Tamkang University

Big Data Mining

Foundations of Big Data Mining in Python

1071BDM05
TLVXM1A (M2244) (8619) (Fall 2018)
(MBA, DBETKU) (3 Credits, Required) [Full English Course]
(Master's Program in Digital Business and Economics)
Mon, 9, 10, 11, (16:10-19:00) (B206)

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

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

2018-10-15

Course Schedule (1/2)

Tamkang University

Week	Date	Subject/Topics
1	2018/09/10	Course Orientation for Big Data Mining
2	2018/09/17	ABC: AI, Big Data, Cloud Computing
3	2018/09/24	Mid-Autumn Festival (Day off)
4	2018/10/01	Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data
5	2018/10/08	Fundamental Big Data: MapReduce Paradigm, Hadoop and Spark Ecosystem
6	2018/10/15	Foundations of Big Data Mining in Python
7	2018/10/22	Supervised Learning: Classification and Prediction
8	2018/10/29	Unsupervised Learning: Cluster Analysis
9	2018/11/05	Unsupervised Learning: Association Analysis

Course Schedule (2/2)

Tamkang University

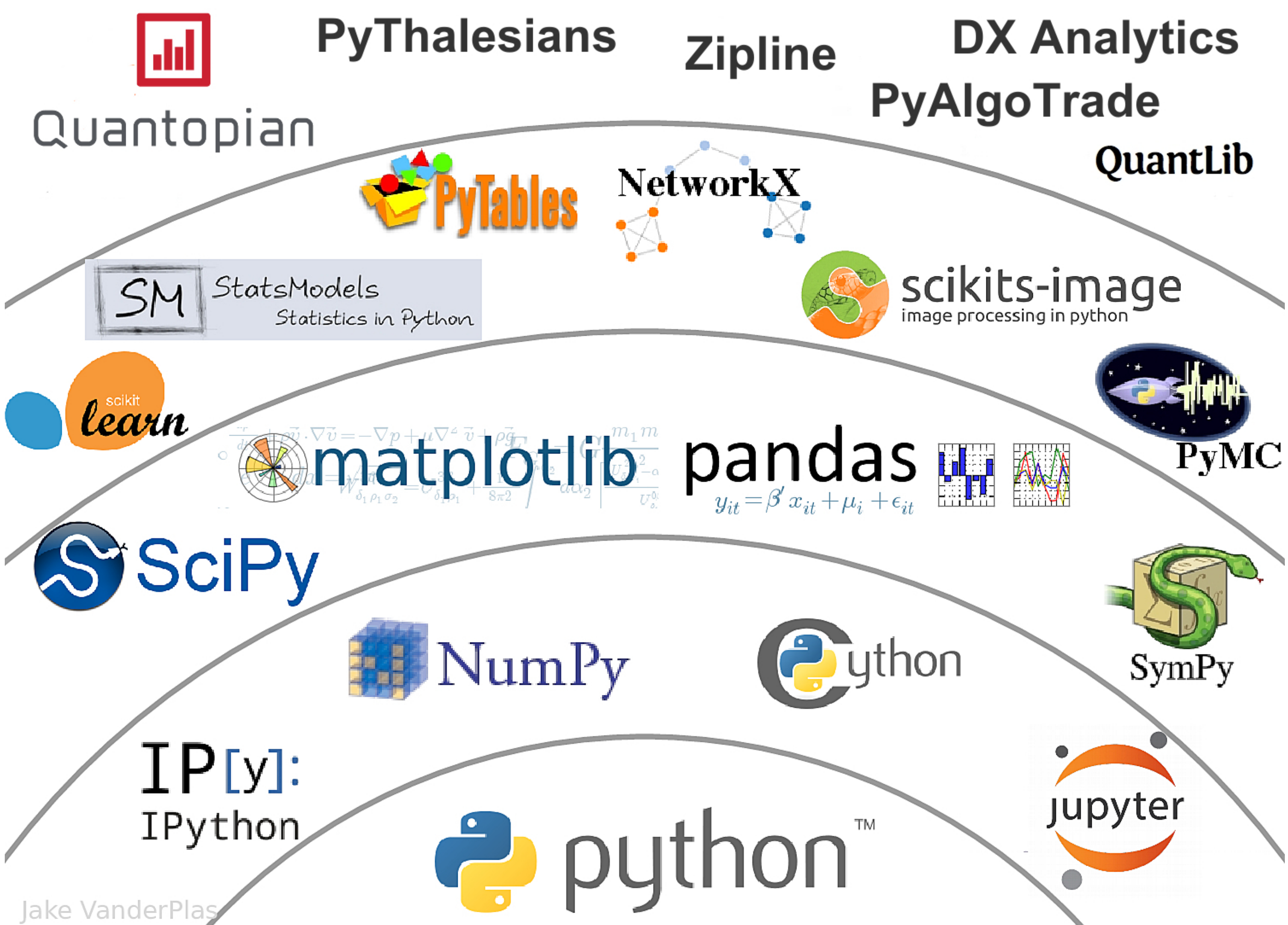
Week	Date	Subject/Topics
10	2018/11/12	Midterm Project Report
11	2018/11/19	Machine Learning with Scikit-Learn in Python
12	2018/11/26	Deep Learning for Finance Big Data with TensorFlow
13	2018/12/03	Convolutional Neural Networks (CNN)
14	2018/12/10	Recurrent Neural Networks (RNN)
15	2018/12/17	Reinforcement Learning (RL)
16	2018/12/24	Social Network Analysis (SNA)
17	2018/12/31	Bridge Holiday (Extra Day Off)
18	2019/01/07	Final Project Presentation

Foundations of Big Data Mining in Python

Ouline

- **Foundations of Big Data Mining in Python**
  - **Python**
    - programming language
  - **Numpy**
    - scientific computing
  - **Pandas**
    - data structures and data analysis tools

The Quant Finance PyData Stack



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

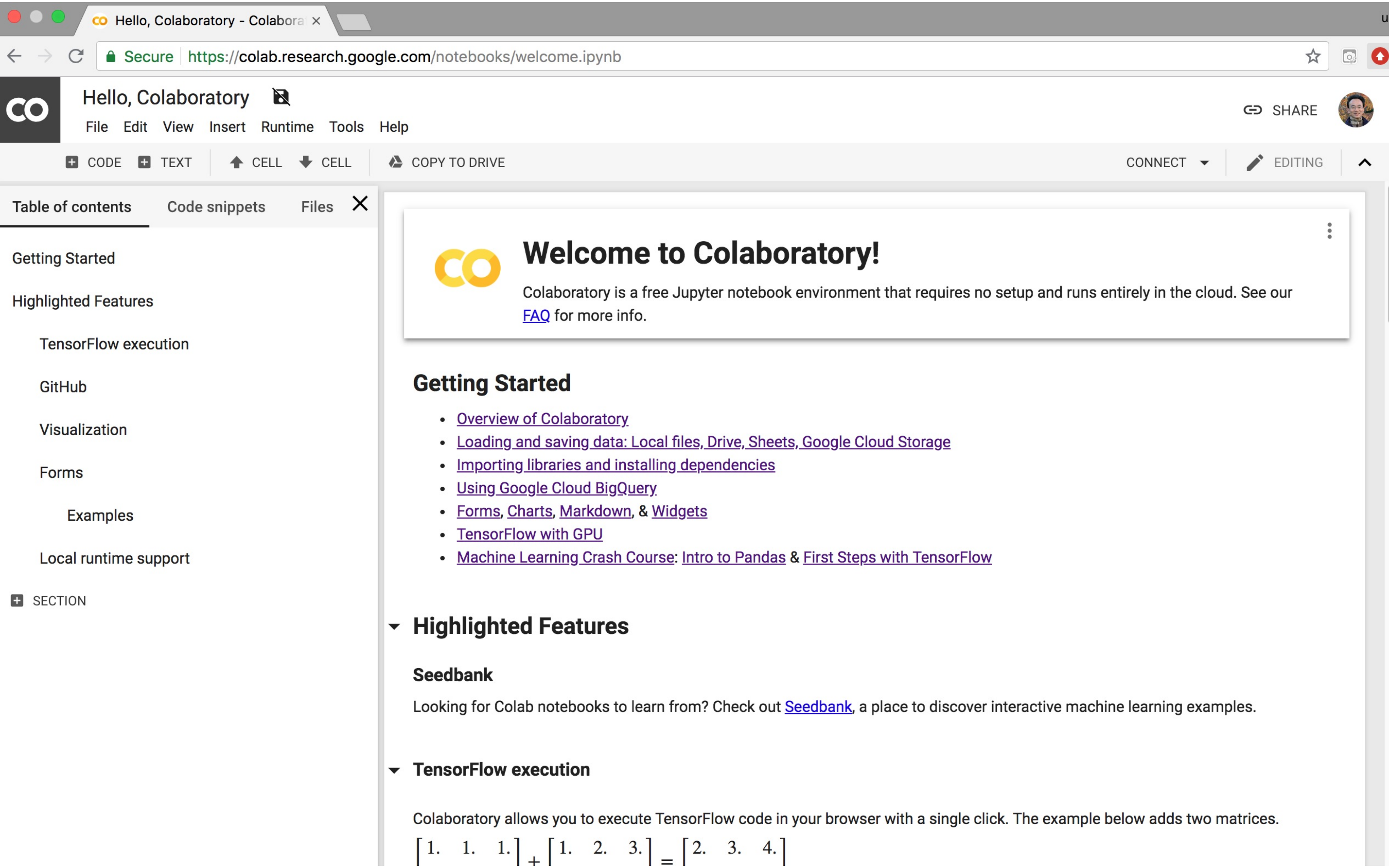
Python

python™

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

Source: https://www.python.org/doc/essays/blurb/

Google Colab



https://colab.research.google.com/notebooks/welcome.ipynb

Python in Google Colab

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

python101.ipynb

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

```
194.87
```

[11]

```python
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
```

[12]

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

```
194.87
```

[13]

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

```
Pass
```

Python matplotlib

Source: https://matplotlib.org/

Python Pandas

pandas

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

http://pandas.pydata.org/

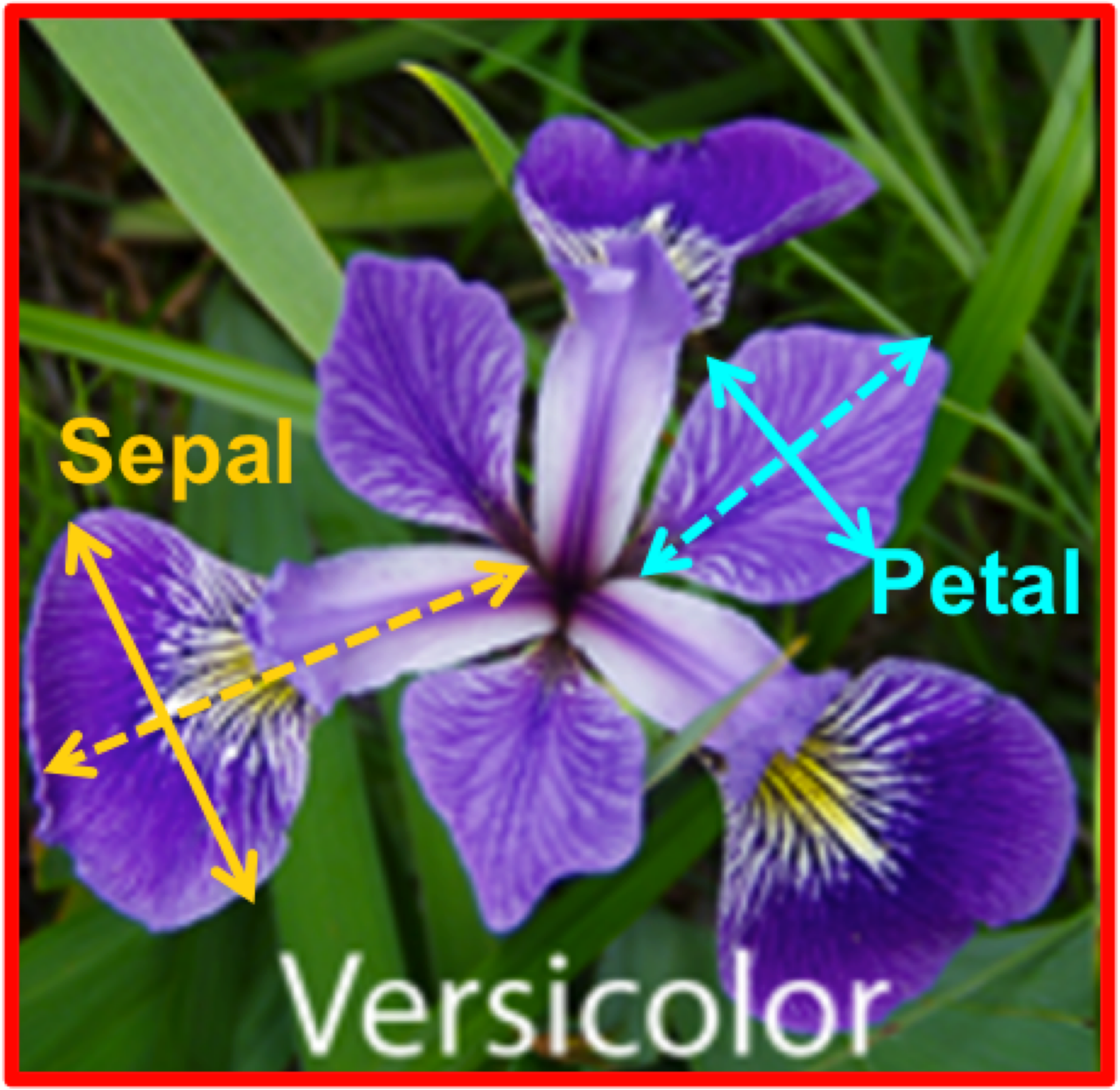
Iris flower data set

setosa **versicolor** **virginica**

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

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

Iris Classfication



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

iris.data

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

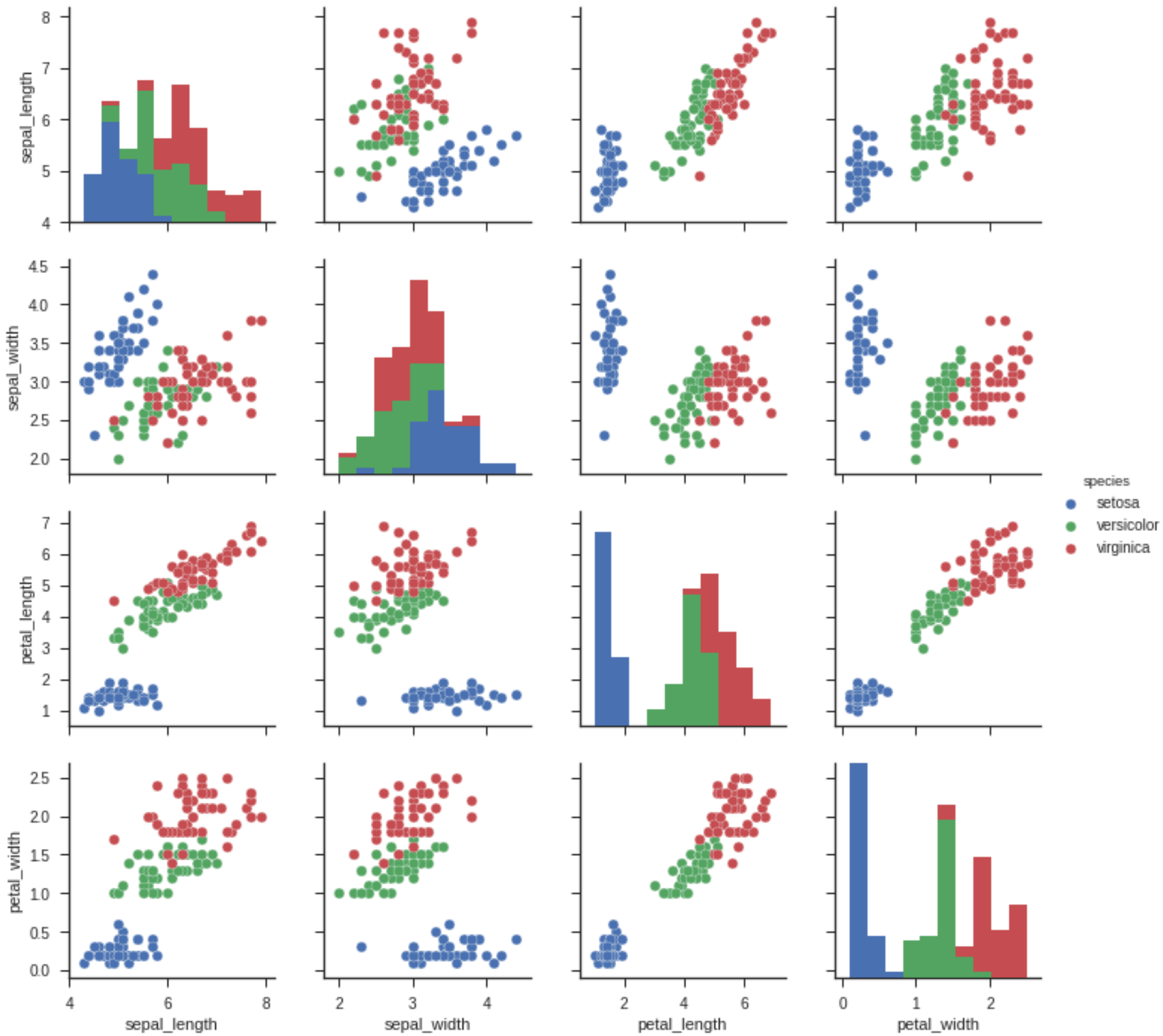
```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
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
```

