

Data Analytics and Visualization with Python

1121PAA08

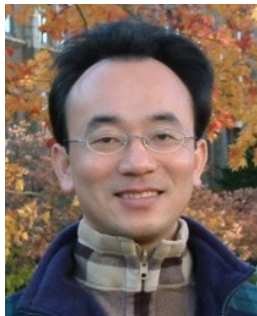
ACC2, NTPU (M5265) (Fall 2023)

Wed 6, 7, 8, (14:10-17:00) (9:10-12:00) (B3F10)

Min-Yuh Day, Ph.D,
Associate Professor

Institute of Information Management, National Taipei University

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



Syllabus

Week Date Subject/Topics

1 2023/09/13 Introduction to Python for Accounting Applications

2 2023/09/20 Python Programming and Data Science

3 2023/09/27 Foundations of Python Programming

4 2023/10/04 Data Structures

5 2023/10/11 Control Logic and Loops

6 2023/10/18 Functions and Modules

7 2023/10/25 Files and Exception Handling

8 2023/11/01 Midterm Project Report

Syllabus

Week Date Subject/Topics

9 2023/11/08 Data Analytics and Visualization with Python

10 2023/11/15 Obtaining Data From the Web with Python

11 2023/11/22 Statistical Analysis with Python

12 2023/11/29 Machine Learning with Python

**13 2023/12/06 Text Analytics with Python and
 Large Language Models (LLMs)**

14 2023/12/13 Applications of Accounting Data Analytics with Python

15 2023/12/20 Applications of ESG Data Analytics with Python

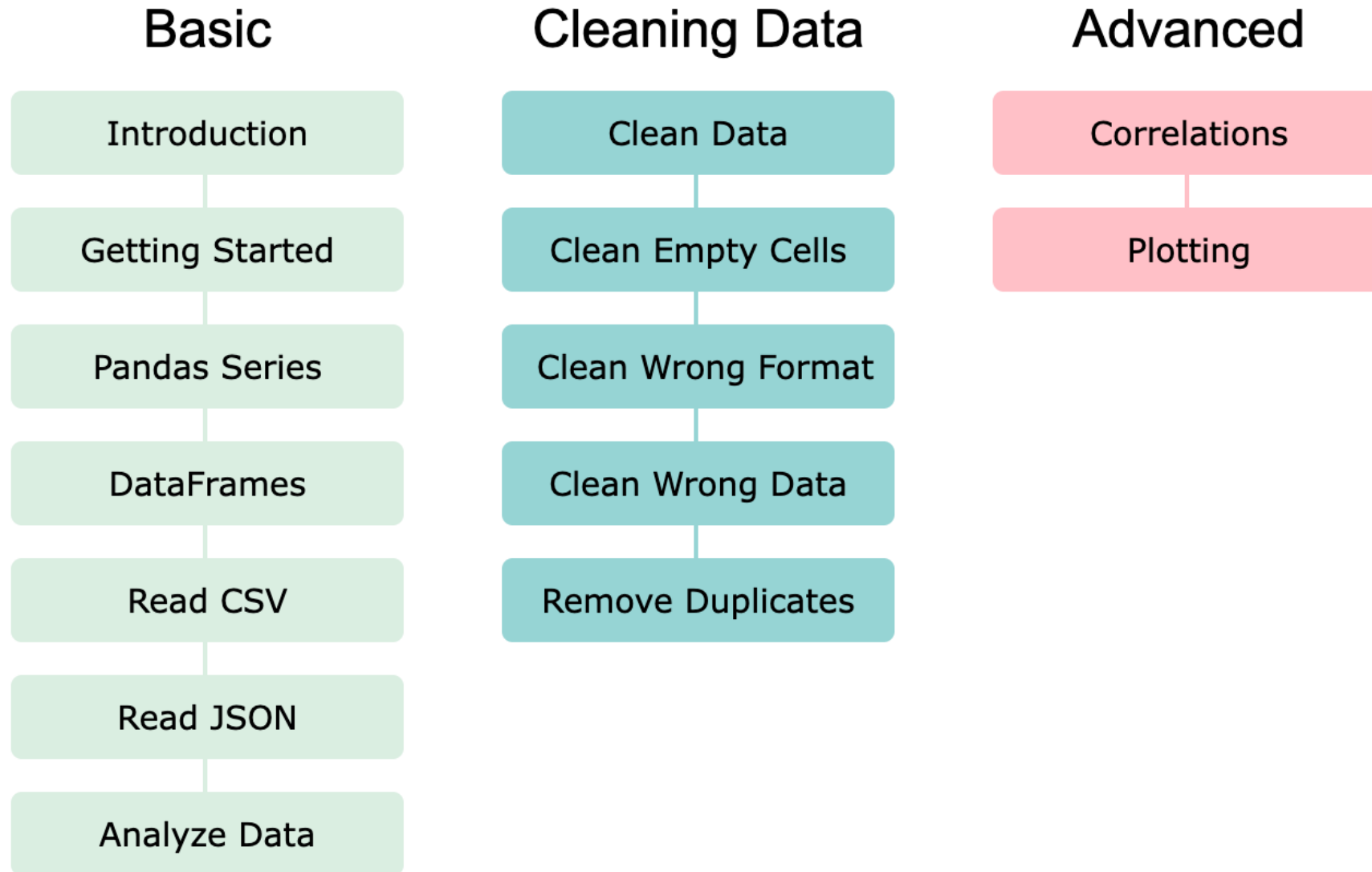
16 2023/12/27 Final Project Report

Data Analytics and Visualization with Python

Outline

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

Pandas: Data Analytics and Visualization



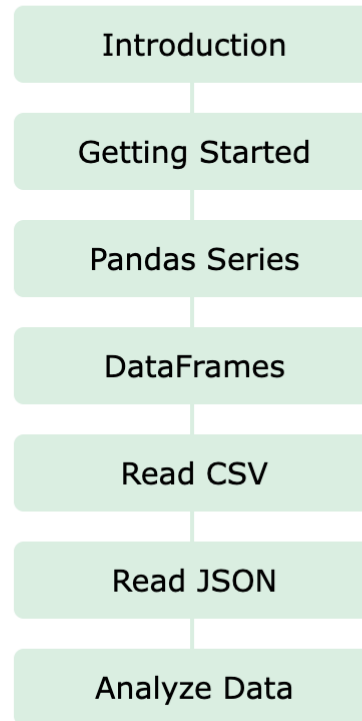
Pandas

Learning by Reading

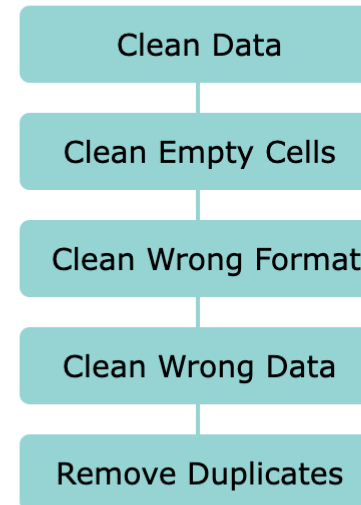
We have created 14 tutorial pages for you to learn more about Pandas.

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

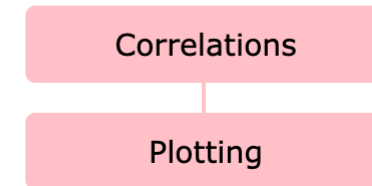
Basic



Cleaning Data



Advanced



Pandas Tutorial

Pandas HOME

Pandas Intro

Pandas Getting Started

Pandas Series

Pandas DataFrames

Pandas Read CSV

Pandas Read JSON

Pandas Analyzing Data

Cleaning Data

Cleaning Data

Cleaning Empty Cells

Cleaning Wrong Format

Cleaning Wrong Data

Removing Duplicates

Correlations

Pandas Correlations

Plotting

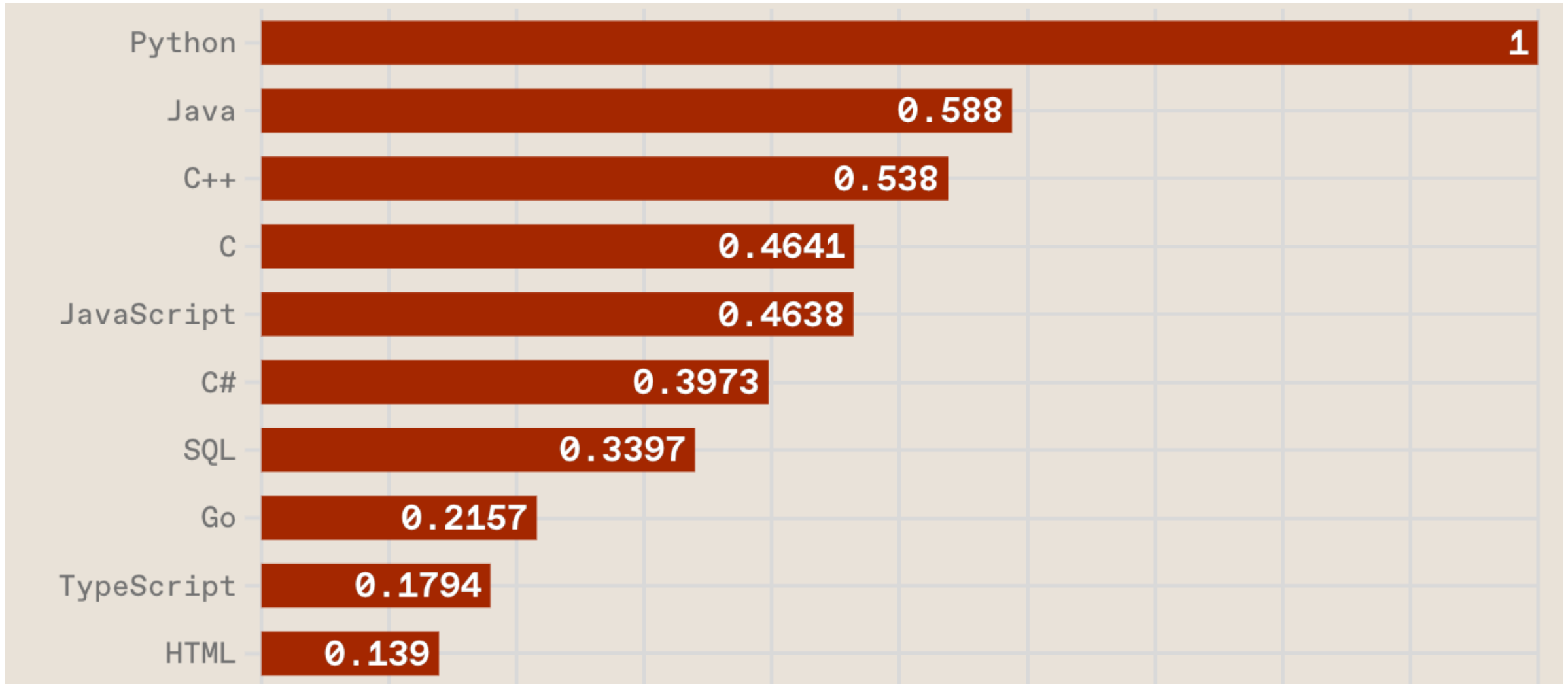
Pandas Plotting



Python

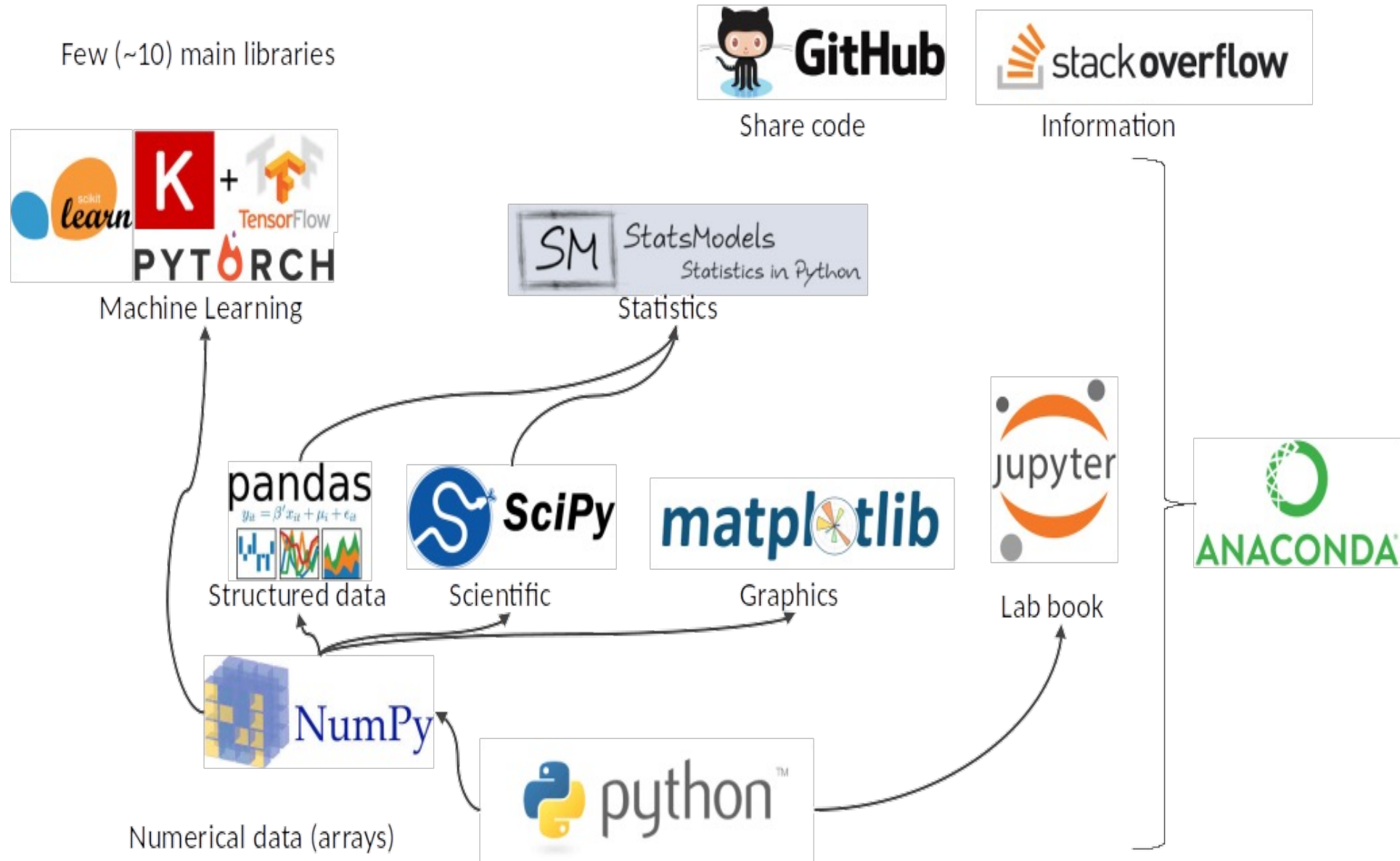
Programming

Top Programming Languages

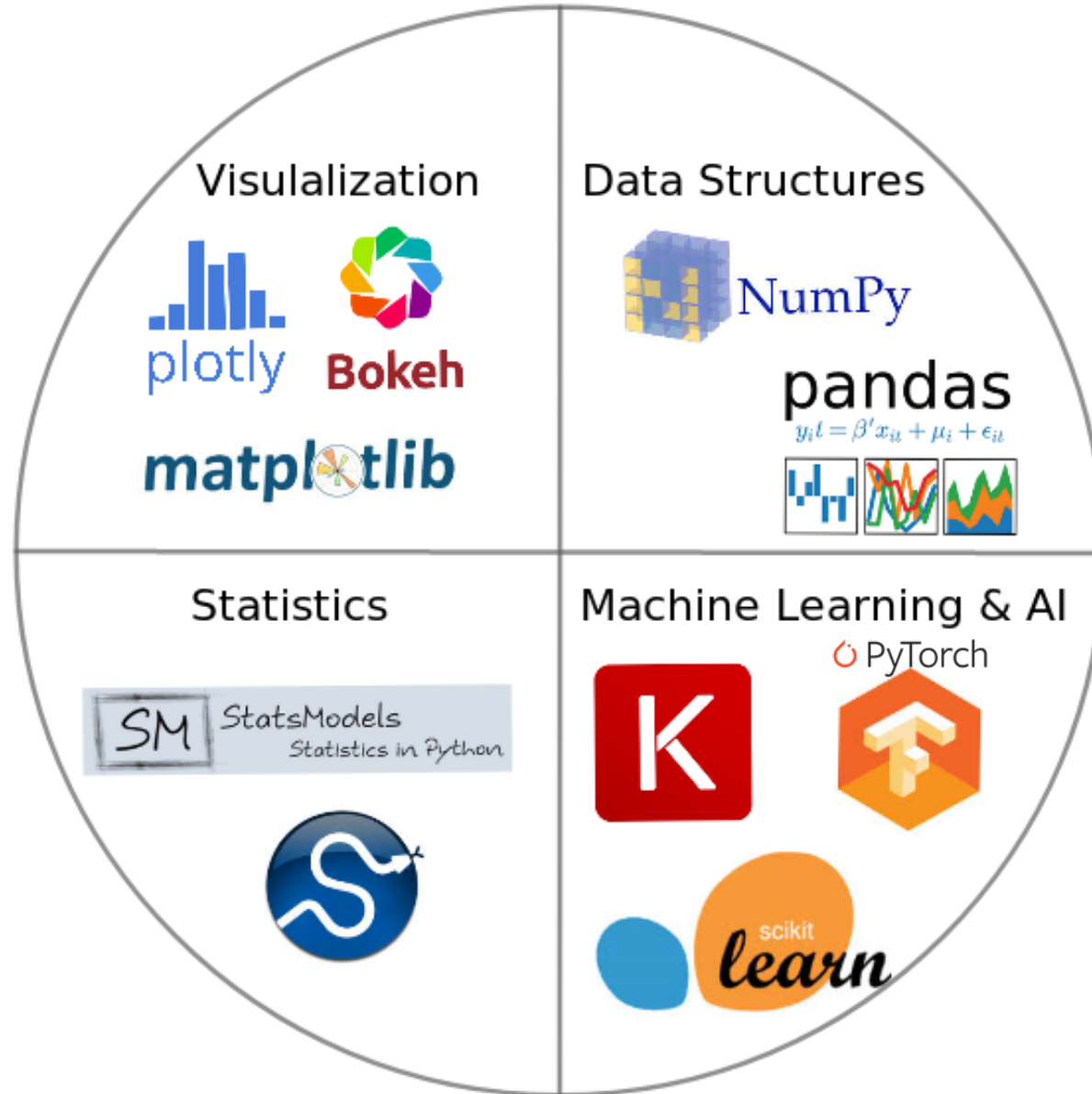


Python is an
interpreted,
object-oriented,
high-level
programming language
with
dynamic semantics.

