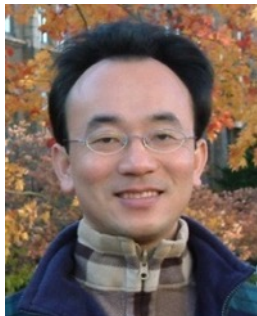


Data Analytics and Visualization with Python

1131PAA06

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>



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

Data Analytics and Visualization with Python

Outline

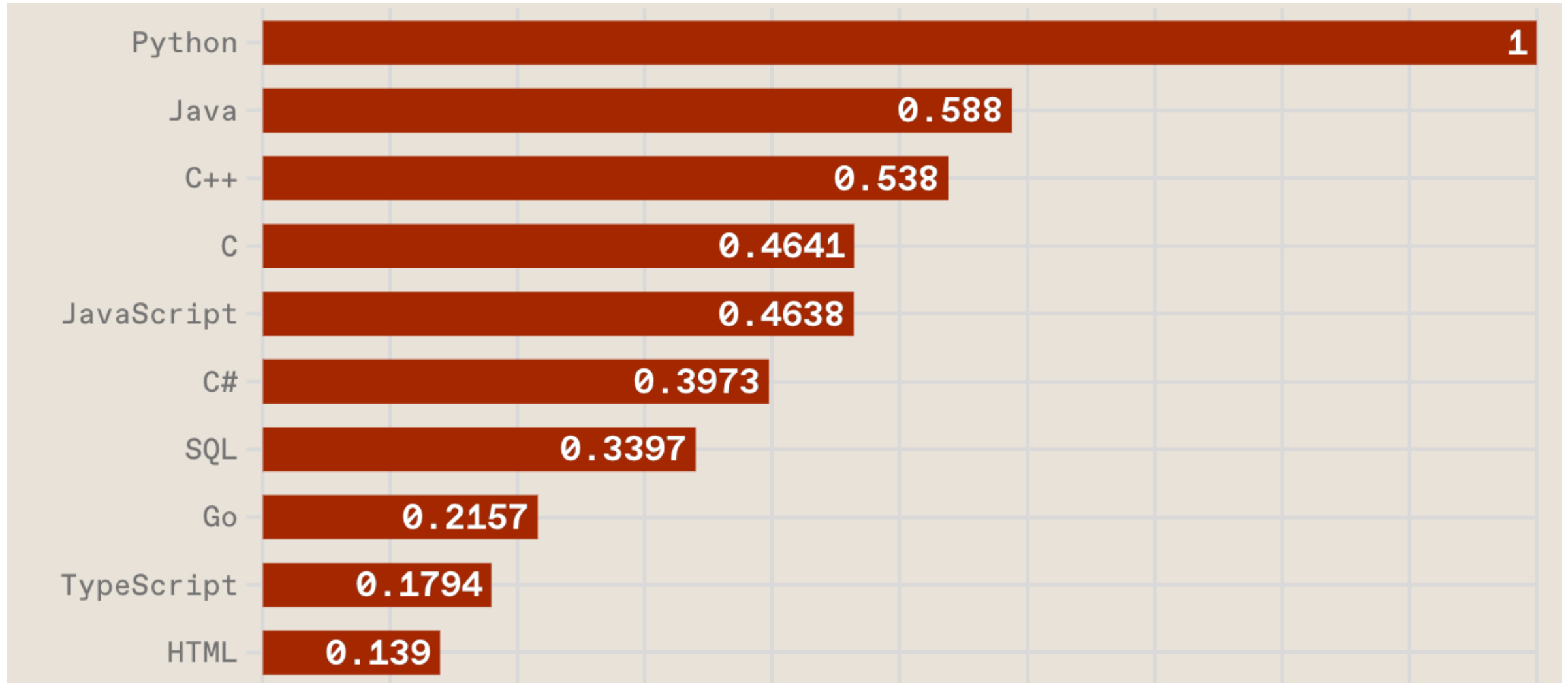
- **NumPy**
 - **Numerical Python N-dimensional array**
- **Pandas**
 - **Data Analytics**
- **Matplotlib**
 - **Basic Data Visualization**
- **Seaborn**
 - **Advanced Visualization**