setosa

virginica

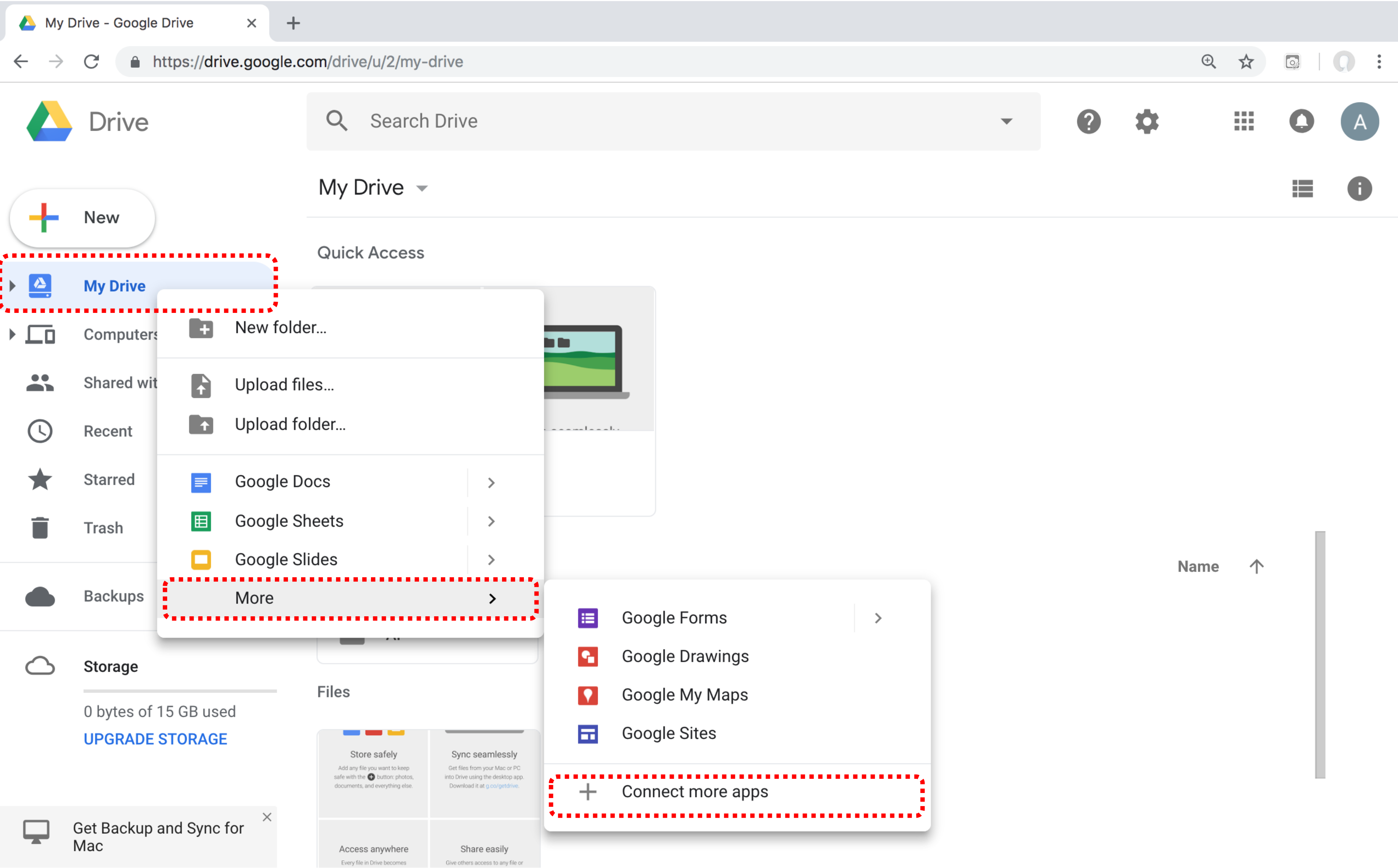
versicolor

Iris Data Visualization

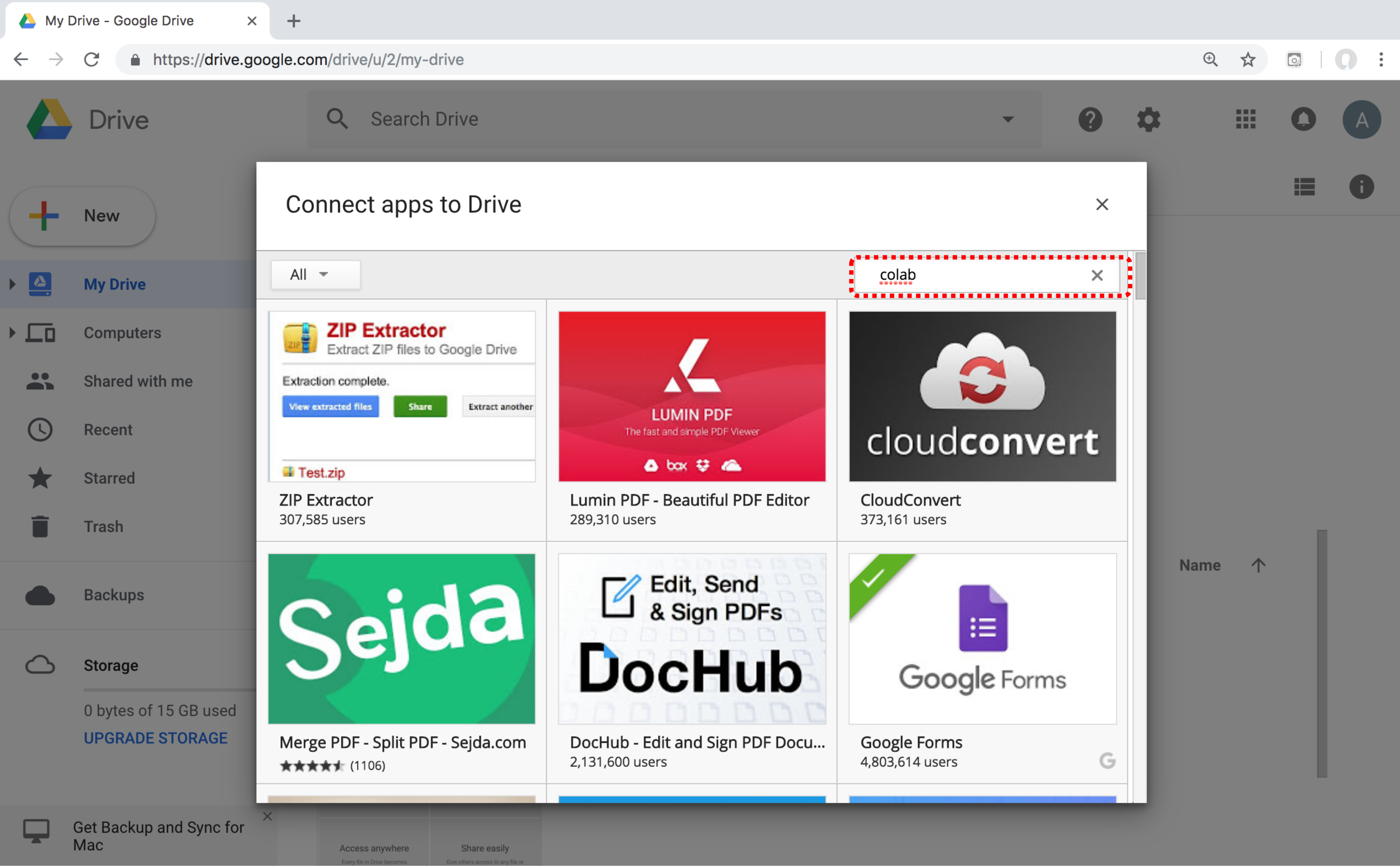


Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html

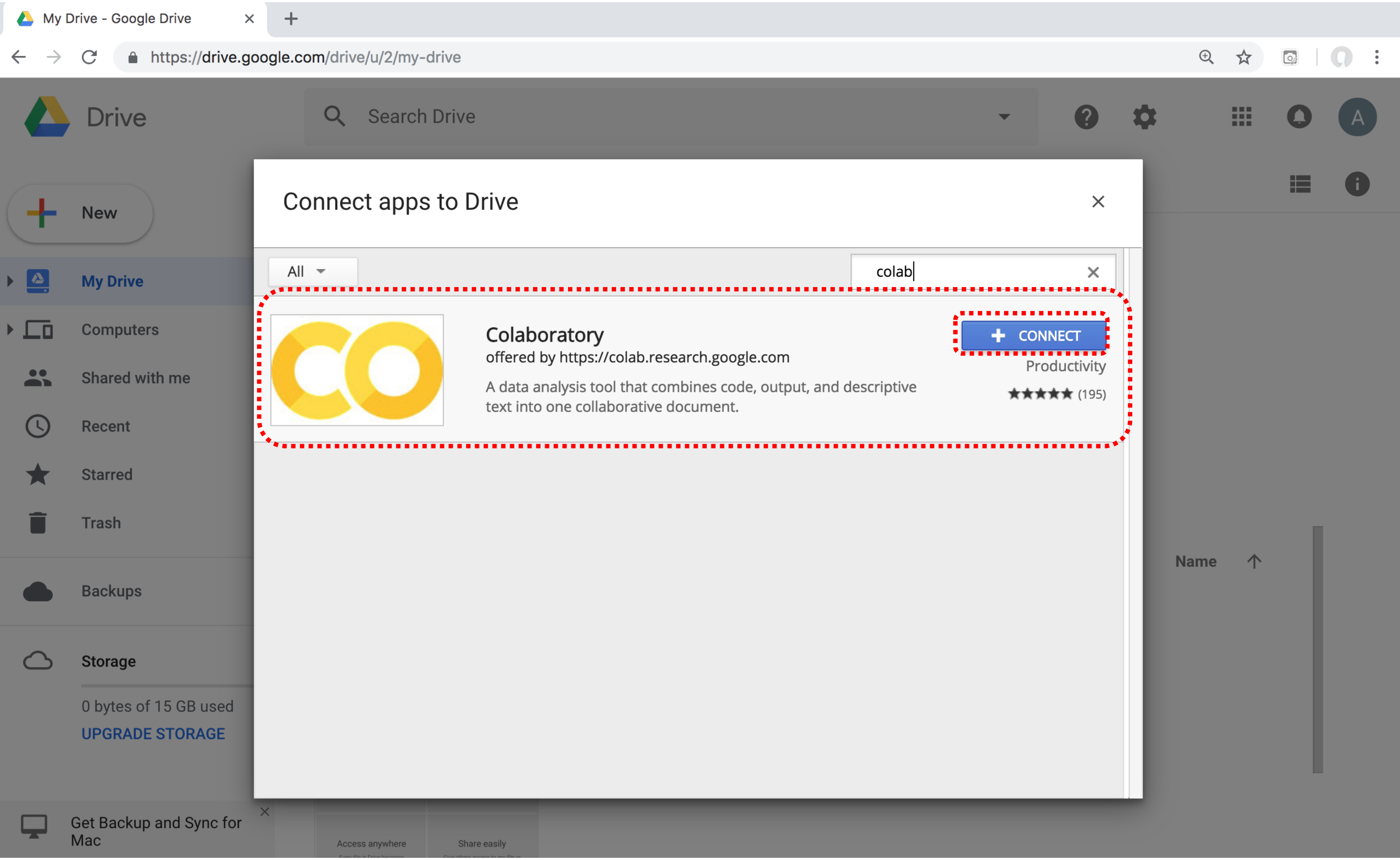
Connect Google Colab in Google Drive



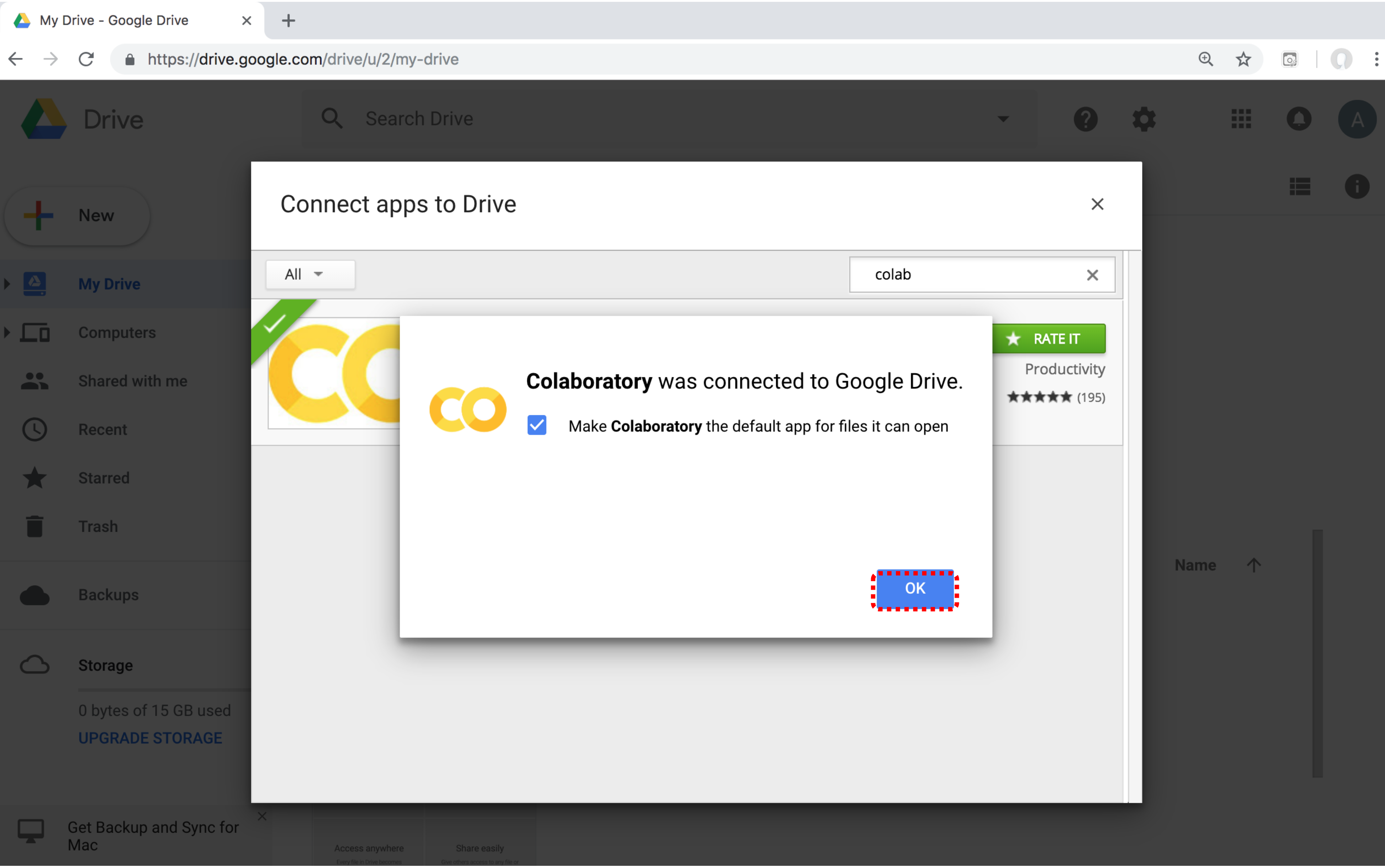
Google Colab



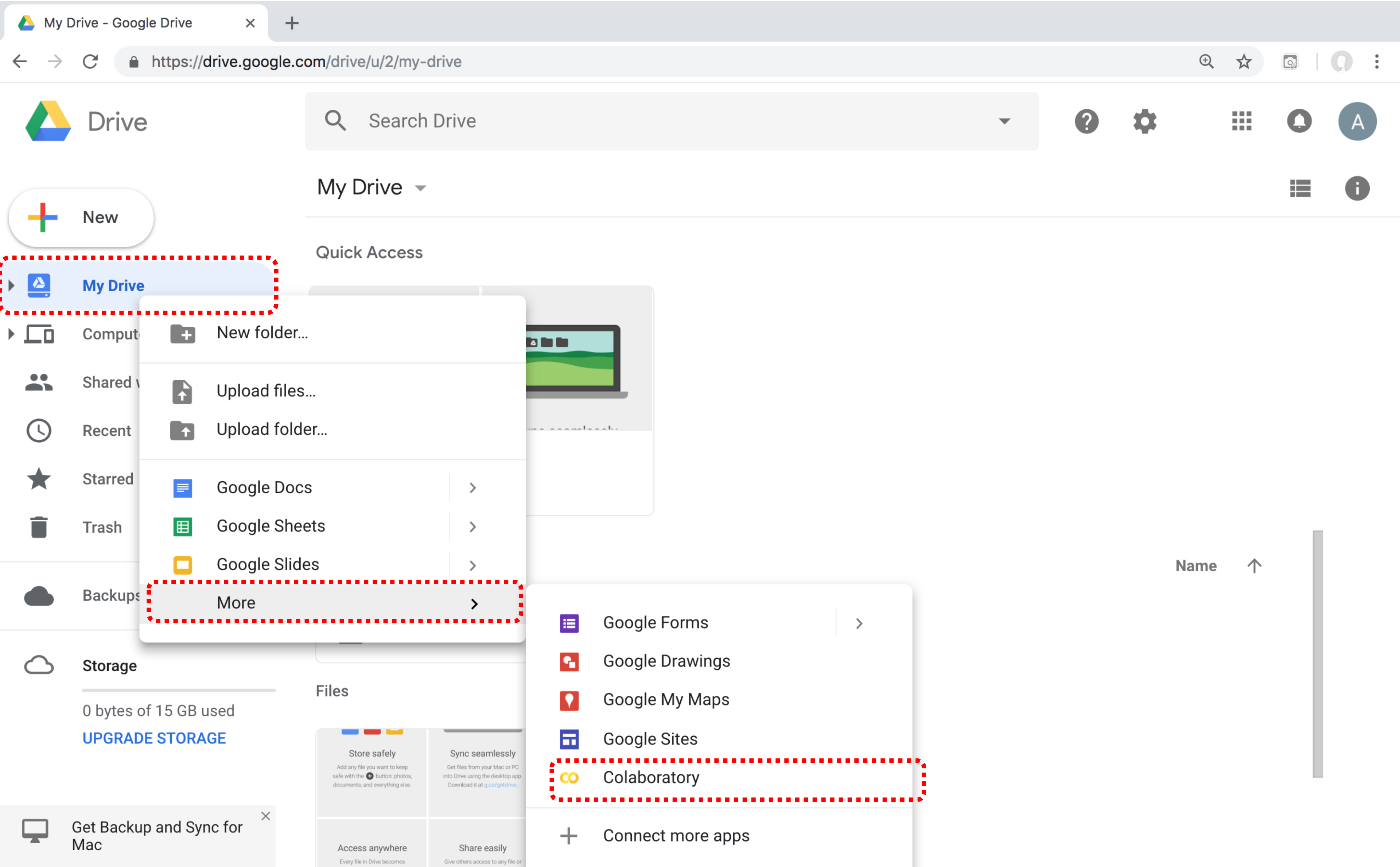
Google Colab



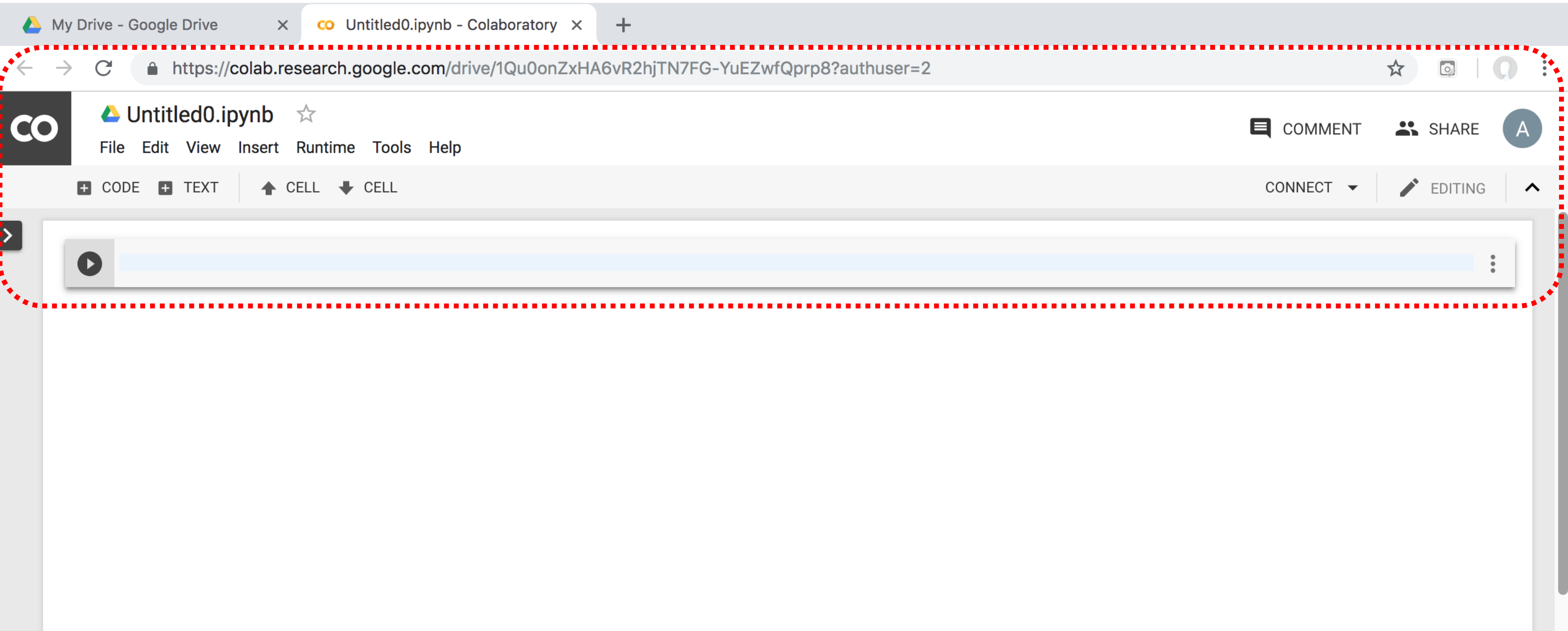
Connect Colaboratory to Google Drive



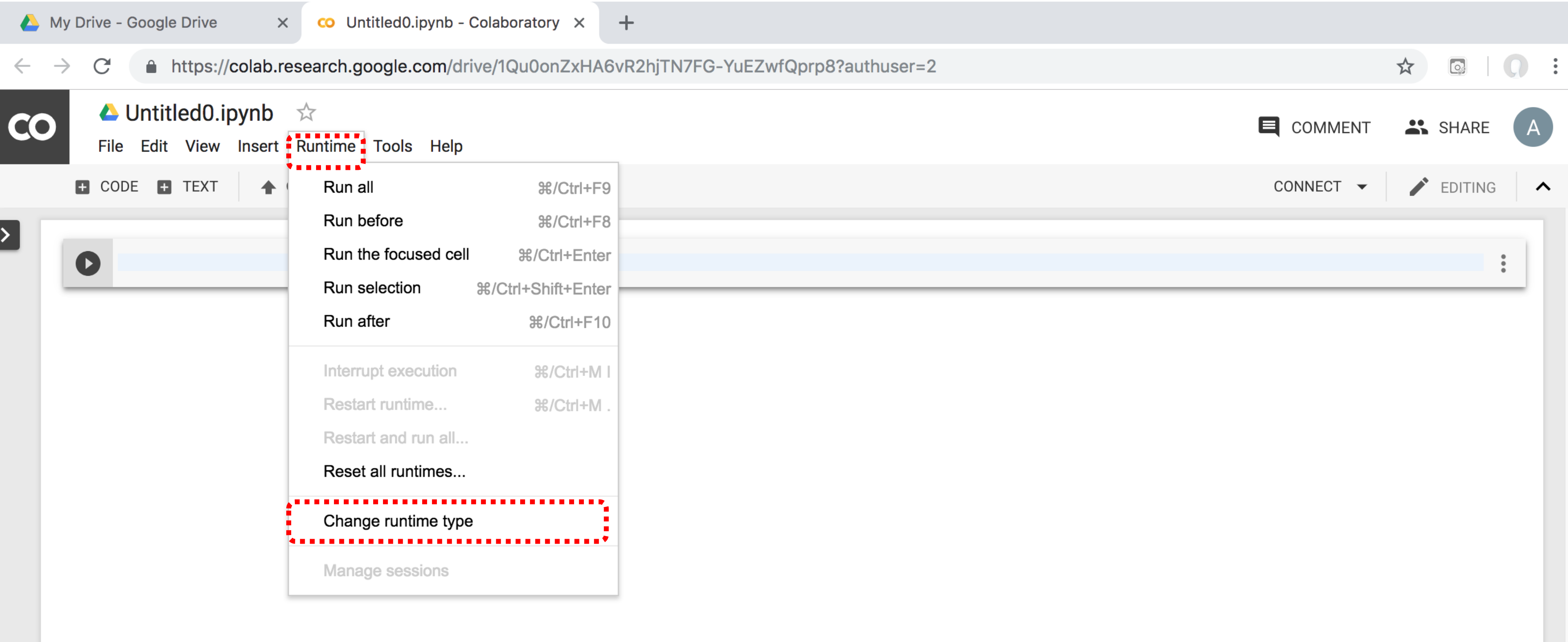
Google Colab



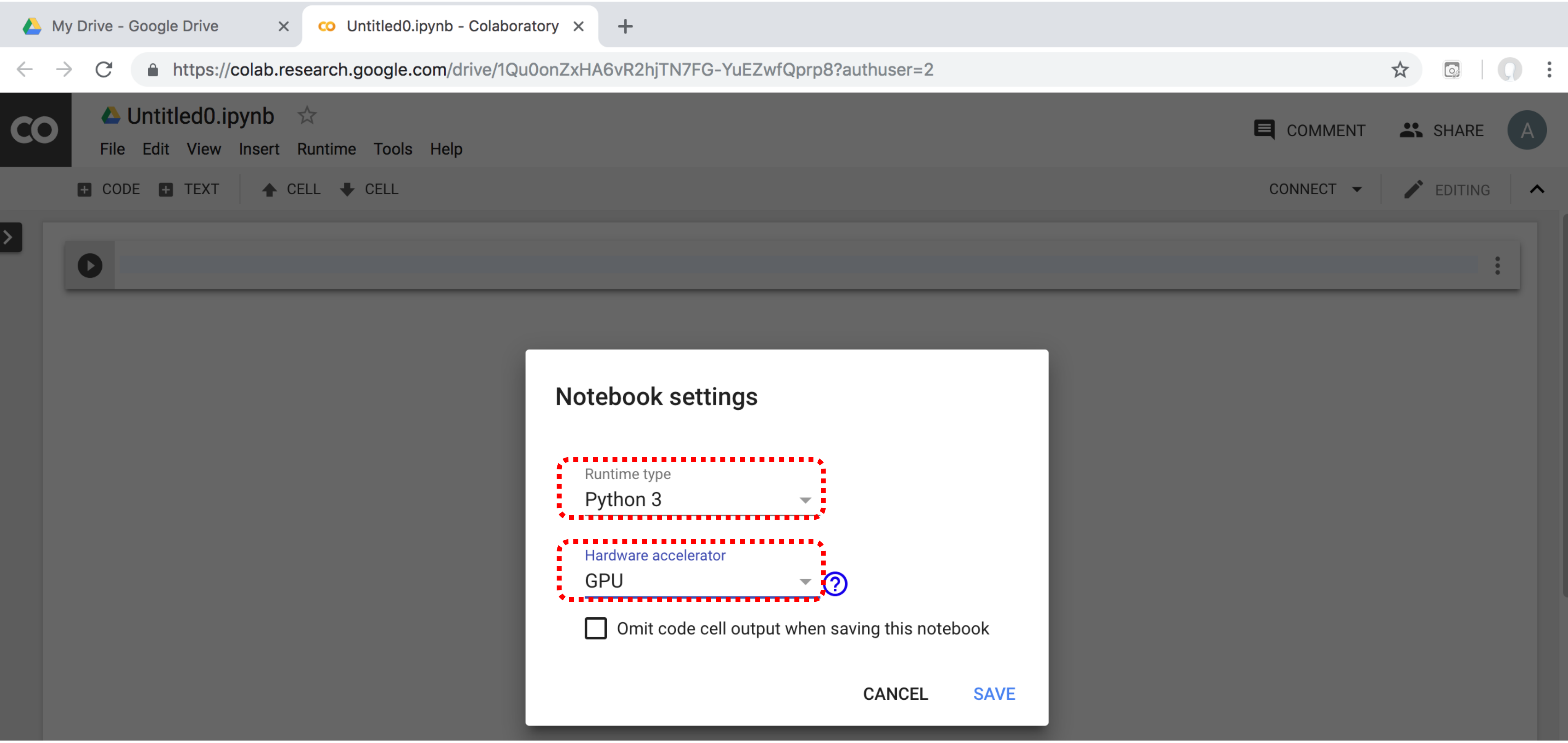
Google Colab



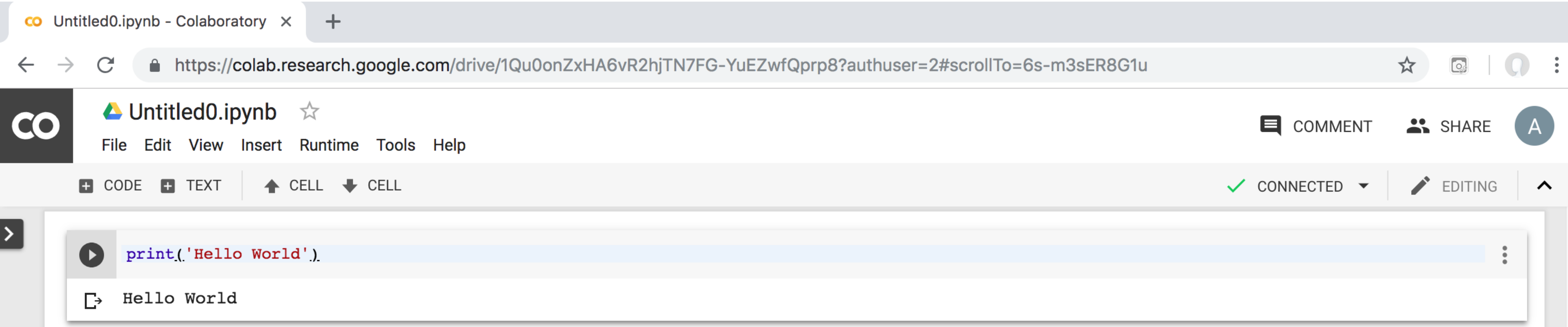
Google Colab



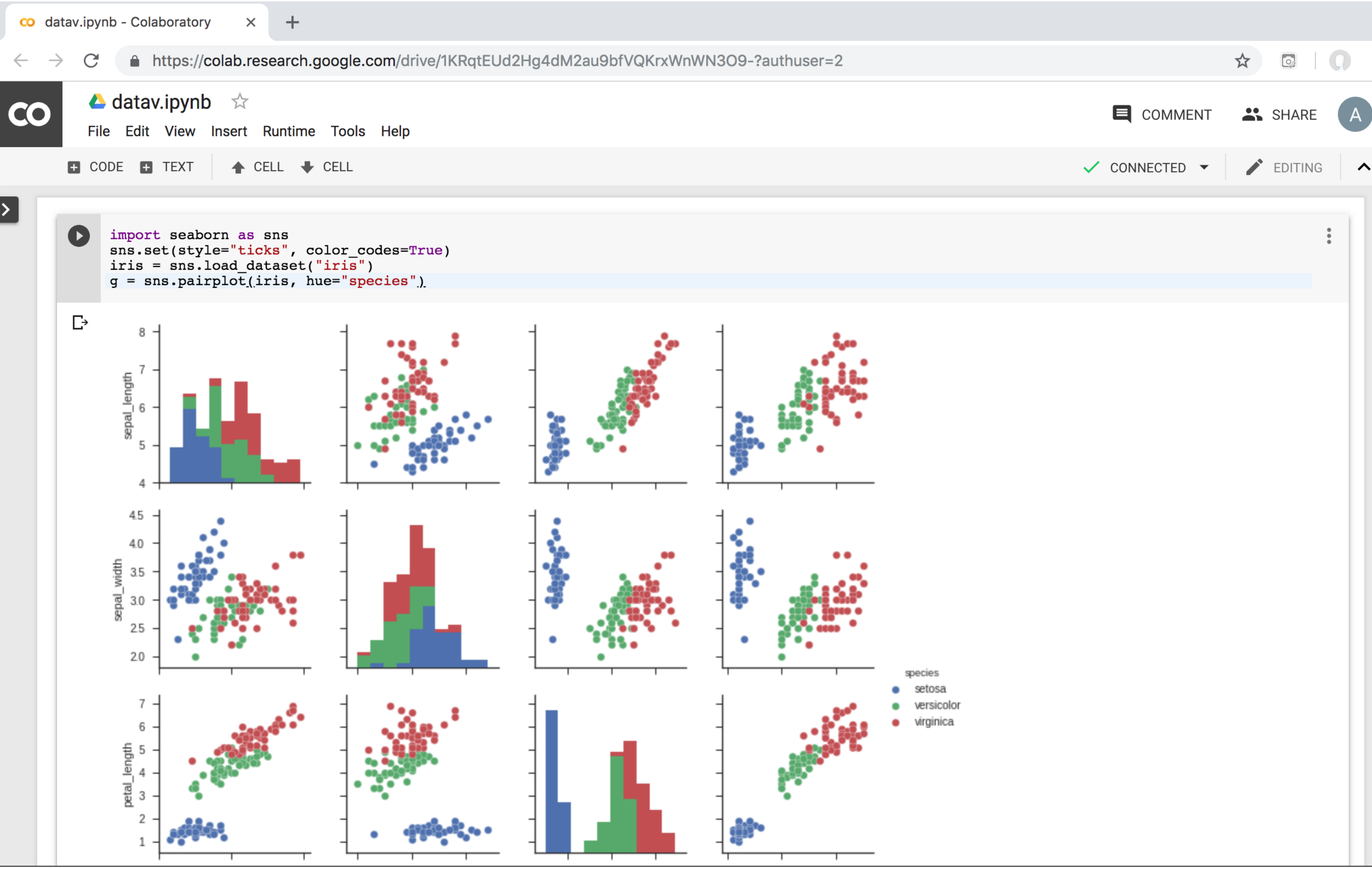
Run Jupyter Notebook Python3 GPU Google Colab



Google Colab Python Hello World
print('Hello World')

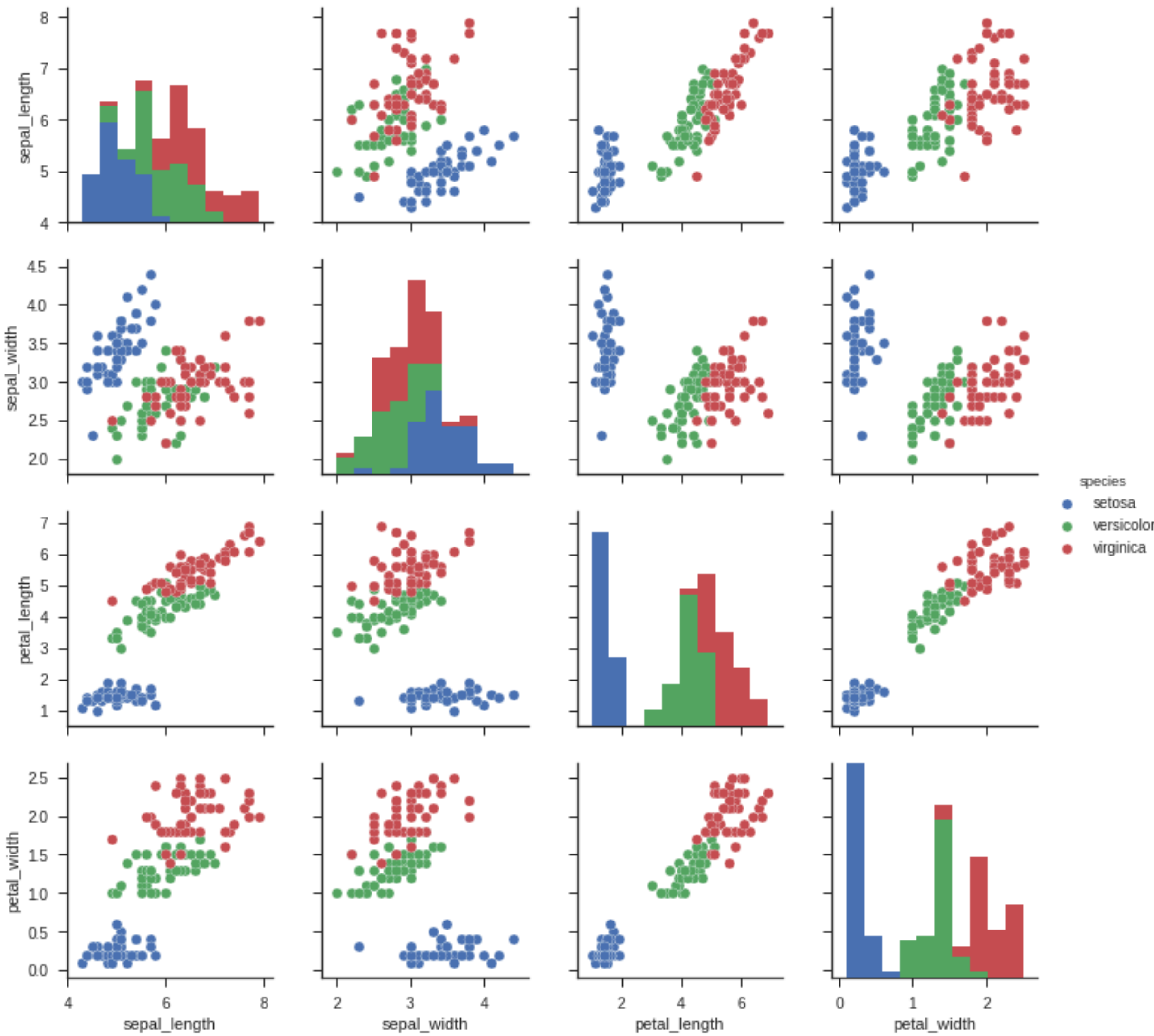


Data Visualization in Google Colab



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")
```



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

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

