Big Data Analysis

Foundations of Big Data Analysis in Python

1112BDA03
MBA, IM, NTPU (M6031) (Spring 2023)
Tue 2, 3, 4 (9:10-12:00) (B8F40)

https://meet.google.com/paj-zhhj-mya

Min-Yuh Day, Ph.D,
Associate Professor

Institute of Information Management, National Taipei University

https://web.ntpu.edu.tw/~myday

2023-03-14

Syllabus

Week	Date	Subject/Topics
1	2023/02/21	Introduction to Big Data Analysis
2	2023/02/28	(Day Off)
3	2023/03/07	AI, Data Science, and Big Data Analysis
4	2023/03/14	Foundations of Big Data Analysis in Python
5	2023/03/21	Case Study on Big Data Analysis I
6	2023/03/28	Machine Learning: SAS Viya, Data Preparation and Algorithm Selection

Syllabus

Week	Date	Subject/Topics
7	2023/04/04	(Children's Day) (Day off)
8	2023/04/11	Midterm Project Report
9	2023/04/18	Machine Learning: Decision Trees and Ensembles of Trees
10	2023/04/25	Machine Learning: Neural Networks (NN) and Support Vector Machines (SVM)
11	2023/05/02	Case Study on Big Data Analysis II
12	2023/05/09	Machine Learning: Model Assessment and Deployment

Syllabus

Week	Date	Subject/Topics
13	2023/05/16	ChatGPT and Large Language Models (LLM) for Big Data Analysis
14	2023/05/23	Deep Learning for Finance Big Data Analysis
15	2023/05/30	Final Project Report I
16	2023/06/06	Final Project Report II
17	2023/06/13	Self-learning
18	2023/06/20	Self-learning

Foundations of Big Data Analysis in Python

Outline

- **Foundations of Big Data Analysis in Python**
  - **Python Ecosystem for Data Science**
  - **Python**
    - Programming language
  - **Numpy**
    - Scientific computing
  - **Pandas**
    - Data structures and data analysis tools

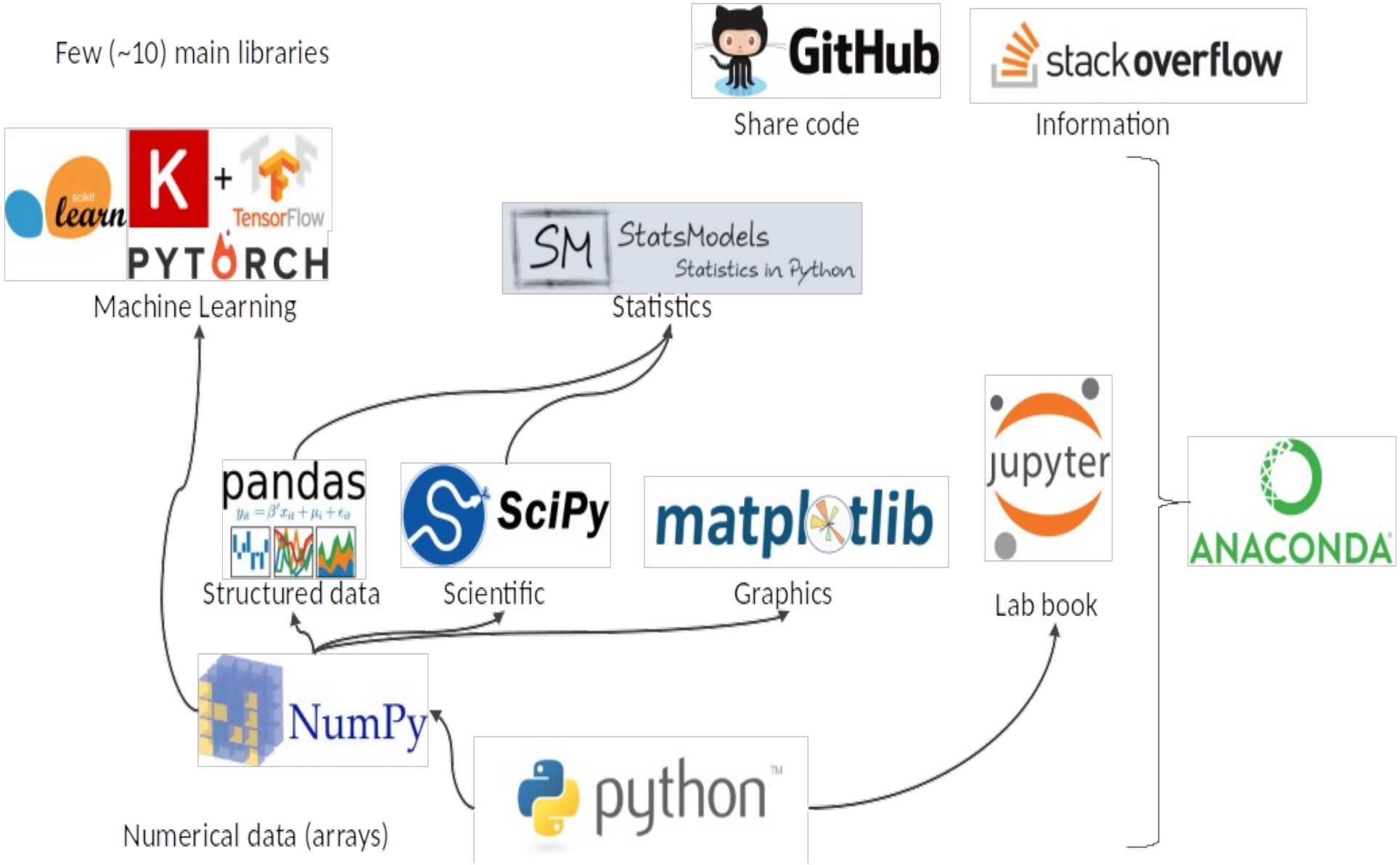
Python

python™

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

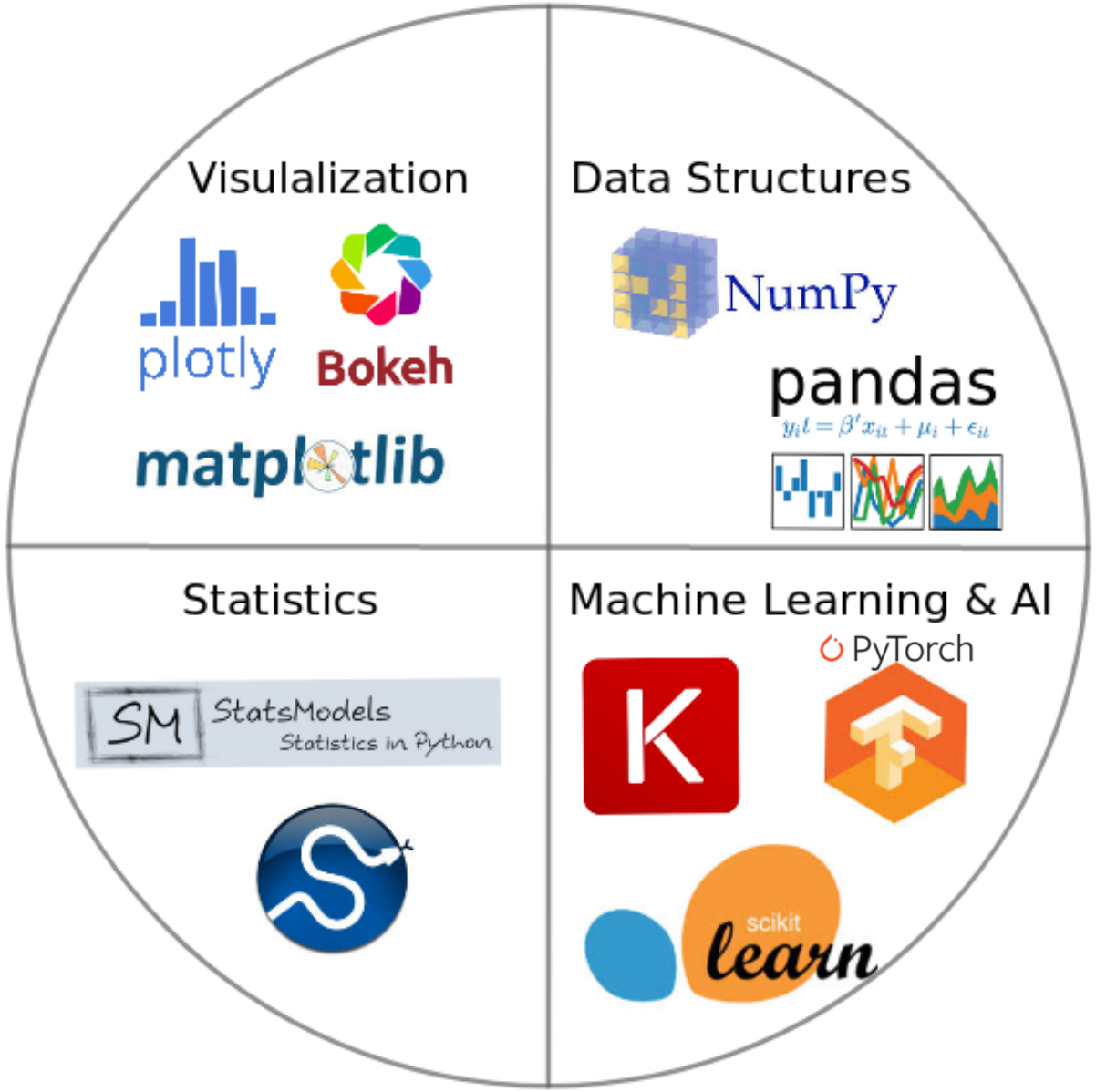
Source: https://www.python.org/doc/essays/blurb/

Python Ecosystem for Data Science



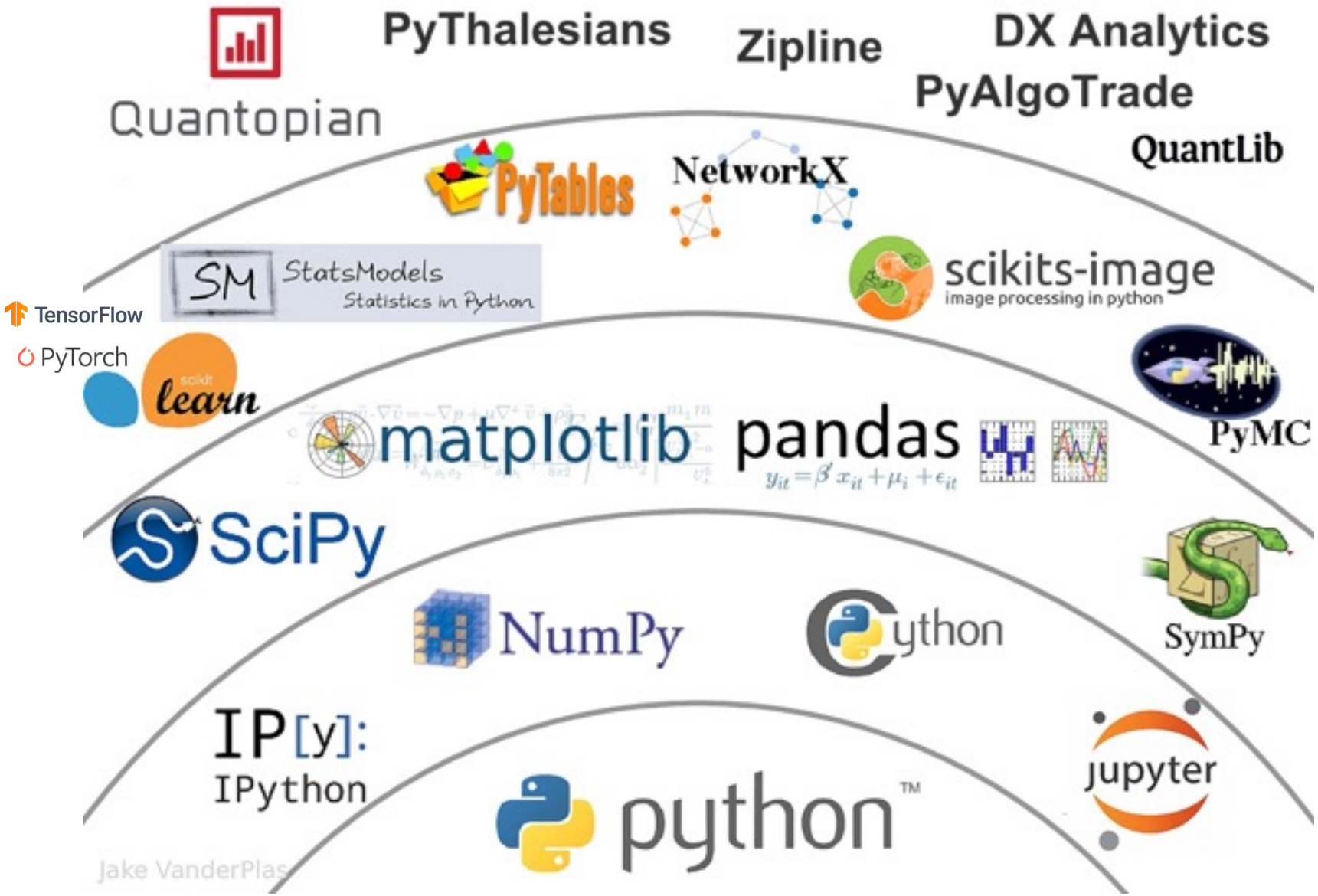
Source: https://medium.com/pyfinance/why-python-is-best-choice-for-financial-data-modeling-in-2019-c0d0d1858c45

Python Ecosystem for Data Science



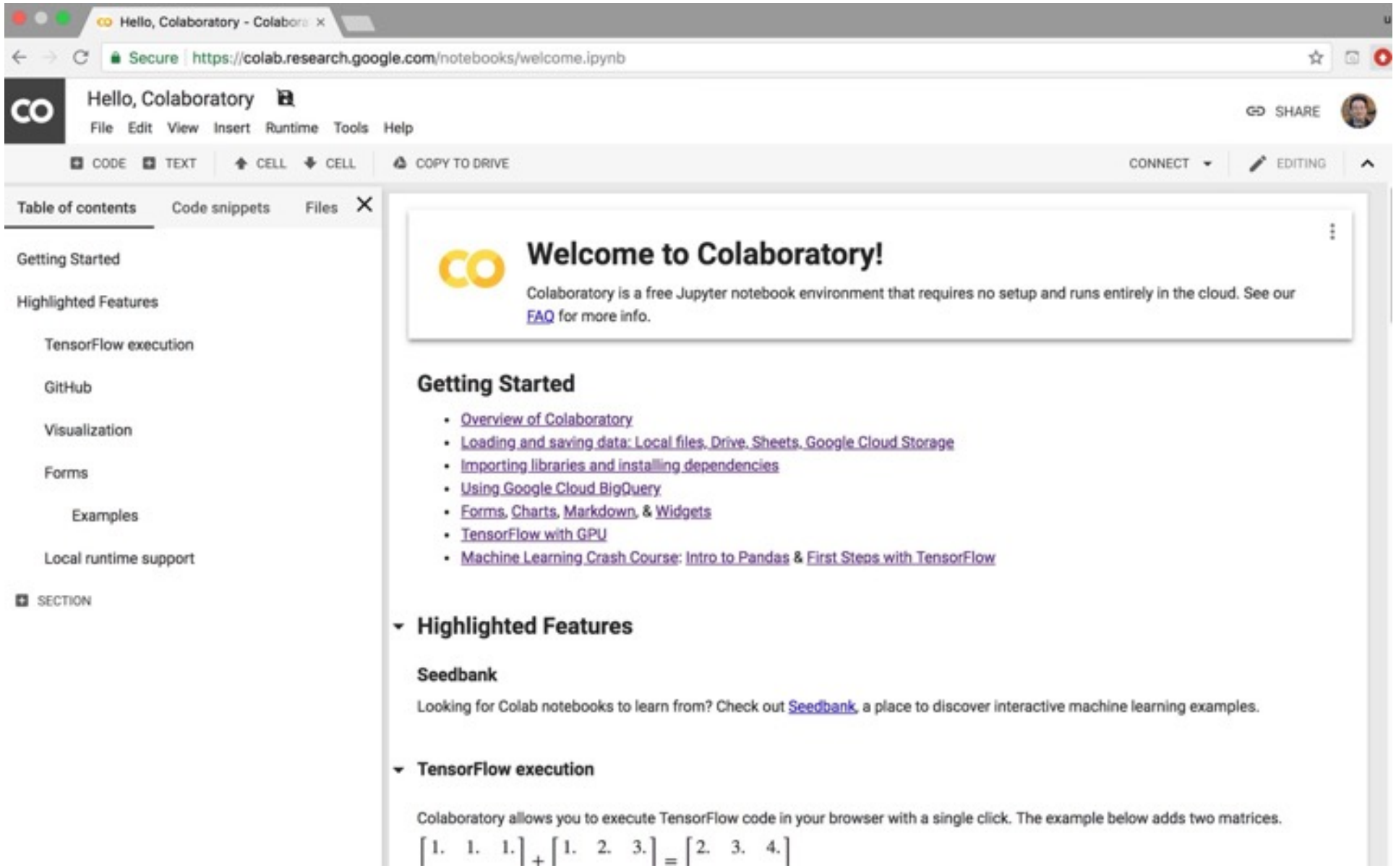
Source:https://duchesnay.github.io/pystatsml/introduction/python_ecosystem.html

The Quant Finance PyData Stack



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

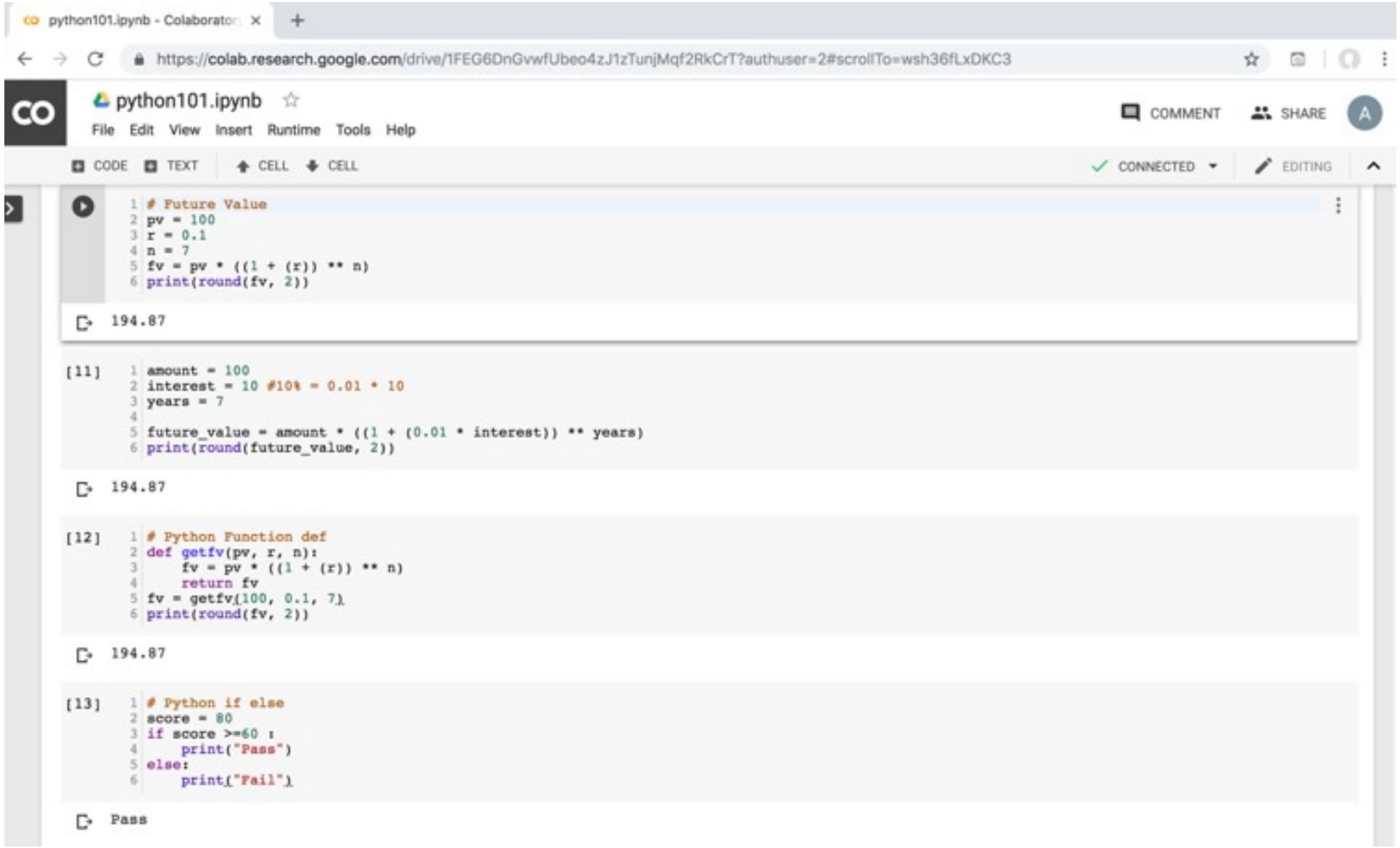
Google Colab



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

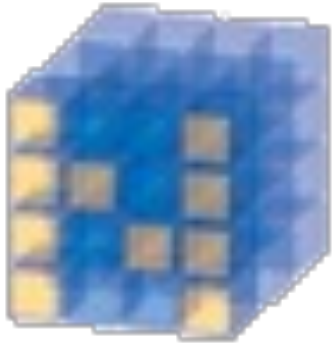
Python in Google Colab (Python101)

