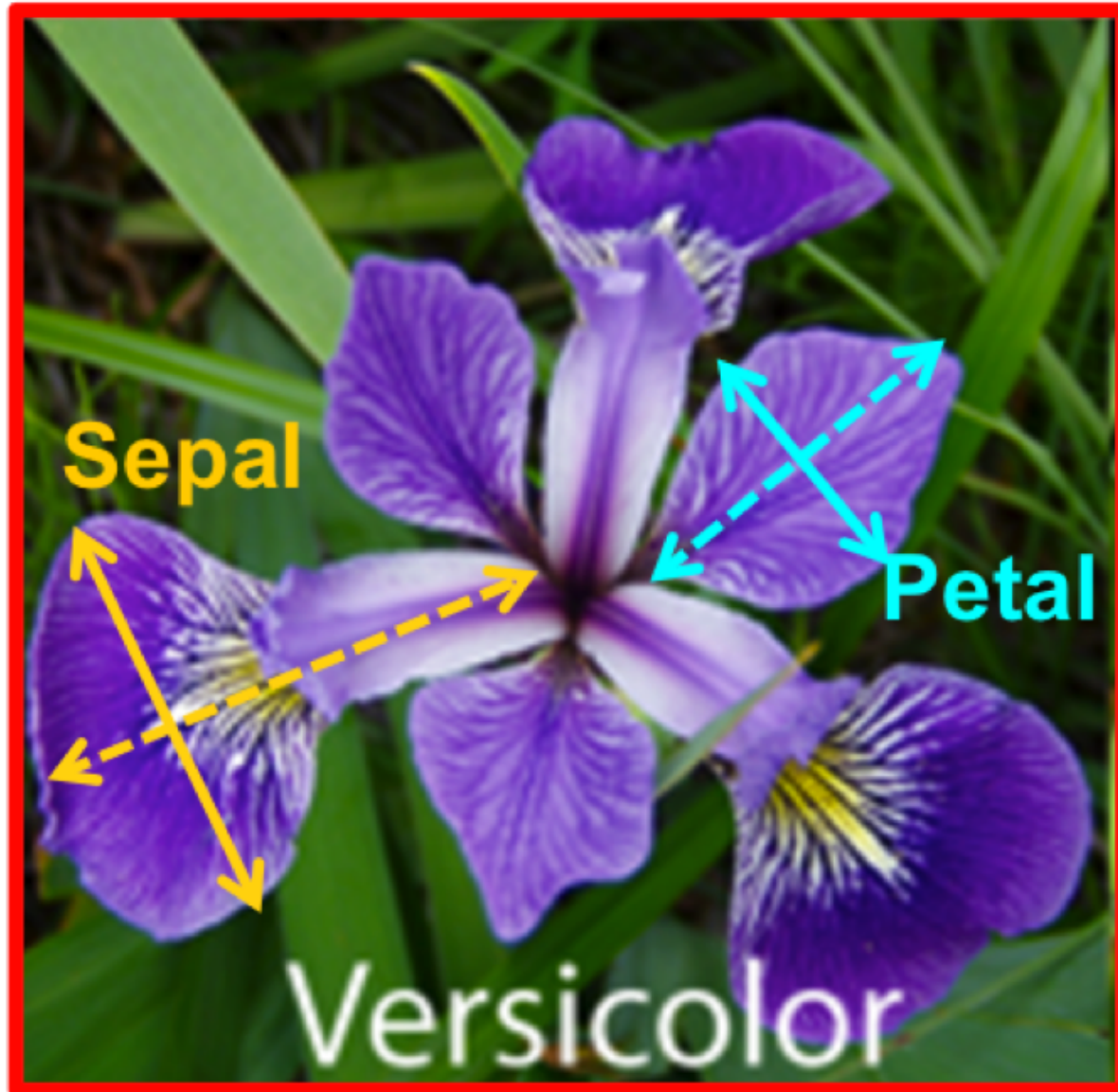
virginica



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

Source: <http://suruchifialoke.com/2016-10-13-machine-learning-tutorial-iris-classification/>

Iris Classification



iris.data

<https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>

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

setosa



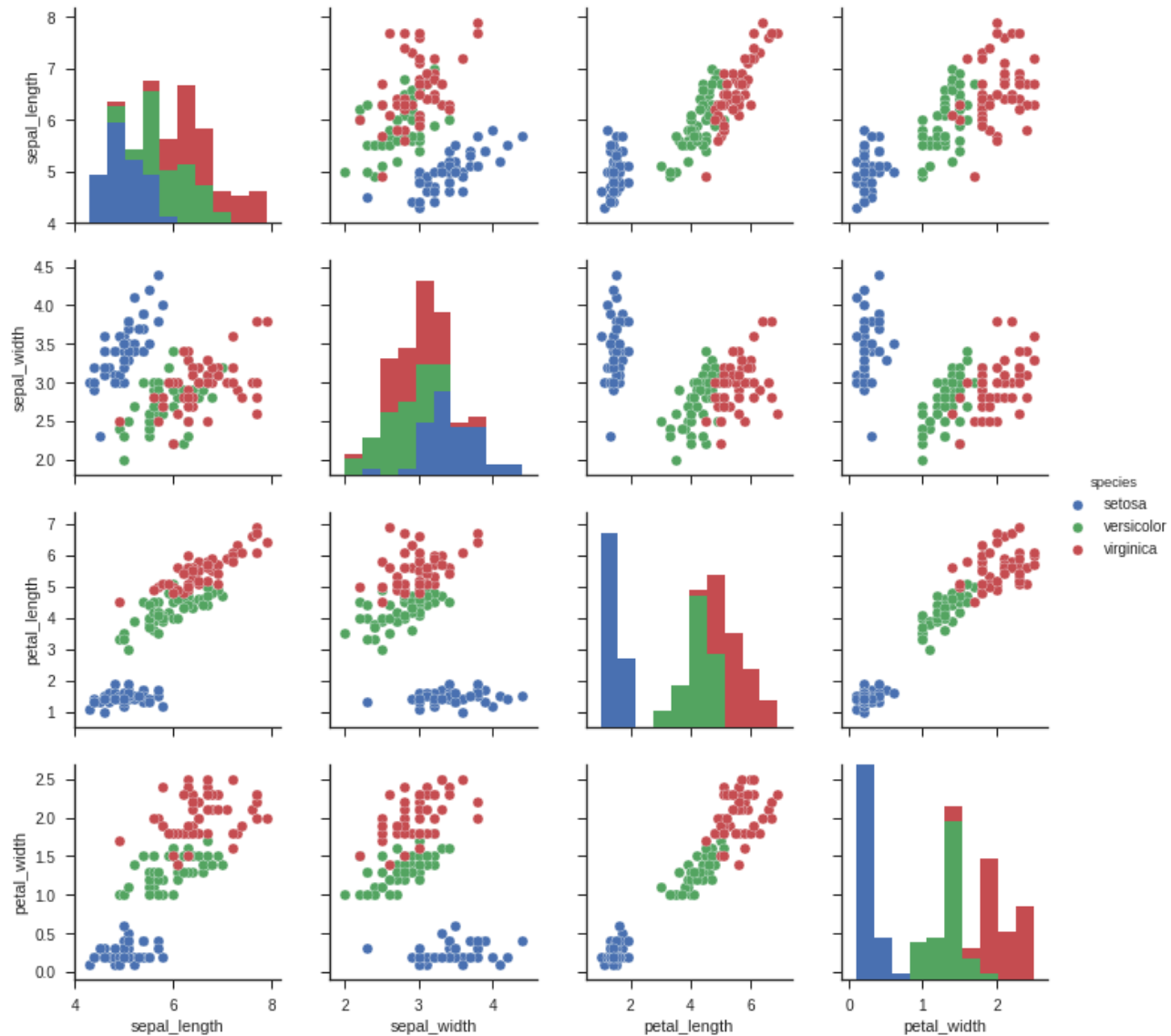
virginica



versicolor



Iris Data Visualization



Connect Google Colab in Google Drive

The screenshot shows the Google Drive web interface. The browser address bar displays `https://drive.google.com/drive/u/2/my-drive`. The Drive logo and a search bar are at the top. On the left sidebar, the 'New' button is highlighted with a red dashed box. A dropdown menu is open, showing options like 'New folder...', 'Upload files...', 'Upload folder...', 'Google Docs', 'Google Sheets', 'Google Slides', and 'More'. The 'More' option is also highlighted with a red dashed box. A second dropdown menu is open from 'More', listing 'Google Forms', 'Google Drawings', 'Google My Maps', 'Google Sites', and a red dashed box around the '+ Connect more apps' option at the bottom. The main area shows 'My Drive' and 'Quick Access' sections. At the bottom, there's a storage usage indicator (0 bytes of 15 GB used) and a 'Get Backup and Sync for Mac' notification.

My Drive - Google Drive

Search Drive

My Drive

Quick Access

New

My Drive

Computers

Shared with me

Recent

Starred

Trash

Backups

Storage

0 bytes of 15 GB used

UPGRADE STORAGE

Get Backup and Sync for Mac

New folder...

Upload files...

Upload folder...

Google Docs

Google Sheets

Google Slides

More

Google Forms

Google Drawings

Google My Maps

Google Sites

+ Connect more apps

Google Colab

My Drive - Google Drive x +

https://drive.google.com/drive/u/2/my-drive

Drive

Search Drive

New

My Drive

Computers

Shared with me

Recent

Starred

Trash

Backups

Storage

0 bytes of 15 GB used

UPGRADE STORAGE

Get Backup and Sync for Mac

Access anywhere

Share easily

Connect apps to Drive

All

colab

**ZIP Extractor**
Extract ZIP files to Google Drive
Extraction complete.
[View extracted files](#) [Share](#) [Extract another](#)
 **Test.zip**
ZIP Extractor
307,585 users

**LUMIN PDF**
The fast and simple PDF Viewer
  
Lumin PDF - Beautiful PDF Editor
289,310 users

**cloudconvert**
CloudConvert
373,161 users

**Sejda**
Merge PDF - Split PDF - Sejda.com
★★★★★ (1106)

**DocHub**
Edit, Send & Sign PDFs
DocHub - Edit and Sign PDF Docu...
2,131,600 users

**Google Forms**
Google Forms
4,803,614 users

Google Colab

My Drive - Google Drive x +

https://drive.google.com/drive/u/2/my-drive

Drive

Search Drive

New

My Drive

Computers

Shared with me

Recent

Starred

Trash

Backups

Storage

0 bytes of 15 GB used

[UPGRADE STORAGE](#)

Get Backup and Sync for Mac

Access anywhere

Share easily

Connect apps to Drive

All colab



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

[+ CONNECT](#)

Productivity
★★★★★ (195)

Name ↑

Connect Colaboratory to Google Drive

The screenshot shows the Google Drive web interface. A dialog box titled "Connect apps to Drive" is open, displaying a list of apps. The "colab" app is selected, and a confirmation message states: "Colaboratory was connected to Google Drive." Below this message, there is a checked checkbox and the text "Make Colaboratory the default app for files it can open". An "OK" button is visible at the bottom right of the dialog box. The background shows the Google Drive sidebar with options like "New", "My Drive", "Computers", "Shared with me", "Recent", "Starred", "Trash", "Backups", and "Storage". The top of the browser shows the address bar with the URL "https://drive.google.com/drive/u/2/my-drive".

