Python for Accounting Applications

Obtaining Data From the Web with Python
Statistical Analysis with Python

1131PAA07
ACC2, NTPU (U2004) (Fall 2024)
Wed 6, 7, 8, (14:10-17:00) (9:10-12:00) (B3F10)

Min-Yuh Day, Ph.D,
Professor

Institute of Information Management, National Taipei University

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

2024-11-20

Syllabus

Week	Date	Subject/Topics
1	2024/09/11	Introduction to Python for Accounting Applications
2	2024/09/18	Python Programming and Data Science
3	2024/09/25	Foundations of Python Programming
4	2024/10/02	Data Structures
5	2024/10/09	Control Logic and Loops
6	2024/10/16	Functions and Modules; Files and Exception Handling
7	2024/10/23	Data Analytics and Visualization with Python
8	2024/10/30	~~Midterm Project Report~~ (Self-Learning)

Syllabus

Week	Date	Subject/Topics
9	2024/11/06	Self-Learning
10	2024/11/13	Midterm Project Report
11	2024/11/20	Obtaining Data From the Web with Python; Statistical Analysis with Python
12	2024/11/27	Machine Learning with Python
13	2024/12/04	Text Analytics with Generative AI and Python
14	2024/12/11	Applications of Accounting Data Analytics with Python
15	2024/12/18	Applications of ESG Data Analytics with Python
16	2024/12/25	Final Project Report

Obtaining Data From the Web with Python
Statistical Analysis with Python

Outline

- Obtaining Data From the Web with Python
  - Requests
  - BeautifulSoup
  - Pandas
- Statistical Analysis with Python
  - Descriptive Statistics
  - Inferential Statistics

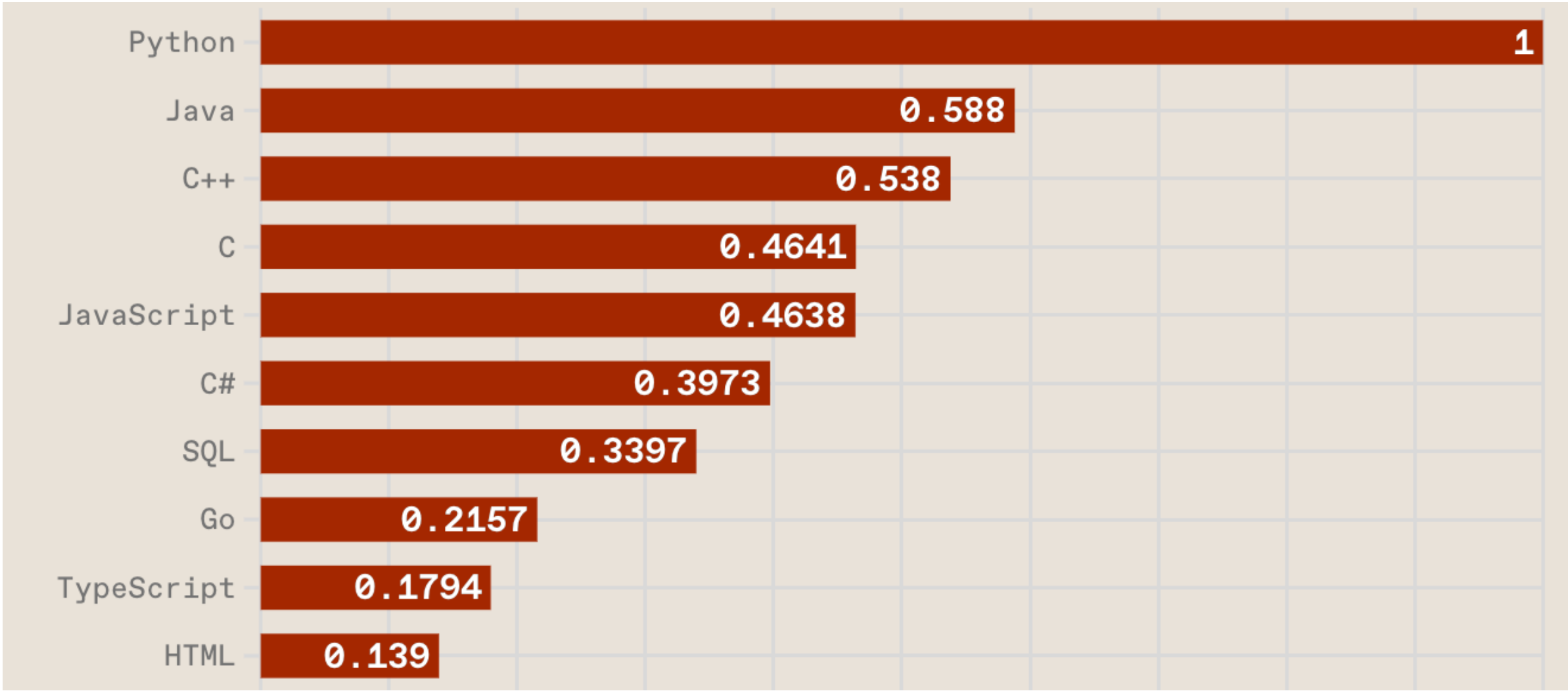
Statistical Analysis with Python

- **Descriptive Statistics**
  - `df.describe()`
  - `df.mean(), df.std() df.median() df.max() df.min()`
  - `df.skew(), df.kurtosis()`
- **Inferential Statistics**
  - `import scipy.stats as stats`
  - `t_statistic, p_value = stats.ttest_ind(ga, gb)`

Python

Programming

Top Programming Languages



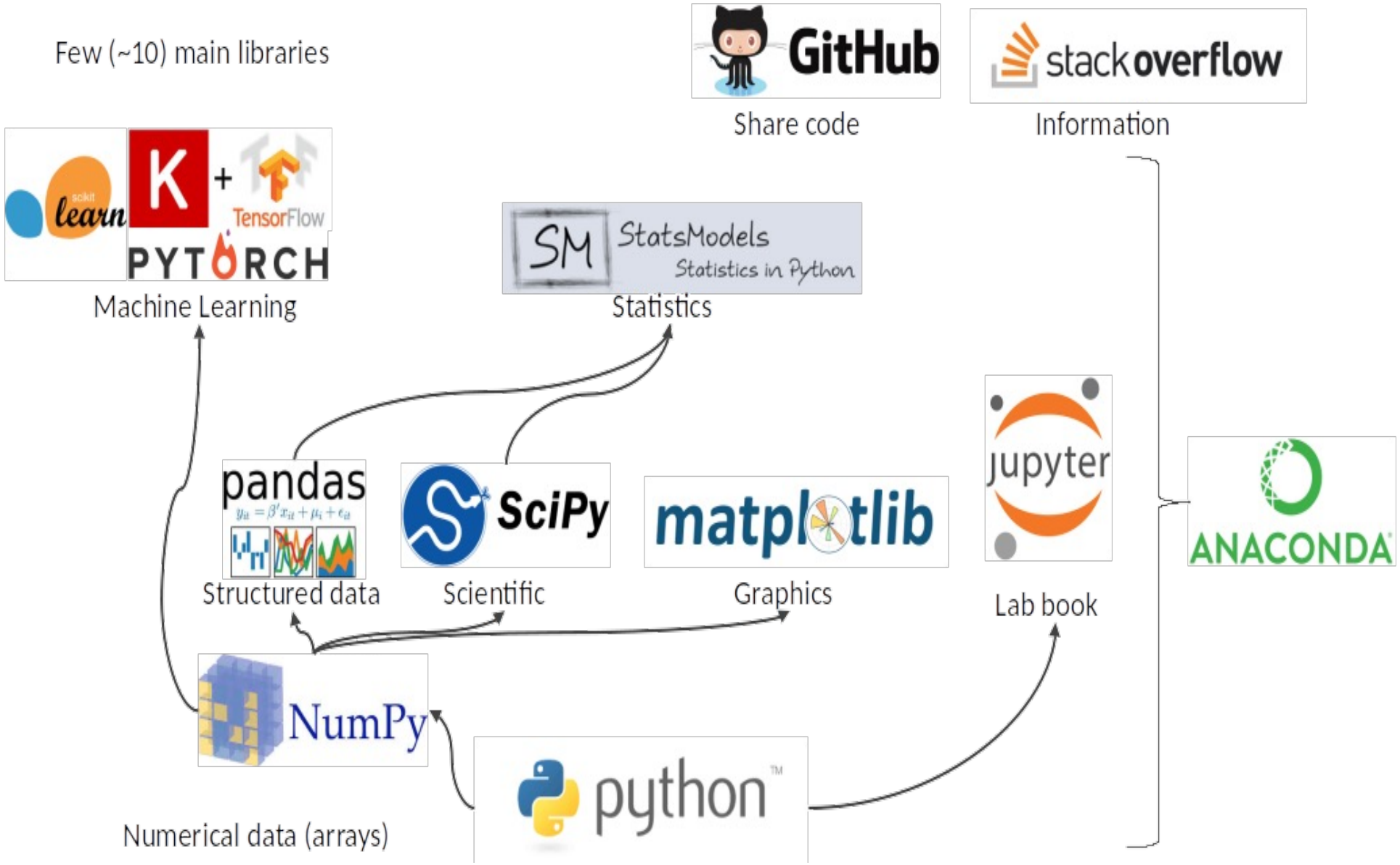
https://spectrum.ieee.org/the-top-programming-languages-2023

python™

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

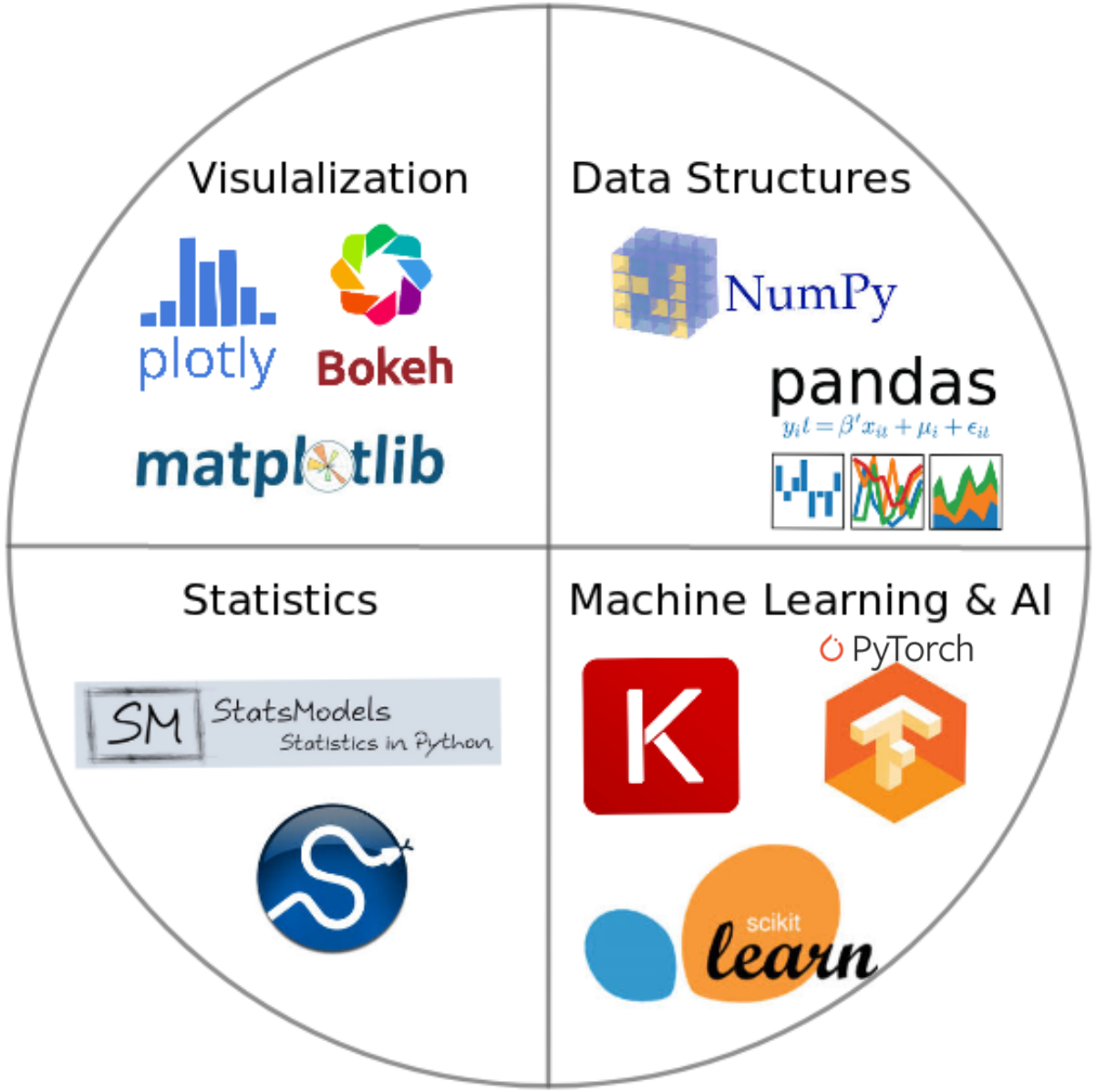
Source: https://www.python.org/doc/essays/blurb/

Python Ecosystem for Data Science



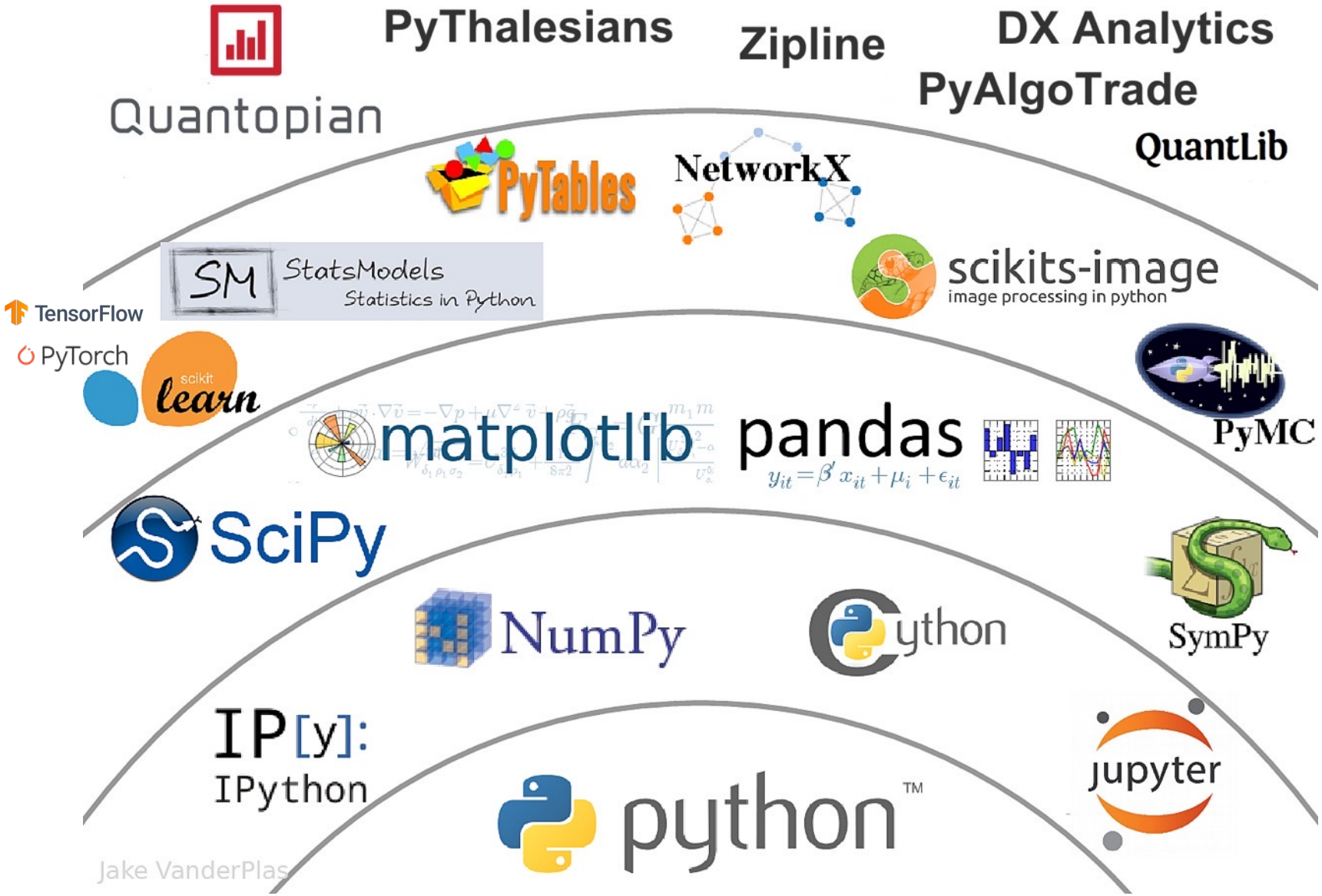
Source: https://medium.com/pyfinance/why-python-is-best-choice-for-financial-data-modeling-in-2019-c0d0d1858c45

Python Ecosystem for Data Science



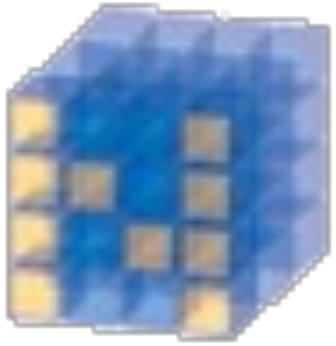
Source:https://duchesnay.github.io/pystatsml/introduction/python_ecosystem.html

The Quant Finance PyData Stack



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

Numpy



NumPy

Base

N-dimensional array

package

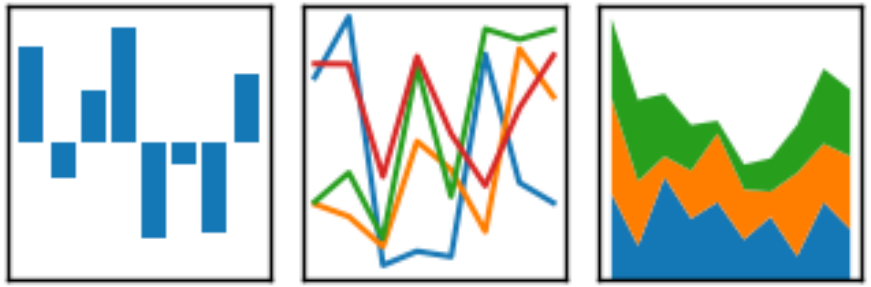
Python matplotlib

Source: https://matplotlib.org/

Python Pandas

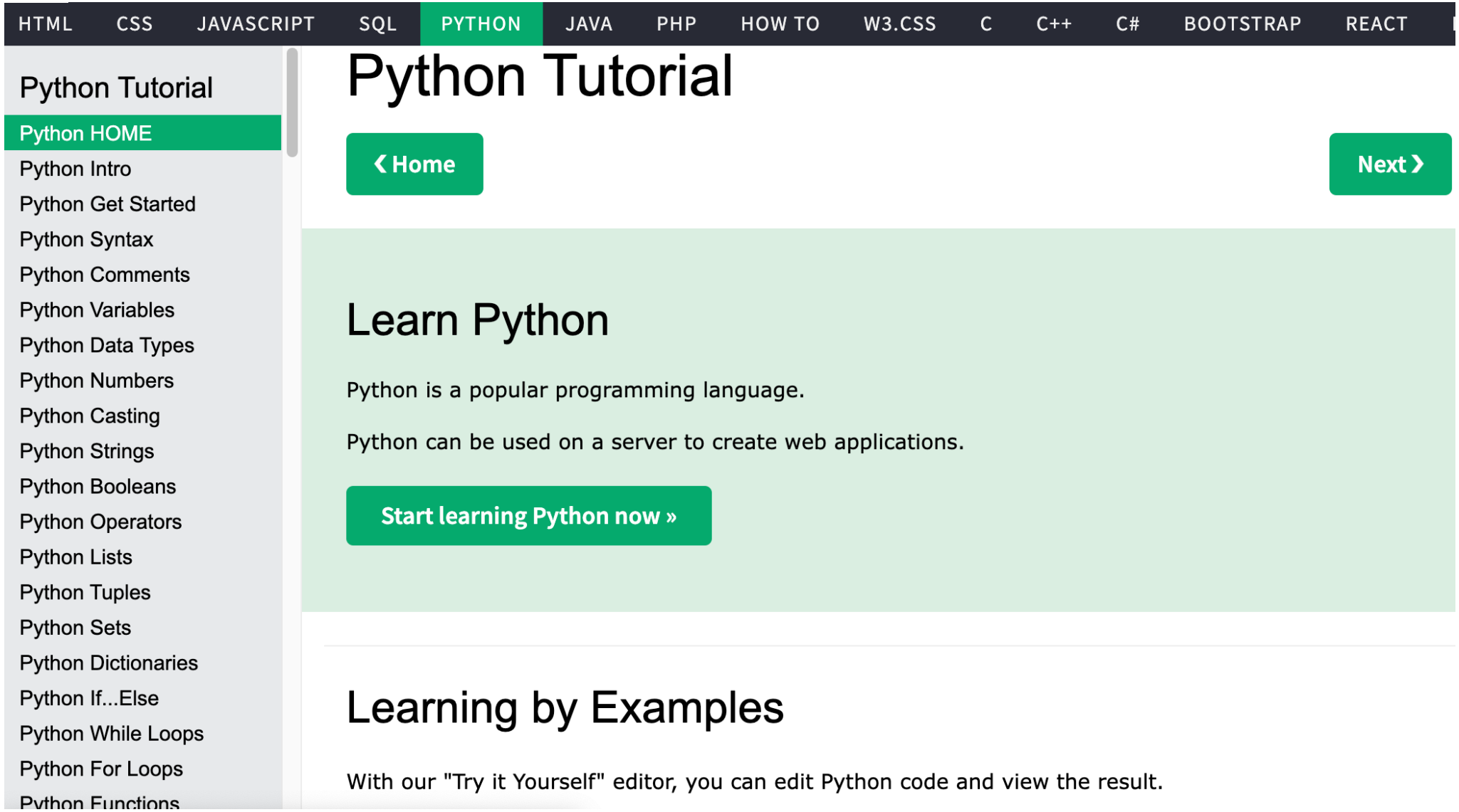
pandas

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



http://pandas.pydata.org/

W3Schools Python



https://www.w3schools.com/python/

W3Schools Python Numpy

HTML CSS JAVASCRIPT SQL PYTHON JAVA PHP HOW TO W3.CSS C C++ C# BOOTSTRAP REACT

Python Modules

- NumPy Tutorial
- Pandas Tutorial
- SciPy Tutorial
- Django Tutorial

NumPy Tutorial

- NumPy HOME
- NumPy Intro
- NumPy Getting Started
- NumPy Creating Arrays
- NumPy Array Indexing
- NumPy Array Slicing
- NumPy Data Types
- NumPy Copy vs View
- NumPy Array Shape
- NumPy Array Reshape
- NumPy Array Iterating
- NumPy Array Join
- NumPy Array Split
- NumPy Array Search
- NumPy Array Sort
- NumPy Array Filter

NumPy Random

- Random Intro
- Data Distribution

NumPy Tutorial

❮ Home Next ❯

NumPy is a Python library.

NumPy is used for working with arrays.

NumPy is short for "Numerical Python".

Learning by Reading

We have created 43 tutorial pages for you to learn more about NumPy.

Starting with a basic introduction and ends up with creating and plotting random data sets, and working with NumPy functions:

Basic Random ufunc

https://www.w3schools.com/python/numpy/default.asp

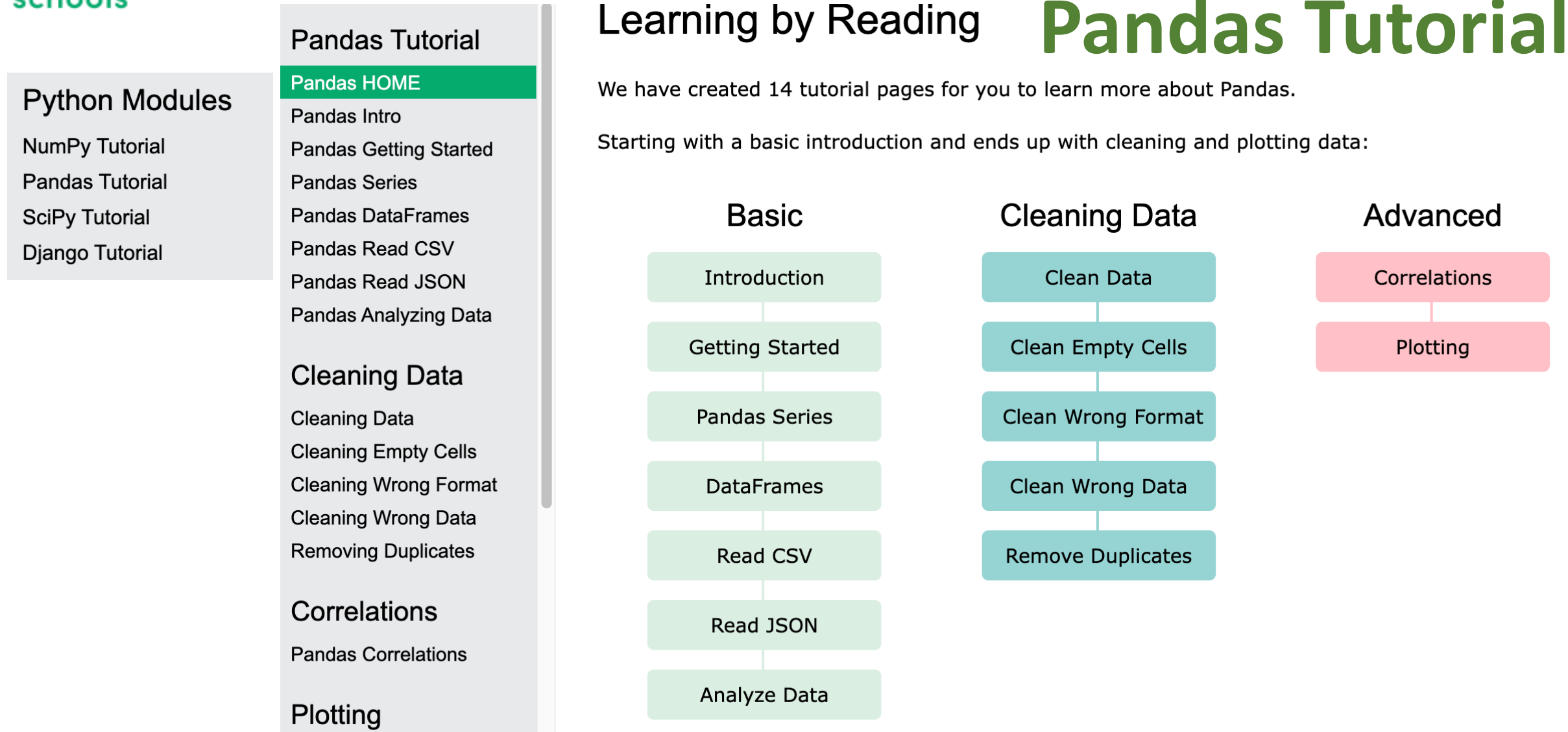
W3Schools Python Pandas

Learning by Reading

Pandas Tutorial

We have created 14 tutorial pages for you to learn more about Pandas.

Starting with a basic introduction and ends up with cleaning and plotting data:



https://www.w3schools.com/python/pandas/default.asp

W3Schools Python Requests Module

HTML CSS JAVASCRIPT SQL PYTHON JAVA PHP HOW TO W3.CSS C C++ C# BOOTSTRAP REACT

Python Reference

- Python Overview
- Python Built-in Functions
- Python String Methods
- Python List Methods
- Python Dictionary Methods
- Python Tuple Methods
- Python Set Methods
- Python File Methods
- Python Keywords
- Python Exceptions
- Python Glossary

Module Reference

- Random Module
- Requests Module
- Statistics Module
- Math Module
- cMath Module

Python Requests Module

❮ Previous Next ❯

Example

Get your own Python Server

Make a request to a web page, and print the response text:

```
import requests

x = requests.get('https://w3schools.com/python/demopage.htm')

print(x.text)
```

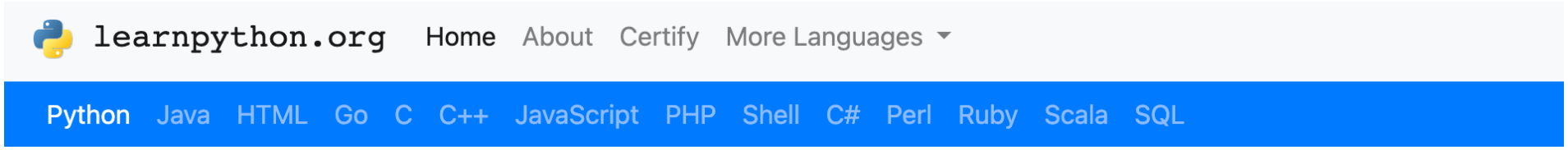
Run Example »

https://www.w3schools.com/python/

W3Schools Python: Try Python

https://www.w3schools.com/python/trypython.asp?filename=demo_default

LearnPython.org



Get started learning Python with DataCamp's free Intro to Python tutorial. Learn Data Science by completing interactive coding challenges and watching videos by expert instructors. Start Now!