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



https://tinyurl.com/aintpupython101

Numpy



NumPy

Base

N-dimensional array

package

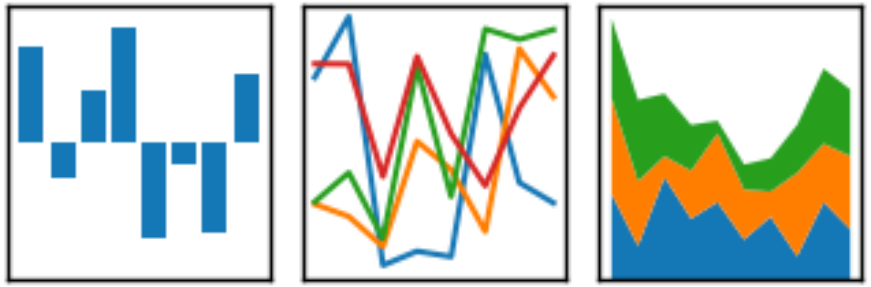
Python matplotlib

Source: https://matplotlib.org/

Python Pandas

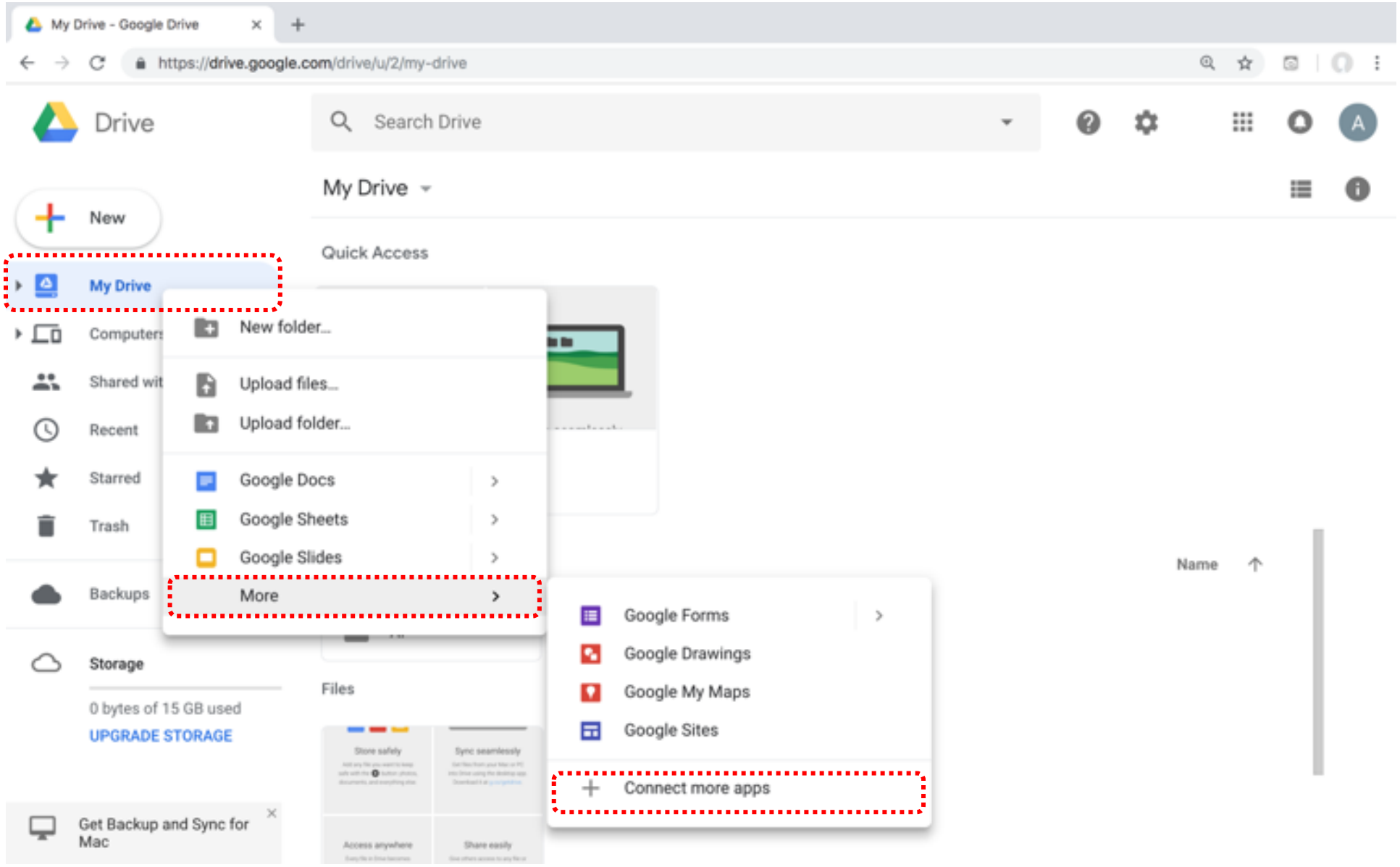
pandas

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

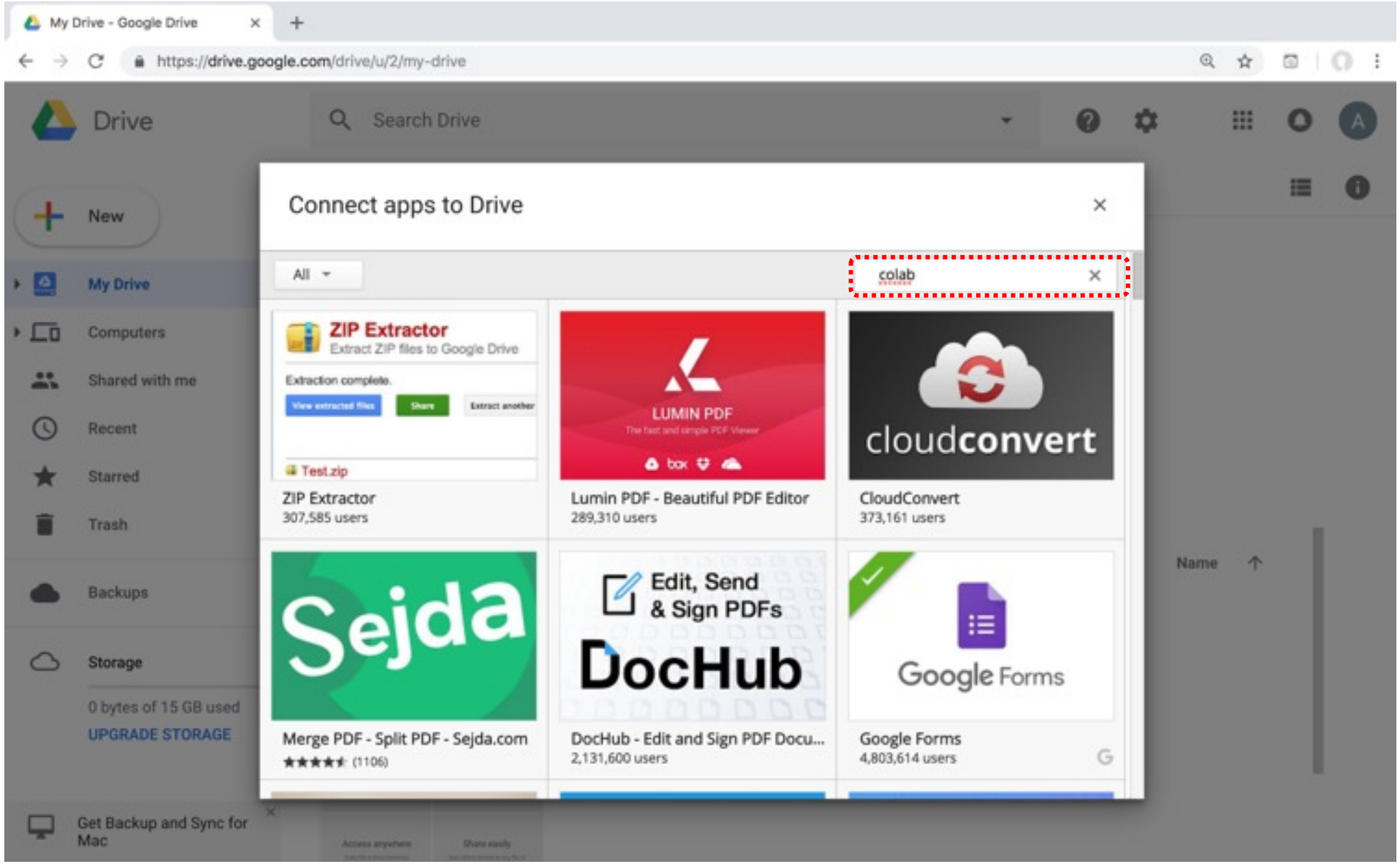


http://pandas.pydata.org/

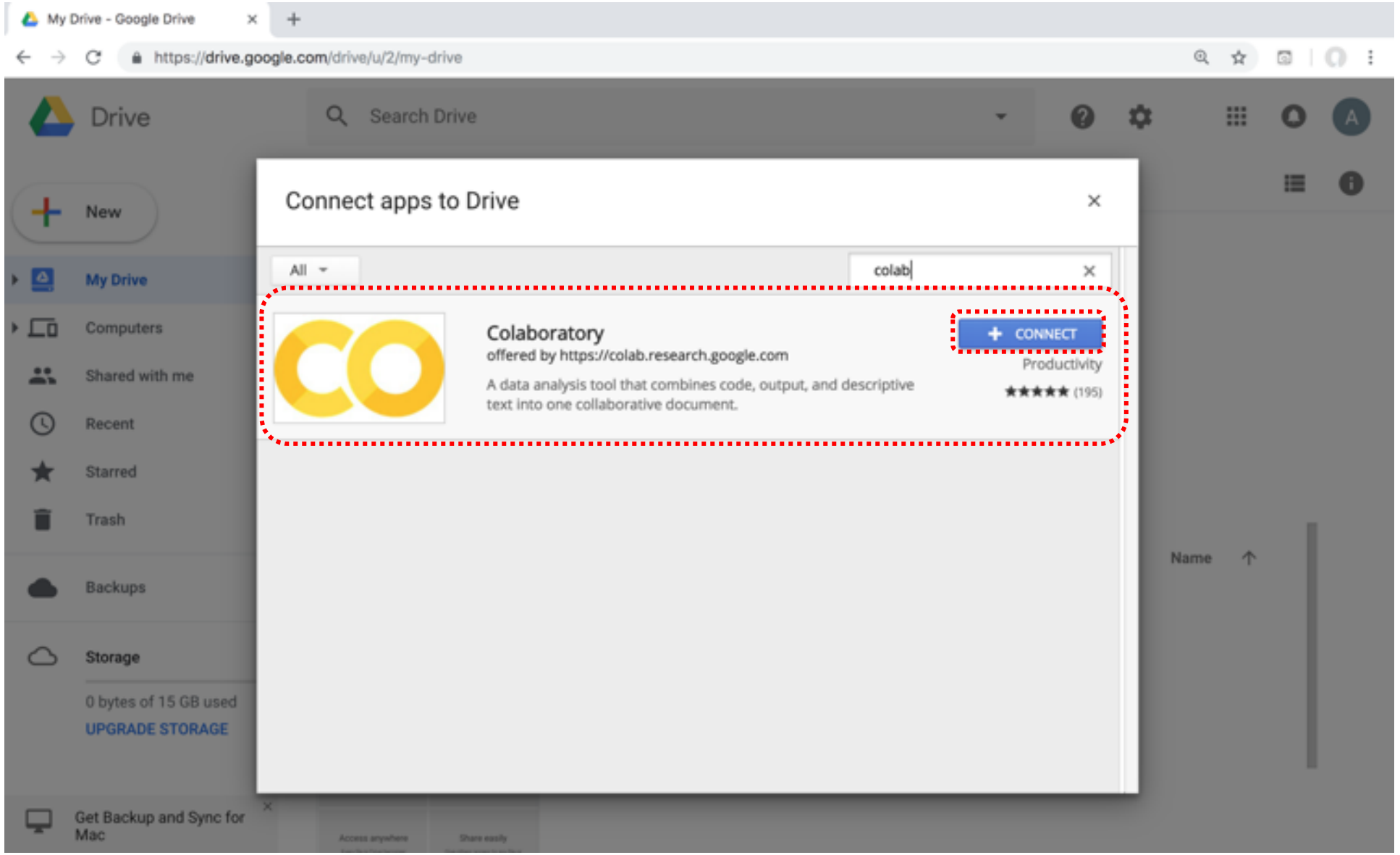
Connect Google Colab in Google Drive