My Drive - Google Drive

Search Drive

Connect apps to Drive

All

colab

Colaboratory was connected to Google Drive.

☒ Make Colaboratory the default app for files it can open

OK

Get Backup and Sync for Mac

Google Colab

The image shows the Google Drive web interface. At the top, the browser tab is labeled 'My Drive - Google Drive' and the address bar shows the URL 'https://drive.google.com/drive/u/2/my-drive'. The Drive logo and a search bar are visible. On the left sidebar, the 'New' button is highlighted with a red dashed box. A dropdown menu is open, showing options like 'New folder...', 'Upload files...', 'Upload folder...', 'Google Docs', 'Google Sheets', 'Google Slides', and 'More'. The 'More' option is also highlighted with a red dashed box. A second dropdown menu is open from 'More', listing 'Google Forms', 'Google Drawings', 'Google My Maps', 'Google Sites', and 'Colaboratory'. The 'Colaboratory' option is highlighted with a red dashed box. At the bottom left, there is a notification for 'Get Backup and Sync for Mac'.

My Drive - Google Drive

https://drive.google.com/drive/u/2/my-drive

Drive

Search Drive

My Drive

Quick Access

New

My Drive

Computers

Shared with me

Recent

Starred

Trash

Backups

Storage

0 bytes of 15 GB used

UPGRADE STORAGE

Get Backup and Sync for Mac

New folder...

Upload files...

Upload folder...

Google Docs

Google Sheets

Google Slides

More

Google Forms

Google Drawings

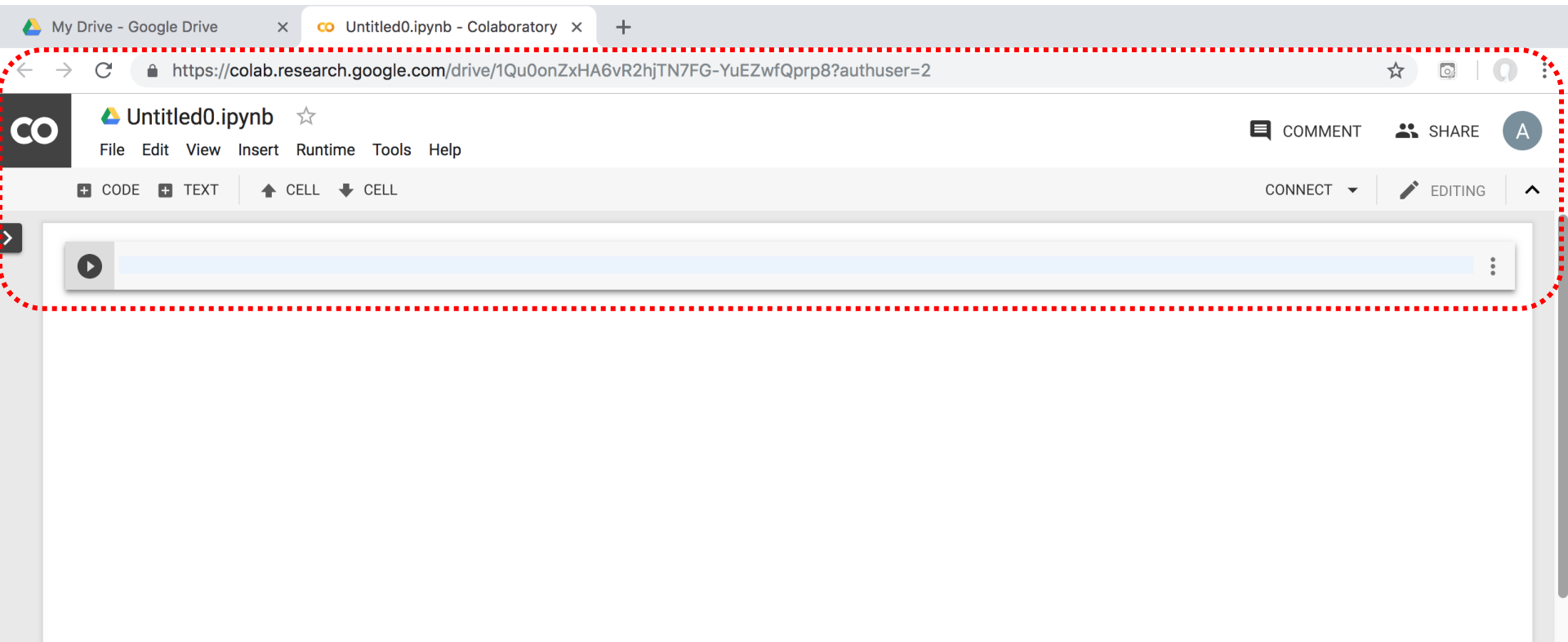
Google My Maps

Google Sites

Colaboratory

Connect more apps

Google Colab



Google Colab

The screenshot displays the Google Colab web interface. At the top, the browser tabs show 'My Drive - Google Drive' and 'Untitled0.ipynb - Colaboratory'. The address bar contains the URL: <https://colab.research.google.com/drive/1Qu0onZxHA6vR2hjTN7FG-YuEZwfQprp8?authuser=2>. The main header area includes the Google Colab logo, the file name 'Untitled0.ipynb', and a star icon. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. The 'Runtime' menu is open, showing a list of options: 'Run all' (⌘/Ctrl+F9), 'Run before' (⌘/Ctrl+F8), 'Run the focused cell' (⌘/Ctrl+Enter), 'Run selection' (⌘/Ctrl+Shift+Enter), 'Run after' (⌘/Ctrl+F10), 'Interrupt execution' (⌘/Ctrl+M I), 'Restart runtime...' (⌘/Ctrl+M .), 'Restart and run all...', 'Reset all runtimes...', 'Change runtime type', and 'Manage sessions'. The 'Runtime' menu item in the top bar and the 'Change runtime type' option in the dropdown are highlighted with red dashed boxes. On the right side of the interface, there are buttons for 'COMMENT', 'SHARE', and a user profile icon. Below these are 'CONNECT' and 'EDITING' buttons. The main workspace area is visible in the background, showing a code editor with a play button icon on the left.

Run Jupyter Notebook Python3 GPU Google Colab

The screenshot shows the Google Colab web interface. The browser tab is titled 'Untitled0.ipynb - Colaboratory'. The address bar shows the URL: <https://colab.research.google.com/drive/1Qu0onZxHA6vR2hjTN7FG-YuEZwfQprp8?authuser=2>. The Colab logo is in the top left. The top navigation bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the right, there are 'COMMENT', 'SHARE', and a user profile icon. Below the navigation bar, there are buttons for '+ CODE', '+ TEXT', '↑ CELL', and '↓ CELL'. On the far right, there are 'CONNECT', 'EDITING', and an upward arrow icon. The main workspace is a large gray area. A 'Notebook settings' dialog box is open in the center. It has a title 'Notebook settings'. Inside, there are two dropdown menus: 'Runtime type' set to 'Python 3' and 'Hardware accelerator' set to 'GPU'. Both dropdowns are highlighted with a red dashed border. Below these, there is a checkbox labeled 'Omit code cell output when saving this notebook' which is currently unchecked. At the bottom right of the dialog are 'CANCEL' and 'SAVE' buttons. A blue question mark icon is next to the 'Hardware accelerator' dropdown.

My Drive - Google Drive x Untitled0.ipynb - Colaboratory x +

← → ↻ <https://colab.research.google.com/drive/1Qu0onZxHA6vR2hjTN7FG-YuEZwfQprp8?authuser=2> ☆ 📷 🔊

co Untitled0.ipynb ☆

File Edit View Insert Runtime Tools Help

+ CODE + TEXT ↑ CELL ↓ CELL

CONNECT ▾ EDITING ^

>

Notebook settings

Runtime type
Python 3 ▾

Hardware accelerator
GPU ▾ ?