```
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)
```

Source: https://machinelearningmastery.com/machine-learning-in-python-step-by-step/

```python
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
```

```python
# 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)
```

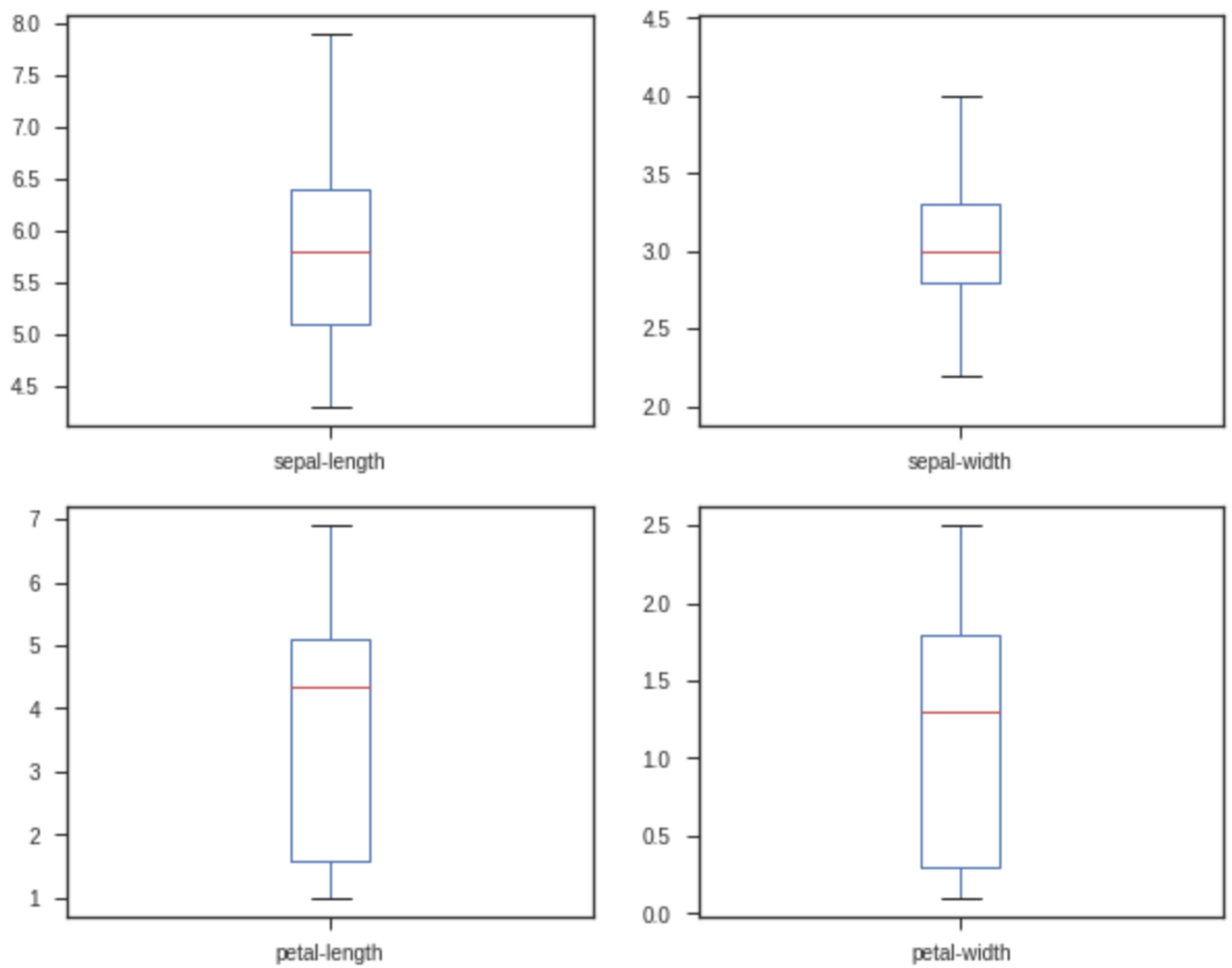
df.groupby('class').size()

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

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

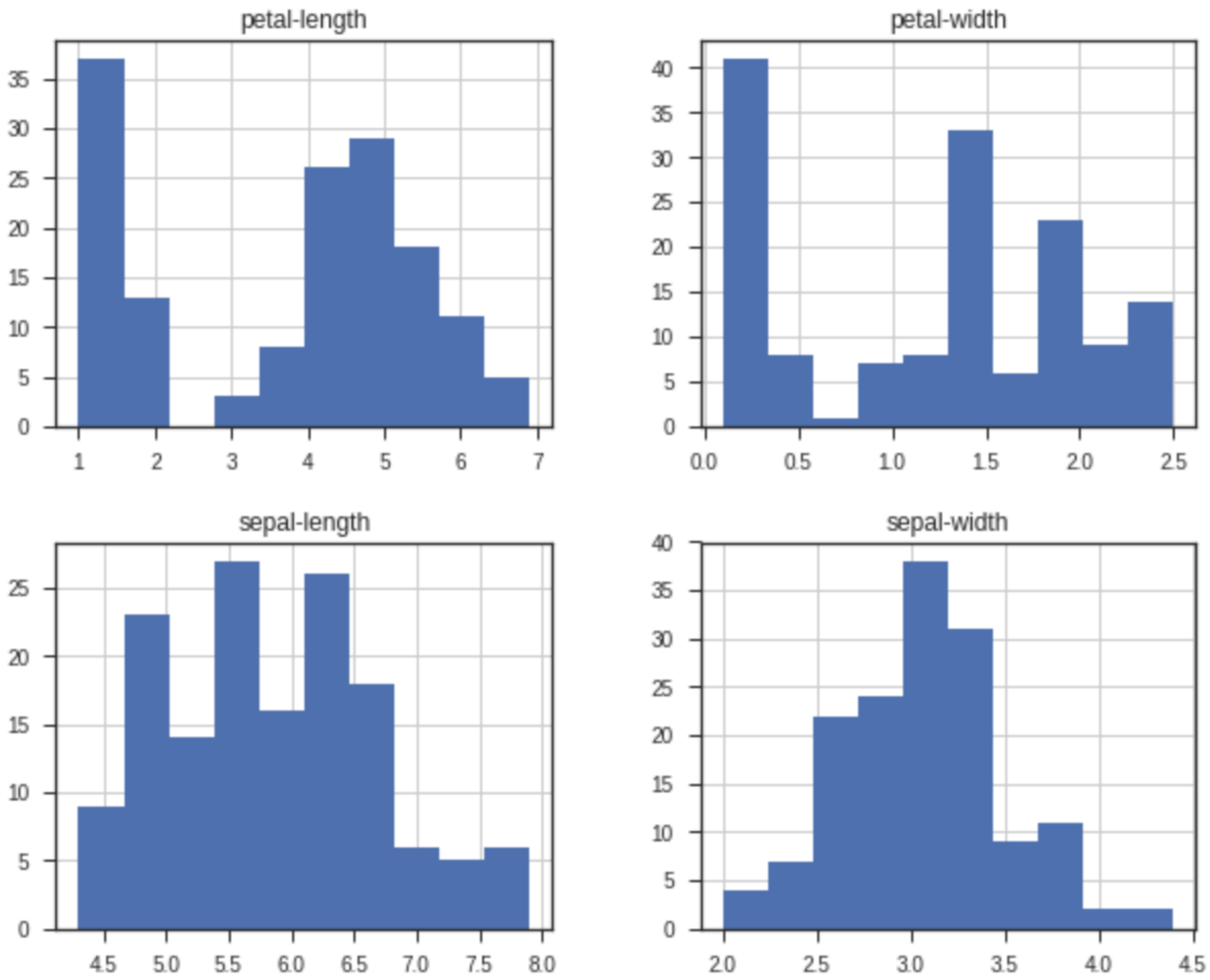
```
plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```