Python

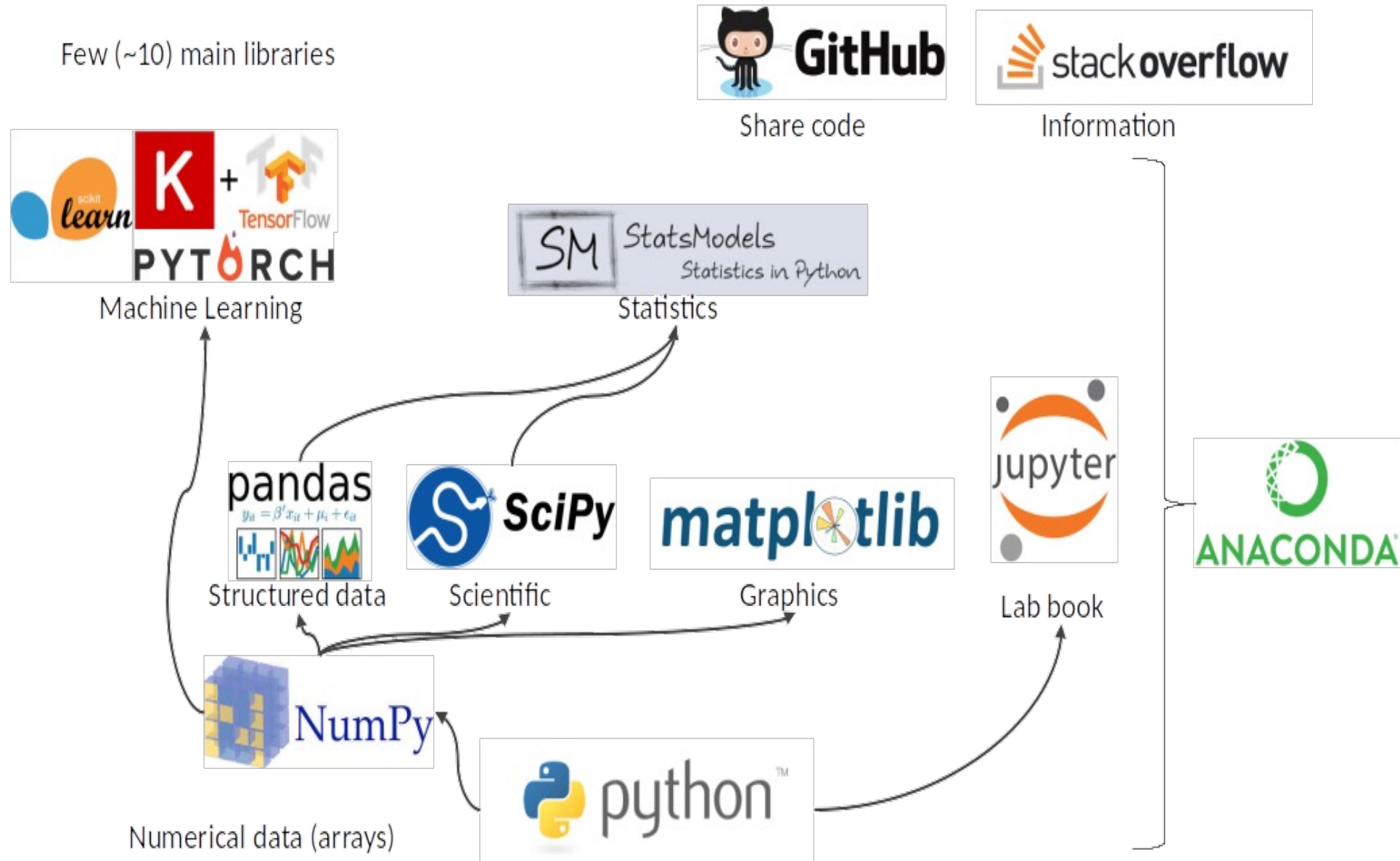
Programming

Top Programming Languages

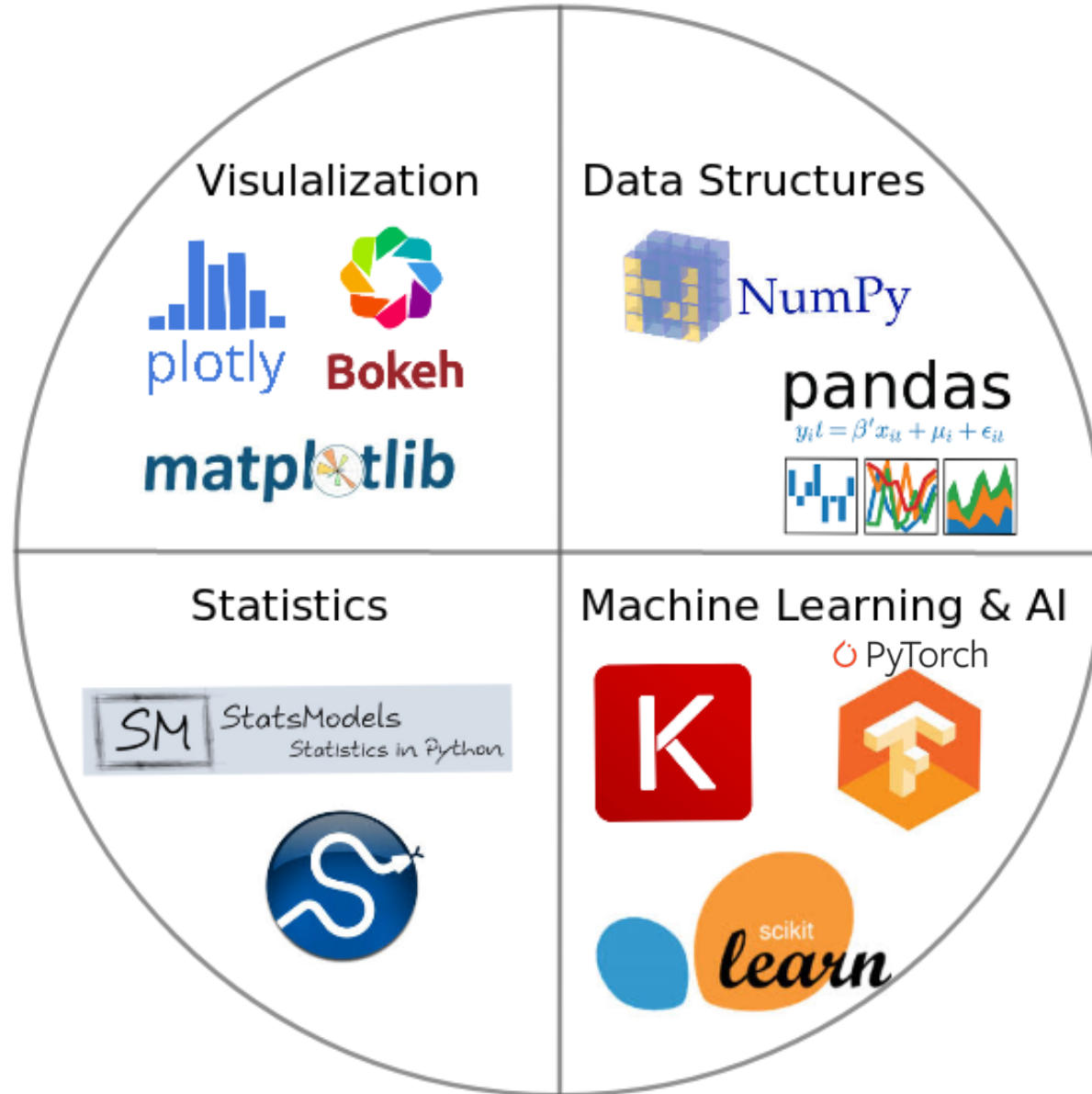


Python is an
interpreted,
object-oriented,
high-level
programming language
with
dynamic semantics.