☐ Omit code cell output when saving this notebook

CANCEL SAVE

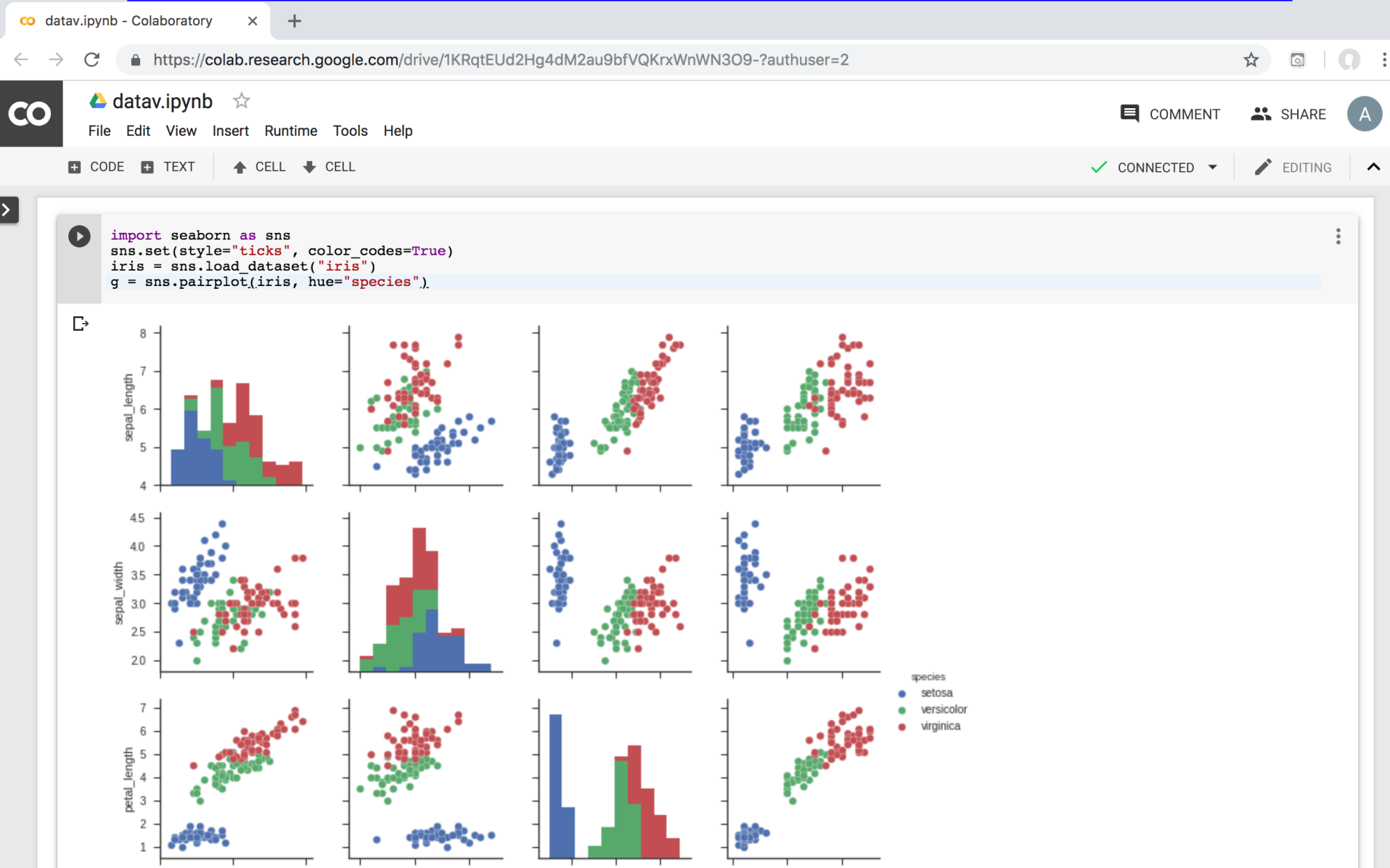
Google Colab Python Hello World

```
print('Hello World')
```



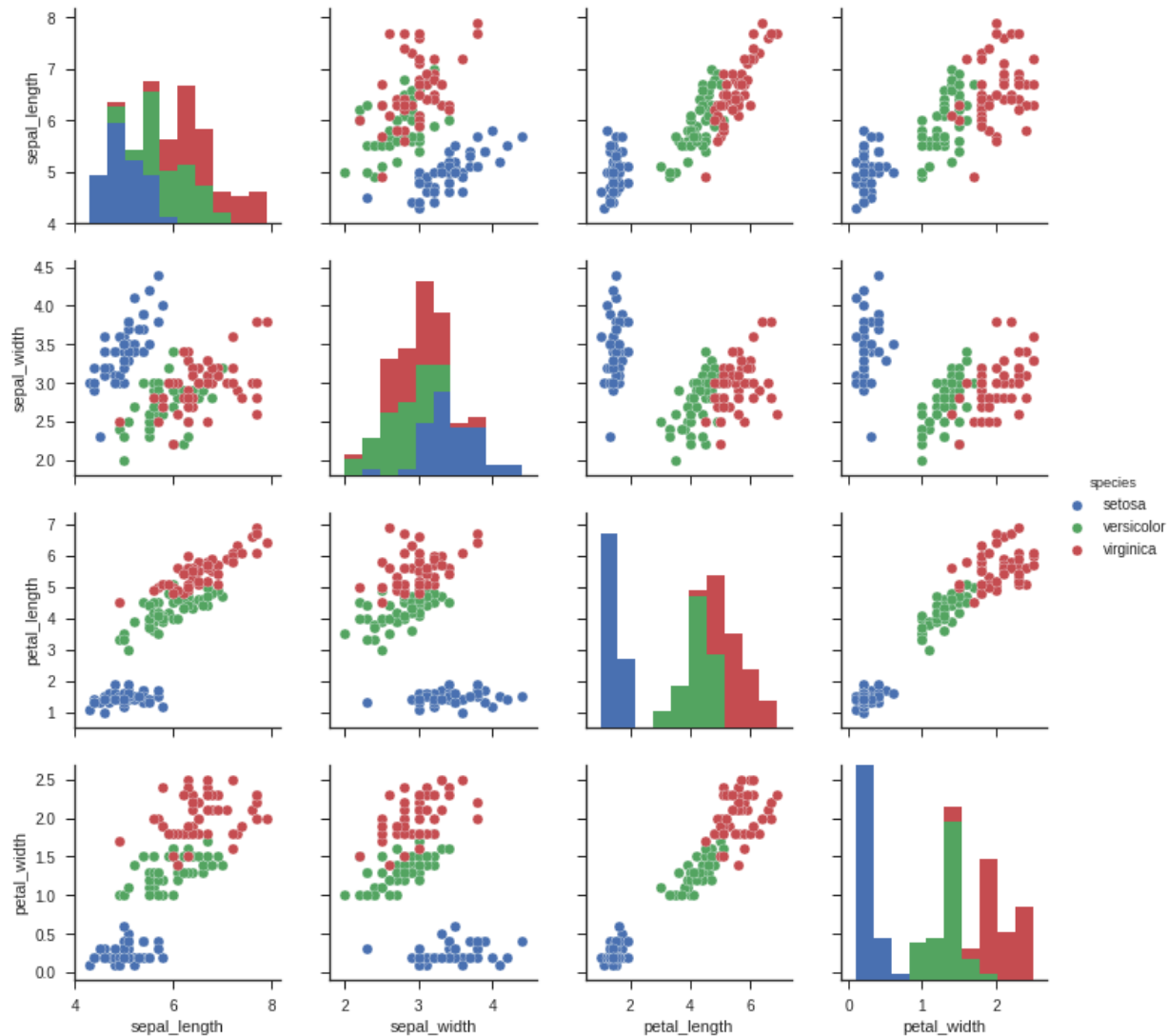
Data Visualization in Google Colab

<https://colab.research.google.com/drive/1KRqtEUd2Hg4dM2au9bfVQKrxWnWN3O9->



Source: <https://seaborn.pydata.org/generated/seaborn.pairplot.html>

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



```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix

# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
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

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)
print(df.head(10))
```

```
# Load dataset
```

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)
print(df.head(10)).
```

	sepal-length	sepal-width	petal-length	petal-width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

df.tail(10)

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

	sepal-length	sepal-width	petal-length	petal-width	class
140	6.7	3.1	5.6	2.4	Iris-virginica
141	6.9	3.1	5.1	2.3	Iris-virginica
142	5.8	2.7	5.1	1.9	Iris-virginica
143	6.8	3.2	5.9	2.3	Iris-virginica
144	6.7	3.3	5.7	2.5	Iris-virginica
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

df.describe()

```
print(df.describe())
```

	sepal-length	sepal-width	petal-length	petal-width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
print(df.info())  
print(df.shape)
```

```
print(df.info())
```

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

```
print(df.shape)
```

```
(150, 5)
```



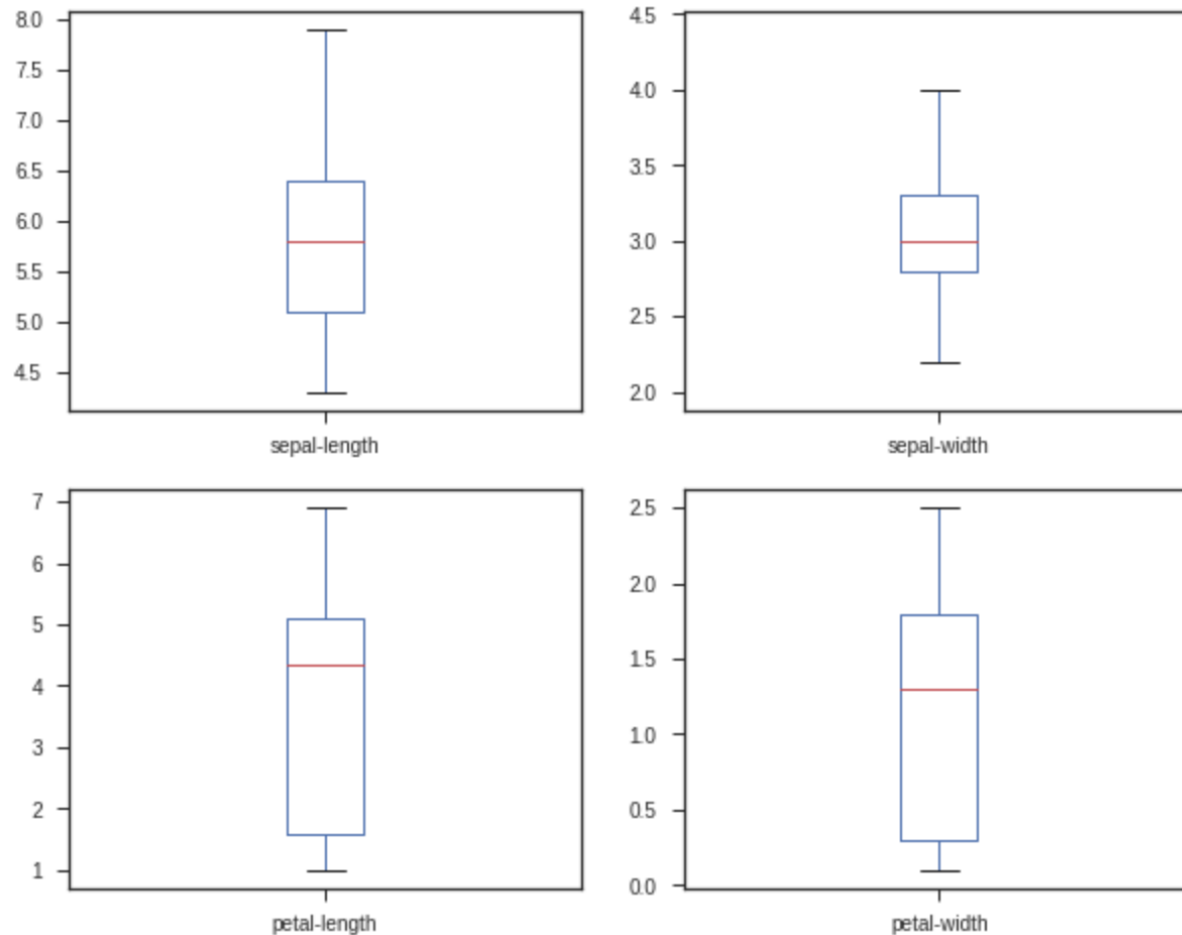
```
df.groupby( 'class' ).size()
```

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

```
class
Iris-setosa      50
Iris-versicolor  50
Iris-virginica   50
dtype: int64
```