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

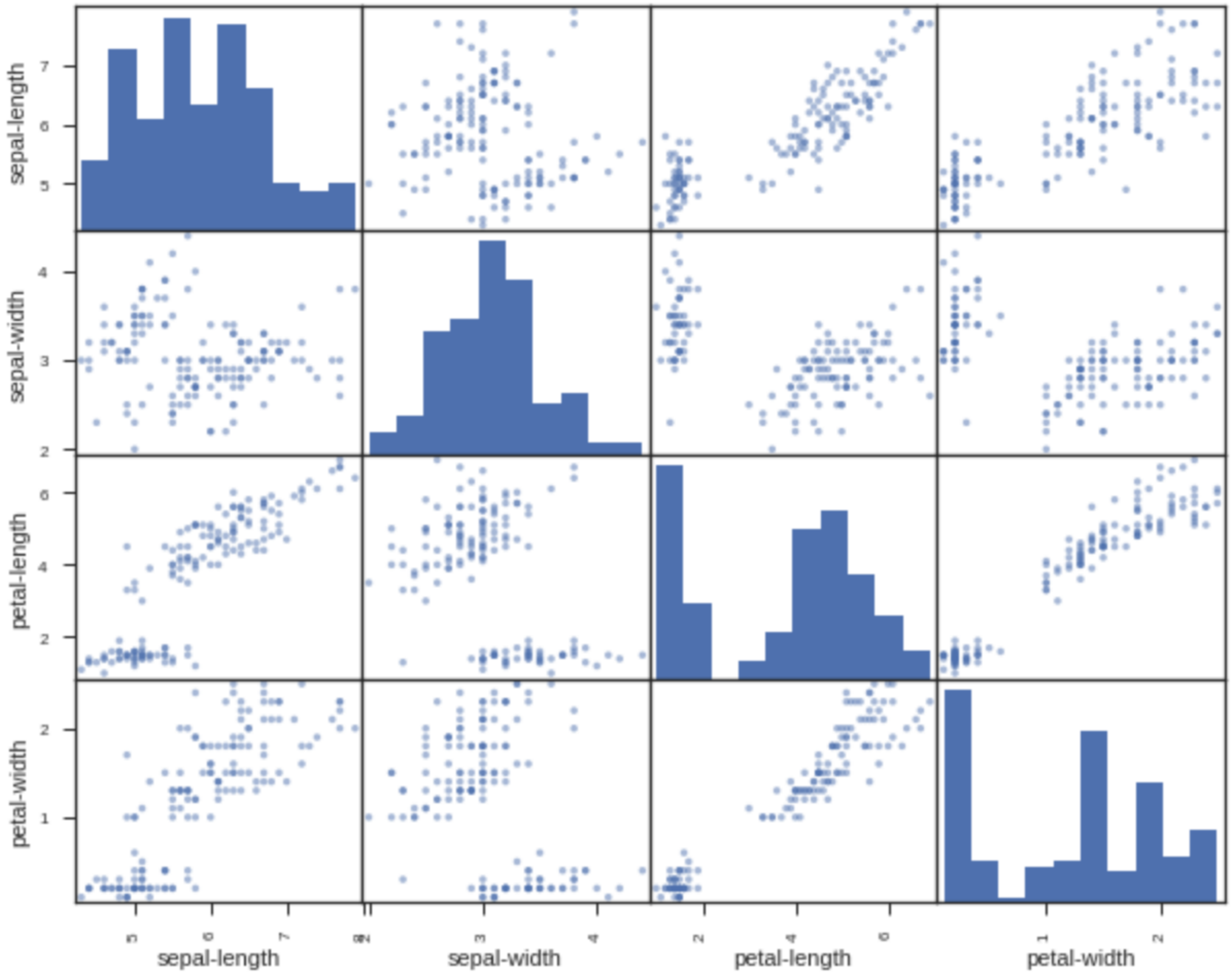
```
df.hist()
plt.show()
```

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



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

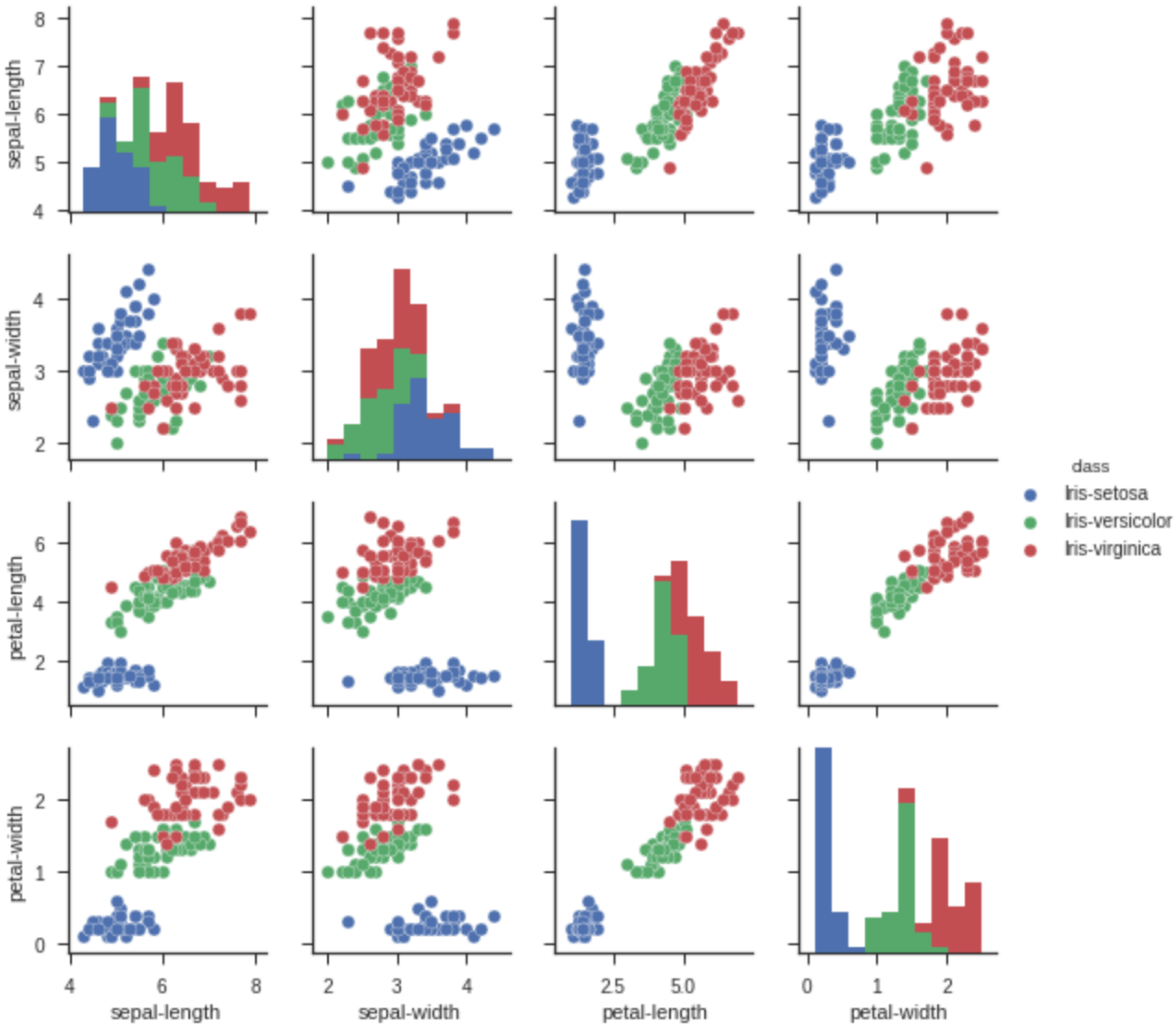
```
scatter_matrix(df)
plt.show()
```



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

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

<seaborn.axisgrid.PairGrid at 0x7f1d21267390>

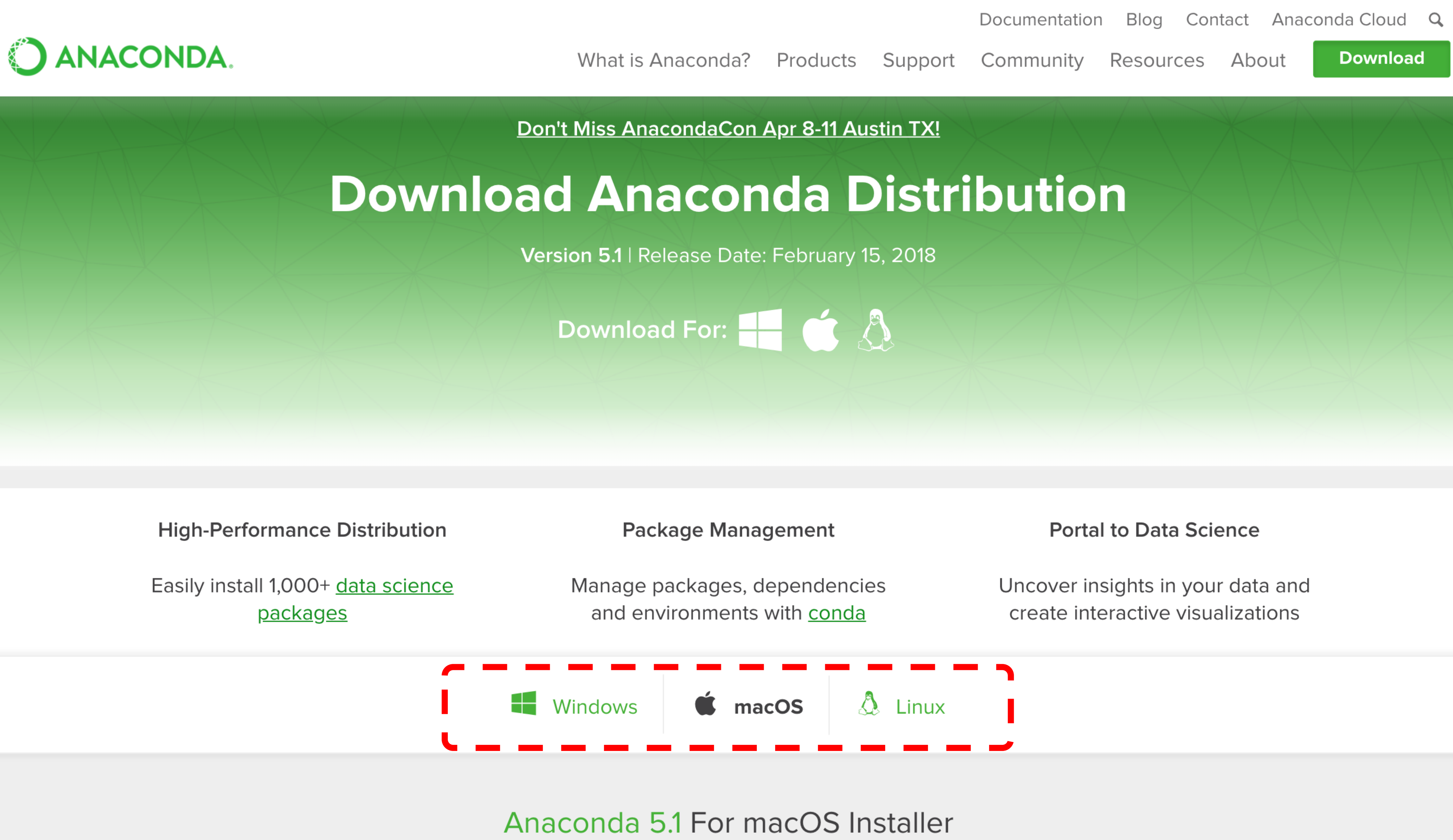


Anaconda

The Most Popular Python Data Science Platform

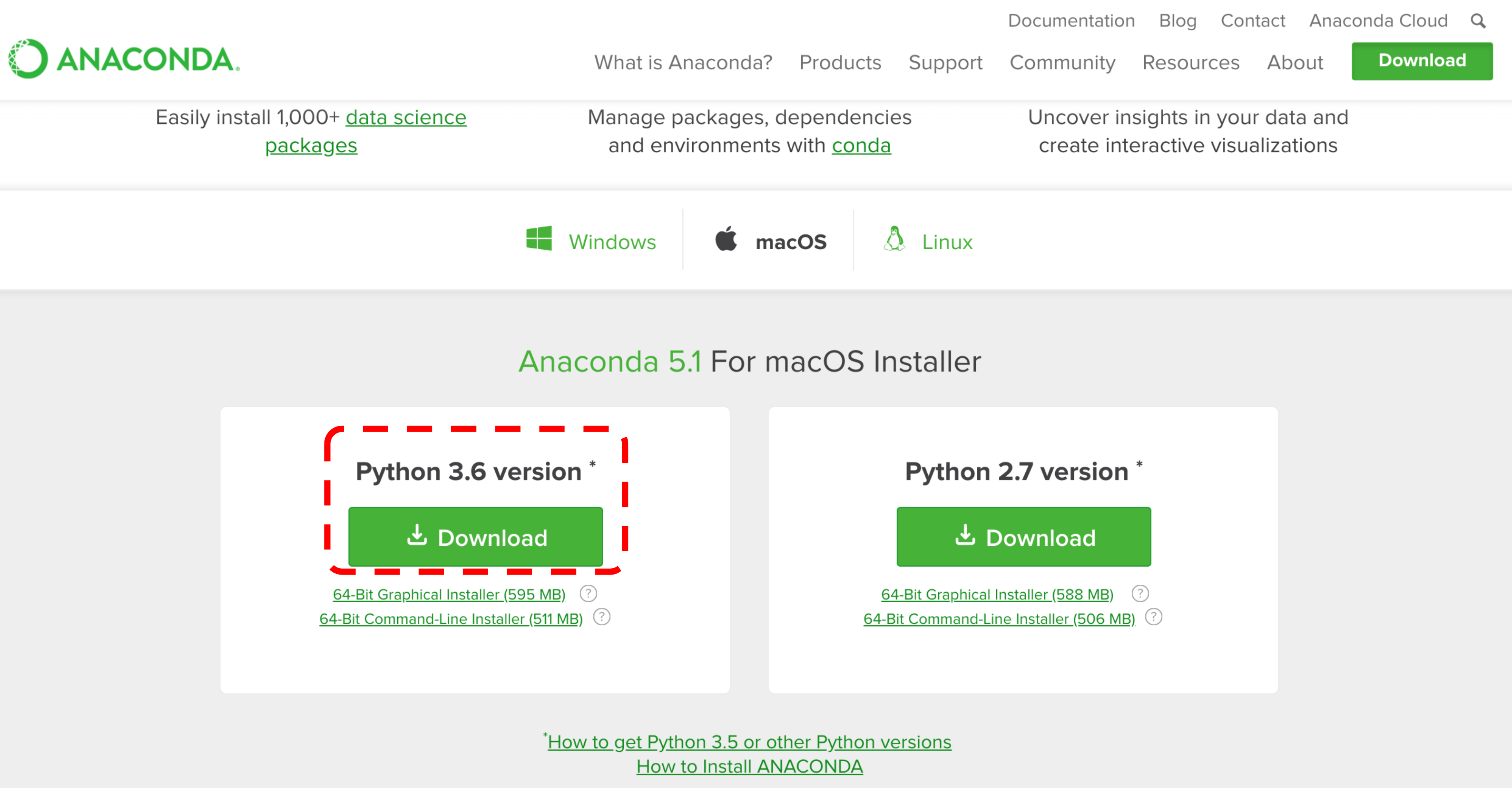
Source: https://www.anaconda.com/

Download Anaconda



https://www.anaconda.com/download

Download Anaconda
Python 3.6 version

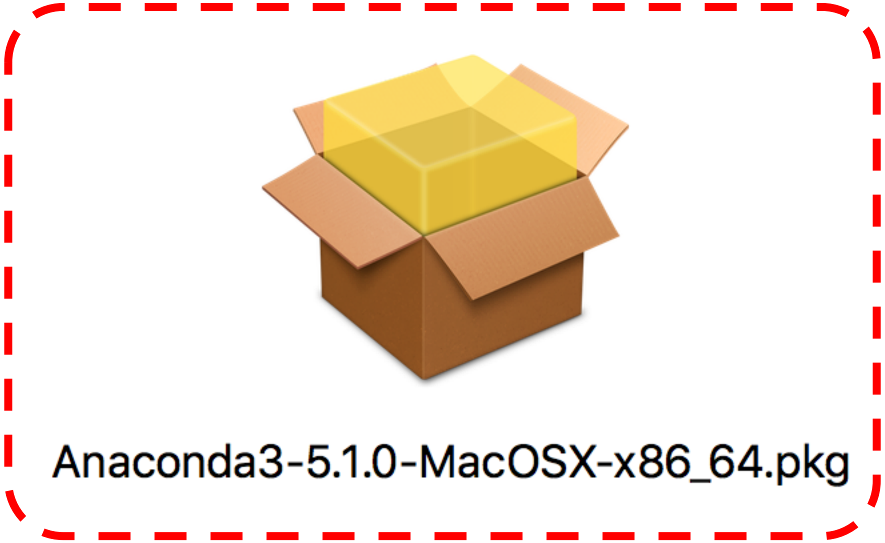


https://www.anaconda.com/download/#macos

Install Anaconda

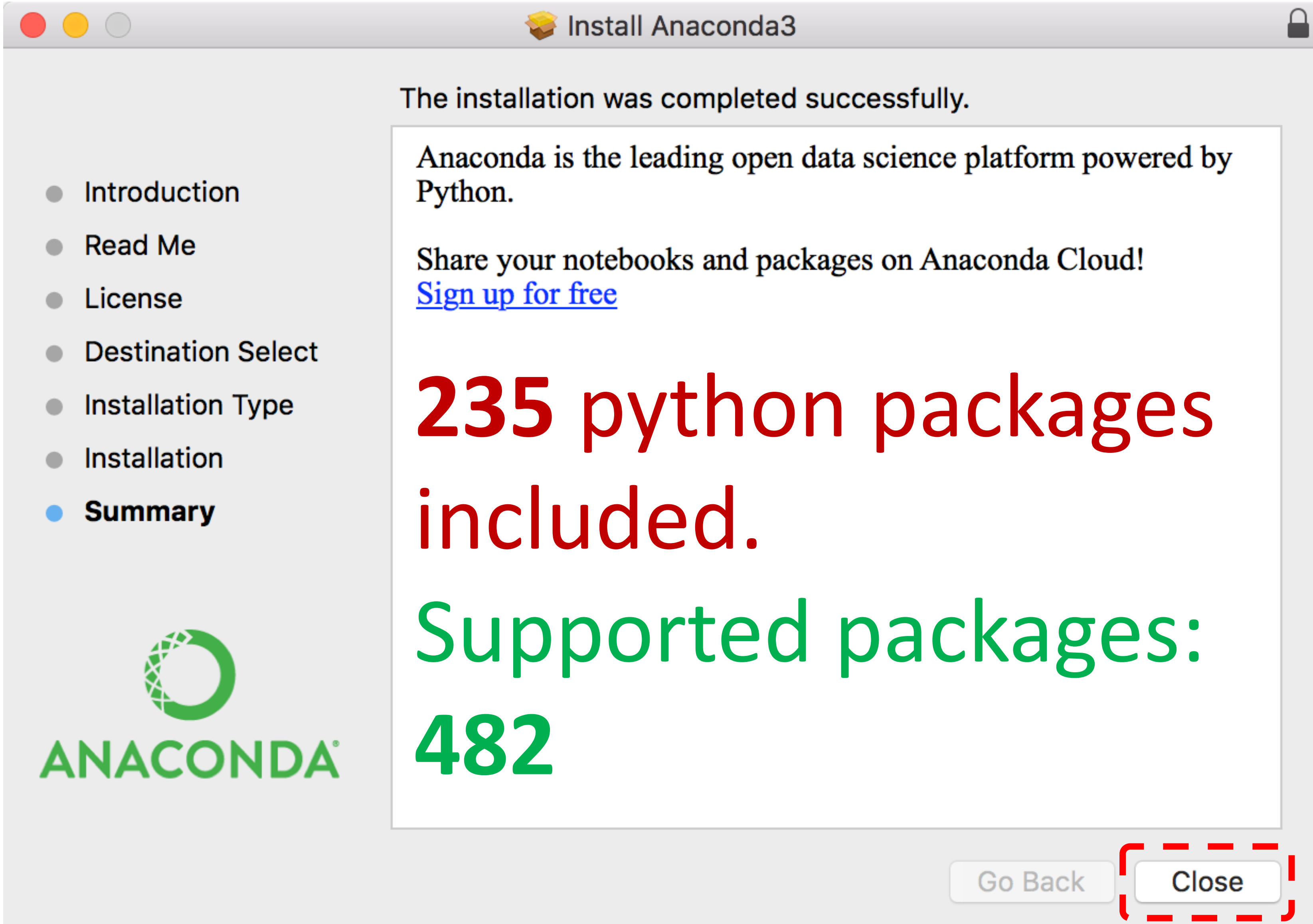
Python 3.6

Anaconda3-5.1.0-MacOSX-x86_64.pkg



Installer package - 623.6 MB

Install Anaconda 3



Source: https://docs.anaconda.com/anaconda/packages/pkg-docs

Install Anaconda 3

```
# Name
_ipyw_jlab_nb_ext_conf
alabaster
anaconda-client
anaconda-navigator
anaconda-project
appnope
appscript
asn1crypto
astroid
astropy
attrs
babel
backports
backports.shutil_get_term
beautifulsoup4
bitarray
bkcharts
blaze
bleach
bokeh
boto
bottleneck
bzip2
ca-certificates
certifi
cffi
chardet
click
cloudpickle
clyent
colorama
conda
conda-build
conda-env
conda-verify
contextlib2
cryptography
curl
cycler
cython
cytoolz
dask
dask-core
datashape
dbus
decorator
distributed
docutils
entrypoints
et_xmlfile
expat
fastcache
filelock
flask
flask-cors
freetype
get_terminal_size
gettext
gevent
glib
glob2
gmp
gmpy2
greenlet
h5py
hdf5
heapdict
html5lib
icu
idna
imageio
imagesize
intel-openmp
ipykernel
ipython
ipython_genutils
ipywidgets
isort
itsdangerous
jbig
jdcal
jedi
jinja2
jpeg
jsonschema
jupyter
jupyter_client
jupyter_console
jupyter_core
jupyterlab
jupyterlab_launcher
lazy-object-proxy
libcurl
libcxx
libcxxabi
libedit
libffi
libgfortran
libiconv
libpng
libsodium
libssh2
libtiff
libxml2
libxslt
llvmlite
locket
lxml
lzo
markupsafe
matplotlib
mccabe
mistune
mkl
mkl-service
mpc
mpfr
mpmath
msgpack-python
multipledispatch
navigator-updater
nbconvert
nbformat
ncurses
networkx
nltk
nose
notebook
numba
numexpr
numpy
numpydoc
odo
olefile
openpyxl
openssl
packaging
pandas
pandoc
pandocfilters
parso
partd
path.py
pathlib2
patsy
pcre
pep8
pexpect
pickleshare
pillow
pip
pkginfo
pluggy
ply
prompt_toolkit
psutil
ptyprocess
py
pycodestyle
pycosat
pycparser
pycrypto
pycurl
pyflakes
pygments
pylint
pyodbc
pyopenssl
pyparsing
pyqt
pysocks
pytables
pytest
python
python-dateutil
python.app
pytz
pywavelets
pyyaml
pyzmq
qt
qtawesome
qtconsole
qtpy
readline
requests
rope                        0.10.7
ruamel_yaml                 0.15.35
scikit-image                0.13.1
scikit-learn                0.19.1
scipy                       1.0.0
seaborn                     0.8.1
send2trash                  1.4.2
setuptools                  38.4.0
simplegeneric               0.8.1
singledispatch              3.4.0.3
sip                         4.18.1
six                         1.11.0
snowballstemmer             1.2.1
sortedcollections           0.5.3
sortedcontainers            1.5.9       py36_0
sphinx                      1.6.6       py36_0
sphinxcontrib               1.0         py36h9364dc8_1
sphinxcontrib-websupport    1.0.1       py36h92f4a7a_1
spyder                      3.2.6       py36_0
sqlalchemy                  1.2.1       py36h1de35cc_0
sqlite                      3.22.0      h3efe00b_0
statsmodels                 0.8.0       py36h9c68fc9_0
sympy                       1.1.1       py36h7f3cf04_0
tblib                       1.3.2       py36hda67792_0
terminado                   0.8.1       py36_1
testpath                    0.3.1       py36h625a49b_0
tk                          8.6.7       h35a86e2_3
toolz                       0.9.0       py36_0
tornado                     4.5.3       py36_0
traitlets                   4.3.2       py36h65bd3ce_0
typing                      3.6.2       py36haa2d9ef_0
unicodecsv                  0.14.1      py36he531d66_0
unixodbc                    2.3.4       h4cb4dde_1
urllib3                     1.22        py36h68b9469_0
wcwidth                     0.1.7       py36h8c6ec74_0
webencodings                0.5.1       py36h3b9701d_1
werkzeug                    0.14.1      py36_0
wheel                       0.30.0      py36h5eb2c71_1
widgetsnbextension          3.1.0       py36_0
wrapt                       1.10.11     py36hc29e774_0
xlrd                        1.1.0       py36h336f4a2_1
xlsxwriter                  1.0.2       py36h3736301_0
xlwings                     0.11.5      py36_0
xlwt                        1.2.0       py36h5ad1178_0
xz                          5.2.3       h0278029_2
yaml                                    4_2
zeromq                                  0_2
zict                                    4_0
zlib                                    b_2
```

235 python packages included.

Verify that conda is installed, check current conda version

- **`conda --version`**

- Update conda to the current version
  - **`conda update conda`**

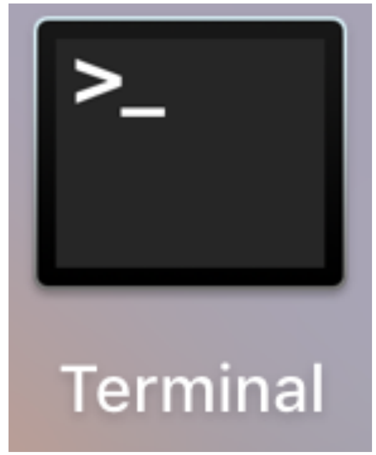
http://conda.pydata.org/docs/using/using.html#verify-that-conda-is-installed-check-current-conda-version

Check current conda version
Check current python version
Check conda environments

- `conda --version`
- `python --version`
- `conda info --envs`

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

Terminal




```
terminal
```

conda list

imyday — -bash — 80×24

```
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
appnope                   0.1.0                    py36_0
appscript                 1.0.1                    py36_0
astroid                   1.4.9                    py36_0
astropy                   1.3                 np111py36_0
babel                     2.3.4                    py36_0
backports                 1.0                      py36_0
beautifulsoup4            4.5.3                    py36_0
bitarray                  0.8.1                    py36_0
blaze                     0.10.1                   py36_0
bokeh                     0.12.4                   py36_0
boto                      2.45.0                   py36_0
bottleneck                1.2.0               np111py36_0
cffi                      1.9.1                    py36_0
chardet                   2.3.0                    py36_0
chest                     0.2.3                    py36_0
```

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

python --version

imyday — -bash — 80×24

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

conda --version

imyday — -bash — 80×24

```
iMyday-MacBook-Pro:~ imyday$ python --version
Python 3.6.0 :: Anaconda 4.3.1 (x86_64)
iMyday-MacBook-Pro:~ imyday$ conda --version
conda 4.3.14
iMyday-MacBook-Pro:~ imyday$ conda info --envs
# conda environments:
#
py27                     /Users/imyday/anaconda/envs/py27
py35                     /Users/imyday/anaconda/envs/py35
root                  *  /Users/imyday/anaconda

iMyday-MacBook-Pro:~ imyday$ source activate py35
(py35) iMyday-MacBook-Pro:~ imyday$ python --version
Python 3.5.3 :: Continuum Analytics, Inc.
(py35) iMyday-MacBook-Pro:~ imyday$ conda --version
conda 4.3.14
(py35) iMyday-MacBook-Pro:~ imyday$ source deactivate py35
iMyday-MacBook-Pro:~ imyday$ conda info --envs
# conda environments:
#
py27                     /Users/imyday/anaconda/envs/py27
py35                     /Users/imyday/anaconda/envs/py35
root                  *  /Users/imyday/anaconda
```