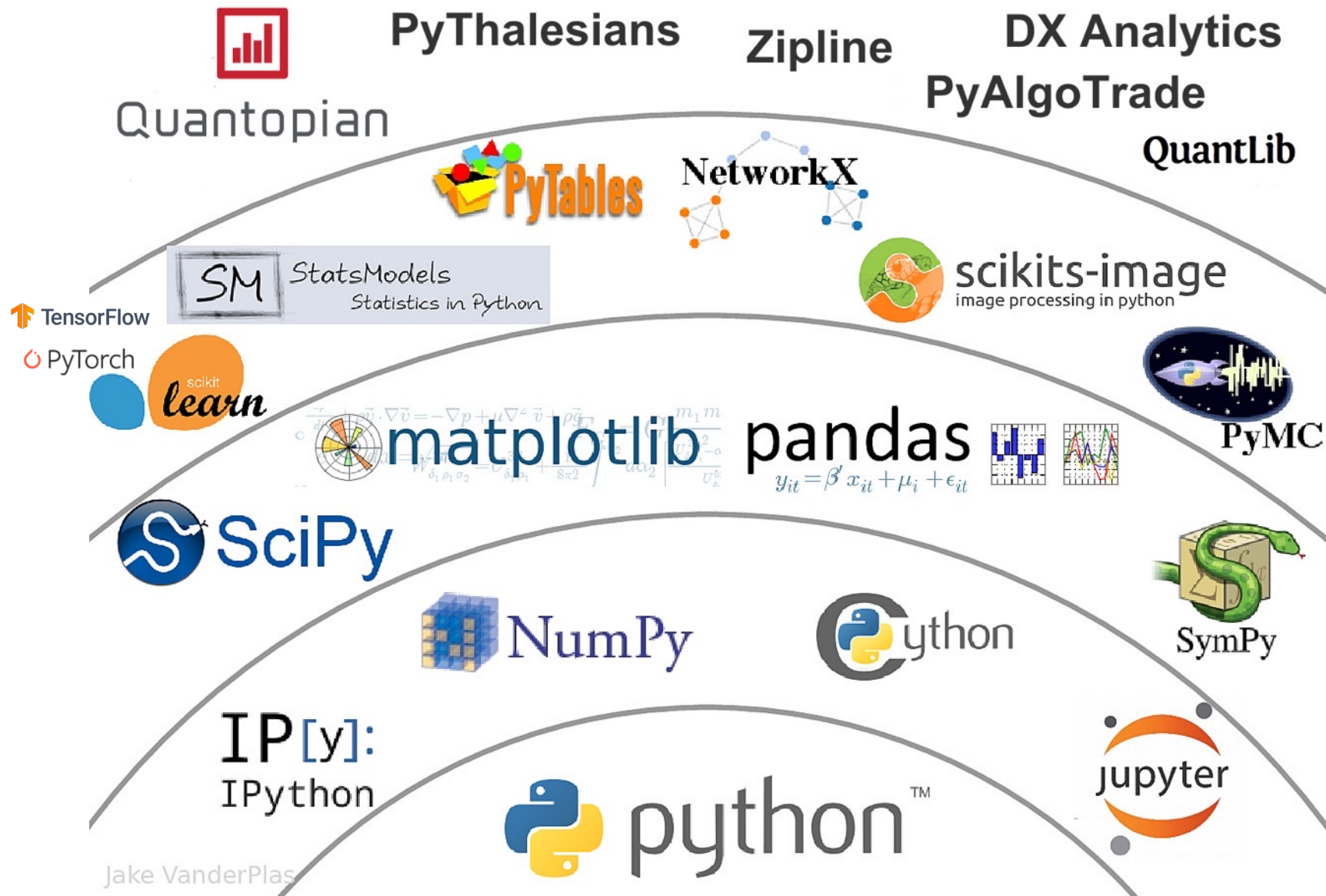
Python Ecosystem for Data Science



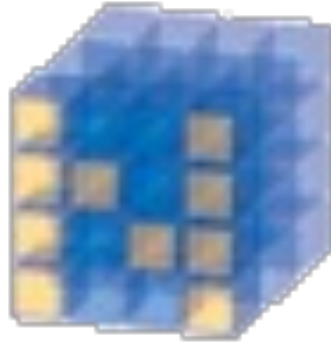
Python Ecosystem for Data Science



The Quant Finance PyData Stack



NumPy



NumPy

Base

N-dimensional array
package

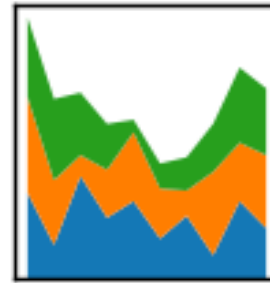
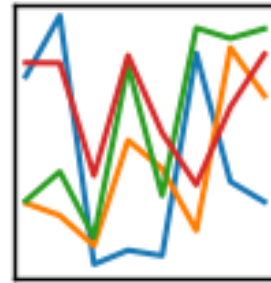
Python
matplotlib
matplotlib

Python

Pandas

pandas

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



Python Tutorial

Python HOME

Python Intro
Python Get Started
Python Syntax
Python Comments
Python Variables
Python Data Types
Python Numbers
Python Casting
Python Strings
Python Booleans
Python Operators
Python Lists
Python Tuples
Python Sets
Python Dictionaries
Python If...Else
Python While Loops
Python For Loops
Python Functions

Python Tutorial

◀ Home

Next ▶

Learn Python

Python is a popular programming language.

Python can be used on a server to create web applications.

[Start learning Python now »](#)

Learning by Examples

With our "Try it Yourself" editor, you can edit Python code and view the result.

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

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

W3Schools Python Pandas

Learning by Reading

Pandas Tutorial

Python Modules

NumPy Tutorial
Pandas Tutorial
SciPy Tutorial
Django Tutorial

Pandas Tutorial

Pandas HOME

Pandas Intro
Pandas Getting Started
Pandas Series
Pandas DataFrames
Pandas Read CSV
Pandas Read JSON
Pandas Analyzing Data

Cleaning Data

Cleaning Data
Cleaning Empty Cells
Cleaning Wrong Format
Cleaning Wrong Data
Removing Duplicates

Correlations

Pandas Correlations

Plotting

Pandas Plotting

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:

Basic

Introduction

Getting Started

Pandas Series

DataFrames

Read CSV

Read JSON

Analyze Data

Cleaning Data

Clean Data

Clean Empty Cells

Clean Wrong Format

Clean Wrong Data

Remove Duplicates

Advanced

Correlations

Plotting

Python Modules

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

Python Matplotlib

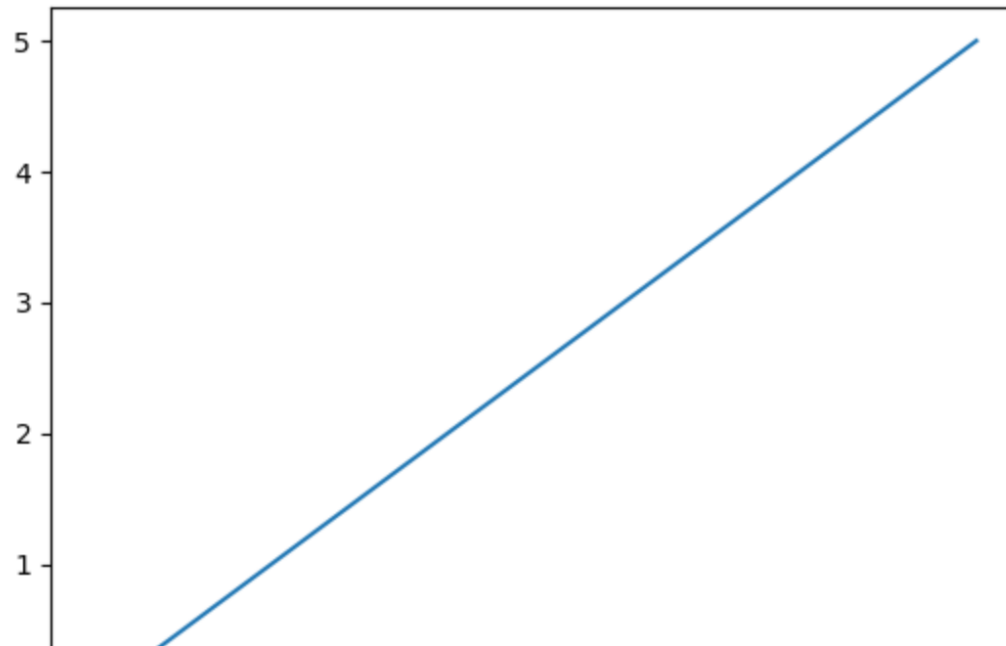
Matplotlib Intro

- 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

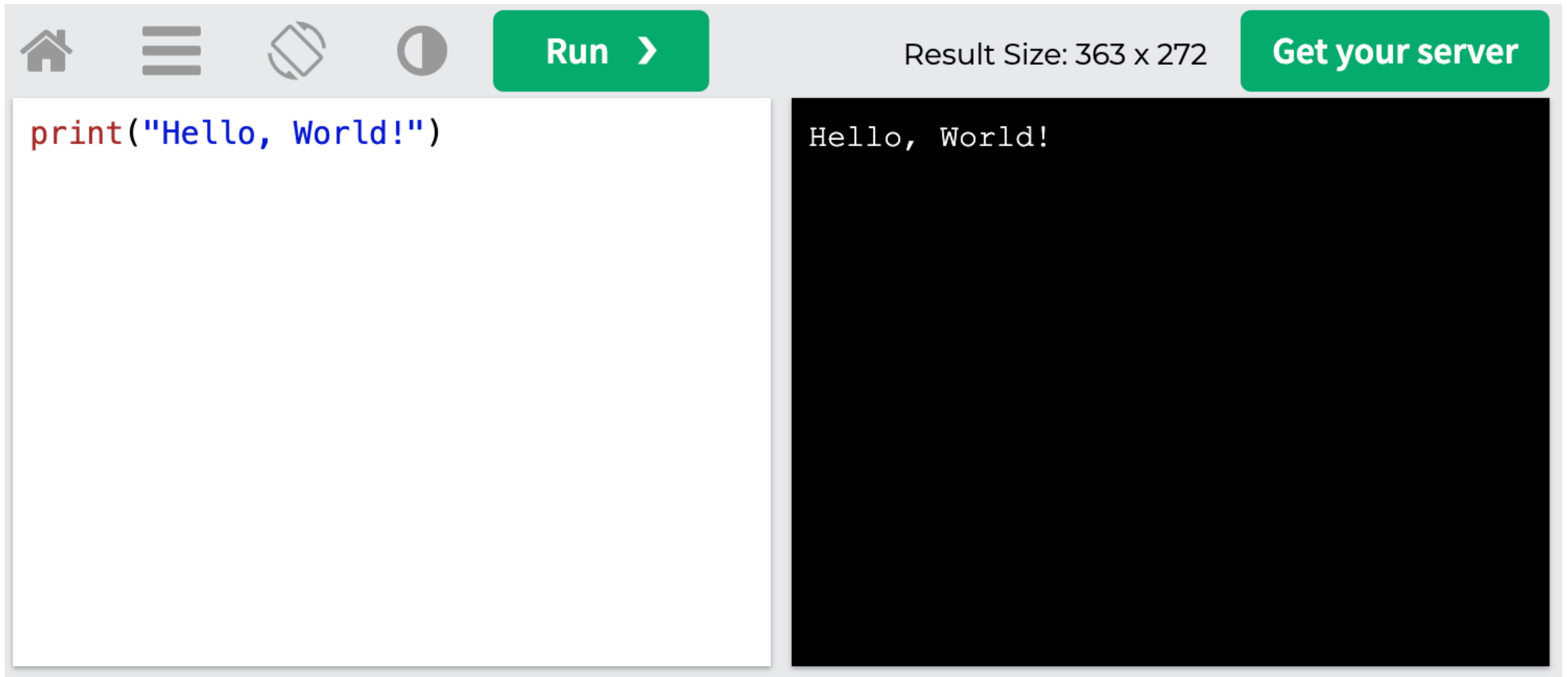
Matplotlib Tutorial

◀ Previous

Next ▶



W3Schools Python: Try Python

A screenshot of the W3Schools Python 'Try Python' interface. The interface has a light gray header bar with navigation icons (home, menu, refresh, moon) on the left, a green 'Run >' button in the center, and 'Result Size: 363 x 272' and a green 'Get your server' button on the right. Below the header, there is a white code editor on the left containing the Python code `print("Hello, World!")` and a black output window on the right displaying the result 'Hello, World!' in white text.

Run >

Result Size: 363 x 272

Get your server

```
print("Hello, World!")
```

Hello, World!

LearnPython.org



learnpython.org

Home

About

Certify

More Languages ▾

Python

Java

HTML

Go

C

C++

JavaScript

PHP

Shell

C#

Perl

Ruby

Scala

SQL

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

Google for Education > Python

Search

English



Filter

Overview

Python Set Up

Python Intro

Strings

Lists

Sorting

Dicts and Files

Regular Expressions

Utilities

Lecture Videos

1.1 Introduction, strings

1.2 Lists and sorting

1.3 Dicts and files

2.1 Regular expr

2.2 Utilities

2.3 Utilities urllib

2.4 Conclusions

Python Exercises

Home > Products > Google for Education > Python

Was this helpful?

Google's Python Class

Welcome to Google's Python Class -- this is a free class for people with a little bit of programming experience who want to learn Python. The class includes written materials, lecture videos, and lots of code exercises to practice Python coding. These materials are used within Google to introduce Python to people who have just a little programming experience. The first exercises work on basic Python concepts like strings and lists, building up to the later exercises which are full programs dealing with text files, processes, and http connections. The class is geared for people who have a little bit of programming experience in some language, enough to know what a "variable" or "if statement" is. Beyond that, you do not need to be an expert programmer to use this material.

To get started, the Python sections are linked at the left -- [Python Set Up](#) to get Python installed on your machine, [Python Introduction](#) for an introduction to the language, and then [Python Strings](#) starts the coding material, leading to the first exercise. The end of each written section includes a link to the code exercise for that section's material. The lecture videos parallel the written materials, introducing Python, then strings, then first exercises, and so on. At Google, all this material makes up an intensive 2-day class, so the videos are organized as the day-1 and day-2 sections.

This material was created by [Nick Parlante](#) working in the engEDU group at Google. Special thanks for the help from my Google colleagues John Cox, Steve Glassman, Piotr Kaminski, and Antoine Picard. And finally thanks to Google and my director Maggie Johnson for the enlightened generosity to put these materials out on the internet for free under the [Creative Commons Attribution 2.5](#) license -- share and enjoy!