Ready to take the test? Head onto LearnX and get your Python Certification!

This site is generously supported by DataCamp. DataCamp offers online interactive Python Tutorials for Data Science. Join **11 millions** other learners and get started learning Python for data science today!

Good news! You can save 25% off your Datacamp annual subscription with the code LEARNPYTHON23ALE25 - Click here to redeem your discount!

Welcome

Welcome to the LearnPython.org interactive Python tutorial.

Whether you are an experienced programmer or not, this website is intended for everyone who wishes to learn the Python programming language.

You are welcome to join our group on Facebook for questions, discussions and updates.

After you complete the tutorials, you can get certified at LearnX and add your certification to your LinkedIn profile.

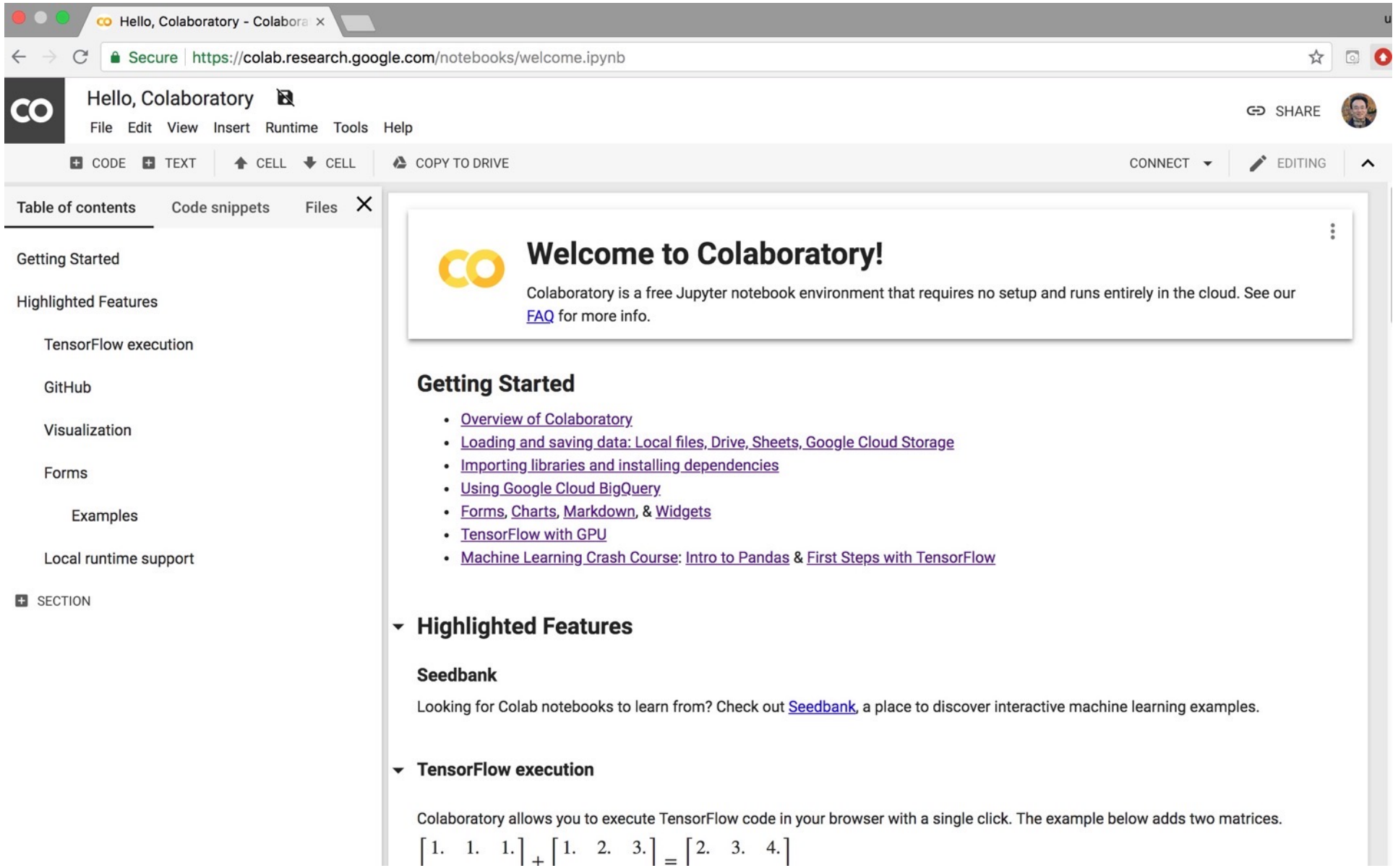
Just click on the chapter you wish to begin from, and follow the instructions. Good luck!

https://www.learnpython.org/

Google's Python Class

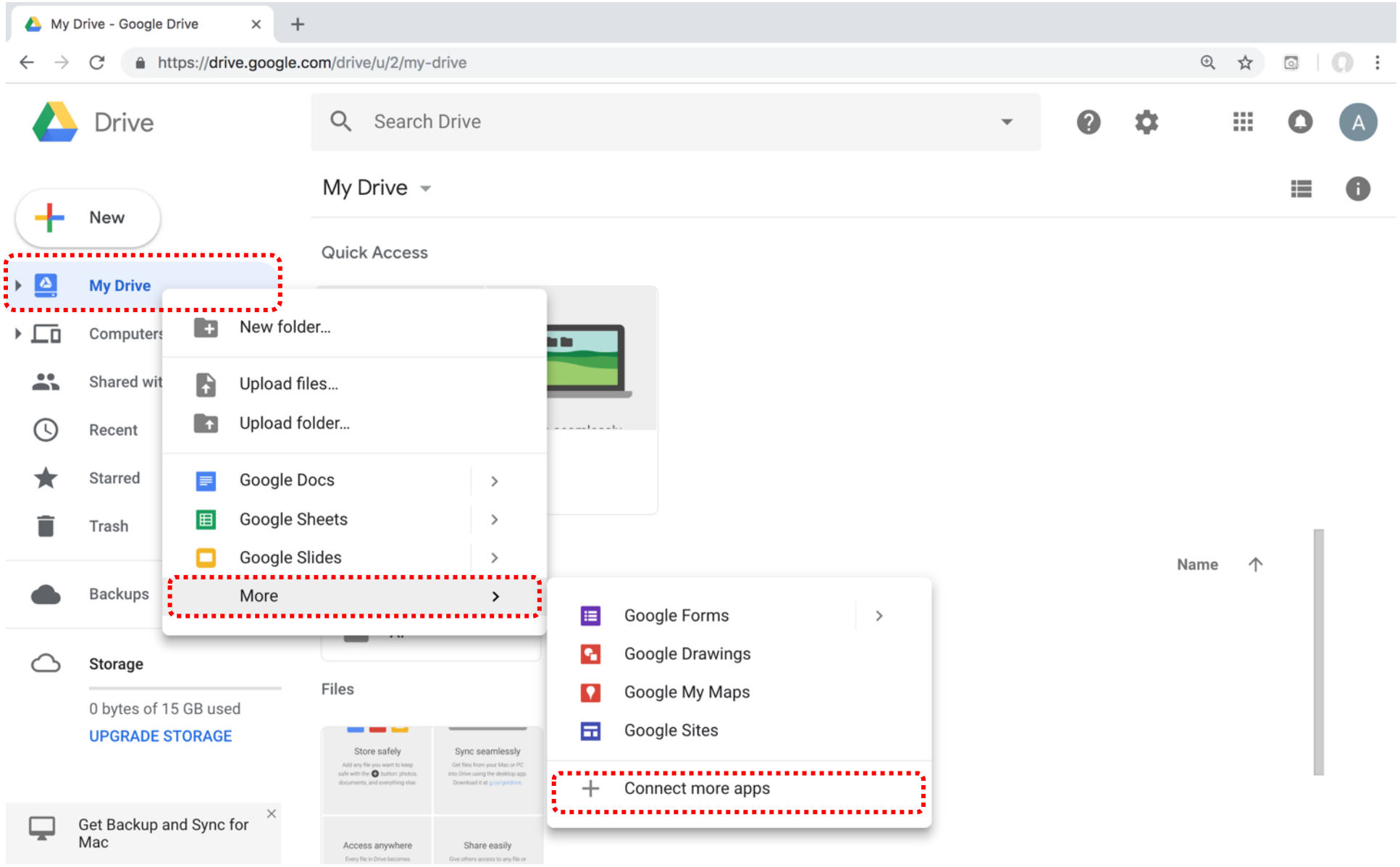
https://developers.google.com/edu/python