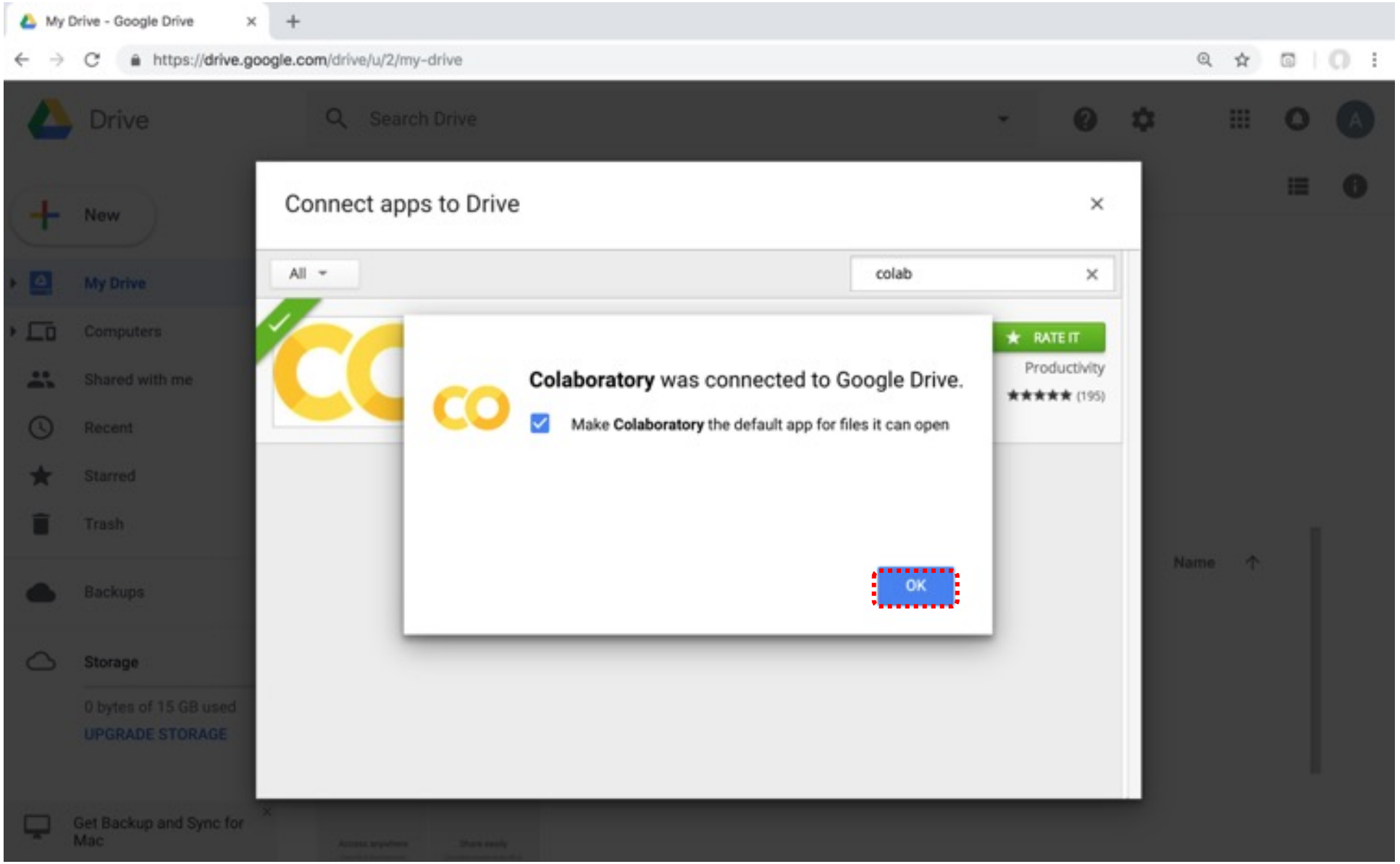
Google Colab



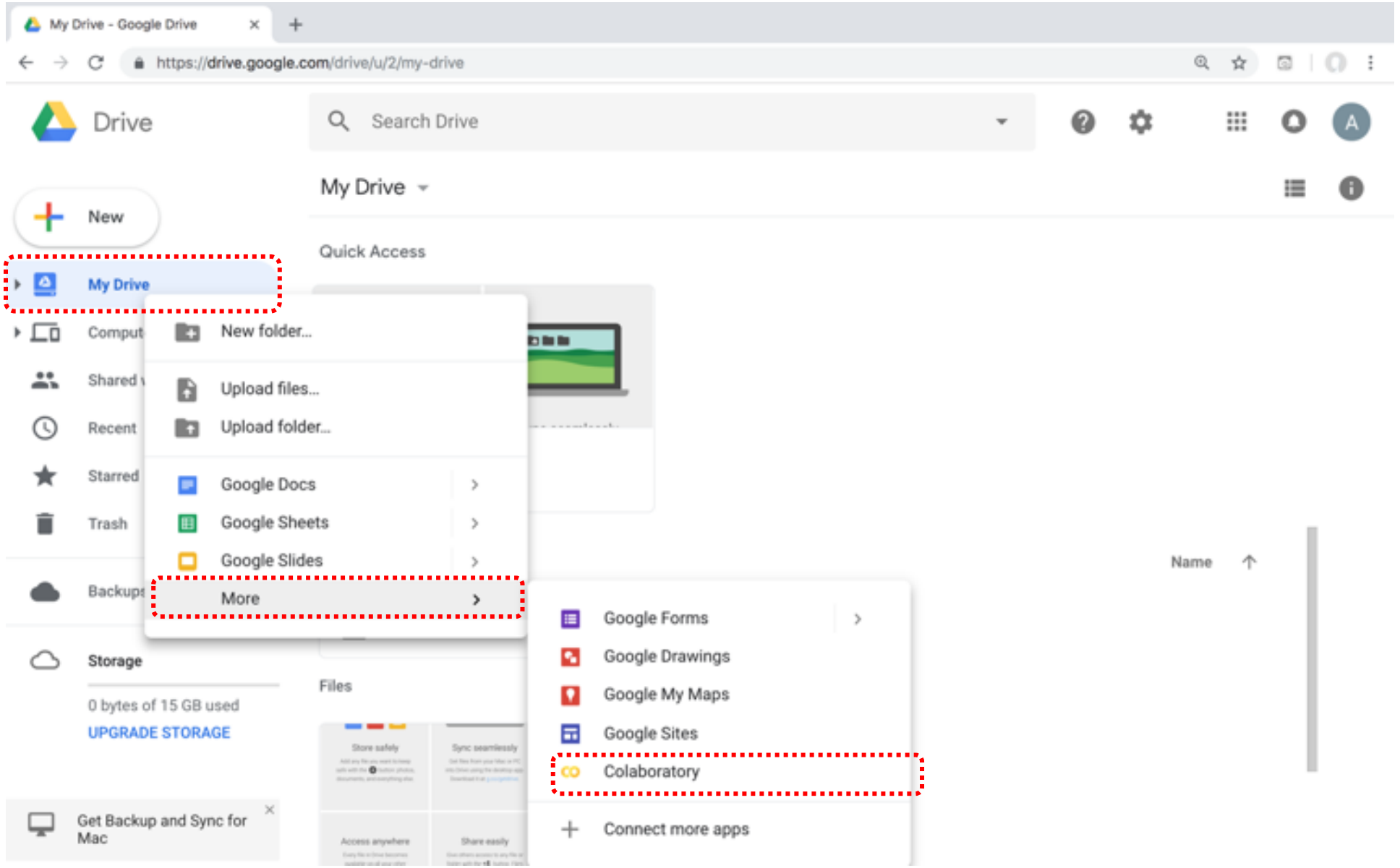
Google Colab



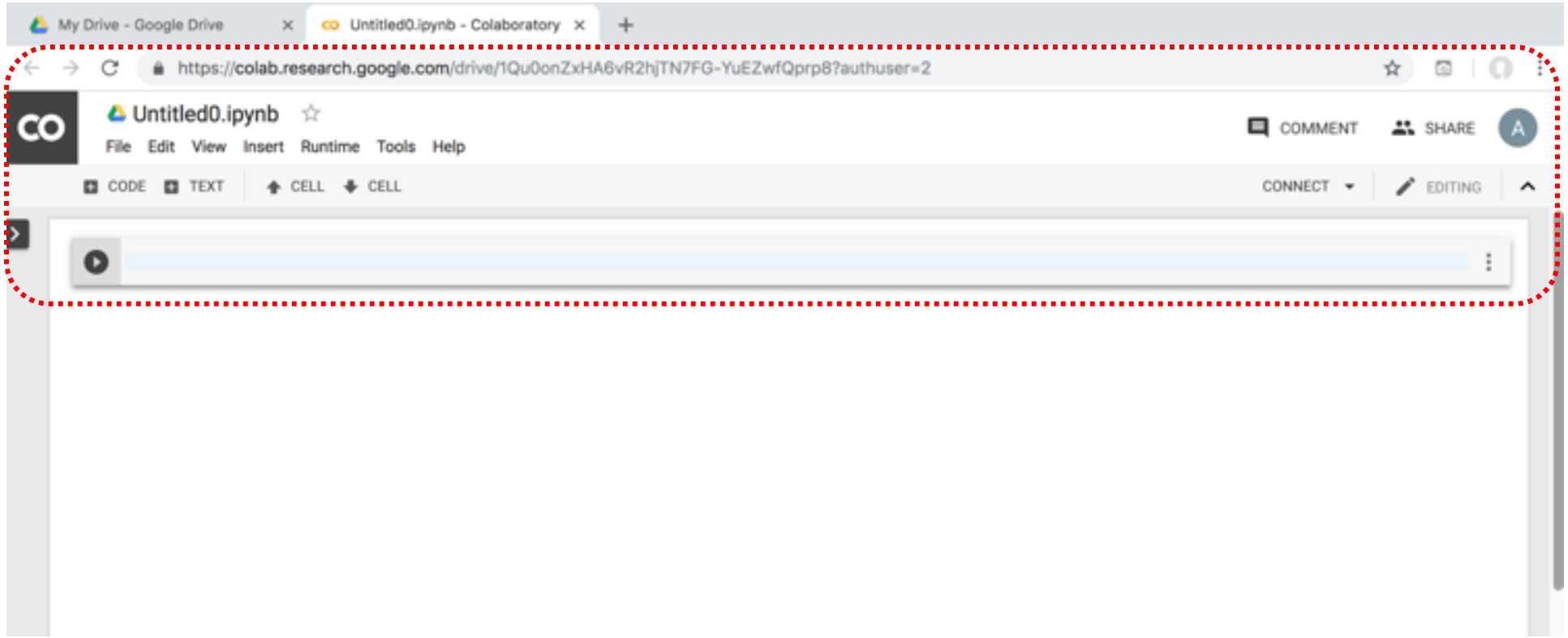
Connect Colaboratory to Google Drive



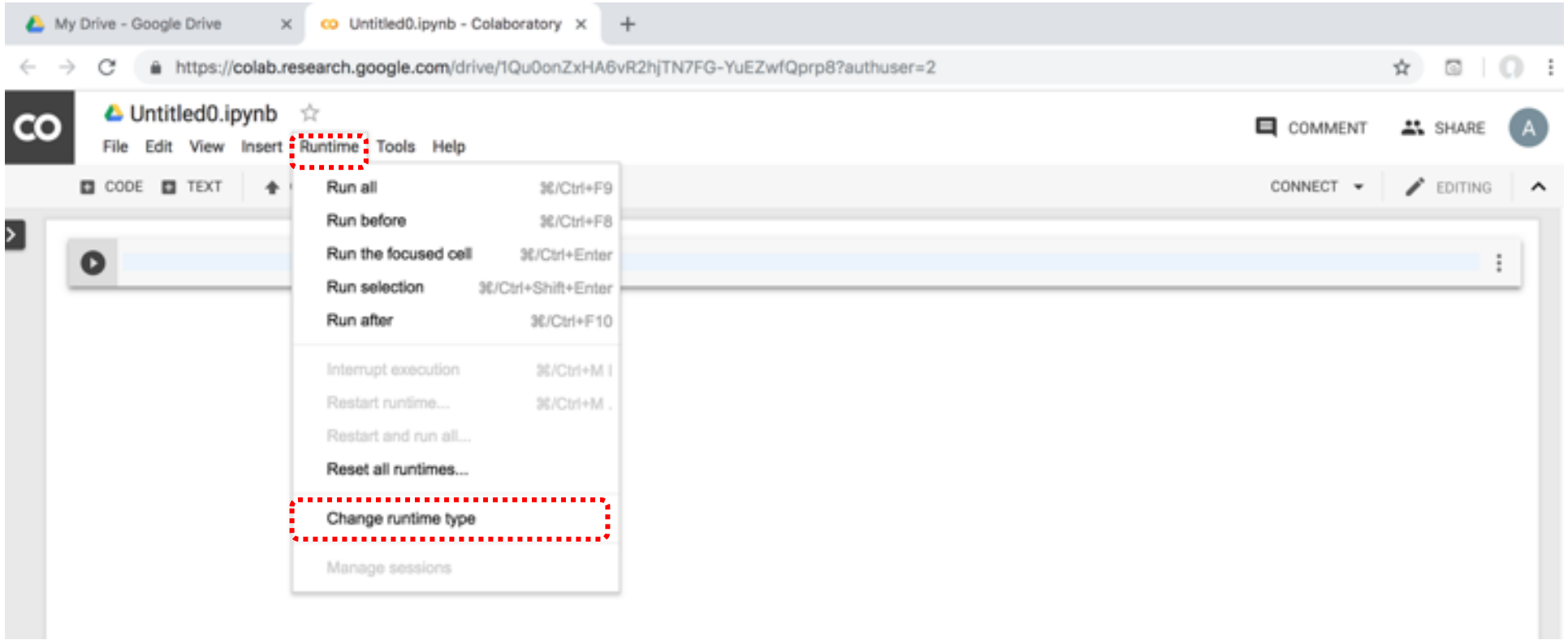
Google Colab



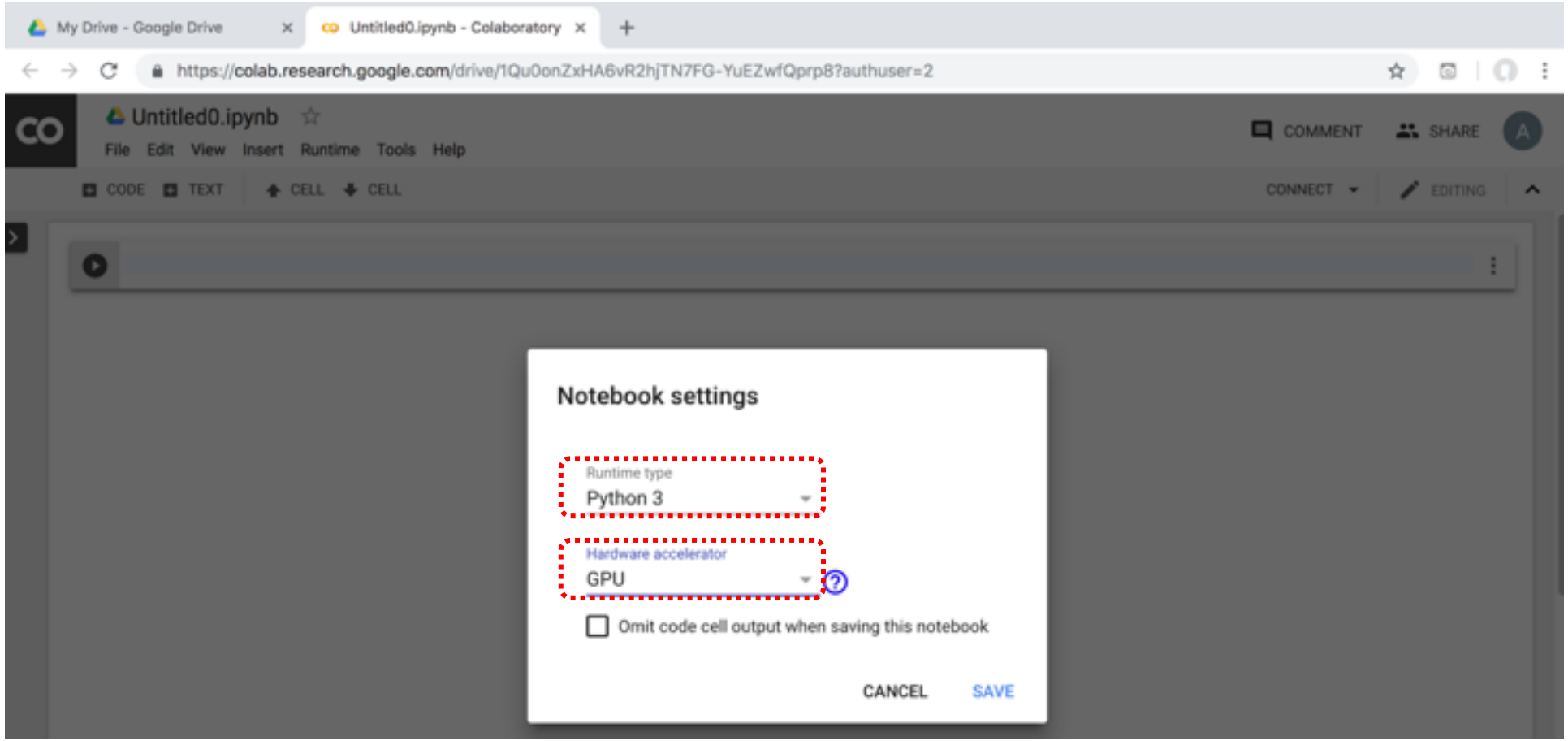
Google Colab



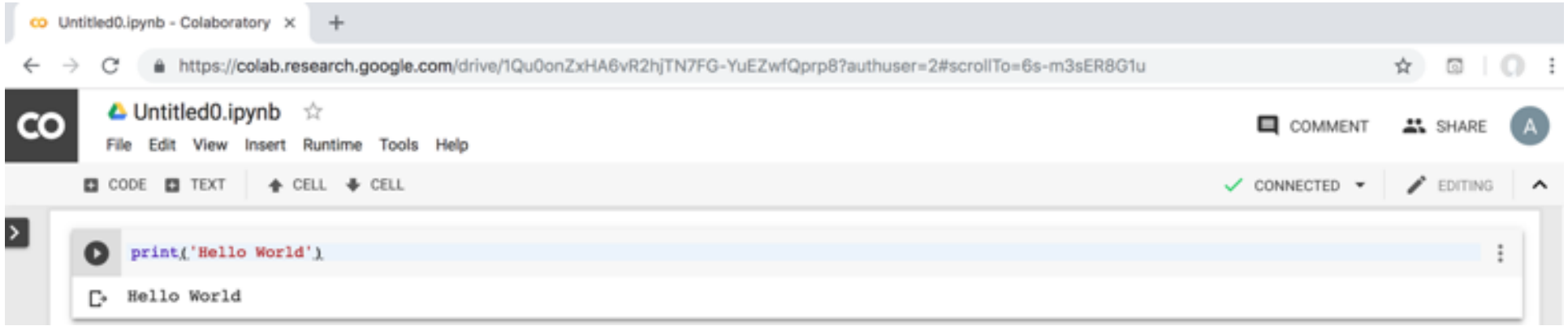
Google Colab



Run Jupyter Notebook
Python3 GPU
Google Colab



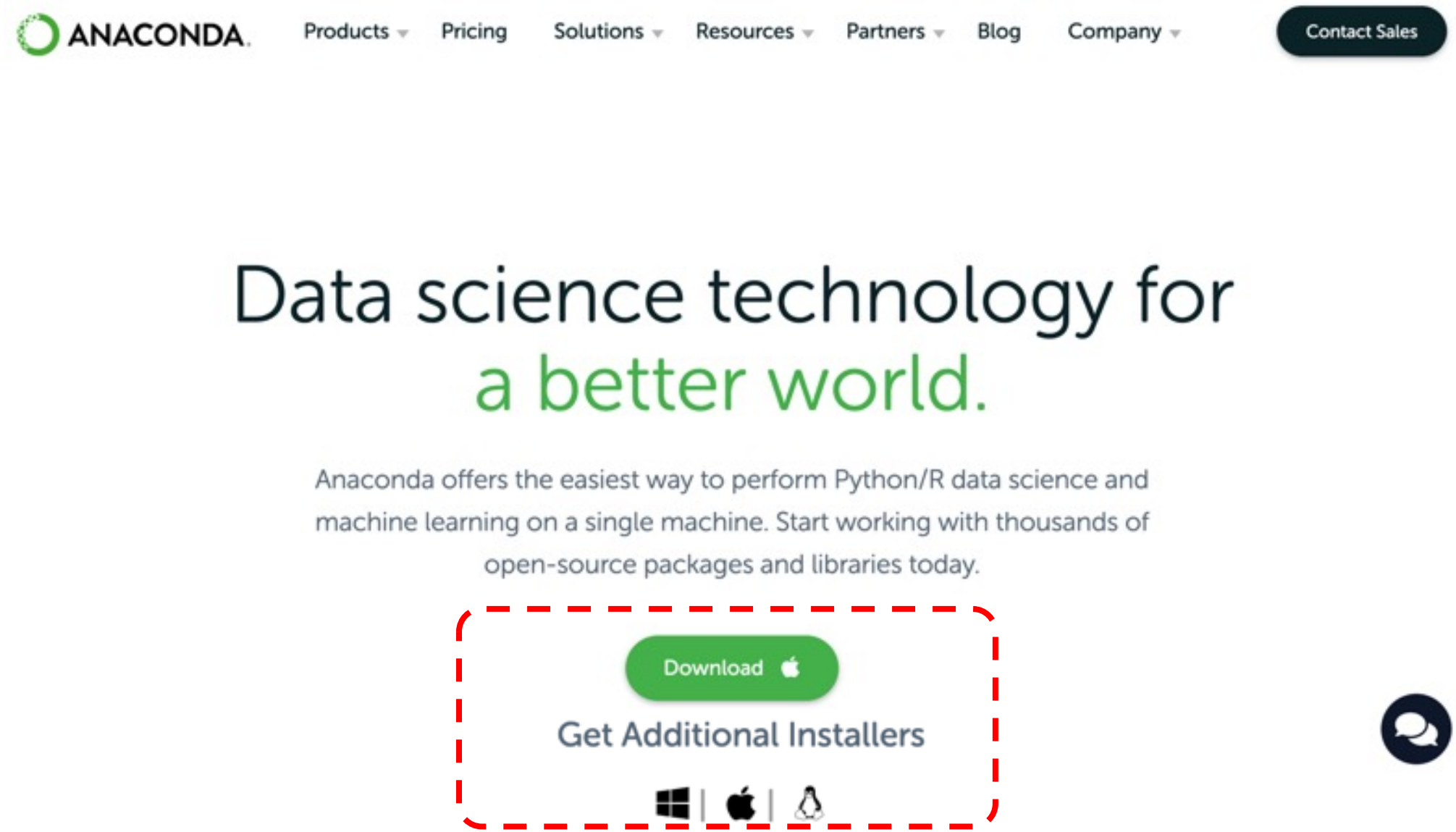
Google Colab Python Hello World
print('Hello World')