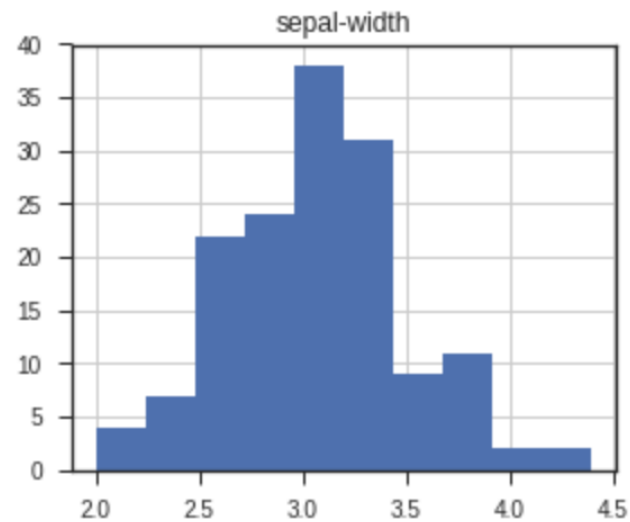
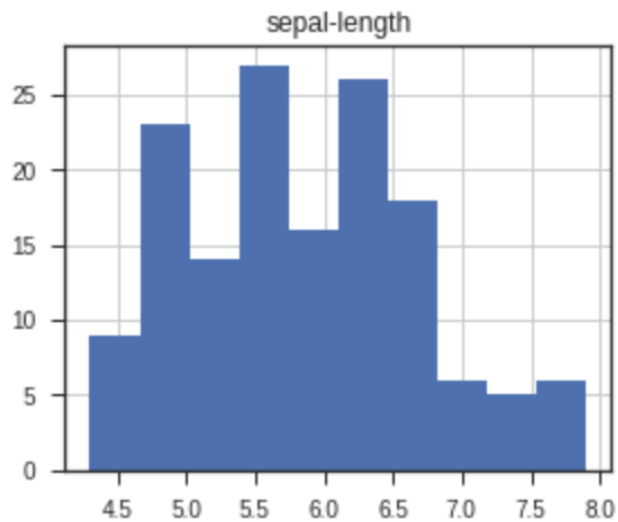
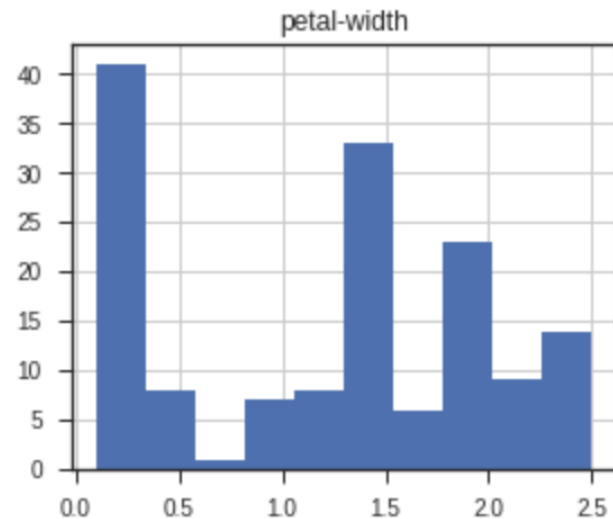
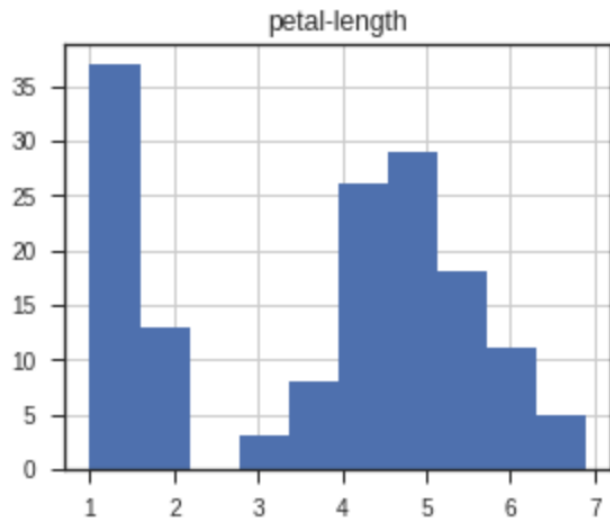
```
plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```

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



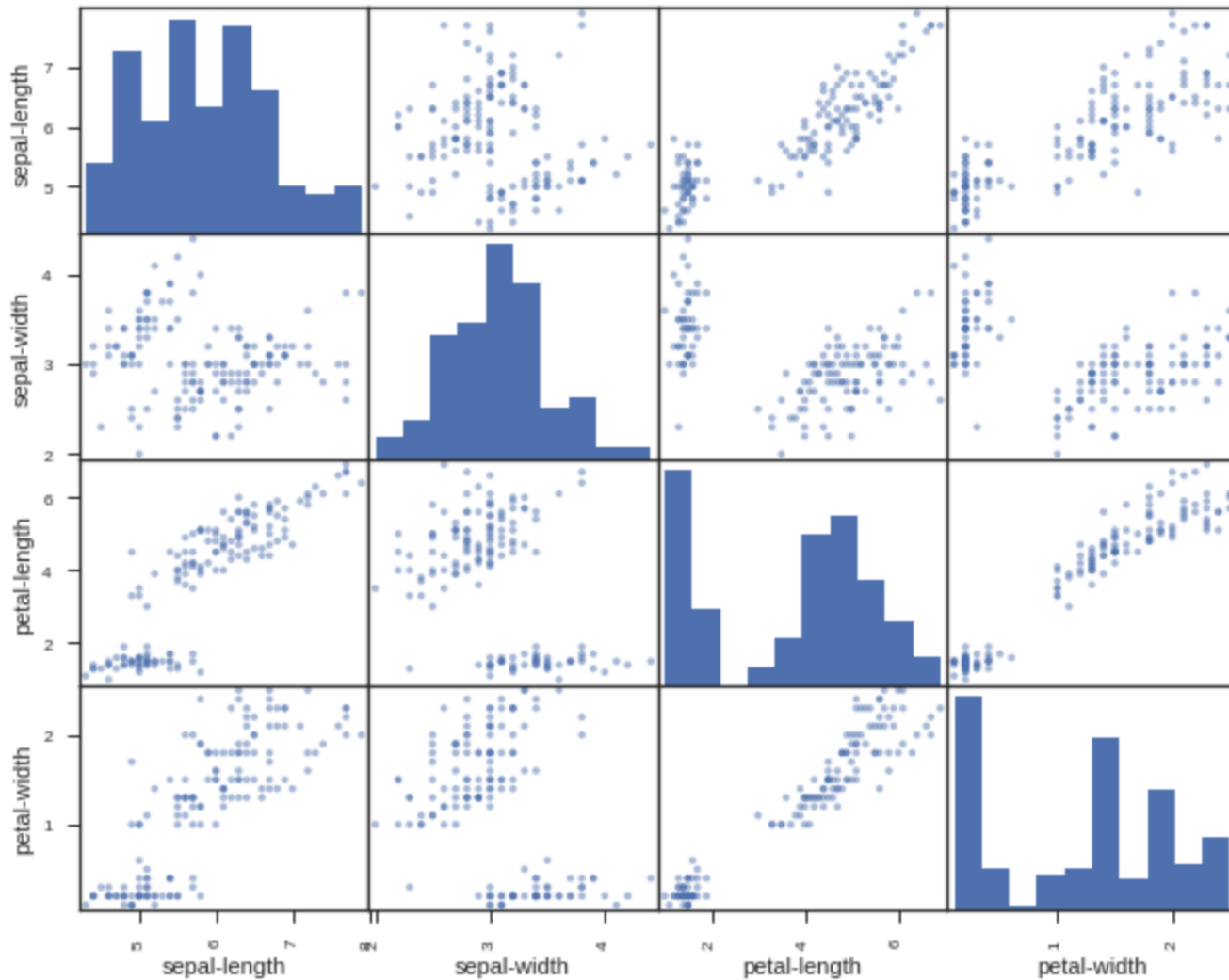
```
df.hist()  
plt.show()
```