Google Colab

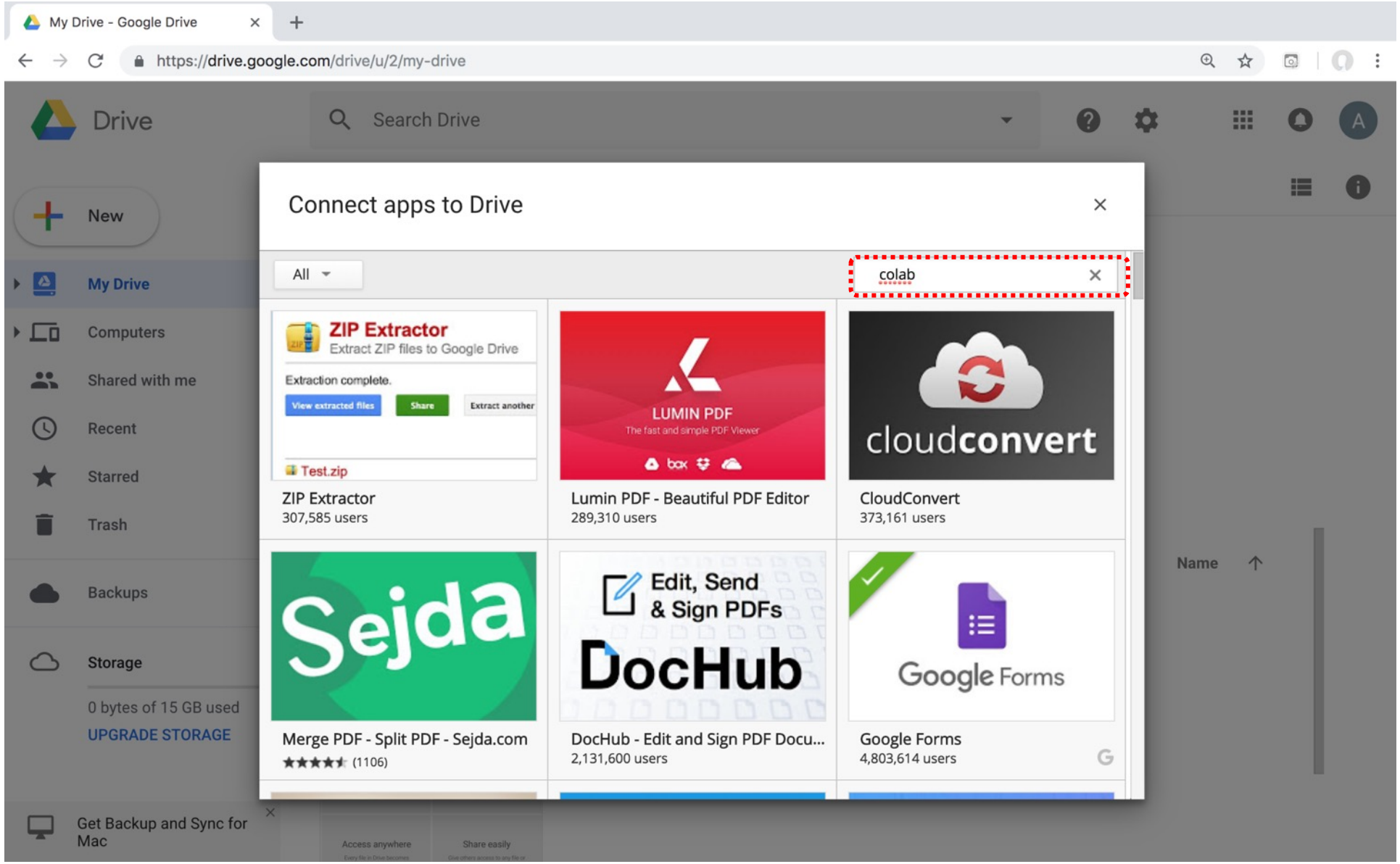


https://colab.research.google.com/notebooks/welcome.ipynb

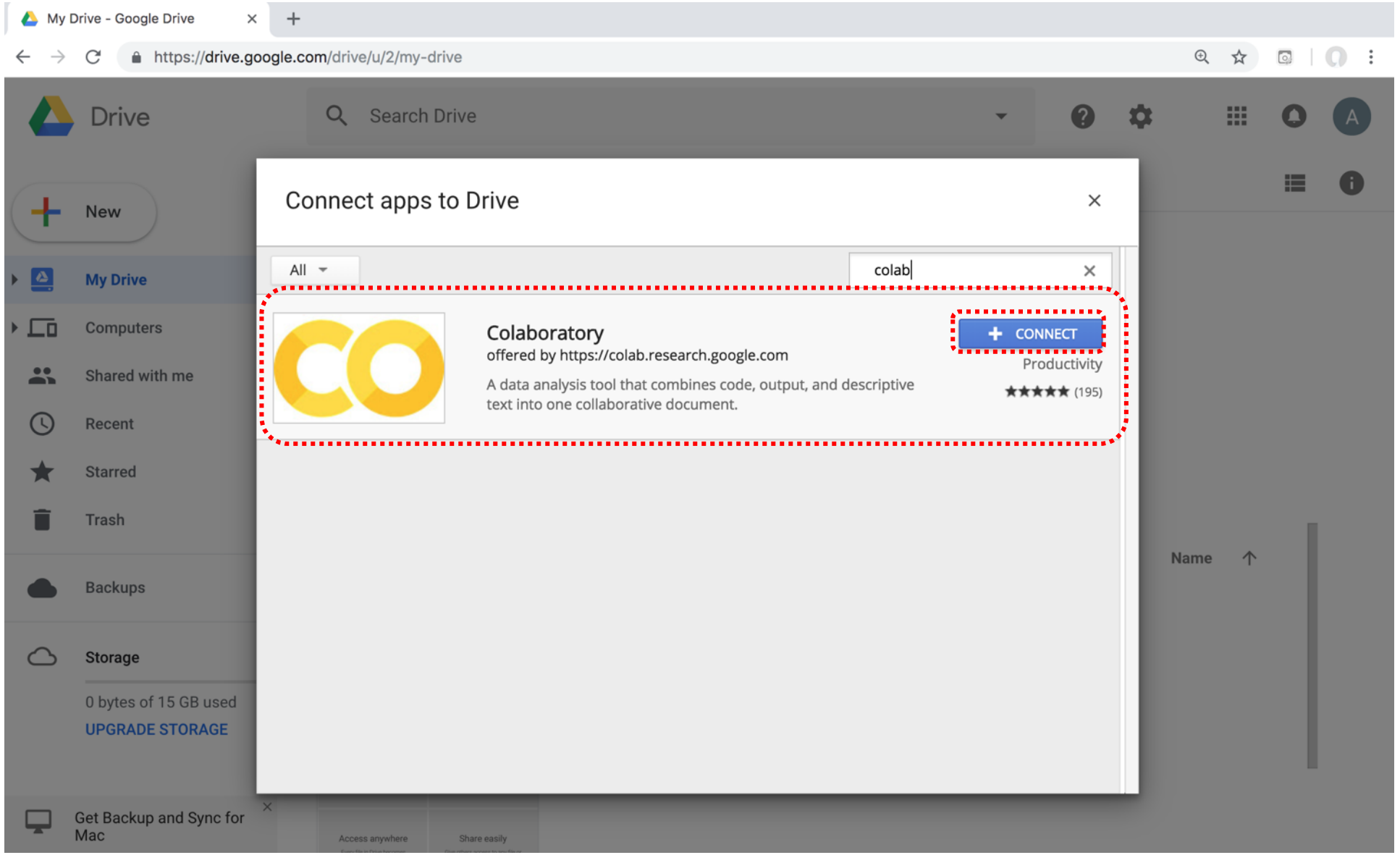
Connect Google Colab in Google Drive



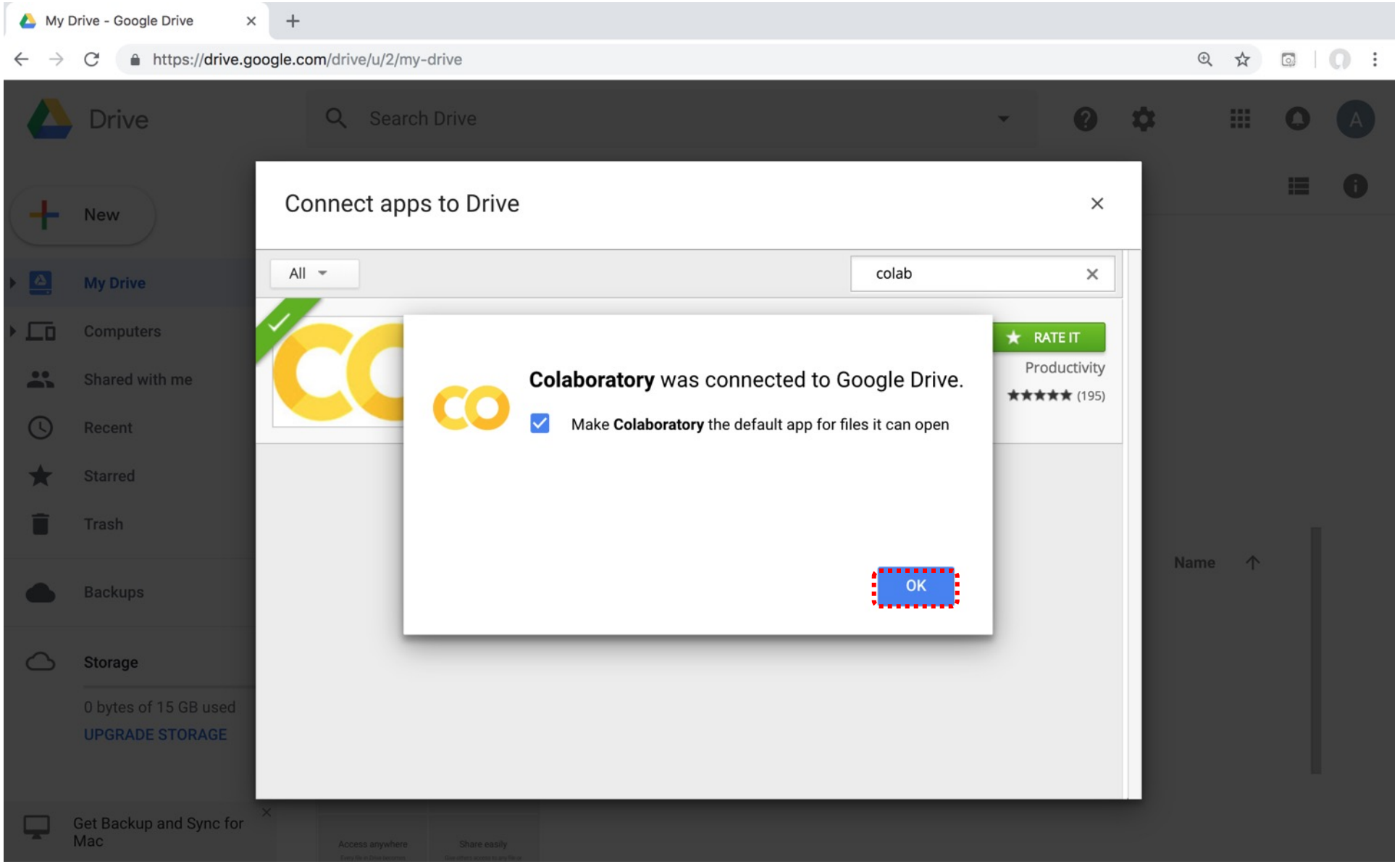
Google Colab



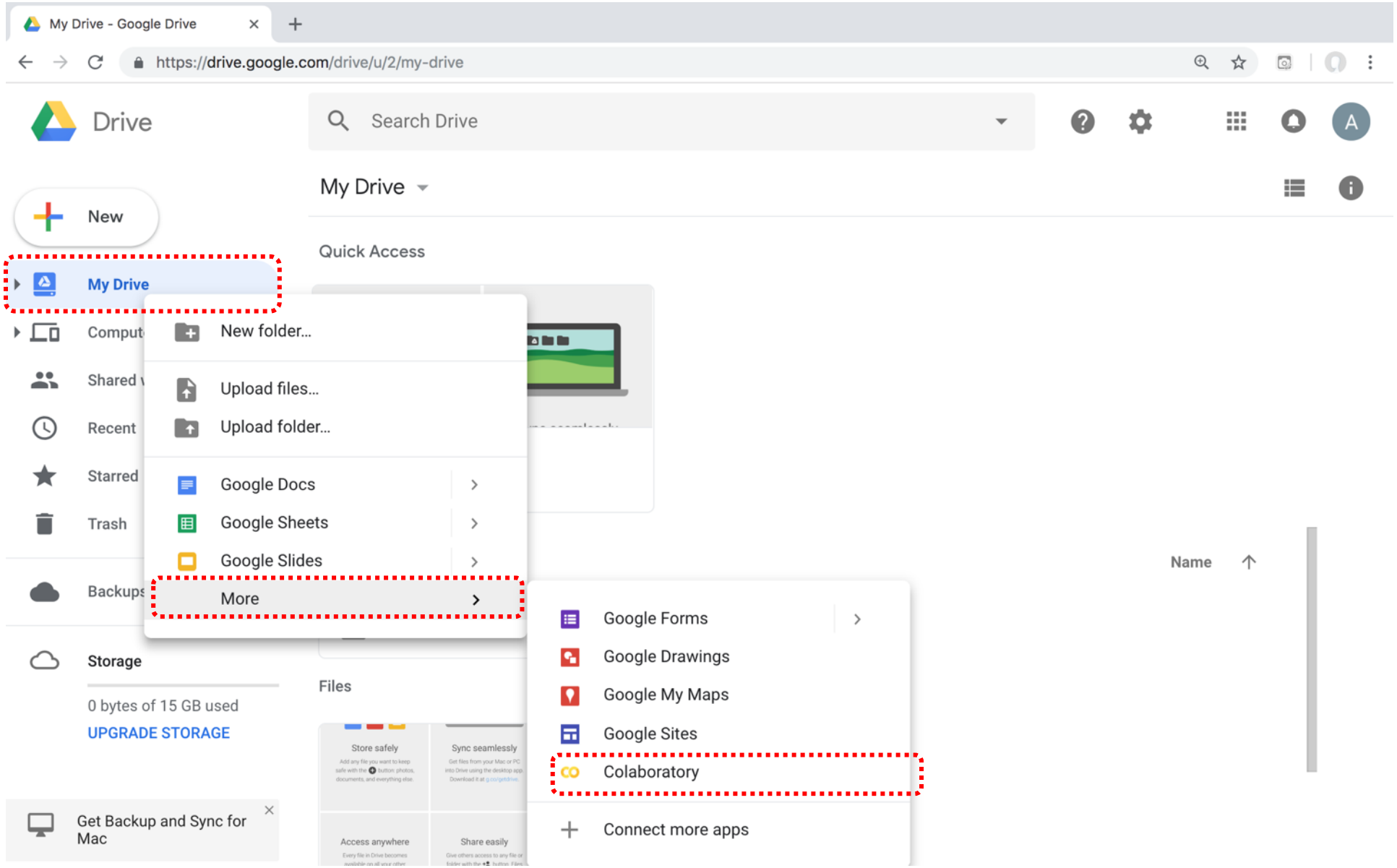
Google Colab



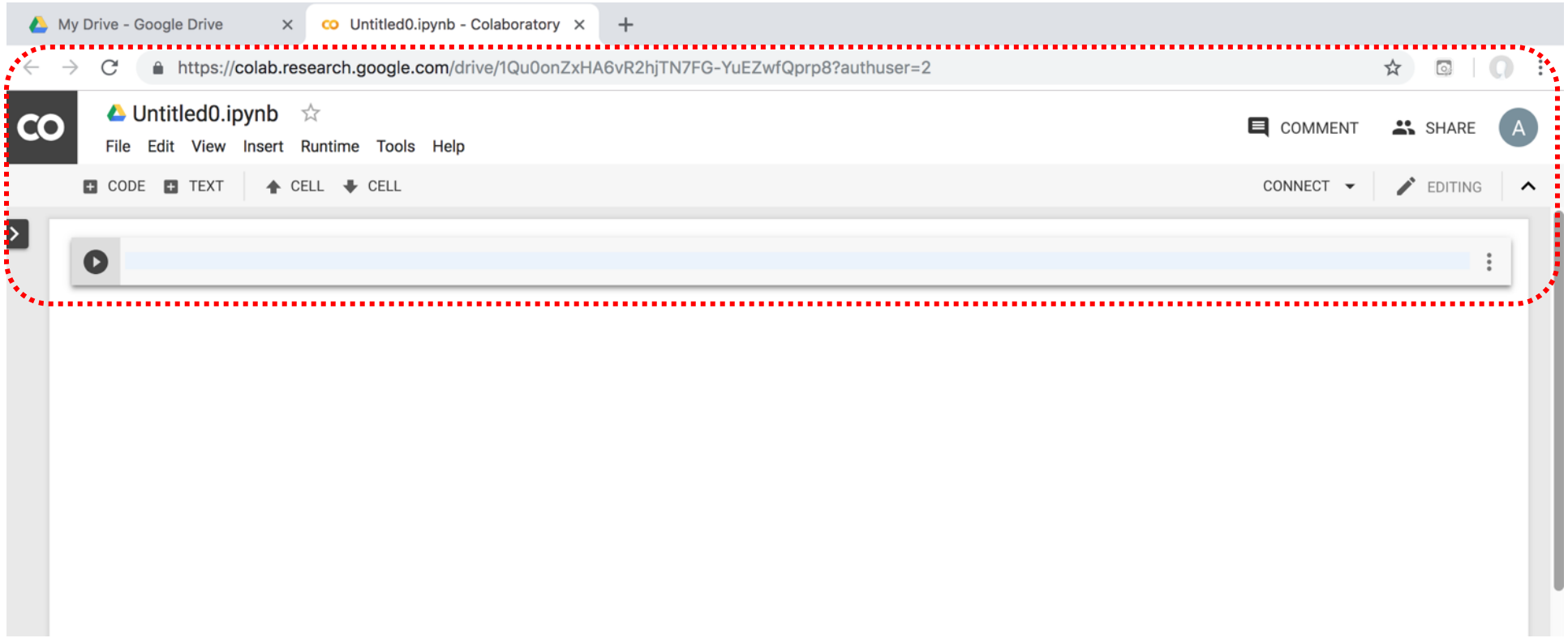
Connect Colaboratory to Google Drive



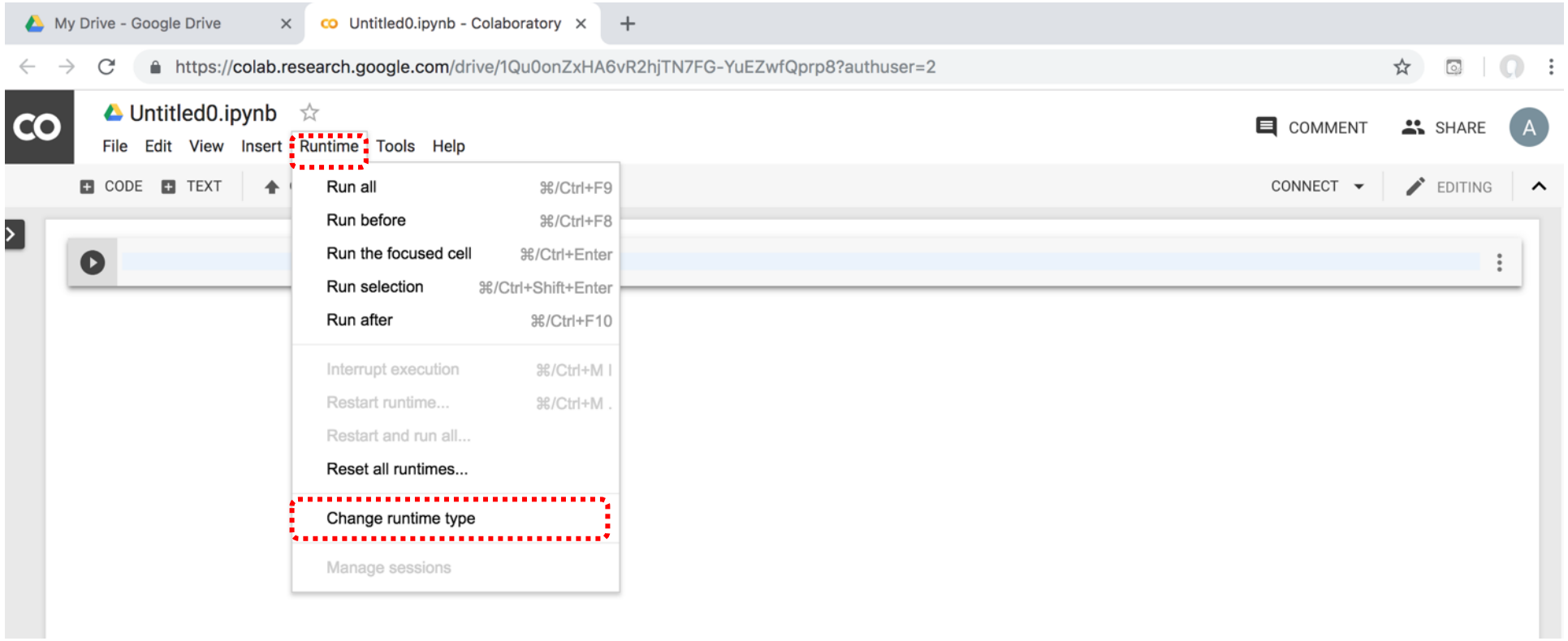
Google Colab



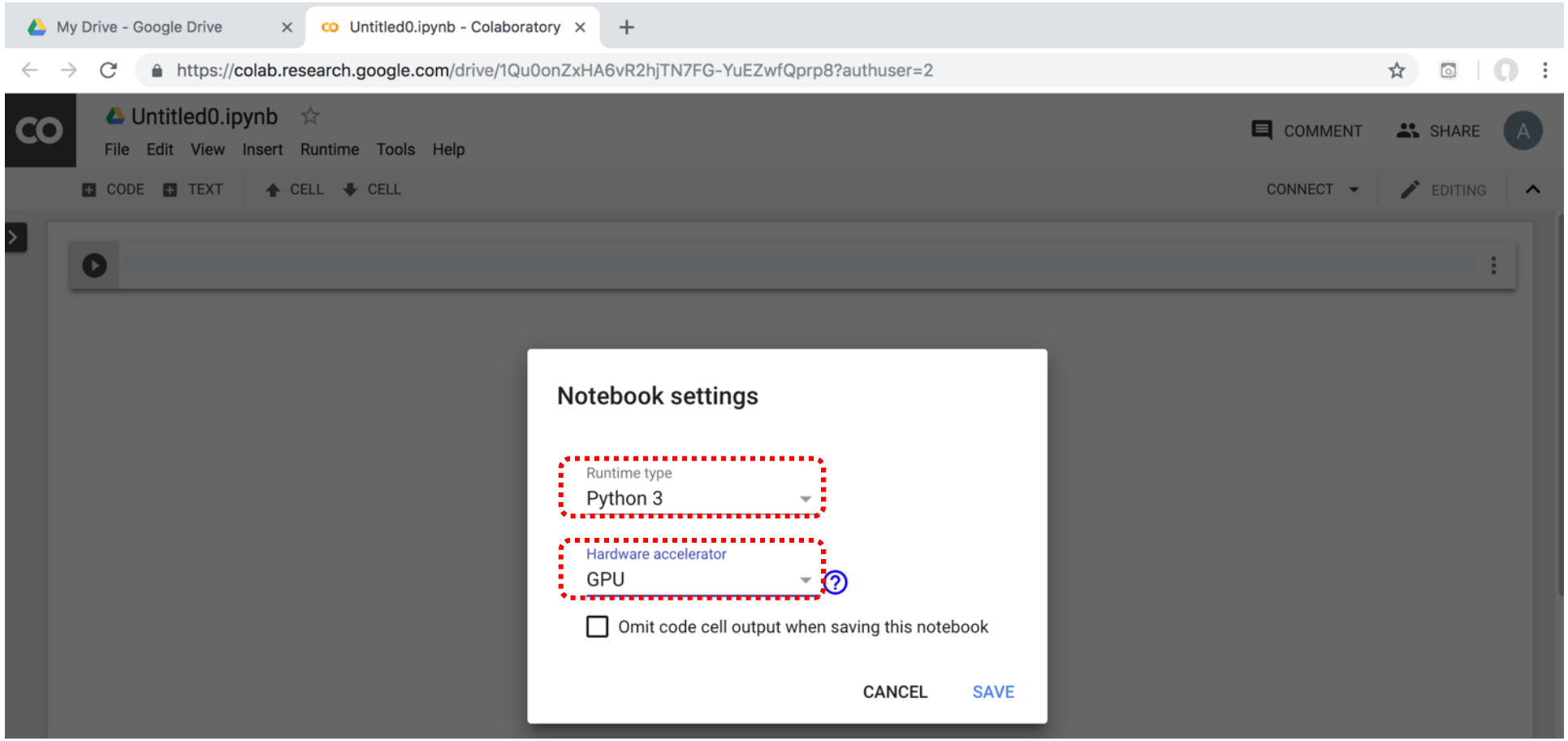
Google Colab



Google Colab



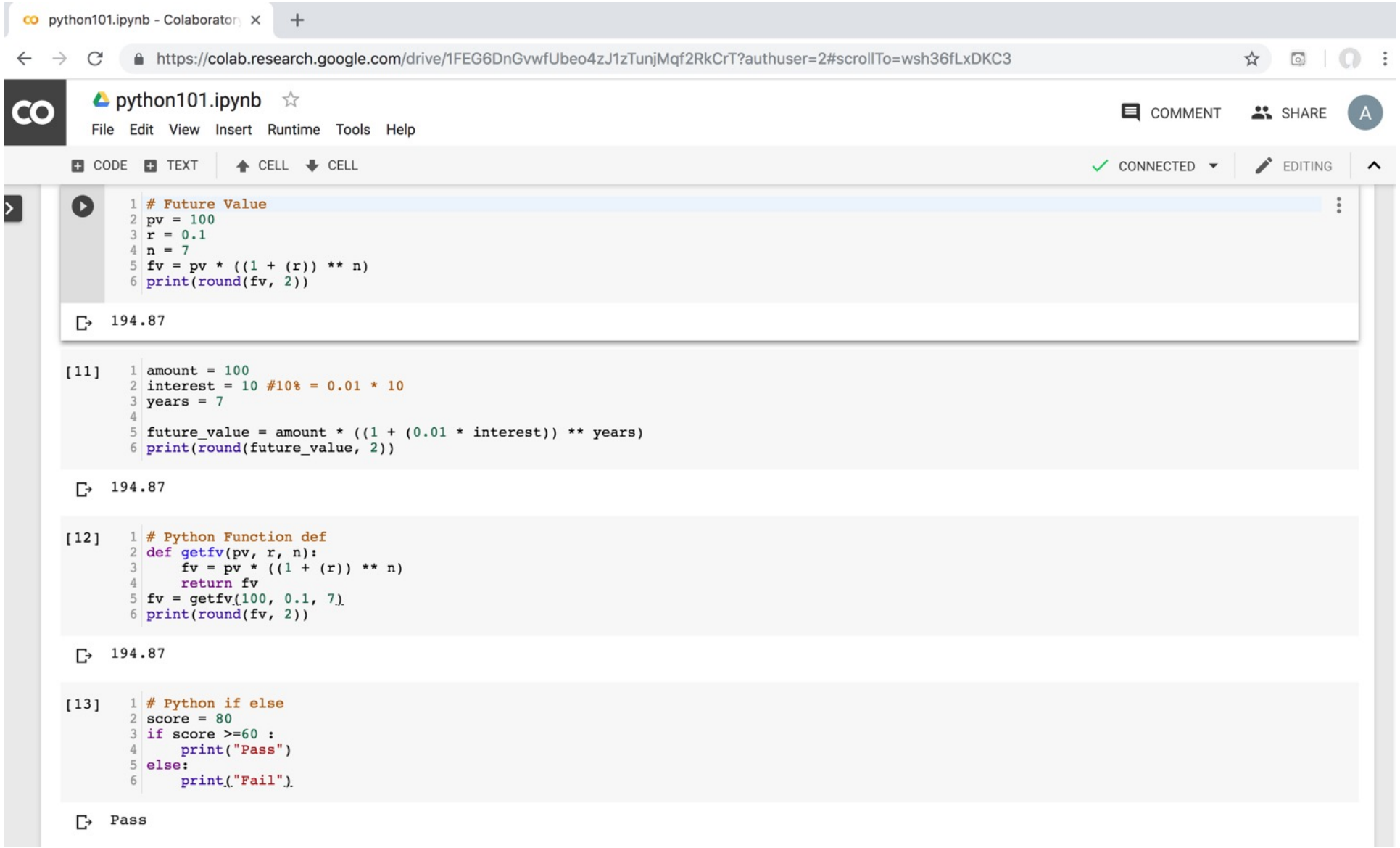
Run Jupyter Notebook
Python3 GPU
Google Colab



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

Python in Google Colab (Python101)

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



https://tinyurl.com/aintpupython101

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

Python

Programming

Data Analytics and Visualization with Python

Data Analytics and Visualization with Python

- NumPy
  - Numerical Python N-dimensional array
- Pandas
  - Data Analytics
- Matplotlib
  - Basic Data Visualization
- Seaborn
  - Advanced Visualization

W3Schools Python Numpy

HTML CSS JAVASCRIPT SQL PYTHON JAVA PHP HOW TO W3.CSS C C++ C# BOOTSTRAP REACT

Python Modules

- NumPy Tutorial
- Pandas Tutorial
- SciPy Tutorial
- Django Tutorial

NumPy Tutorial

- NumPy HOME
- NumPy Intro
- NumPy Getting Started
- NumPy Creating Arrays
- NumPy Array Indexing
- NumPy Array Slicing
- NumPy Data Types
- NumPy Copy vs View
- NumPy Array Shape
- NumPy Array Reshape
- NumPy Array Iterating
- NumPy Array Join
- NumPy Array Split
- NumPy Array Search
- NumPy Array Sort
- NumPy Array Filter

NumPy Random

- Random Intro
- Data Distribution

NumPy Tutorial

❮ Home Next ❯

NumPy is a Python library.

NumPy is used for working with arrays.

NumPy is short for "Numerical Python".

Learning by Reading

We have created 43 tutorial pages for you to learn more about NumPy.

Starting with a basic introduction and ends up with creating and plotting random data sets, and working with NumPy functions:

Basic Random ufunc

https://www.w3schools.com/python/numpy/default.asp

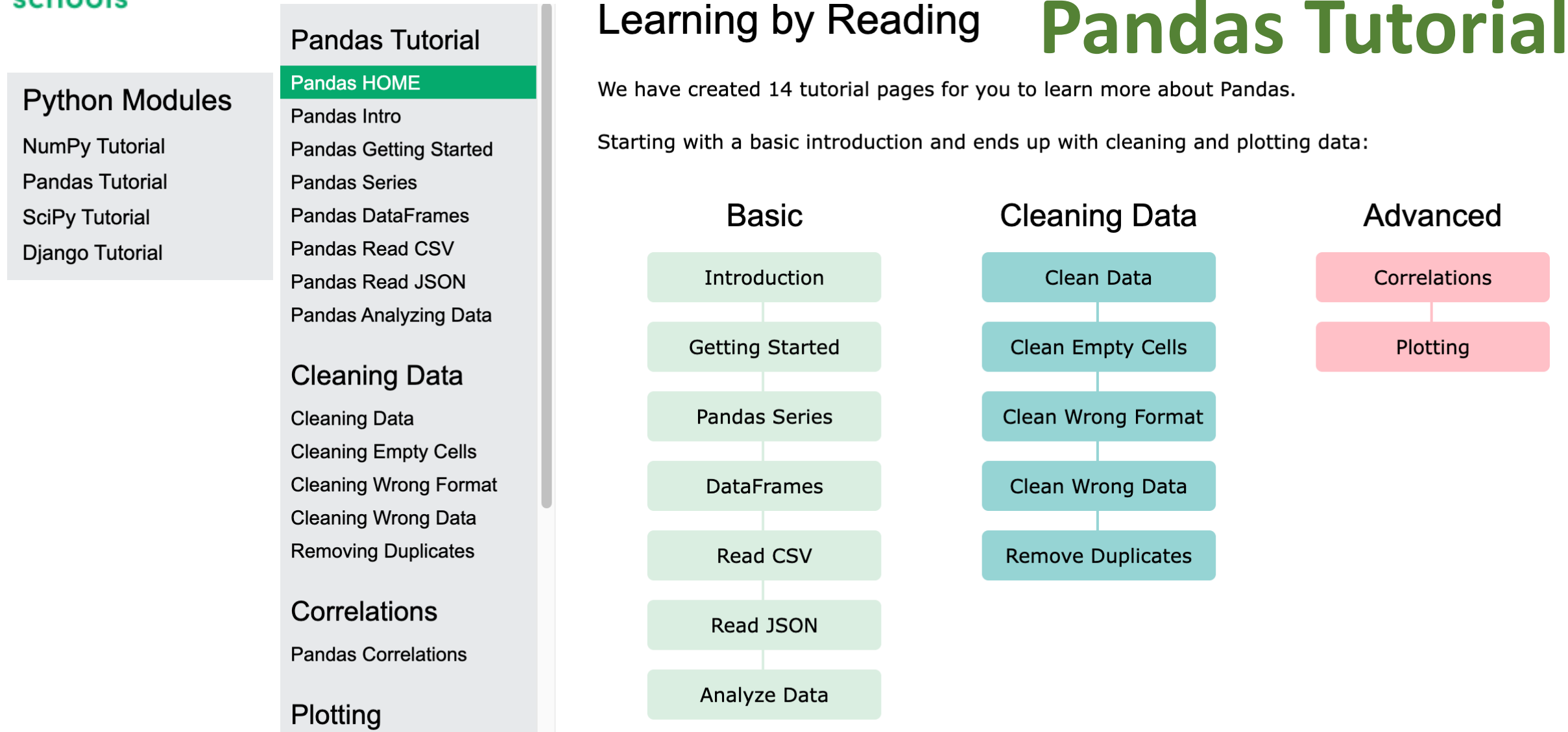
W3Schools Python Pandas

Learning by Reading

Pandas Tutorial

We have created 14 tutorial pages for you to learn more about Pandas.

Starting with a basic introduction and ends up with cleaning and plotting data:



https://www.w3schools.com/python/pandas/default.asp

W3Schools Python

HTML CSS JAVASCRIPT SQL PYTHON JAVA PHP HOW TO W3.CSS C C++ C# BOOTSTRAP REACT

Python Modules

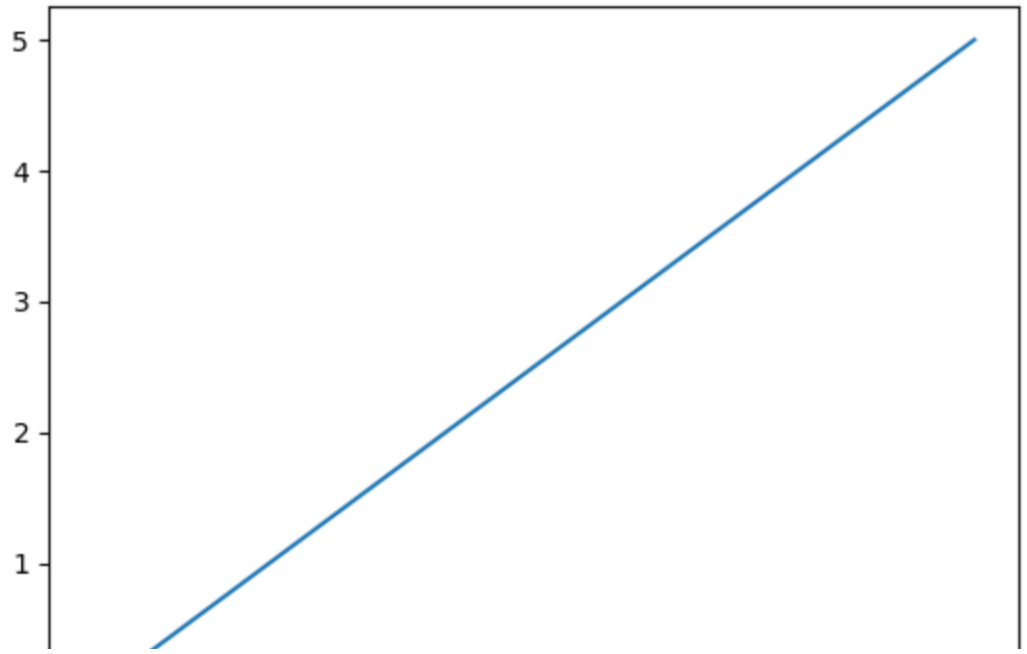
- NumPy Tutorial
- Pandas Tutorial
- SciPy Tutorial
- Django Tutorial

Python Matplotlib

- Matplotlib Intro
- Matplotlib Get Started
- Matplotlib Pyplot
- Matplotlib Plotting
- Matplotlib Markers
- Matplotlib Line
- Matplotlib Labels
- Matplotlib Grid
- Matplotlib Subplot
- Matplotlib Scatter
- Matplotlib Bars
- Matplotlib Histograms
- Matplotlib Pie Charts

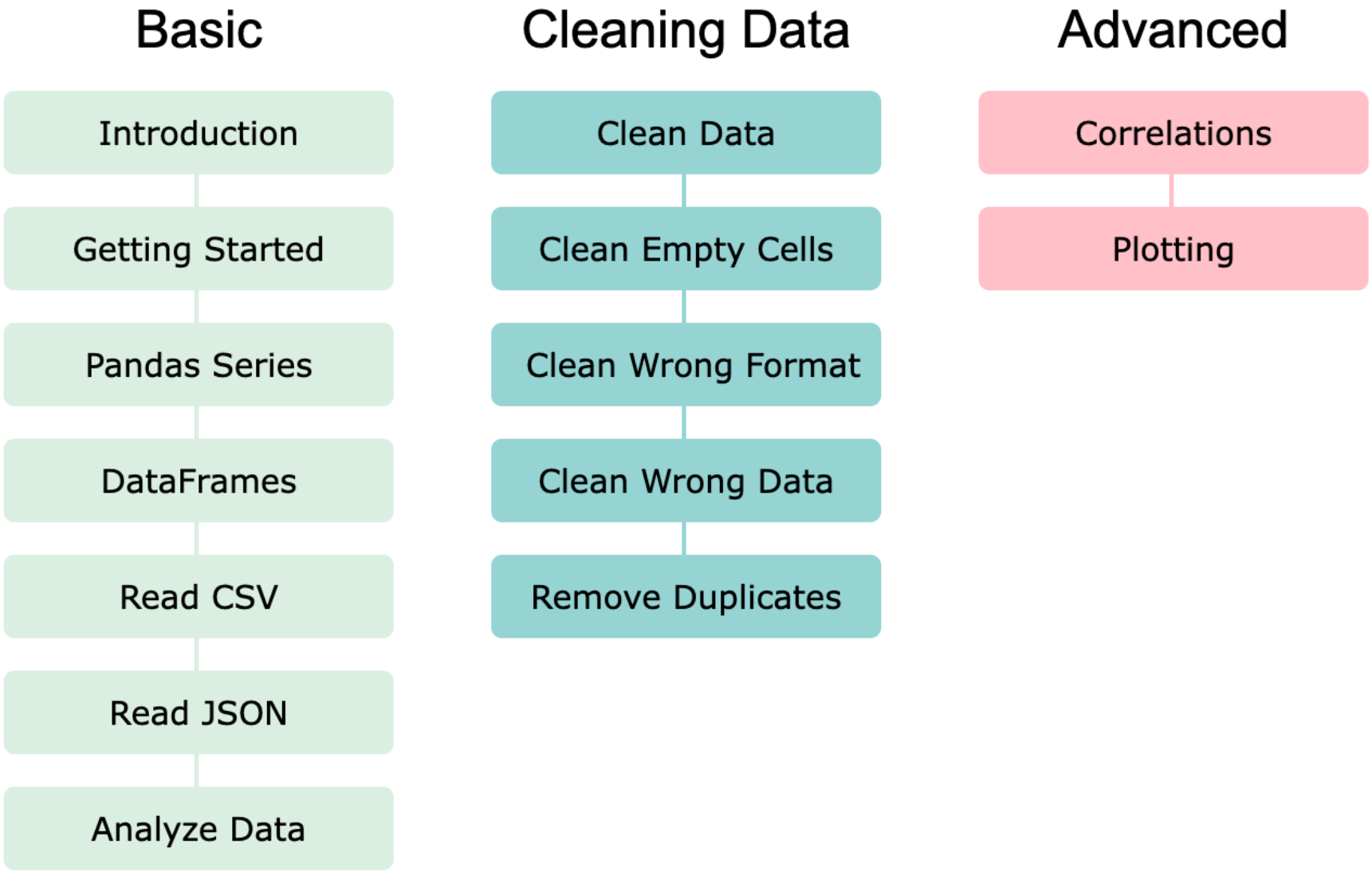
Matplotlib Tutorial

❮ Previous Next ❯



https://www.w3schools.com/python/

Pandas: Data Analytics and Visualization



https://www.w3schools.com/python/pandas/default.asp

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

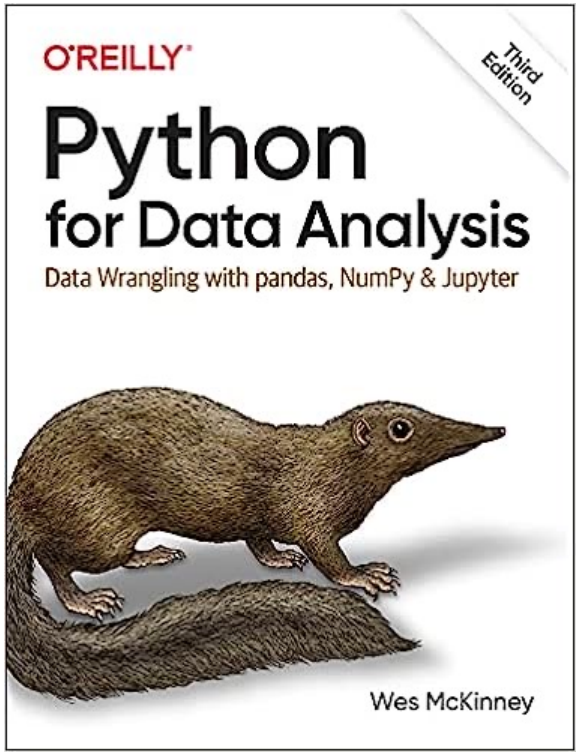
3rd-edition | 3 branches | 0 tags | Go to file | Code

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

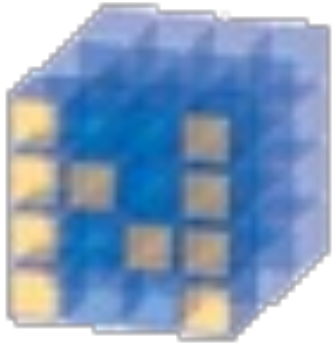
About

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



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

Numpy



NumPy

Base

N-dimensional array

package

NumPy

is the

fundamental package

for

scientific computing

with Python.

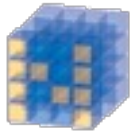
Source: http://www.numpy.org/

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

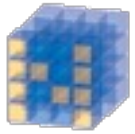
NumPy ndarray

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**

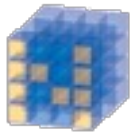


NumPy

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])
```

Python Data Structures

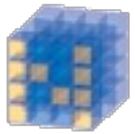
```
fruits = ["apple", "banana", "cherry"] #lists []
colors = ("red", "green", "blue") #tuples ()
animals = {'cat', 'dog'} #sets {}
person = {"name" : "Tom", "age" : 20} #dictionaries {}
```

https://tinyurl.com/aintpupython101

Lists []

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

```
4
60
70
90
```



NumPy

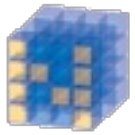
NumPy Create Array

```
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
```

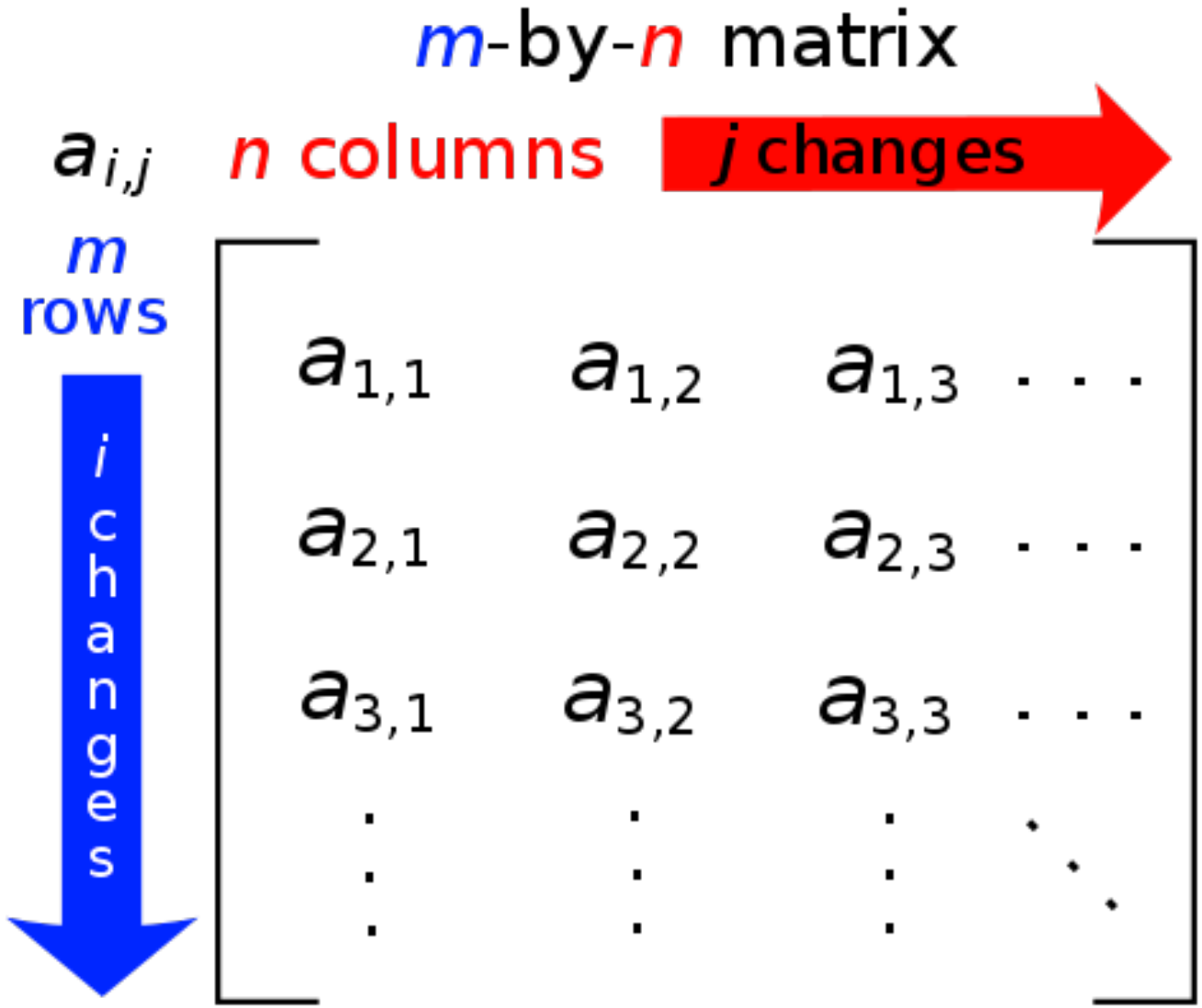
```
2
```

```
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)