Anaconda
The Most Popular
Python
Data Science Platform

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

Download Anaconda

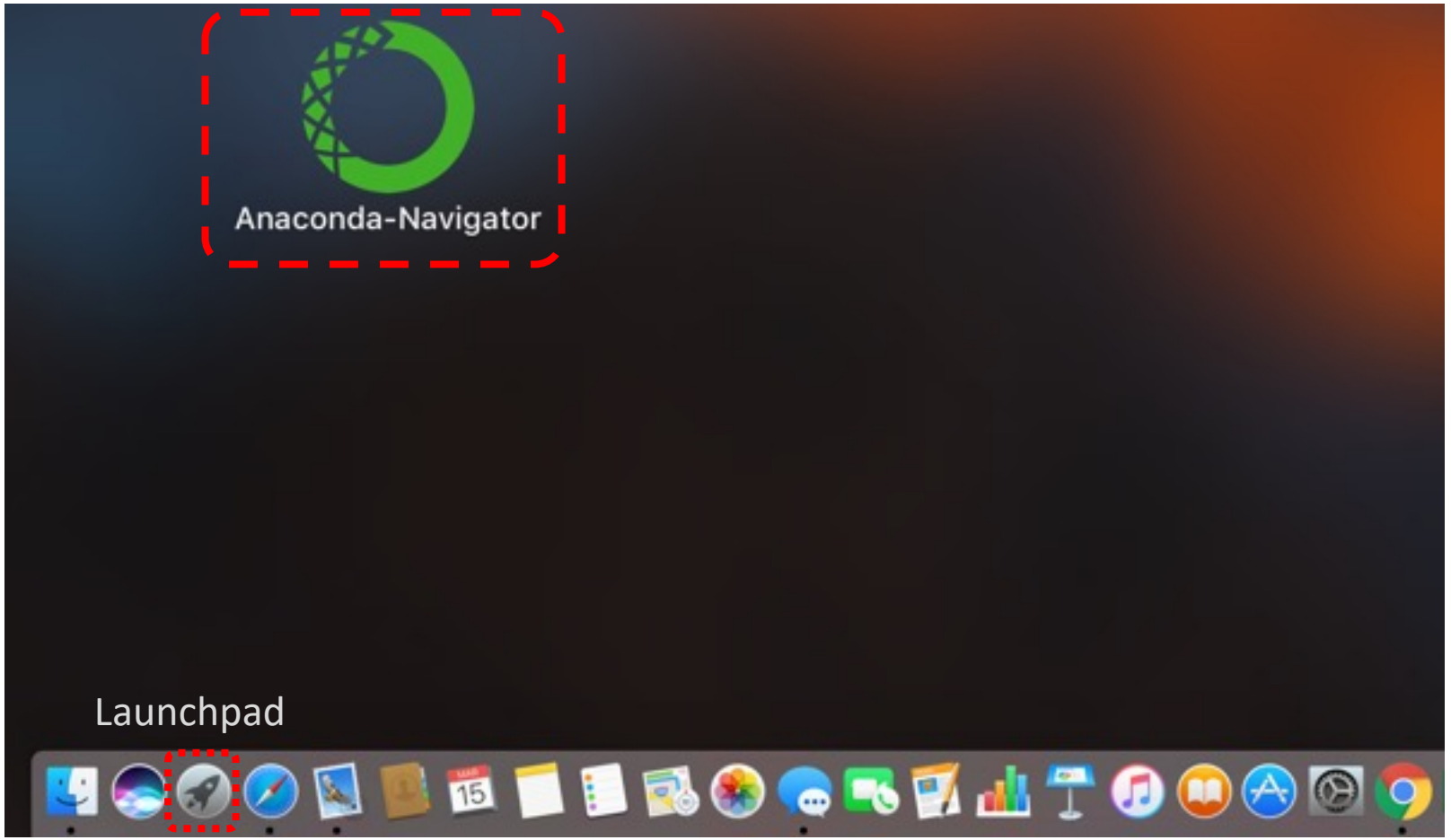


https://www.anaconda.com/download

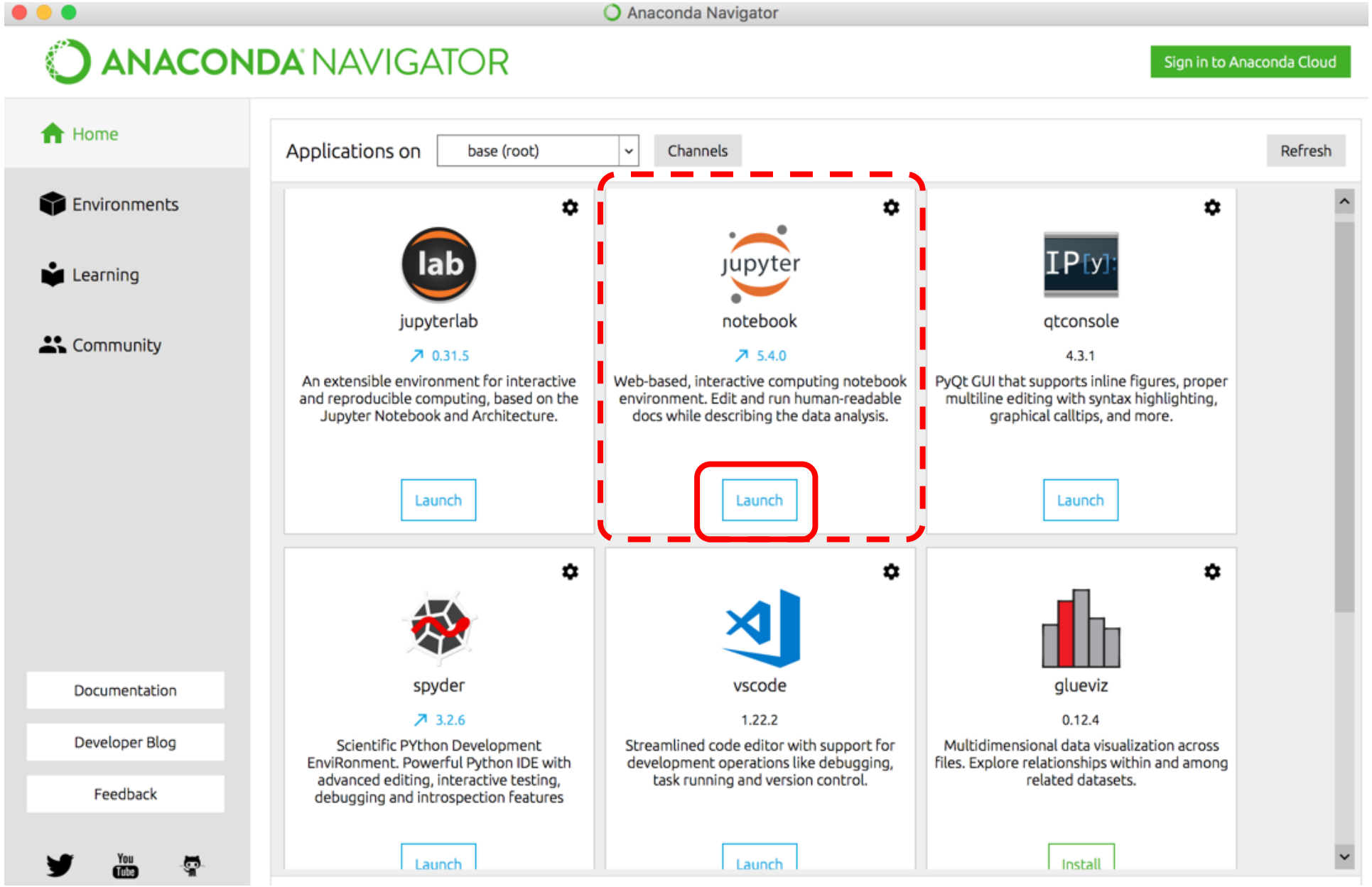
Python

HelloWorld

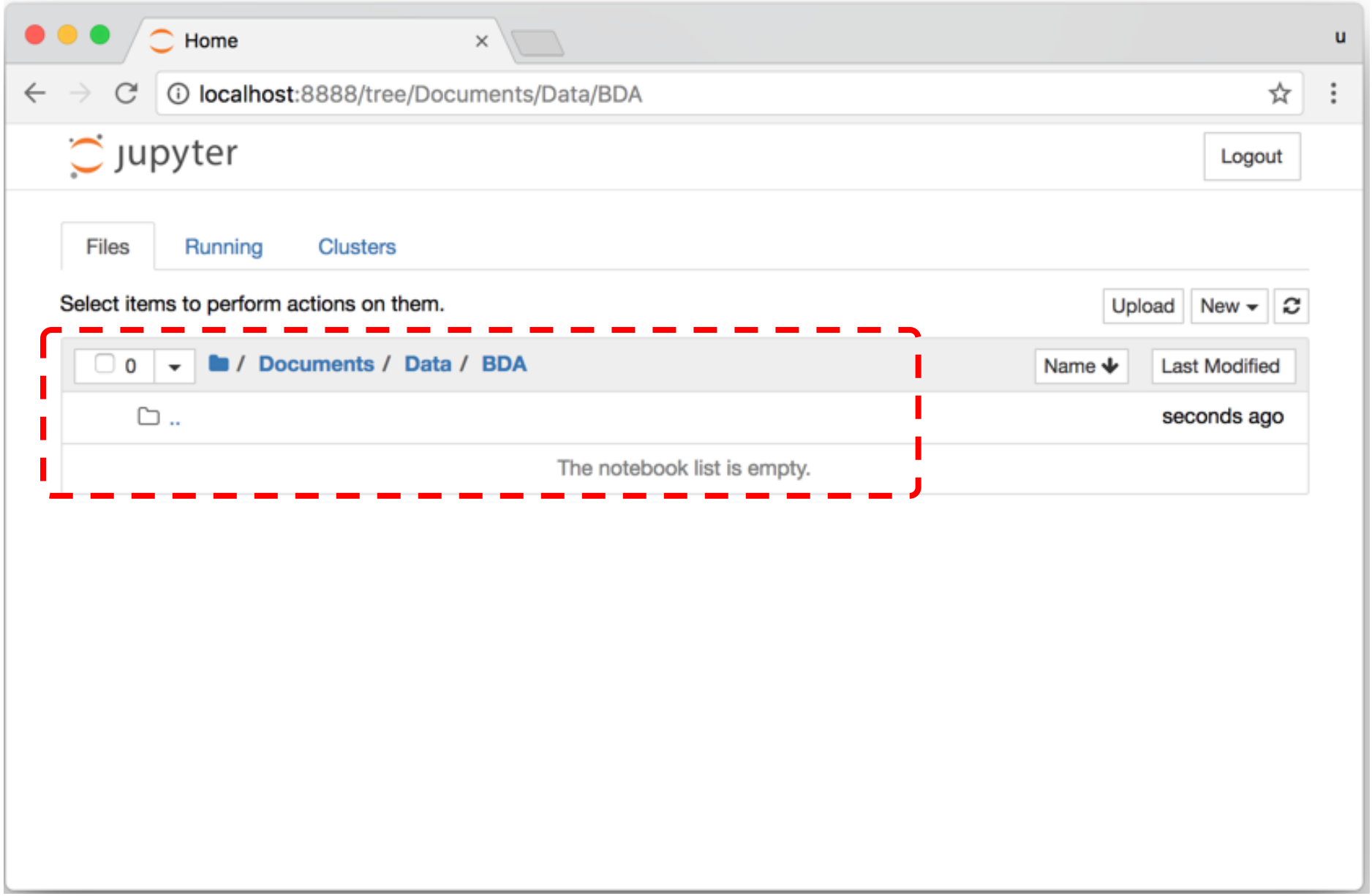
Anaconda-Navigator



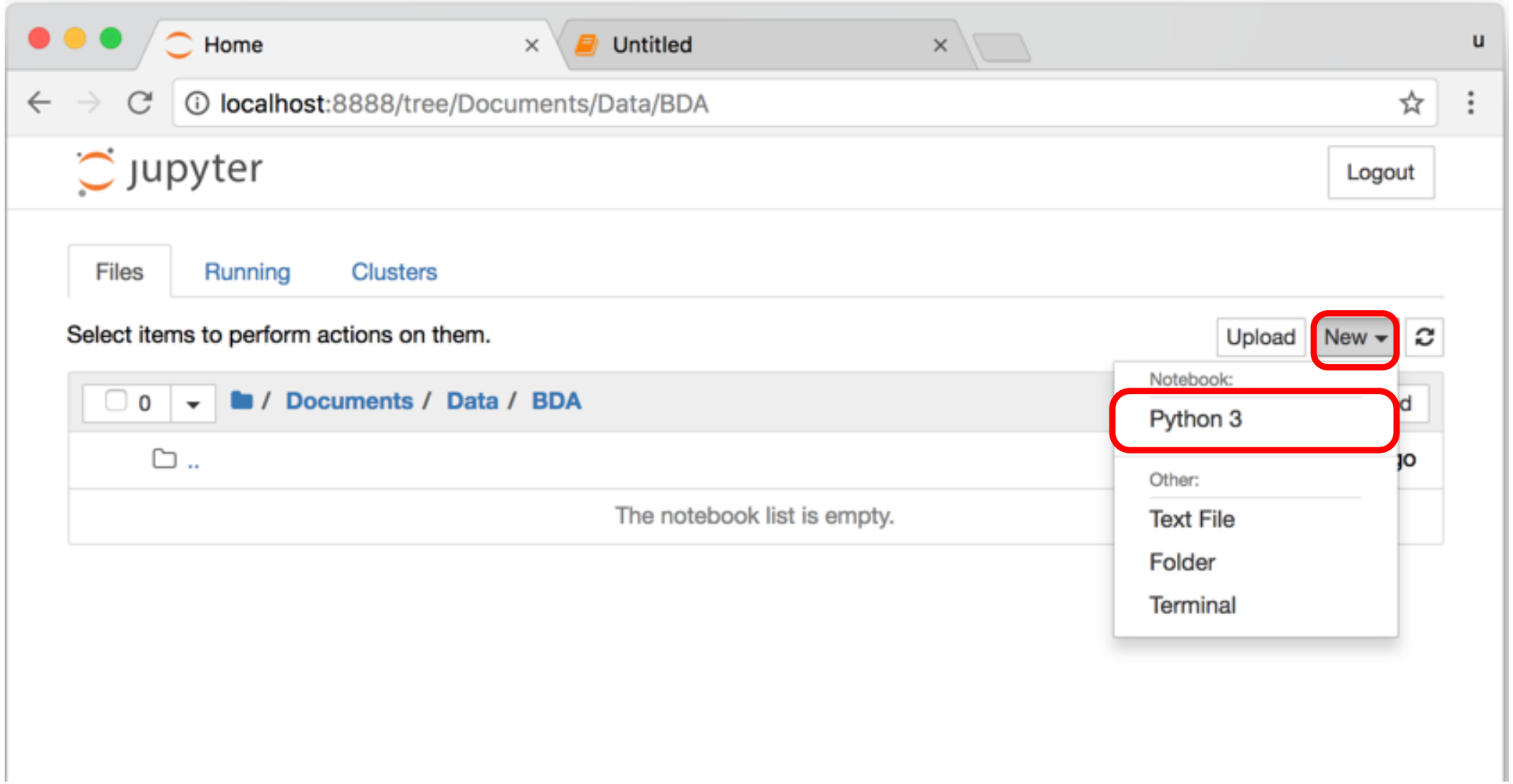
Anaconda Navigator



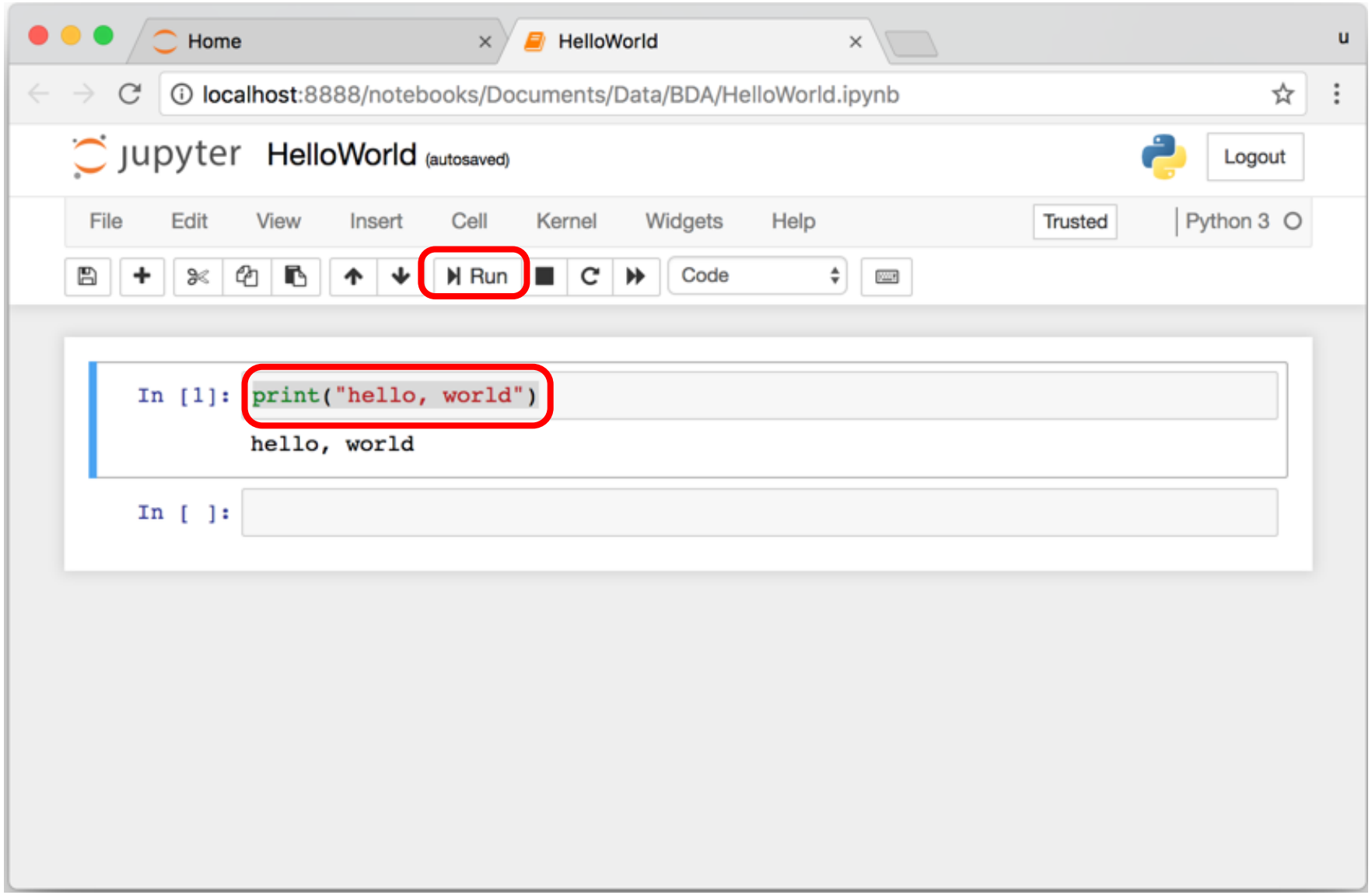
Jupyter Notebook



Jupyter Notebook
New Python 3



print("hello, world")

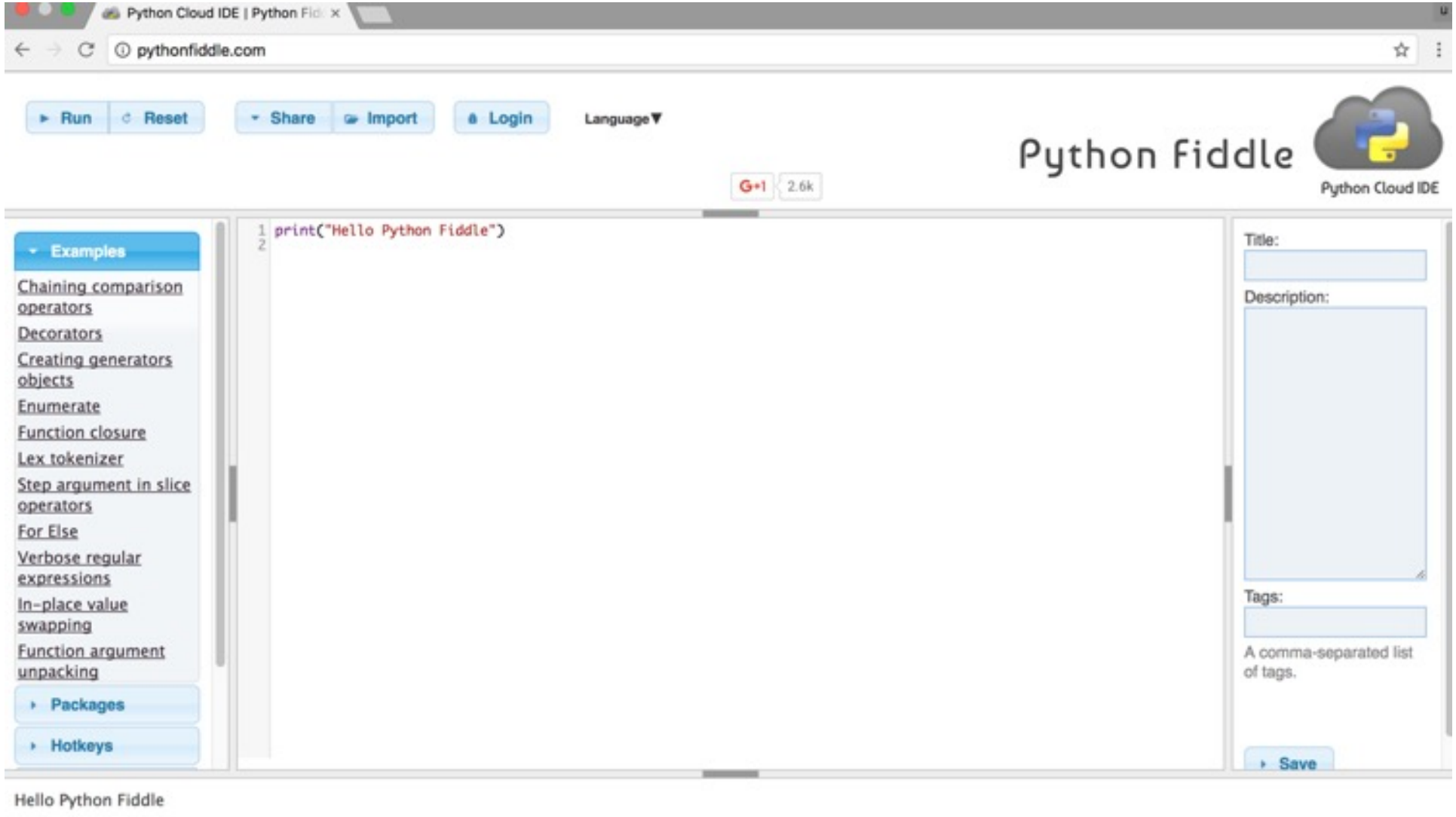


Python

Programming

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

Python Fiddle



http://pythonfiddle.com/

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: ")
```

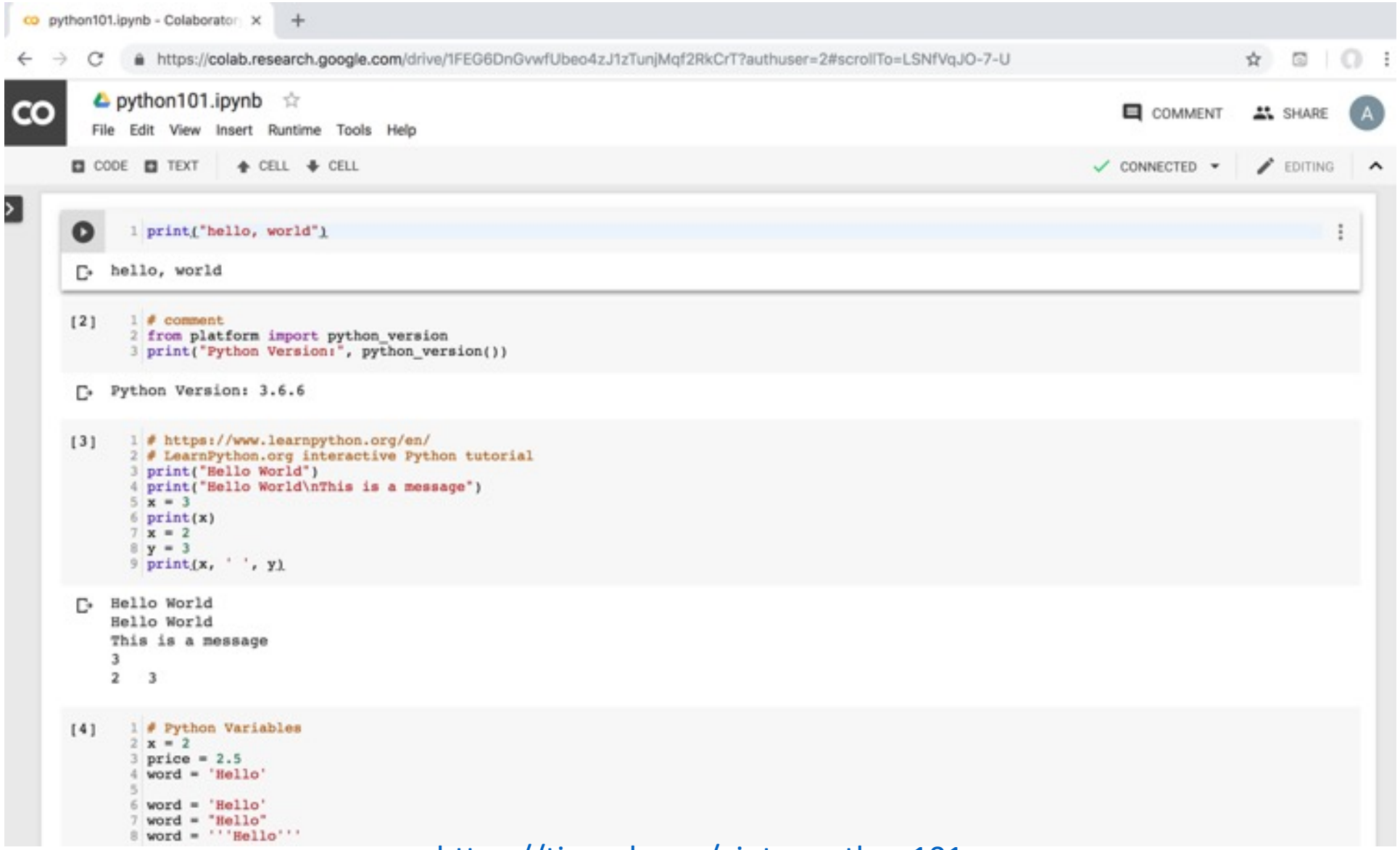
```
x = int(input("What is x? "))
```

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

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

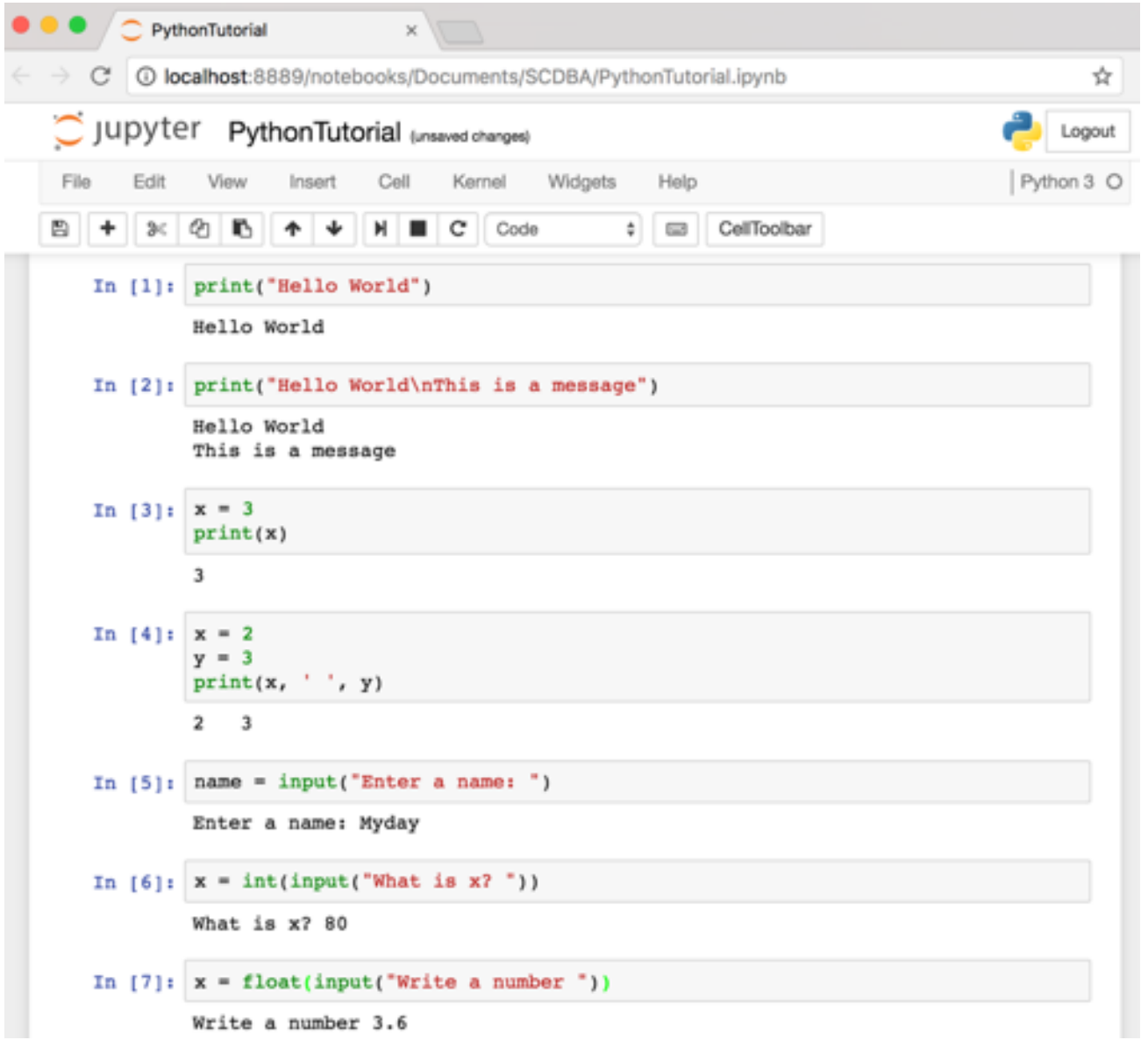
Python in Google Colab

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



https://tinyurl.com/aintpupython101

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 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]:
```
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