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

Google Colab

The screenshot shows the Google Colaboratory web interface. At the top, a browser window displays the URL <https://colab.research.google.com/notebooks/welcome.ipynb>. The page header includes the 'Hello, Colaboratory' logo, a menu (File, Edit, View, Insert, Runtime, Tools, Help), a 'SHARE' button, and a user profile icon. Below the header is a toolbar with buttons for 'CODE', 'TEXT', 'CELL', and 'COPY TO DRIVE'. A left sidebar contains a 'Table of contents' with links to 'Getting Started', 'Highlighted Features', 'TensorFlow execution', 'GitHub', 'Visualization', 'Forms', 'Examples', and 'Local runtime support'. The main content area features a 'Welcome to Colaboratory!' message, followed by a 'Getting Started' section with a list of links: 'Overview of Colaboratory', 'Loading and saving data: Local files, Drive, Sheets, Google Cloud Storage', 'Importing libraries and installing dependencies', 'Using Google Cloud BigQuery', 'Forms, Charts, Markdown, & Widgets', 'TensorFlow with GPU', and 'Machine Learning Crash Course: Intro to Pandas & First Steps with TensorFlow'. Below this is a 'Highlighted Features' section with a 'Seedbank' subsection and a 'TensorFlow execution' subsection. The TensorFlow section includes a text description and a matrix addition example.

Table of contents

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

SECTION

Welcome to Colaboratory!

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. See our [FAQ](#) for more info.

Getting Started

- [Overview of Colaboratory](#)
- [Loading and saving data: Local files, Drive, Sheets, Google Cloud Storage](#)
- [Importing libraries and installing dependencies](#)
- [Using Google Cloud BigQuery](#)
- [Forms, Charts, Markdown, & Widgets](#)
- [TensorFlow with GPU](#)
- [Machine Learning Crash Course: Intro to Pandas & First Steps with TensorFlow](#)

Highlighted Features

Seedbank

Looking for Colab notebooks to learn from? Check out [Seedbank](#), a place to discover interactive machine learning examples.

TensorFlow execution

Colaboratory allows you to execute TensorFlow code in your browser with a single click. The example below adds two matrices.

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

Connect Google Colab in Google Drive

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

My Drive - Google Drive

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

Drive

Search Drive

My Drive

Quick Access

New

My Drive

Computers

Shared with me

Recent

Starred

Trash

Backups

Storage

0 bytes of 15 GB used

UPGRADE STORAGE

Get Backup and Sync for Mac

New folder...

Upload files...

Upload folder...

Google Docs

Google Sheets

Google Slides

More

Google Forms

Google Drawings

Google My Maps

Google Sites

Connect more apps

Google Colab

The screenshot shows the Google Drive web interface. A modal window titled "Connect apps to Drive" is open in the center. The modal has a search bar at the top with the text "colab" entered and highlighted by a red dashed box. Below the search bar, there is a grid of app cards. The cards are arranged in two rows and three columns. The first row contains "ZIP Extractor", "LUMIN PDF", and "cloudconvert". The second row contains "Sejda", "DocHub", and "Google Forms". Each card displays the app's logo, name, and user count. The background of the Drive interface is dimmed, showing the left sidebar with navigation options like "My Drive", "Computers", "Shared with me", "Recent", "Starred", "Trash", "Backups", and "Storage". The top of the browser shows the address bar with the URL "https://drive.google.com/drive/u/2/my-drive".

My Drive - Google Drive

Search Drive

Connect apps to Drive

colab

ZIP Extractor
Extract ZIP files to Google Drive
Extraction complete.
View extracted files Share Extract another
Test.zip
ZIP Extractor
307,585 users

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

cloudconvert
CloudConvert
373,161 users

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

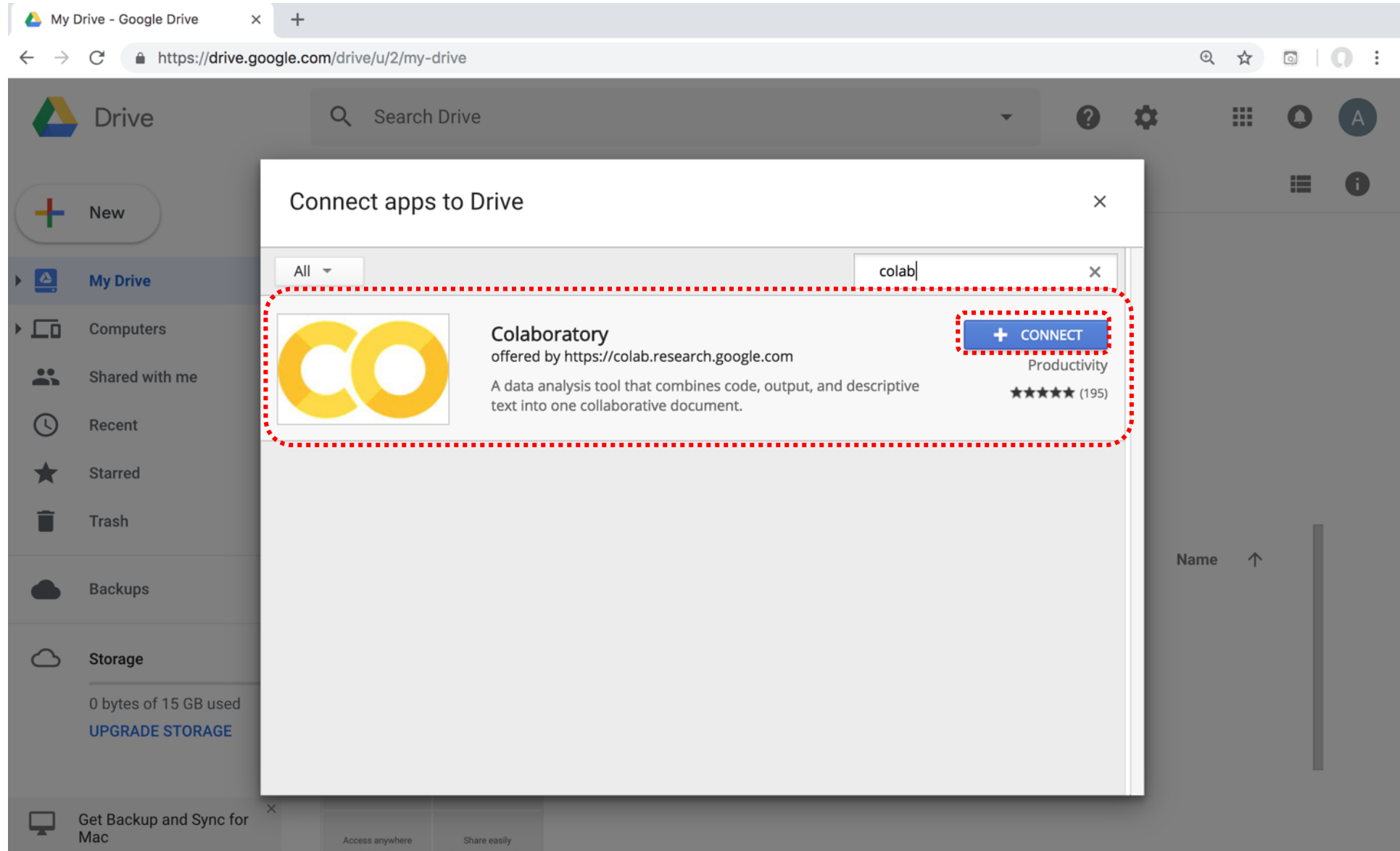
DocHub
Edit, Send & Sign PDFs
DocHub - Edit and Sign PDF Docu...
2,131,600 users