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

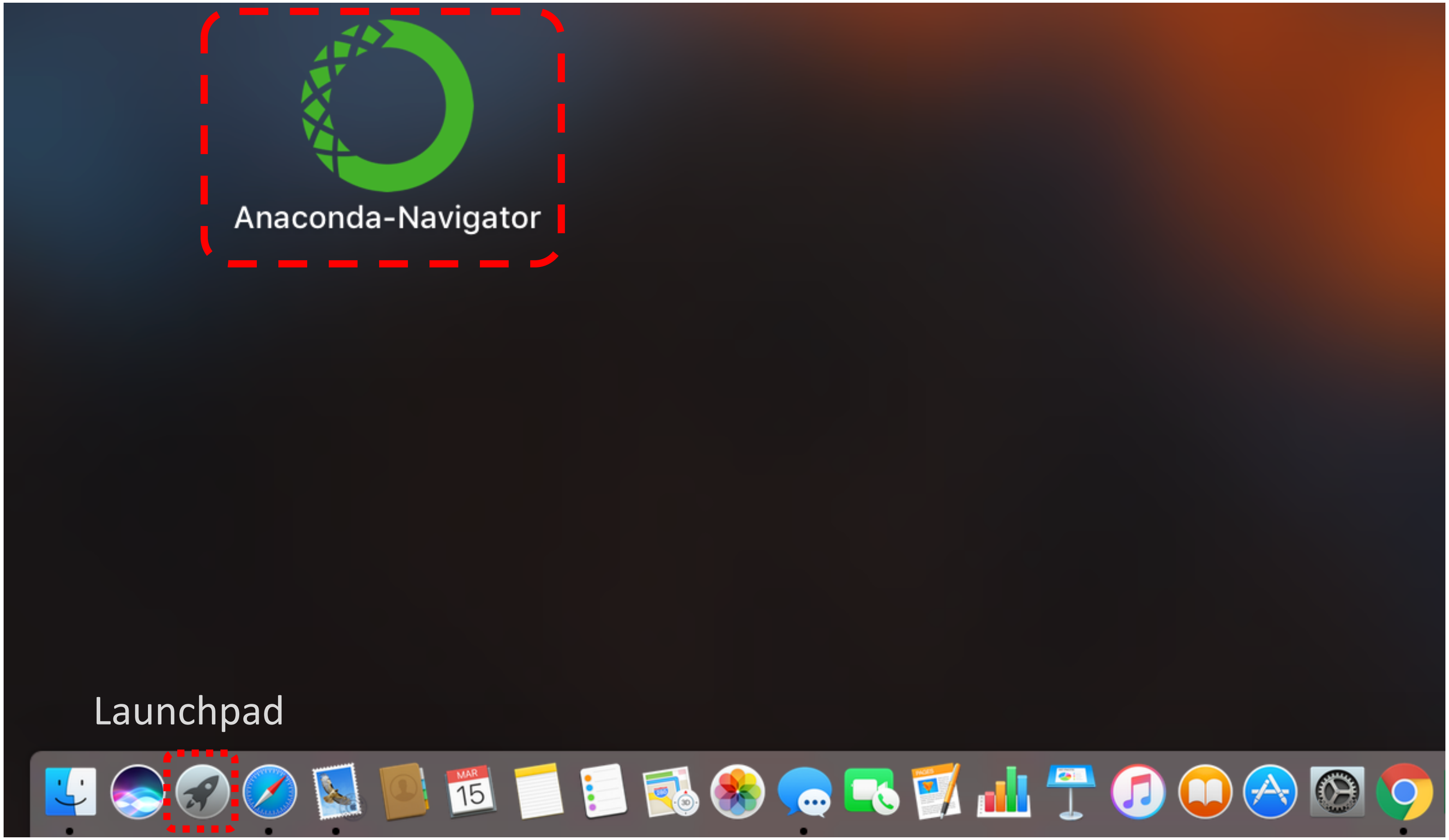
source activate py35

source deactivate py35

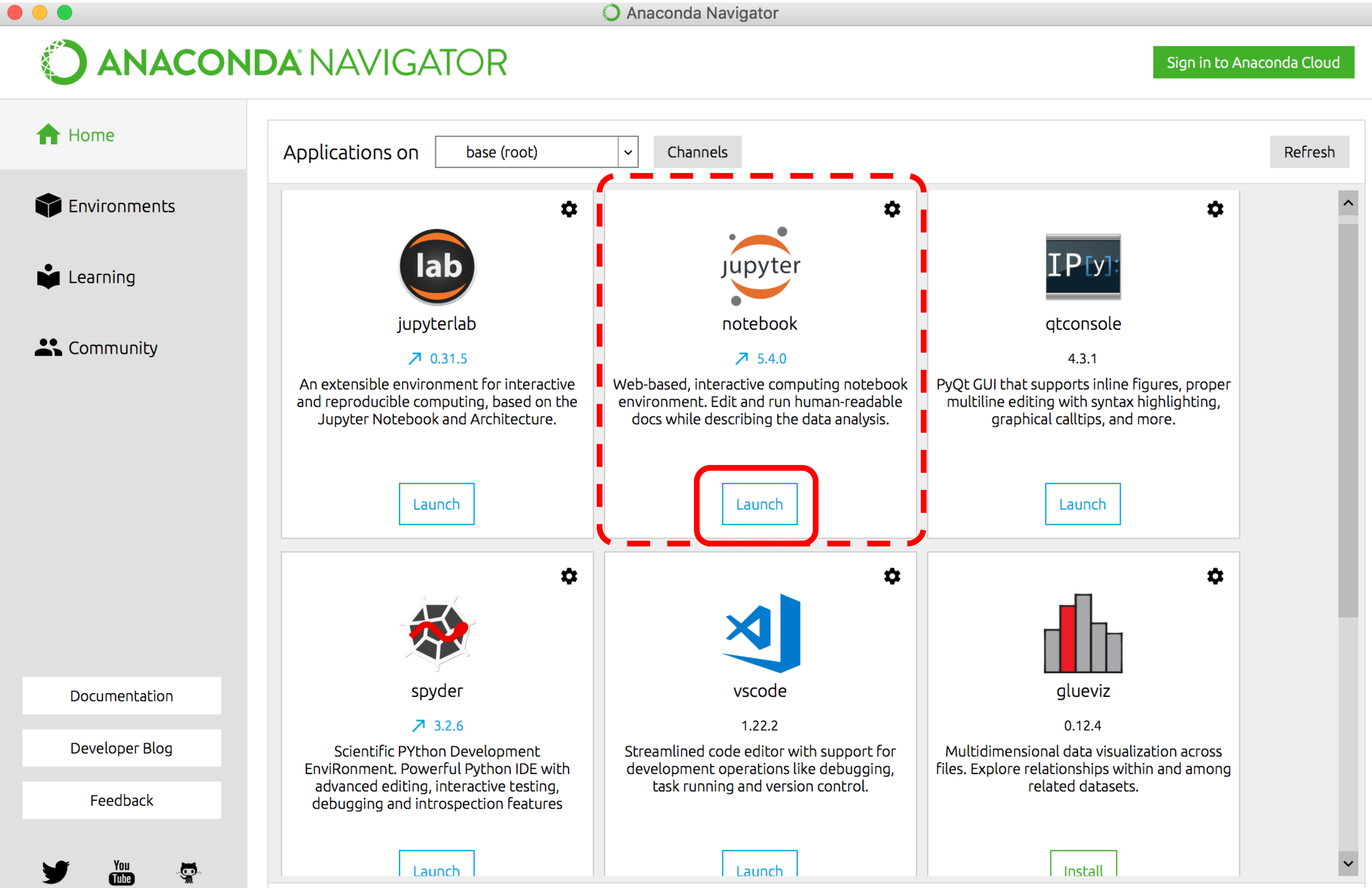
Python

HelloWorld

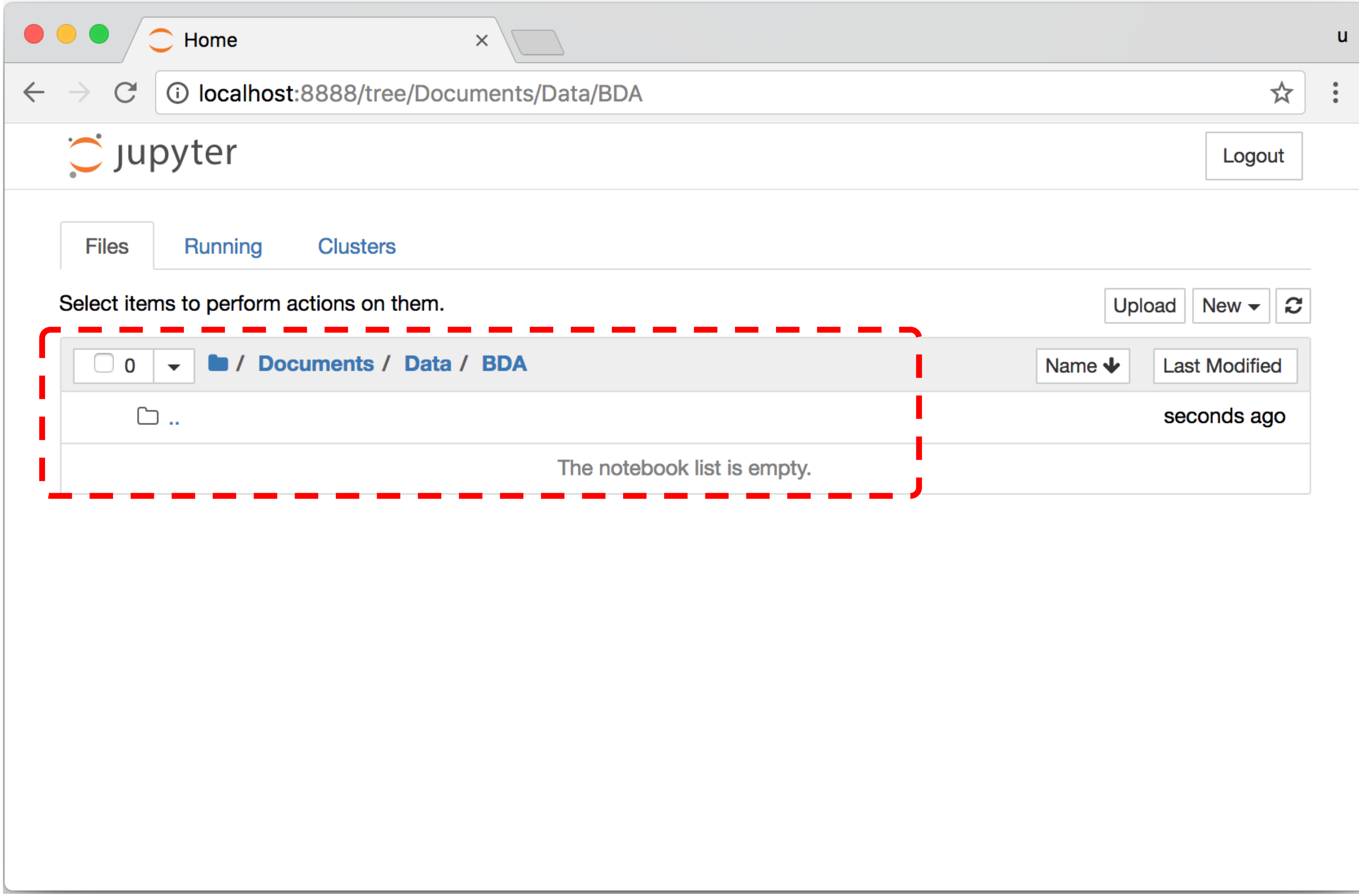
Anaconda-Navigator



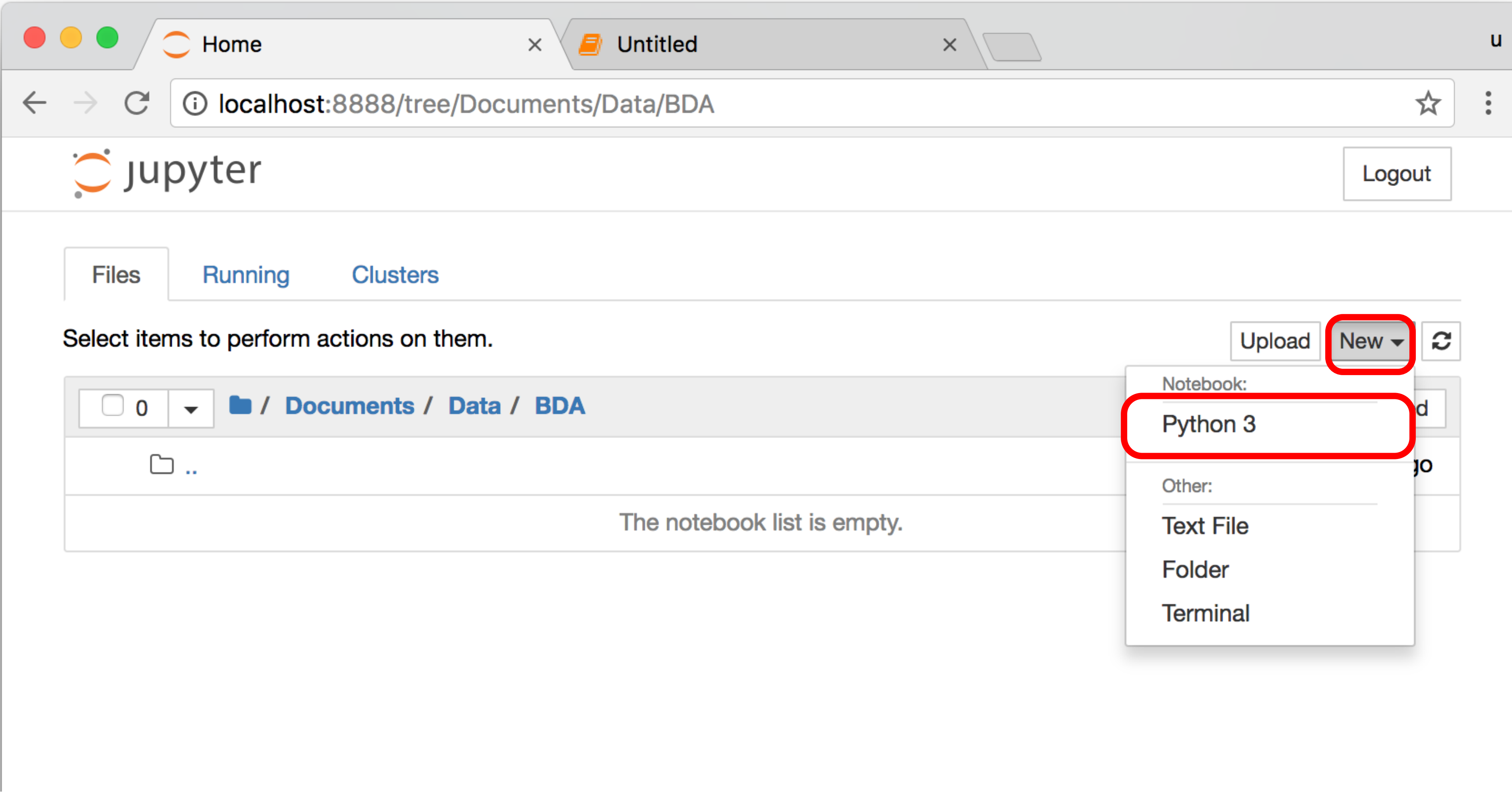
Anaconda Navigator



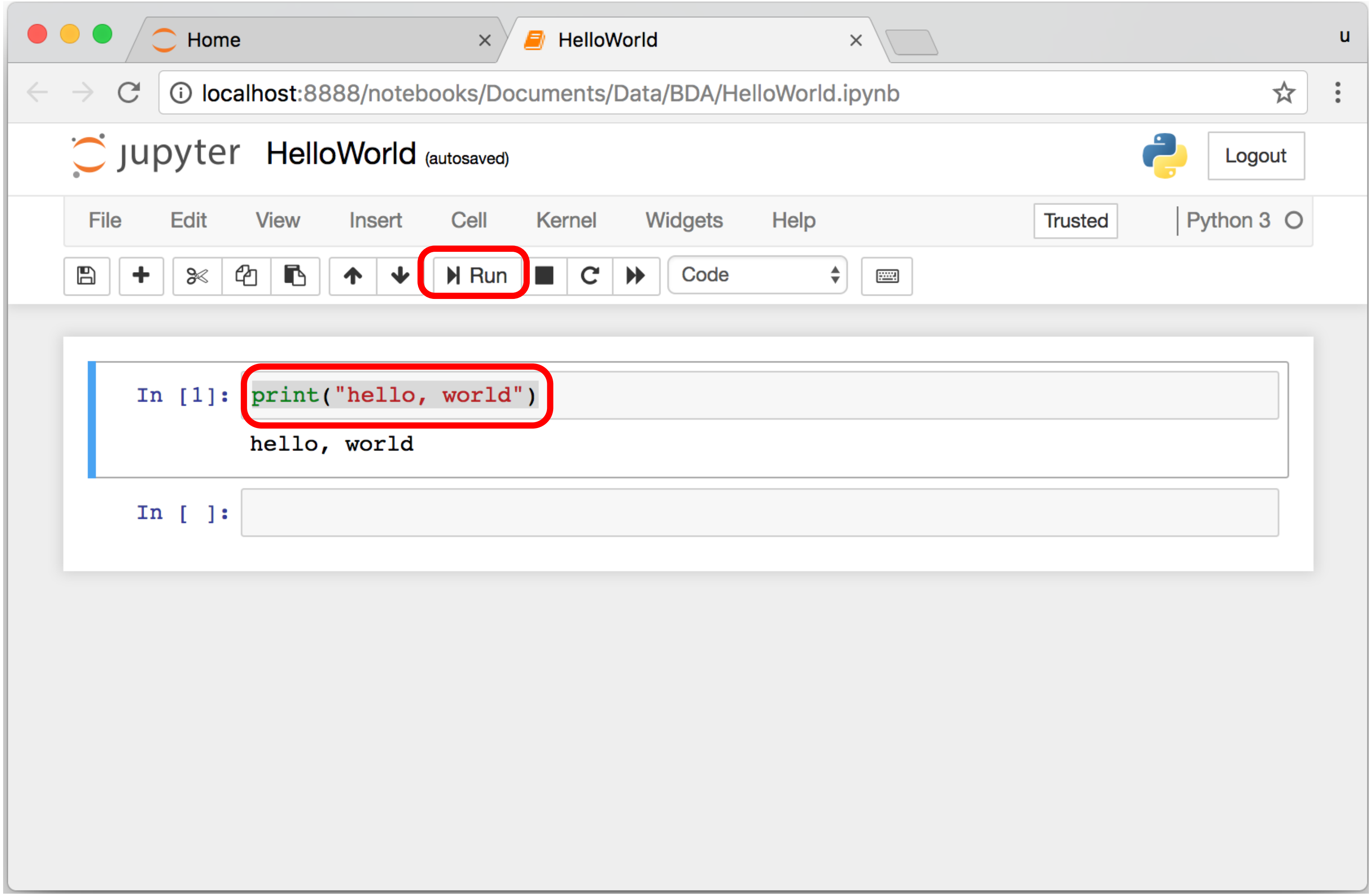
Jupyter Notebook



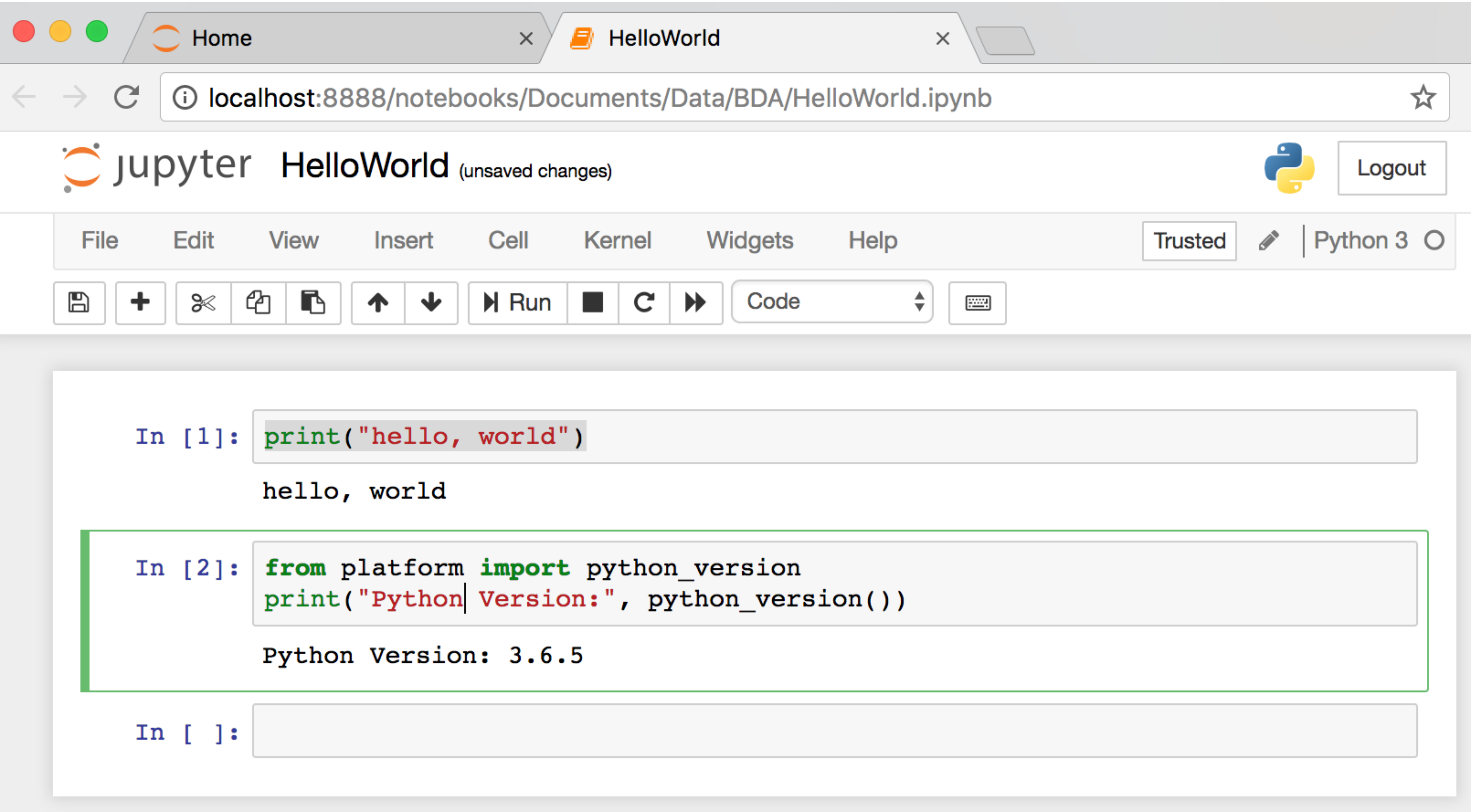
Jupyter Notebook
New Python 3



`print("hello, world")`



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

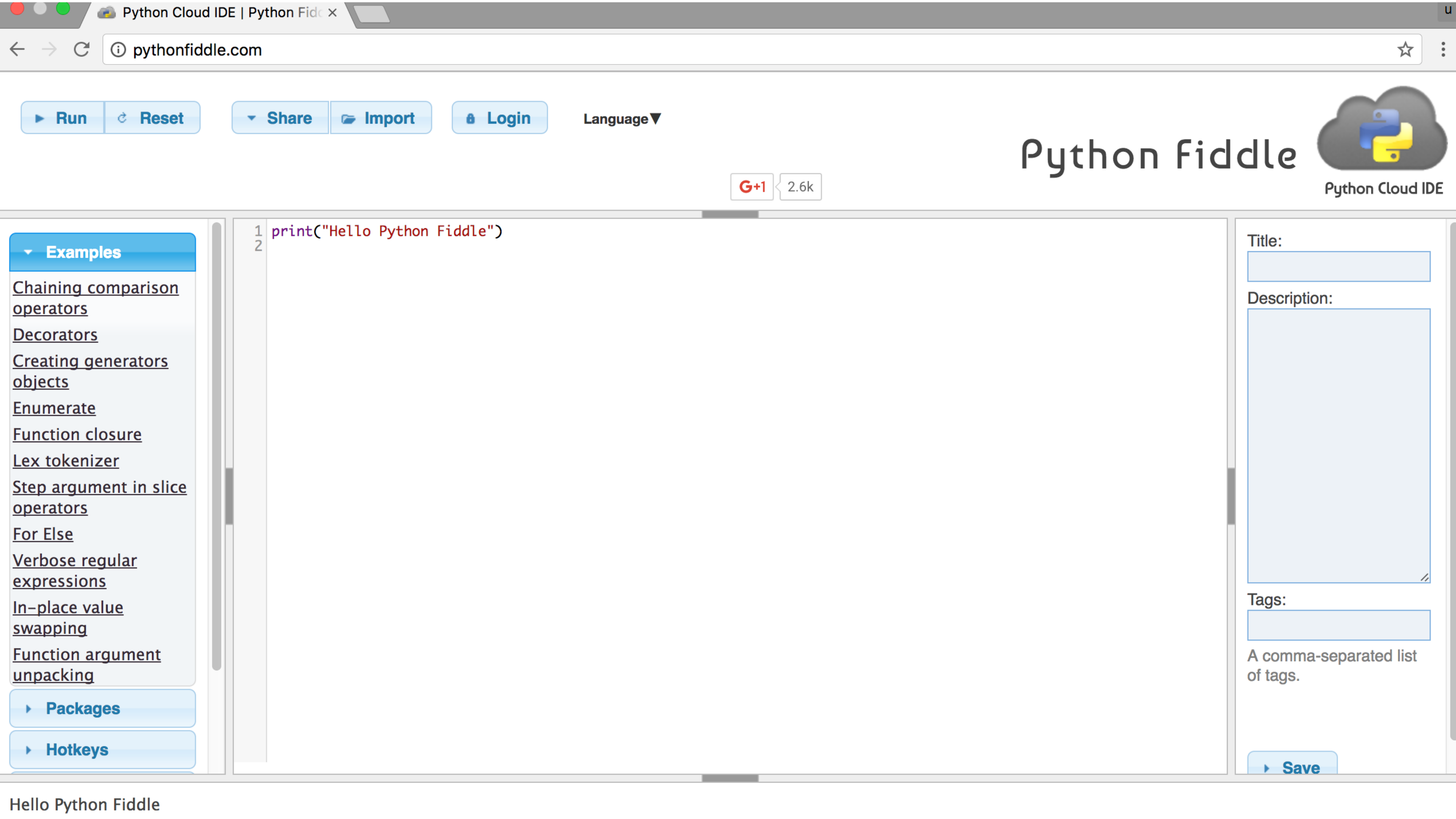


Python

Programming

Source: https://www.python.org/community/logos/

Python Fiddle



http://pythonfiddle.com/

python™

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 "))
```

Source: http://pythonprogramminglanguage.com/text-input-and-output/

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

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

```
hello, world
```

[2]

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

```
Python Version: 3.6.6
```

[3]

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

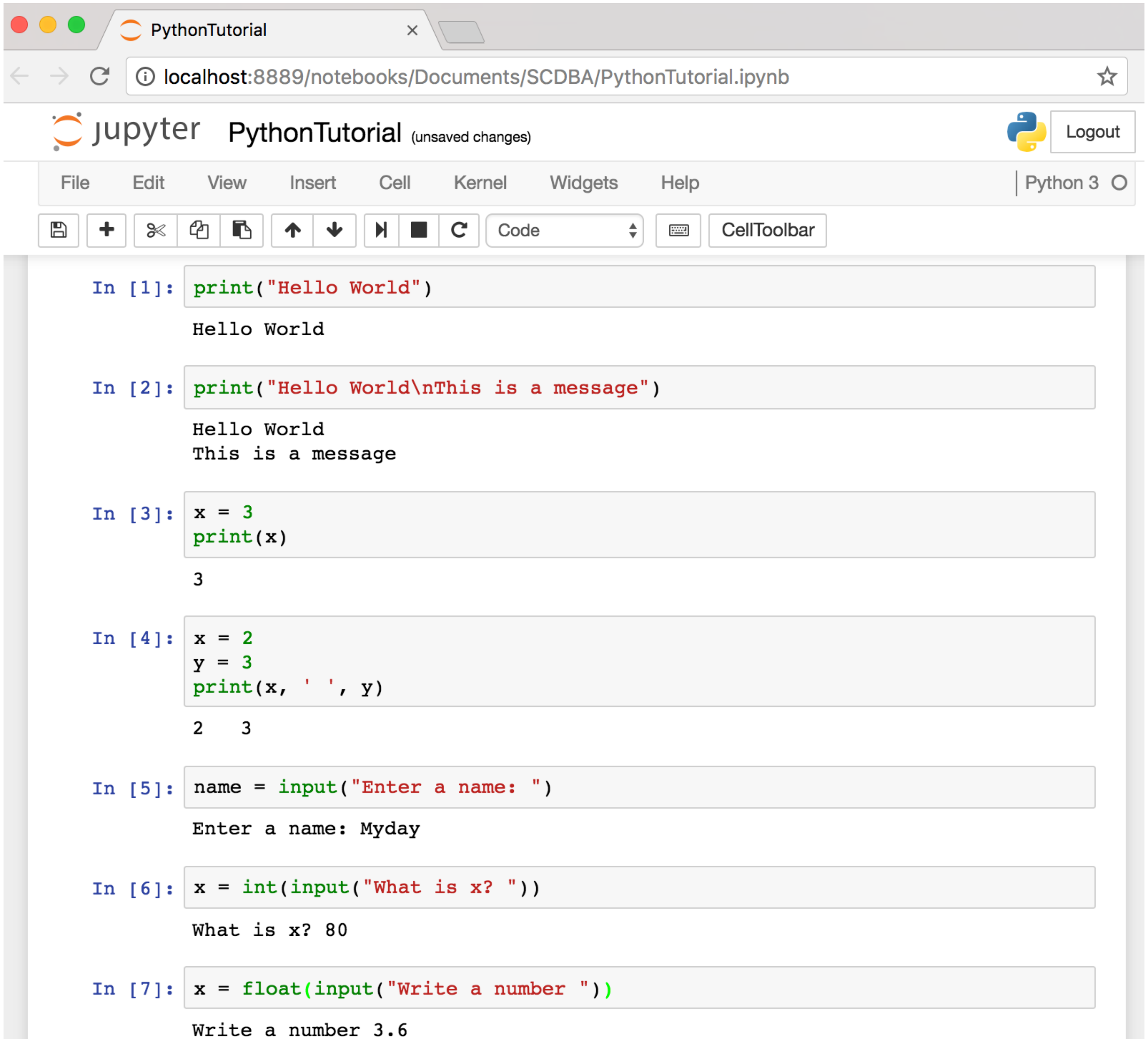
```
Hello World
Hello World
This is a message
3
2   3
```

[4]

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

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

Text input and output



Source: http://pythonprogramminglanguage.com/text-input-and-output/

python™

Variables

```
x = 2
price = 2.5
word = 'Hello'
```

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

```
x = 2
x = x + 1
x = 5
```

Source: http://pythonprogramminglanguage.com/

python™

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)))
```

Source: http://code.activestate.com/recipes/580615-bmi-code/

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]:

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

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

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

```
Enter your height in cm: 170
Enter your weight in kg: 60
Your BMI is: 20.8
```

In [ ]:

Source: http://code.activestate.com/recipes/580615-bmi-code/

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

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

Future Value (FV)

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

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

```
194.87171000000012
```

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

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
```

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

if statements

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

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

```
Pass
```

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

Source: http://pythonprogramminglanguage.com/

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

Source: http://pythonprogramminglanguage.com/

for loops

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

```
1
2
3
4
5
6
7
8
9
10
```

Source: http://pythonprogramminglanguage.com/

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
```

Source: http://pythonprogramminglanguage.com/

python™

while loops

```
age = 10

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

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

Source: https://learnpython.trinket.io/learn-python-part-8-loops#/while-loops/about-while-loops

Functions

```python
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.

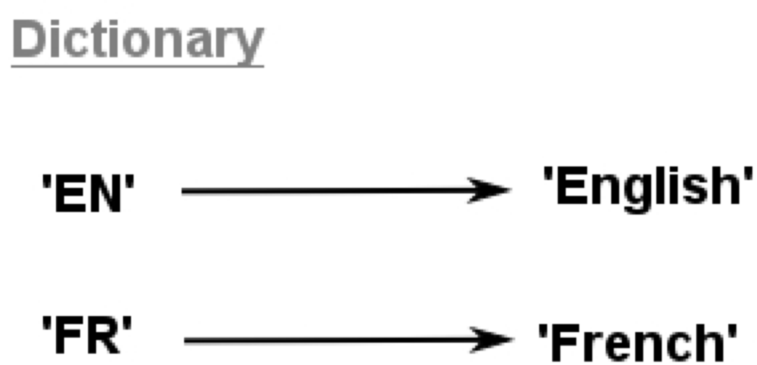
```
x = (10, 20, 30, 40, 50)
print(x[0])
print(x[1])
print(x[2])
print(x[-1])
```

```
10
20
30
50
```

Source: http://pythonprogramminglanguage.com/tuples/

Dictionary

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




```
English
```

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

Sets

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

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

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

```
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'
```

Source: https://github.com/TiesdeKok/LearnPythonforResearch/blob/master/0_python_basics.ipynb

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
```

Source: https://github.com/TiesdeKok/LearnPythonforResearch/blob/master/0_python_basics.ipynb

```python
# Python Strings indexing
word = 'Python'
print(word[0])  # character in position 0
#'P'
print(word[5])
#'n'
print(word[-1])
#'n'
print(word[-2])
#'o'
print(word[-6])
#'P'
```

```python
# Python Strings indexing
word = 'Python'
print(word[0])  # character in position 0
#'P'
print(word[5])  # character in position 5
#'n'
print(word[-1])  # last character
#'n'
print(word[-2])  # second-last character
#'o'
print(word[-6])
#'P'
```

```
P
n
n
o
P
```

```python
t1 = 'The mouse ran up the clock'
t2 = 'The mouse ran down'
s1 = t1.lower().split(' ')
s2 = t2.lower().split(' ')
terms = s1 + s2
sortedset = sorted(set(terms))
print('terms =', terms)
print('sortedset =', sortedset)
```


```python
t1 = 'The mouse ran up the clock'
t2 = 'The mouse ran down'
s1 = t1.lower().split(' ')
s2 = t2.lower().split(' ')
terms = s1 + s2
sortedset = sorted(set(terms))
print('terms =', terms)
print('sortedset =', sortedset)
```


```
terms = ['the', 'mouse', 'ran', 'up', 'the', 'clock', 'the', 'mouse', 'ran', 'down']
sortedset = ['clock', 'down', 'mouse', 'ran', 'the', 'up']
```

```
t1 = 'The mouse ran up the clock'
t2 = 'The mouse ran down'
s1 = t1.lower().split(' ')
s2 = t2.lower().split(' ')
terms = s1 + s2
print(terms)

tfdict = {}
for term in terms:
    if term not in tfdict:
        tfdict[term] = 1
    else:
        tfdict[term] += 1

a = []
for k,v in tfdict.items():
    a.append('{}, {}'.format(k,v))
print(a)
```

```
['the', 'mouse', 'ran', 'up', 'the', 'clock', 'the', 'mouse', 'ran', 'down']
['the, 3', 'mouse, 2', 'ran, 2', 'up, 1', 'clock, 1', 'down, 1']
```

```
sorted_by_value_reverse = sorted(tfdict.items(),
key=lambda kv: kv[1], reverse=True)

sorted_by_value_reverse_dict =
dict(sorted_by_value_reverse)

id2word = {id: word for id, word in
enumerate(sorted_by_value_reverse_dict)}

word2id = dict([(v, k) for (k, v) in
id2word.items()])
```

```
sorted_by_value:  [('up', 1), ('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3)]
sorted_by_value2:  ['the', 'mouse', 'ran', 'up', 'clock', 'down']
sorted_by_value_reverse:  [('the', 3), ('mouse', 2), ('ran', 2), ('up', 1), ('clock', 1), ('down', 1)]
sorted_by_value_reverse_dict {'the': 3, 'mouse': 2, 'ran': 2, 'up': 1, 'clock': 1, 'down': 1}
id2word {0: 'the', 1: 'mouse', 2: 'ran', 3: 'up', 4: 'clock', 5: 'down'}
word2id {'the': 0, 'mouse': 1, 'ran': 2, 'up': 3, 'clock': 4, 'down': 5}
len_words: 6
sorted_by_key:  [('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3), ('up', 1)]
the, 3
mouse, 2
ran, 2
up, 1
clock, 1
down, 1
```

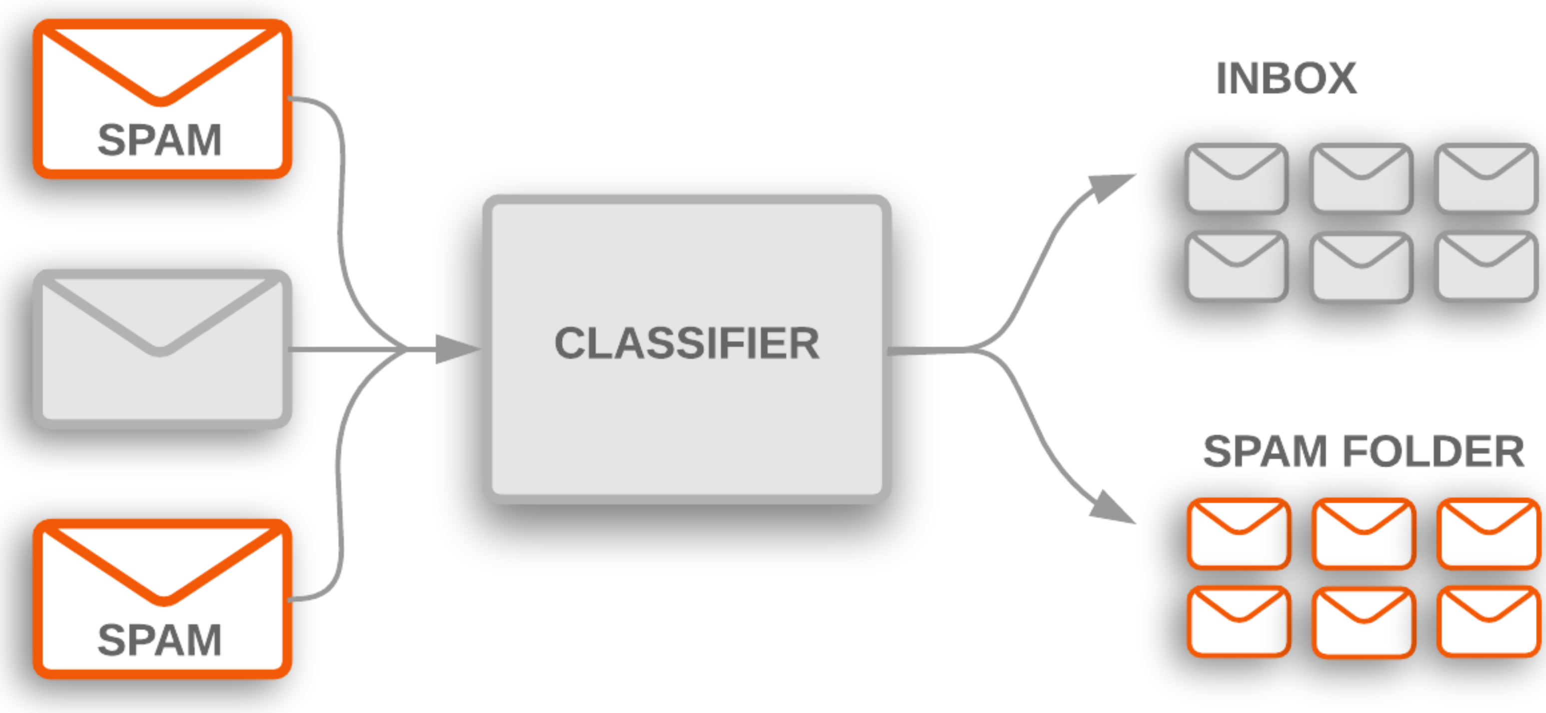
```
sorted_by_value = sorted(tfdict.items(), key=lambda kv: kv[1])
print('sorted_by_value: ', sorted_by_value)
sorted_by_value2 = sorted(tfdict, key=tfdict.get, reverse=True)
print('sorted_by_value2: ', sorted_by_value2)
sorted_by_value_reverse = sorted(tfdict.items(), key=lambda kv: kv[1], reverse=True)
print('sorted_by_value_reverse: ', sorted_by_value_reverse)
sorted_by_value_reverse_dict = dict(sorted_by_value_reverse)
print('sorted_by_value_reverse_dict', sorted_by_value_reverse_dict)
id2word = {id: word for id, word in enumerate(sorted_by_value_reverse_dict)}
print('id2word', id2word)
word2id = dict([(v, k) for (k, v) in id2word.items()])
print('word2id', word2id)
print('len_words:', len(word2id))

sorted_by_key = sorted(tfdict.items(), key=lambda kv: kv[0])
print('sorted_by_key: ', sorted_by_key)