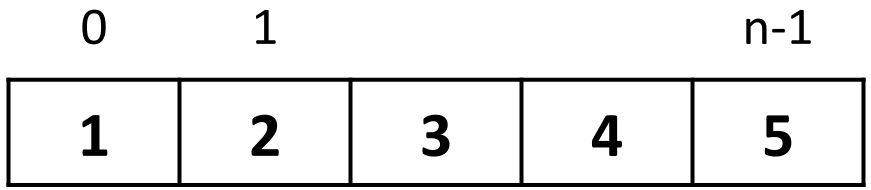
0	1			n-1
1	**2**	**3**	**4**	**5**

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)
```

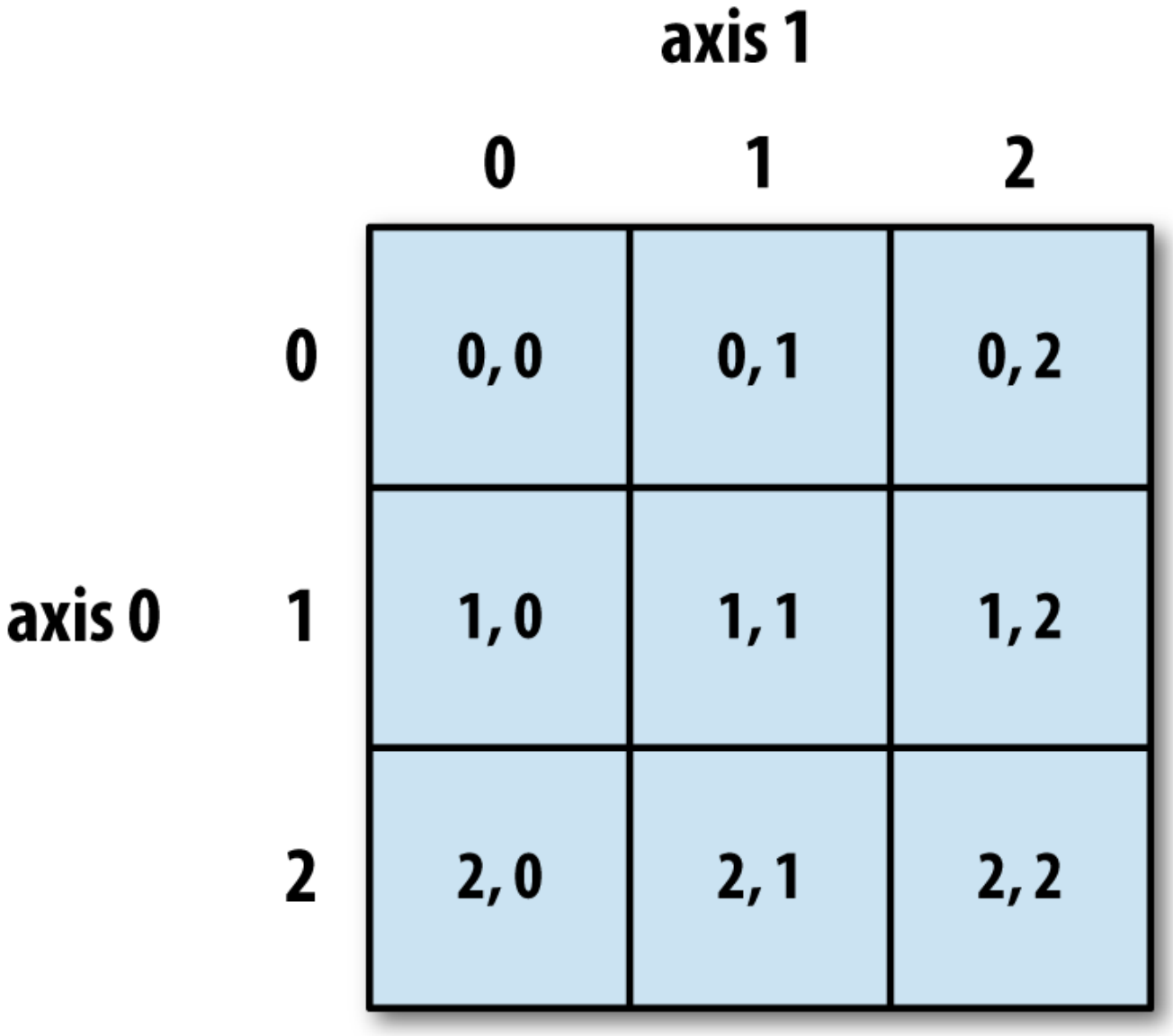
```
(3, 4)
```

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

NumPy Basics: Arrays and Vectorized Computation

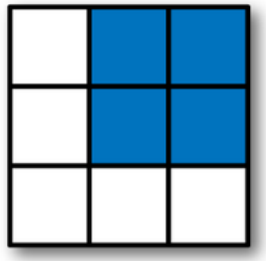
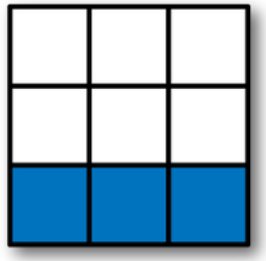
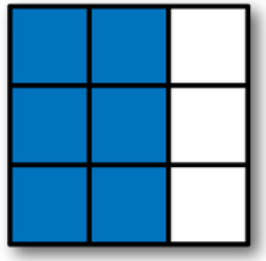
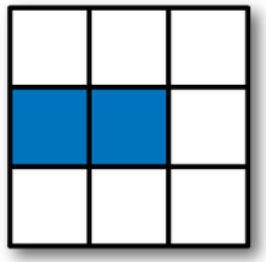
Source: https://www.safaribooksonline.com/library/view/python-for-data/9781449323592/ch04.html

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	.

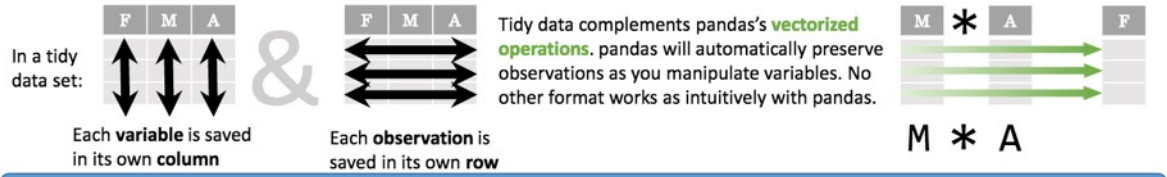
Source: http://pandas.pydata.org/pandas-docs/stable/comparison_with_sas.html

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

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

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

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.

		a	b	c
n	v			
d	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
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'])))
```

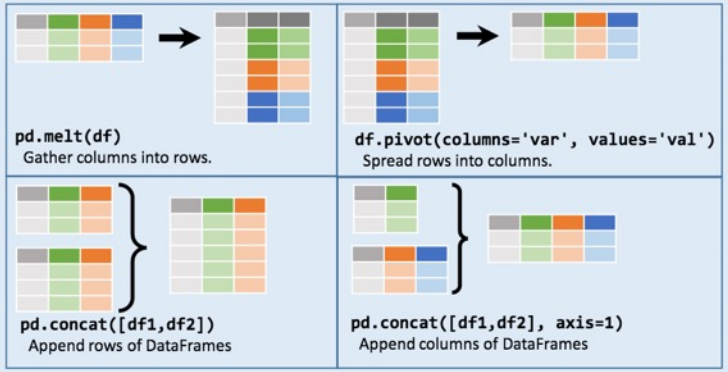
Create DataFrame with a MultiIndex

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).

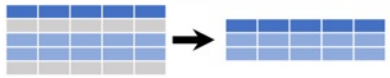
df=df.rename(columns = {'y':'year'})
Rename the columns of a DataFrame

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.

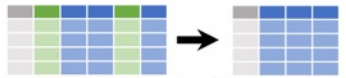
df.nlargest(n, 'value')
Select and order top n entries.

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 .

http://pandas.pydata.org/ This cheat sheet inspired by Rstudio Data Wrangling Cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf) Written by Irv Lustig, Princeton Consultants

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

pandas

```
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 Accounting Application with Pandas

```python
import pandas as pd

# Create a DataFrame to store transactions
columns = ['Date', 'Description', 'Amount']
ledger = pd.DataFrame(columns=columns)

# Function to add a transaction
def add_transaction(date, description, amount):
global ledger
new_transaction = pd.DataFrame([[date, description, amount]], columns=columns)
ledger = pd.concat([ledger, new_transaction], ignore_index=True)

# Function to view the ledger
def view_ledger():
print(ledger)

# Function to get the current balance
def get_balance():
return ledger['Amount'].sum()

# Adding sample transactions
add_transaction('2023-11-01', 'Income', 1000)
add_transaction('2023-11-02', 'Groceries', -200)
add_transaction('2023-11-03', 'Utilities', -100)

# Viewing the ledger
view_ledger()

# Checking the current balance
print("Current Balance:", get_balance())
```

```
  Date Description Amount
0 2023-11-01 Income 1000
1 2023-11-02 Groceries -200
2 2023-11-03 Utilities -100
Current Balance: 700
```

https://tinyurl.com/aintpupython101

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

matplotlib
Version 3.3.4

Fork me on GitHub

Installation Documentation Examples Tutorials Contributing Search

home | contents » Matplotlib: Python plotting modules | index

Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Matplotlib makes easy things easy and hard things possible.

Create

- Develop publication quality plots with just a few lines of code
- Use interactive figures that can zoom, pan, update...

Customize

- Take full control of line styles, font properties, axes properties...
- Export and embed to a number of file formats and interactive environments

Extend

- Explore tailored functionality provided by third party packages
- Learn more about Matplotlib through the many external learning resources

Latest stable release
3.3.4: docs | changelog

Last release for Python 2
2.2.5: docs | changelog

Development version
docs

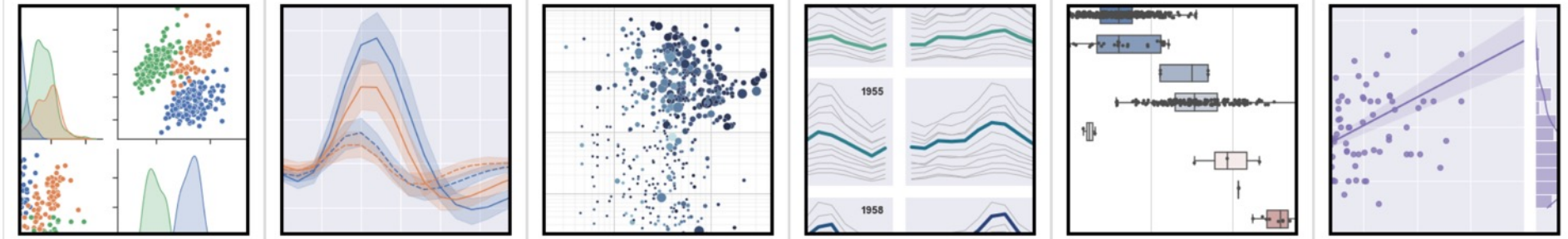
Support Matplotlib

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

plotly | Graphing Libraries

Star 9,085 DO MORE WITH DASH

Search...

Quick Start

- Getting Started
- Is Plotly Free?
- Figure Reference
- API Reference
- Dash
- GitHub
- community.plotly.com

Examples

- Fundamentals
- Basic Charts
- Statistical Charts
- Artificial Intelligence and Machine Learning
- Scientific Charts
- Financial Charts
- Maps
- 3D Charts

Plotly Python Open Source Graphing Library

Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.
Plotly.py is free and open source and you can view the source, report issues or contribute on GitHub.

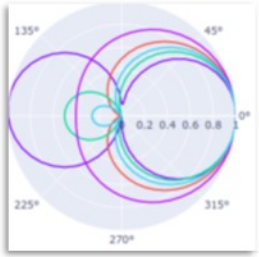
Our recommended IDE for Plotly's Python graphing library is Dash Enterprise's Data Science Workspaces, which has both Jupyter notebook and Python code file support.
Find out if your company is using Dash Enterprise.

Install Dash Enterprise on Azure | Install Dash Enterprise on AWS

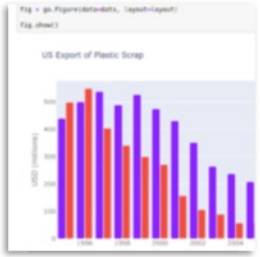
Fundamentals

More Fundamentals »

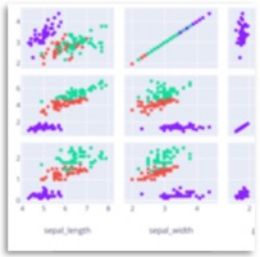
The Figure Data Structure



Creating and Updating Figures



Displaying Figures



Plotly Express

Analytical Apps with Dash

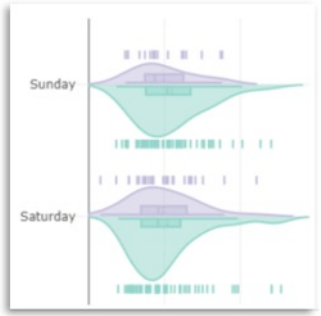
https://plotly.com/python/

plotly

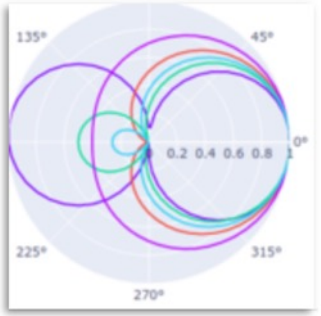
Python Plotly Graphing Library

Fundamentals

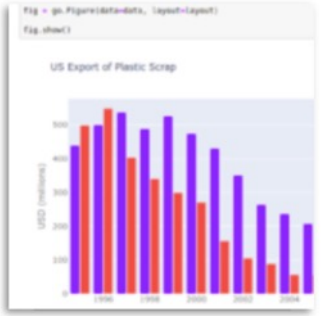
More Fundamentals »



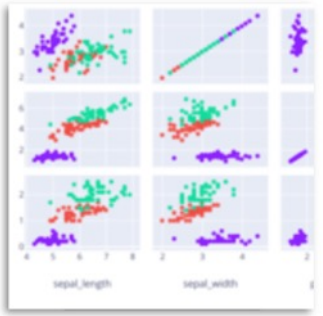
The Figure Data Structure



Creating and Updating Figures



Displaying Figures

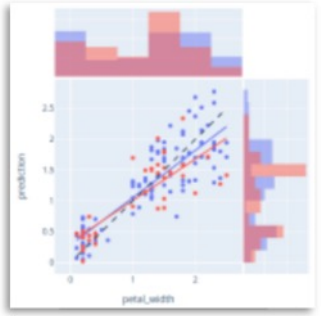


Plotly Express

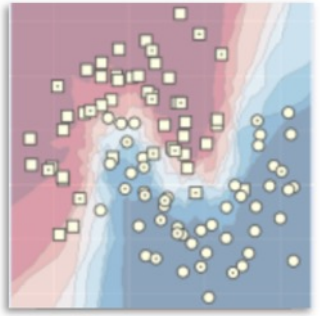
Analytical Apps with Dash

Artificial Intelligence and Machine Learning

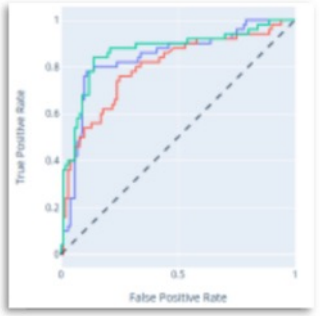
More AI and ML »



ML Regression



kNN Classification



ROC and PR Curves

PCA Visualization

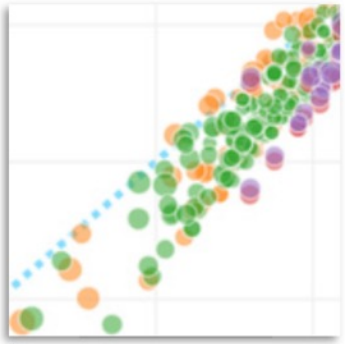
AI/ML Apps with Dash

https://plotly.com/python/

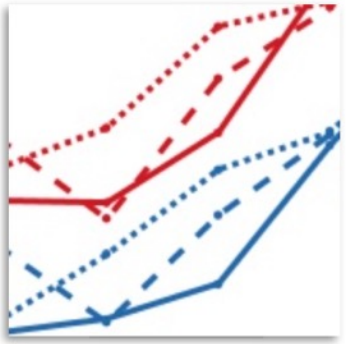
Python Plotly Graphing Library

Basic Charts

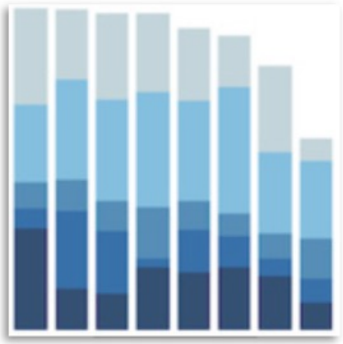
More Basic Charts »



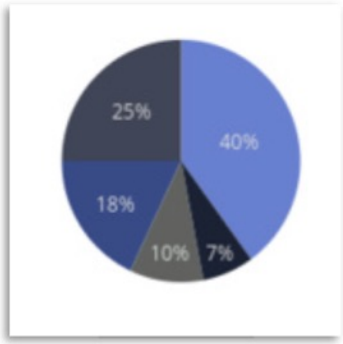
Scatter Plots



Line Charts



Bar Charts

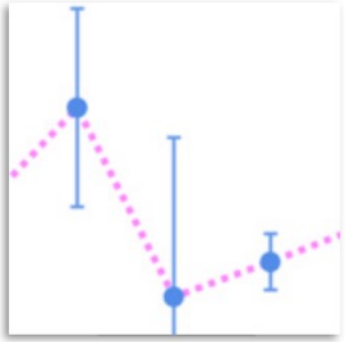


Pie Charts

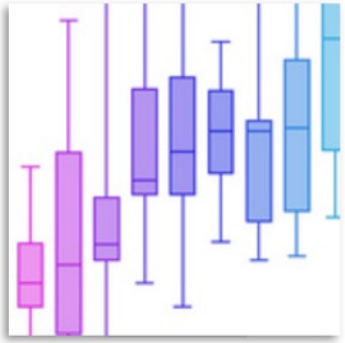
Bubble Charts

Statistical Charts

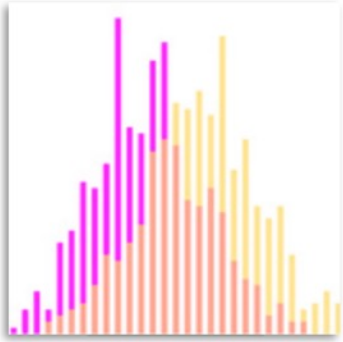
More Statistical Charts »



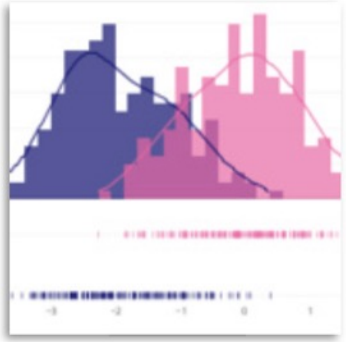
Error Bars



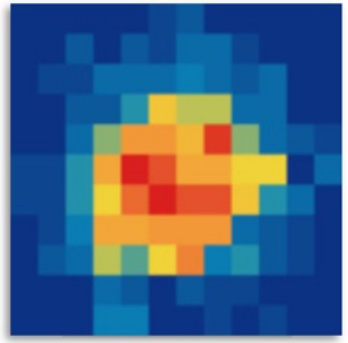
Box Plots



Histograms



Distplots



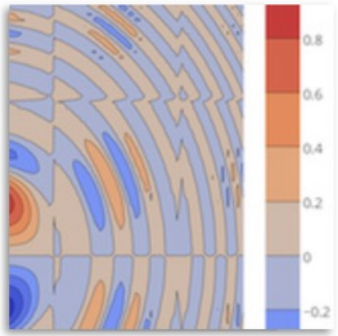
2D Histograms

https://plotly.com/python/

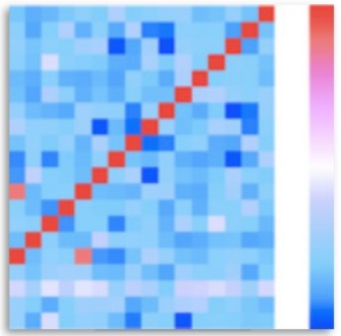
Python Plotly Graphing Library

Scientific Charts

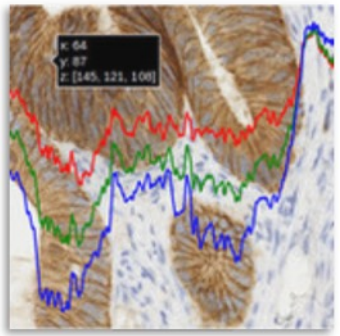
More Scientific Charts »



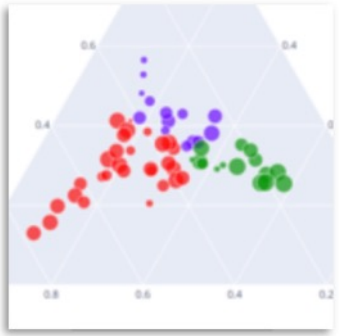
Contour Plots



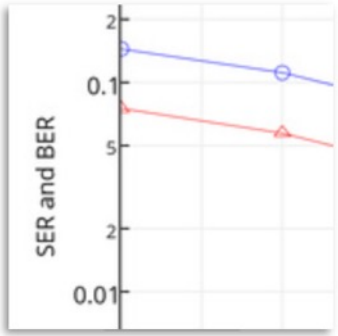
Heatmaps



Imshow



Ternary Plots



Log Plots

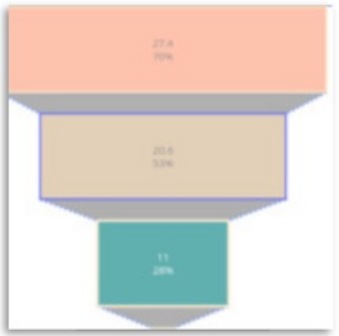
Financial Charts

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

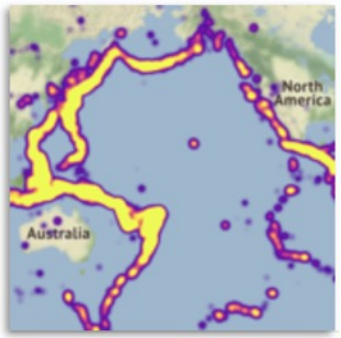
More Maps »

Mapbox Choropleth Maps

Lines on Mapbox

Filled Area on Maps

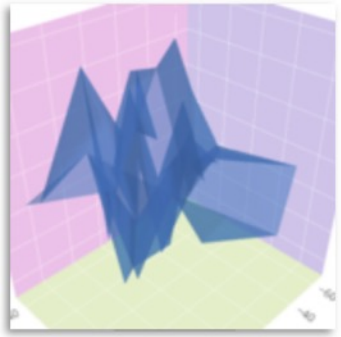
Bubble Maps



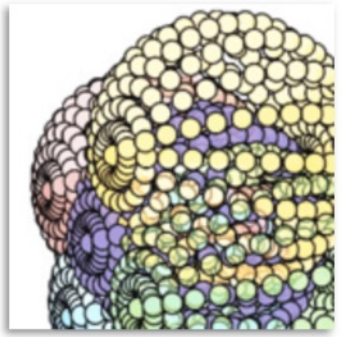
Mapbox Density Heatmap

3D Charts

More 3D Charts »

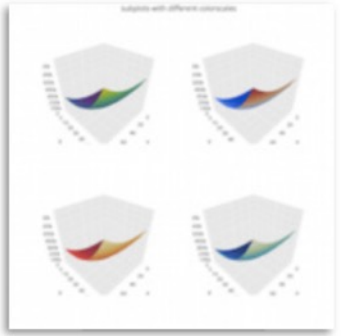


3D Axes

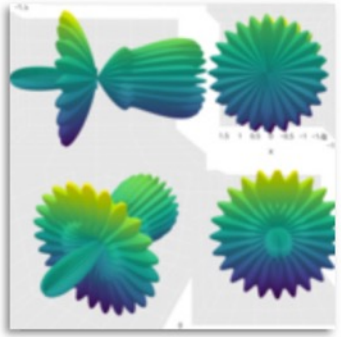


3D Scatter Plots

3D Surface Plots



3D Subplots



3D Camera Controls

https://plotly.com/python/

Python Plotly Graphing Library

Subplots

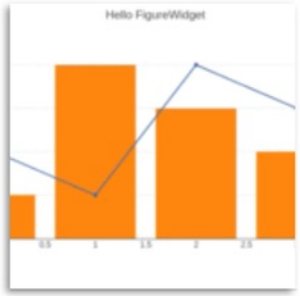
Mixed Subplots

Map Subplots

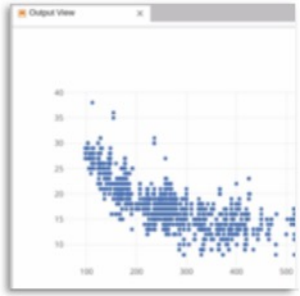
Table and Chart Subplots

Figure Factory Subplots

Jupyter Widgets Interaction

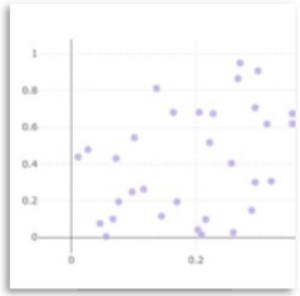


Plotly FigureWidget Overview



Jupyter Lab with FigureWidget

Interactive Data Analysis with FigureWidget ipywidgets



Click Events

https://plotly.com/python/

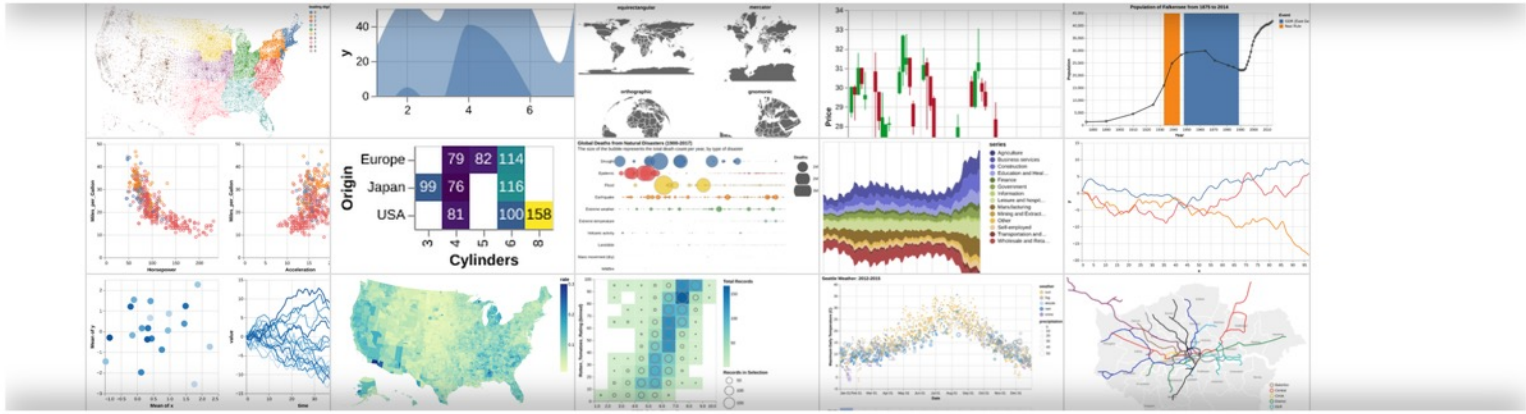
Python Bokeh

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

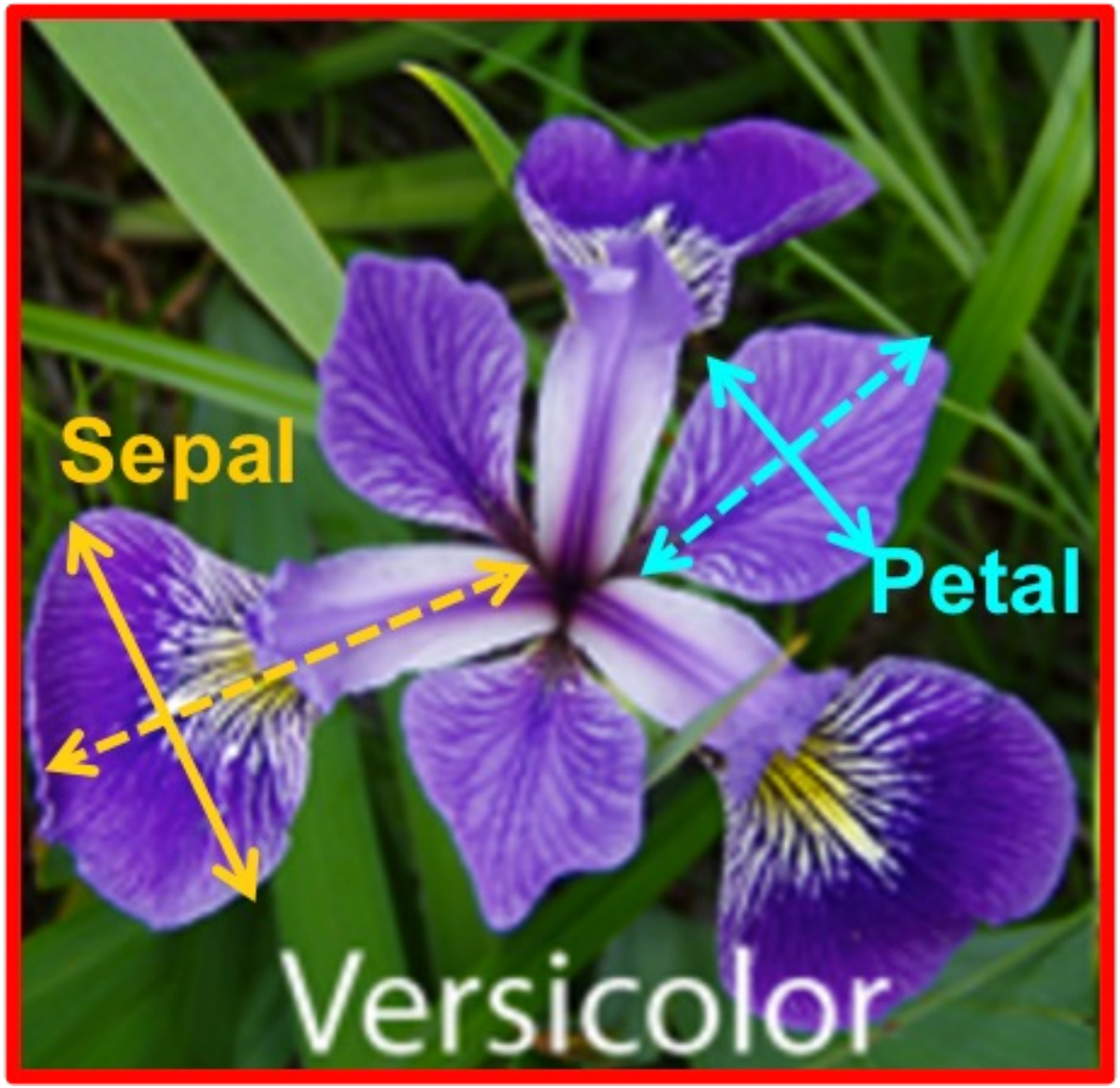
versicolor

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

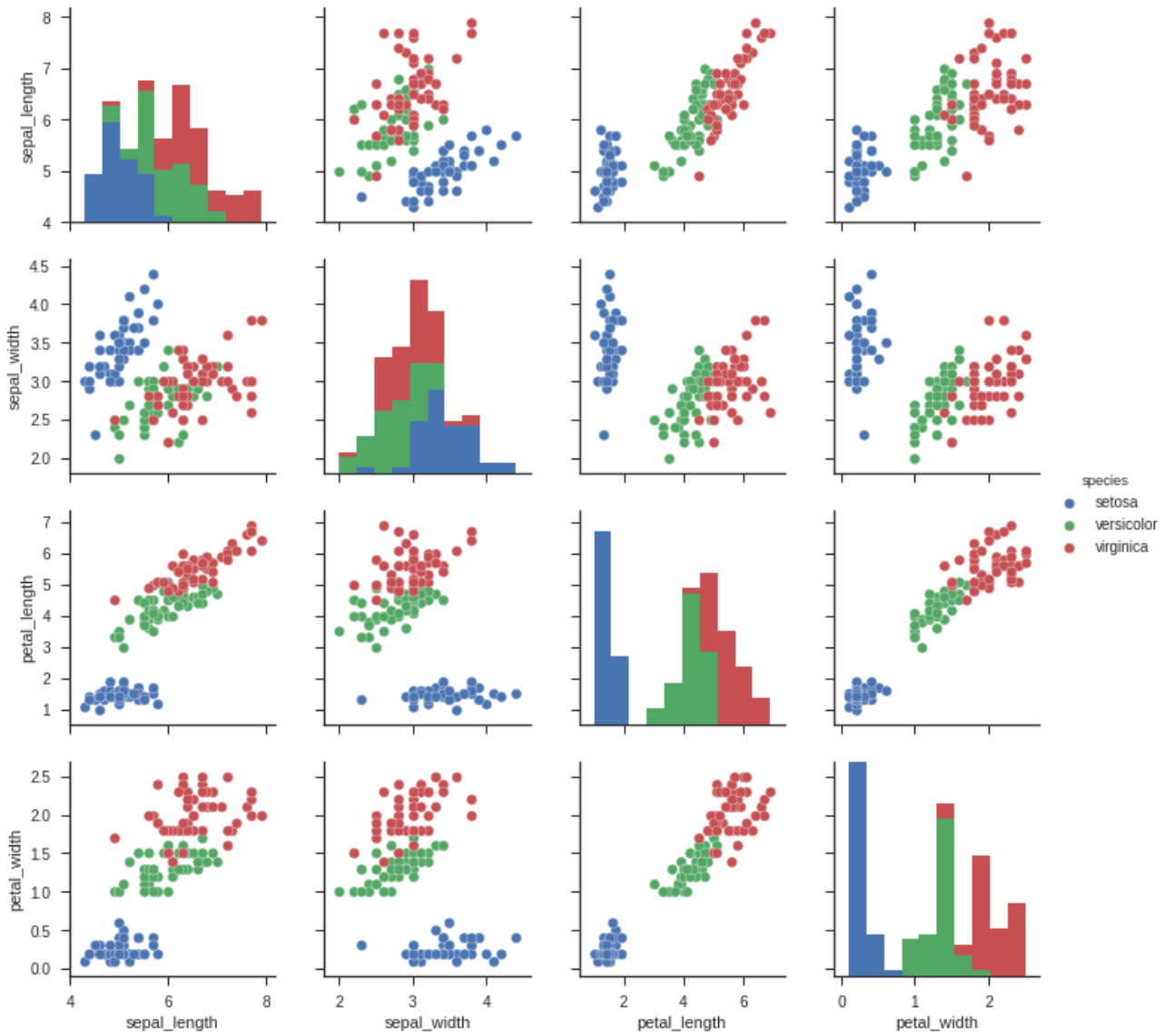
```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa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