Google Forms
Google Forms
4,803,614 users

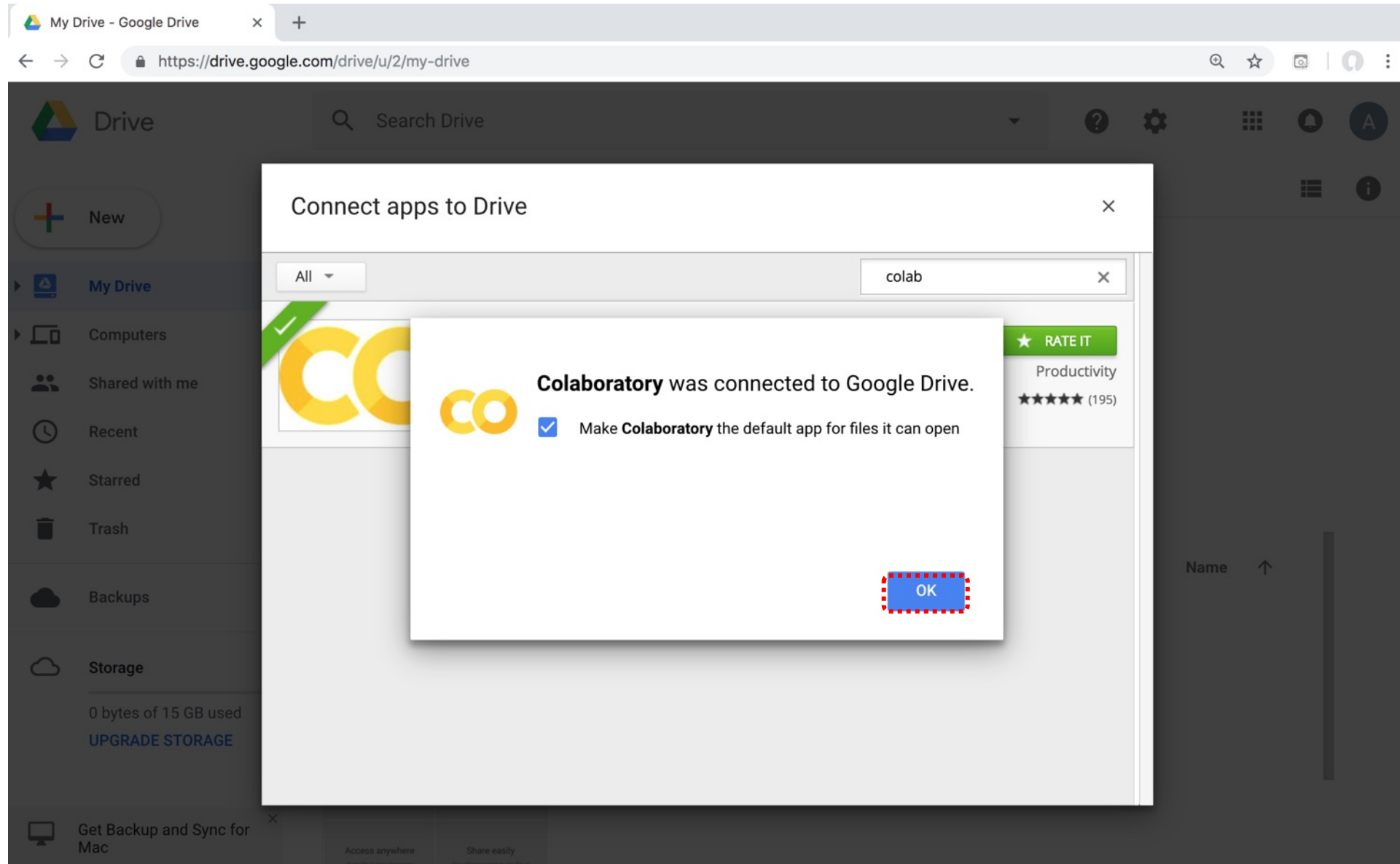
Name ↑

Get Backup and Sync for Mac

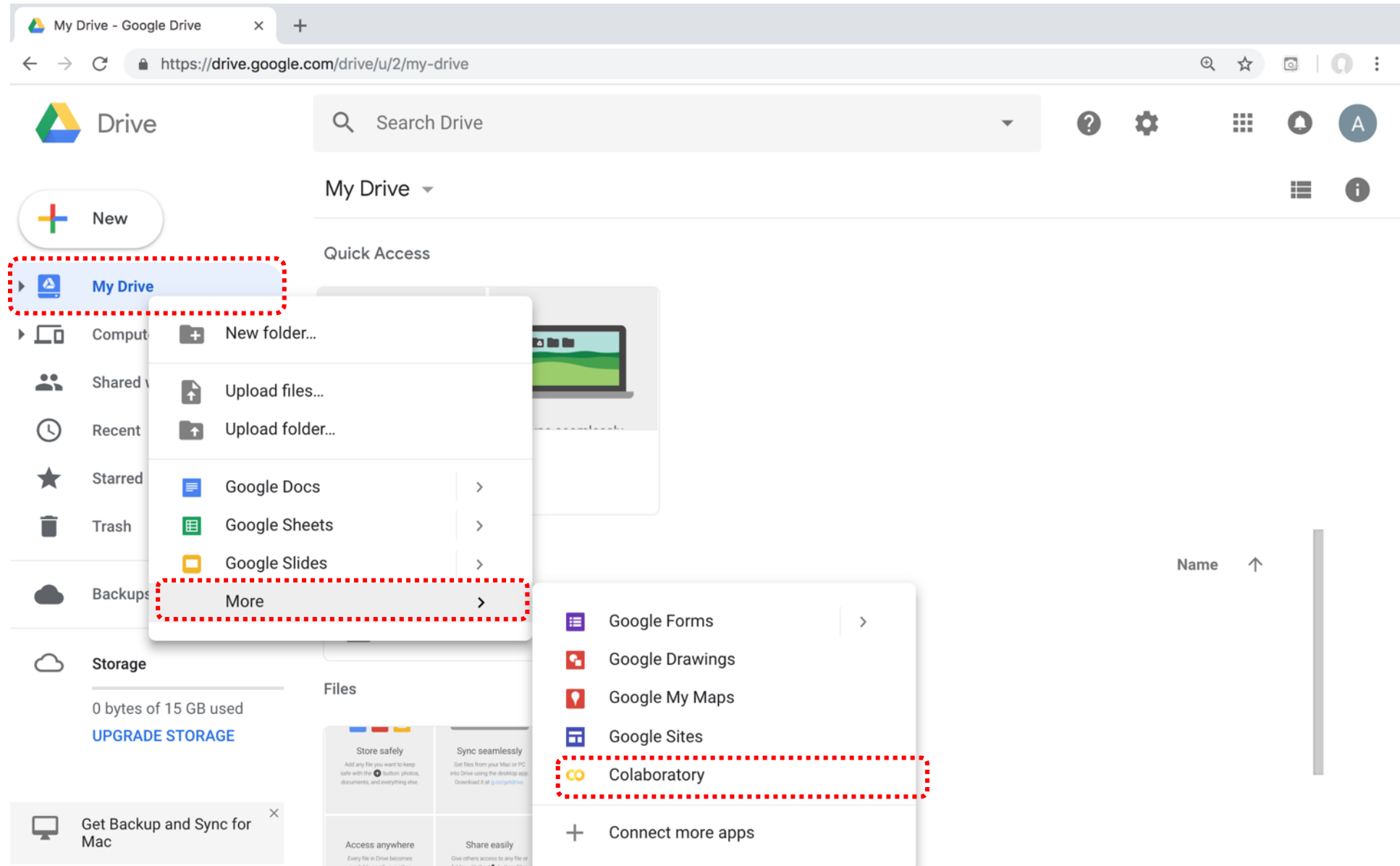
Google Colab



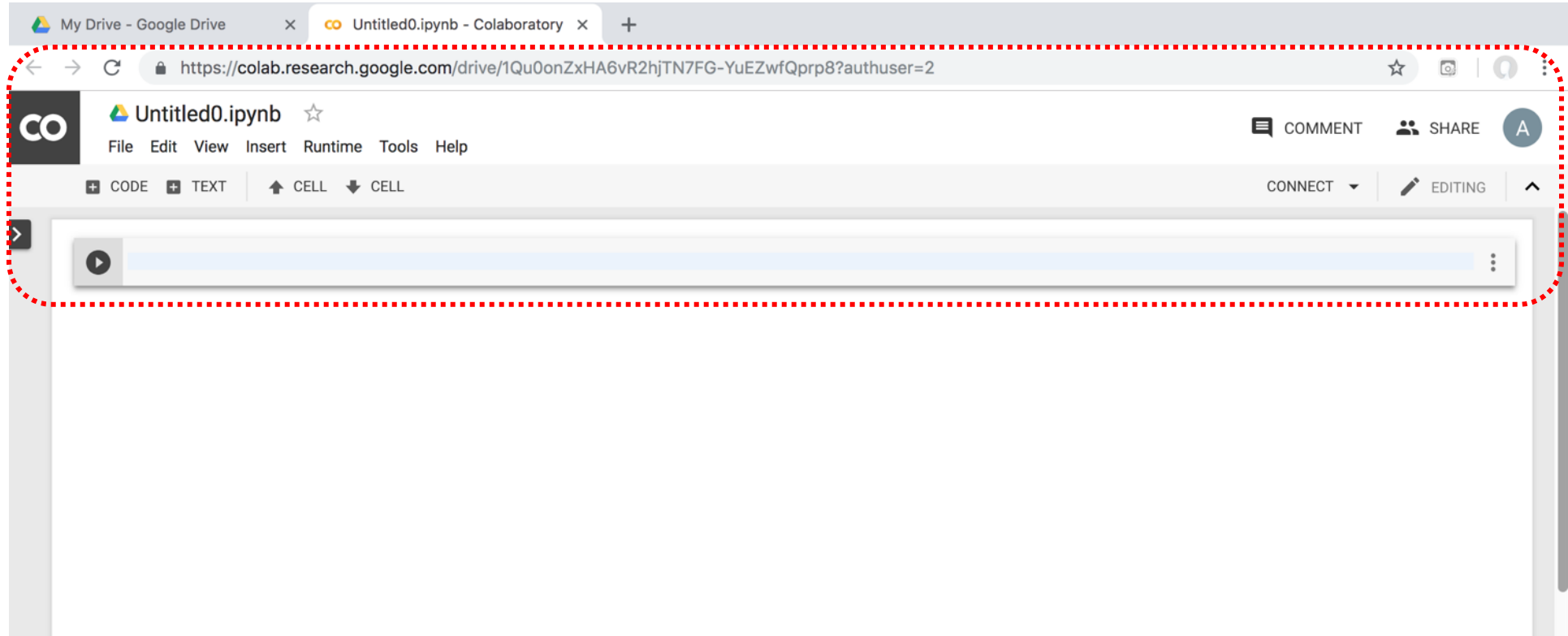
Connect Colaboratory to Google Drive



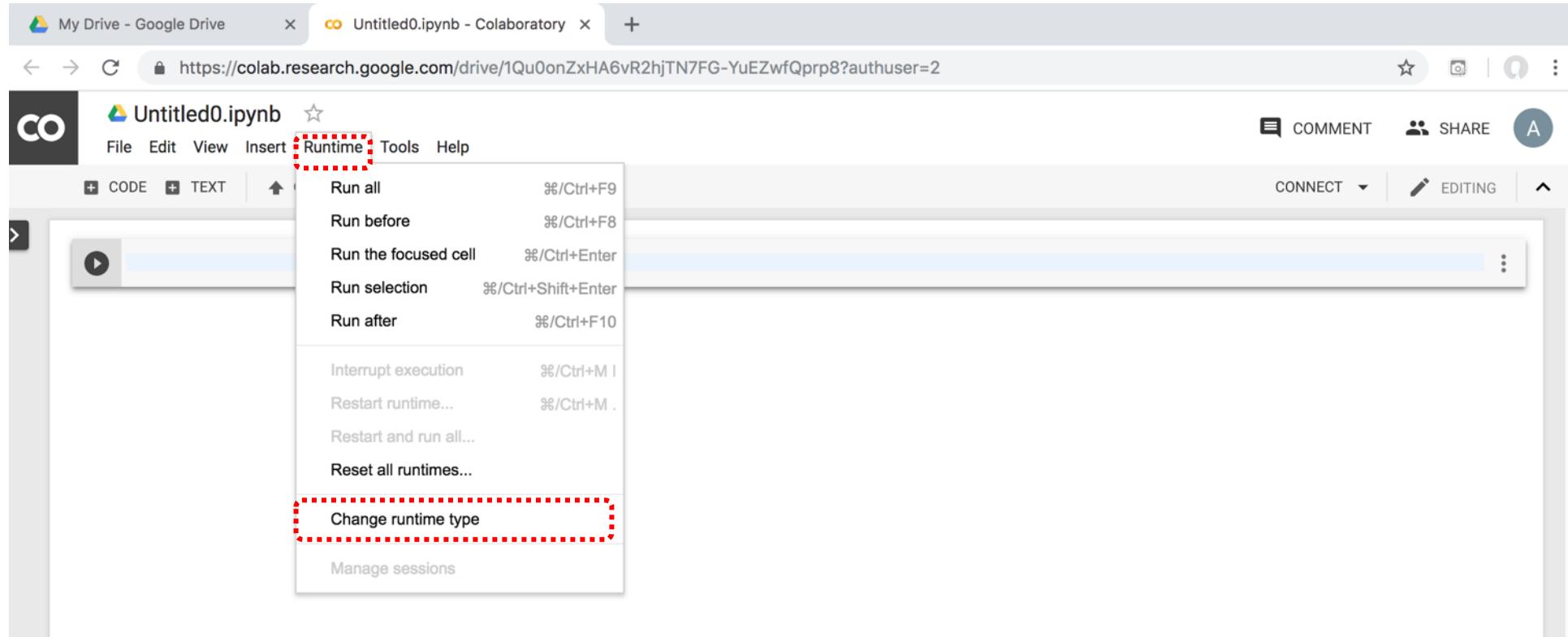
Google Colab



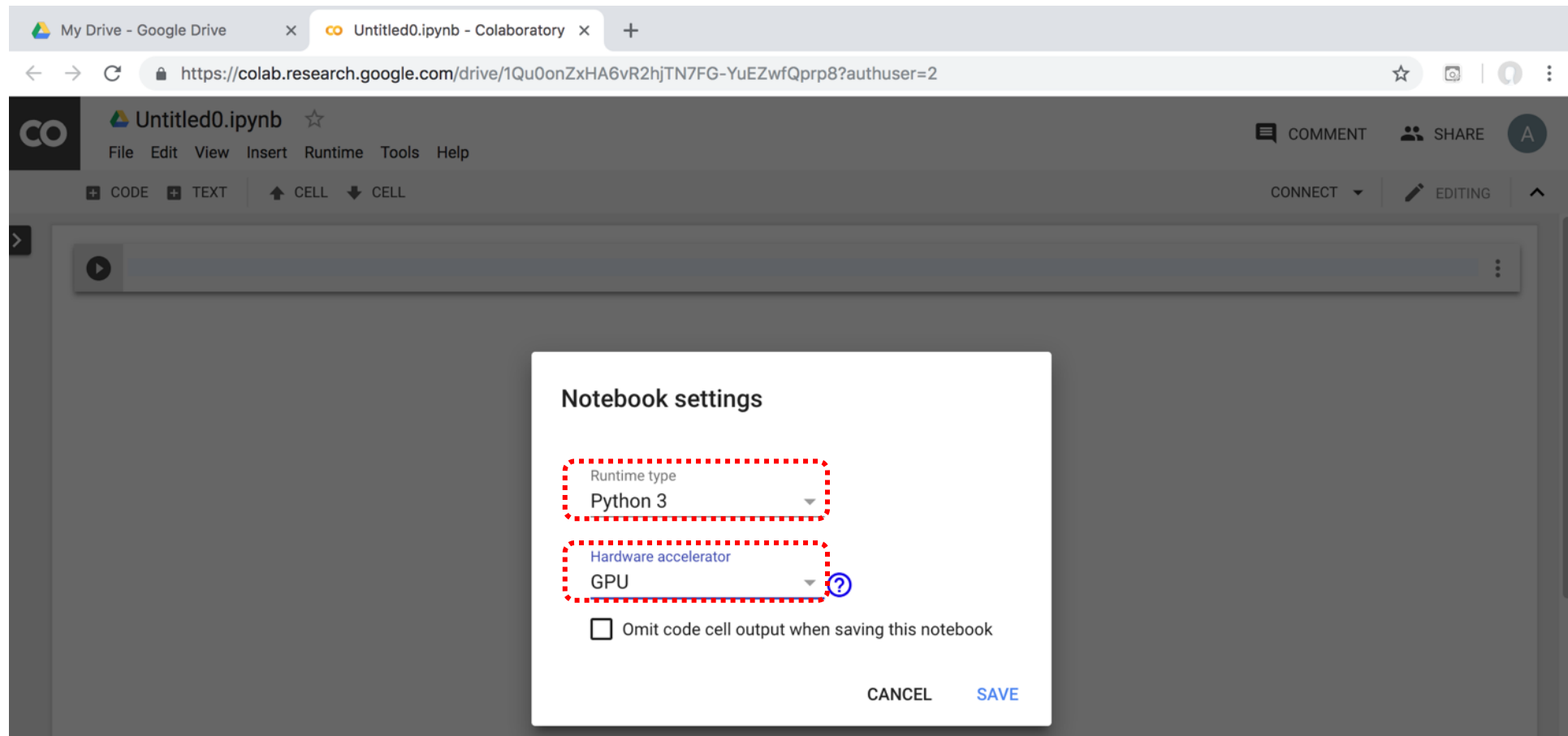
Google Colab



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>

python101.ipynb - Colaboratory

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

python101.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL

CONNECTED EDITING

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

194.87

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

194.87

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

194.87

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

Pass

<https://tinyurl.com/aintpuppython101>





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

Python Modules

NumPy Tutorial
Pandas Tutorial
SciPy Tutorial
Django Tutorial

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

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

W3Schools Python Pandas

Learning by Reading

Pandas Tutorial

Python Modules

NumPy Tutorial
Pandas Tutorial
SciPy Tutorial
Django Tutorial

Pandas Tutorial

Pandas HOME

Pandas Intro
Pandas Getting Started
Pandas Series
Pandas DataFrames
Pandas Read CSV
Pandas Read JSON
Pandas Analyzing Data

Cleaning Data

Cleaning Data
Cleaning Empty Cells
Cleaning Wrong Format
Cleaning Wrong Data
Removing Duplicates

Correlations

Pandas Correlations

Plotting

Pandas Plotting

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:

Basic

Introduction

Getting Started

Pandas Series

DataFrames

Read CSV

Read JSON

Analyze Data

Cleaning Data

Clean Data

Clean Empty Cells

Clean Wrong Format

Clean Wrong Data

Remove Duplicates

Advanced

Correlations

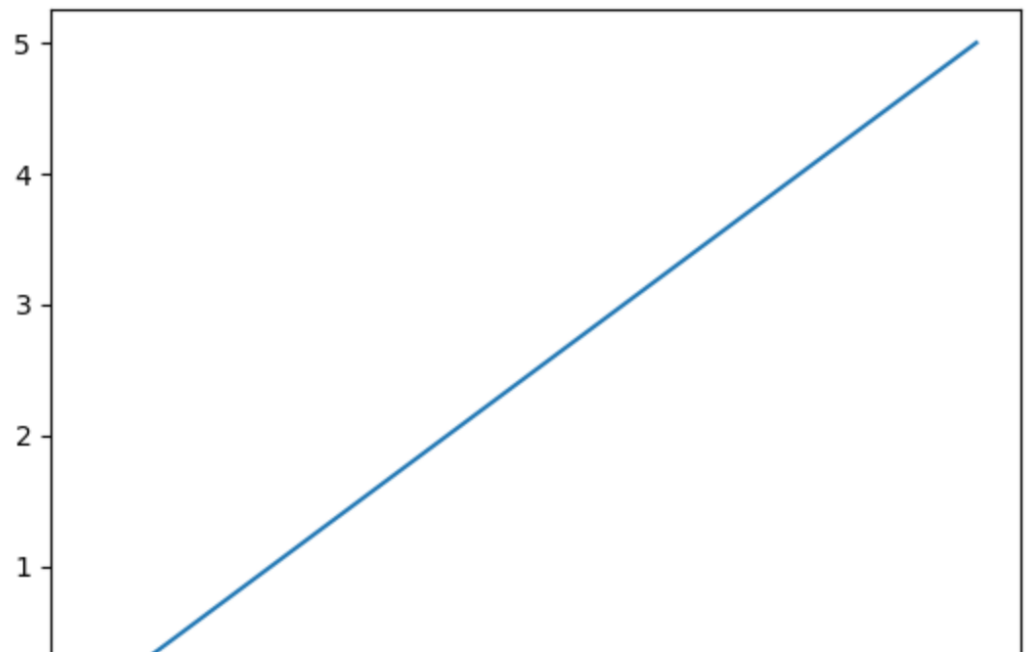
Plotting

- 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

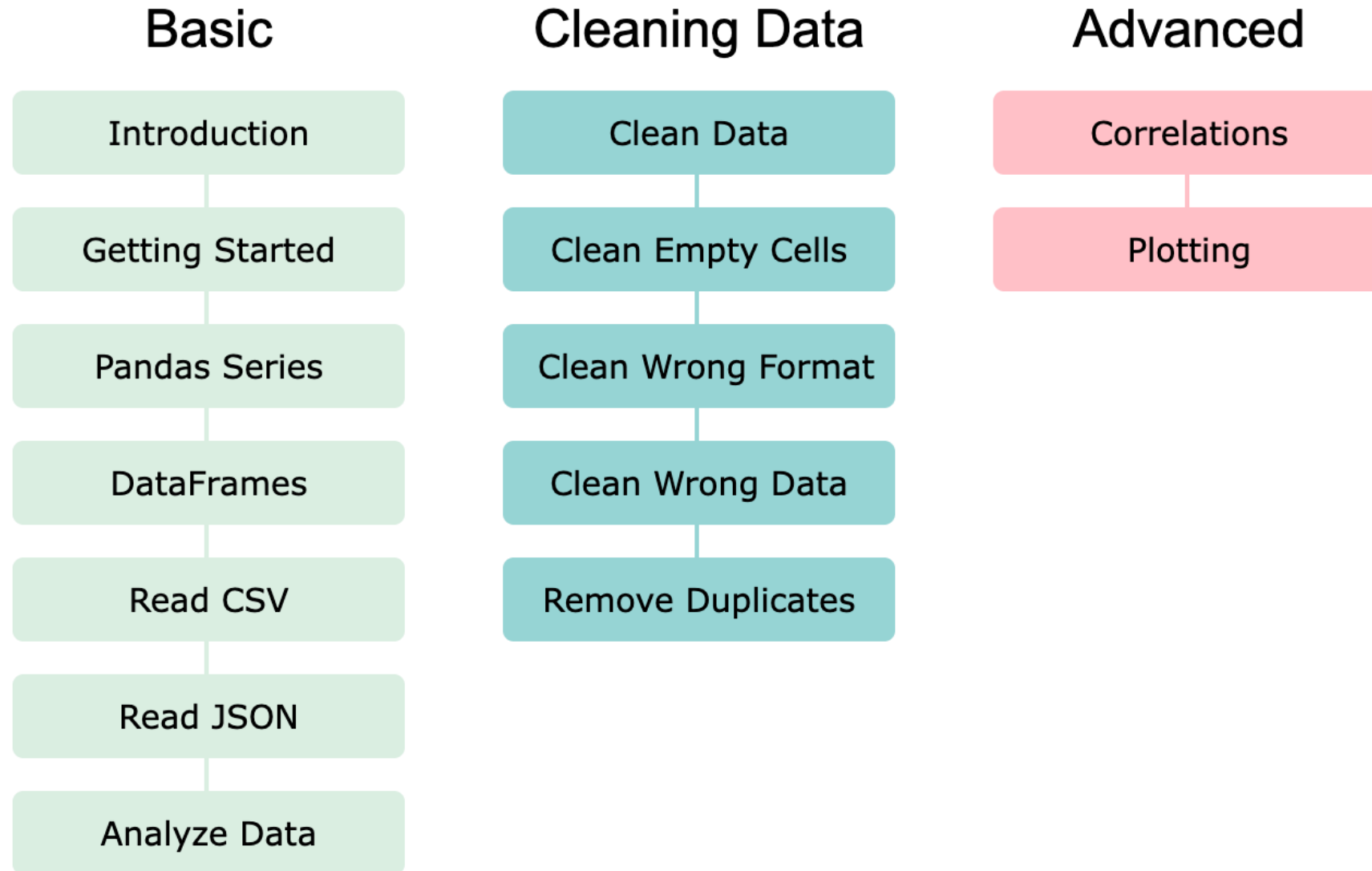
Matplotlib Tutorial

[< Previous](#)

[Next >](#)



Pandas: Data Analytics and Visualization



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

wesm / pydata-bookPublic

NotificationsFork14.1kStar19k

<> CodeIssues2Pull requests1ActionsProjectsWikiSecurityInsights

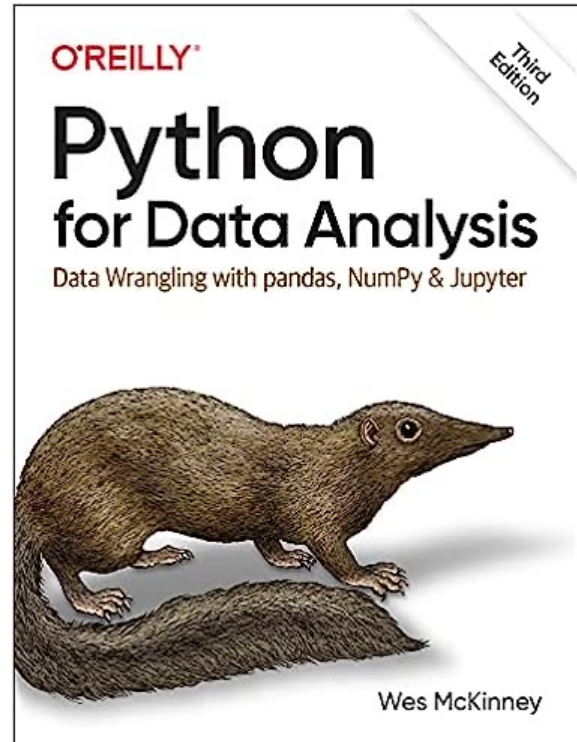
3rd-edition3 branches0 tagsGo to fileCode

wesm Upload cleaner notebook files without internal build toolchai...f1757b83 days ago70 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>

43

NumPy



NumPy

Base

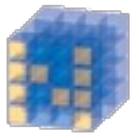
N-dimensional array
package

NumPy
is the
fundamental package
for
scientific computing
with Python.