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

```
Pass
```


```
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/

python™

for loops

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

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

Source: http://pythonprogramminglanguage.com/

python™

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/

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

def Functions

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

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

```
1.8
```

Lists []

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

```
4
60
70
90
```

Tuples ()

A tuple in Python is a collection that cannot be modified.

A tuple is defined using parenthesis.

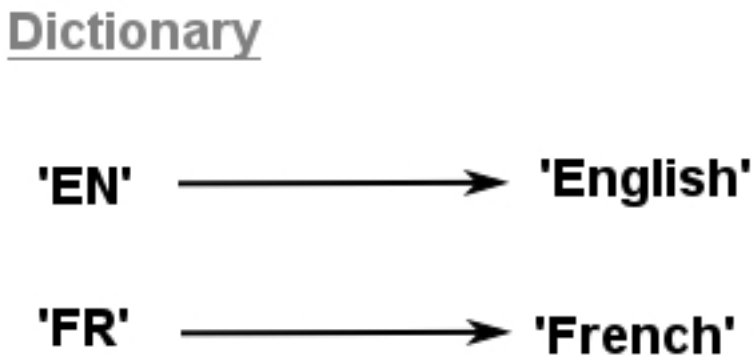
```
x = (10, 20, 30, 40, 50)
print(x[0])
print(x[1])
print(x[2])
print(x[-1])
```

```
10
20
30
50
```

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

Dictionary {key : value}

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



English

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

python™

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))
```

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

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

try except finally

```
try:
    file = open("myfile.txt")
    #file = open("myfile.txt", 'w')
    file.write("Python write file")
    print("file saved")
except:
    print("Exception file Error")
finally:
    file.close()
    print("finally process")
```

```
Exception file Error
finally process
```

Source: https://pythonbasics.org/try-except/

class

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person("Alan", 20)
p1.myfunc()
print(p1.name)
print(p1.age)
```

```
Hello my name is Alan
Alan
20
```

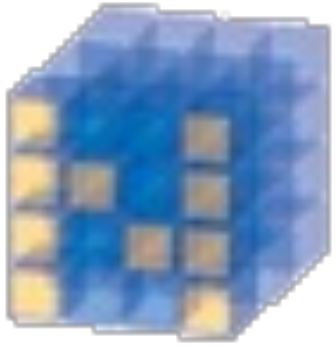
Source: https://www.w3schools.com/python/python_classes.asp

Big Data Analytics with

Numpy

in Python

Numpy



NumPy

Base

N-dimensional array

package

NumPy

is the

fundamental package

for

scientific computing

with Python.

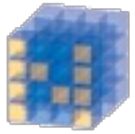
Source: http://www.numpy.org/

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

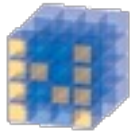
NumPy ndarray

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**

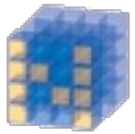


NumPy

NumPy

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

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



NumPy

Base
N-dimensional
array package

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

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

```
1 2 * v
```

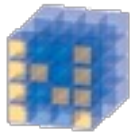
```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

```
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

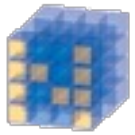
NumPy Create Array

```
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])
```

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/

```
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
```

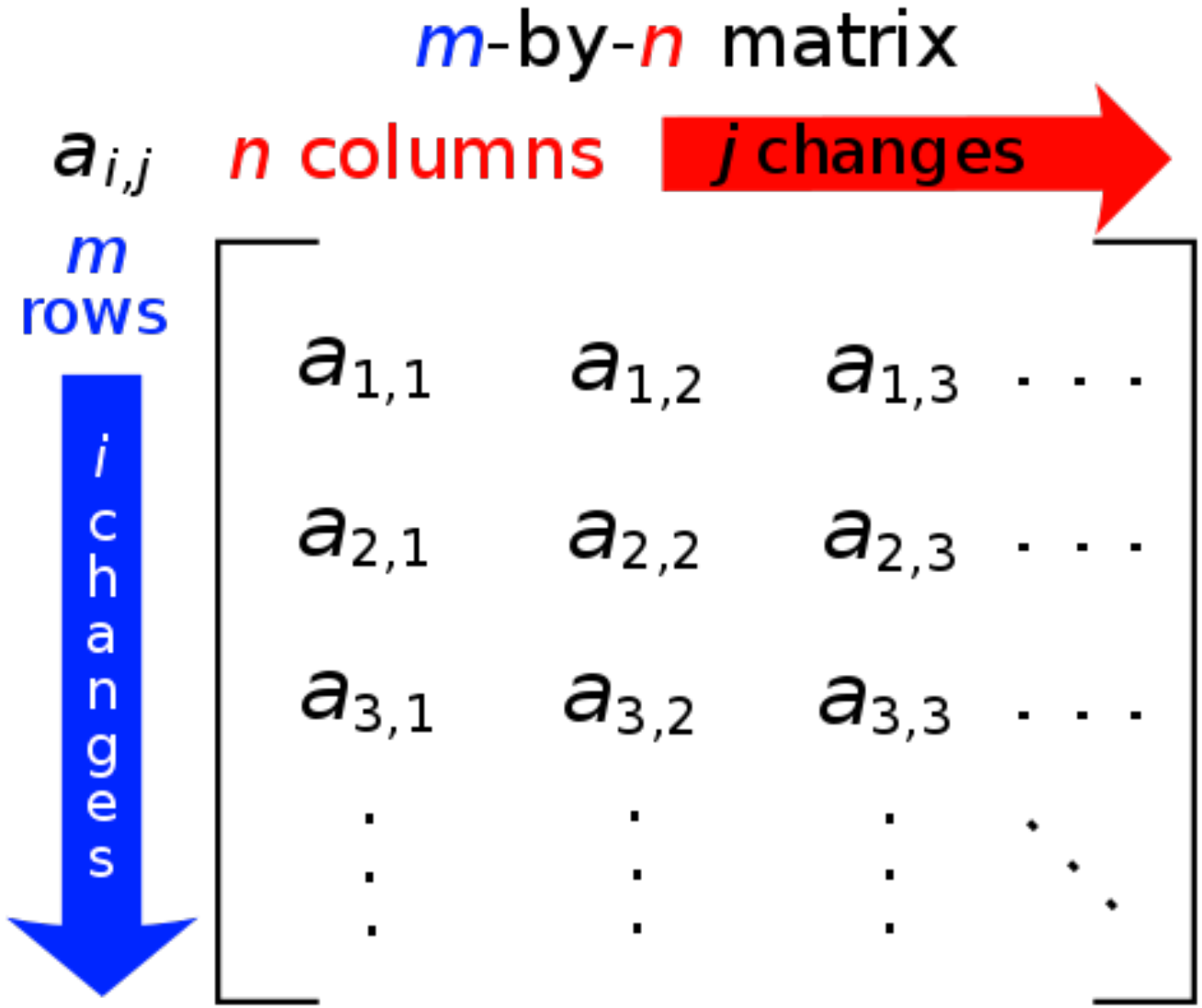
```
2
```

```
a.dtype.name
```

```
'int64'
```

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

Matrix



Source: https://simple.wikipedia.org/wiki/Matrix_(mathematics)

NumPy ndarray:
Multidimensional Array Object

NumPy ndarray

One-dimensional Array (1-D Array)

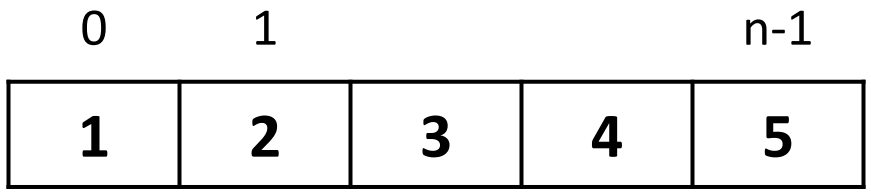
0	1			n-1
1	**2**	**3**	**4**	**5**

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)




```
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)
```

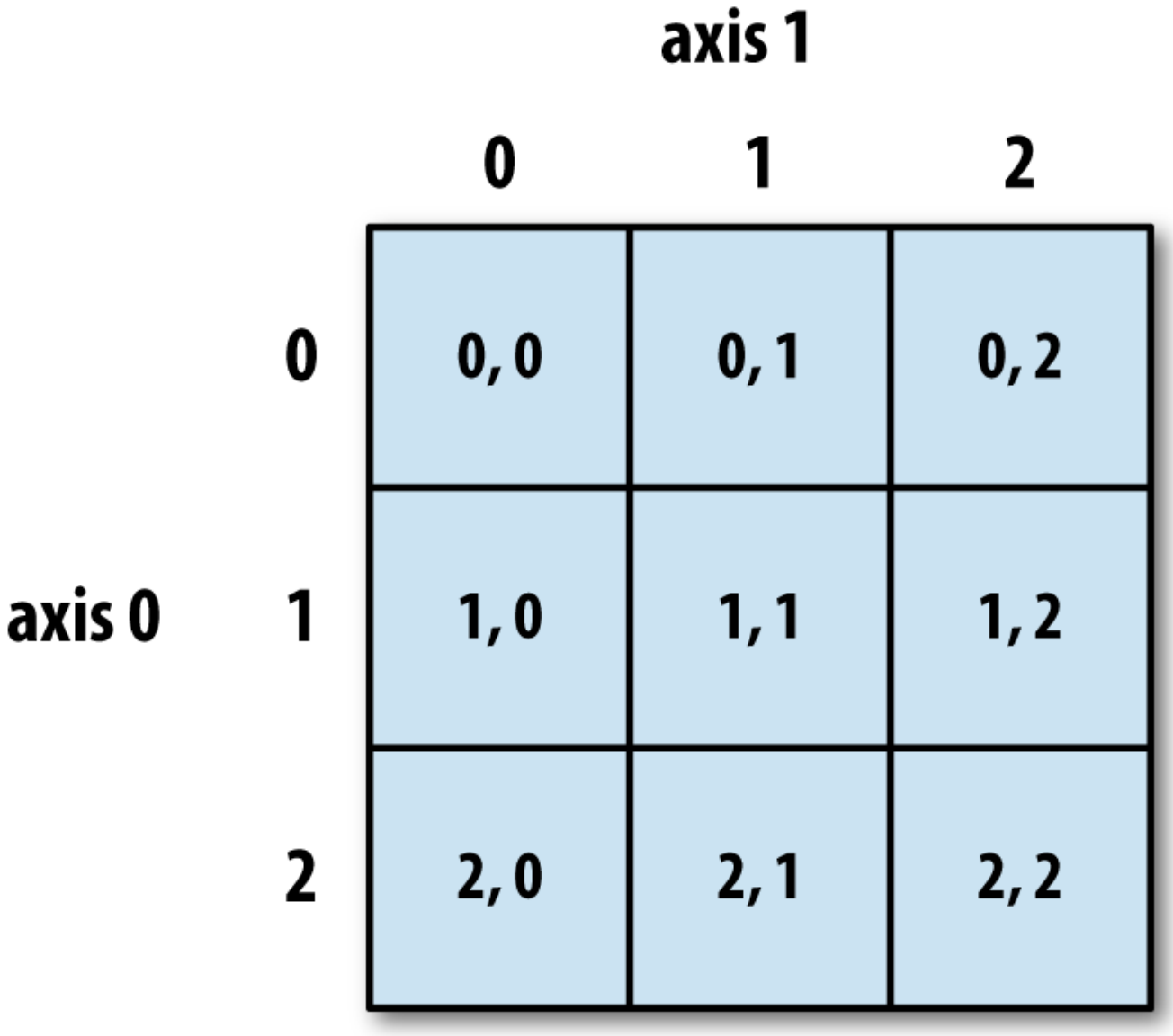
```
(3, 4)
```

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

NumPy Basics: Arrays and Vectorized Computation

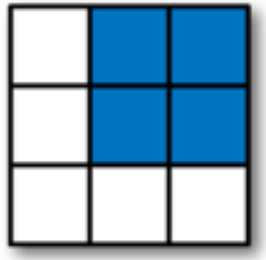
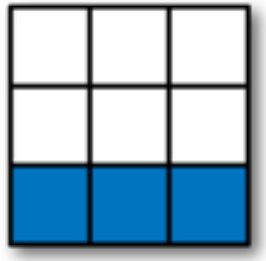
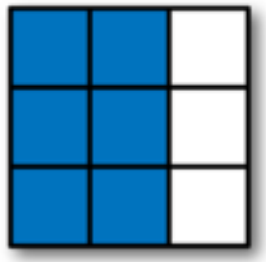
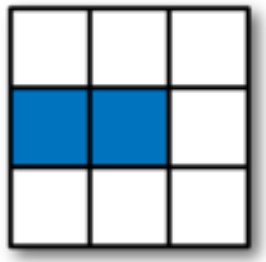
Source: https://www.safaribooksonline.com/library/view/python-for-data/9781449323592/ch04.html

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

TensorFlow

Tensor

- 3
  - a rank 0 tensor; this is a scalar with shape []
- [1. ,2., 3.]
  - a rank 1 tensor; this is a vector with shape [3]
- [[1., 2., 3.], [4., 5., 6.]]
  - a rank 2 tensor; a matrix with shape [2, 3]
- [[[1., 2., 3.]], [[7., 8., 9.]]]
  - a rank 3 tensor with shape [2, 1, 3]

https://www.tensorflow.org/

Scalar 80

Vector [50 60 70]

Matrix

$$\begin{bmatrix} 50 & 60 & 70 \\ 55 & 65 & 75 \end{bmatrix}$$

Tensor

$$\begin{bmatrix} [50 & 60 & 70] & [70 & 80 & 90] \\ [55 & 65 & 75] & [75 & 85 & 95] \end{bmatrix}$$

pandas

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/

pandas:
powerful Python data analysis toolkit

- **Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet**
- **Ordered and unordered (not necessarily fixed-frequency) time series data.**
- **Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels**
- **Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure**

Source: http://pandas.pydata.org/pandas-docs/stable/

pandas

Series
DataFrame

- Primary data structures of pandas
  - Series (1-dimensional)
  - DataFrame (2-dimensional)
- Handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.

Source: http://pandas.pydata.org/pandas-docs/stable/

pandas

pandas DataFrame

- **DataFrame provides everything that R's data.frame provides and much more.**
- **pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.**

pandas

pandas
Comparison with SAS

pandas	SAS
DataFrame	data set
column	variable
row	observation
groupby	BY-group
NaN	.

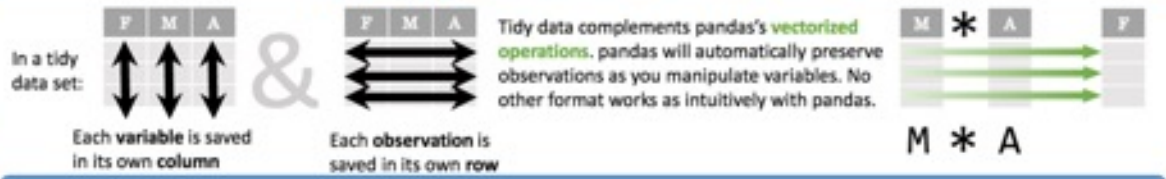
Source: http://pandas.pydata.org/pandas-docs/stable/comparison_with_sas.html

Python Pandas Cheat Sheet

Data Wrangling with pandas Cheat Sheet

http://pandas.pydata.org

Tidy Data – A foundation for wrangling in pandas



Tidy data complements pandas's vectorized operations. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.

Syntax – Creating DataFrames

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

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```
Specify values for each row.

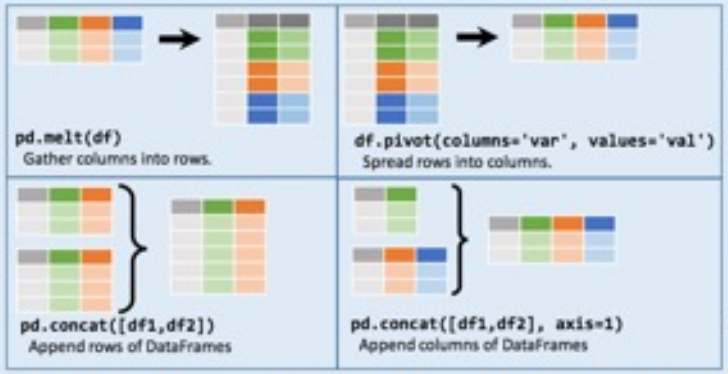
```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
        [('d',1),('d',2),('e',2)],
         names=['n','v'])))
```
Create DataFrame with a MultiIndex

Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
        .rename(columns={
                'variable' : 'var',
                'value' : 'val'})
        .query('val >= 200')
    )
```

Reshaping Data – Change the layout of a data set



pd.melt(df)
Gather columns into rows.

df.pivot(columns='var', values='val')
Spread rows into columns.

pd.concat([df1,df2])
Append rows of DataFrames

pd.concat([df1,df2], axis=1)
Append columns of DataFrames

df=df.sort_values('mpg')
Order rows by values of a column (low to high).

df=df.sort_values('mpg',ascending=False)
Order rows by values of a column (high to low).

df=df.rename(columns = {'y':'year'})
Rename the columns of a DataFrame

df=df.sort_index()
Sort the index of a DataFrame

df=df.reset_index()
Reset index of DataFrame to row numbers, moving index to columns.

df=df.drop(['Length','Height'], axis=1)
Drop columns from DataFrame

Subset Observations (Rows)

df[df.Length > 7]
Extract rows that meet logical criteria.

df.drop_duplicates()
Remove duplicate rows (only considers columns).

df.head(n)
Select first n rows.

df.tail(n)
Select last n rows.

df.sample(frac=0.5)
Randomly select fraction of rows.

df.sample(n=10)
Randomly select n rows.

df.iloc[10:20]
Select rows by position.

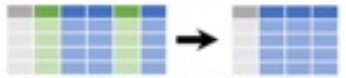
df.nlargest(n, 'value')
Select and order top n entries.

df.nsmallest(n, 'value')
Select and order bottom n entries.

Logic in Python (and pandas)

<	Less than	!=	Not equal to
>	Greater than	df.column.isin(values)	Group membership
==	Equals	pd.isnull(obj)	Is NaN
<=	Less than or equals	pd.notnull(obj)	Is not NaN
>=	Greater than or equals	&,|,~,^,df.any(),df.all()	Logical and, or, not, xor, any, all

Subset Variables (Columns)



df[['width','length','species']]
Select multiple columns with specific names.

df['width'] or **df.width**
Select single column with specific name.

df.filter(regex='regex')
Select columns whose name matches regular expression *regex*.

regex (Regular Expressions) Examples

'\.'	Matches strings containing a period '.'
'Length$'	Matches strings ending with word 'Length'
'^Sepal'	Matches strings beginning with the word 'Sepal'
'^x[1-5]$'	Matches strings beginning with 'x' and ending with 1,2,3,4,5
'^(?!Species$).*'	Matches strings except the string 'Species'

df.loc[:,'x2':'x4']
Select all columns between x2 and x4 (inclusive).

df.iloc[:,[1,2,5]]
Select columns in positions 1, 2 and 5 (first column is 0).

df.loc[df['a'] > 10, ['a','c']]
Select rows meeting logical condition, and only the specific columns .