tfstring = '\n'.join(a)
print(tfstring)
tf = tfdict.get('mouse')
print(tf)
```

```
sorted_by_value:  [('up', 1), ('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3)]
sorted_by_value2:  ['the', 'mouse', 'ran', 'up', 'clock', 'down']
sorted_by_value_reverse:  [('the', 3), ('mouse', 2), ('ran', 2), ('up', 1), ('clock', 1), ('down', 1)]
sorted_by_value_reverse_dict {'the': 3, 'mouse': 2, 'ran': 2, 'up': 1, 'clock': 1, 'down': 1}
id2word {0: 'the', 1: 'mouse', 2: 'ran', 3: 'up', 4: 'clock', 5: 'down'}
word2id {'the': 0, 'mouse': 1, 'ran': 2, 'up': 3, 'clock': 4, 'down': 5}
len_words: 6
sorted_by_key:  [('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3), ('up', 1)]
the, 3
mouse, 2
ran, 2
up, 1
clock, 1
down, 1
```

Text Classification



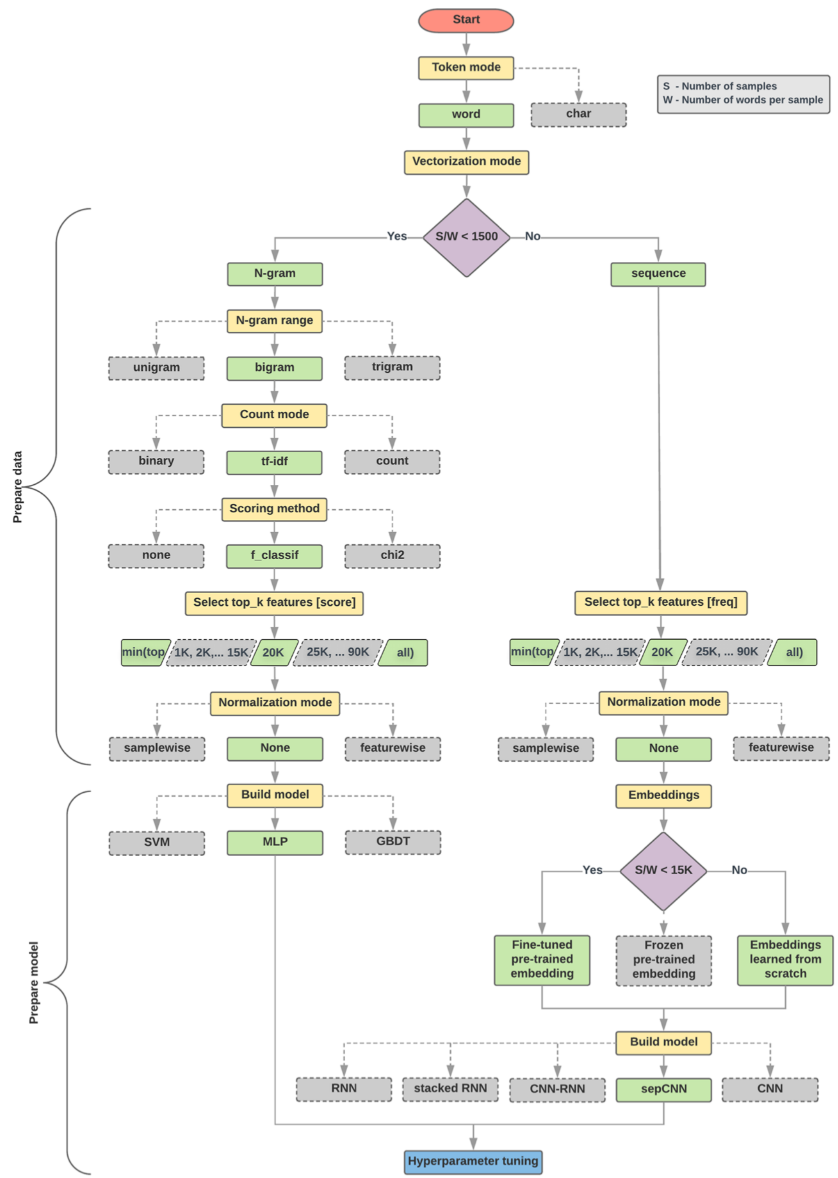
Source: https://developers.google.com/machine-learning/guides/text-classification/

Text Classification Workflow

- Step 1: Gather Data
- Step 2: Explore Your Data
- Step 2.5: Choose a Model*
- Step 3: Prepare Your Data
- Step 4: Build, Train, and Evaluate Your Model
- Step 5: Tune Hyperparameters
- Step 6: Deploy Your Model

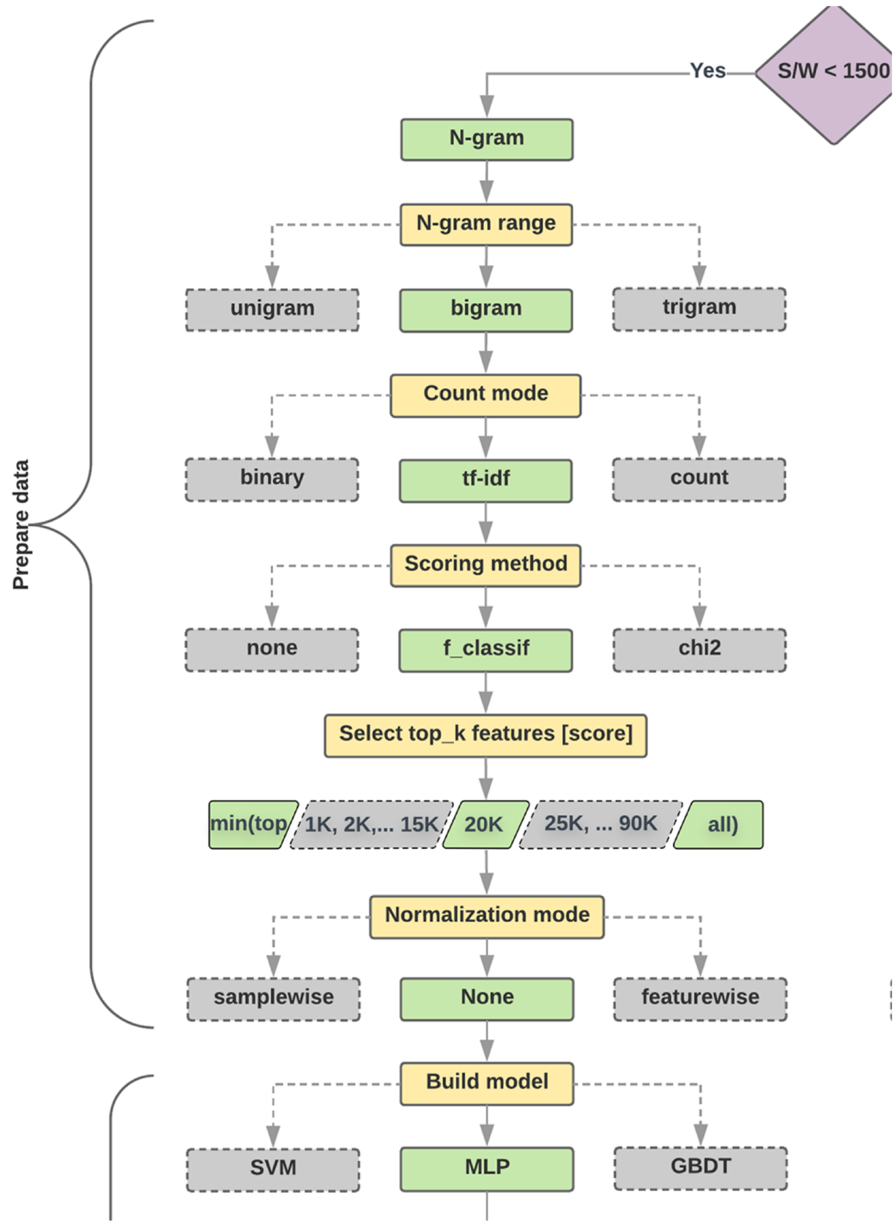
Source: https://developers.google.com/machine-learning/guides/text-classification/

Text Classification Flowchart



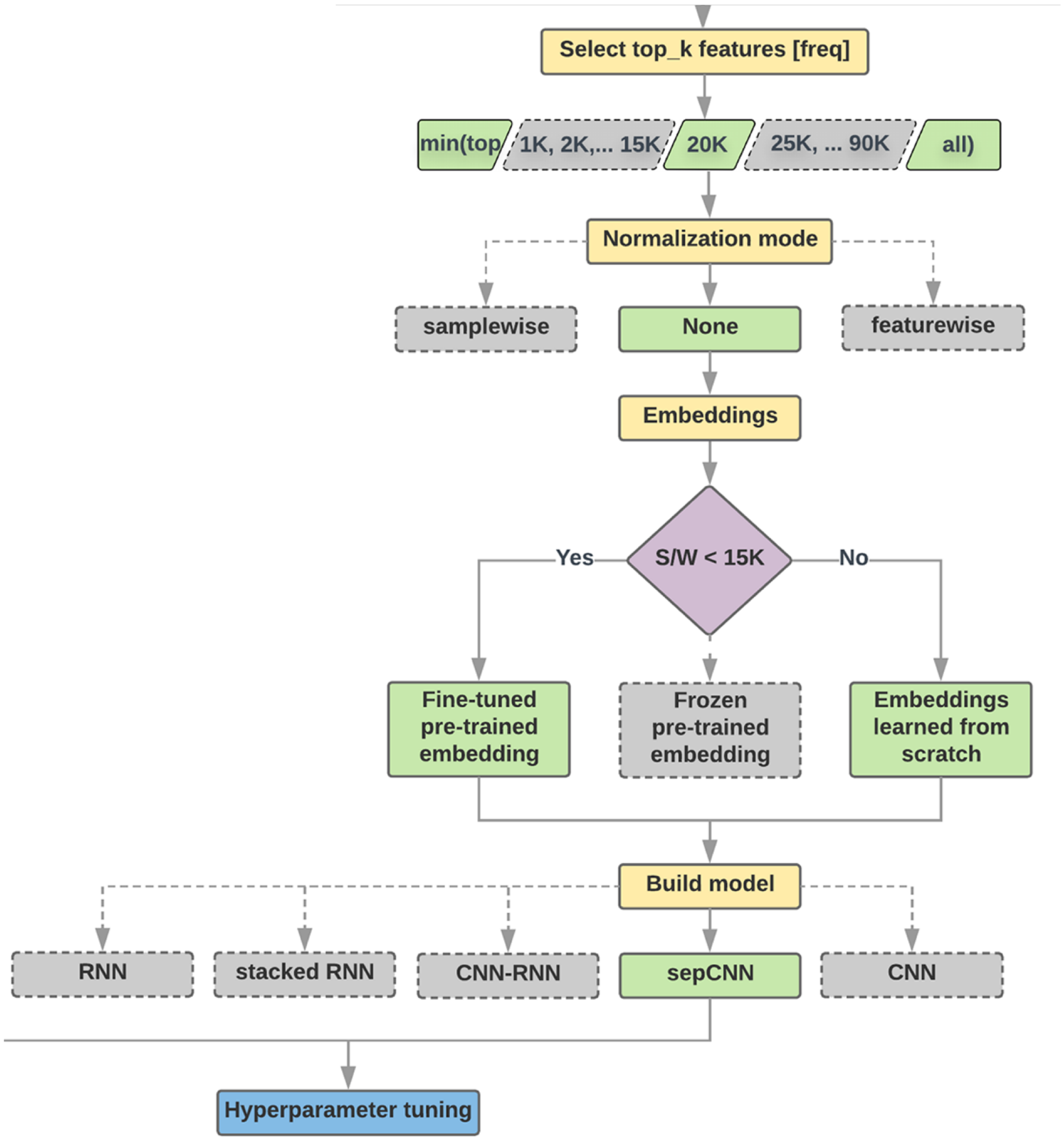
Source: https://developers.google.com/machine-learning/guides/text-classification/step-2-5

Text Classification S/W<1500: N-gram



Source: https://developers.google.com/machine-learning/guides/text-classification/step-2-5

Text Classification S/W>=1500: Sequence

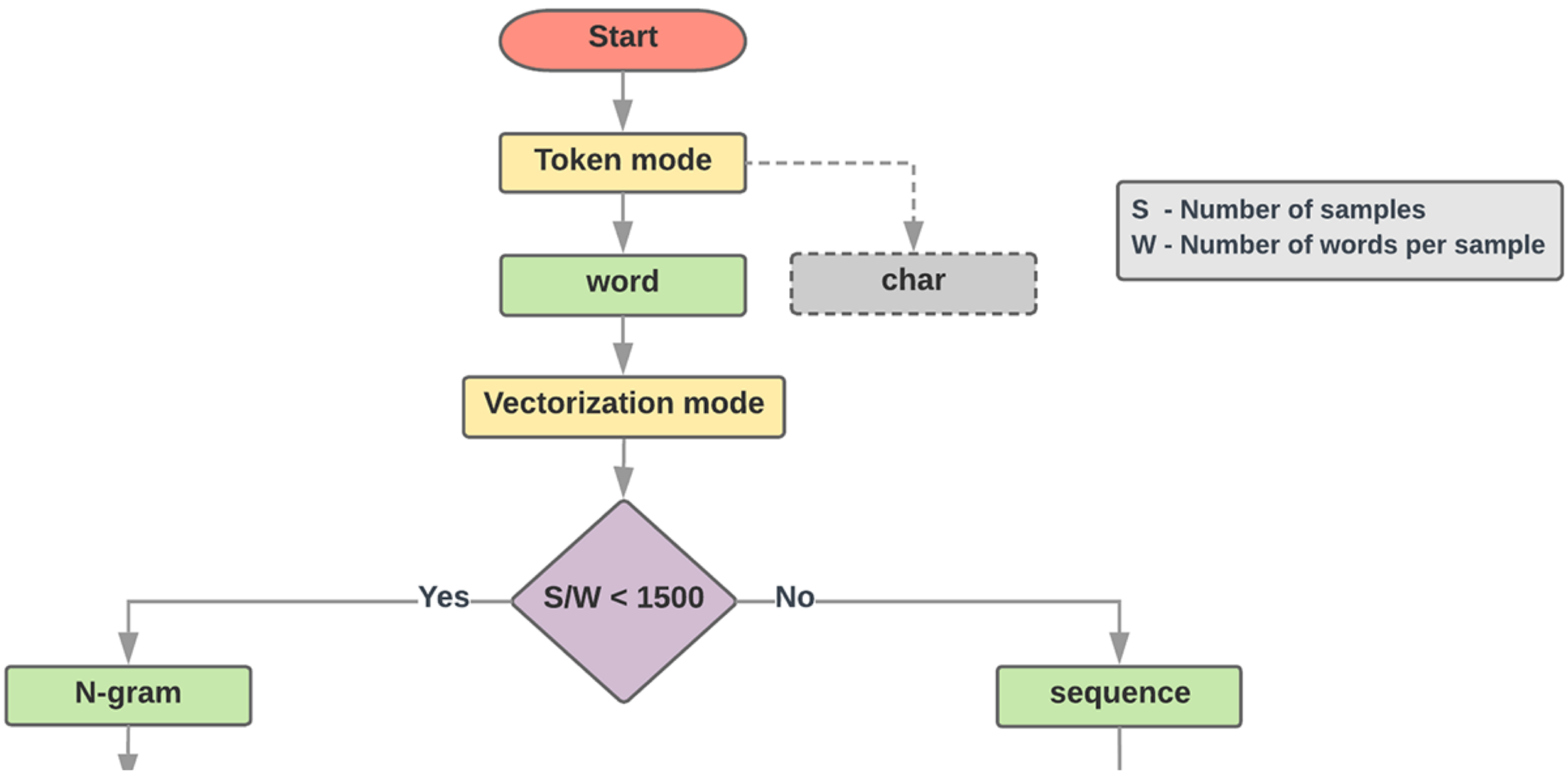


Source: https://developers.google.com/machine-learning/guides/text-classification/step-2-5

Step 2.5: Choose a Model

Samples/Words < 1500

150,000/100 = 1500



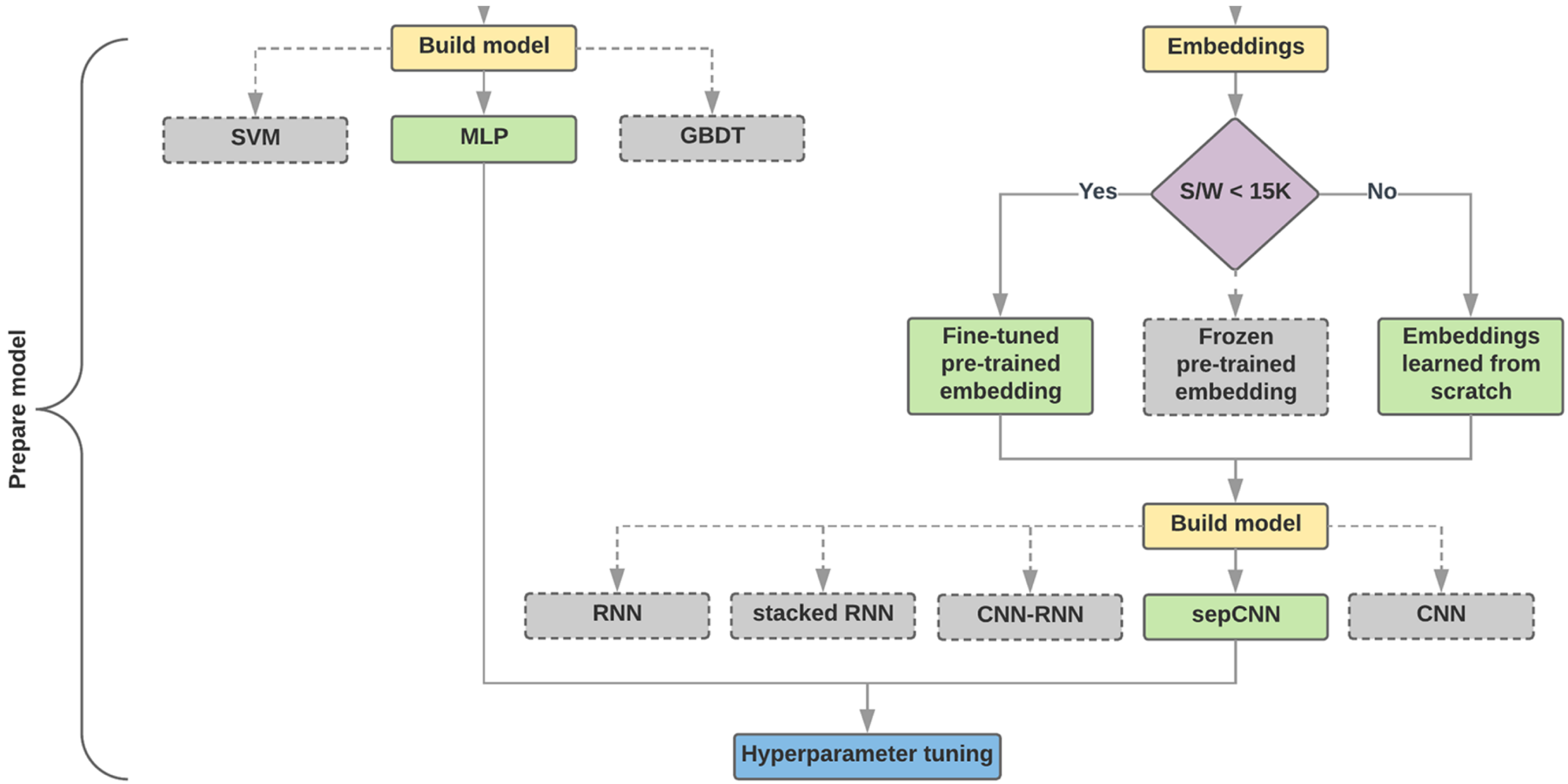
IMDb review dataset,
the samples/words-per-sample ratio is ~ 144

Source: https://developers.google.com/machine-learning/guides/text-classification/step-2-5

Step 2.5: Choose a Model

Samples/Words < 15,000

1,500,000/100 = 15,000



Source: https://developers.google.com/machine-learning/guides/text-classification/step-2-5

Step 3: Prepare Your Data

```
Texts:
T1: ’The mouse ran up the clock'
T2: ’The mouse ran down’

Token Index:
{'the': 1, 'mouse’: 2, 'ran': 3, 'up': 4, 'clock': 5, 'down': 6,}.
    NOTE: 'the' occurs most frequently,
            so the index value of 1 is assigned to it.
            Some libraries reserve index 0 for unknown tokens,
            as is the case here.

Sequence of token indexes:
T1: ‘The mouse ran up the clock’ =
        [1,  2,   3, 4,  1,  5]
T1: ’The mouse ran down’ =
        [1,   2,   3,  6]
```

Source: https://developers.google.com/machine-learning/guides/text-classification/step-3

One-hot encoding

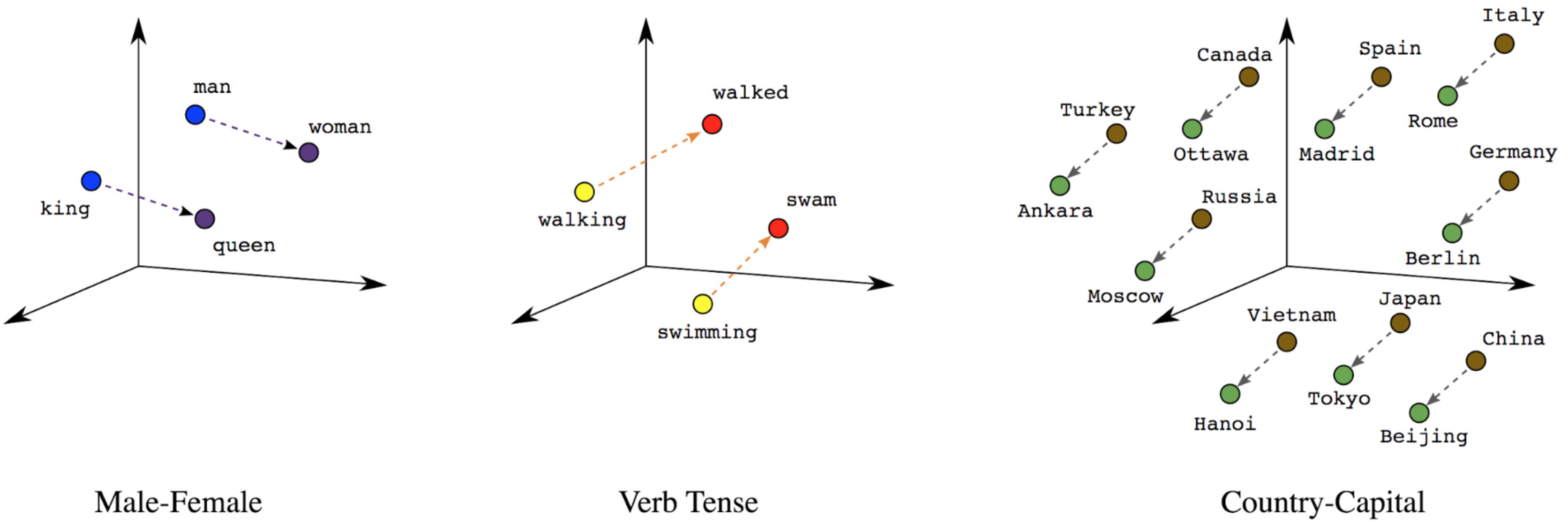
```
'The mouse ran up the clock' =

The      1   [ [0, 1, 0, 0, 0, 0, 0],
mouse    2     [0, 0, 1, 0, 0, 0, 0],
ran      3     [0, 0, 0, 1, 0, 0, 0],
up       4     [0, 0, 0, 0, 1, 0, 0],
the      1     [0, 1, 0, 0, 0, 0, 0],
clock    5     [0, 0, 0, 0, 0, 1, 0] ]

               [0, 1, 2, 3, 4, 5, 6]
```

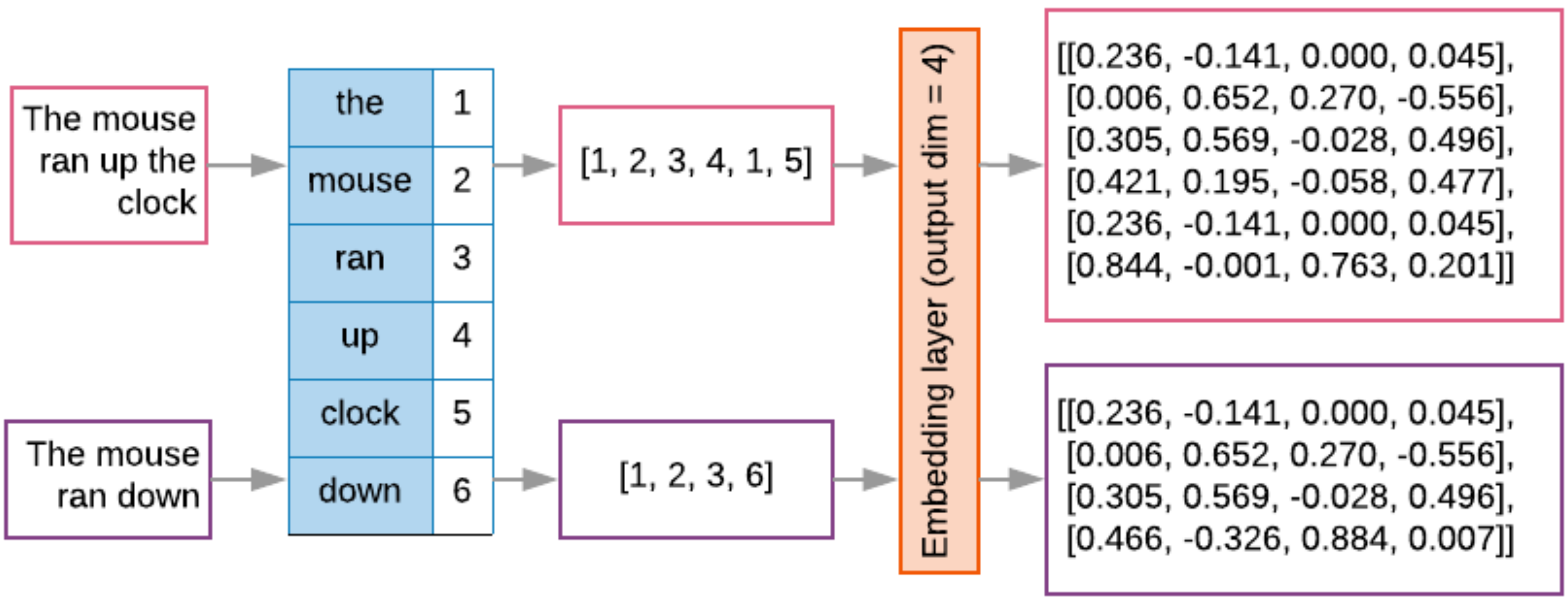
Source: https://developers.google.com/machine-learning/guides/text-classification/step-3

Word embeddings



Source: https://developers.google.com/machine-learning/guides/text-classification/step-3

Word embeddings



Source: https://developers.google.com/machine-learning/guides/text-classification/step-3

from keras.preprocessing.text import Tokenizer

```
from keras.preprocessing.text import Tokenizer
# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
# create the tokenizer
t = Tokenizer()
# fit the tokenizer on the documents
t.fit_on_texts(docs)
print('docs:', docs)
print('word_counts:', t.word_counts)
print('document_count:', t.document_count)
print('word_index:', t.word_index)
print('word_docs:', t.word_docs)
# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='count')
print('texts_to_matrix:')
print(texts_to_matrix)
```

```
docs: ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
word_counts: OrderedDict([('well', 1), ('done', 1), ('good', 1), ('work', 2), ('great', 1), ('effort', 1), ('ni
document_count: 5
word_index: {'work': 1, 'well': 2, 'done': 3, 'good': 4, 'great': 5, 'effort': 6, 'nice': 7, 'excellent': 8}
word_docs: {'done': 1, 'well': 1, 'work': 2, 'good': 1, 'great': 1, 'effort': 1, 'nice': 1, 'excellent': 1}
texts_to_matrix:
[[0. 0. 1. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

Source: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

from keras.preprocessing.text import Tokenizer

```
from keras.preprocessing.text import Tokenizer
# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice
work', 'Excellent!']
# create the tokenizer
t = Tokenizer()
# fit the tokenizer on the documents
t.fit_on_texts(docs)
print('docs:', docs)
print('word_counts:', t.word_counts)
print('document_count:', t.document_count)
print('word_index:', t.word_index)
print('word_docs:', t.word_docs)
# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='count')
print('texts_to_matrix:')
print(texts_to_matrix)
```

Source: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

texts_to_matrix = t.texts_to_matrix(docs, mode='count')

```
docs: ['Well done!', 'Good work', 'Great effort',
'nice work', 'Excellent!']
word_counts: OrderedDict([('well', 1), ('done', 1),
('good', 1), ('work', 2), ('great', 1), ('effort', 1),
('nice', 1), ('excellent', 1)])
document_count: 5
word_index: {'work': 1, 'well': 2, 'done': 3, 'good':
4, 'great': 5, 'effort': 6, 'nice': 7, 'excellent': 8}
word_docs: {'done': 1, 'well': 1, 'work': 2, 'good': 1,
'great': 1, 'effort': 1, 'nice': 1, 'excellent': 1}
texts_to_matrix:
[[0. 0. 1. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

Source: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

t.texts_to_matrix(docs, mode='tfidf')

```
from keras.preprocessing.text import Tokenizer
# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice work',
'Excellent!']
# create the tokenizer
t = Tokenizer()
# fit the tokenizer on the documents
t.fit_on_texts(docs)
print('docs:', docs)
print('word_counts:', t.word_counts)
print('document_count:', t.document_count)
print('word_index:', t.word_index)
print('word_docs:', t.word_docs)
# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='tfidf')
print('texts_to_matrix:')
print(texts_to_matrix)
```