NumPy

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



NumPy

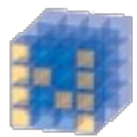
NumPy ndarray

One-dimensional Array (1-D Array)

0	1			n-1
1	2	3	4	5

Two-dimensional Array (2-D Array)

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



NumPy

NumPy

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

```
v
```

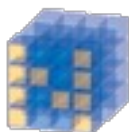
```
2 * v
```

```
import numpy as np
```

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

```
v
```

```
2 * v
```



NumPy

Base

N-dimensional
array package

```
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 {}
```

Lists []

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

4
60
70
90

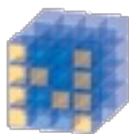


NumPy Create Array

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
c = a * b
c
```

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
c = a * b
c
```

```
array([ 4, 10, 18])
```

NumPy

NumPy

```
1 import numpy as np
2
3 a = np.zeros((2,2)) # Create an array of all zeros
4 print(a)           # Prints "[[ 0.  0.]
5                     #           [ 0.  0.]]"
6
7 b = np.ones((1,2)) # Create an array of all ones
8 print(b)           # Prints "[[ 1.  1.]]"
9
10 c = np.full((2,2), 7) # Create a constant array
11 print(c)             # Prints "[[ 7.  7.]
12                     #           [ 7.  7.]]"
13
14 d = np.eye(2)        # Create a 2x2 identity matrix
15 print(d)            # Prints "[[ 1.  0.]
16                     #           [ 0.  1.]]"
17
18 e = np.random.random((2,2)) # Create an array filled with random values
19 print(e)              # Might print "[[ 0.91940167  0.08143941]
20                      #           [ 0.68744134  0.87236687]]"
```

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

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

a.shape

a.ndim

a.dtype.name

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

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

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

```
(3, 5)
```

```
a.ndim
```

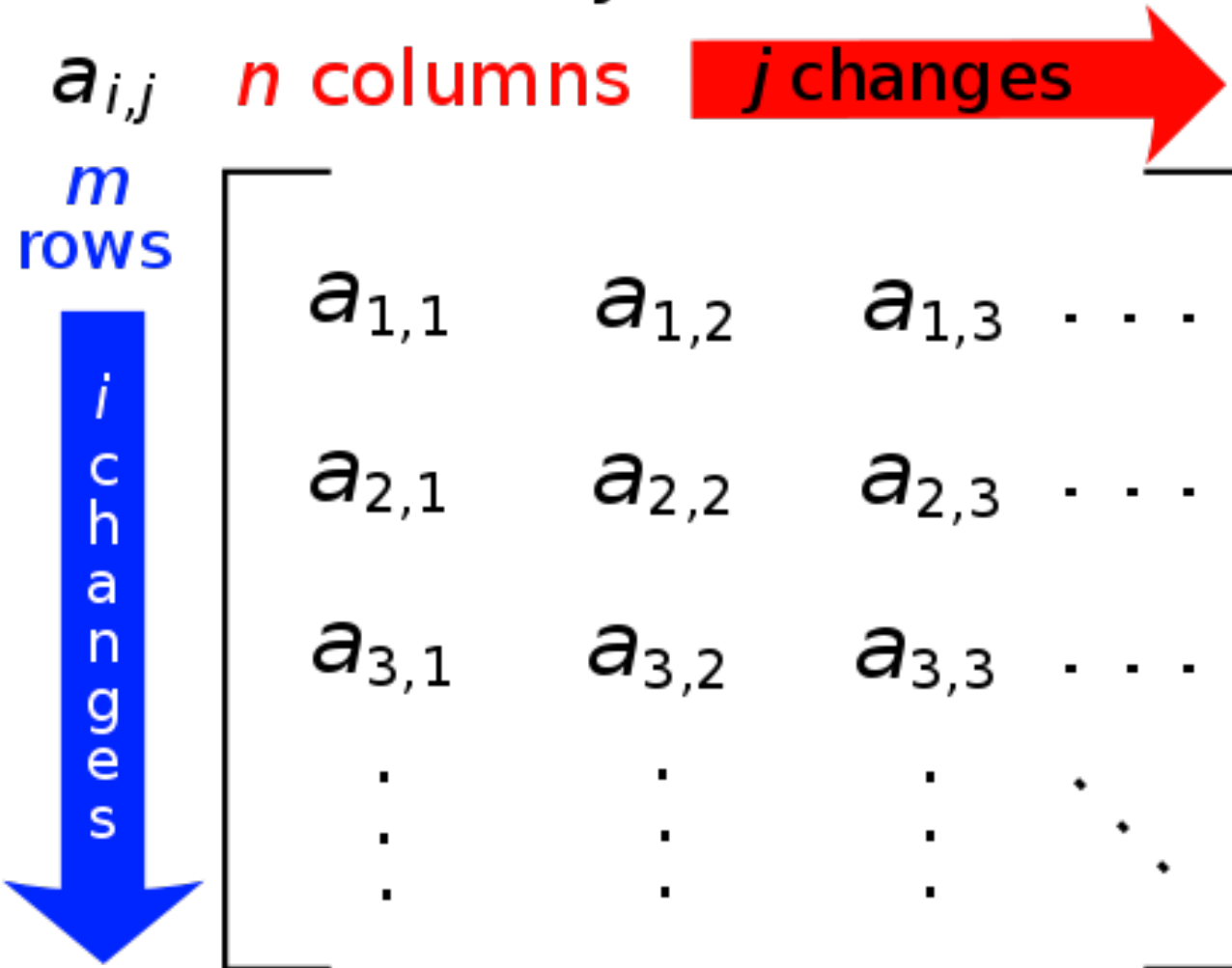
```
2
```

```
a.dtype.name
```

```
'int64'
```

Matrix

m -by- n matrix



NumPy ndarray: Multidimensional Array Object

NumPy ndarray

One-dimensional Array (1-D Array)

0	1			n-1
1	2	3	4	5

Two-dimensional Array (2-D Array)

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

```
import numpy as np  
a = np.array([1,2,3,4,5])
```

One-dimensional Array (1-D Array)

0	1			n-1
1	2	3	4	5

```
a = np.array([1,2,3,4,5])  
a
```

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

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

Two-dimensional Array (2-D Array)

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

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

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

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

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


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

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

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

```
print(a.ndim)
```

```
2
```

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

```
(3, 4)
```

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

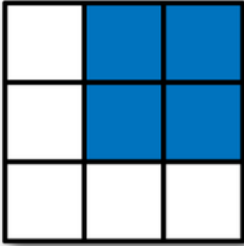
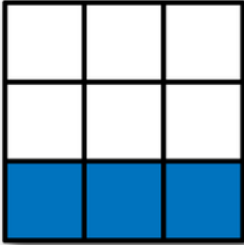
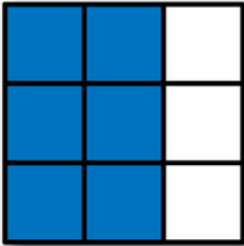
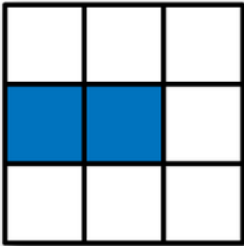
NumPy Basics: Arrays and Vectorized Computation

NumPy Array

axis 1

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

Numpy Array

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

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]

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

Python Data Analysis Library

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

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

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.**

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

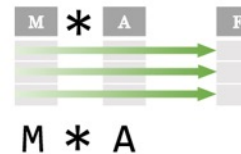
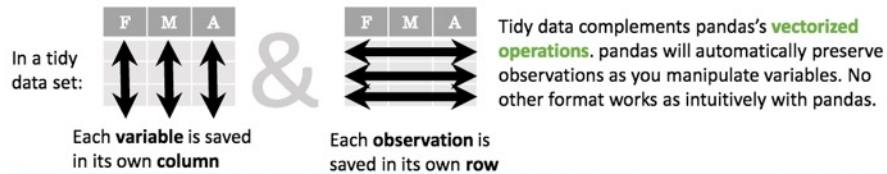
Comparison with SAS

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

Python Pandas Cheat Sheet

Data Wrangling
with pandas
Cheat Sheet
<http://pandas.pydata.org>

Tidy Data – A foundation for wrangling in 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	1	4	7
d	2	5	8
e	2	6	9

```
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']))
```

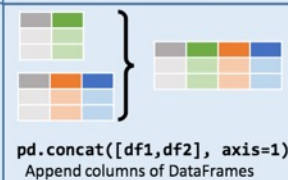
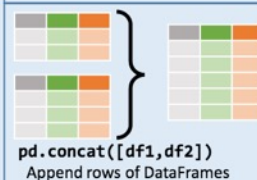
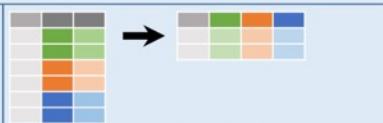
Create DataFrame with a MultiIndex

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



```
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).

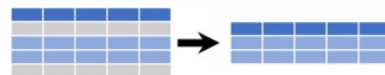
df=df.rename(columns = {'y':'year'})
Rename the columns of a DataFrame

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.
```

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

regex	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.
```

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

<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

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])
```

Pandas DataFrame

```
type(df)
```

```
type(df)
```

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

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

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

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

```
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')
```



```
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