```
df.hist()  
plt.show()
```



```
scatter_matrix(df)  
plt.show()
```

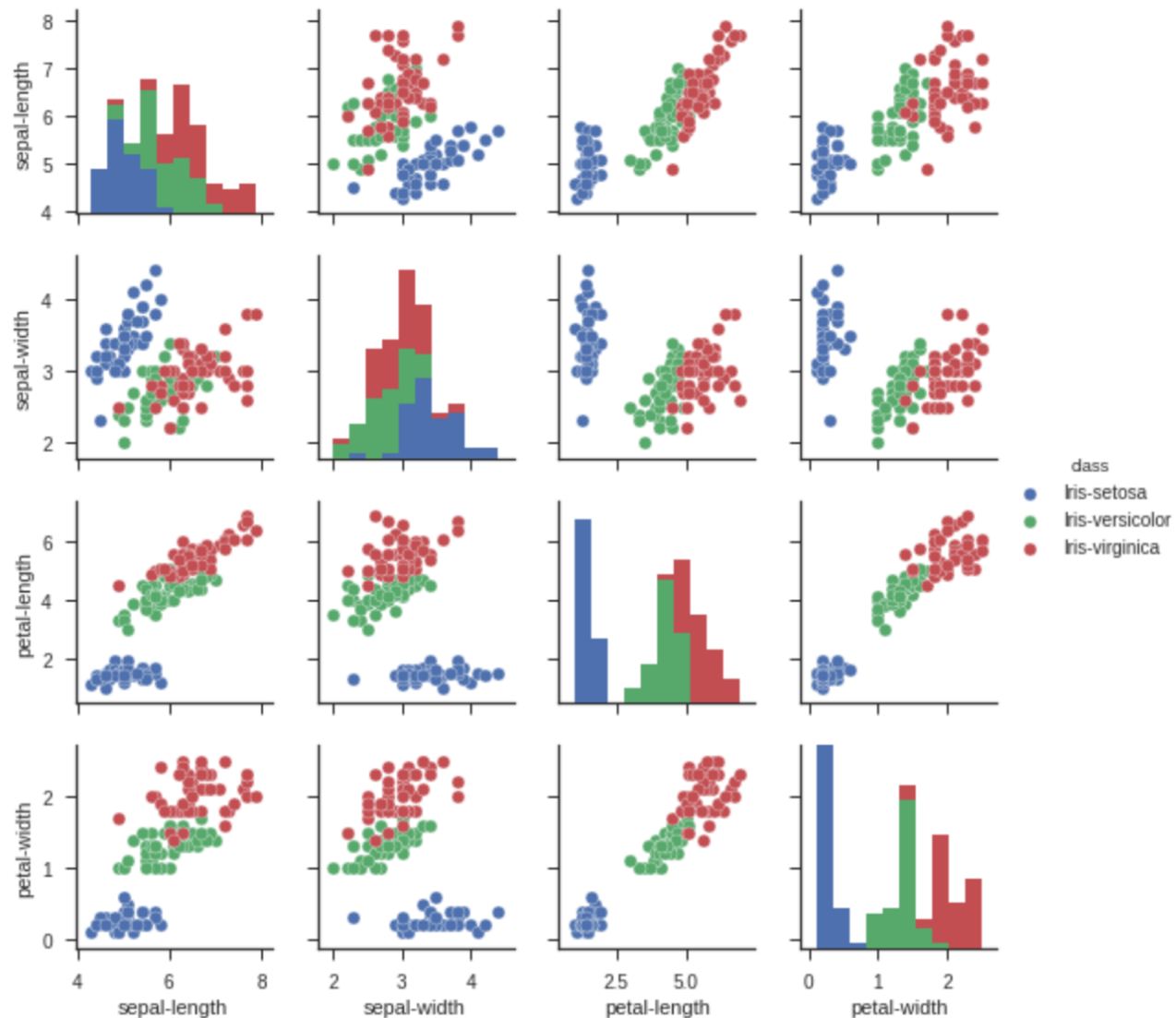
```
scatter_matrix(df)  
plt.show(.)
```



`sns.pairplot(df, hue="class", size=2)`

```
sns.pairplot(df, hue="class", size=2)
```

<seaborn.axisgrid.PairGrid at 0x7f1d21267390>





Anaconda

The Most Popular

Python

Data Science Platform

Download Anaconda



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

Download Anaconda Distribution

Version 5.1 | Release Date: February 15, 2018

Download For:   

High-Performance Distribution

Easily install 1,000+ [data science packages](#)

Package Management

Manage packages, dependencies and environments with [conda](#)

Portal to Data Science

Uncover insights in your data and create interactive visualizations



Anaconda 5.1 For macOS Installer

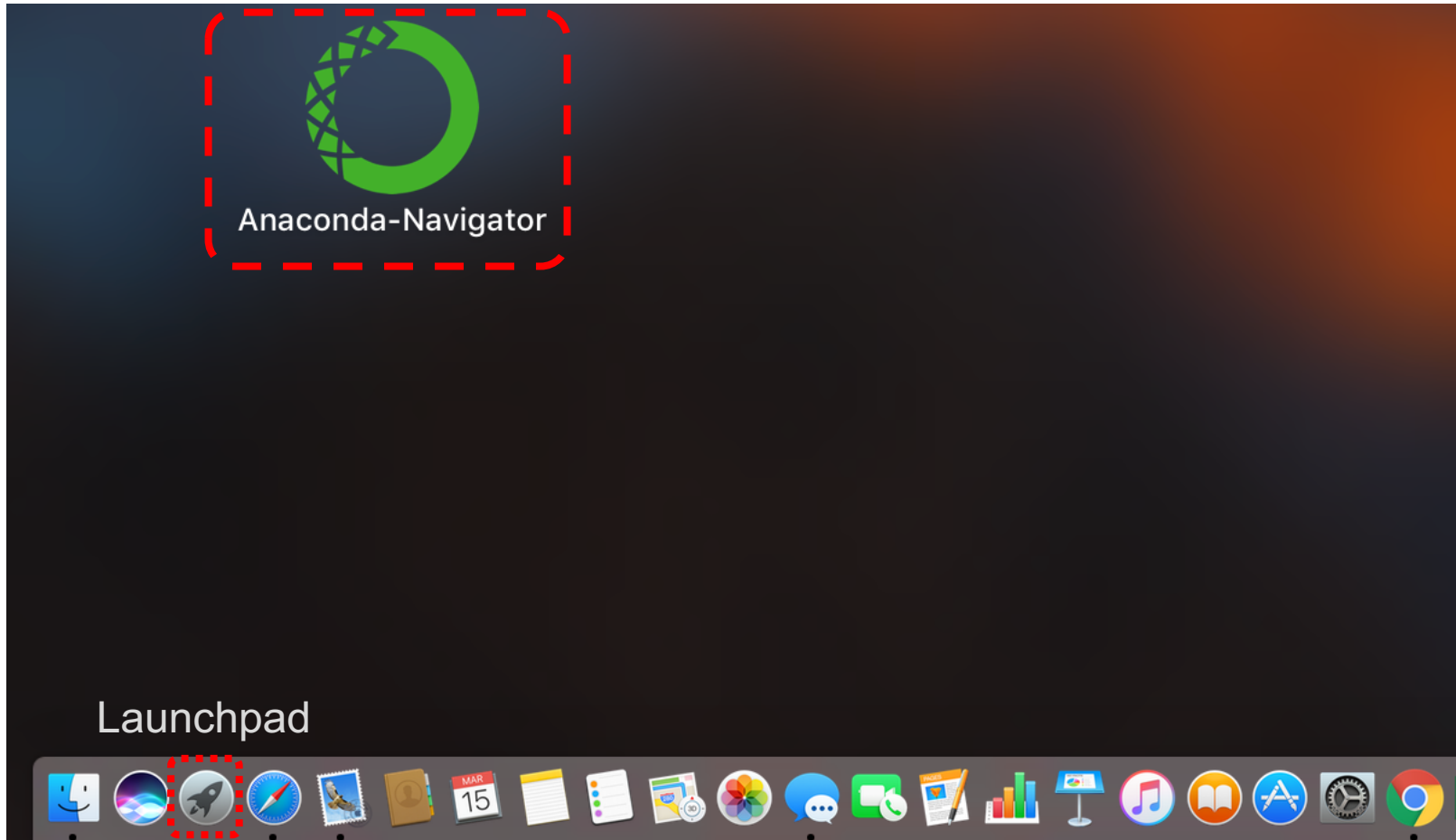
<https://www.anaconda.com/download>



Python

HelloWorld

Anaconda-Navigator



Anaconda Navigator



Home



Environments



Learning



Community

Documentation

Developer Blog

Feedback



Applications on

base (root)

Channels

Refresh



jupyterlab

0.31.5

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Launch



notebook

5.4.0

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Launch



qtconsole

4.3.1

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Launch



spyder

3.2.6

Scientific PYTHON Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Launch

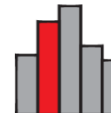


vscode

1.22.2

Streamlined code editor with support for development operations like debugging, task running and version control.

Launch



glueviz

0.12.4

Multidimensional data visualization across files. Explore relationships within and among related datasets.

Install

Jupyter Notebook

The screenshot shows the Jupyter Notebook web interface in a browser window. The address bar displays `localhost:8888/tree/Documents/Data/BDA`. The Jupyter logo and a "Logout" button are in the top right. Below the navigation tabs ("Files", "Running", "Clusters"), there is a message "Select items to perform actions on them." and buttons for "Upload", "New", and a refresh icon. A table lists the files in the current directory, with columns for "Name" and "Last Modified". The table contains one entry: a folder named `..` modified "seconds ago". A red dashed box highlights the first row of the table. Below the table, a message states "The notebook list is empty."

Home

localhost:8888/tree/Documents/Data/BDA

jupyter Logout

Files Running Clusters

Select items to perform actions on them. Upload New ↕ ↻

	Name ↓	Last Modified
☐ 0	/ Documents / Data / BDA	
..		seconds ago

The notebook list is empty.

Jupyter Notebook

New Python 3

The screenshot shows a web browser window with the Jupyter Notebook interface. The browser tabs include 'Home' and 'Untitled'. The address bar shows 'localhost:8888/tree/Documents/Data/BDA'. The Jupyter logo and a 'Logout' button are in the top right. Below the logo are tabs for 'Files', 'Running', and 'Clusters'. A message says 'Select items to perform actions on them.' Below this is a file browser showing the path '/ Documents / Data / BDA'. To the right of the file browser are 'Upload' and 'New' buttons. The 'New' button is highlighted with a red box, and its dropdown menu is open, showing options: 'Notebook:', 'Python 3' (highlighted with a red box), 'Other:', 'Text File', 'Folder', and 'Terminal'.

Home x Untitled x u

localhost:8888/tree/Documents/Data/BDA

jupyter Logout

Files Running Clusters

Select items to perform actions on them.

0 ▾ / Documents / Data / BDA

..

The notebook list is empty.

Upload New ▾

Notebook:

Python 3

Other:

Text File

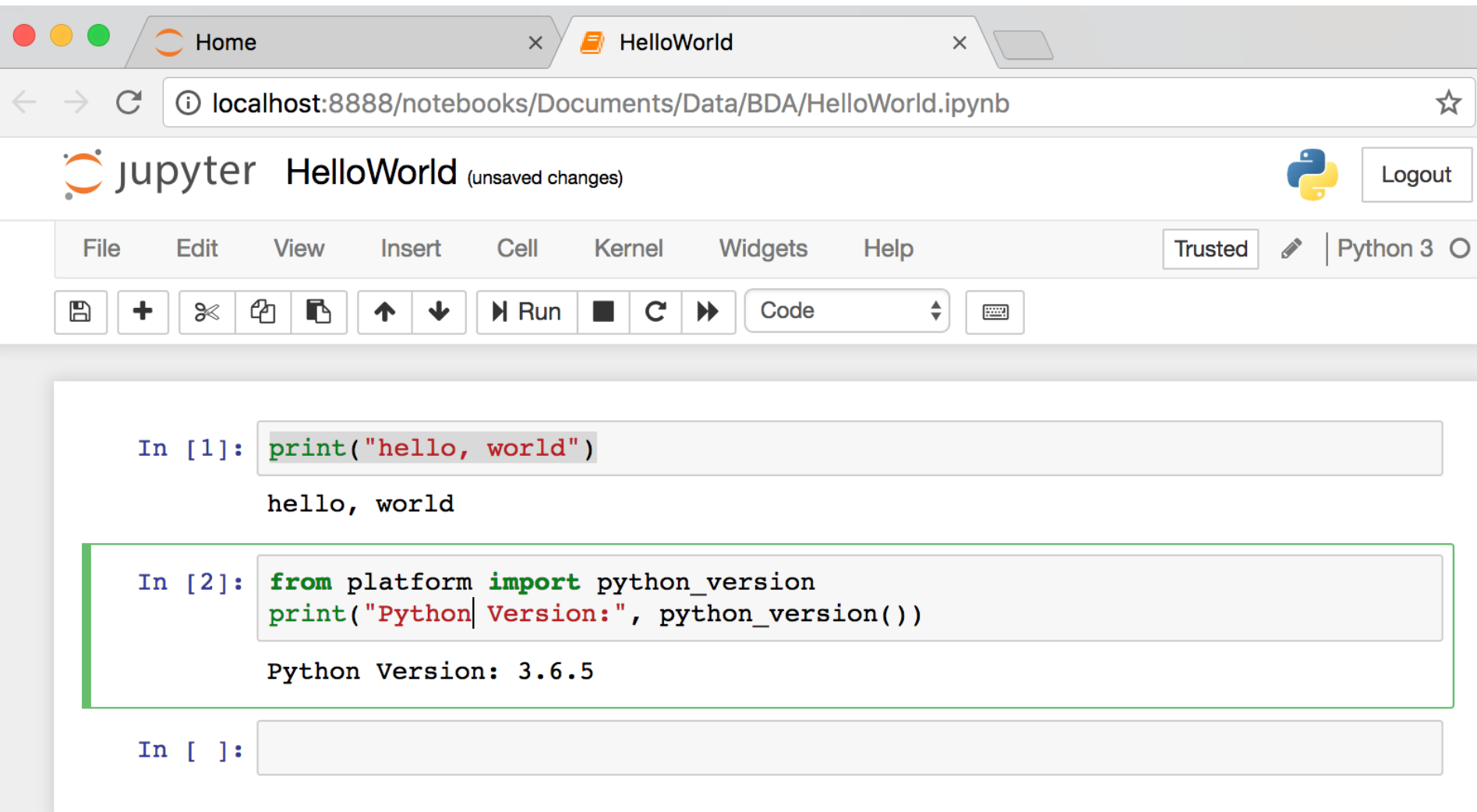
Folder

Terminal

print("hello, world")

The screenshot shows a web browser window with two tabs: 'Home' and 'HelloWorld'. The address bar shows the URL `localhost:8888/notebooks/Documents/Data/BDA/HelloWorld.ipynb`. The Jupyter Notebook interface is displayed, with the title 'jupyter HelloWorld (autosaved)' and a 'Logout' button. The top menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. Below the menu is a toolbar with icons for saving, adding, deleting, and running cells. The 'Run' button, represented by a play icon, is highlighted with a red rectangle. The main area shows a code cell with the input `In [1]:` and the code `print("hello, world")`, which is also highlighted with a red rectangle. The output of the cell is `hello, world`. Below the code cell is an empty input field labeled `In [ ]:`.

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



The screenshot shows a web browser window with two tabs: 'Home' and 'HelloWorld'. The address bar shows 'localhost:8888/notebooks/Documents/Data/BDA/HelloWorld.ipynb'. The Jupyter interface includes a top bar with the 'jupyter' logo, the notebook name 'HelloWorld (unsaved changes)', a Python logo, and a 'Logout' button. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. To the right of the menu bar are 'Trusted' and 'Python 3' buttons. A toolbar contains icons for saving, adding, deleting, and running cells. The notebook content area shows two code cells. The first cell, labeled 'In [1]:', contains the code `print("hello, world")` and has output `hello, world`. The second cell, labeled 'In [2]:', contains the code `from platform import python_version` and `print("Python Version:", python_version())` and has output `Python Version: 3.6.5`. A third cell labeled 'In []:' is empty.

Home HelloWorld

localhost:8888/notebooks/Documents/Data/BDA/HelloWorld.ipynb

jupyter HelloWorld (unsaved changes) Python Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Save Add Delete Copy Paste Up Down Run Stop Restart Code Keyboard

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

In [2]: `from platform import python_version`
`print("Python Version:", python_version())`
Python Version: 3.6.5

In []:



Python

Programming



Python Fiddle

The screenshot shows the Python Fiddle web application in a browser. The browser's address bar displays 'pythonfiddle.com'. The application's header includes a navigation bar with buttons for 'Run', 'Reset', 'Share', 'Import', and 'Login', along with a 'Language' dropdown menu. On the right side of the header, the 'Python Fiddle' logo is displayed next to the 'Python Cloud IDE' logo. Below the header, the main workspace is divided into three sections. On the left is a sidebar with a list of 'Examples' including 'Chaining comparison operators', 'Decorators', 'Creating generators objects', 'Enumerate', 'Function closure', 'Lex tokenizer', 'Step argument in slice operators', 'For Else', 'Verbose regular expressions', 'In-place value swapping', and 'Function argument unpacking'. Below the examples are buttons for 'Packages' and 'Hotkeys'. The central workspace contains a code editor with two lines of Python code: '1 print("Hello Python Fiddle")' and '2'. To the right of the code editor is a form for saving the fiddle, with fields for 'Title:', 'Description:', and 'Tags:', and a 'Save' button at the bottom. The 'G+1' and '2.6k' social share buttons are visible above the code editor.

Python Cloud IDE | Python Fiddle x

pythonfiddle.com

Run Reset Share Import Login Language

Python Fiddle Python Cloud IDE

G+1 2.6k

Examples

- [Chaining comparison operators](#)
- [Decorators](#)
- [Creating generators objects](#)
- [Enumerate](#)
- [Function closure](#)
- [Lex tokenizer](#)
- [Step argument in slice operators](#)
- [For Else](#)
- [Verbose regular expressions](#)
- [In-place value swapping](#)
- [Function argument unpacking](#)

Packages Hotkeys