```
texts_to_matrix:
[[0. 0. 1.25276297 1.25276297 0. 0. 0. 0. 0. ]
[0. 0.98082925 0. 0. 1.25276297 0. 0. 0. 0. ]
[0. 0. 0. 0. 0. 1.25276297 1.25276297 0. 0. ]
[0. 0.98082925 0. 0. 0. 0. 0. 1.25276297 0. ]
[0. 0. 0. 0. 0. 0. 0. 0. 1.25276297]]
```

Source: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

Big Data Analytics with Numpy in Python

Numpy

NumPy

Base

N-dimensional array

package

NumPy

is the

fundamental package

for

scientific computing

with Python.

Source: http://www.numpy.org/

NumPy

NumPy

- NumPy provides a **multidimensional array** object to store homogenous or heterogeneous data;
  it also provides optimized functions/methods to operate on this array object.

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

NumPy

NumPy

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

http://www.numpy.org/

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

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

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

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

```
2 * v
```

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

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

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

```
2 * v
```

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

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])
```

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

NumPy

NumPy

```python
import numpy as np

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

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

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

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

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

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

Source: http://cs231n.github.io/python-numpy-tutorial/

Numpy Quickstart Tutorial

SciPy.org Sponsored By ENTHOUGHT

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

https://docs.scipy.org/doc/numpy-dev/user/quickstart.html

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

```python
a.shape
a.ndim
a.dtype.name
```

```python
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]])
```

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

```
(3, 5)
```

```python
a.ndim
```

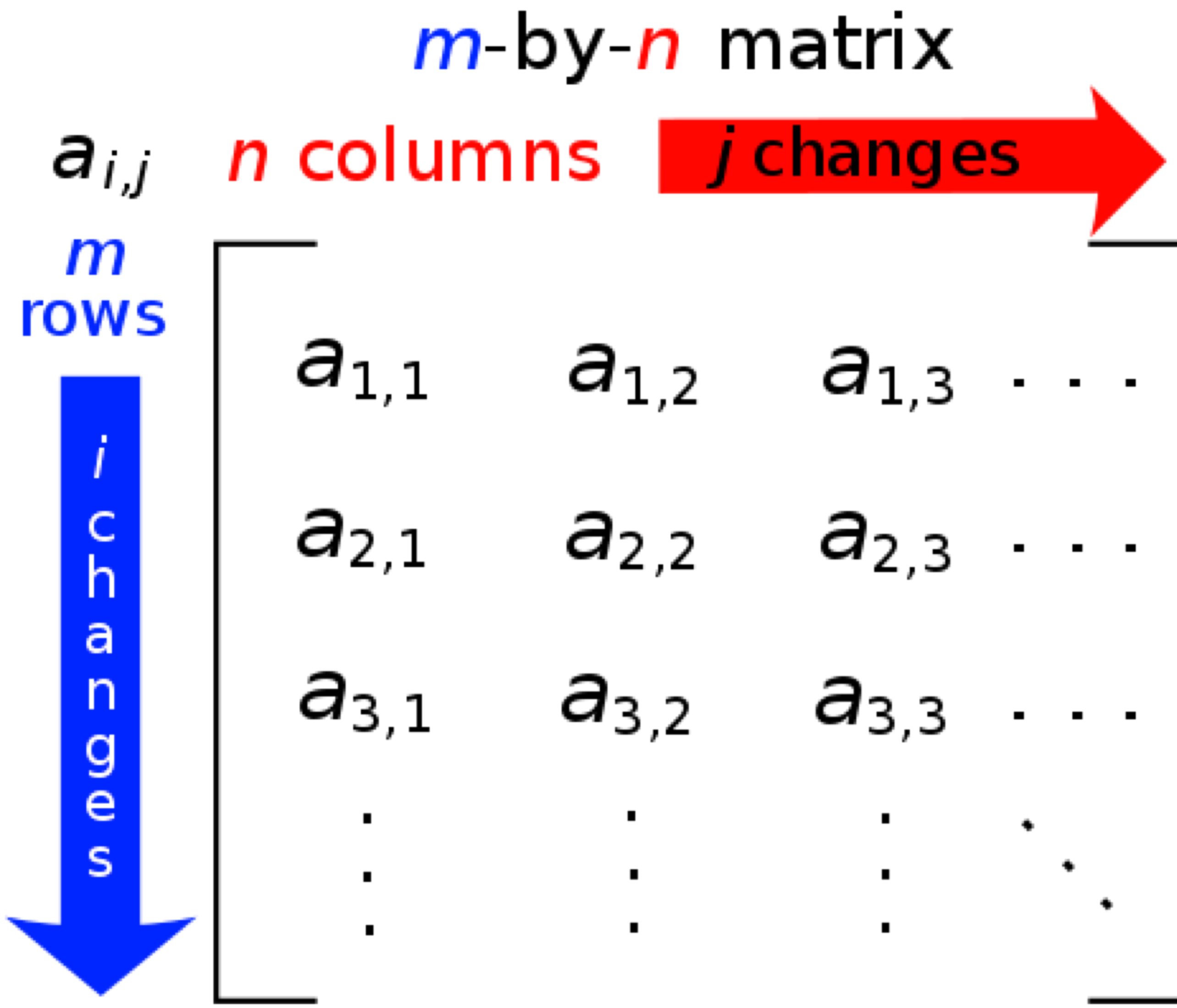
```
2
```

```python
a.dtype.name
```

```
'int64'
```

Source: https://docs.scipy.org/doc/numpy-dev/user/quickstart.html

Matrix


m-by-n matrix
$a_{i,j}$
n columns
j changes
m rows
i changes
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ ...
$a_{2,1}$ $a_{2,2}$ $a_{2,3}$ ...
$a_{3,1}$ $a_{3,2}$ $a_{3,3}$ ...

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**

```python
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**

```python
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)
```

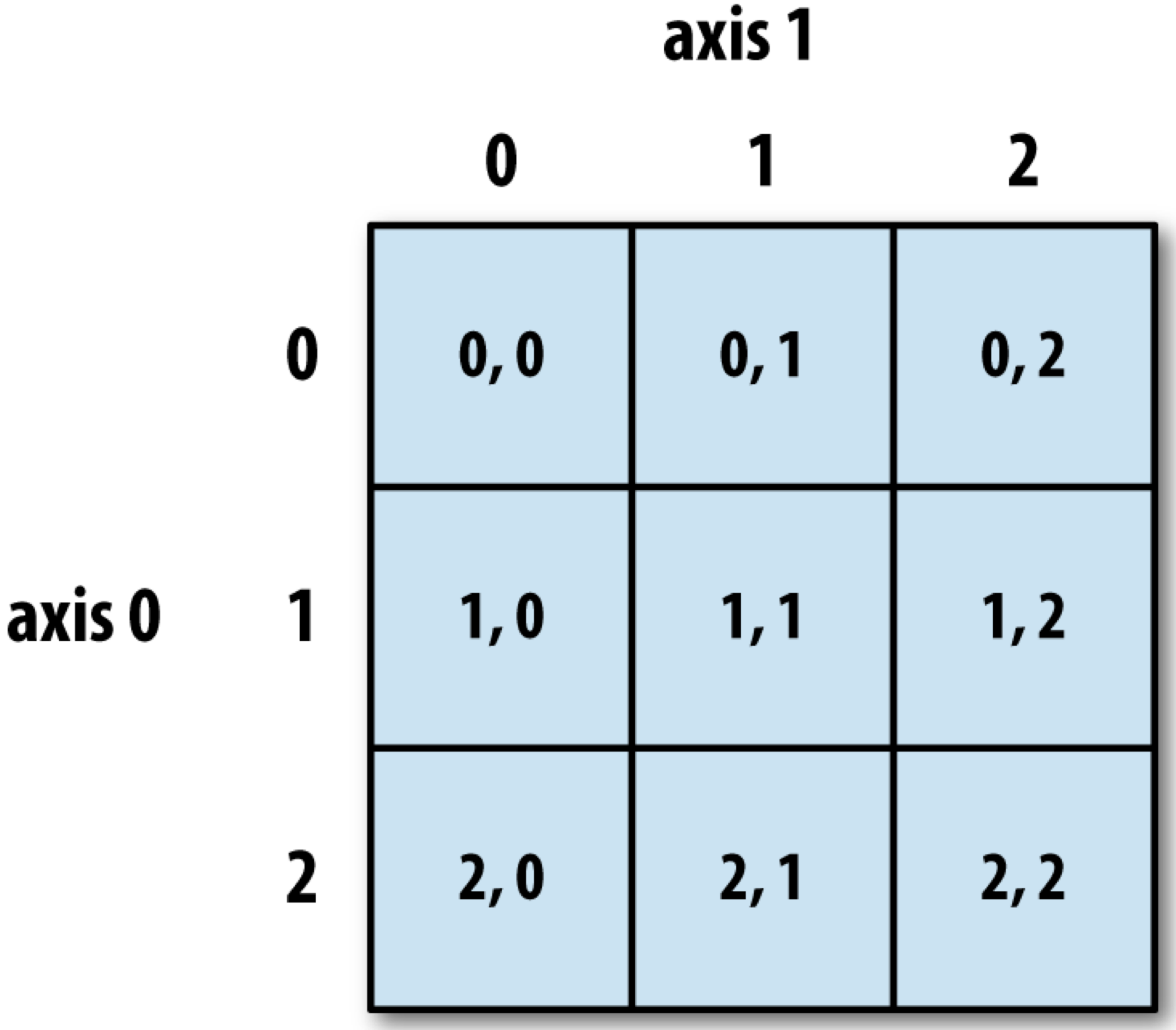
```
(3, 4)
```

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

NumPy Basics: Arrays and Vectorized Computation

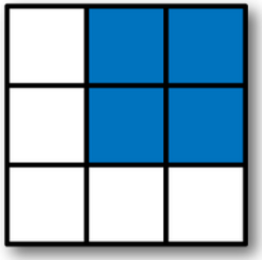
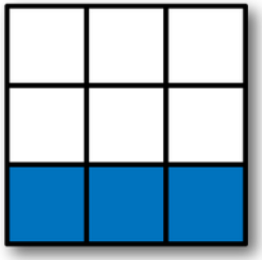
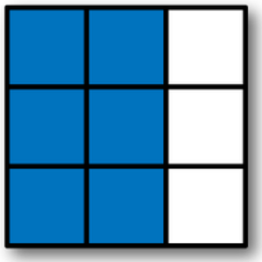
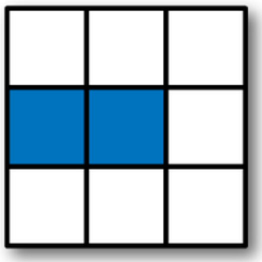
Source: https://www.safaribooksonline.com/library/view/python-for-data/9781449323592/ch04.html

NumPy Array



Source: https://www.safaribooksonline.com/library/view/python-for-data/9781449323592/ch04.html

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)

Source: https://www.safaribooksonline.com/library/view/python-for-data/9781449323592/ch04.html

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
```

Source: https://github.com/wesm/pydata-book/blob/2nd-edition/ch04.ipynb

Python Pandas

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

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

http://pandas.pydata.org/

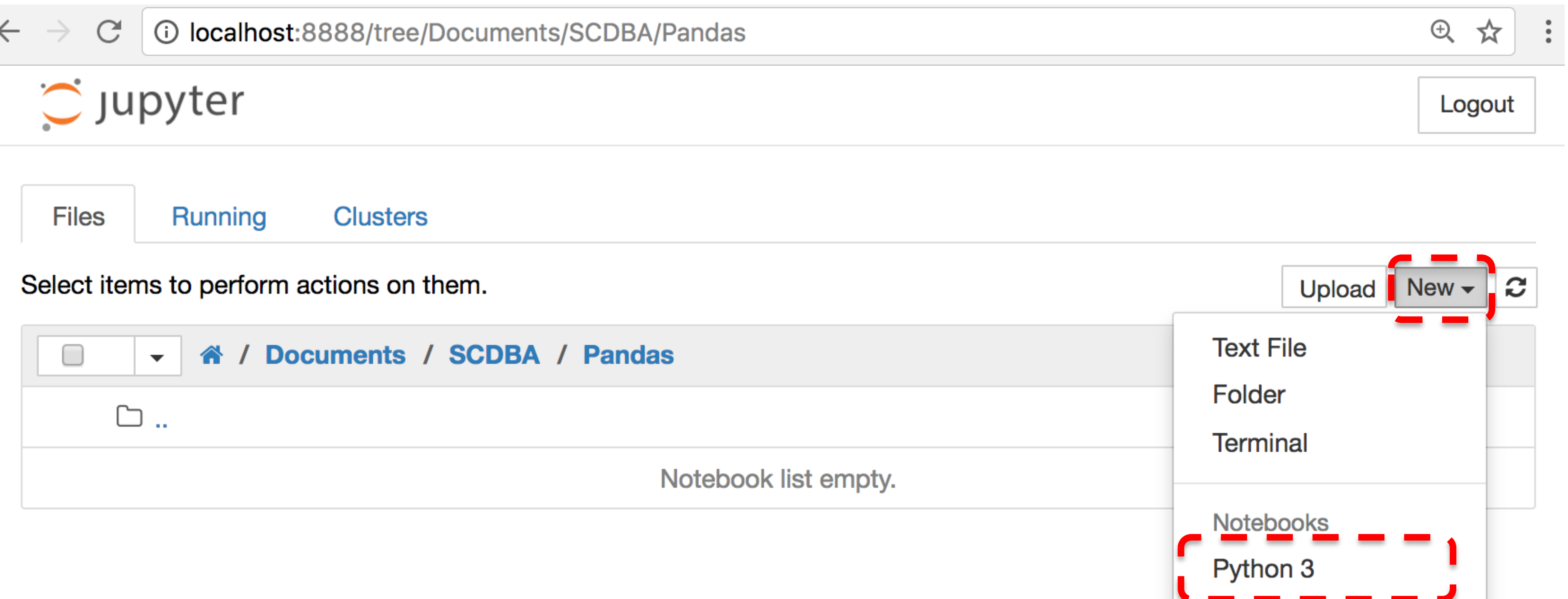
pandas

Python Data Analysis Library

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

Source: http://pandas.pydata.org/

Jupyter Notebook New Python 3



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

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

Source: https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf

Pandas DataFrame

```
type(df)
```

```
type(df)
```

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

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

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

```
dates = pd.date_range('20181001',
periods=6)
dates
```

Source: http://pandas.pydata.org/pandas-docs/stable/10min.html

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
print('pandas imported')
```

```
pandas imported
```

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

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

```python
dates = pd.date_range('20181001', periods=6)
dates
```

```
DatetimeIndex(['2018-10-01', '2018-10-02', '2018-10-03', '2018-10-04',
               '2018-10-05', '2018-10-06'],
              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
2018-10-01	-0.336188	0.584621	-1.061433	-0.036278
2018-10-02	0.903683	-0.839723	-0.270219	-1.099606
2018-10-03	0.920208	-0.240353	-0.818598	-1.105489
2018-10-04	0.221045	-0.314589	0.042071	-1.447280
2018-10-05	0.946862	-1.570305	-1.009180	-0.375659
2018-10-06	-0.225148	0.510691	2.002372	-0.335005

```
df = pd.DataFrame(np.random.randn(3,5),
index=['student1','student2','student3']
, columns=list('ABCDE'))
df
```

```
df = pd.DataFrame(np.random.randn(3,5), index=['student1','student2','student3'], columns=list('ABCDE'))
df
```

	A	B	C	D	E
student1	-0.346884	-1.232934	-0.302072	-1.345084	-0.723880
student2	1.090955	-0.010483	1.280072	-0.253958	-0.030604
student3	0.325660	0.808956	-0.395820	-1.498926	1.603471

```
df2 = pd.DataFrame({ 'A' : 1.,
'B' : pd.Timestamp('20181001'),
'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('20181001'),
'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	2018-10-01	2.5	3	test	foo
1	1.0	2018-10-01	2.5	3	train	foo
2	1.0	2018-10-01	2.5	3	test	foo
3	1.0	2018-10-01	2.5	3	train	foo

df2.dtypes

```
df2.dtypes
```

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

Python Pandas for Finance

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

```
! pip install pandas_datareader
```

```
! pip install pandas_datareader
```

```
Collecting pandas_datareader
  Downloading https://files.pythonhosted.org/packages/cc/5c/ea5b6dcfd0f55c5fb1e37fb45335ec01cceca199b8a79339137f5ed269e0/pandas_datarea
    100% |████████████████████████████████| 112kB 2.7MB/s
Collecting lxml (from pandas_datareader)
  Downloading https://files.pythonhosted.org/packages/03/a4/9eea8035fc7c7670e5eab97f34ff2ef0ddd78a491bf96df5accedb0e63f5/lxml-4.2.5-cp3
    100% |████████████████████████████████| 5.8MB 7.5MB/s
Requirement already satisfied: pandas>=0.19.2 in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (0.22.0)
Requirement already satisfied: requests>=2.3.0 in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (2.18.4)
Requirement already satisfied: wrapt in /usr/local/lib/python3.6/dist-packages (from pandas_datareader) (1.10.11)
Requirement already satisfied: python-dateutil>=2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19.2->pandas_datareader) (2
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19.2->pandas_datareader) (1.14.6)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-packages (from pandas>=0.19.2->pandas_datareader) (2018.5)
Requirement already satisfied: idna<2.7,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (2.6)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader) (
Requirement already satisfied: urllib3<1.23,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests>=2.3.0->pandas_datareader
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2->pandas>=0.19.2->pandas_data
Installing collected packages: lxml, pandas-datareader
Successfully installed lxml-4.2.5 pandas-datareader-0.7.0
```

conda install pandas-datareader

```
imyday — -bash — 80×24
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
Adj Close
MA05
MA20
MA60
220
200
180
160
140
120
100
80
2016-02
2016-06
2016-10
2017-02
2017-06
2017-10
2018-02
2018-06
2018-10
Date

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()
```

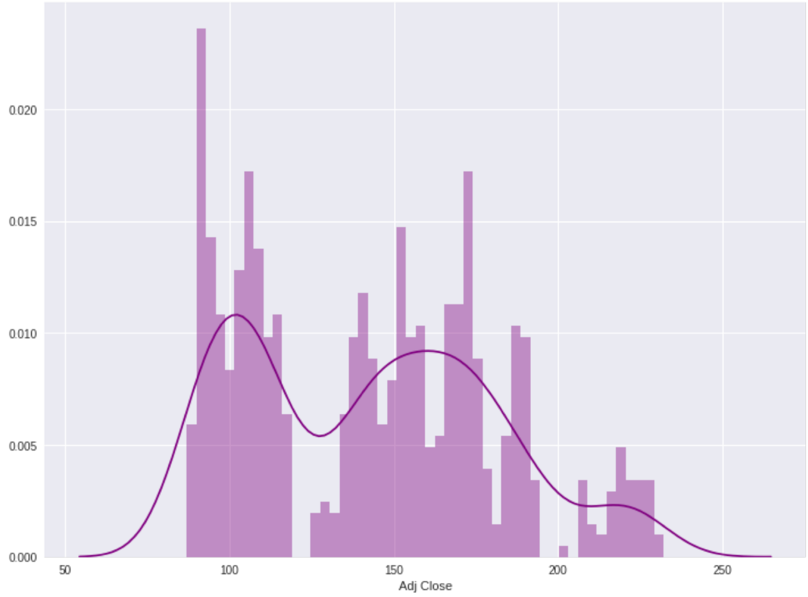
```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL’, label='Adj Close')
```

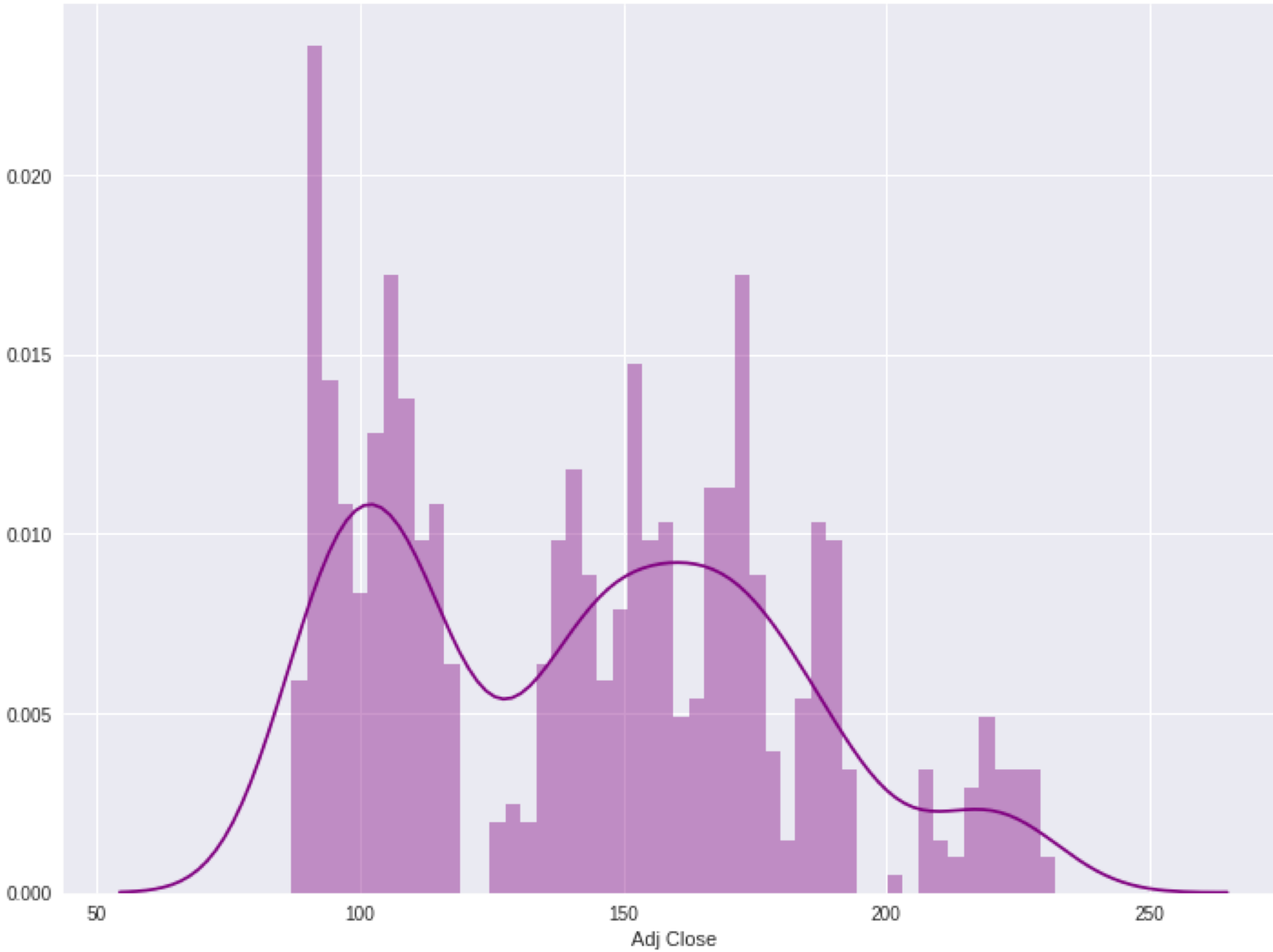
```
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0),
rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0),
rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'],
color='blue') #df.index gives the dates
bottom.bar(df.index, df['Volume'])
```


AAPL
Adj Close
220
200
180
160
140
120
100
80
1e8
Volume
1.0
0.5
0.0
2016-02
2016-06
2016-10
2017-02
2017-06
2017-10
2018-02
2018-06
2018-10

```
 # 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')
```




0.020
0.015
0.010
0.005
0.000
50
100
150
200
250
Adj Close

```
# 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
Adj Close
MA05
MA20
MA60
220
200
180
160
140
120
100
80
2016-02
2016-06
2016-10
2017-02
2017-06
2017-10
2018-02
2018-06
2018-10
Date

```
# !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()
```

```python
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%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()
```

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()
```