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

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
```

pandas

```
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

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

```
pandas imported
```

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

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

```
1 dates = pd.date_range('20181001', periods=6)
2 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')
```

pandas

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

```
1 df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))
2 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
```

```
1 df = pd.DataFrame(np.random.randn(3,5), index=['student1','student2','student3'], columns=list('ABCDE'))
2 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

pandas

```
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
```

```
1 df2 = pd.DataFrame({ 'A' : 1.,
2 'B' : pd.Timestamp('20181001'),
3 'C' : pd.Series(2.5,index=list(range(4)),dtype='float32'),
4 'D' : np.array([3] * 4,dtype='int32'),
5 'E' : pd.Categorical(["test","train","test","train"]),
6 'F' : 'foo' })
7 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

pandas

df2.dtypes

```
df2.dtypes
```

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

Python Data Analysis and Visualization

Python Pandas

http://pandas.pydata.org/

Python matplotlib

Source: https://matplotlib.org/

Python seaborn

seaborn

Source: https://seaborn.pydata.org/

Python plotly

Source: https://plotly.com/python/

Python bokeh

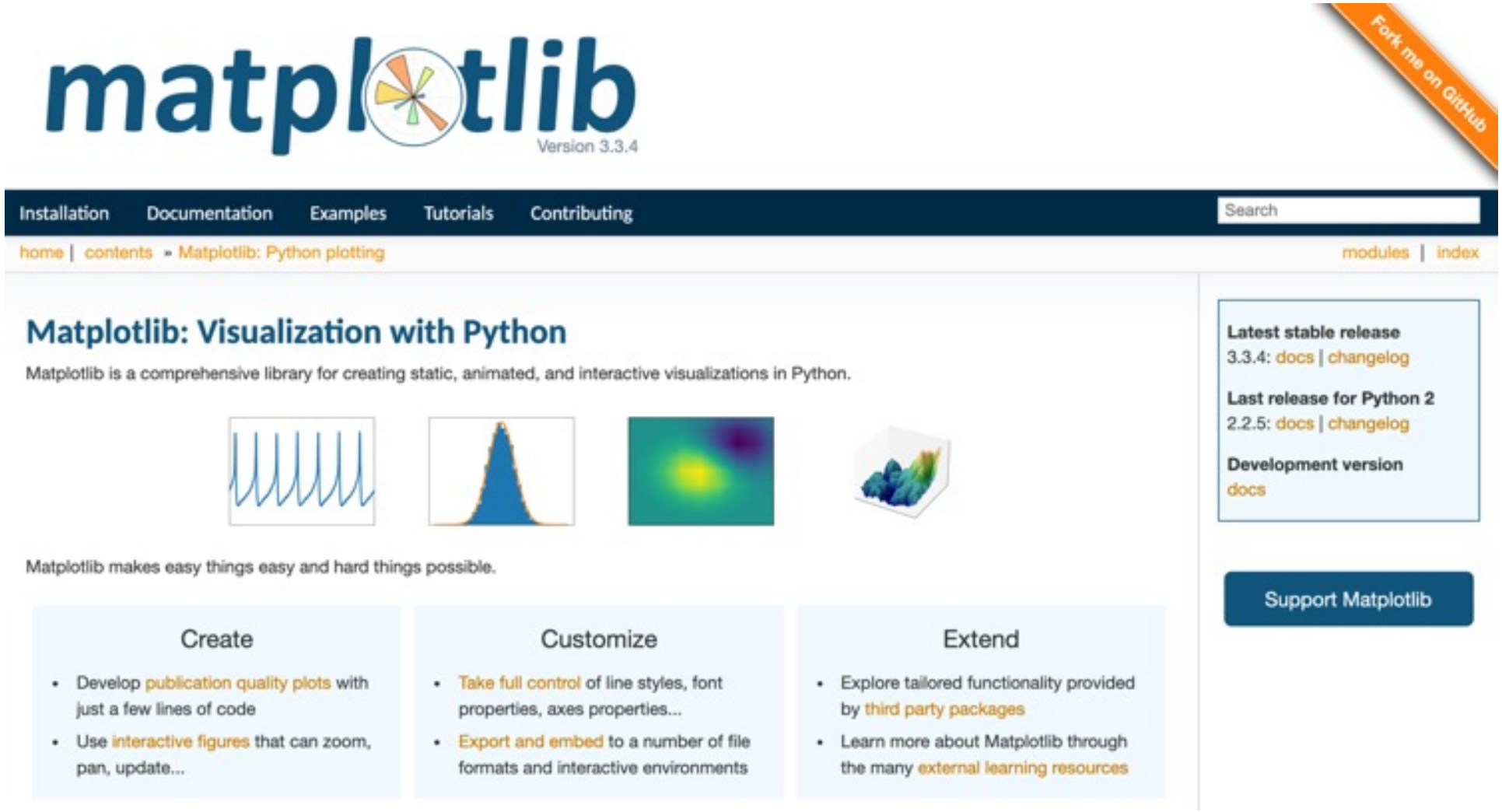
Source: https://bokeh.org/

Python Altair

Altair

Source: https://altair-viz.github.io/

Python matplotlib

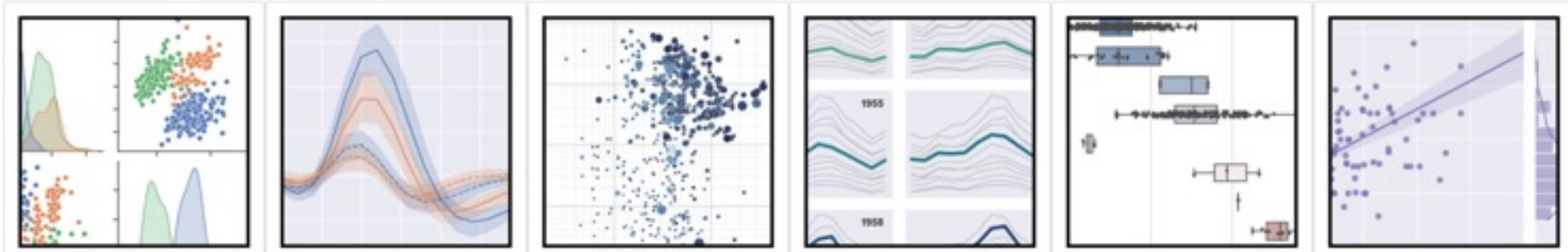


https://matplotlib.org/

Python Seaborn

seaborn 0.11.1 Gallery Tutorial API Site Page Search

seaborn: statistical data visualization



Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

For a brief introduction to the ideas behind the library, you can read the introductory notes. Visit the installation page to see how you can download the package and get started with it. You can browse the example gallery to see what you can do with seaborn, and then check out the tutorial and API reference to find out how.

To see the code or report a bug, please visit the GitHub repository. General support questions are most at home on stackoverflow or discourse, which have dedicated channels for seaborn.

Contents

- Introduction
- Release notes
- Installing
- Example gallery
- Tutorial
- API reference

Features

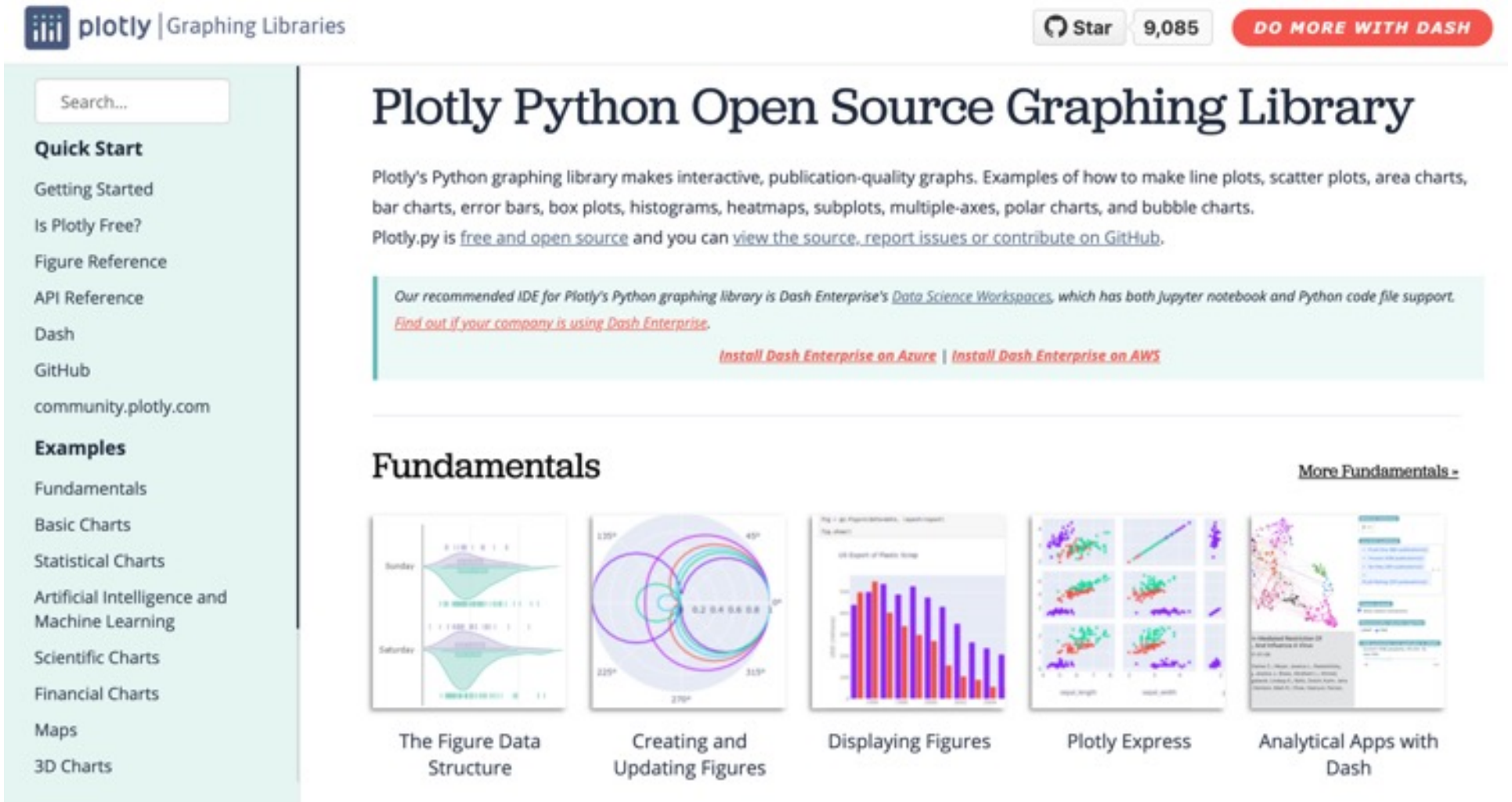
- Relational: API | Tutorial
- Distribution: API | Tutorial
- Categorical: API | Tutorial
- Regression: API | Tutorial
- Multiples: API | Tutorial
- Style: API | Tutorial
- Color: API | Tutorial

Back to top

https://seaborn.pydata.org/

plotly

Python Plotly Graphing Library



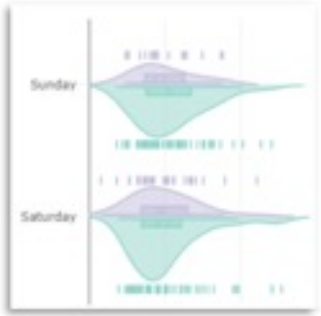
https://plotly.com/python/

plotly

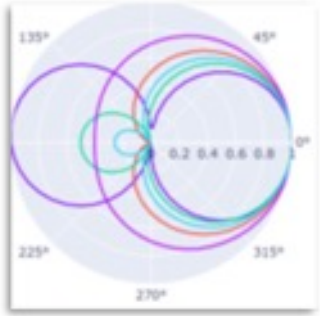
Python Plotly Graphing Library

Fundamentals

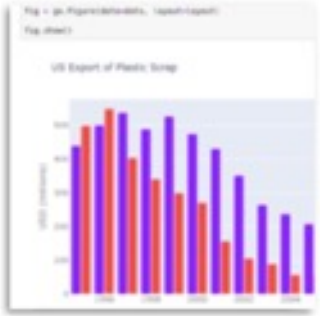
More Fundamentals »



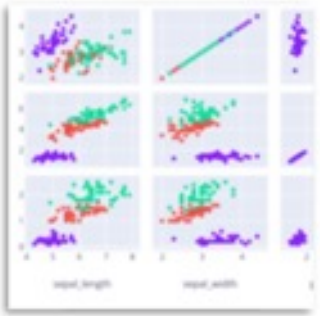
The Figure Data Structure



Creating and Updating Figures



Displaying Figures

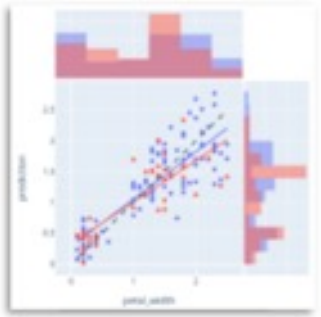


Plotly Express

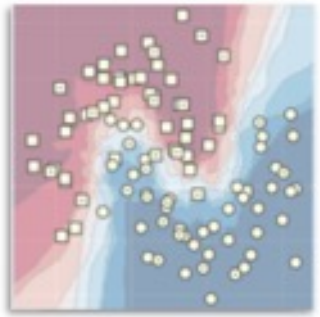
Analytical Apps with Dash

Artificial Intelligence and Machine Learning

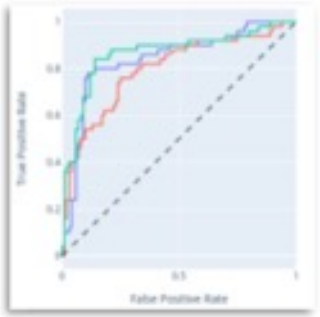
More AI and ML »



ML Regression



kNN Classification



ROC and PR Curves

PCA Visualization

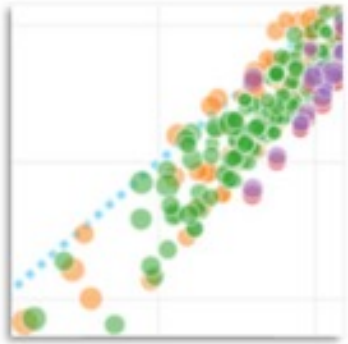
AI/ML Apps with Dash

https://plotly.com/python/

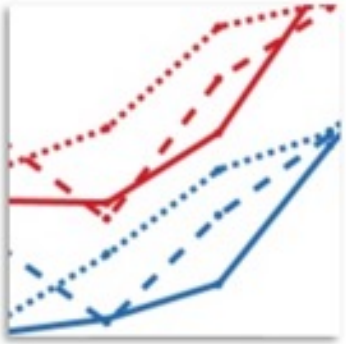
Python Plotly Graphing Library

Basic Charts

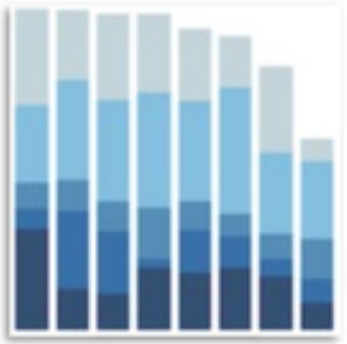
More Basic Charts »



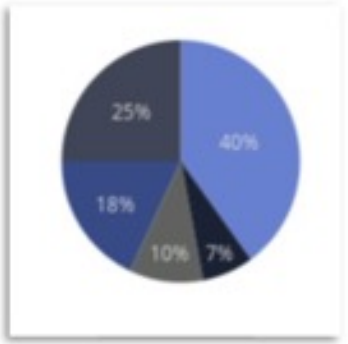
Scatter Plots



Line Charts



Bar Charts



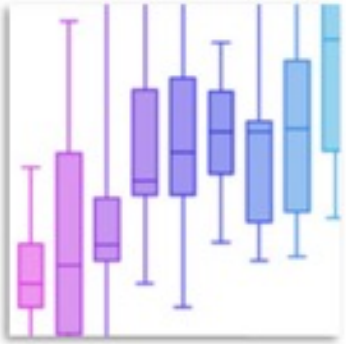
Pie Charts

Bubble Charts

Statistical Charts

More Statistical Charts »

Error Bars



Box Plots

Histograms

Distplots

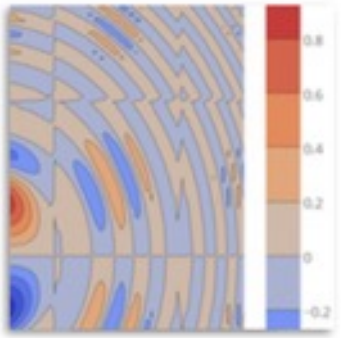
2D Histograms

https://plotly.com/python/

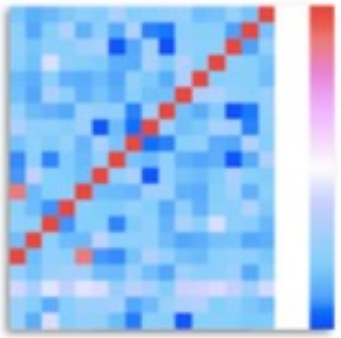
Python Plotly Graphing Library

Scientific Charts

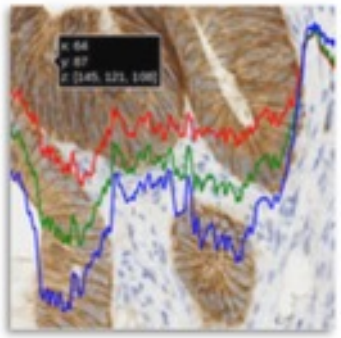
More Scientific Charts »



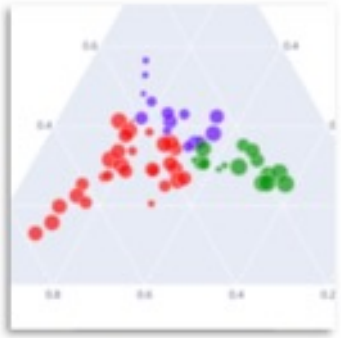
Contour Plots



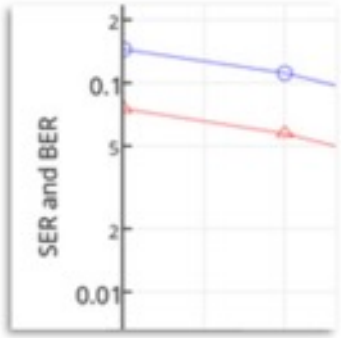
Heatmaps



Imshow



Ternary Plots



Log Plots

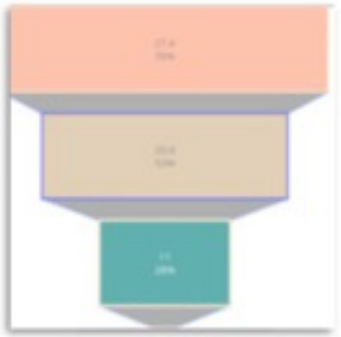
Financial Charts

More Financial Charts »

Time Series and Date Axes

Candlestick Charts

Waterfall Charts



Funnel Chart

OHLC Charts

https://plotly.com/python/

Python Plotly Graphing Library

Maps

More Maps »

Mapbox Choropleth Maps

Lines on Mapbox

Filled Area on Maps

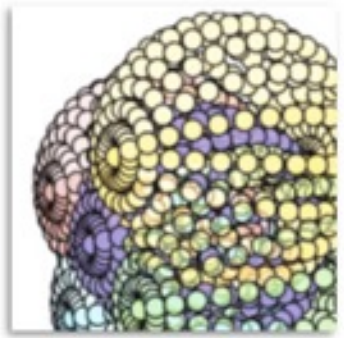
Bubble Maps

Mapbox Density Heatmap

3D Charts

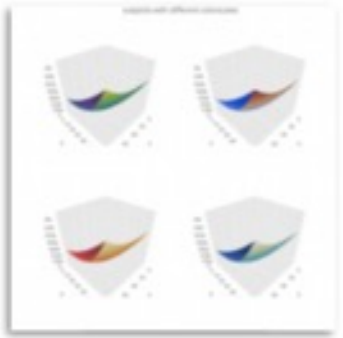
More 3D Charts »

3D Axes

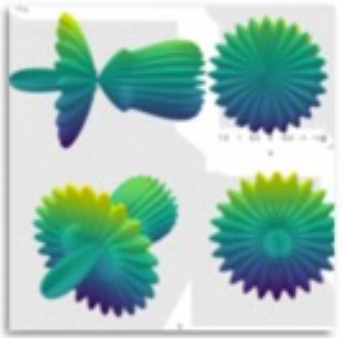


3D Scatter Plots

3D Surface Plots



3D Subplots



3D Camera Controls

https://plotly.com/python/

Python Plotly Graphing Library

Subplots

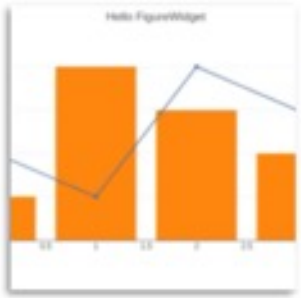
Mixed Subplots

Map Subplots

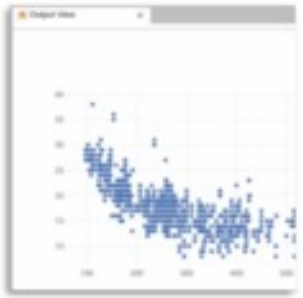
Table and Chart Subplots

Figure Factory Subplots

Jupyter Widgets Interaction

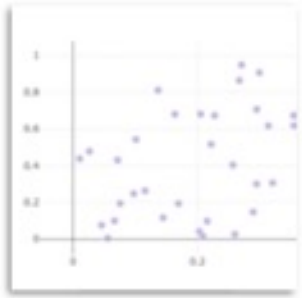


Plotly FigureWidget Overview



Jupyter Lab with FigureWidget

Interactive Data Analysis with FigureWidget ipywidgets



Click Events

https://plotly.com/python/

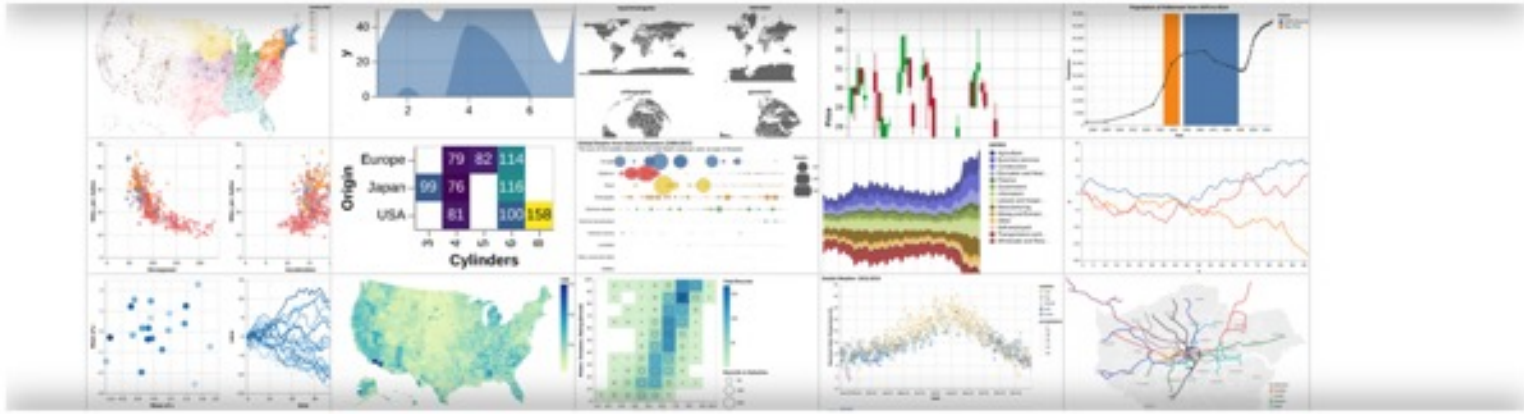
Python Bokeh

https://bokeh.org/

Python Altair

Vega-Altair Getting Started User Guide Examples API Reference Ecosystem Release Notes

Vega-Altair: Declarative Visualization in Python



Vega-Altair is a declarative statistical visualization library for Python, based on Vega and Vega-Lite.