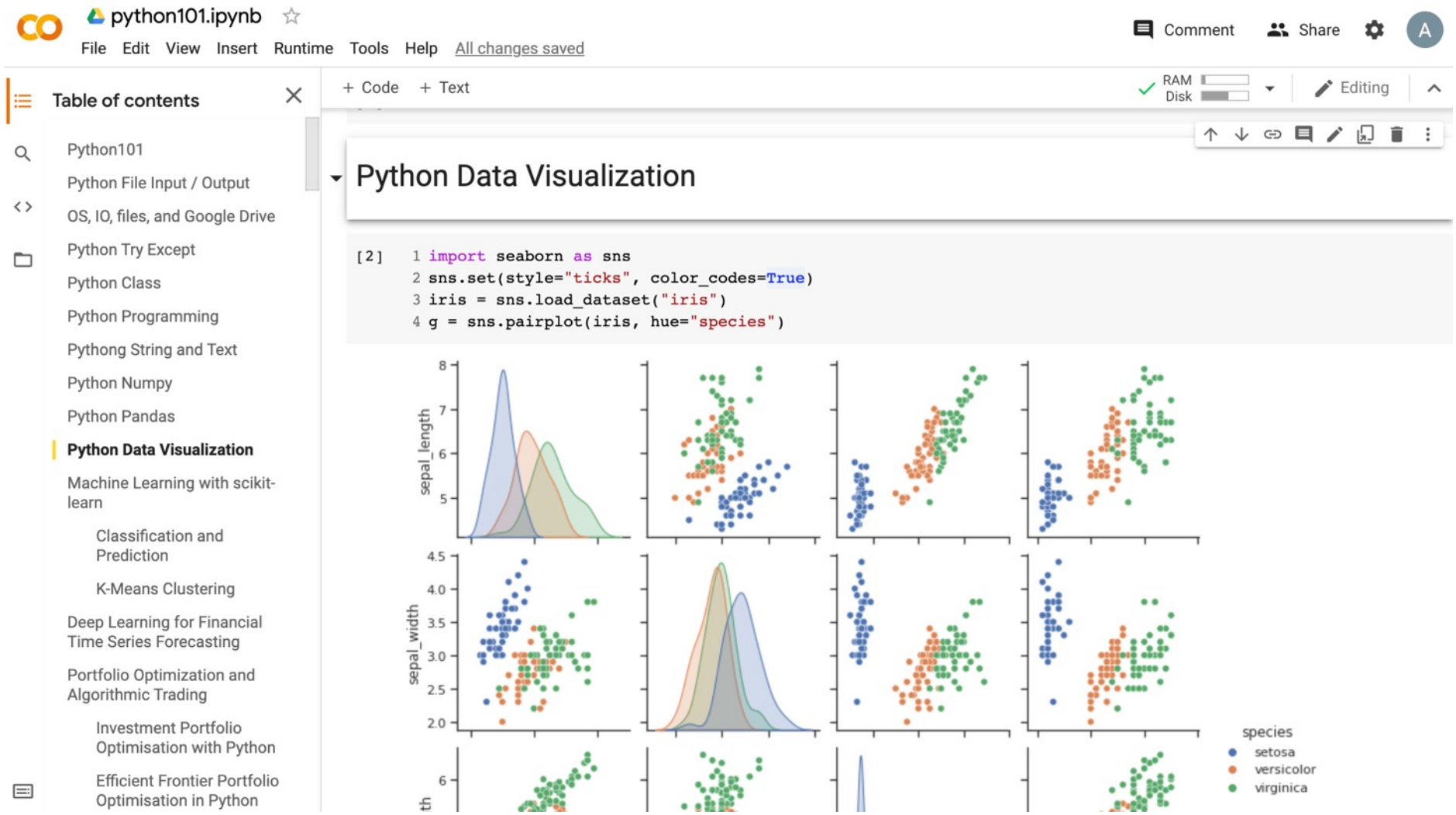
versicolor

Iris Data Visualization



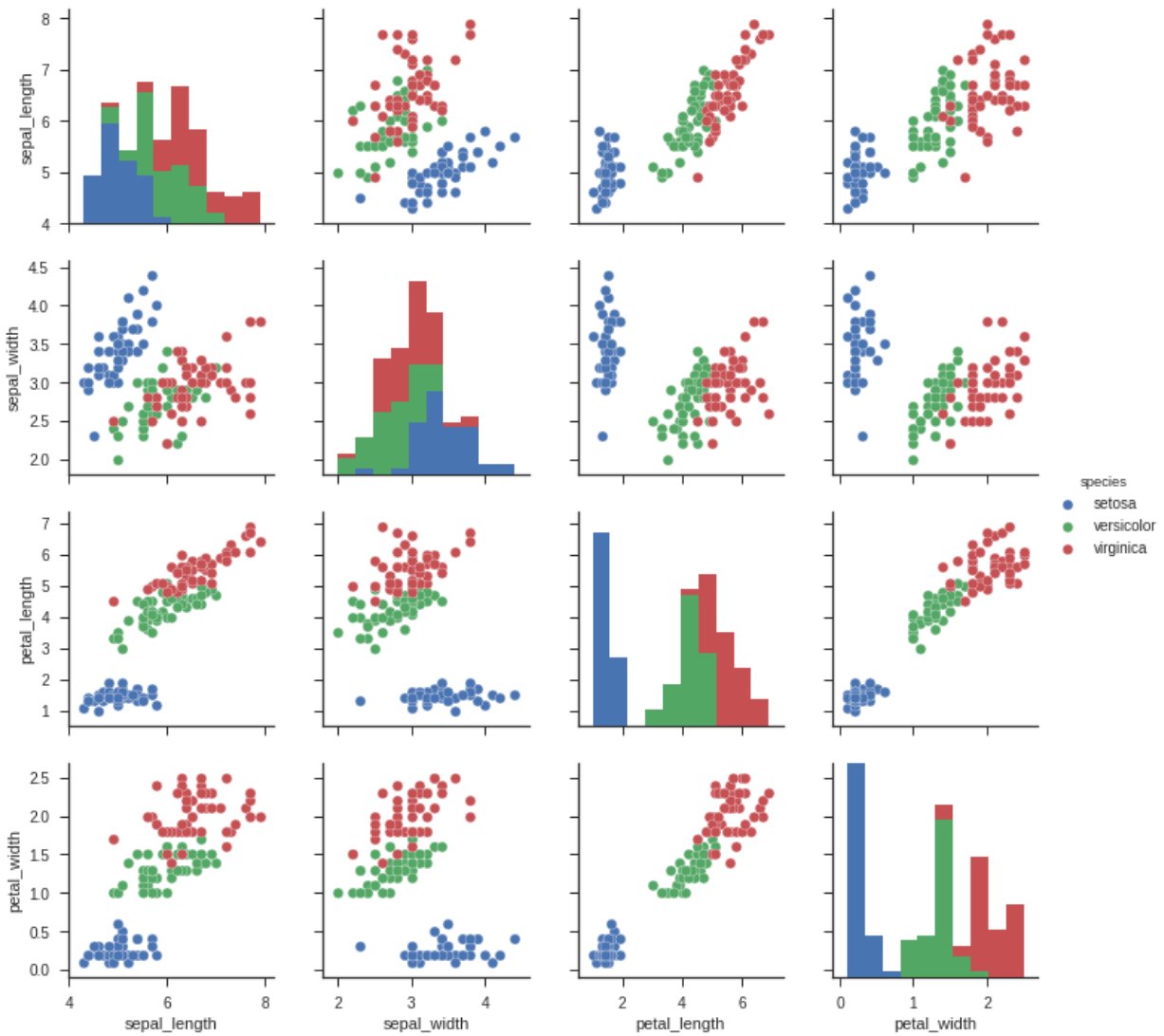
Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html

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)
```

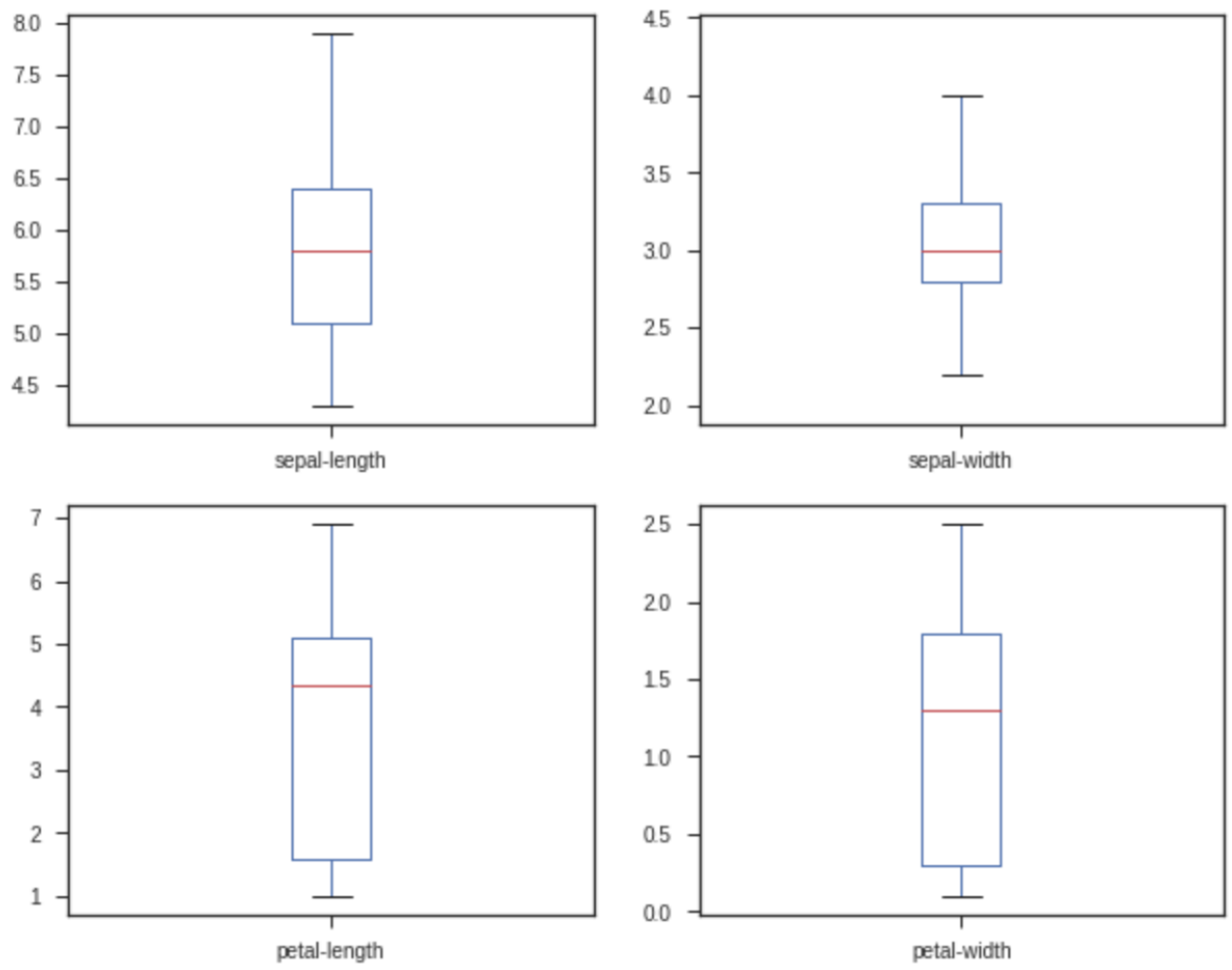
```
(150, 5)
```

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

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

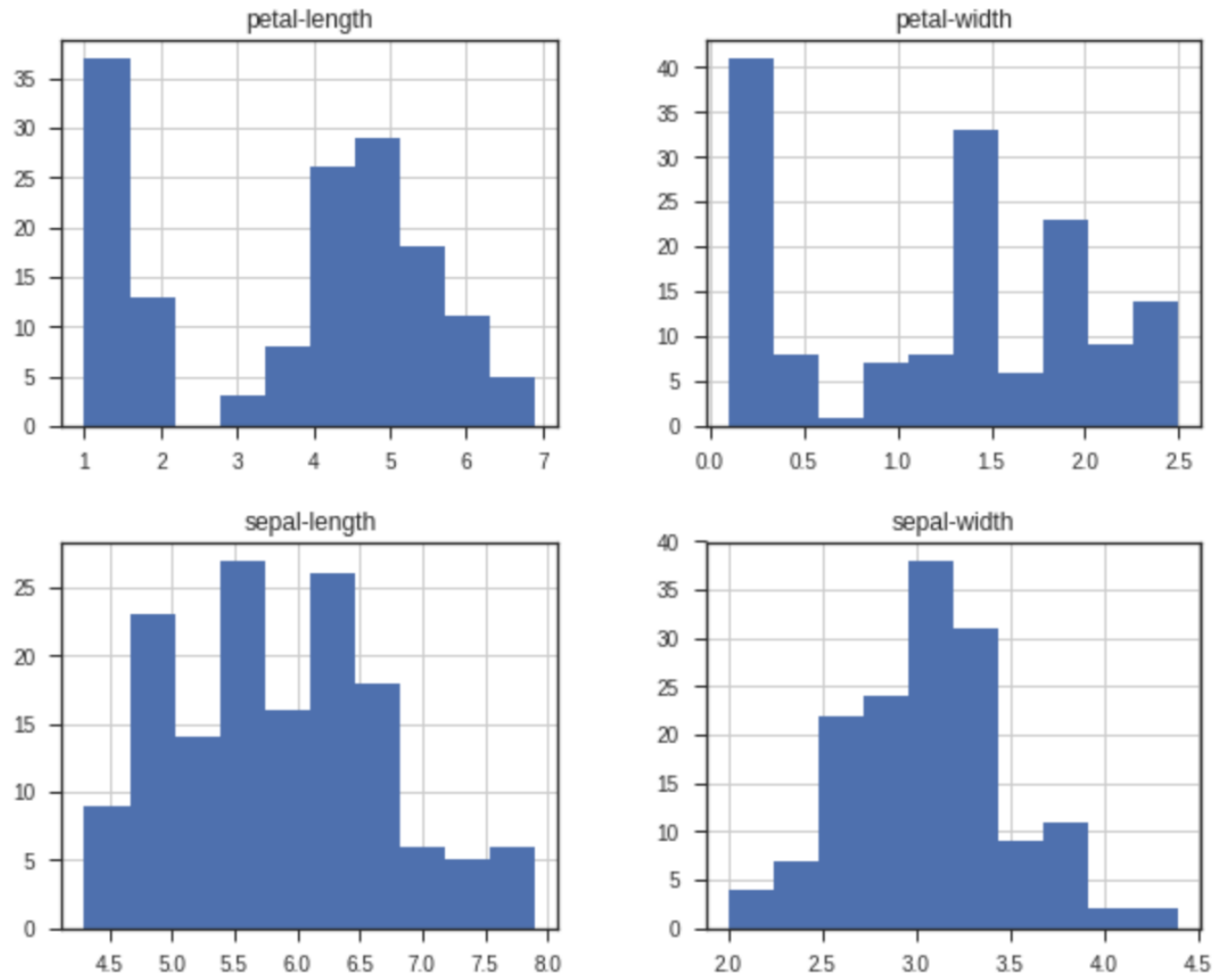
```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

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



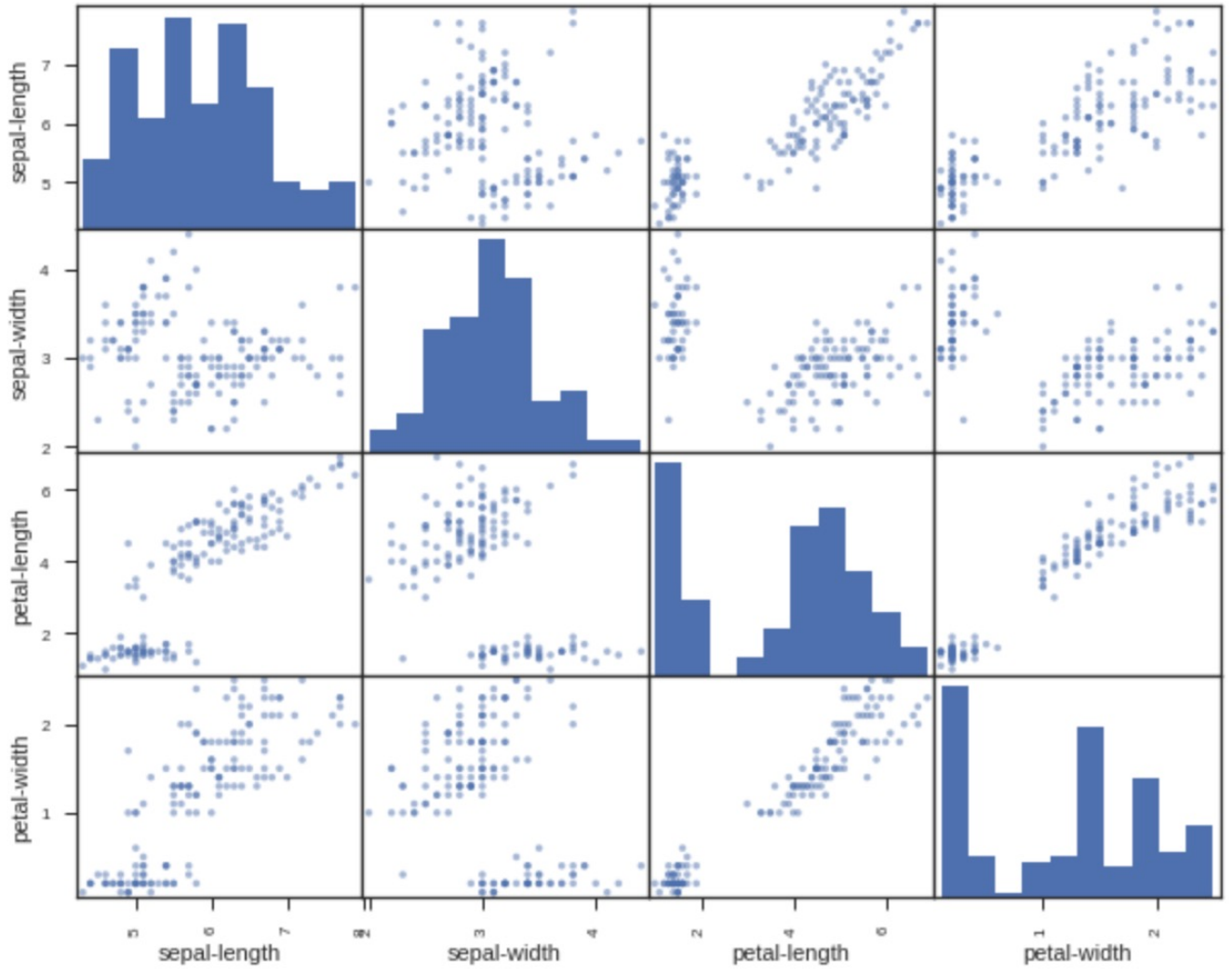
```
df.hist()
plt.show()
```

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



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

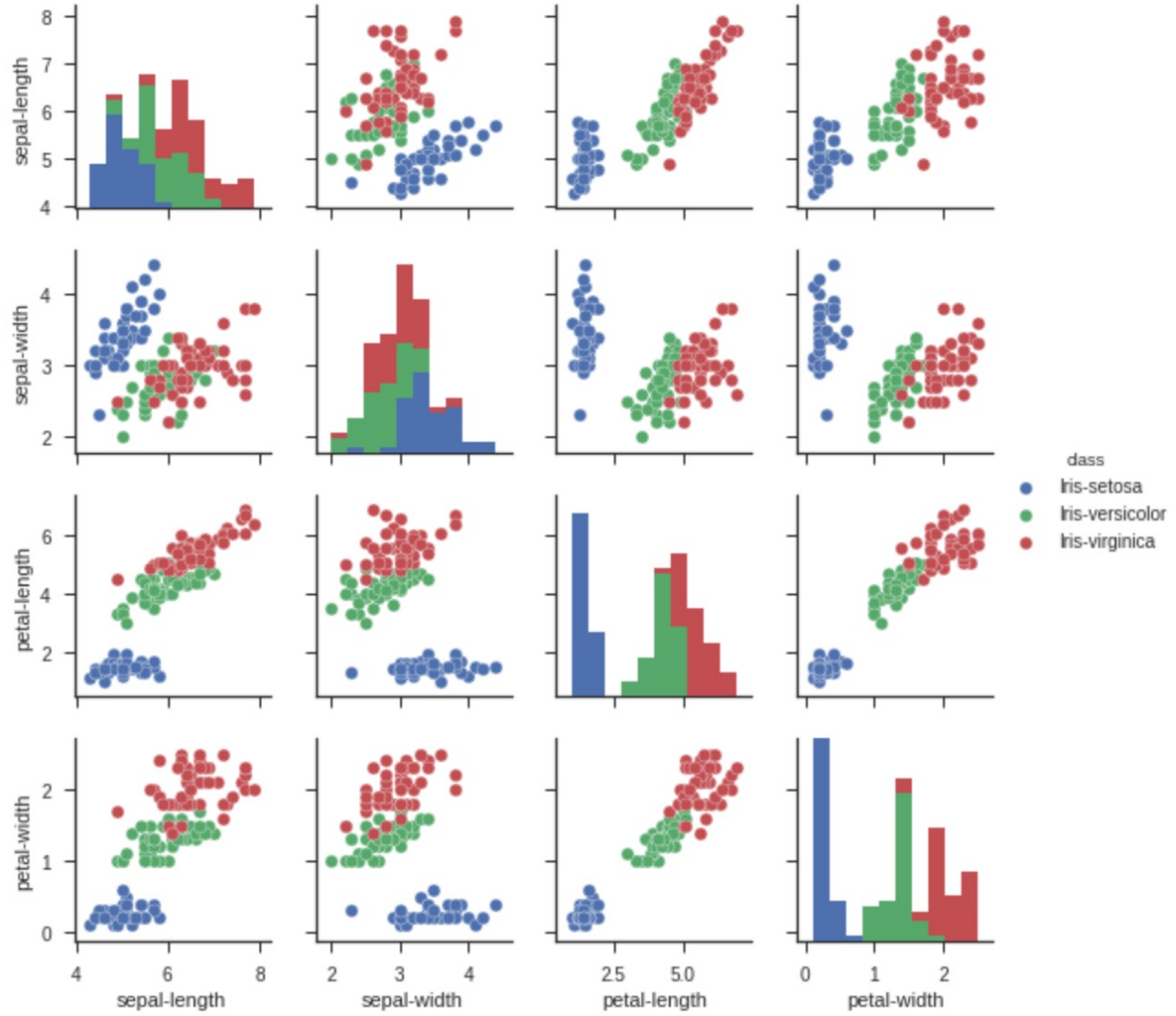

```
scatter_matrix(df)
plt.show()
```

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

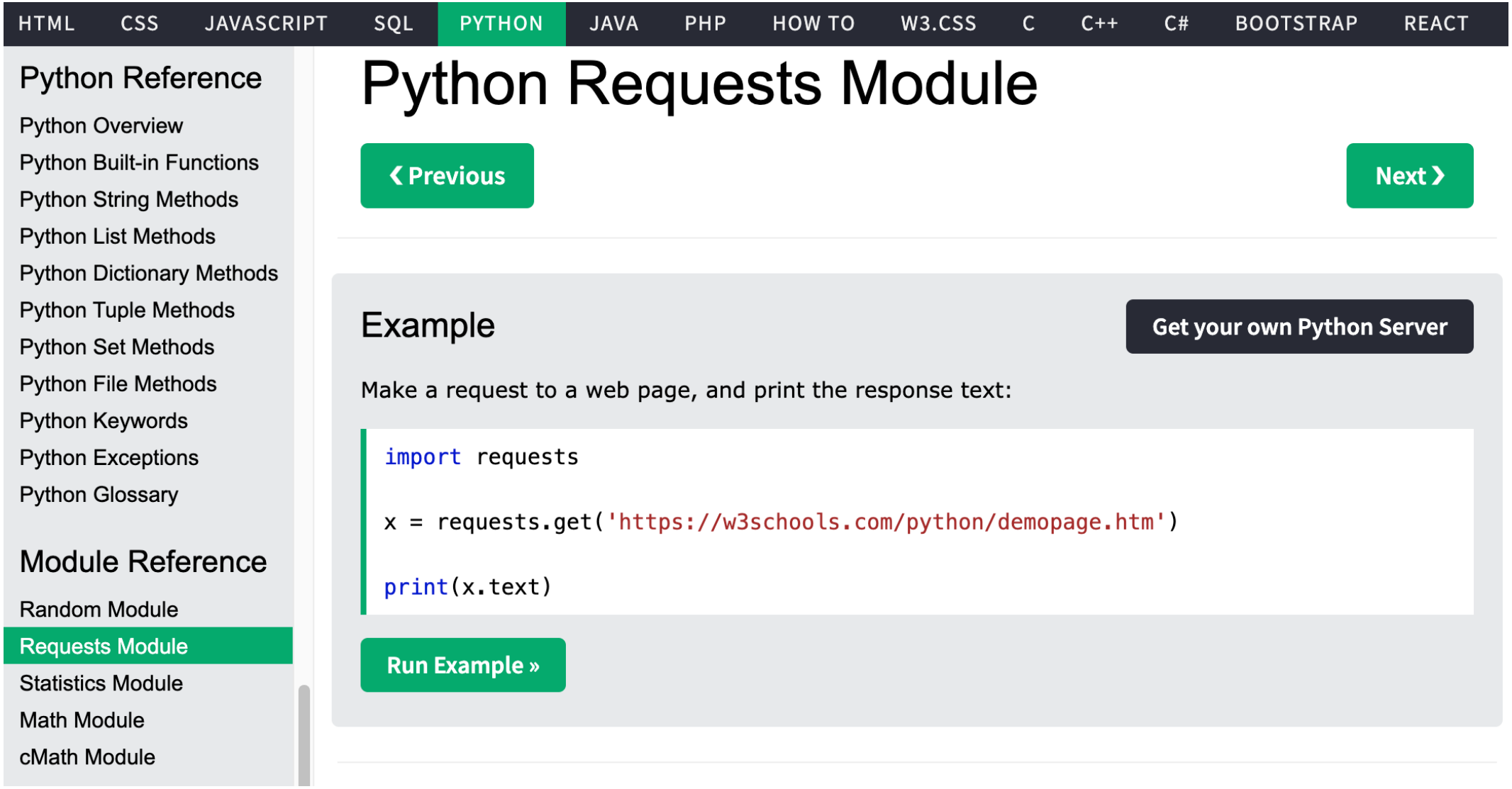
```
sns.pairplot(df, hue="class", size=2)
```

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



Obtaining Data From the Web with Python

W3Schools Python Requests Module



https://www.w3schools.com/python/

Python requests

```
! pip install requests
```

```
import requests

x = requests.get('https://w3schools.com/python/demopage.htm')

print(x.text)
```

```
<!DOCTYPE html>
<html>
<body>
<h1>This is a Test Page</h1>
</body>
</html>
```

Requests Methods

Method	Description
delete(*url, args*)	Sends a DELETE request to the specified url
get(*url, params, args*)	Sends a GET request to the specified url
head(*url, args*)	Sends a HEAD request to the specified url
patch(*url, data, args*)	Sends a PATCH request to the specified url
post(*url, data, json, args*)	Sends a POST request to the specified url
put(*url, data, args*)	Sends a PUT request to the specified url
request(*method, url, args*)	Sends a request of the specified method to the specified url

https://www.w3schools.com/python/module_requests.asp

Statistical Analysis with Python

Statistical Analysis with Python

- **Descriptive Statistics**
  - `df.describe()`
  - `df.mean(), df.std() df.median() df.max() df.min()`
  - `df.skew(), df.kurtosis()`
- **Inferential Statistics**
  - `import scipy.stats as stats`
  - `t_statistic, p_value = stats.ttest_ind(ga, gb)`

SciPy: Scientific Python

- **SciPy is a scientific computation library that uses NumPy underneath.**
- **SciPy stands for Scientific Python.**
- **SciPy provides more utility functions for optimization, stats and signal processing.**

SciPy Statistical Significance Tests

- **In statistics, statistical significance means that the result that was produced has a reason behind it, it was not produced randomly, or by chance.**

- **SciPy provides us with a module called scipy.stats, which has functions for performing statistical significance tests.**

https://www.w3schools.com/python/scipy/scipy_statistical_significance_tests.php

ESG Score and Financial Performance

How Python can be used to analyze and visualize ESG-related financial data, combining statistical analysis with effective data visualization techniques.