```
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

df2.dtypes

```
df2.dtypes
```

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

Python Accounting Application with Pandas

```
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

Python Data Analysis and Visualization



Python

Pandas



Python
matplotlib
matplotlib

Python seaborn



seaborn

Python plotly



Python

bokeh



Python

Altair



Altair

Python matplotlib



Fork me on GitHub

[Installation](#) [Documentation](#) [Examples](#) [Tutorials](#) [Contributing](#)

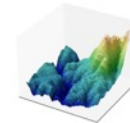
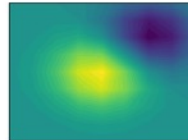
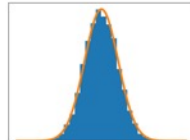
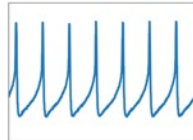
Search

[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

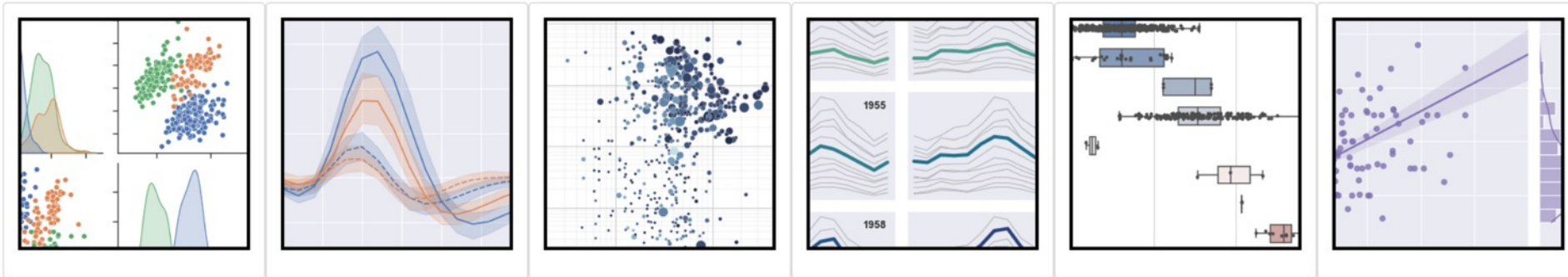
2.2.5: [docs](#) | [changelog](#)

Development version

[docs](#)

Support Matplotlib

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](#)

Python Plotly Graphing Library

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](#).

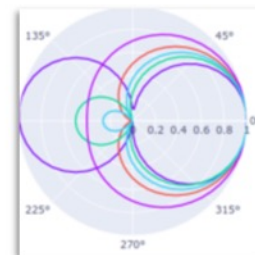
Our recommended IDE for Plotly's Python graphing library is Dash Enterprise's [Data Science Workspaces](#), which has both Jupyter notebook and Python code file support. [Find out if your company is using Dash Enterprise](#).

[Install Dash Enterprise on Azure](#) | [Install Dash Enterprise on AWS](#)

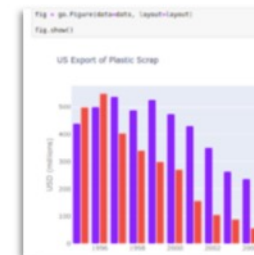
Fundamentals

[More Fundamentals »](#)

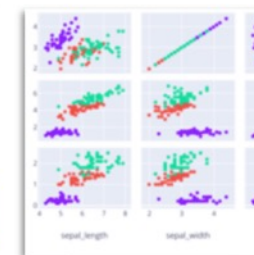

The Figure Data Structure



Creating and Updating Figures



Displaying Figures



Plotly Express

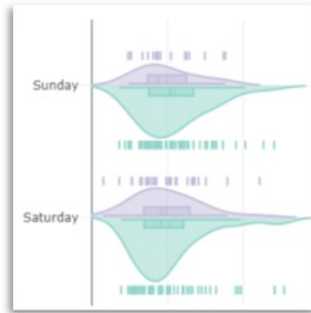


Analytical Apps with Dash

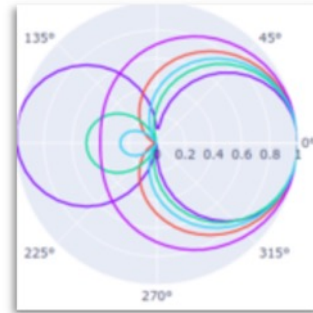
Python Plotly Graphing Library

Fundamentals

[More Fundamentals »](#)



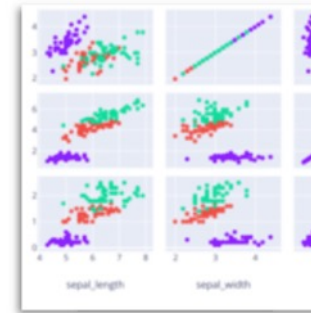
The Figure Data Structure



Creating and Updating Figures



Displaying Figures



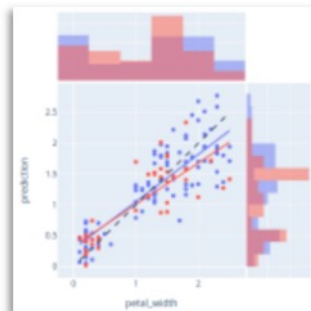
Plotly Express



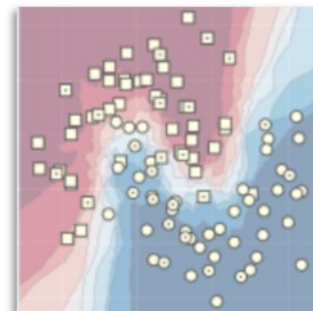
Analytical Apps with Dash

Artificial Intelligence and Machine Learning

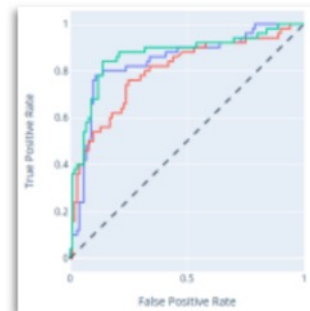
[More AI and ML »](#)



ML Regression



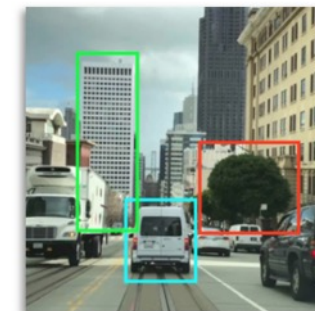
kNN Classification



ROC and PR Curves



PCA Visualization

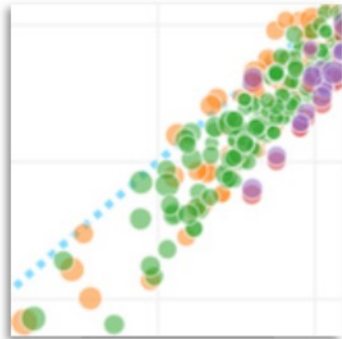


AI/ML Apps with Dash

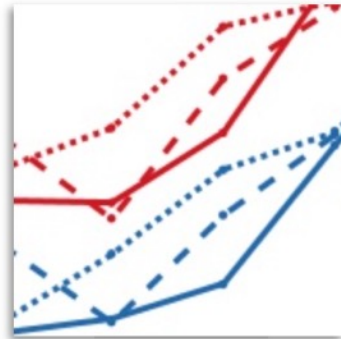
Python Plotly Graphing Library

Basic Charts

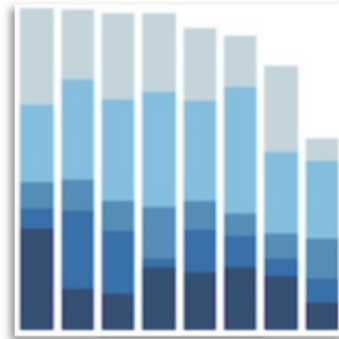
[More Basic Charts »](#)



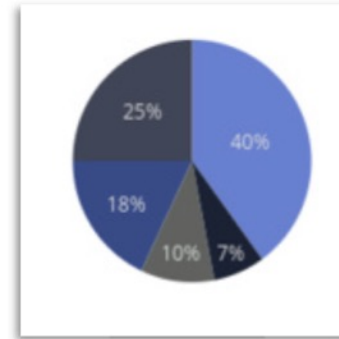
Scatter Plots



Line Charts



Bar Charts



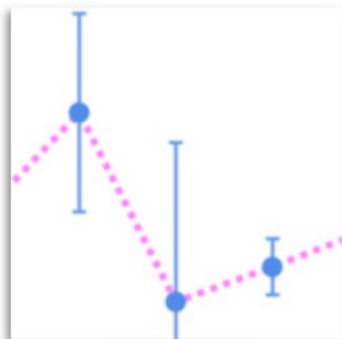
Pie Charts



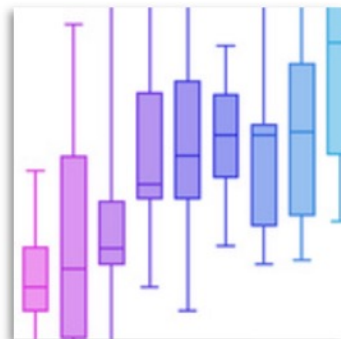
Bubble Charts

Statistical Charts

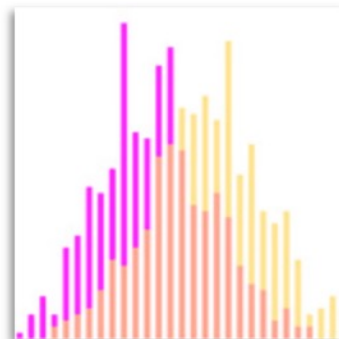
[More Statistical Charts »](#)