```
1 print("Hello Python Fiddle")
2
```

Title:

Description:

Tags:

A comma-separated list of tags.

Save

Hello Python Fiddle

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 "))
```

Python in Google Colab

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

python101.ipynb - Colaboratory

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT?authuser=2#scrollTo=LSNfVqJO-7-U

python101.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL

CONNECTED EDITING

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

hello, world

```
[2] 1 # comment
    2 from platform import python_version
    3 print("Python Version:", python_version())
```

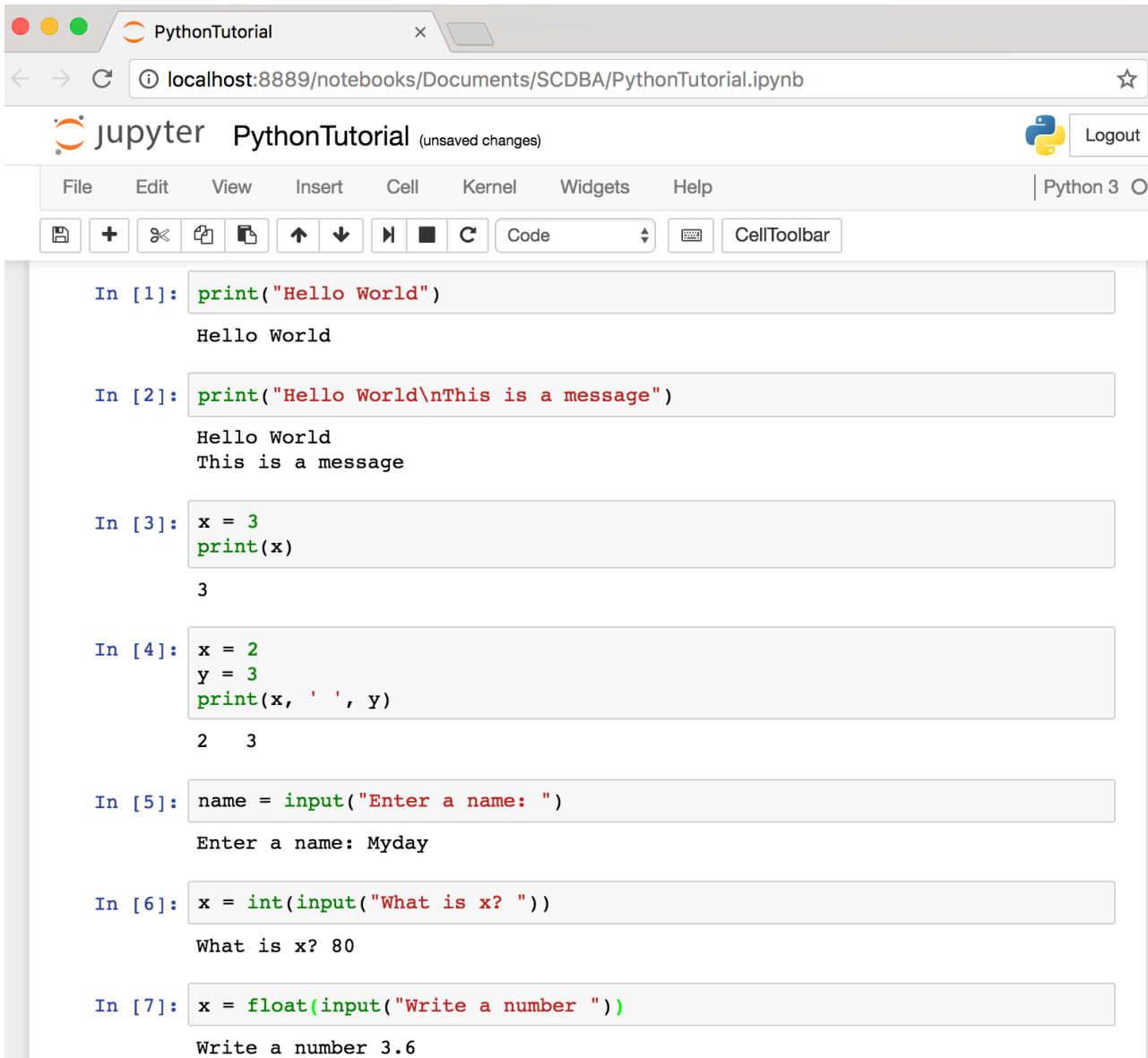
Python Version: 3.6.6

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

Hello World
Hello World
This is a message
3
2 3

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

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 saving, adding, deleting, and running code. The notebook contains seven input cells, each followed by its 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])
```

10

```
print(x[1])
```

20

```
print(x[2])
```

30

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

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)
```

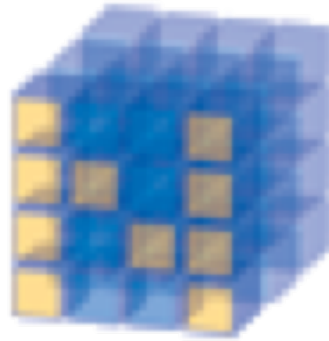
Hello World
This is Python File Input Output
New line

**Big Data Analytics
with**

Numpy

in Python

Numpy



NumPy
Base

N-dimensional array
package

NumPy
is the
fundamental package
for
scientific computing
with Python.