1. Two datasets for high and low ESG score companies are generated, each with normally distributed ESG scores and financial performance metrics.

2. Descriptive statistics provide insights into the central tendency and variability of the data.

3. An inferential t-test is used to determine if there is a statistically significant difference in financial performance between the two groups.

4. Data visualizations, including histograms and box plots, illustrate the distribution of ESG scores and financial performance, helping to visualize the differences between the high and low ESG score companies.

Python Statistical Analysis

```
import pandas as pd
import numpy as np
import scipy.stats as stats
```

Python Statistical Analysis

```
#Step 1: Generate Datasets
import pandas as pd
import numpy as np
import scipy.stats as stats
# Seed for reproducibility
np.random.seed(3)
# Number of companies in each group
n_companies = 50
# High ESG Score Companies (ESG scores and financial performance follow a normal
distribution)
high_esg_means = {'ESG_Score': 80, 'Financial_Performance': 15}
high_esg_stds = {'ESG_Score': 5, 'Financial_Performance': 5}
high_esg_data = {
'Company': [f'High_ESG_{i}' for i in range(n_companies)],
'ESG_Score': np.random.normal(high_esg_means['ESG_Score'],
high_esg_stds['ESG_Score'], n_companies),
'Financial_Performance': np.random.normal(high_esg_means['Financial_Performance'],
high_esg_stds['Financial_Performance'], n_companies)
}
high_esg_df = pd.DataFrame(high_esg_data)
print(high_esg_df)
```

Python Statistical Analysis

```
# Low ESG Score Companies
low_esg_means = {'ESG_Score': 30, 'Financial_Performance': 10}
low_esg_stds = {'ESG_Score': 5, 'Financial_Performance': 5}
low_esg_data = {
'Company': [f'Low_ESG_{i}' for i in range(n_companies)],
'ESG_Score': np.random.normal(low_esg_means['ESG_Score'], low_esg_stds['ESG_Score'],
n_companies),
'Financial_Performance': np.random.normal(low_esg_means['Financial_Performance'],
low_esg_stds['Financial_Performance'], n_companies)
}
low_esg_df = pd.DataFrame(low_esg_data)
print(low_esg_df)
```

```
   Company  ESG_Score  Financial_Performance
0    Low_ESG_0  34.371429               5.688458
1    Low_ESG_1  23.532317              15.358420
2    Low_ESG_2  29.601295               3.895490
3    Low_ESG_3  32.822428              10.298077
4    Low_ESG_4  36.167355              10.012221
```

Python Statistical Analysis

	Company	ESG_Score	Financial_Performance
0	High_ESG_0	88.943142	20.065917
1	High_ESG_1	82.182549	19.263989
2	High_ESG_2	80.482487	20.540937
3	High_ESG_3	70.682536	20.596953
4	High_ESG_4	78.613059	22.437716
5	High_ESG_5	78.226205	9.408497
6	High_ESG_6	79.586293	19.229167
7	High_ESG_7	76.864997	5.695552
8	High_ESG_8	79.780909	11.985574
9	High_ESG_9	77.613910	5.427640
45	High_ESG_45	77.057028	14.452728
46	High_ESG_46	75.630589	18.395358
47	High_ESG_47	80.148569	10.722814
48	High_ESG_48	68.758711	13.498970
49	High_ESG_49	78.661191	25.790747

Python Statistical Analysis

	Company	ESG_Score	Financial_Performance
0	Low_ESG_0	34.371429	5.688458
1	Low_ESG_1	23.532317	15.358420
2	Low_ESG_2	29.601295	3.895490
3	Low_ESG_3	32.822428	10.298077
4	Low_ESG_4	36.167355	10.012221
5	Low_ESG_5	30.744932	12.123179
6	Low_ESG_6	27.347089	6.372833
7	Low_ESG_7	26.347367	9.825283
8	Low_ESG_8	33.225310	9.296900
9	Low_ESG_9	31.565302	14.985442
45	Low_ESG_45	33.657329	6.086455
46	Low_ESG_46	32.129834	19.942448
47	Low_ESG_47	29.254931	15.975292
48	Low_ESG_48	34.179220	9.523812
49	Low_ESG_49	32.460595	7.364061

https://tinyurl.com/aintpupython101

Descriptive Statistics

```
#Step 2: Descriptive Statistics
print("High ESG Companies Statistics:")
print(high_esg_df.describe())

print("\nLow ESG Companies Statistics:")
print(low_esg_df.describe())
```

Descriptive Statistics

```
high_esg_df.describe()
```

```
High ESG Companies Statistics:
       ESG_Score  Financial_Performance
count  50.000000              50.000000
mean   78.530366              15.383263
std     5.043011               5.529527
min    67.904584               0.421311
25%    75.443427              11.014420
50%    78.222426              15.758906
75%    81.008308              19.438516
max    89.880554              25.790747
```

```
Low ESG Companies Statistics:
       ESG_Score  Financial_Performance
count  50.000000              50.000000
mean   30.145838               9.524720
std     3.909959               4.894234
min    21.279449              -0.151968
25%    27.353736               6.158049
50%    30.541058               9.357192
75%    32.815212              12.202473
max    39.546384              23.380562
```

Inferential Statistics

```
#Step 3: Inferential Statistics
#Perform a t-test to compare the financial performance between the two groups.


# T-test for Financial Performance
t_stat, p_value =
stats.ttest_ind(high_esg_df['Financial_Perfor
mance'], low_esg_df['Financial_Performance'])
print(f"T-statistic: {t_stat}, P-value: {p_value}")
```

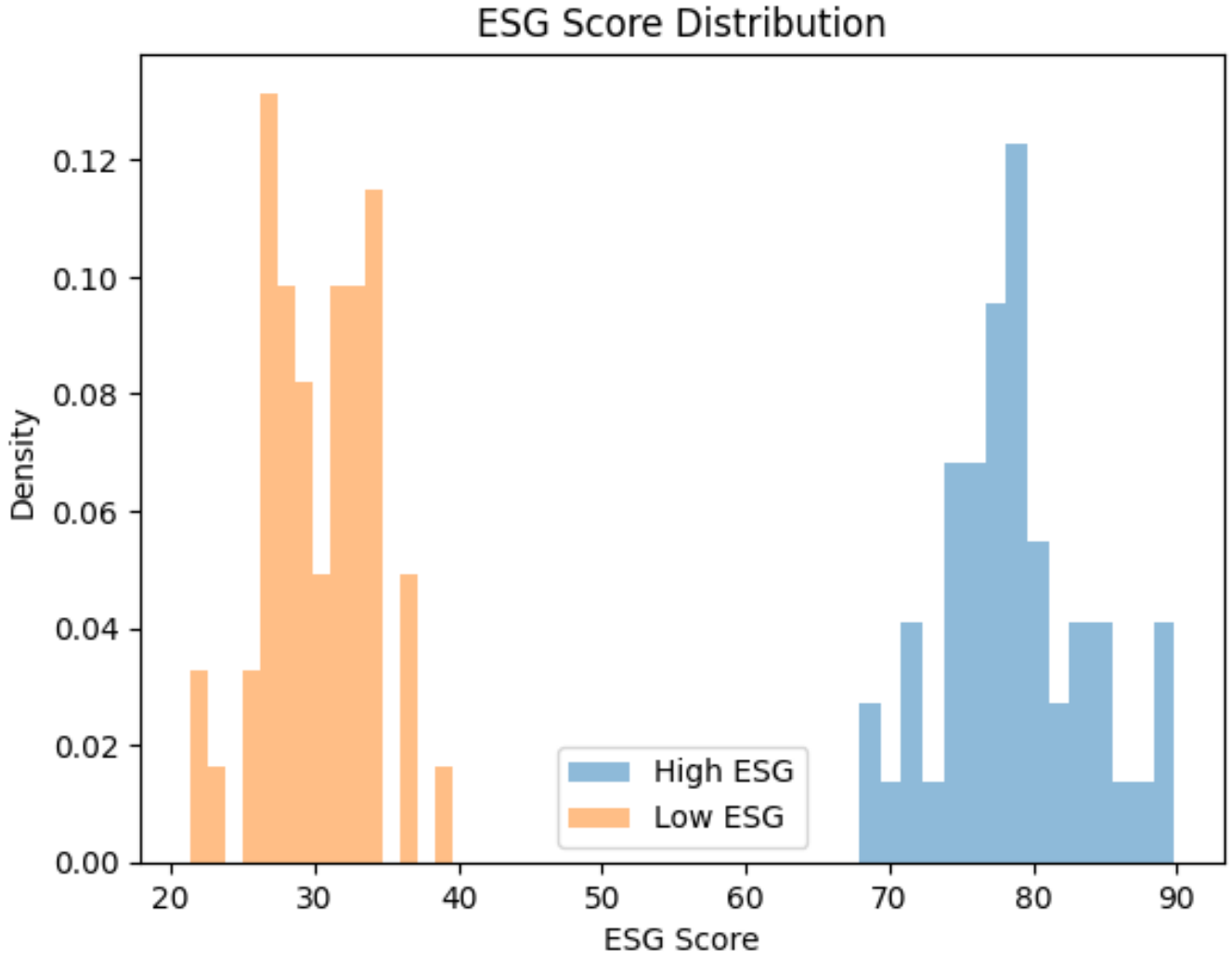
```
T-statistic: 5.609964050849995,
P-value: 1.8760117751911166e-07
```

Python Statistical Analysis

```
#Step 4: Data Visualization
#Visualize the data with histograms and box plots.
#Histograms of ESG Scores
import matplotlib.pyplot as plt
import seaborn as sns
plt.hist(high_esg_df['ESG_Score'], alpha=0.5, label='High
ESG', bins=15, density=True)
plt.hist(low_esg_df['ESG_Score'], alpha=0.5, label='Low
ESG', bins=15, density=True)
plt.title('ESG Score Distribution')
plt.xlabel('ESG Score')
plt.ylabel('Density')
plt.legend()
plt.show()
```

Histograms of ESG Scores

```
plt.hist(high_esg_df['ESG_Score'], alpha=0.5, label='High ESG', bins=15, density=True)
plt.hist(low_esg_df['ESG_Score'], alpha=0.5, label='Low ESG', bins=15, density=True)
```

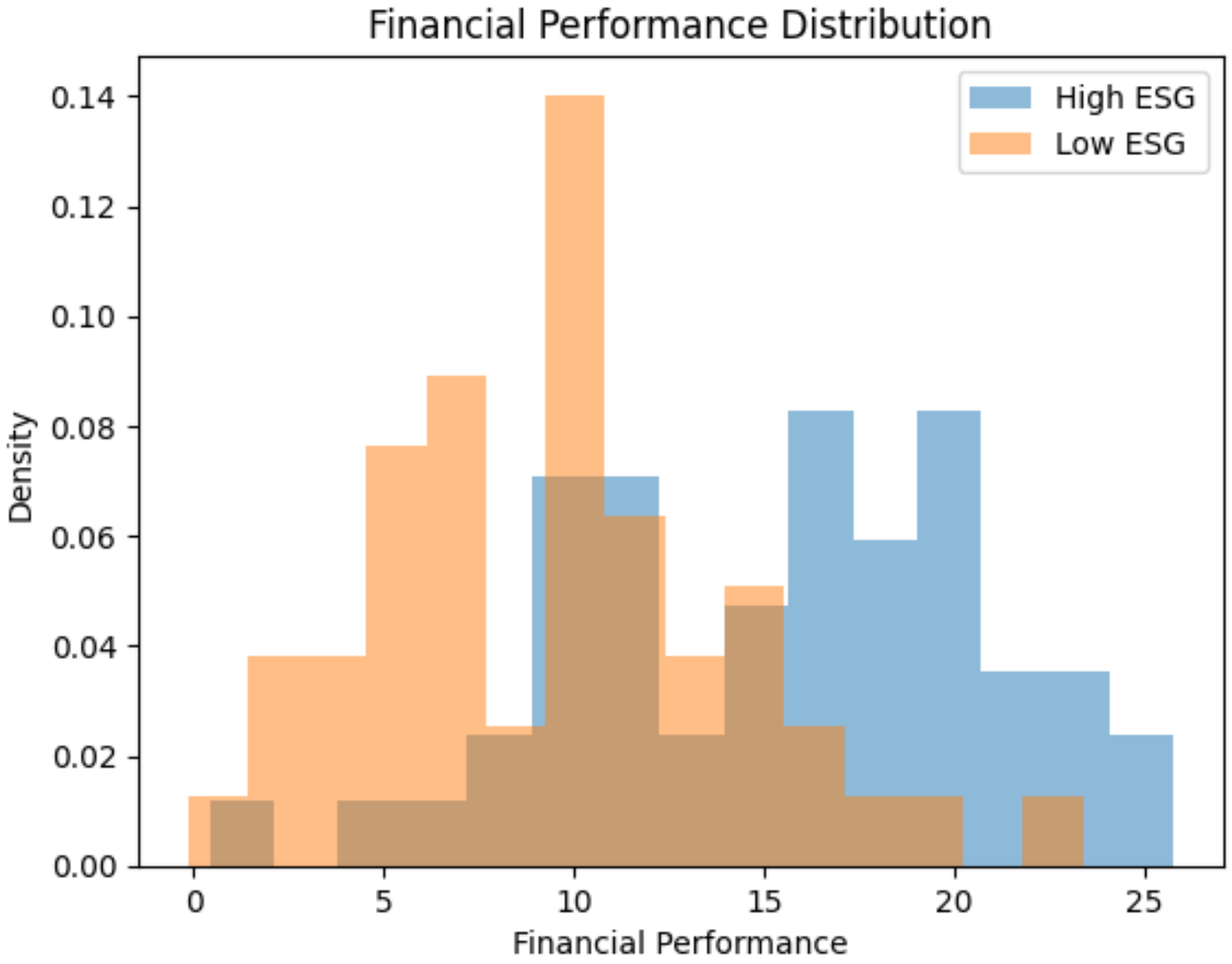


https://tinyurl.com/aintpupython101

Histograms of Financial Performance

```
#Histograms of Financial Performance
plt.hist(high_esg_df['Financial_Performance'],
alpha=0.5, label='High ESG', bins=15, density=True)
plt.hist(low_esg_df['Financial_Performance'],
alpha=0.5, label='Low ESG', bins=15, density=True)
plt.title('Financial Performance Distribution')
plt.xlabel('Financial Performance')
plt.ylabel('Density')
plt.legend()
plt.show()
```

Histograms of Financial Performance



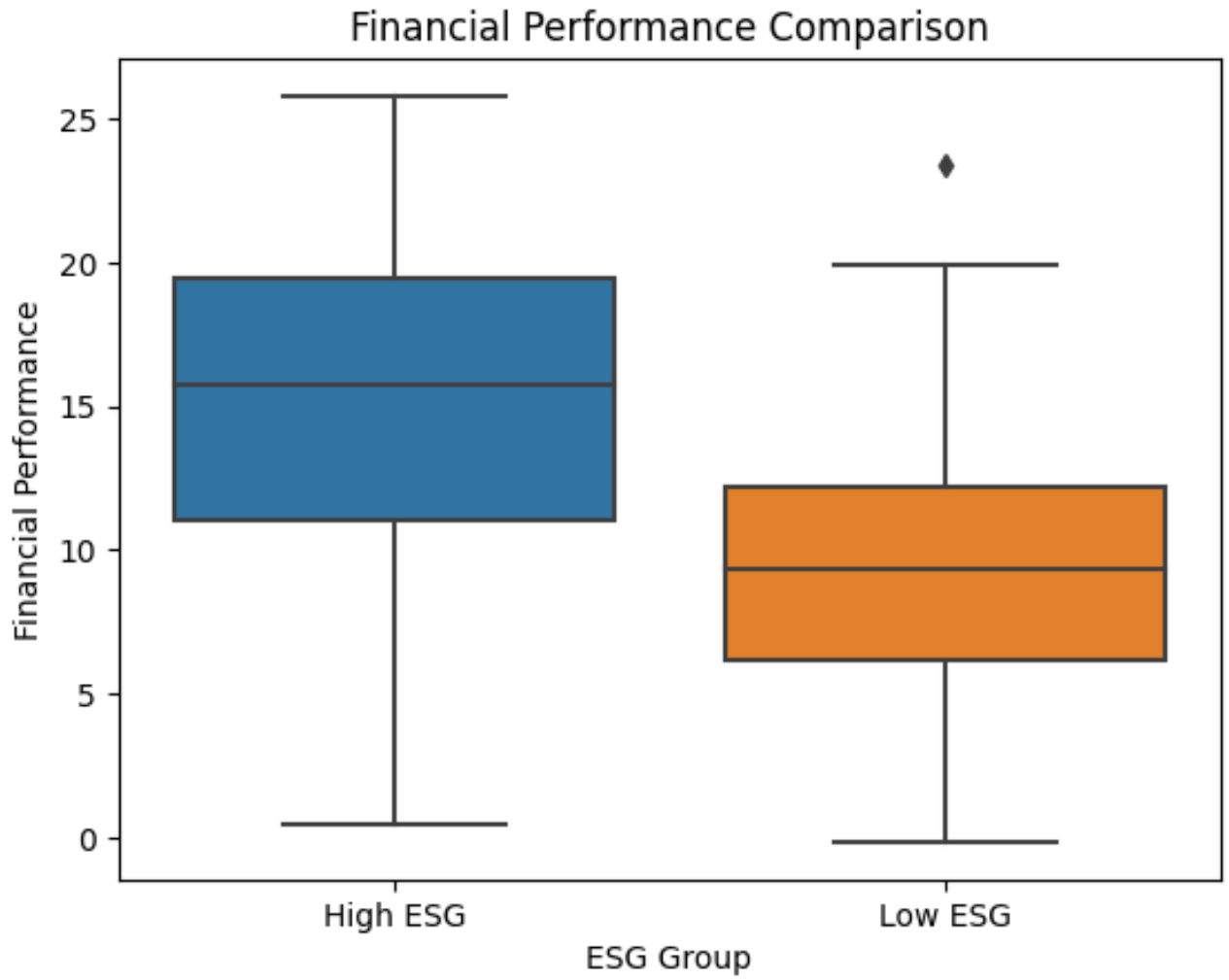
https://tinyurl.com/aintpupython101

Box Plot for Financial Performance

```
# Box Plot for Financial Performance
sns.boxplot(x='Group', y='Financial_Performance',
data=pd.concat([high_esg_df.assign(Group='High
ESG'), low_esg_df.assign(Group='Low ESG')]))
plt.title('Financial Performance Comparison')
plt.xlabel('ESG Group')
plt.ylabel('Financial Performance')
plt.show()
```

Box Plot for Financial Performance

```
sns.boxplot(x='Group', y='Financial_Performance',
data=pd.concat([high_esg_df.assign(Group='High ESG'), low_esg_df.assign(Group='Low ESG')]))
```



https://tinyurl.com/aintpupython101

Python Statistical Analysis

```
import pandas as pd
from scipy import stats

# Calculate basic statistics for each group with formatting
def basic_statistics(df):
    stats_dict = {
        'Mean': f"{df.mean():.4f}",
        'Std': f"{df.std():.4f}",
        'Median': f"{df.median():.4f}",
        'Max': f"{df.max():.4f}",
        'Min': f"{df.min():.4f}",
        'Skewness': f"{df.skew():.4f}",
        'Kurtosis': f"{df.kurtosis():.4f}"
        }
    return pd.Series(stats_dict)
```

Python Statistical Analysis

```
high_esg_stats = basic_statistics(high_esg_df['Financial_Performance'])
low_esg_stats = basic_statistics(low_esg_df['Financial_Performance'])

# Perform t-test
t_stat, p_value = stats.ttest_ind(high_esg_df['Financial_Performance'],
low_esg_df['Financial_Performance'])

# Determine significance level
def significance(p_value):
    if p_value < 0.001:
        return '***'
    elif p_value < 0.01:
        return '**'
    elif p_value < 0.05:
        return '*'
    else:
        return ''
```

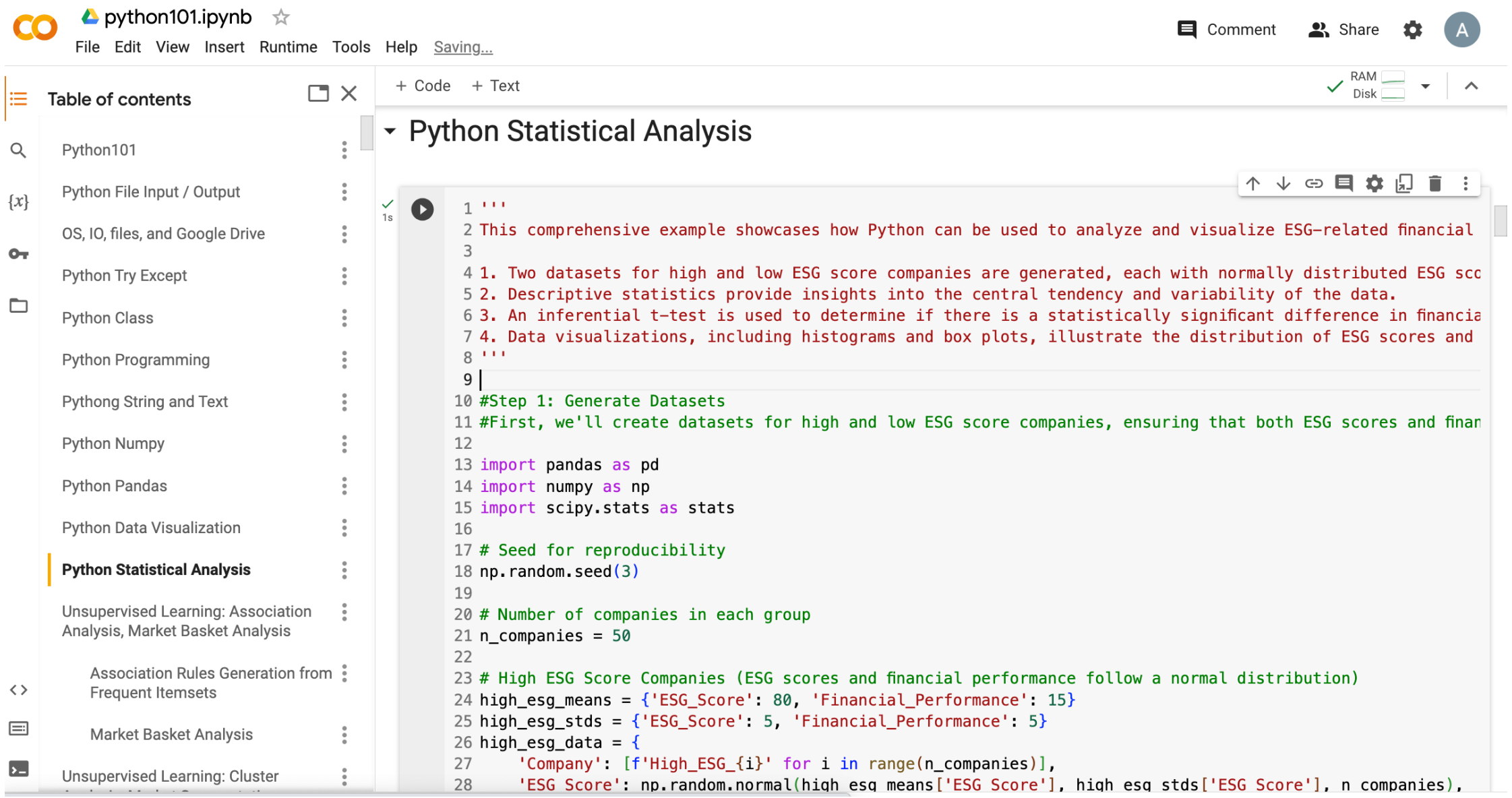
Python Statistical Analysis

```
# Create table
table_data = {
    'Statistic': ['Mean', 'Std', 'Median', 'Max', 'Min',
    'Skewness', 'Kurtosis', 'T-statistic', 'P-value'],
    'High ESG': high_esg_stats.tolist() + [f"{t_stat:.4f}",
    f"{p_value:.4f}"],
    'Low ESG': low_esg_stats.tolist() + ['', ''],
    'Significance': [''] * 7 + ['', significance(p_value)]
    }

table_df = pd.DataFrame(table_data)

# Printing the table
print(table_df)
```

Python Statistical Analysis

	Statistic	High ESG	Low ESG	Significance
0	Mean	15.3833	9.5247	
1	Std	5.5295	4.8942	
2	Median	15.7589	9.3572	
3	Max	25.7907	23.3806	
4	Min	0.4213	-0.1520	
5	Skewness	-0.3003	0.5287	
6	Kurtosis	-0.2029	0.3327	
7	T-statistic	5.6100		
8	P-value	0.0000		***