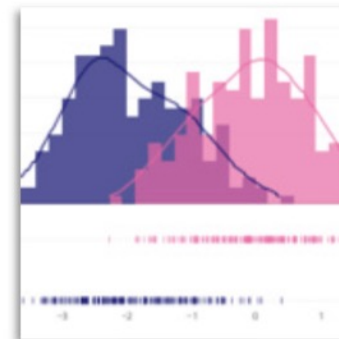
Error Bars



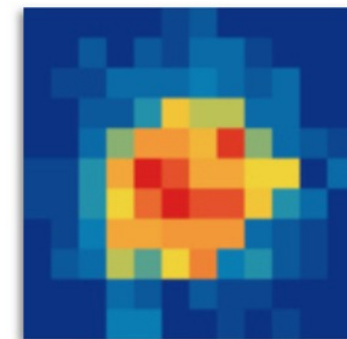
Box Plots



Histograms



Distplots

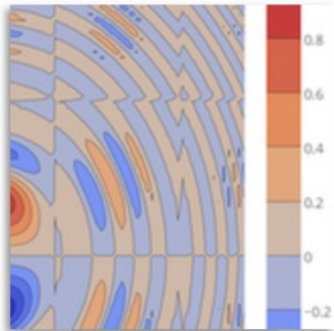


[2D Histograms](#)

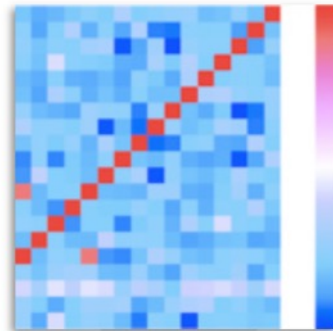
Python Plotly Graphing Library

Scientific Charts

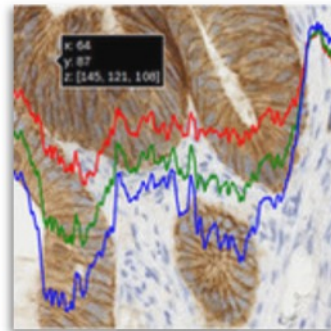
[More Scientific Charts »](#)



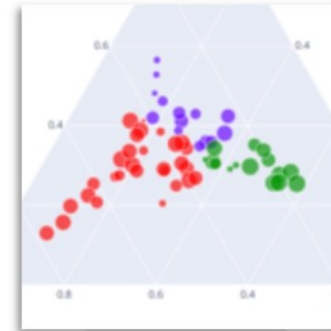
Contour Plots



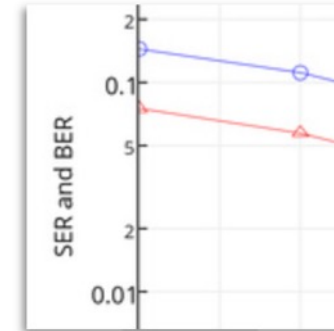
Heatmaps



Imshow



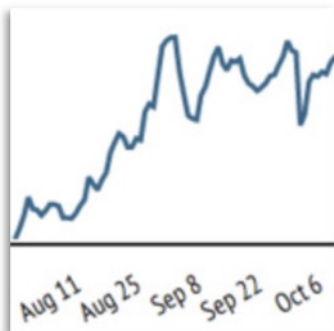
Ternary Plots



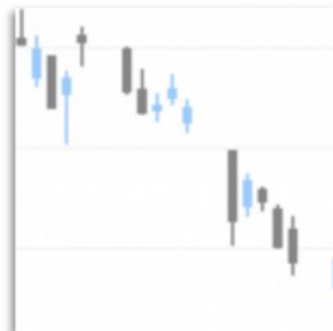
Log Plots

Financial Charts

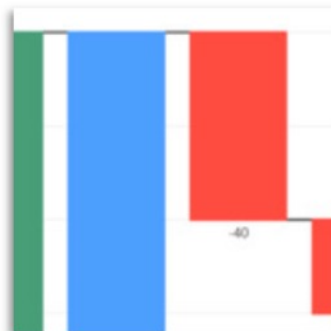
[More Financial Charts »](#)



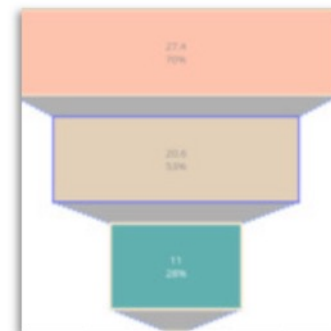
Time Series and Date
Axes



Candlestick Charts



Waterfall Charts



Funnel Chart

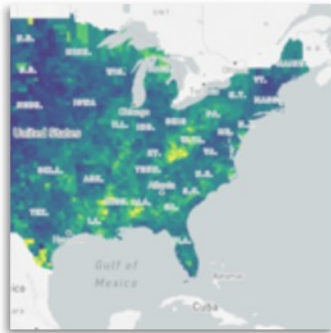


[OHLC Charts](#)

Python Plotly Graphing Library

Maps

[More Maps »](#)



Mapbox Choropleth Maps



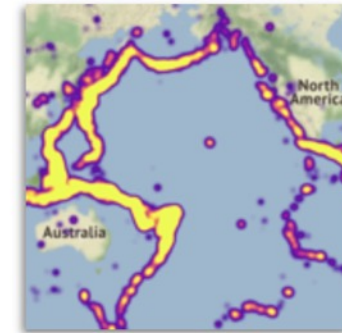
Lines on Mapbox



Filled Area on Maps



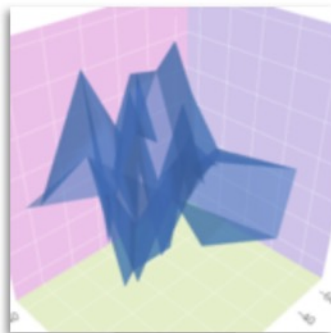
Bubble Maps



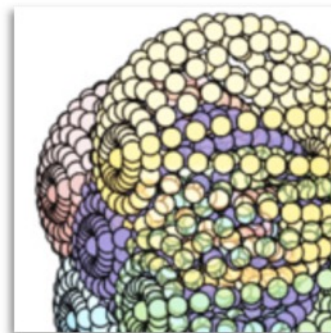
Mapbox Density Heatmap

3D Charts

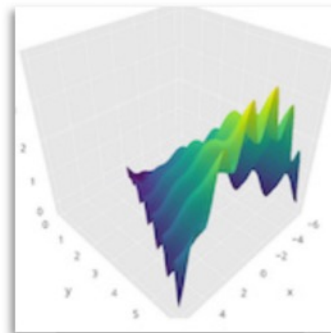
[More 3D Charts »](#)



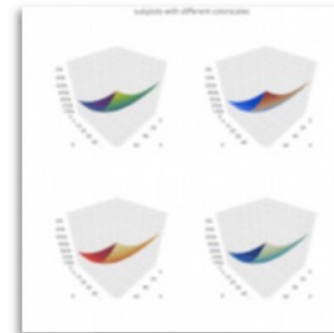
3D Axes



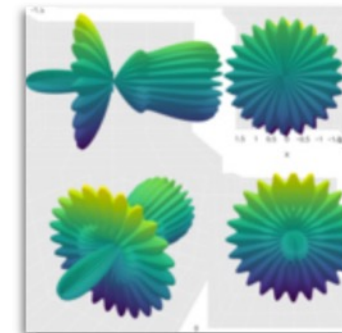
3D Scatter Plots



3D Surface Plots



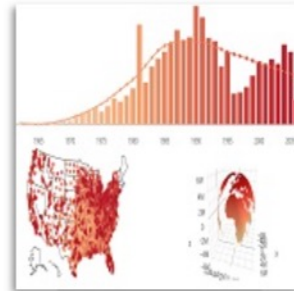
3D Subplots



[3D Camera Controls](#)

Python Plotly Graphing Library

Subplots



Mixed Subplots



Map Subplots

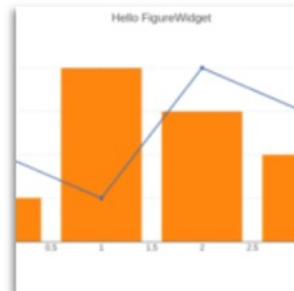


Table and Chart Subplots

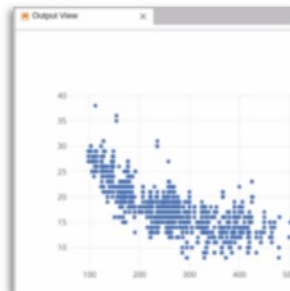


Figure Factory Subplots

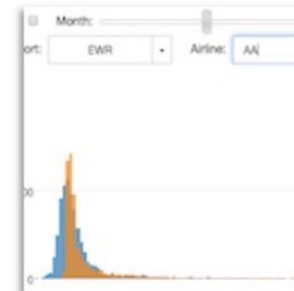
Jupyter Widgets Interaction



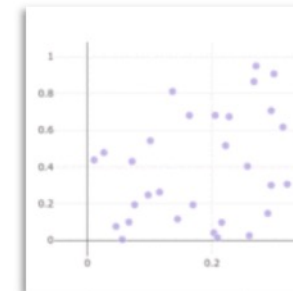
Plotly FigureWidget Overview



Jupyter Lab with FigureWidget



Interactive Data Analysis with FigureWidget ipywidgets



Click Events

bokeh Python Bokeh



2.3.0 ▾

First steps

User guide

Gallery

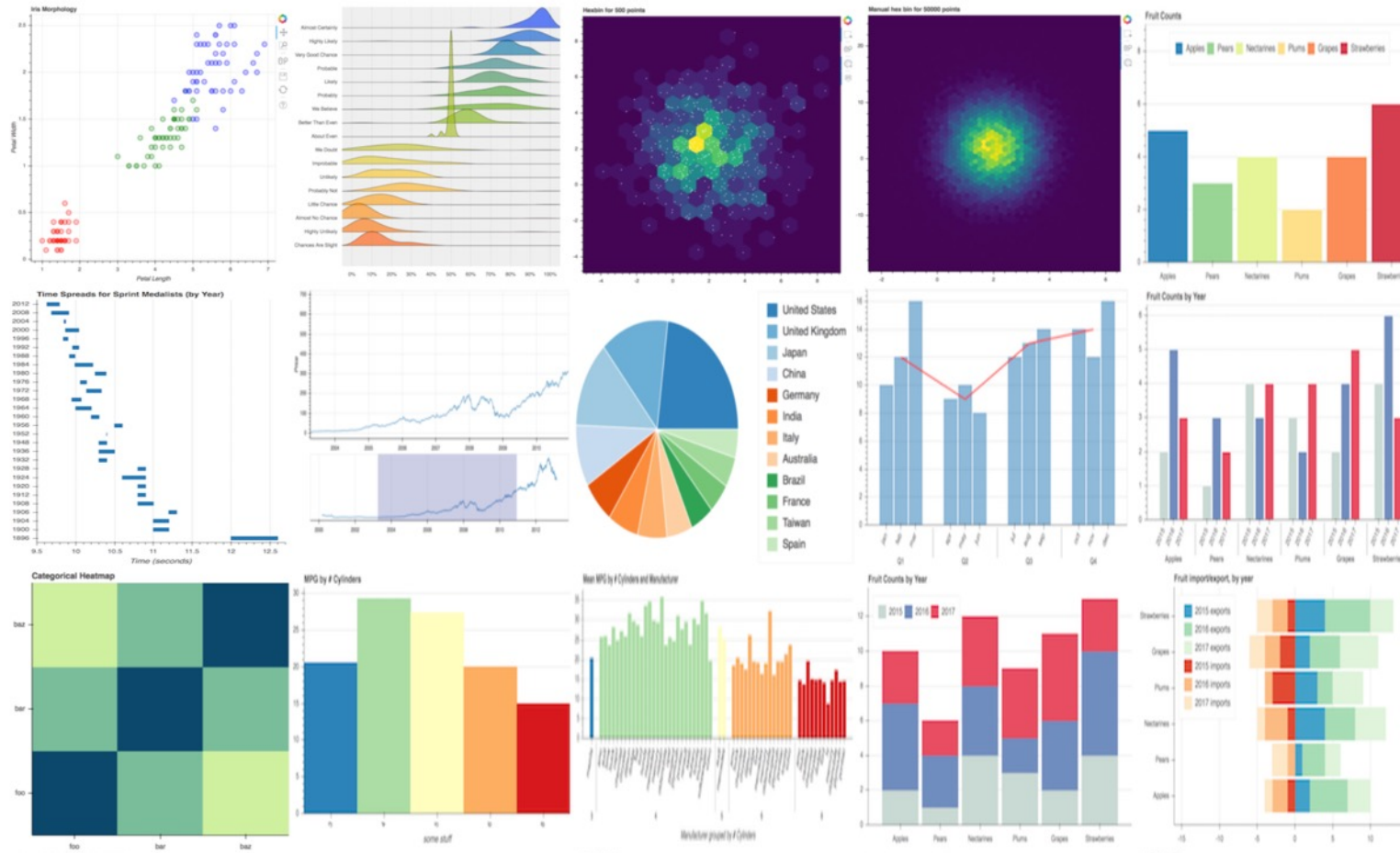
Reference

Developers

Releases

Tutorial

Community





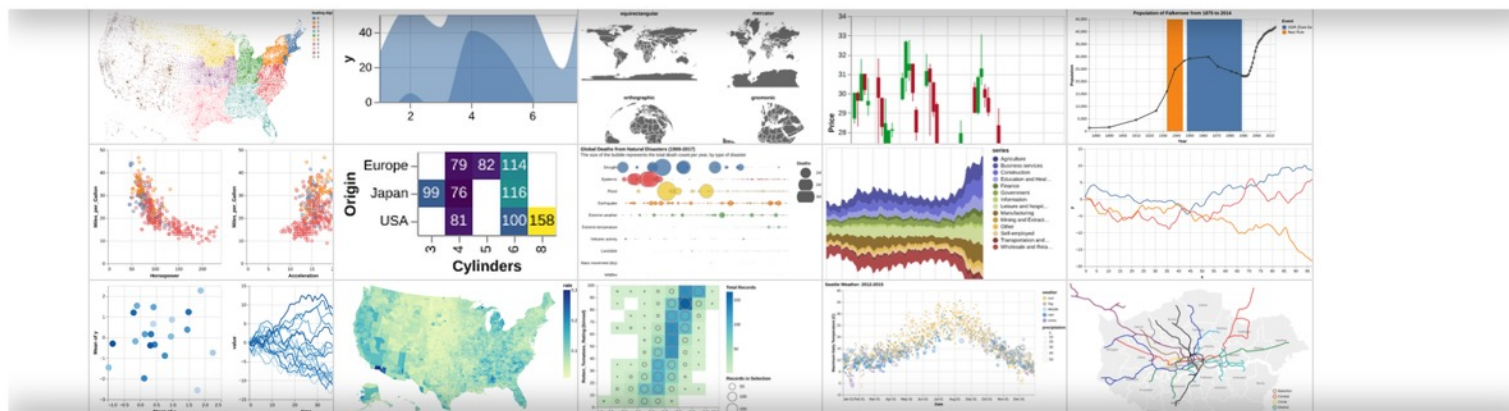
Python Altair



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



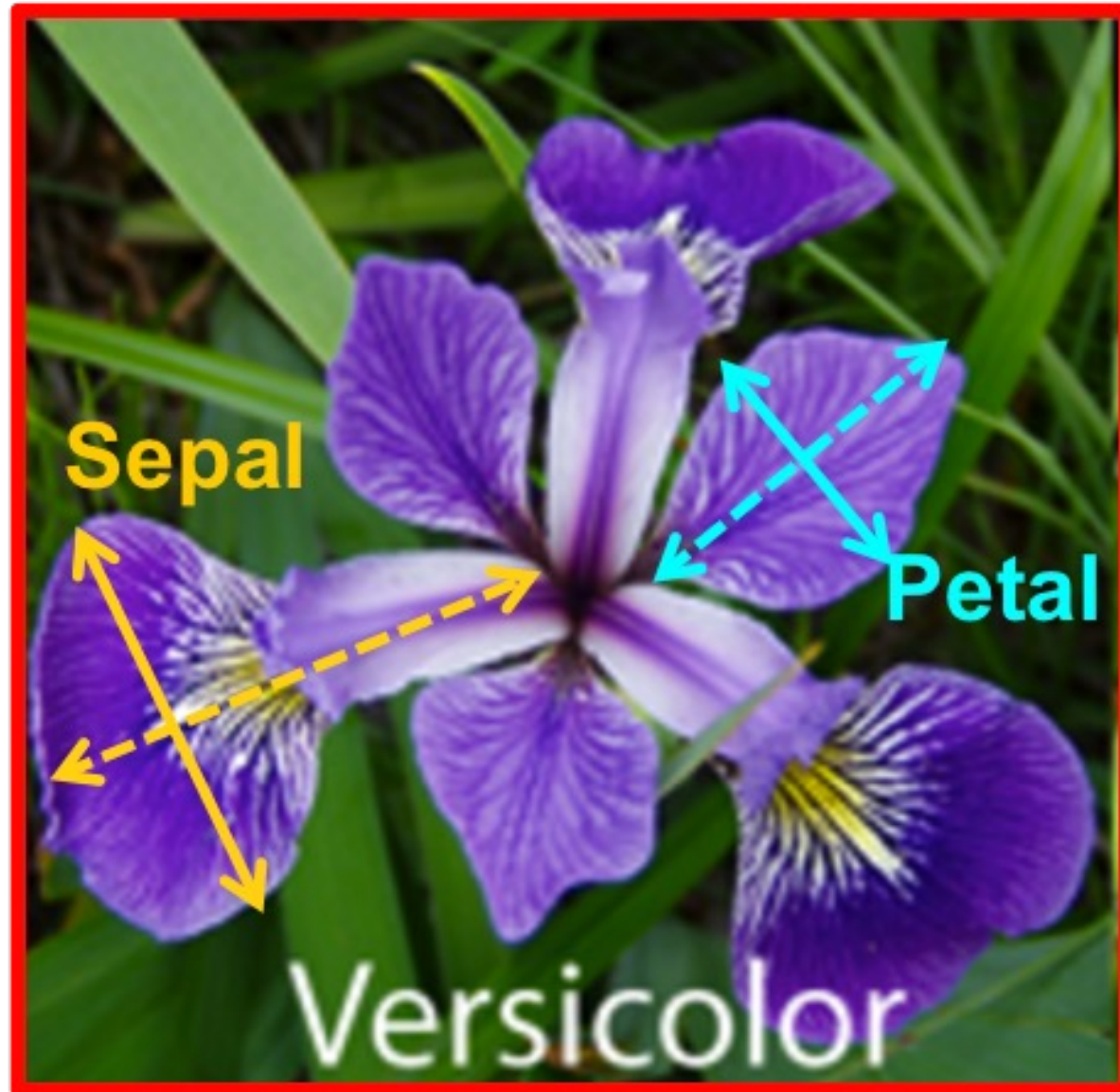
virginica



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

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

Iris Classification



iris.data

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

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

setosa



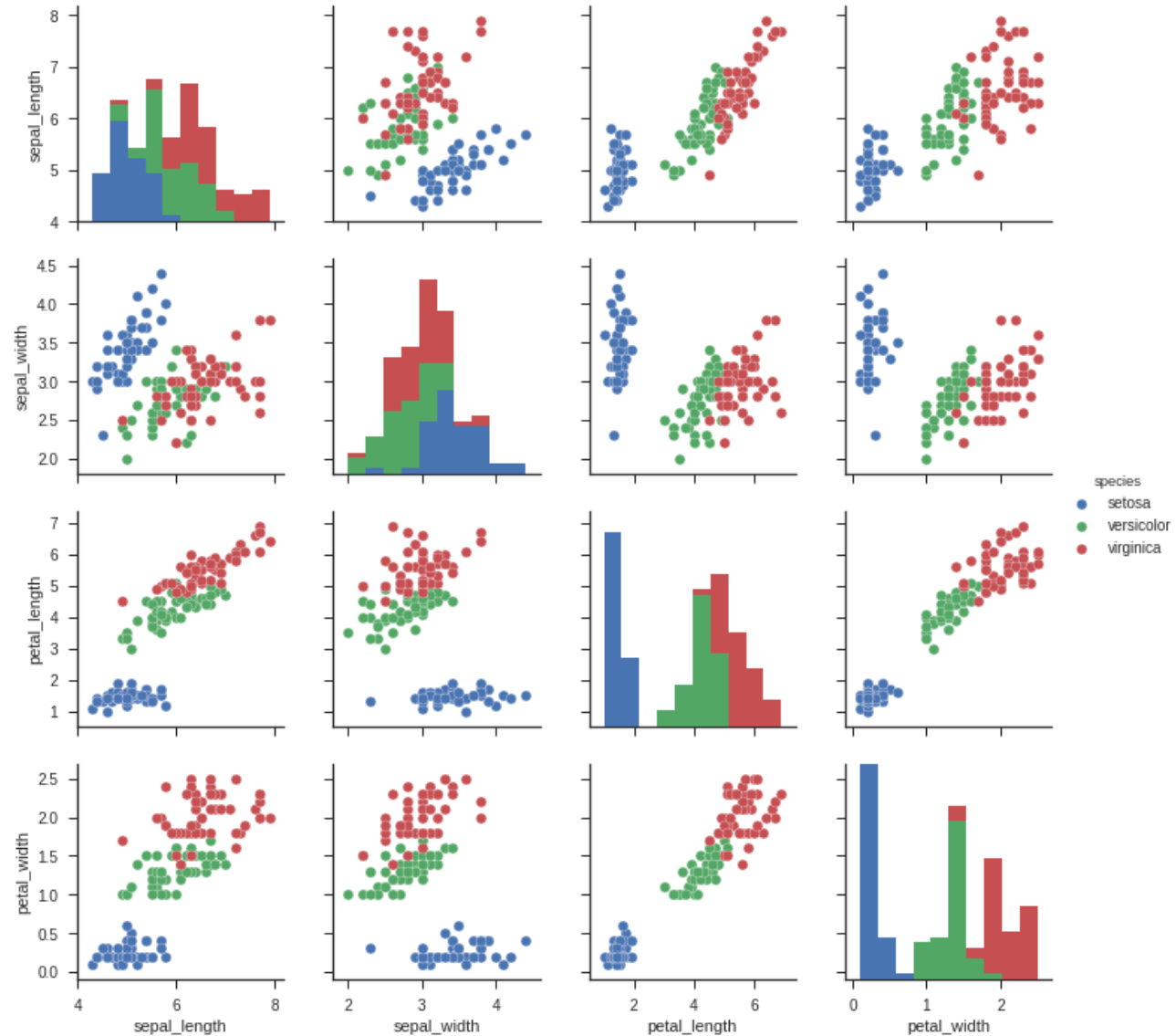
virginica



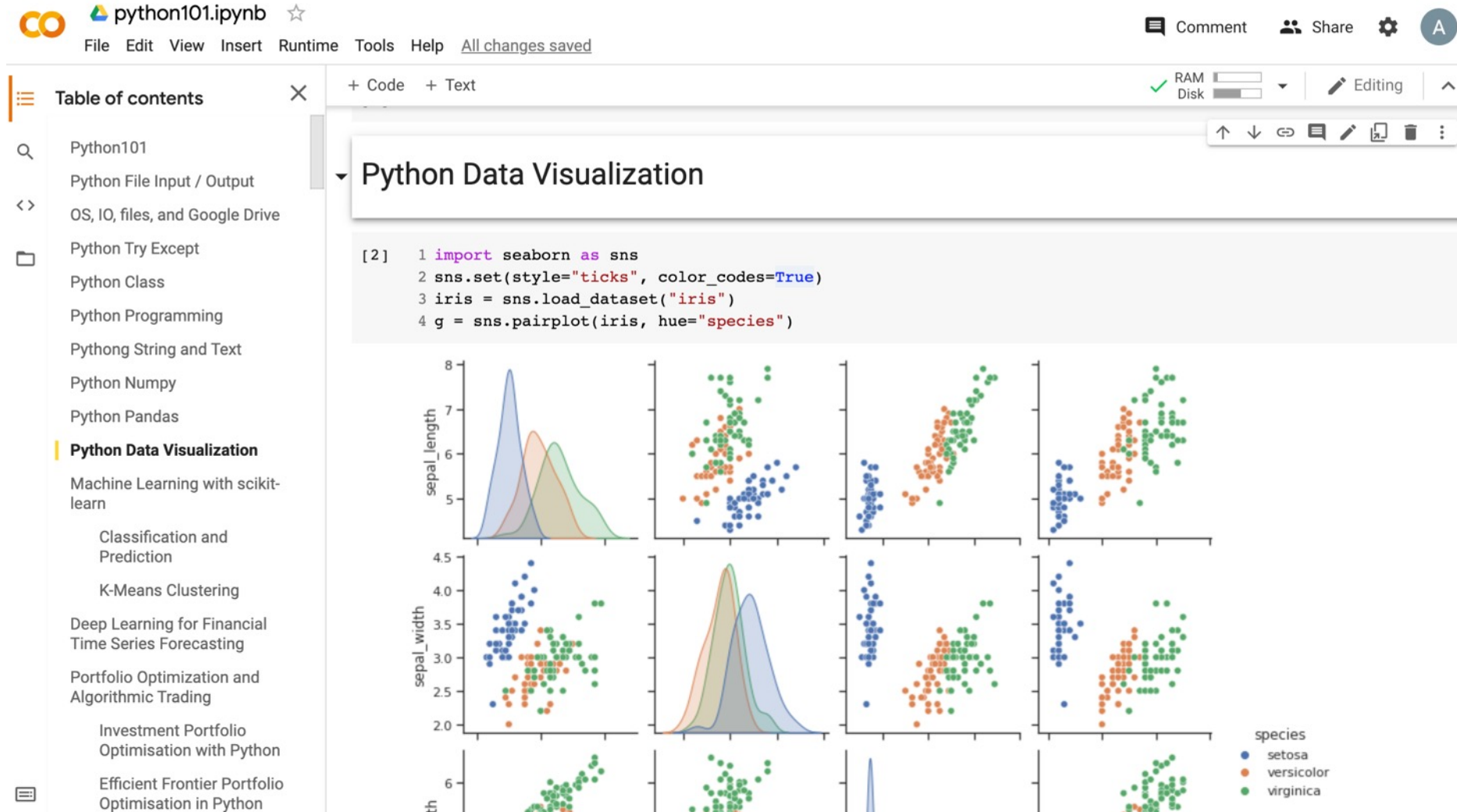
versicolor



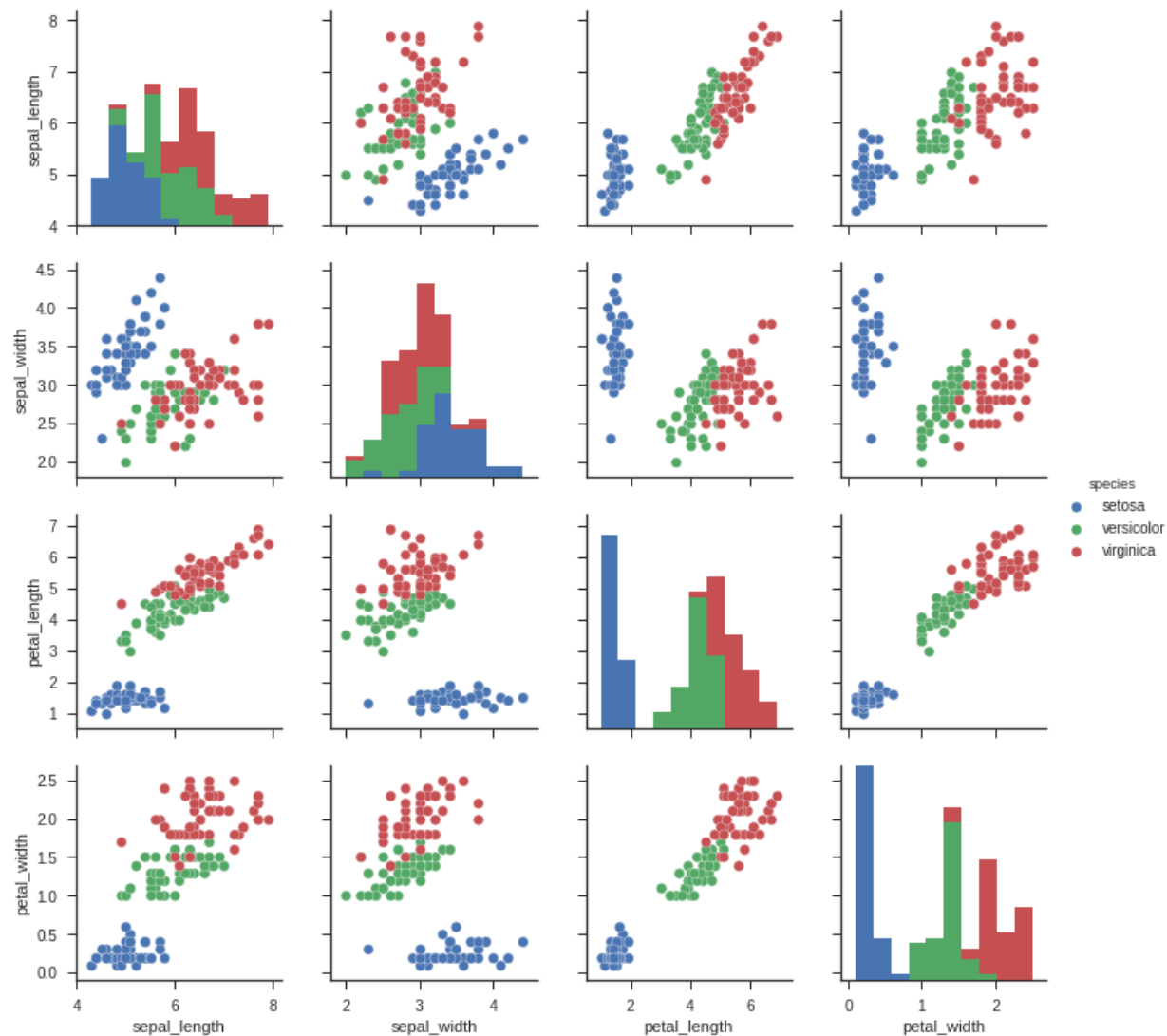
Iris Data Visualization



Data Visualization in Google Colab



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



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