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

[Scipy.org](http://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

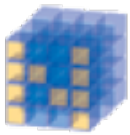
- [Getting NumPy](#)
- [Installing the SciPy Stack](#)
- [NumPy and SciPy documentation page](#)
- [NumPy Tutorial](#)
- [NumPy for MATLAB® Users](#)
- [NumPy functions by category](#)
- [NumPy Mailing List](#)

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](#)



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 = list(range(1, 6))
```

```
v
```

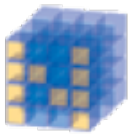
```
2 * v
```

```
import numpy as np
```

```
v = np.arange(1, 6)
```

```
v
```

```
2 * v
```



NumPy

Base
N-dimensional
array package

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

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

```
1 2 * v
```

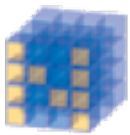
```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

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

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

```
1 2 * v
```

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

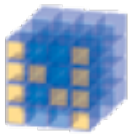
NumPy

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

```
1 import numpy as np
2
3 a = np.zeros((2,2)) # Create an array of all zeros
4 print(a)           # Prints "[[ 0.  0.]
5                     #                [ 0.  0.]]"
6
7 b = np.ones((1,2)) # Create an array of all ones
8 print(b)           # Prints "[[ 1.  1.]]"
9
10 c = np.full((2,2), 7) # Create a constant array
11 print(c)             # Prints "[[ 7.  7.]
12                     #                [ 7.  7.]]"
13
14 d = np.eye(2)        # Create a 2x2 identity matrix
15 print(d)            # Prints "[[ 1.  0.]
16                     #                [ 0.  1.]]"
17
18 e = np.random.random((2,2)) # Create an array filled with random values
19 print(e)              # Might print "[[ 0.91940167  0.08143941]
20                      #                [ 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]]
```

Numpy Quickstart Tutorial

[SciPy.org](#)[Docs](#)[NumPy v1.13.dev0 Manual](#)[NumPy User Guide](#)[index](#)[next](#)[previous](#)

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
```

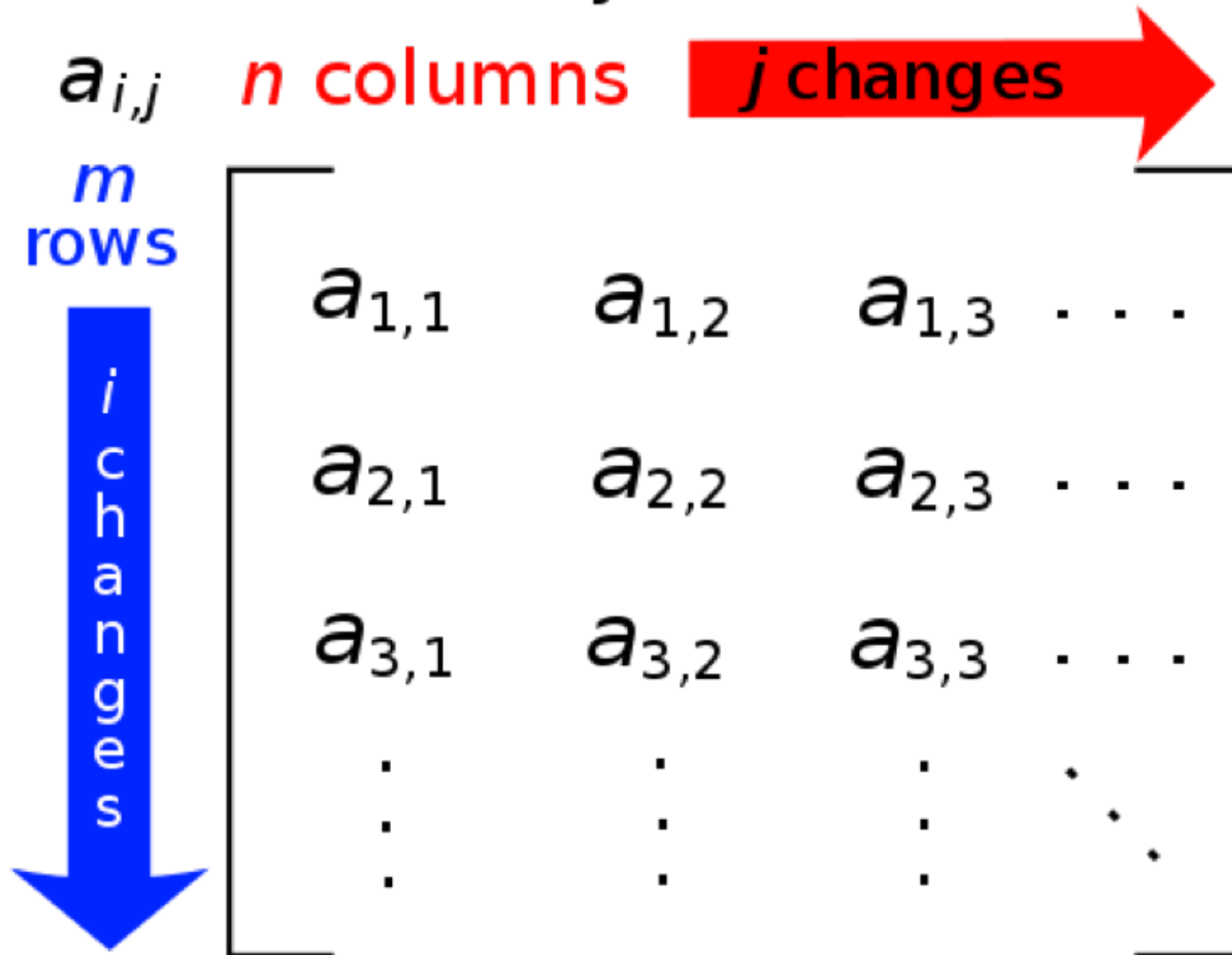
```
2
```

```
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

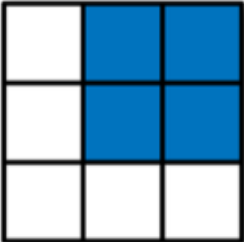
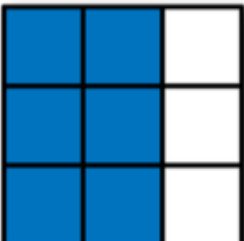
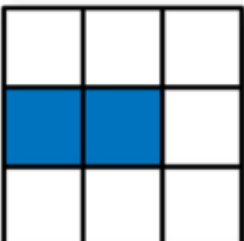
NumPy Basics: Arrays and Vectorized Computation

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
	<code>arr[:2, 1:]</code>	<code>(2, 2)</code>
	<code>arr[2]</code> <code>arr[2, :]</code> <code>arr[2:, :]</code>	<code>(3,)</code> <code>(3,)</code> <code>(1, 3)</code>
	<code>arr[:, :2]</code>	<code>(3, 2)</code>
	<code>arr[1, :2]</code> <code>arr[1:2, :2]</code>	<code>(2,)</code> <code>(1, 2)</code>

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))
```

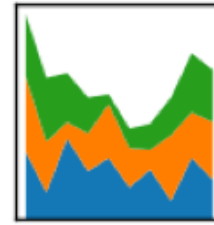
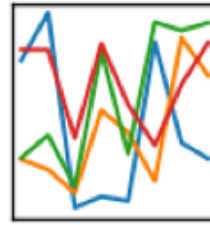
```
In [ ]: %time for _ in range(10): my_arr2 = my_arr * 2
%time for _ in range(10): my_list2 = [x * 2 for x in my_list]
```

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

Python Pandas for Finance

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
```

AAPL



Finance Data from Yahoo Finance

```
# !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'])
```