Python Statistical Analysis

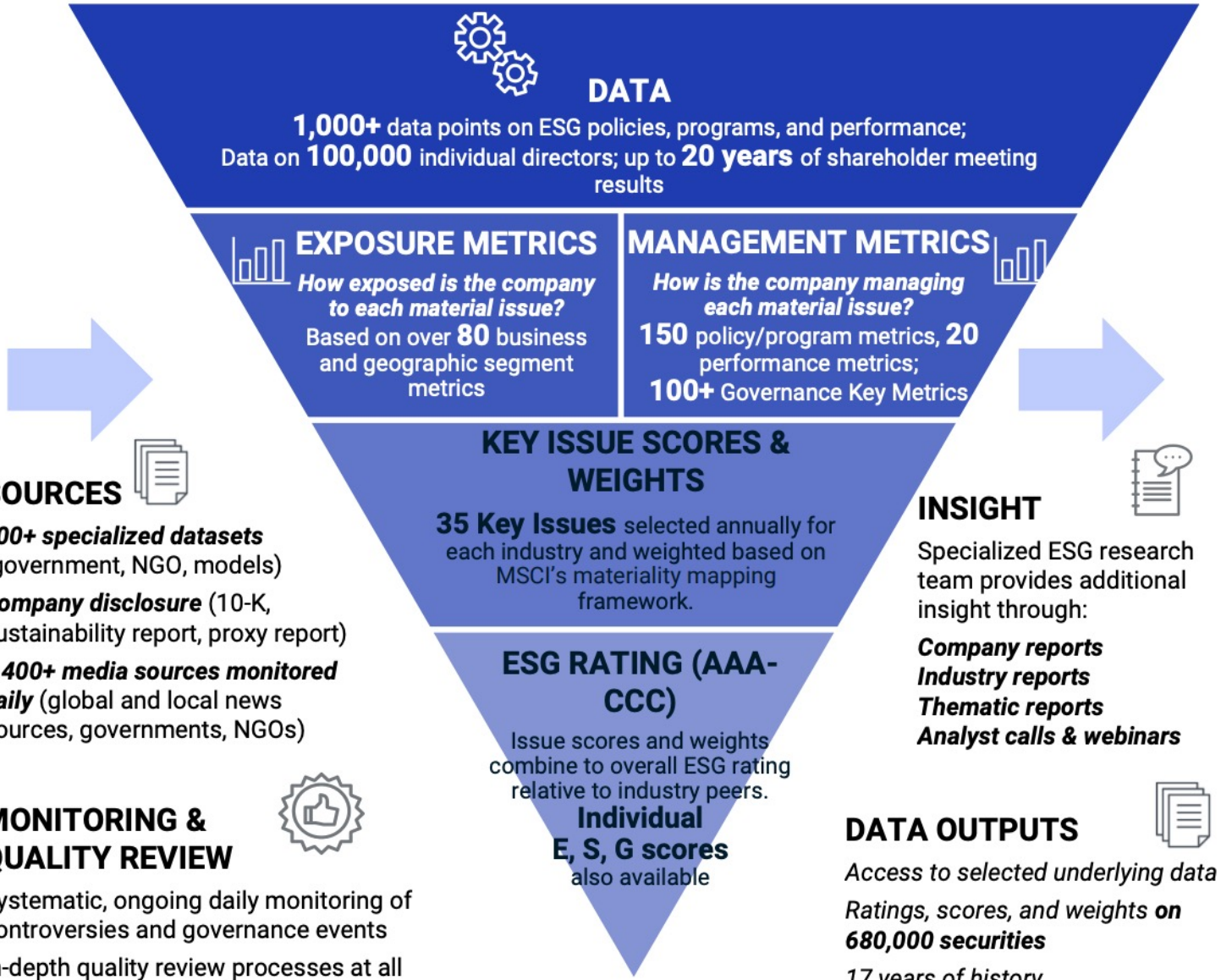


https://tinyurl.com/aintpupython101

ESG Indexes

- MSCI ESG Index
- Dow Jones Sustainability Indices (DJSI)
- FTSE ESG Index

MSCI ESG Rating Framework



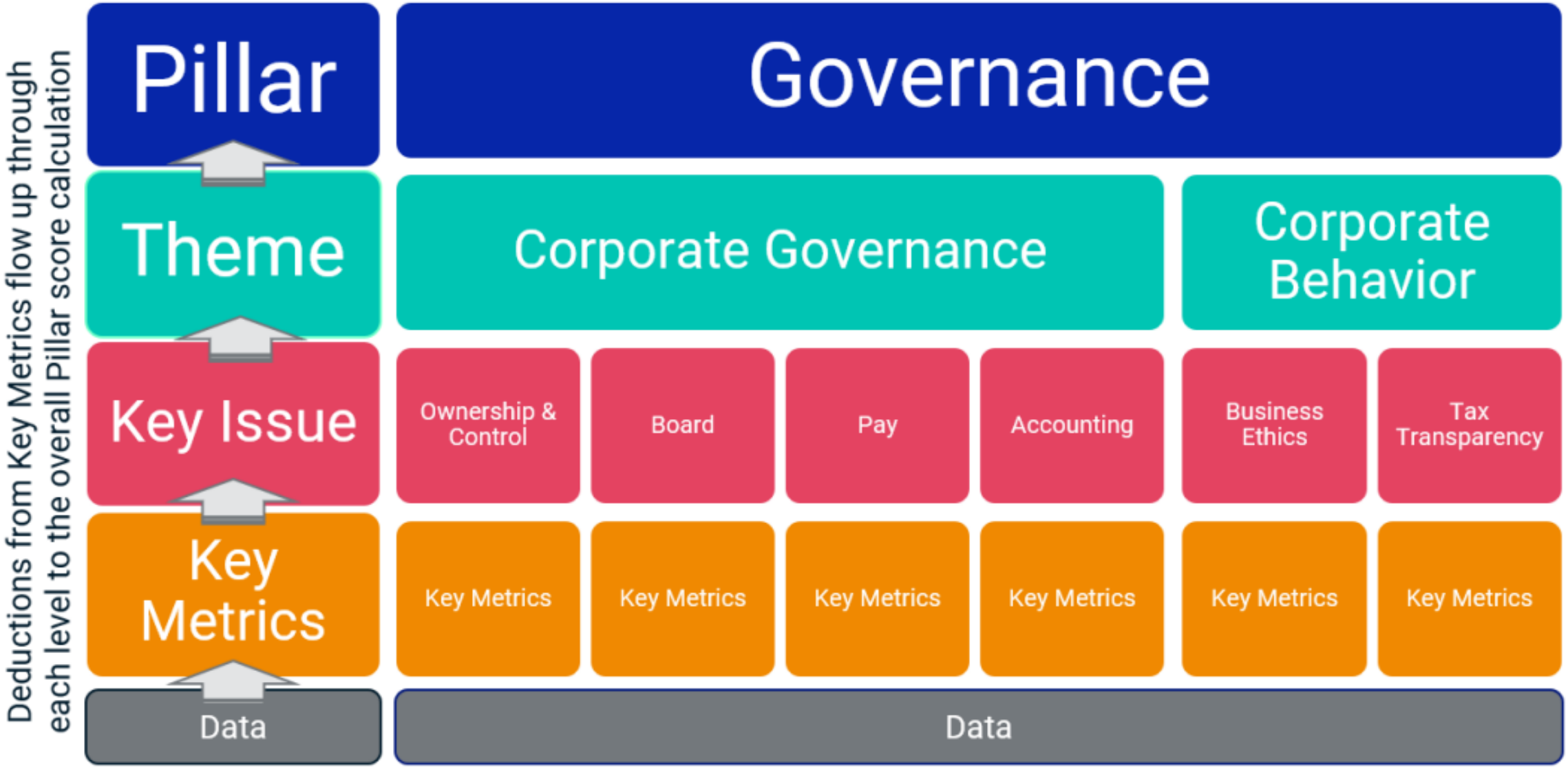
Source: https://www.msci.com/documents/1296102/21901542/ESG-Ratings-Methodology-Exec-Summary.pdf

MSCI ESG Key Issue Hierarchy

3 Pillars	10 Themes	35 ESG Key Issues	
Environment	**Climate Change**	Carbon Emissions	Financing Environmental Impact
		Product Carbon Footprint	Climate Change Vulnerability
	Natural Capital	Water Stress	Raw Material Sourcing
		Biodiversity & Land Use	
	Pollution & Waste	Toxic Emissions & Waste	Electronic Waste
		Packaging Material & Waste	
	Environmental Opportunities	Opportunities in Clean Tech	Opportunities in Renewable Energy
		Opportunities in Green Building	
Social	**Human Capital**	Labor Management	Human Capital Development
		Health & Safety	Supply Chain Labor Standards
	Product Liability	Product Safety & Quality	Privacy & Data Security
		Chemical Safety	Responsible Investment
		Consumer Financial Protection	Health & Demographic Risk
	Stakeholder Opposition	Controversial Sourcing	
		Community Relations	
	Social Opportunities	Access to Communications	Access to Health Care
		Access to Finance	Opportunities in Nutrition & Health
Governance	**Corporate Governance**	Ownership & Control	Pay
		Board	Accounting
	Corporate Behavior	Business Ethics	
		Tax Transparency	

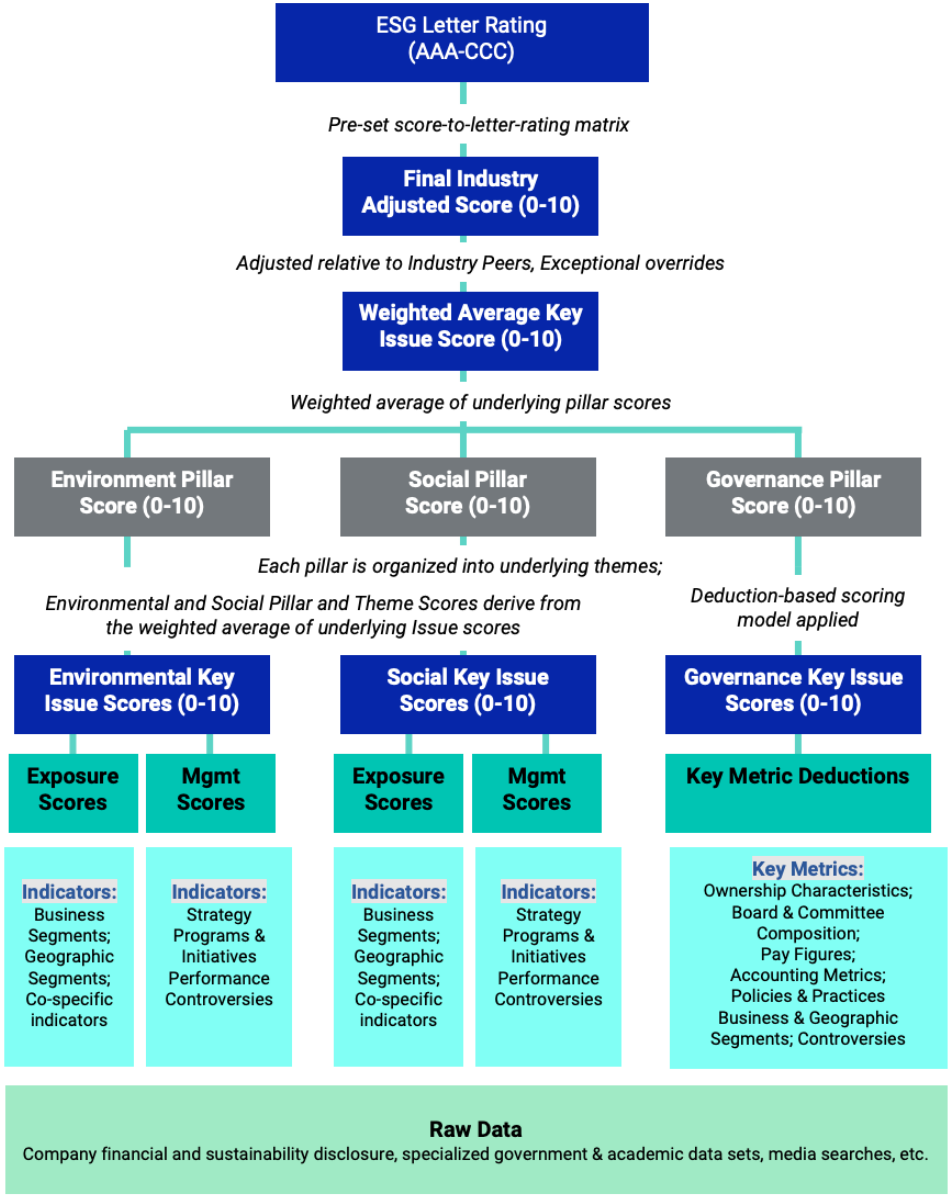
Source: https://www.msci.com/documents/1296102/21901542/ESG-Ratings-Methodology-Exec-Summary.pdf

MSCI Governance Model Structure



Source: https://www.msci.com/documents/1296102/21901542/ESG-Ratings-Methodology-Exec-Summary.pdf

MSCI Hierarchy of ESG Scores



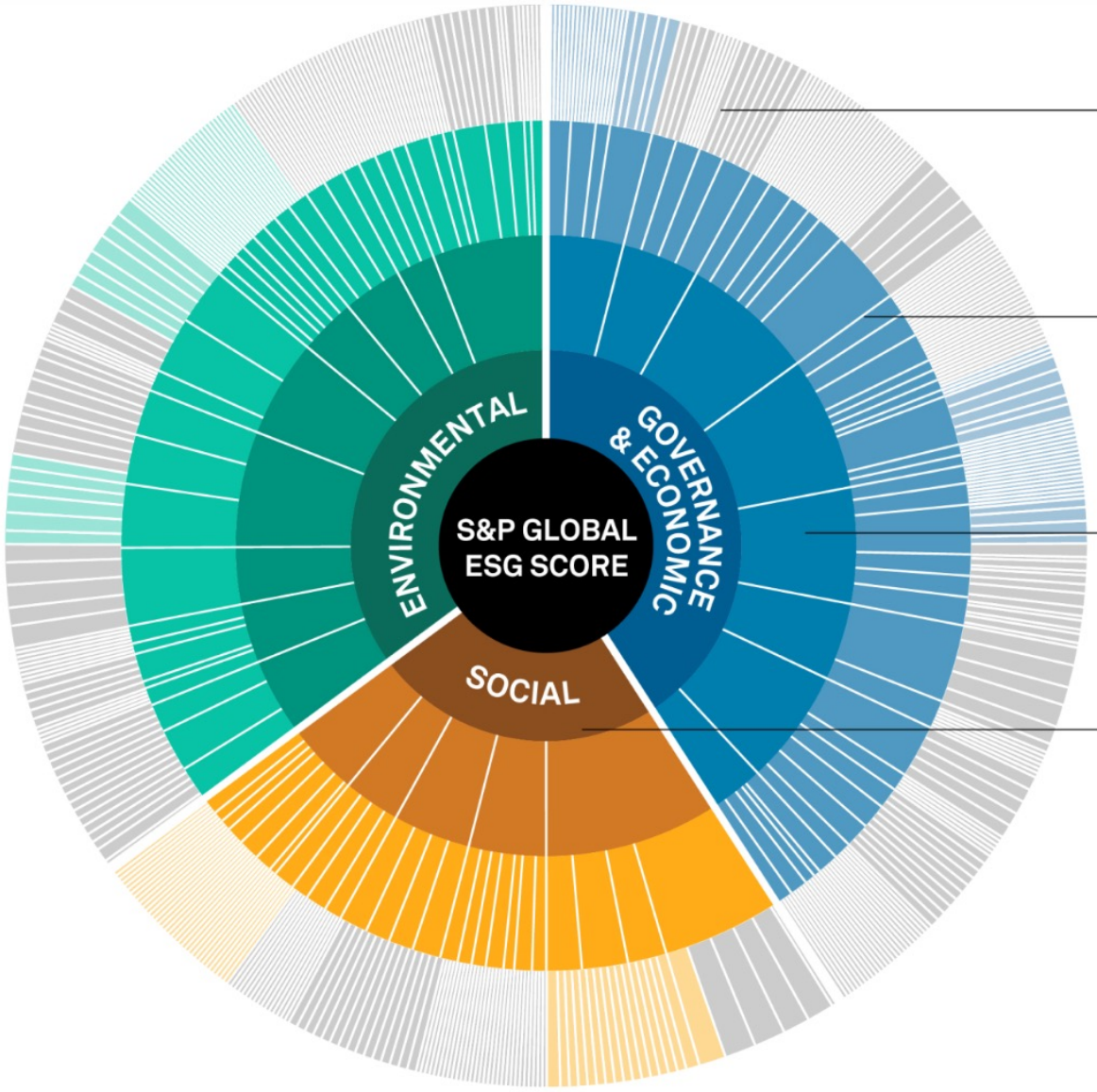
Source: https://www.msci.com/documents/1296102/21901542/ESG-Ratings-Methodology-Exec-Summary.pdf

DJSI S&P Global ESG Score

8,000
Companies

90%
Global market capitalization

340,000+
Current Research Universe and Active Securities



Approx.
1,000
Datapoints

Assessed values, text, checkboxes, documents
Sources: Web-based questionnaire and company documents

130+
Questions

Weighted data point scores
Up to 50% industry-specific

Ave.
30+
Criteria scores

Weighted question scores
61 industry specific approaches, with tailored questions, criteria and related weightings

3
Dimension scores

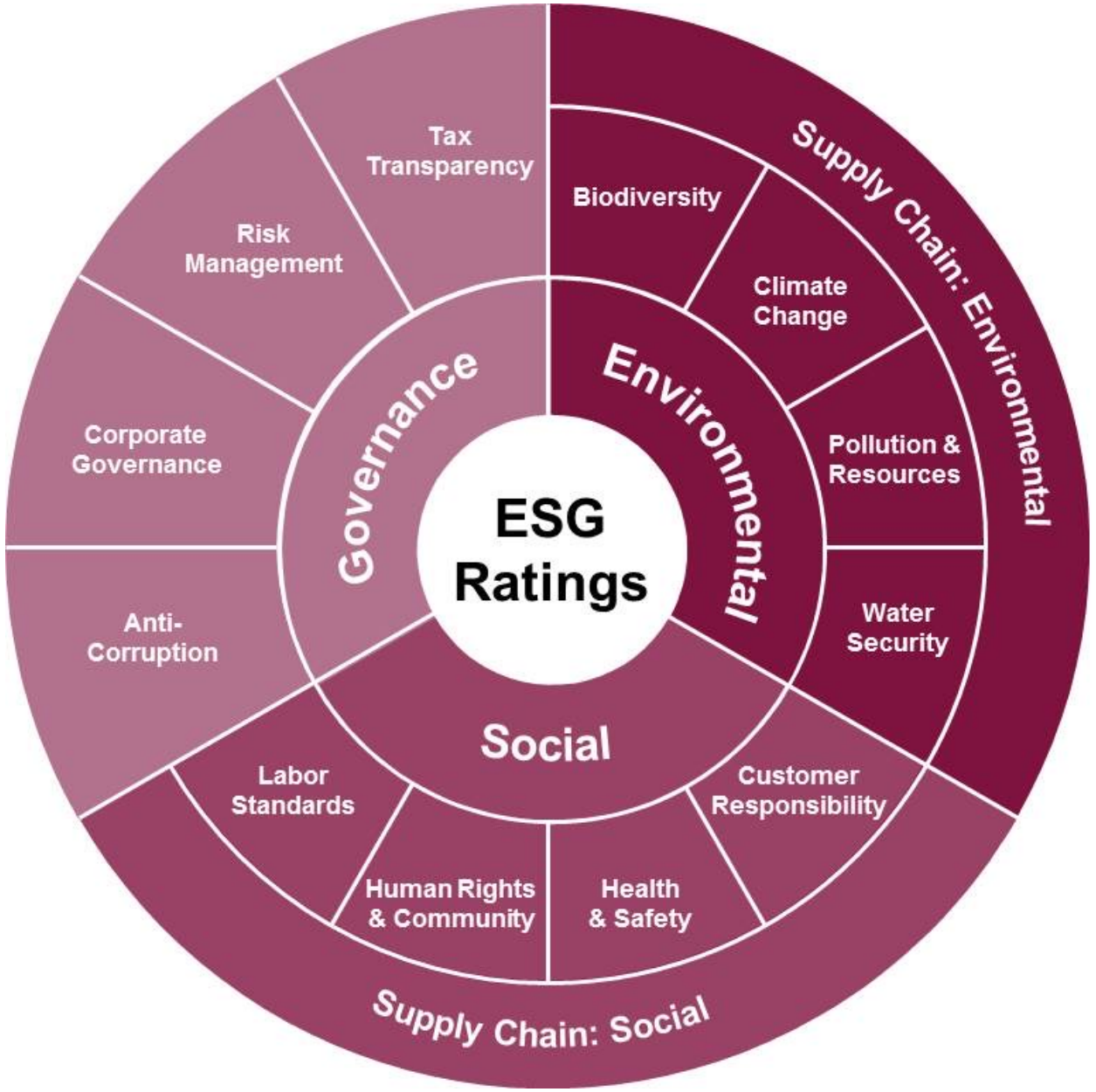
Weighted criteria scores
Adjusted for corporate ESG controversies where applicable

1
S&P Global ESG Score

Sum of weighted dimension scores

Source: DJSI, S&P Global Corporate Sustainability Assessment (CSA), https://www.spglobal.com/esg/solutions/data-intelligence-esg-scores

FTSE Russell ESG Ratings



Source: https://www.ftserussell.com/data/sustainability-and-esg-data/esg-ratings

SUSTAINALYTICS
a Morningstar company

Sustainalytics ESG Risk Ratings

Analyst-based approach

Sustainalytics' ESG Risk Ratings measure a company's exposure to industry-specific material ESG risks and how well a company is managing those risks.

Negligible	Low	Medium	High	Severe
0 - 10	10 - 20	20 - 30	30 - 40	40+

Source: https://www.sustainalytics.com/esg-data

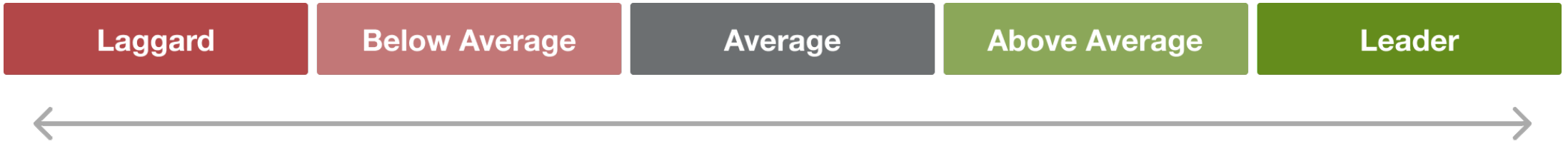
TruValue Labs

FACTSET

Truvalue ESG Ranks

Machine-based approach

- Truvalue Labs applies AI to analyze over 100,000 sources and uncover ESG risks and opportunities hidden in unstructured text.
- The ESG Ranks data service produces an overall company rank based on industry percentile leveraging the 26 ESG categories defined by the Sustainability Accounting Standards Board (SASB).
- The data feed covers 20,000+ companies with more than 13 years of history.



Source: :https://developer.truvaluelabs.com/data/esg-ranks

Analyst-driven vs. AI-driven ESG

Analyst-driven ESG research

Derives ratings in a structured data model

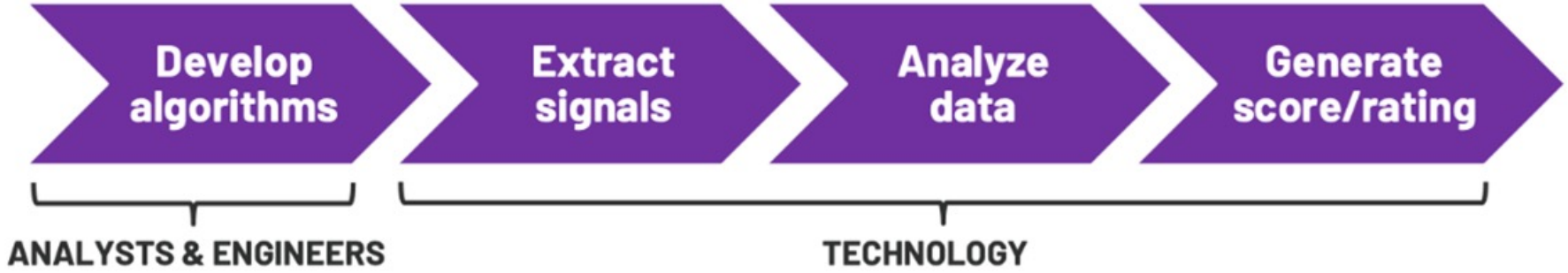
Sustainalytics

Analyst role at the end of the process allows subjectivity to color results

AI-driven ESG research

Derives signals from unstructured data

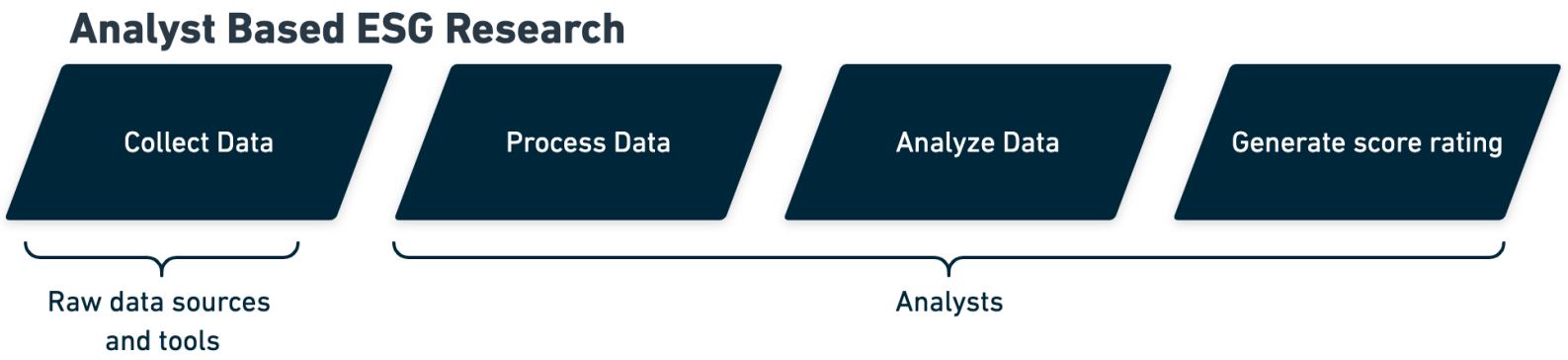
Truvalue Labs



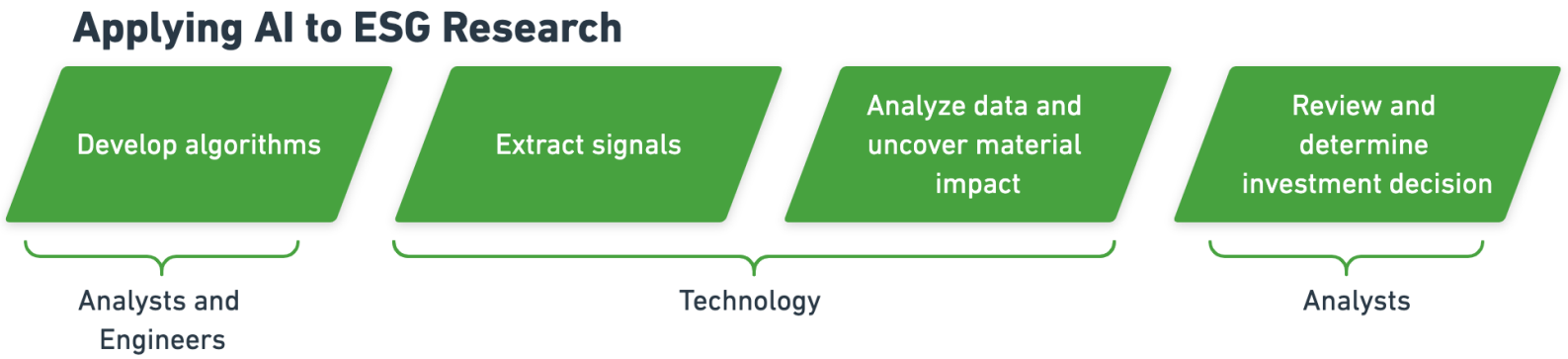
Analyst expertise at the beginning of the process produces consistent results

Source: Mark Tulay (2020), Man vs. machine: A tale of two sustainability ratings systems, GreenBiz, https://www.greenbiz.com/article/man-vs-machine-tale-two-sustainability-ratings-systems

Analyst based ESG Research

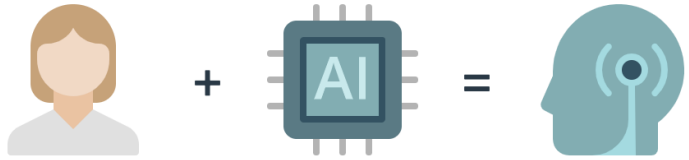


AI based ESG Research



It would take an analyst over 5 years to do what our AI can in 1 week

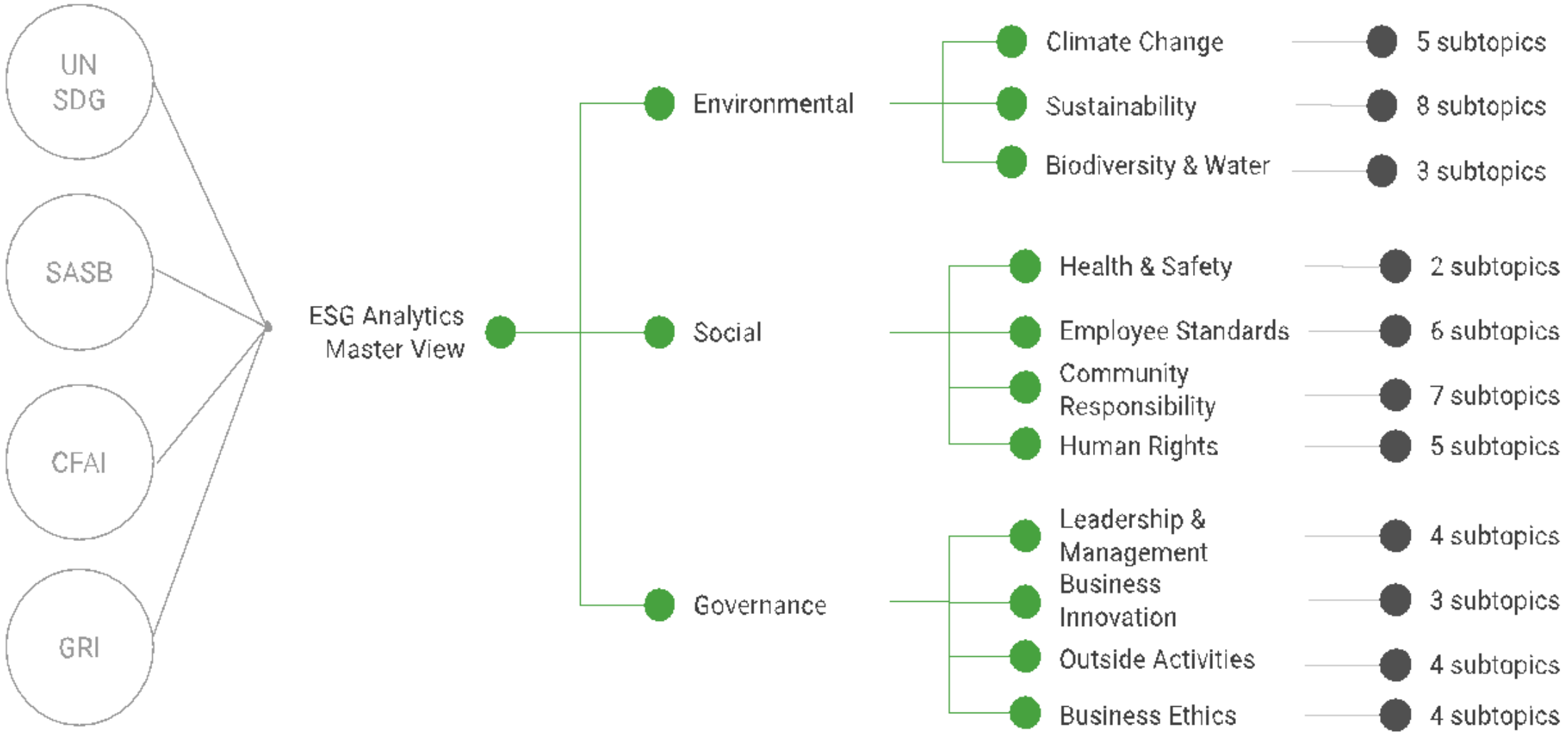
Combining analysts with AI creates gives you the full picture



ESG ANALYTICS
Invest where it matters.