```
# Import Libraries
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
print('imported')
```

imported

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

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

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

df.tail(10)

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

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

df.describe()

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

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


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

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

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

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

```
(150, 5)
```

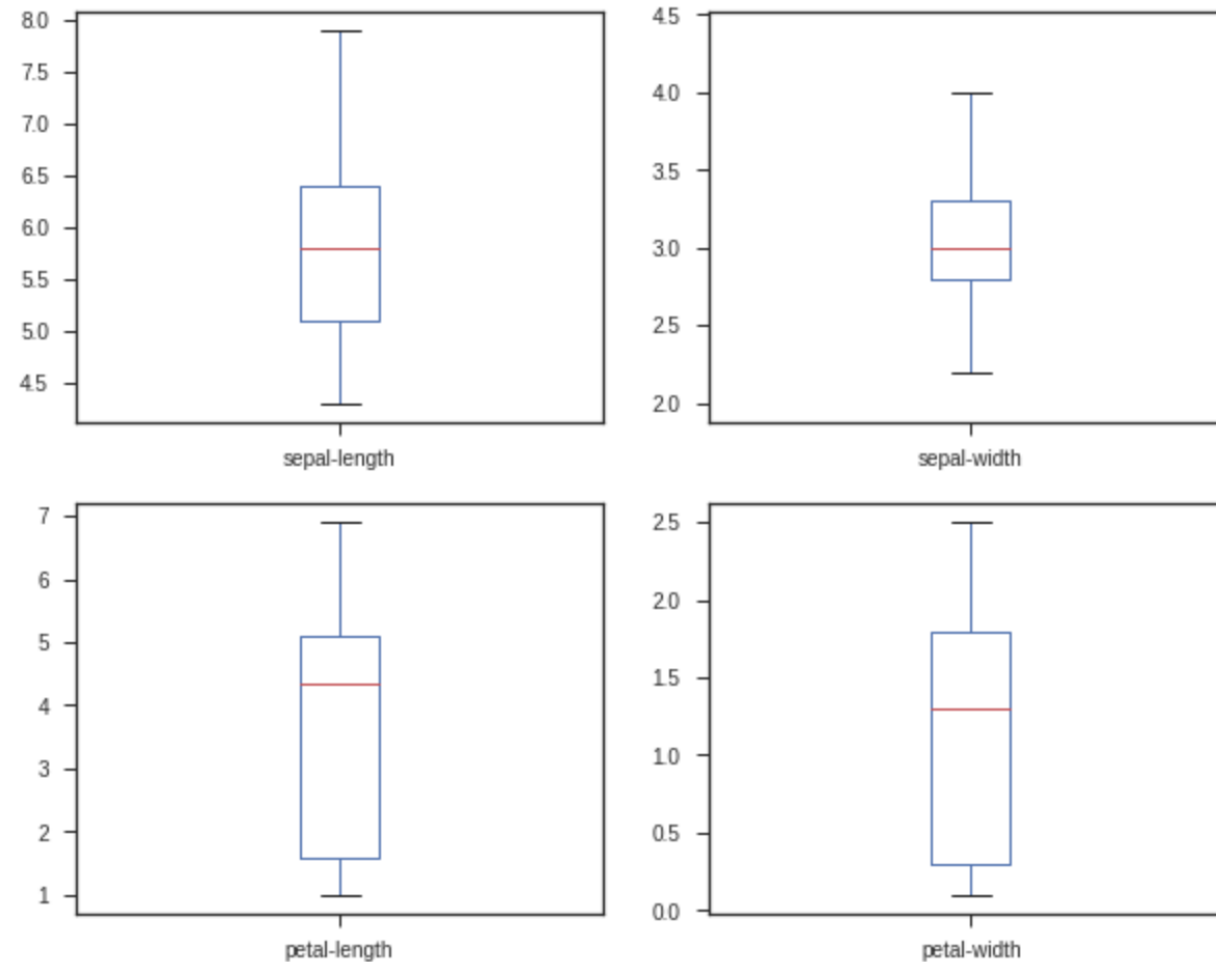
```
df.groupby('class').size()
```

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

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

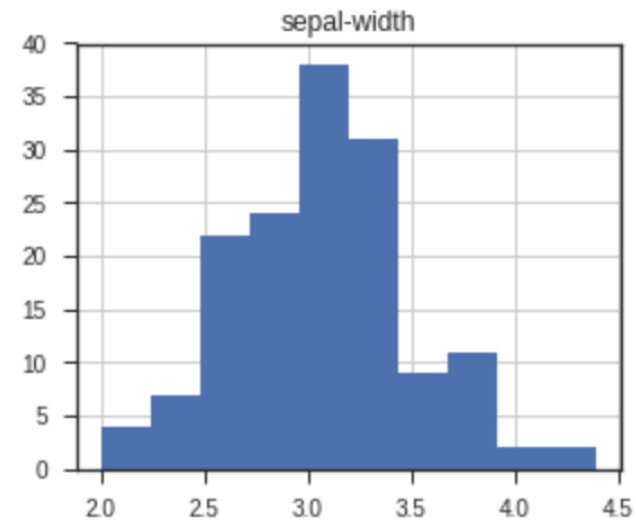
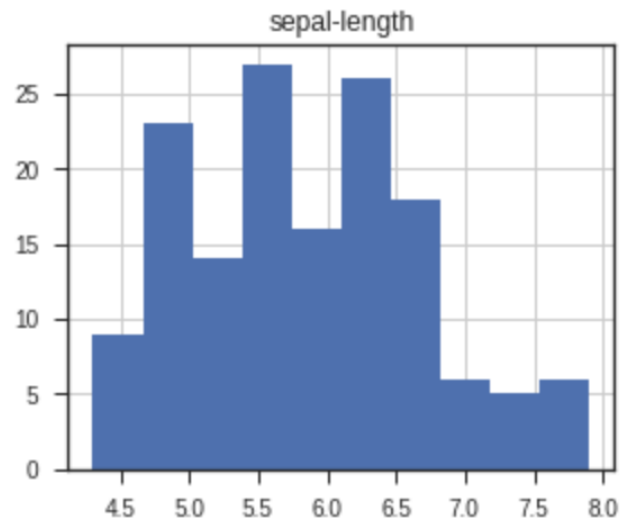
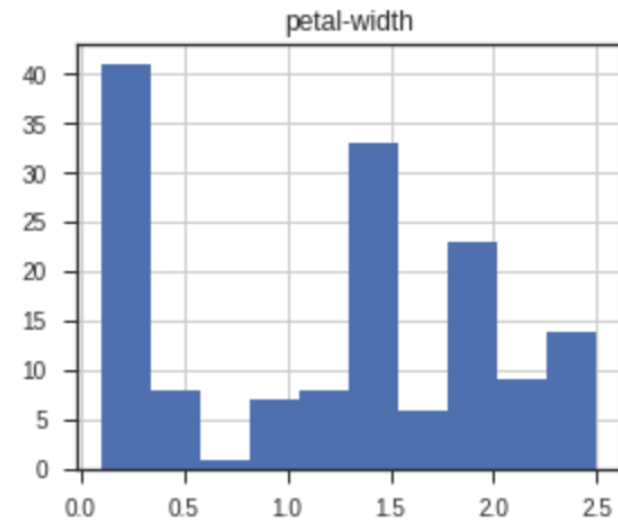
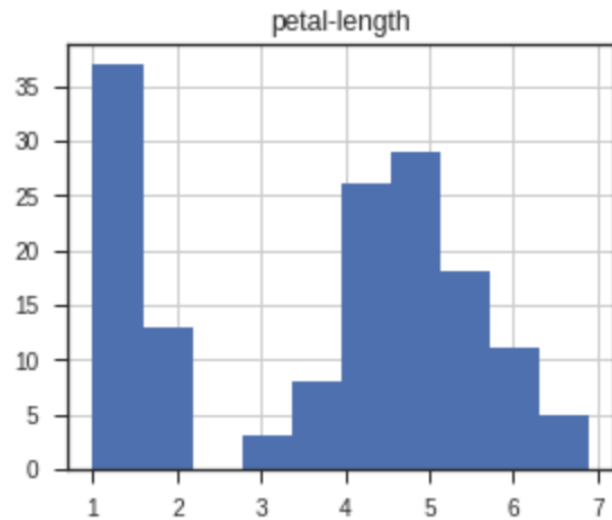
```
plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```

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



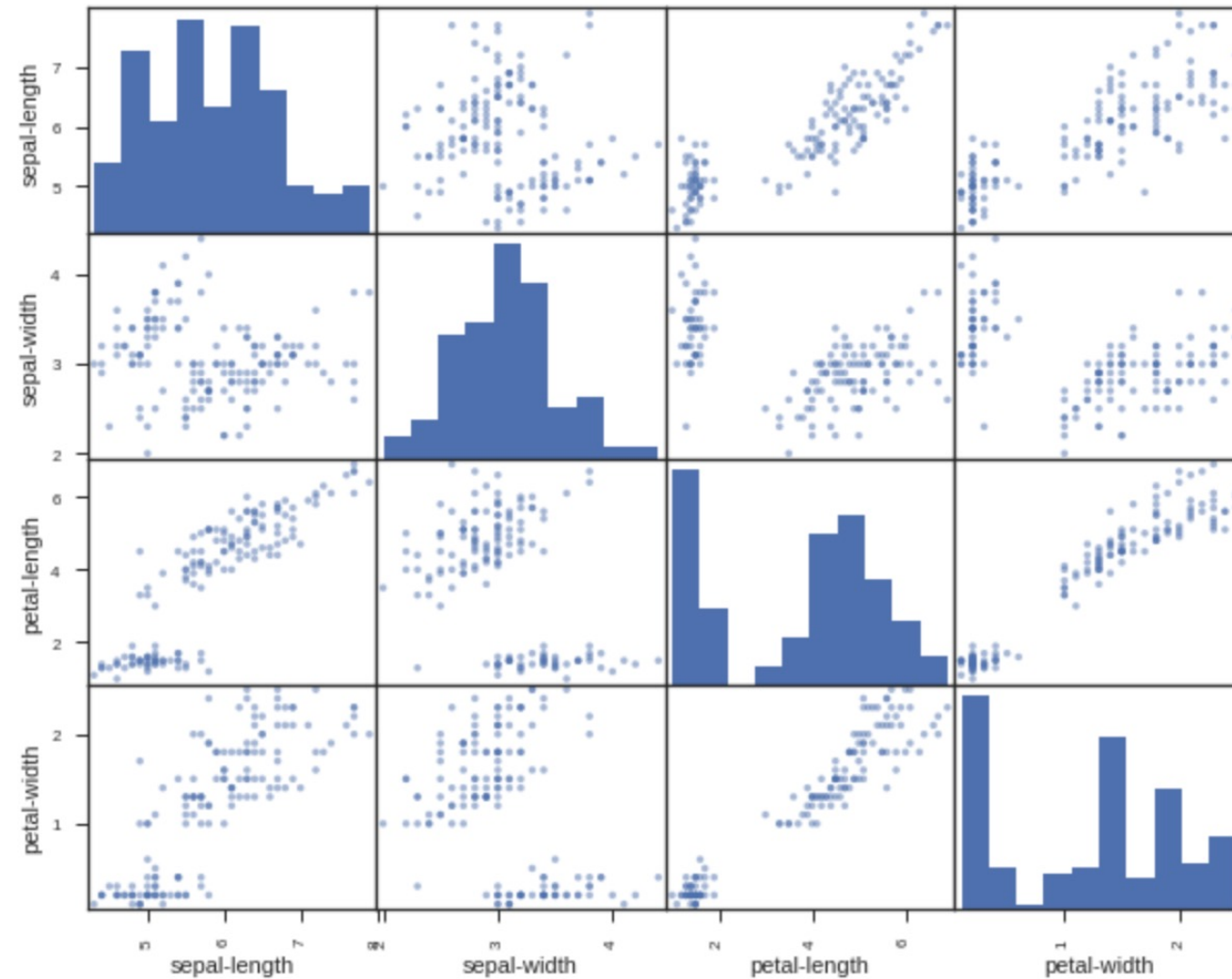
```
df.hist()  
plt.show()
```

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



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

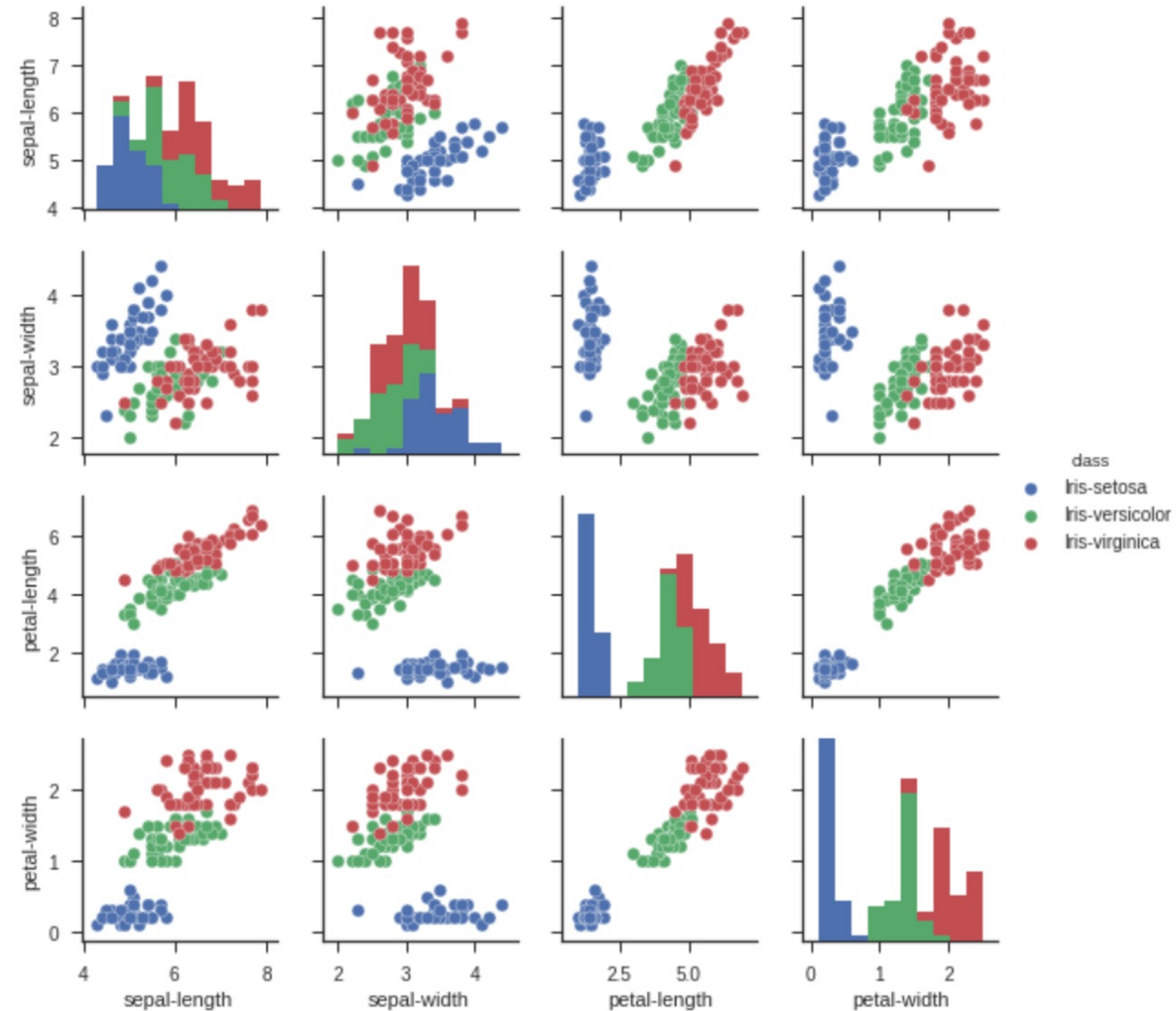
```
scatter_matrix(df)  
plt.show(.)
```



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

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

<seaborn.axisgrid.PairGrid at 0x7f1d21267390>



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 About

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

O'REILLY

Third Edition

Python
for Data Analysis

Data Wrangling with pandas, NumPy & Jupyter



Wes McKinney

<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>

python101.ipynb - Colaboratory

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

python101.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL

CONNECTED EDITING

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

194.87

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

194.87

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

194.87

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

Pass

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

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

Colab interface showing the Python101 notebook. The left sidebar contains a Table of contents with the following items:

- Python101
- Python File Input / Output
- OS, IO, files, and Google Drive
- Python Try Except
- Python Class
- Python Programming
- Pythong String and Text
- Python Numpy
- Python Pandas
- Python Data Visualization**
 - Machine Learning with scikit-learn
 - Classification and Prediction
 - K-Means Clustering
 - Deep Learning for Financial Time Series Forecasting
 - Portfolio Optimization and Algorithmic Trading
 - Investment Portfolio Optimisation with Python
 - Efficient Frontier Portfolio Optimisation in Python

The main content area displays the following code cell:

```
[2] 1 import seaborn as sns
     2 sns.set(style="ticks", color_codes=True)
     3 iris = sns.load_dataset("iris")
     4 g = sns.pairplot(iris, hue="species")
```

Below the code, a pairplot is generated, showing the relationship between the variables sepal_length, sepal_width, and petal_width (labeled as 'th' on the x-axis). The plot is a 3x3 grid of scatter plots, with the diagonal showing kernel density estimates (KDEs) for each variable. The data points are colored by species: setosa (blue), versicolor (orange), and virginica (green). The legend indicates the species for each color.

<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](#)

Summary

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

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.
- 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>
- Min-Yuh Day (2023), Python 101, <https://tinyurl.com/aintpupython101>