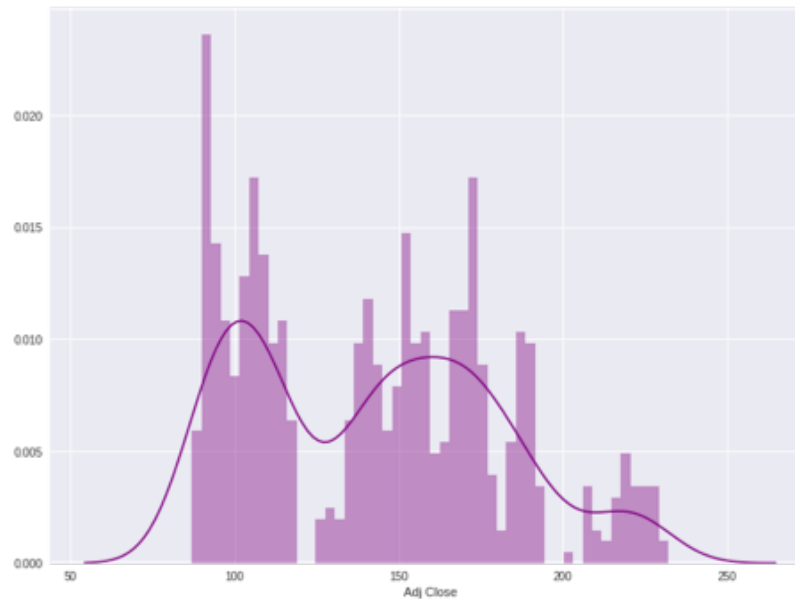


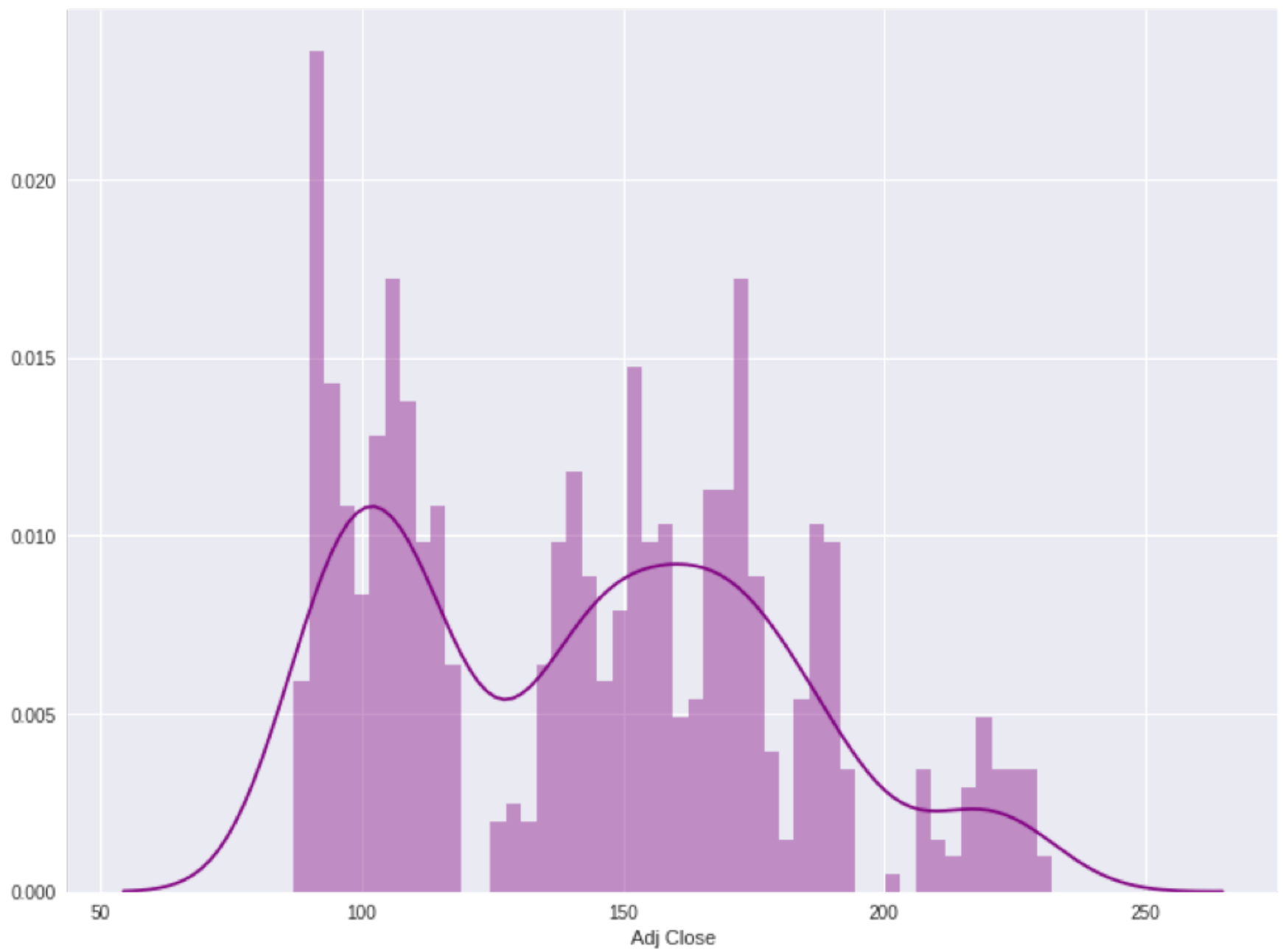
AAPL



```
# 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()
```

AAPL



```

# !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()

```

```

1 # !pip install pandas_datareader
2 import pandas as pd
3 import pandas_datareader.data as web
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 import datetime as dt
7 %matplotlib inline
8
9 #Read Stock Data from Yahoo Finance
10 end = dt.datetime.now()
11 #start = dt.datetime(end.year-2, end.month, end.day)
12 start = dt.datetime(2016, 1, 1)
13 df = web.DataReader("AAPL", 'yahoo', start, end)
14 df.to_csv('AAPL.csv')
15 df.from_csv('AAPL.csv')
16 df.tail()
17
18 df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
19 plt.figure(figsize=(12,9))
20 top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
21 bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
22 top.plot(df.index, df['Adj Close'], color='blue') #df.index gives the dates
23 bottom.bar(df.index, df['Volume'])
24
25 # set the labels
26 top.axes.get_xaxis().set_visible(False)
27 top.set_title('AAPL')
28 top.set_ylabel('Adj Close')
29 bottom.set_ylabel('Volume')
30
31 plt.figure(figsize=(12,9))
32 sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')
33
34 # simple moving averages
35 df['MA05'] = df['Adj Close'].rolling(5).mean() #5 days
36 df['MA20'] = df['Adj Close'].rolling(20).mean() #20 days
37 df['MA60'] = df['Adj Close'].rolling(60).mean() #60 days
38 df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60']})
39 df2.plot(figsize=(12, 9), legend=True, title='AAPL')
40 df2.to_csv('AAPL_MA.csv')
41 fig = plt.gcf()
42 fig.set_size_inches(12, 9)
43 fig.savefig('AAPL_plot.png', dpi=300)
44 plt.show()

```


Finance Data from Quandl

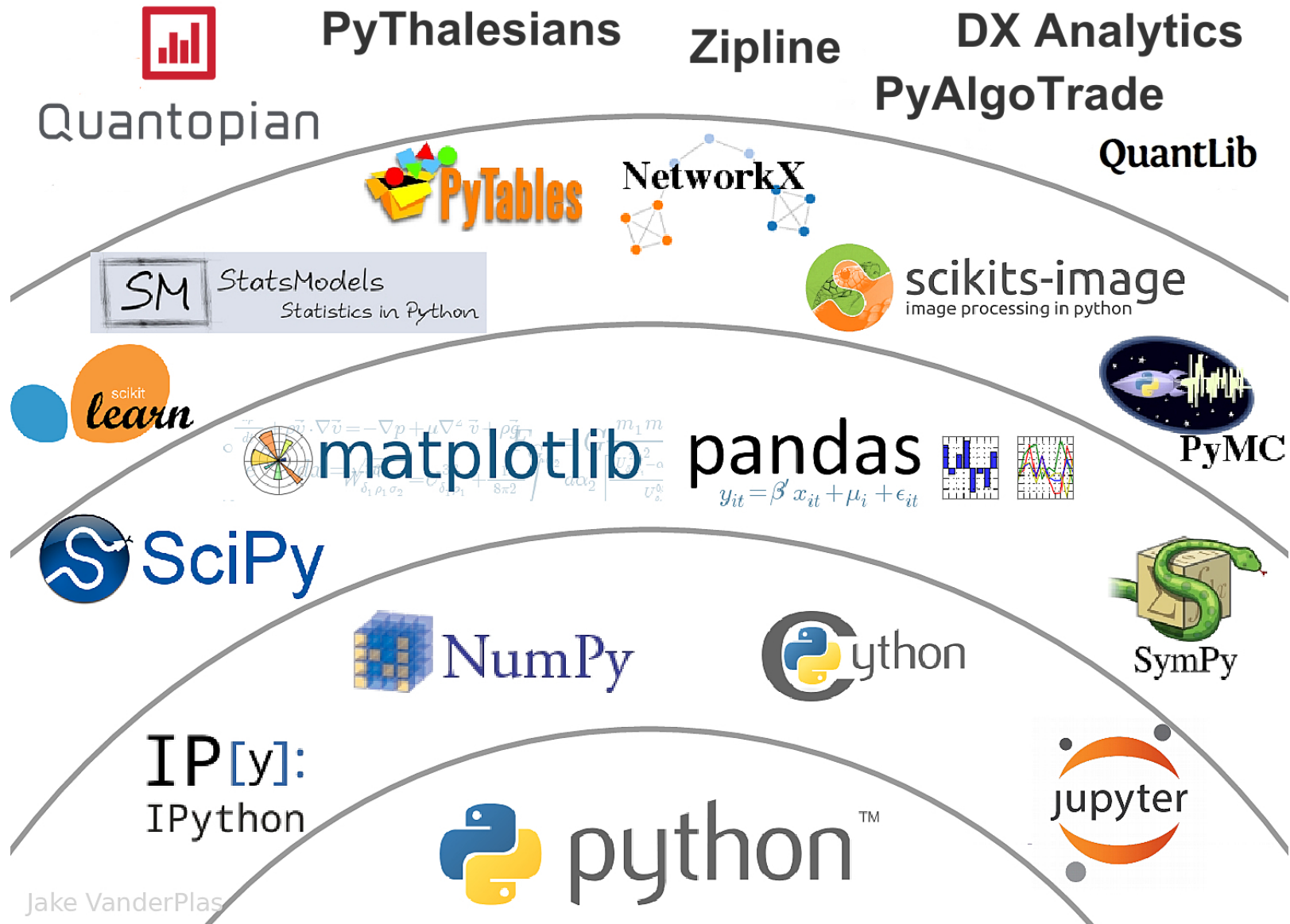
```
# ! pip install quandl
import quandl
# quandl.ApiConfig.api_key = "YOURAPIKEY"
df = quandl.get("WIKI/AAPL", start_date="2016-01-01", end_date="2017-12-31")
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

```
1 # ! pip install quandl
2 import quandl
3 # quandl.ApiConfig.api_key = "YOURAPIKEY"
4 df = quandl.get("WIKI/AAPL", start_date="2016-01-01", end_date="2017-12-31")
5 df.to_csv('AAPL.csv')
6 df.from_csv('AAPL.csv')
7 df.tail()
```

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: FutureWarning: from_csv is deprecated. Please use read_csv(...) instead
"""

	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
Date												
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	32968167.0	0.0	1.0	170.80	171.470	169.679	170.57	32968167.0
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



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

- 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>
- Ties de Kok (2017), Learn Python for Research,
<https://github.com/TiesdeKok/LearnPythonforResearch>
- Avinash Jain (2017), Introduction To Python Programming, Udemy,
<https://www.udemy.com/pythonforbeginnersintro/>
- 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>
- Jason Brownlee (2016), Your First Machine Learning Project in Python Step-By-Step,
<https://machinelearningmastery.com/machine-learning-in-python-step-by-step/>