Source: https://www.esganalytics.io/insights/how-data-is-accelerating-esg

ESG Analytics: NLP Taxonomy

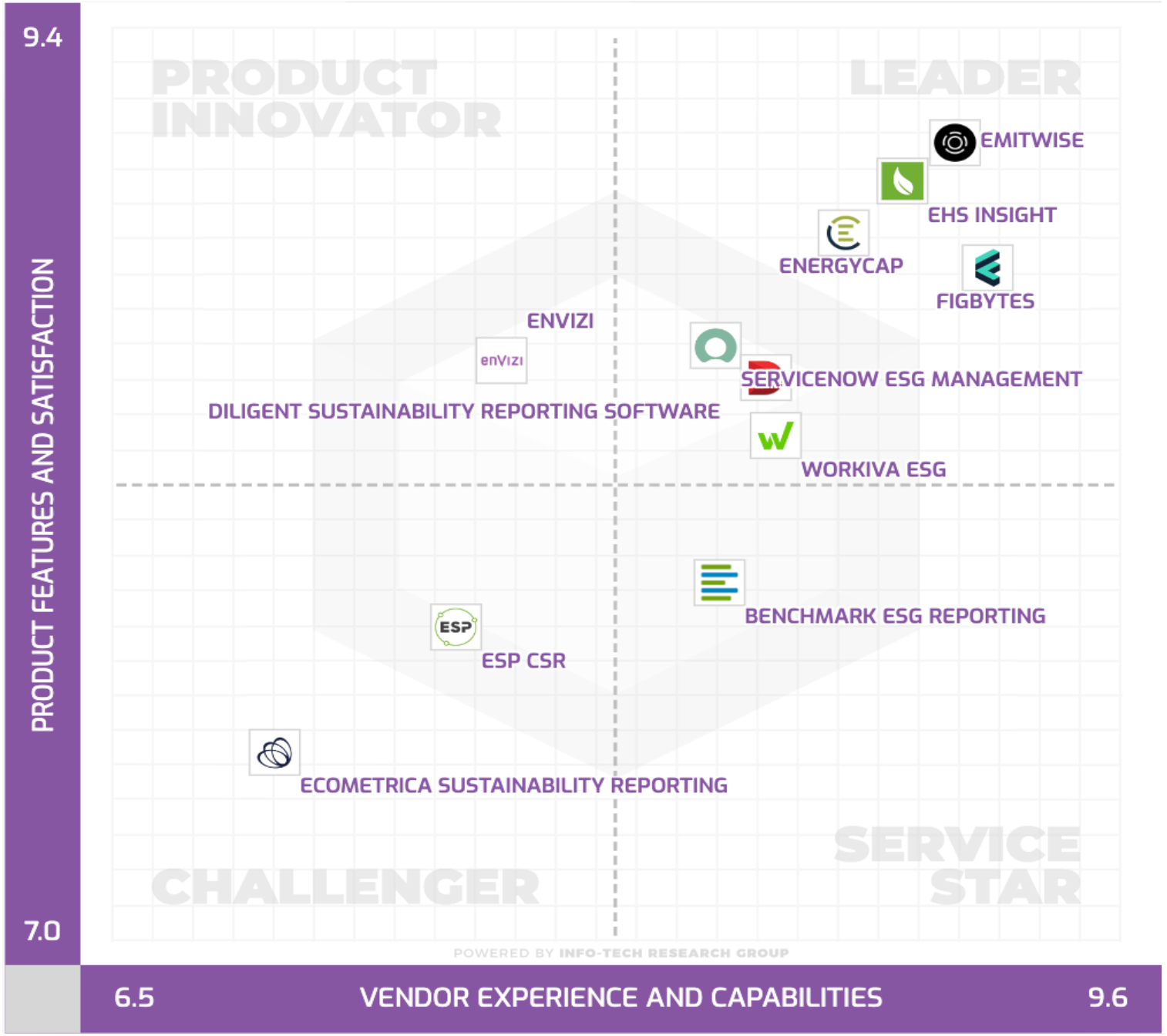


ESG ANALYTICS

Source: https://www.esganalytics.io/insights/how-data-is-accelerating-esg

Top ESG Reporting Software

Environmental, Social and Governance (ESG) Reporting software or **Sustainability software** helps organizations manage their operational data, evaluate their impact on the environment and provide reporting to perform audits.



Source: https://www.softwarereviews.com/categories/environmental-social-and-governance-reporting

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

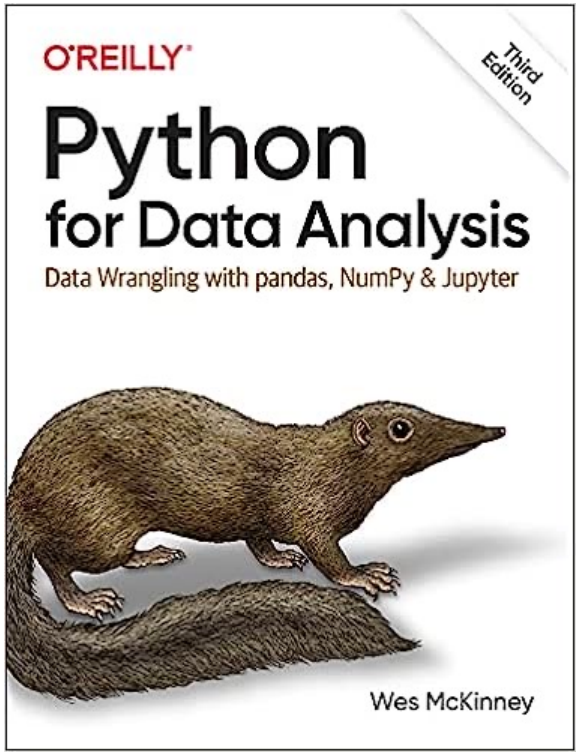
3rd-edition | 3 branches | 0 tags | Go to file | Code

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))
```

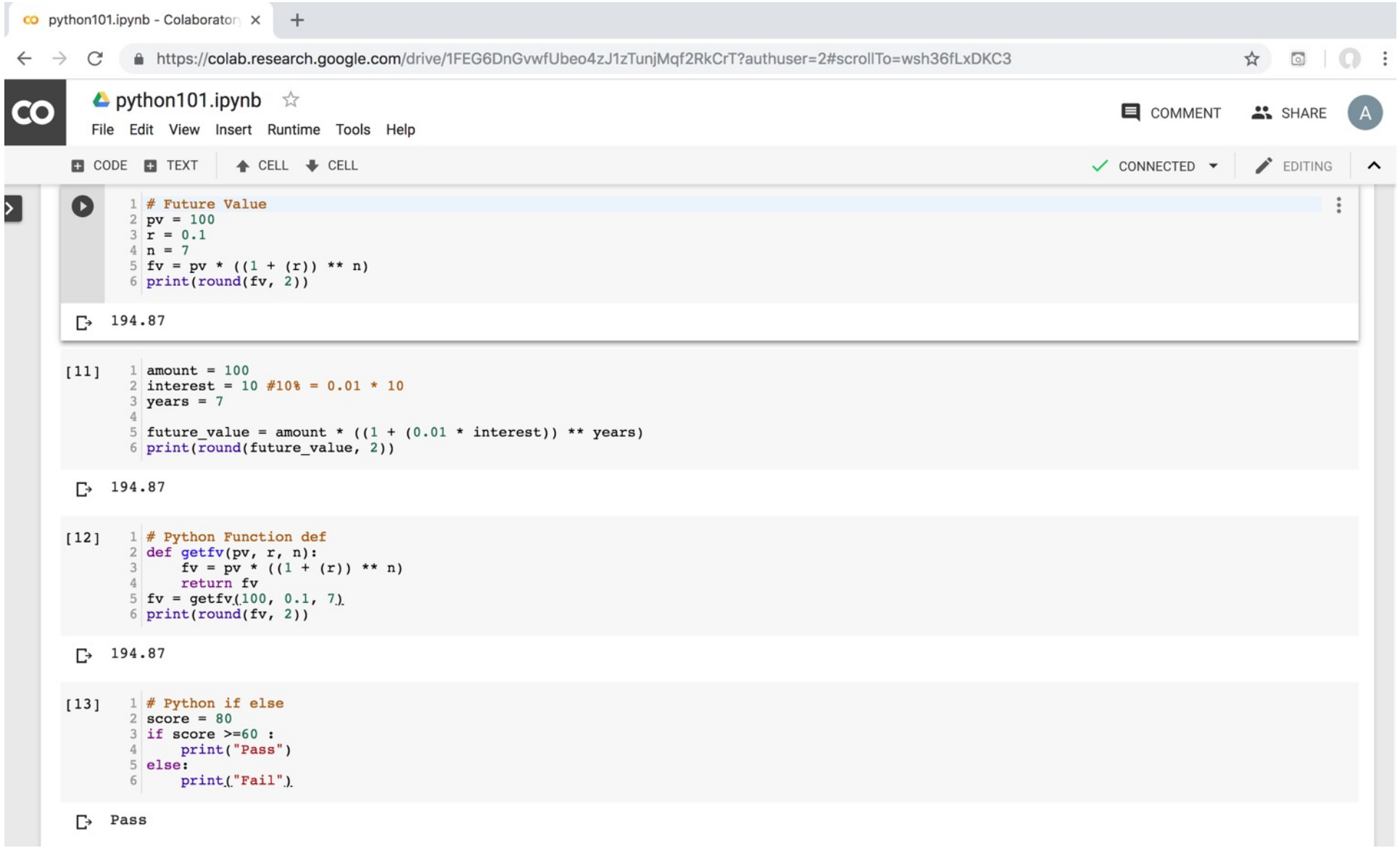
In [3]:
```
%timeit my_arr2 = my_arr * 2
%timeit my_list2 = [x * 2 for x in my_list]
```

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)

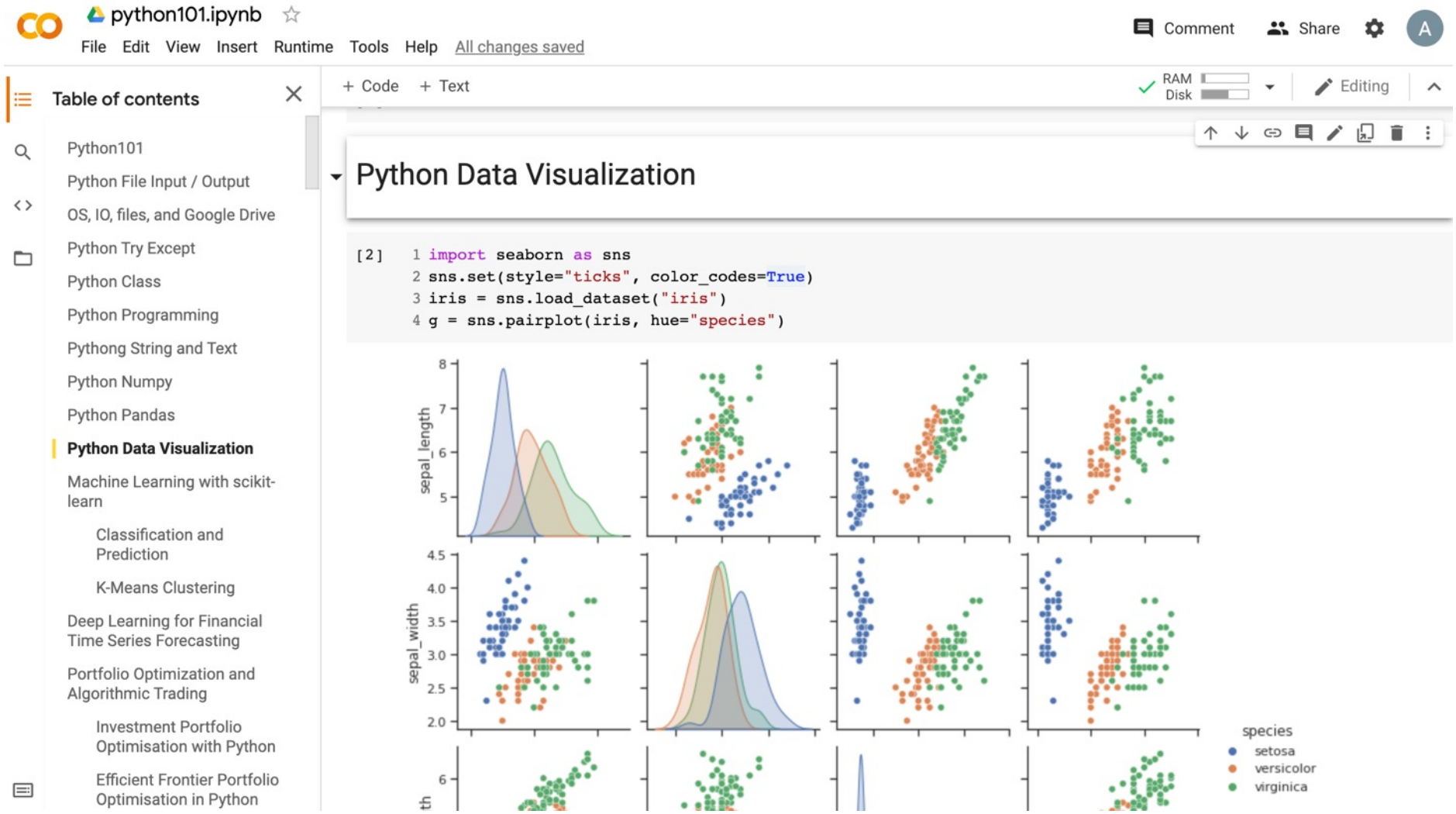
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



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

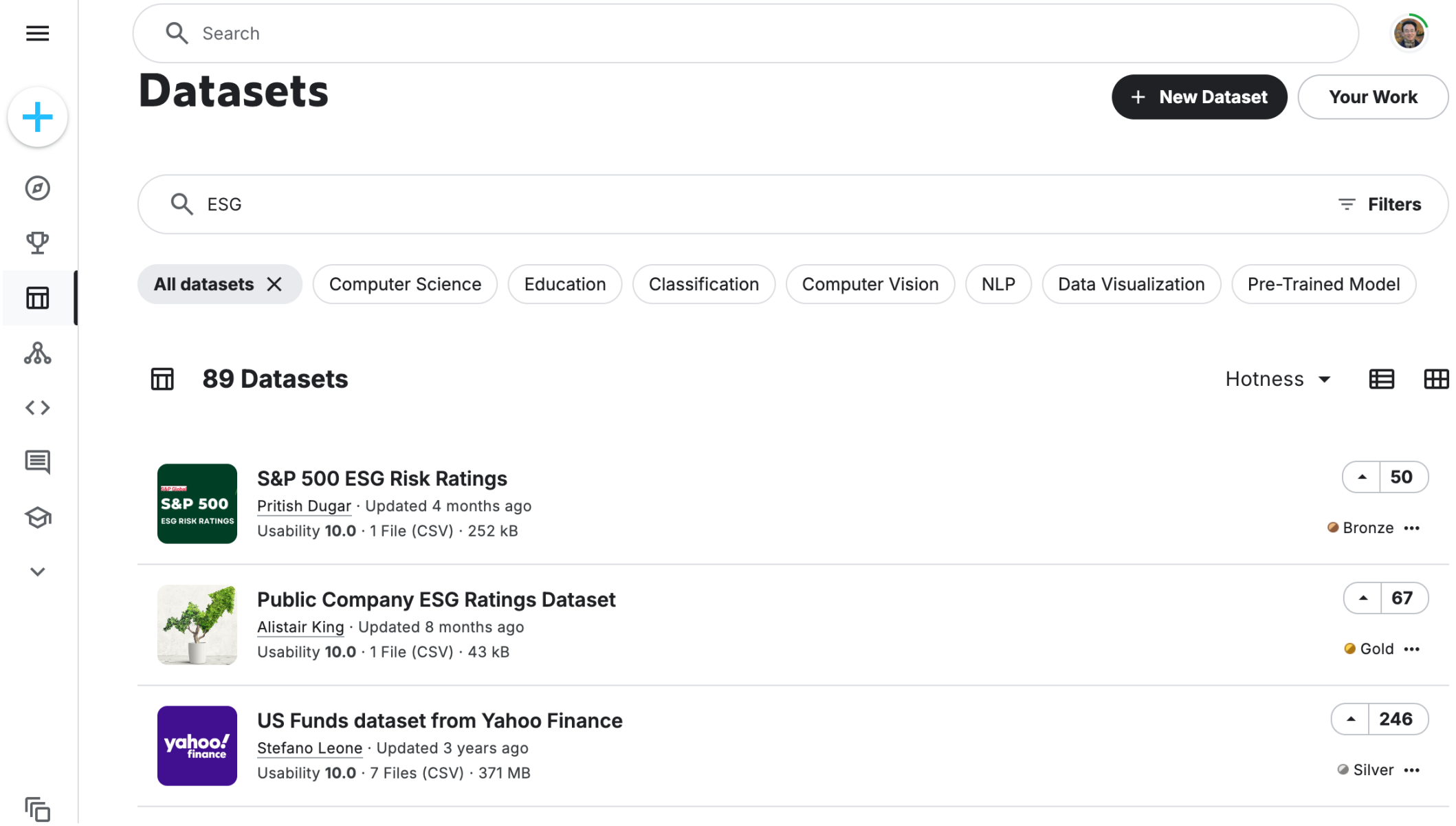
931 papers with code

See all 664 tasks

https://paperswithcode.com/sota

Kaggle Datasets for Data Science

Kaggle Datasets for Machine Learning



Source: https://www.kaggle.com/datasets?search=ESG

Kaggle Datasets for Machine Learning

Search

Datasets

+ New Dataset | Your Work

credit card | Filters

All datasets | Computer Science | Education | Classification | Computer Vision | NLP | Data Visualization | Pre-Trained Model

1,125 Datasets | Most Votes

Credit Card Fraud Detection
Machine Learning Group - ULB · Updated 7 years ago
Usability 8.5 · 1 File (CSV) · 69 MB
11606 | Gold

Supermarket sales
Aung Pyae · Updated 5 years ago
Usability 8.8 · 1 File (CSV) · 37 kB
2580 | Gold

Credit Card customers
Sakshi Goyal · Updated 4 years ago
Usability **10.0** · 1 File (CSV) · 388 kB
2256 | Gold

Source: https://www.kaggle.com/datasets?search=credit+card&sort=votes

Kaggle Datasets: Credit Card Fraud Detection

Search

MACHINE LEARNING GROUP - ULB AND 1 COLLABORATOR · UPDATED 7 YEARS AGO

11606 | New Notebook | Download

Credit Card Fraud Detection

Anonymized credit card transactions labeled as fraudulent or genuine

Data Card | Code (4966) | Discussion (107) | Suggestions (0)

About Dataset

Context

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

Content

The dataset contains transactions made by credit cards in September 2013 by European cardholders.

Usability
8.53

License
Database: Open Database, Cont...

Expected update frequency
Not specified

Tags

Source: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Kaggle Code: Credit Card Fraud Detection

Credit Card Fraud Detection

11606 | New Notebook | Download

Data Card | Code (4966) | Discussion (107) | Suggestions (0)

All | Your Work | Shared With You | Bookmarks | Most Votes

Credit Fraud || Dealing with Imbalanced Datasets — 5453

Updated 5y ago

657 comments · Credit Card Fraud Detection

Gold

Outlier!!! The Silent Killer — 1070

Updated 3y ago

138 comments · Titanic - Machine Learning from Disaster +16

Gold

In depth skewed data classif. (93% recall acc now) — 796

Updated 8y ago

125 comments · Credit Card Fraud Detection

Gold

Credit Card Fraud Detection Predictive Models — 489

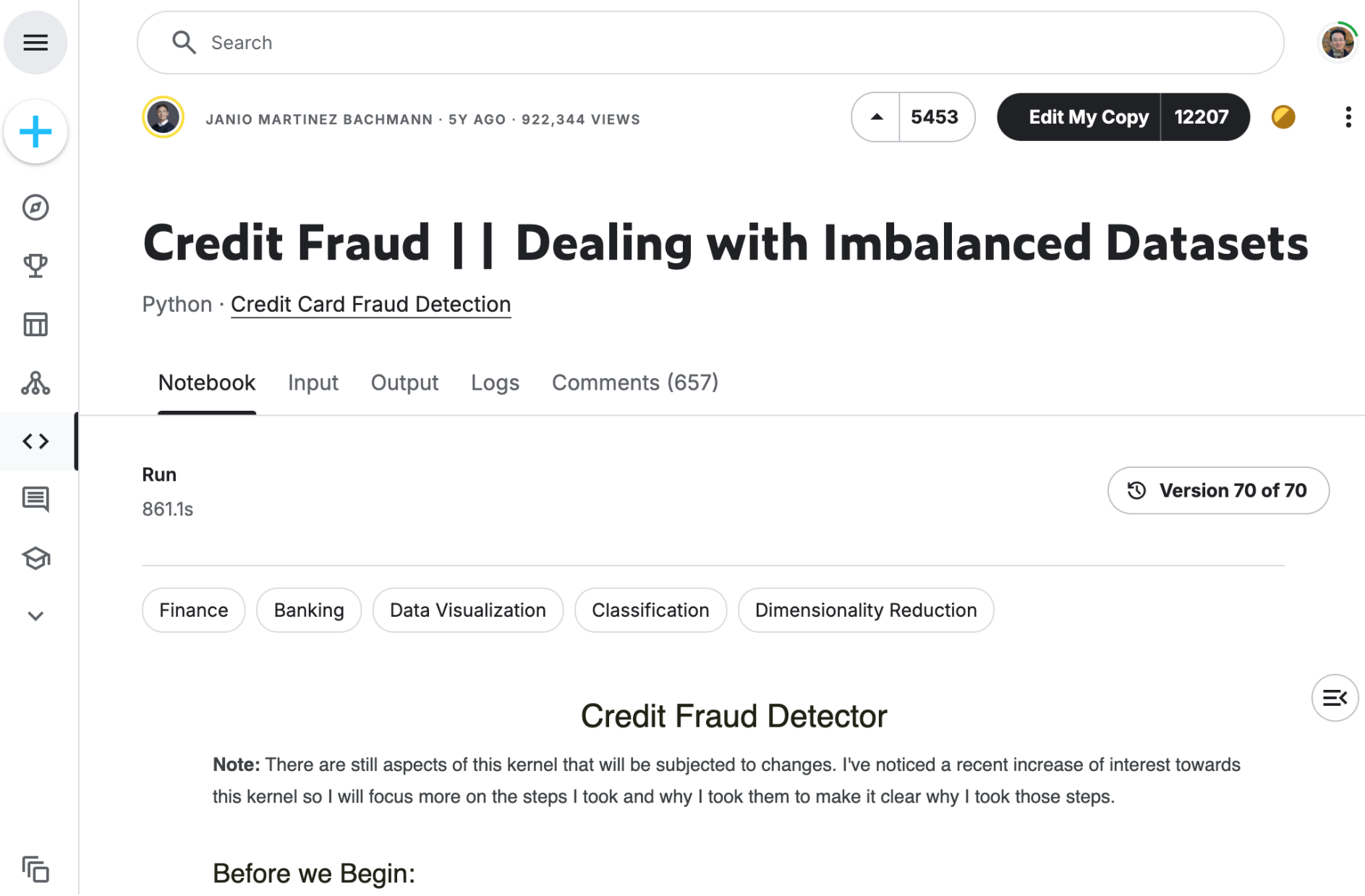
Updated 4y ago

41 comments · Credit Card Fraud Detection

Gold

Source: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/code?datasetId=310&sortBy=voteCount

Kaggle Code: Credit Card Fraud Detection



Source: https://www.kaggle.com/code/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets

Summary

- Obtaining Data From the Web with Python
  - Requests
  - BeautifulSoup
  - Pandas
- Statistical Analysis with Python
  - Descriptive Statistics
  - Inferential Statistics

References

- Wes McKinney (2022), "Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter", 3rd Edition, O'Reilly Media.
- Aurélien Géron (2023), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 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.
- Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson.
- Denis Rothman (2024), Transformers for Natural Language Processing and Computer Vision - Third Edition: Explore Generative AI and Large Language Models with Hugging Face, ChatGPT, GPT-4V, and DALL-E 3, 3rd ed. Edition, Packt Publishing
- Ben Auffarth (2023), Generative AI with LangChain: Build large language model (LLM) apps with Python, ChatGPT and other LLMs, Packt Publishing.
- Varun Grover, Roger HL Chiang, Ting-Peng Liang, and Dongsong Zhang (2018), "Creating Strategic Business Value from Big Data Analytics: A Research Framework", Journal of Management Information Systems, 35, no. 2, pp. 388-423.
- Junliang Wang, Chuqiao Xu, Jie Zhang, and Ray Zhong (2022). "Big data analytics for intelligent manufacturing systems: A review." Journal of Manufacturing Systems 62 (2022): 738-752.
- Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
- 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/
- W3Schools Python, https://www.w3schools.com/python/
- Learn Python, https://www.learnpython.org/
- Google's Python Class, https://developers.google.com/edu/python
- Harvard University (2024), CS50x 2024 - Lecture 6 - Python, http://www.youtube.com/watch?v=EHi0RDZ31VA
- Min-Yuh Day (2024), Python 101, https://tinyurl.com/aintpupython101