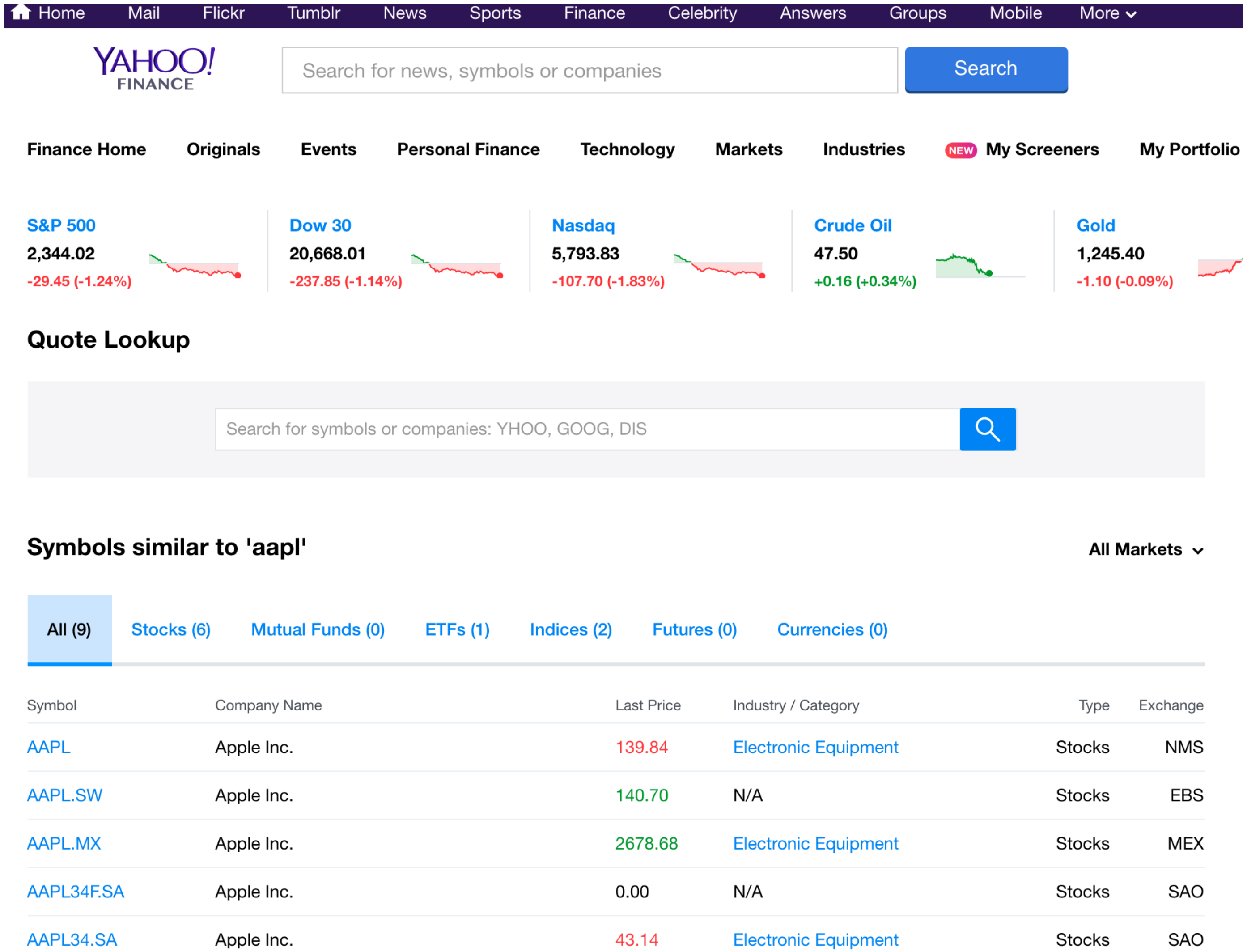
```
# ! pip install quandl
import quandl
# quandl.ApiConfig.api_key = "YOURAPIKEY"
df = quandl.get("WIKI/AAPL", start_date="2016-01-01", end_date="2017-12-31")
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

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

Date	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
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

Source: https://www.quandl.com/tools/python

Yahoo Finance Symbols: AAPL Apple Inc. (AAPL)



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

Apple Inc. (AAPL) -NasdaqGS

YAHOO! FINANCE

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%)
At close: 4:00PM EDT

139.35 -0.49 (-0.35%)
After hours: 7:59PM EDT

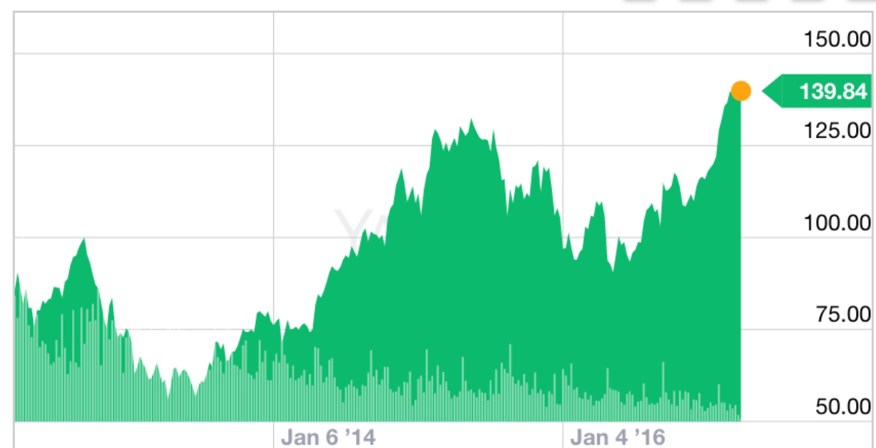
Summary | Conversations | Statistics | Profile | Financials | Options | Holders | Historical Data | Analysts

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

Trade prices are not sourced from all markets

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

Interactive chart

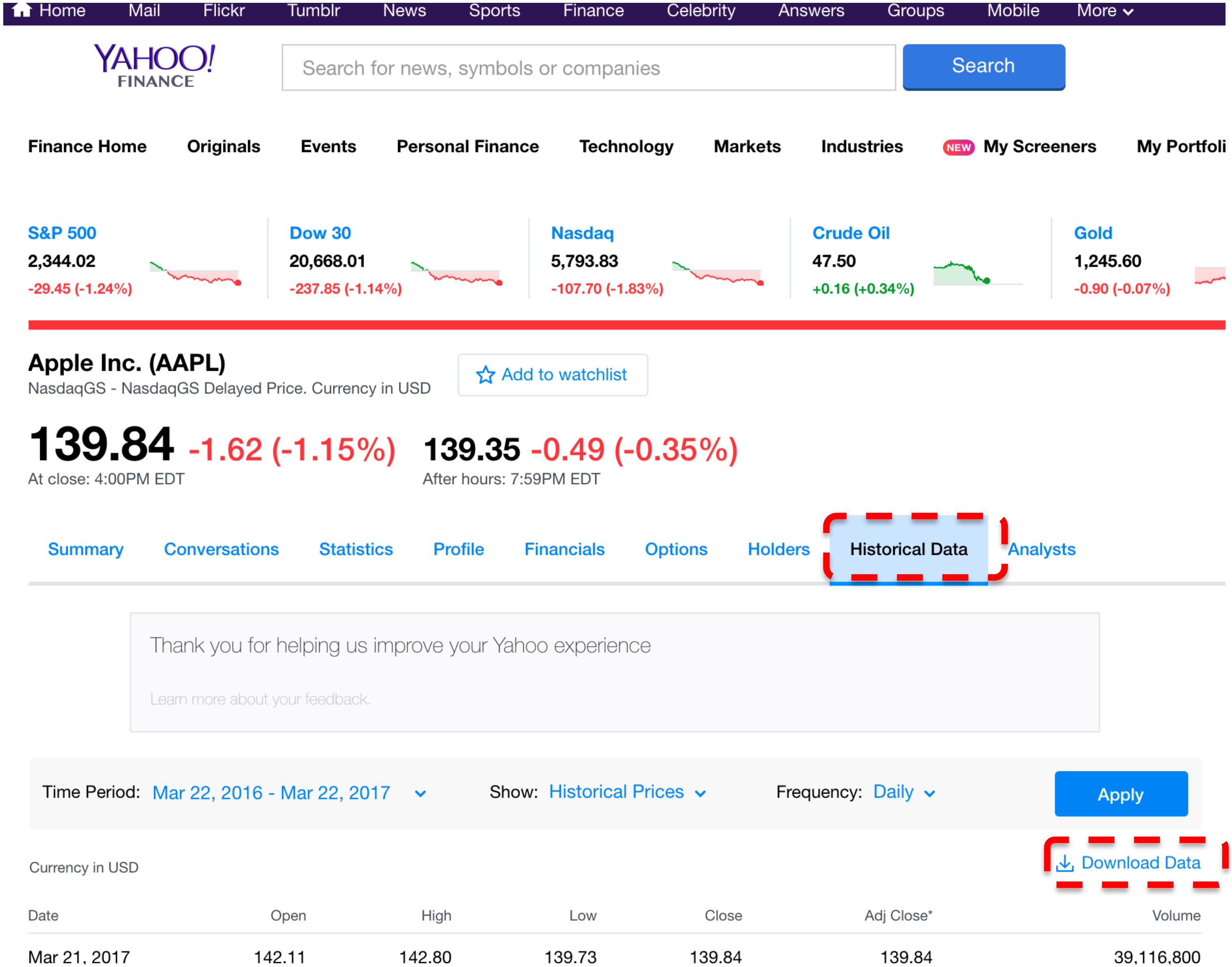


http://finance.yahoo.com/quote/AAPL?p=AAPL

Yahoo Finance Charts: Apple Inc. (AAPL)

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

Apple Inc. (AAPL) Historical Data



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

Yahoo Finance Historical Prices
Apple Inc. (AAPL)

YAHOO! FINANCE

Search for news, symbols or companies | Search

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
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)

YAHOO! FINANCE

Search for news, symbols or companies | Search

Time Period: Dec 12, 1980 - Mar 22, 2017 | Show: Historical Prices | Frequency: Daily | Apply

1D 5D 3M 6M
YTD 1Y 5Y MAX

Start Date: 12/12/1980
End Date: 3/22/2017

Done | Cancel

Currency i

Download Data

Date		High	Low	Close	Adj Close*	Volume
Mar 21, 2		42.80	139.73	139.84	139.84	39,116,800
Mar 20, 2		41.50	140.23	141.46	141.46	21,542,000
Mar 17, 2		41.00	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

http://finance.yahoo.com/quote/AAPL/history?period1=345398400&period2=1490112000&interval=1d&filter=history&frequency=1d

Yahoo Finance Historical Prices Apple Inc. (AAPL)

YAHOO! FINANCE

Search for news, symbols or companies | Search

Time Period: Dec 12, 1980 - 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
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

http://finance.yahoo.com/quote/AAPL/history?period1=345398400&period2=1490112000&interval=1d&filter=history&frequency=1d

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)

http://finance.yahoo.com/echarts?s=GOOG+Interactive#{"showArea":false,"showLine":false,"showCandle":true,"lineType":"candle","range":"5y","allowChartStacking":true}

Dow Jones Industrial Average (^DJI)

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

TSEC weighted index (^TWII) - Taiwan

http://finance.yahoo.com/chart/^TWII

Taiwan Semiconductor Manufacturing Company Limited (2330.TW)

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	Dow 30	Nasdaq	Crude Oil	Gold
2,344.02	20,668.01	5,793.83	47.50	1,244.90
-29.45 (-1.24%)	-237.85 (-1.14%)	-107.70 (-1.83%)	+0.16 (+0.34%)	-1.60 (-0.13%)

Taiwan Semiconductor Manufacturing Company Limited (2330.TW)

Taiwan - Taiwan Delayed Price. Currency in TWD

Add to watchlist

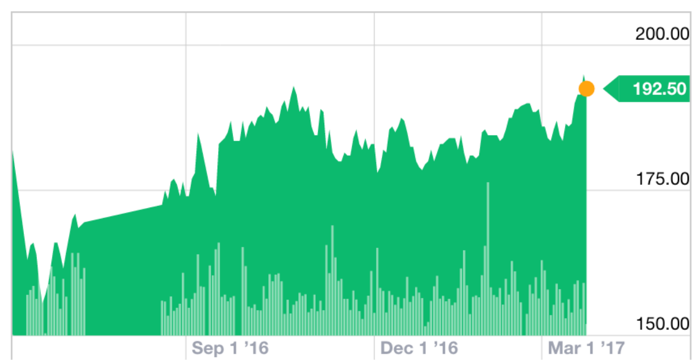
192.50 -2.50 (-1.28%)

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)

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

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

```
import fix_yahoo_finance as yf
data = yf.download("^TWII", start="2017-07-01", end="2017-11-15")
data.to_csv('TWII_201707_201711.csv')
data.tail()
```

```
import fix_yahoo_finance as yf
data = yf.download("^TWII", start="2017-07-01", end="2017-11-15")
data.to_csv('TWII_201707_201711.csv')
data.tail()
```

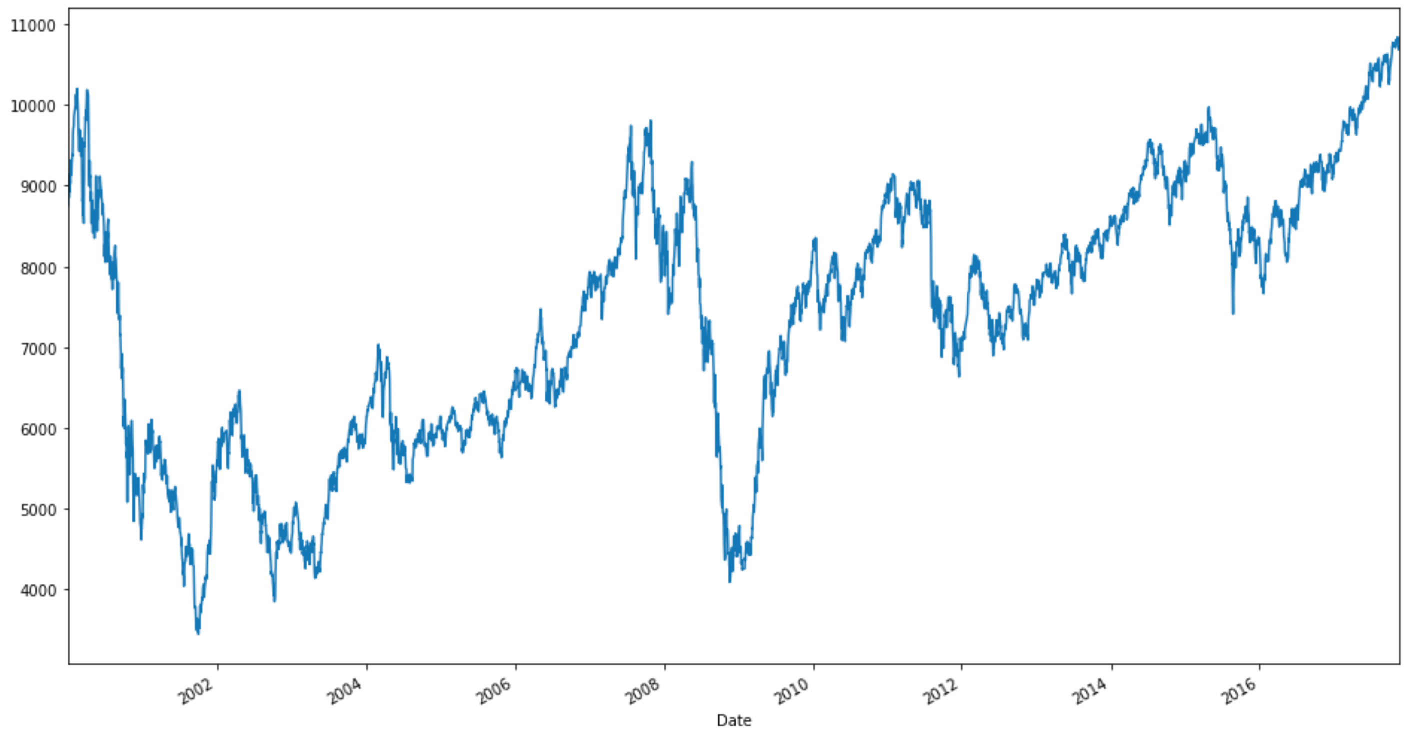
```
[*********************100%***********************]  1 of 1 downloaded
```

	Open	High	Low	Close	Adj Close	Volume
Date						
2017-11-08	10839.440430	10844.740234	10806.009766	10818.990234	10818.990234	2438000
2017-11-09	10802.950195	10831.379883	10721.870117	10743.269531	10743.269531	2917800
2017-11-10	10713.669922	10742.610352	10659.290039	10732.669922	10732.669922	2277000
2017-11-13	10728.219727	10749.389648	10683.919922	10683.919922	10683.919922	2765100
2017-11-14	10716.589844	10735.080078	10654.580078	10687.179688	10687.179688	2596600

df.loc[start:end]

```
df = df.loc['2017-10-01':'2017-11-15']
```

```
import matplotlib.pyplot as plt
%matplotlib inline
import fix_yahoo_finance as yf
df = yf.download("^TWII", start="2000-01-01", end="2017-11-15")
df.to_csv('YF_TWII_2000_2017.csv')
print(df.head())
fig = plt.figure(figsize=(16,9))
df["Adj Close"].plot()
fig.show()
```



candlestick_ohlc

```
import matplotlib.pyplot as plt

from matplotlib.finance
import candlestick_ohlc
```

Source: https://matplotlib.org/examples/pylab_examples/finance_demo.html

```python
import matplotlib.pyplot as plt
from matplotlib.finance import candlestick_ohlc
```

Source: https://ntguardian.wordpress.com/2016/09/19/introduction-stock-market-data-python-1/

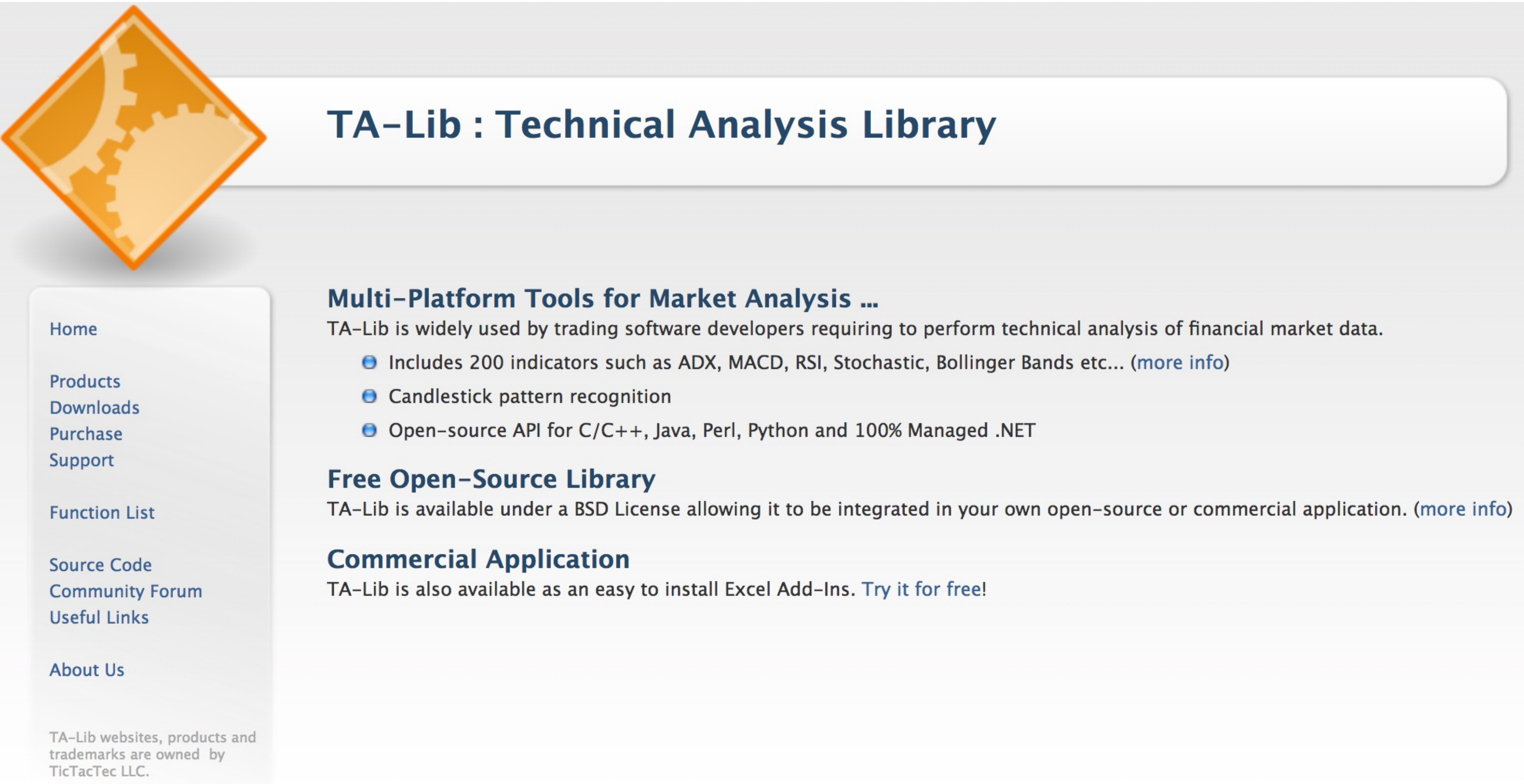
daily_to_weekly

```
#Convert Daily Data to Weekly Data
def daily_to_weekly(df):
     #dfWeekly = daily_to_weekly(df)
    #df.sort_index(axis=0, level=None, ascending=True, inplace=True)
    Open = df.Open.resample('W-Fri').first() #W #W-MON #W-Fri
    High = df.High.resample('W-Fri').max()
    Low = df.Low.resample('W-Fri').min()
    Close = df.Close.resample('W-Fri').last()
    Volume = df.Volume.resample('W-Fri').sum()
    Adj_Close = df["Adj Close"].resample('W-Fri').last()
    dfWeekly = pd.concat([Open, High, Low, Close, Volume, Adj_Close], axis=1)
    dfWeekly = dfWeekly[pd.notnull(dfWeekly['Adj Close'])]
    return dfWeekly
```

daily_to_monthly

```
#Convert Daily Data to Monthly Data
def daily_to_monthly(df):
    #dfMonthly = daily_to_monthly(df)
    Open = df.Open.resample('M').first()
    High = df.High.resample('M').max()
    Low = df.Low.resample('M').min()
    Close = df.Close.resample('M').last()
    Volume = df.Volume.resample('M').sum()
    Adj_Close = df["Adj Close"].resample('M').last()
    dfMonthly = pd.concat([Open, High, Low, Close, Volume, Adj_Close], axis=1)
    dfMonthly = dfMonthly[pd.notnull(dfMonthly['Adj Close'])]
    return dfMonthly
```

TA-Lib:
Technical Analysis Library



http://ta-lib.org/

Stochastic Oscillator (KD)

```
#Stochastic oscillator %D
def KDJ(df, n, m1, m2):
    #KDJ(df, 9, 3, 3)
    KDJ_n = n
    KDJ_m1 = m1
    KDJ_m2 = m2

    df['Low_n'] = pd.rolling_min(df['Low'], KDJ_n)
    df['Low_n'].fillna(value=pd.expanding_min(df['Low']), inplace=True)
    df['High_n'] = pd.rolling_max(df['High'], KDJ_n)
    df['High_n'].fillna(value=pd.expanding_max(df['High']), inplace=True)

    df['RSV'] = (df['Close'] - df['Low_n']) / (df['High_n'] - df['Low_n']) * 100

    df['KDJ_K'] = pd.ewma(df['RSV'], KDJ_m1)
    df['KDJ_D'] = pd.ewma(df['KDJ_K'], KDJ_m2)
    df['KDJ_J'] = 3 * df['KDJ_K'] - 2 * df['KDJ_D']
    return df
```

Source: https://www.quantopian.com/posts/technical-analysis-indicators-without-talib-code

Bollinger Bands

```
#Bollinger Bands
def BBANDS20(df, n):
    MA = pd.Series(pd.rolling_mean(df['Close'], n))
    MSD = pd.Series(pd.rolling_std(df['Close'], n))
    b1 = 4 * MSD / MA
    B1 = pd.Series(b1, name = 'BollingerB_' + str(n))
    df = df.join(B1)
    b2 = (df['Close'] - MA + 2 * MSD) / (4 * MSD)
    B2 = pd.Series(b2, name = 'Bollinger%b_' + str(n))
    df = df.join(B2)
    return df
```

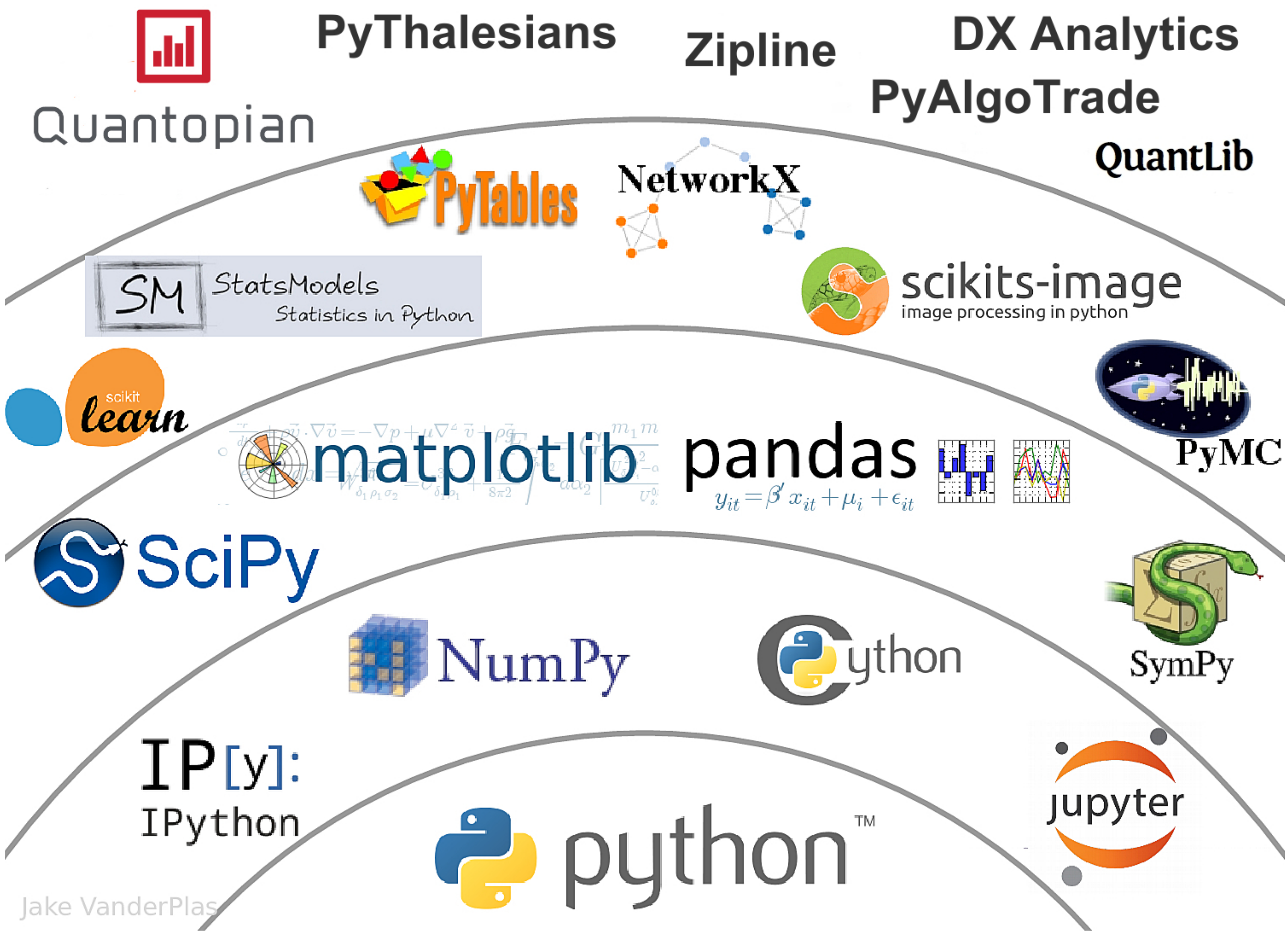
Source: https://www.quantopian.com/posts/technical-analysis-indicators-without-talib-code

Bollinger Bands

```
#BB Bollinger Bands BB_20
def BB_20(df):
    df['BB_MA20'] = pd.stats.moments.rolling_mean(df["Adj Close"], 20)
    df['BB_SD20'] = pd.stats.moments.rolling_std(df['Adj Close'],20)
    df['BB_UpperBand'] = df['BB_MA20'] + (df['BB_SD20']*2) # Default 2*SD
    df['BB_LowerBand'] = df['BB_MA20'] - (df['BB_SD20']*2)
    df['BB_PB'] = (df['Adj Close'] - df['BB_LowerBand']) / (df['BB_UpperBand'] -
df['BB_LowerBand'])
    df['BB_BW'] = (df['BB_UpperBand'] - df['BB_LowerBand']) / df['BB_MA20']
    df['BB_UpperBand_1SD'] = df['BB_MA20'] + (df['BB_SD20'])
    df['BB_LowerBand_1SD'] = df['BB_MA20'] - (df['BB_SD20'])
    #BB_PB: Bollinger Band Percent b (PB)
    #BB_BW: Bollinger Band Band Width (BW)
    return df
```

Source: https://www.quantopian.com/posts/technical-analysis-indicators-without-talib-code

The Quant Finance PyData Stack



Summary

- **Foundations of Big Data Mining in 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
- 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
- Wes McKinney (2012), Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, O'Reilly Media
- Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
- Michael Heydt (2015) , Mastering Pandas for Finance, Packt Publishing
- Michael Heydt (2015), Learning Pandas - Python Data Discovery and Analysis Made Easy, Packt Publishing
- James Ma Weiming (2015), Mastering Python for Finance, Packt Publishing
- Fabio Nelli (2015), Python Data Analytics: Data Analysis and Science using PANDAs, matplotlib and the Python Programming Language, Apress
- 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/
- 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/