The Vega-Altair open source project is not affiliated with Altair Engineering, Inc.

With Vega-Altair, you can spend more time understanding your data and its meaning. Altair's API is simple, friendly and consistent and built on top of the powerful Vega-Lite visualization grammar. This elegant simplicity produces beautiful and effective visualizations with a minimal amount of code.

You can browse this documentation via the links in the top navigation panel. In addition to reading this documentation page, it can be helpful to also browse the Vega-Lite documentation.

https://altair-viz.github.io/

Iris flower data set

setosa

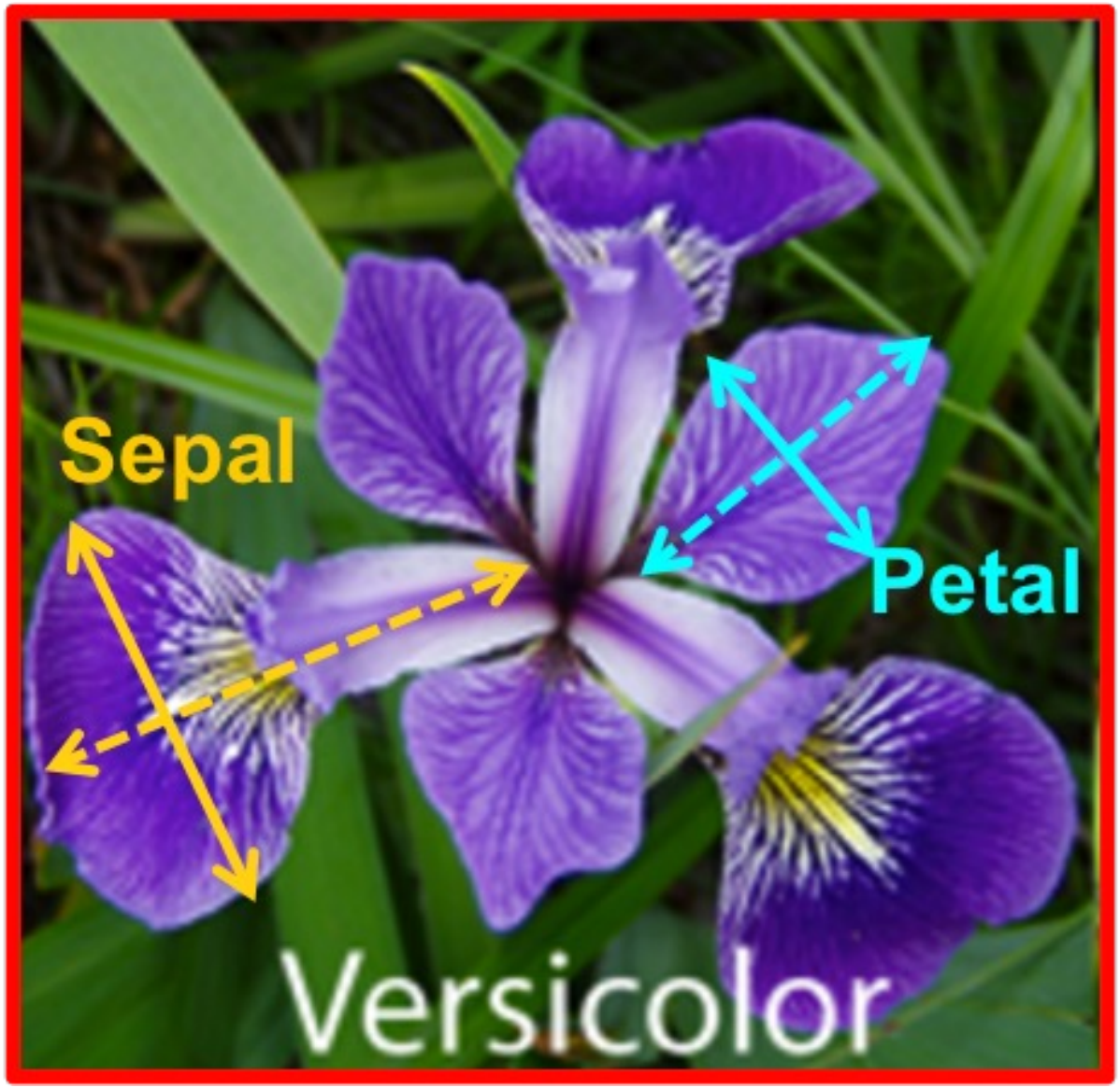
versicolor

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

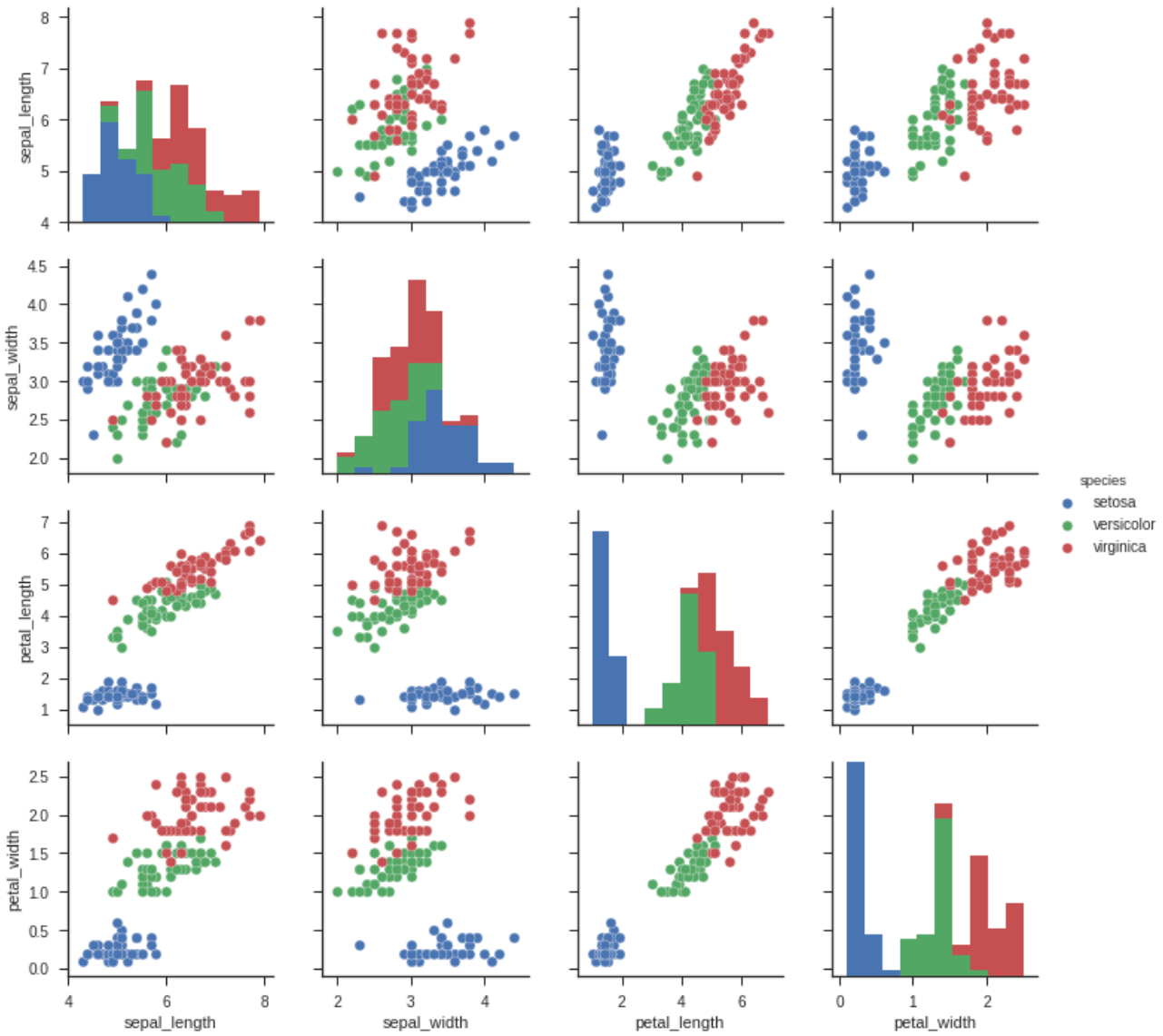
```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa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

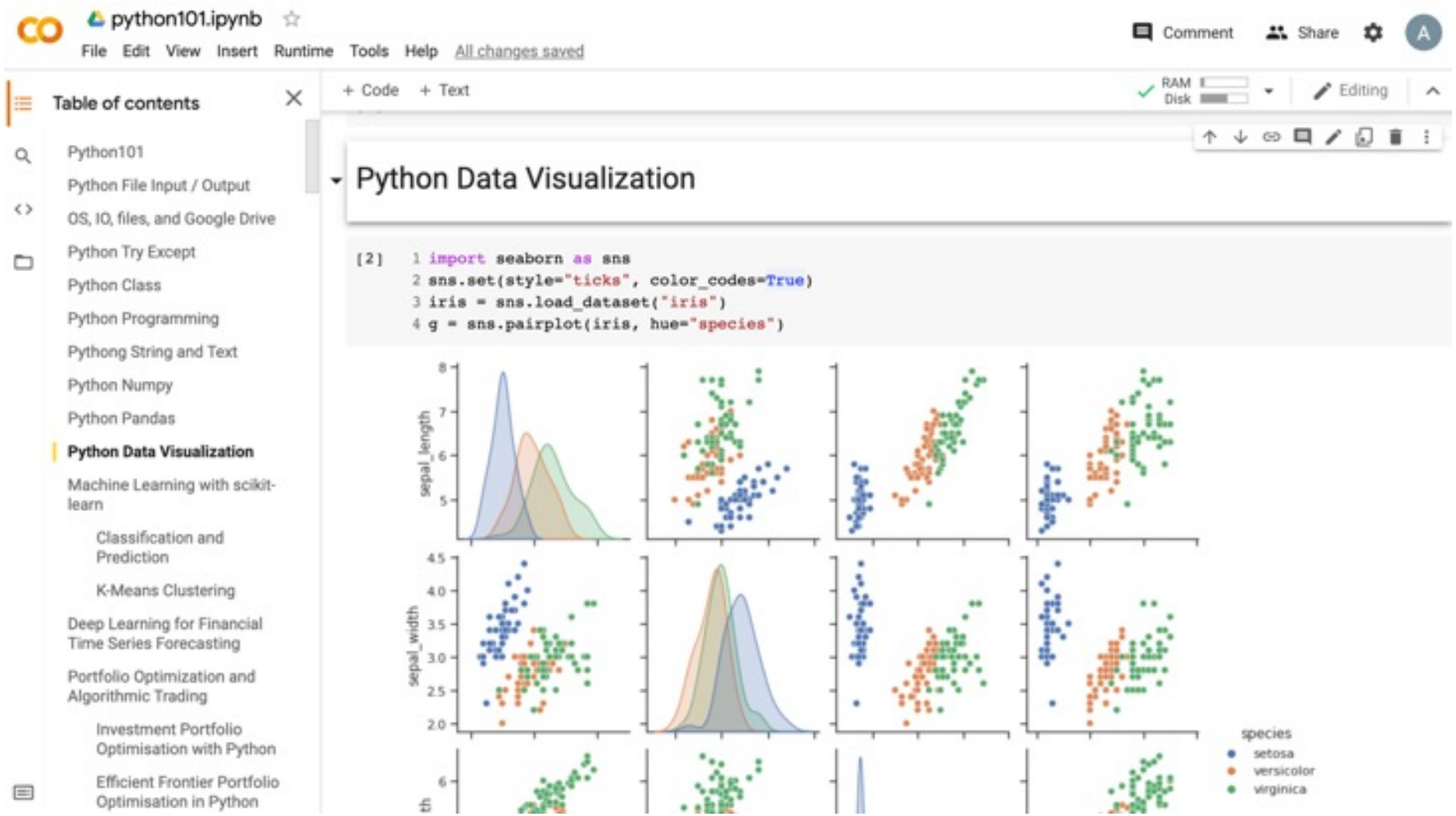
versicolor

Iris Data Visualization



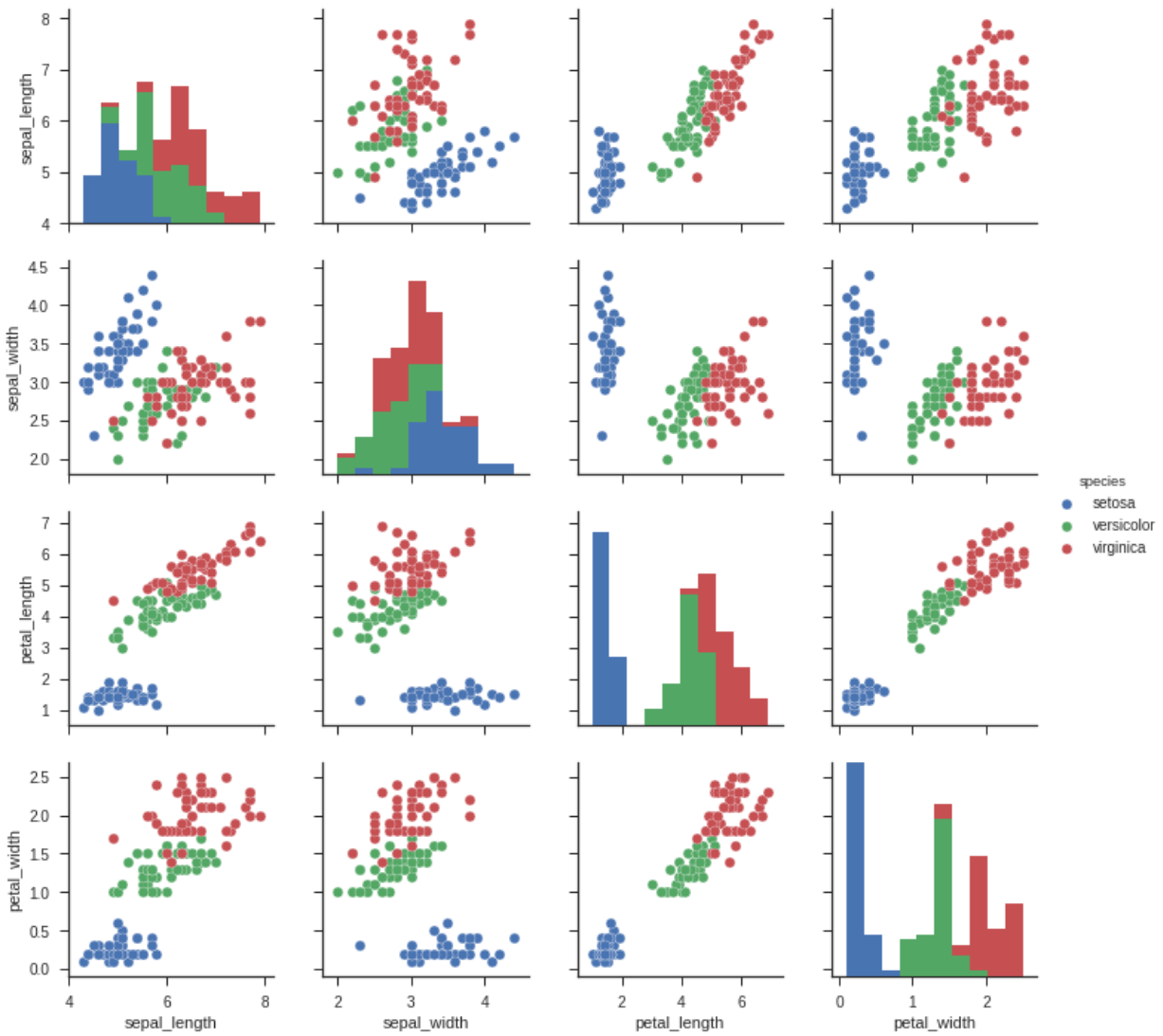
Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html

Data Visualization in Google Colab



https://tinyurl.com/aintpupython101

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



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

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

```
# 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)
```

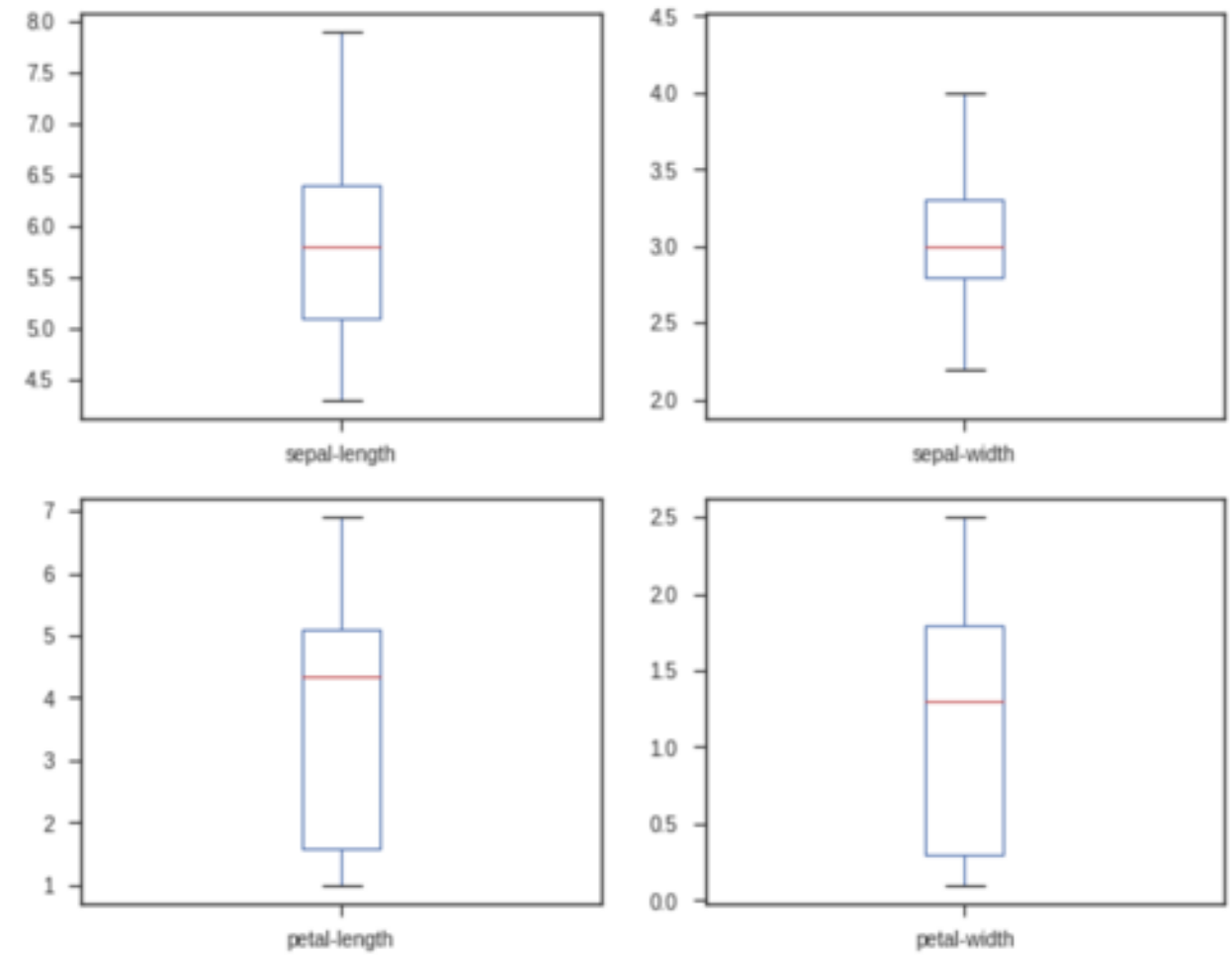
```
(150, 5)
```

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

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

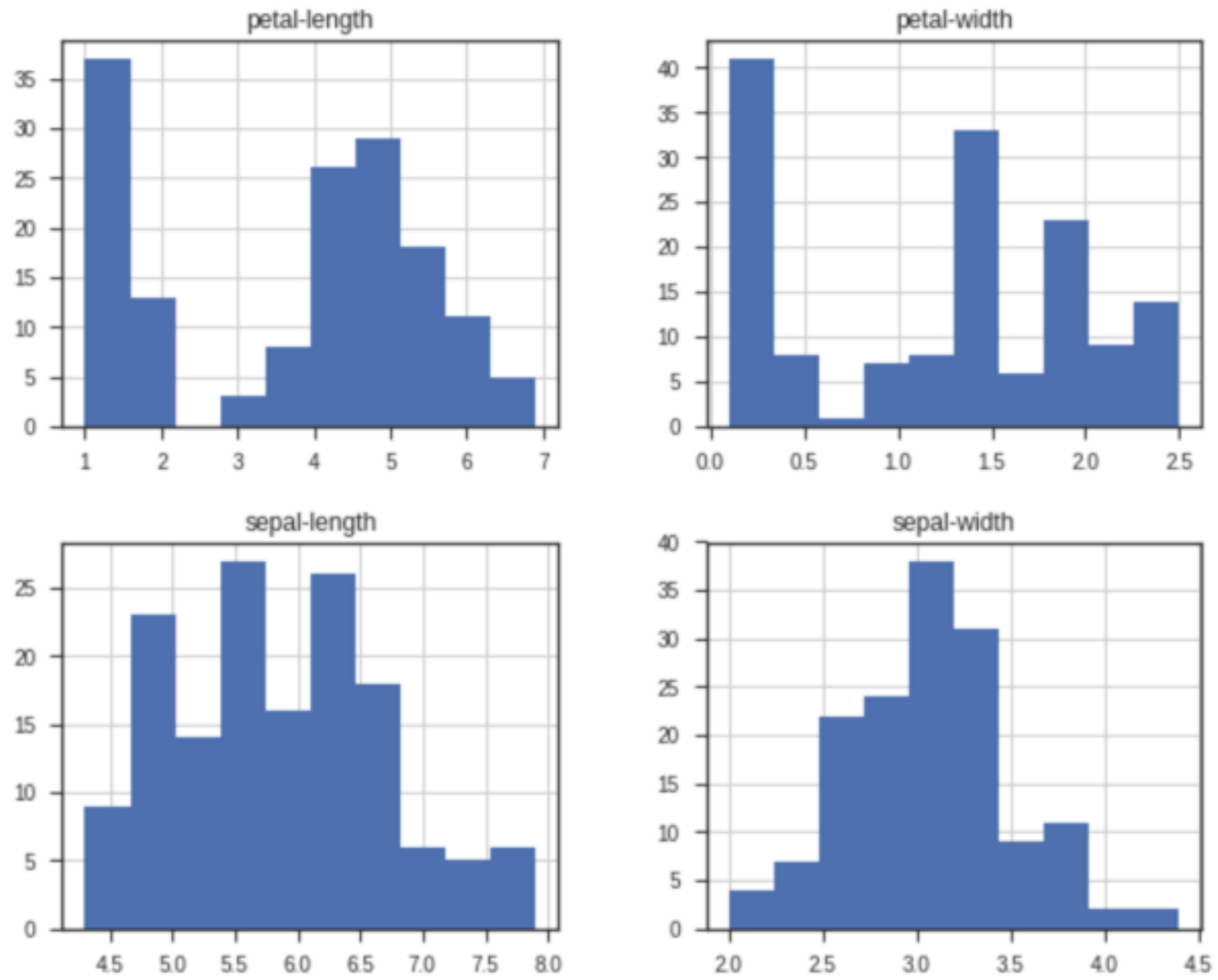
```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

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



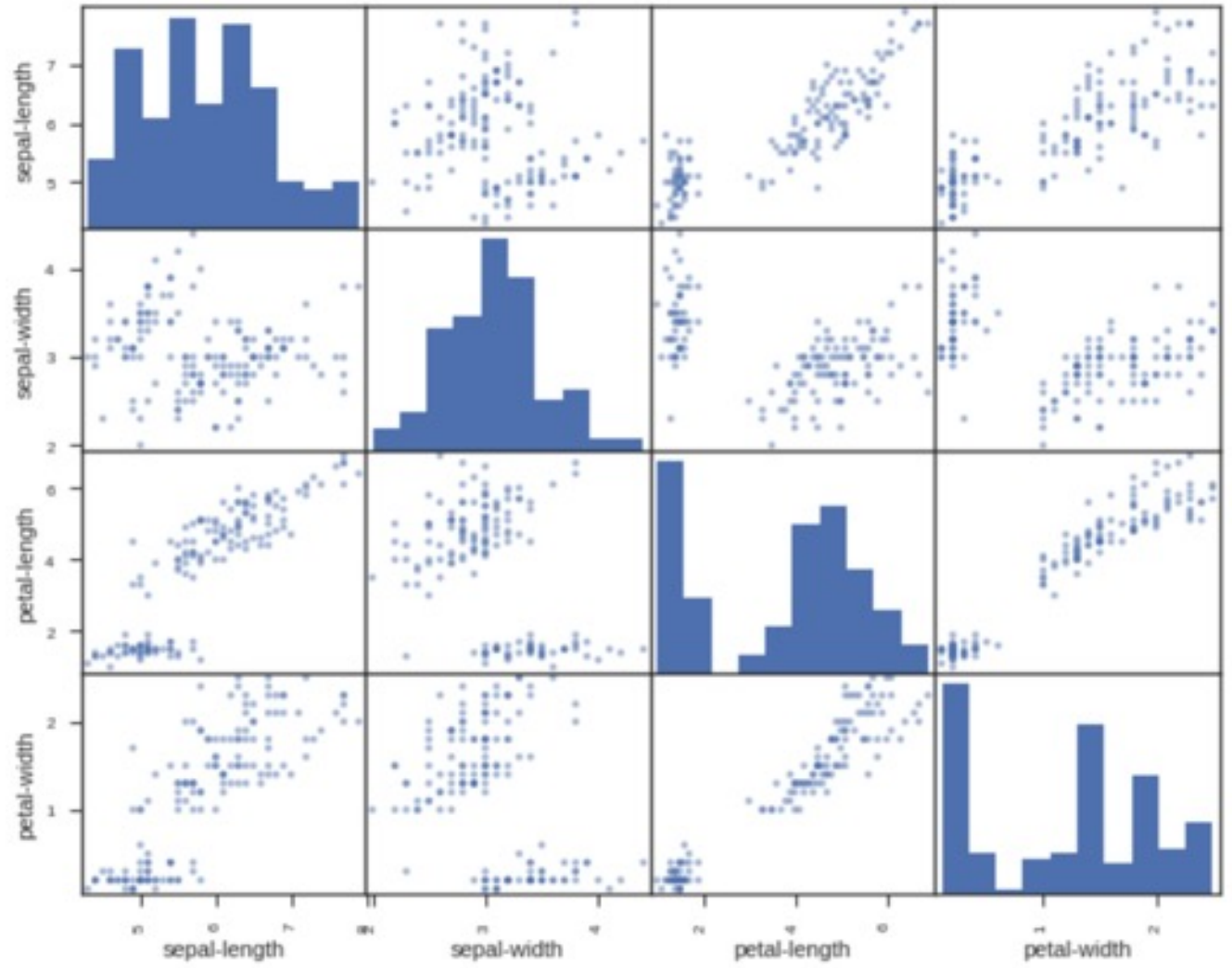
df.hist()
plt.show()

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



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

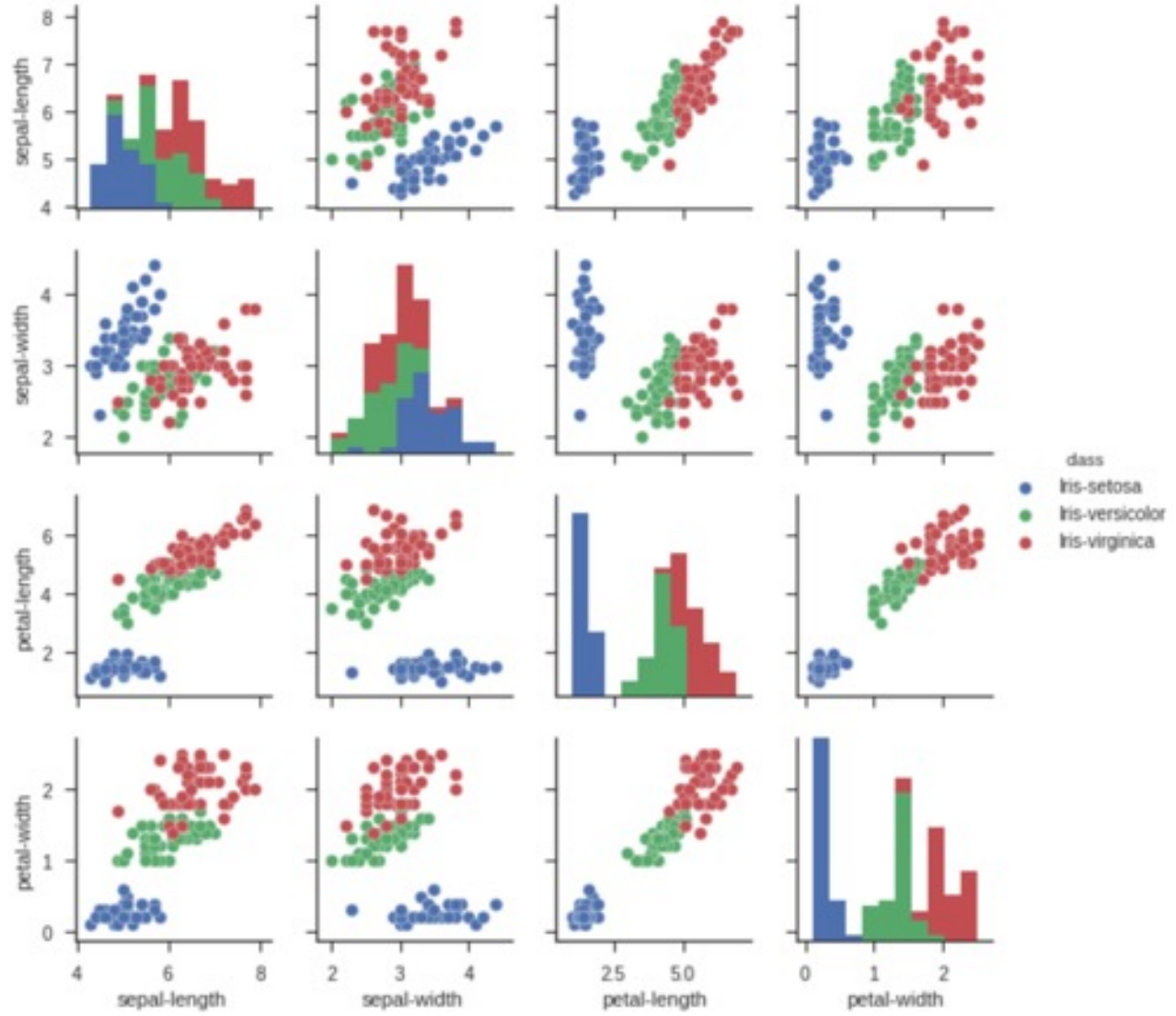
```
scatter_matrix(df)
plt.show()
```



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

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

```
<seaborn.axisgrid.PairGrid at 0x7f1d21267390>
```



Wes McKinney (2022), "Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter", 3rd Edition, O'Reilly Media.

wesm / pydata-book Public

Notifications | Fork 14.1k | Star 19k

Code | Issues 2 | Pull requests 1 | Actions | Projects | Wiki | Security | Insights

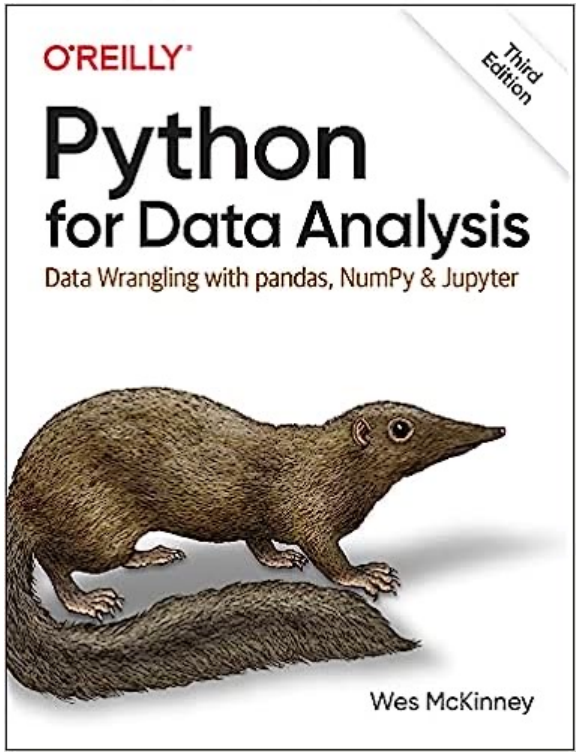
3rd-edition | 3 branches | 0 tags | Go to file | Code

wesm Upload cleaner notebook files without internal build toolchai... ... f1757b8 3 days ago 70 commits

datasets	Add fec.parquet	10 months ago
examples	Simplifying datasets	10 months ago
.gitignore	Add gitignore	8 years ago
COPYING	Update COPYING	4 months ago
README.md	Update notebooks in advance of publication	7 months ago
appa.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
appb.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch02.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch03.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch04.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch05.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch06.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago

About

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



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

Wes McKinney (2022), "Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter", 3rd Edition, O'Reilly Media.

3rd-edition ▾ pydata-book / ch04.ipynb Go to file ···

wesm Upload cleaner notebook files without internal build toolchain instru... ··· Latest commit f1757b8 3 days ago History

3 contributors

1224 lines (1224 sloc) | 21.9 KB Raw Blame

In [1]:
```
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 [2]:
```
import numpy as np

my_arr = np.arange(1_000_000)
my_list = list(range(1_000_000))
```

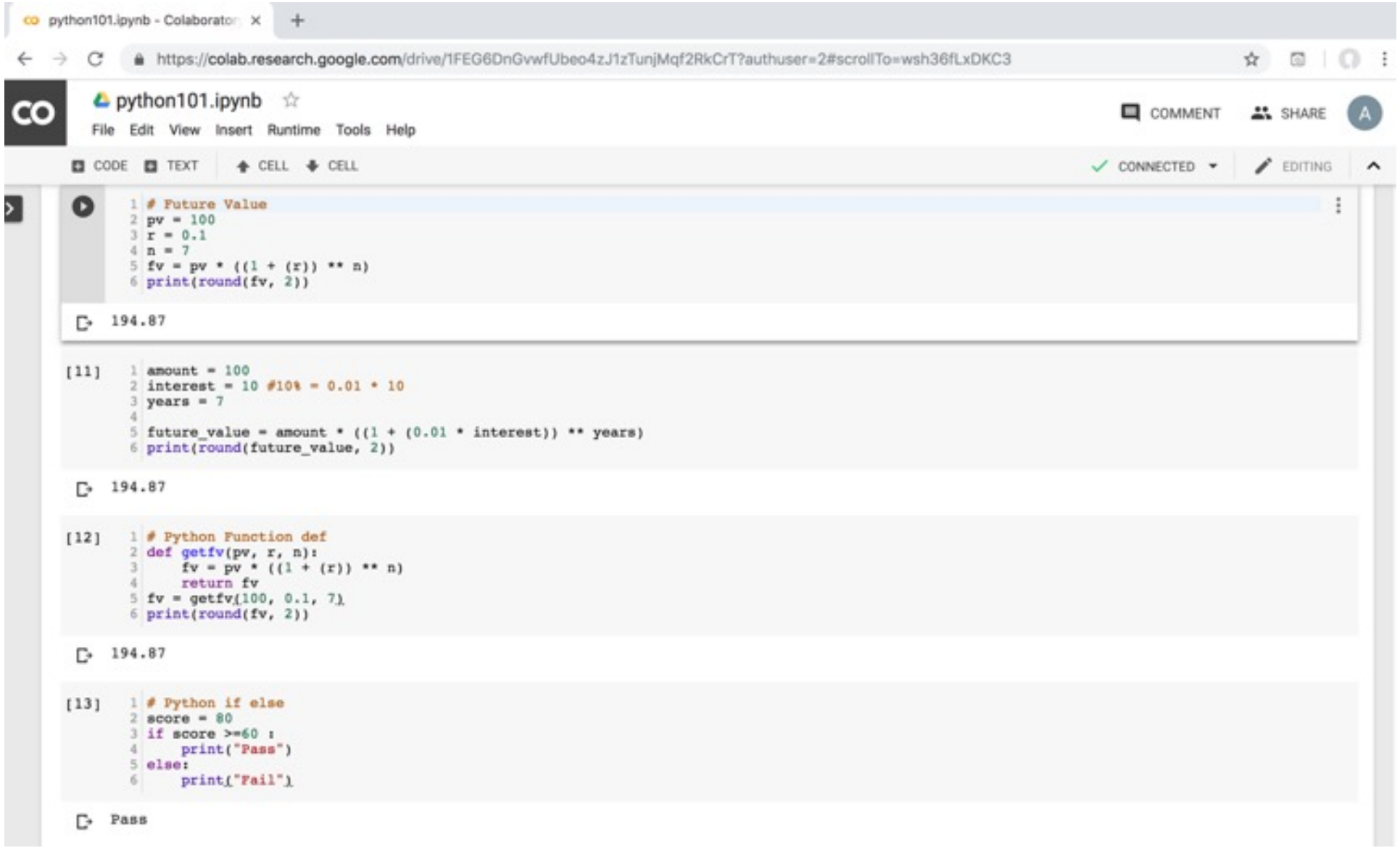
In [3]:
```
%timeit my_arr2 = my_arr * 2
%timeit my_list2 = [x * 2 for x in my_list]
```

In [4]:
```
import numpy as np
data = np.array([[1.5, -0.1, 3], [0, -3, 6.5]])
data
```

Source: https://github.com/wesm/pydata-book/blob/3rd-edition/ch04.ipynb

Python in Google Colab (Python101)

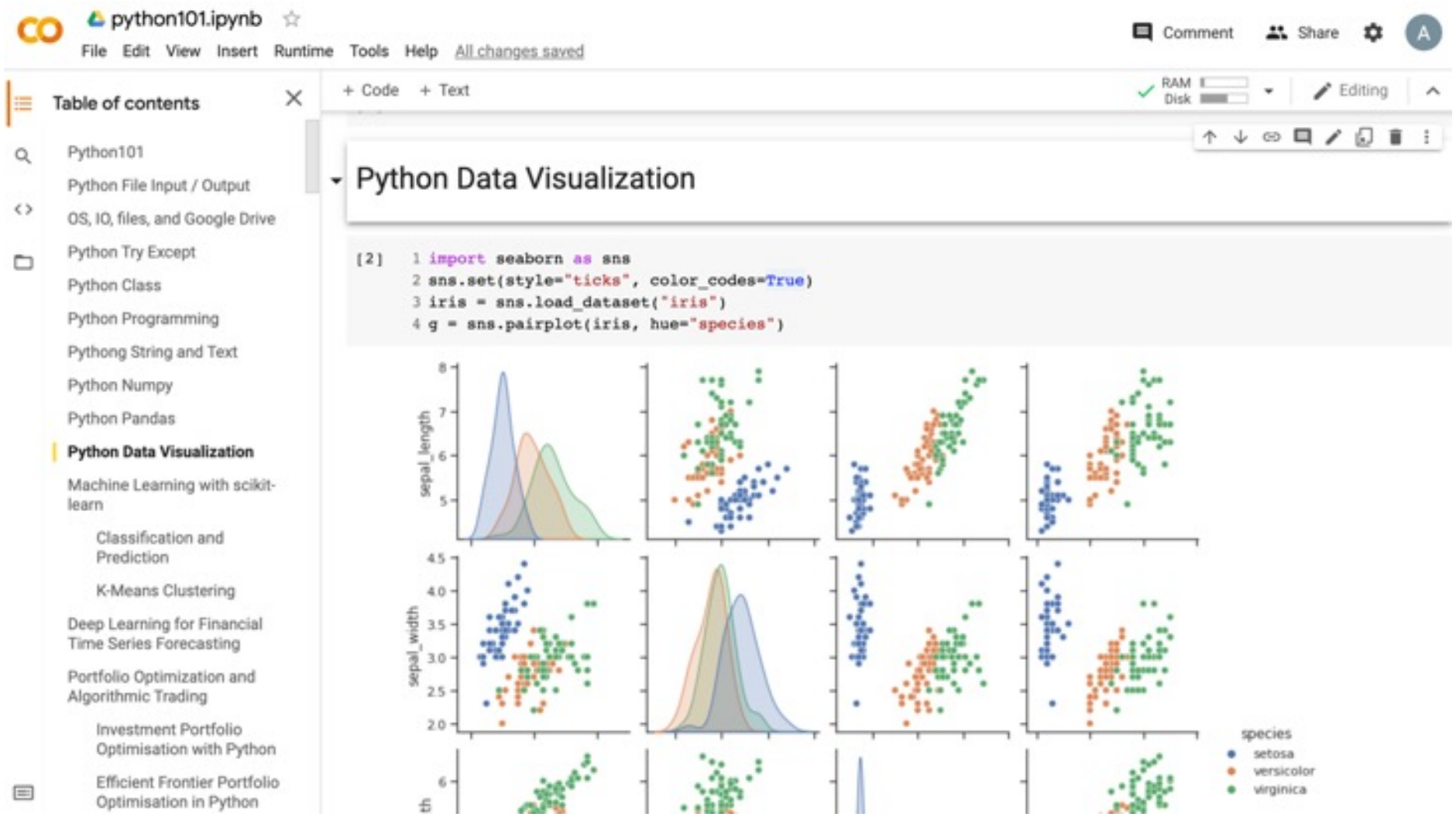
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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



https://tinyurl.com/aintpupython101

Papers with Code
State-of-the-Art (SOTA)

Computer Vision

Semantic Segmentation

185 benchmarks

3397 papers with code

Image Classification

390 benchmarks

2778 papers with code

Object Detection

269 benchmarks

2559 papers with code

Contrastive Learning

2 benchmarks

1119 papers with code

Image Generation

208 benchmarks

1097 papers with code

See all 1415 tasks

Natural Language Processing

Language Modelling

458 benchmarks

2248 papers with code

Question Answering

181 benchmarks

1818 papers with code

Machine Translation

78 benchmarks

1721 papers with code

Sentiment Analysis

87 benchmarks

1040 papers with code

Text Generation

242 benchmarks

931 papers with code

See all 664 tasks

https://paperswithcode.com/sota

Summary

- **Foundations of Big Data Analysis in Python**
  - **Python Ecosystem for Data Science**
  - **Python**
    - Programming language
  - **Numpy**
    - Scientific computing
  - **Pandas**
    - Data structures and data analysis tools

References

- Wes McKinney (2022), "Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter", 3rd Edition, O'Reilly Media.
- Steven D'Ascoli (2022), Artificial Intelligence and Deep Learning with Python: Every Line of Code Explained For Readers New to AI and New to Python, Independently published.
- Aurélien Géron (2019), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition, O'Reilly Media.
- 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/
- Min-Yuh Day (2023), Python 101, https://tinyurl.com/aintpupython101