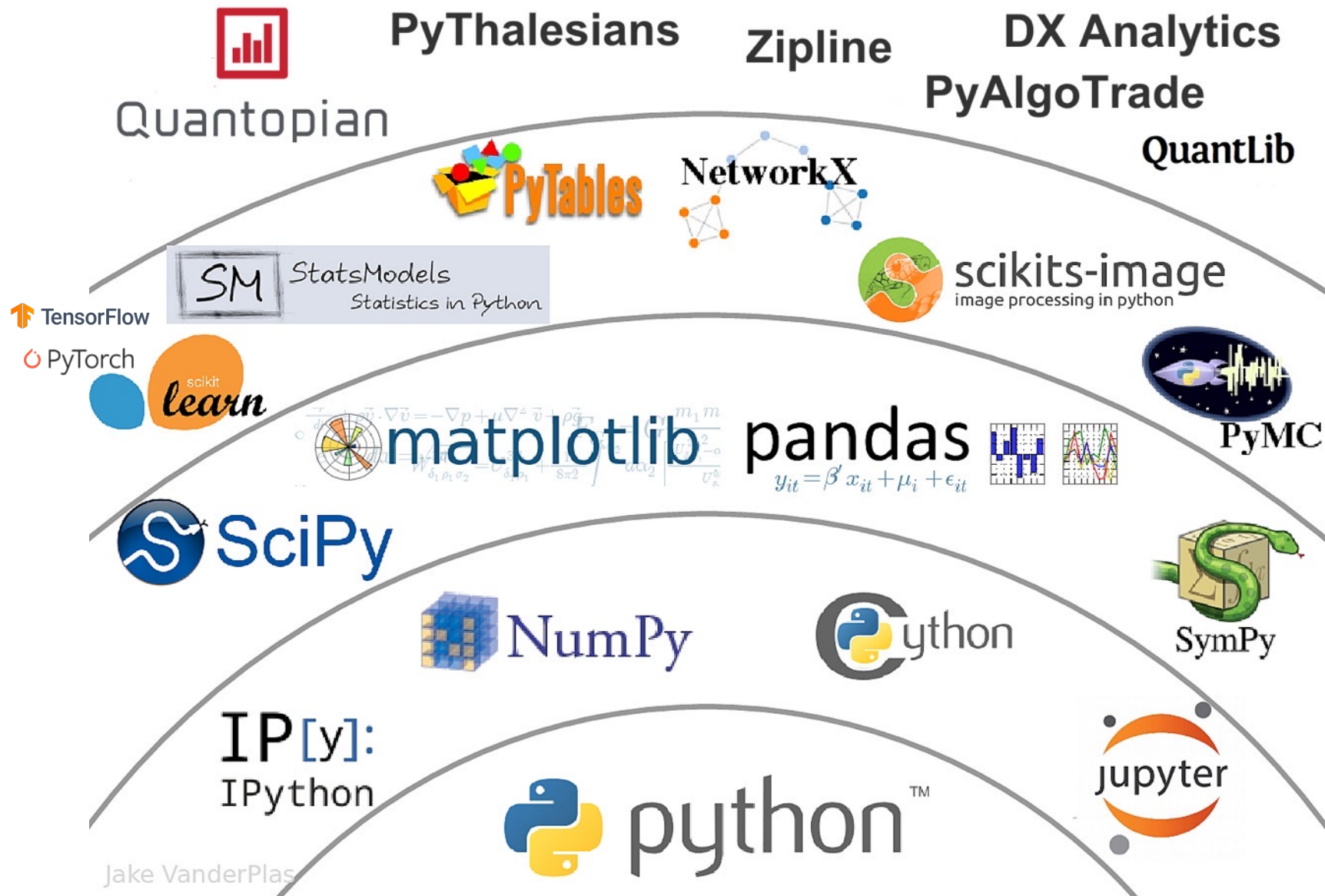
Python Ecosystem for Data Science



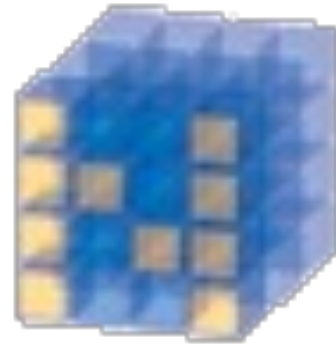
Python Ecosystem for Data Science



The Quant Finance PyData Stack



NumPy



NumPy

Base

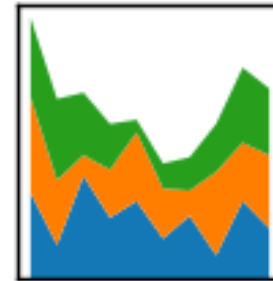
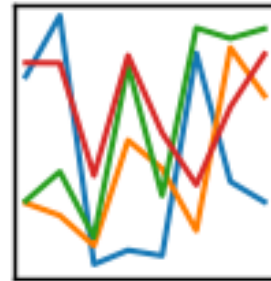
N-dimensional array
package

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

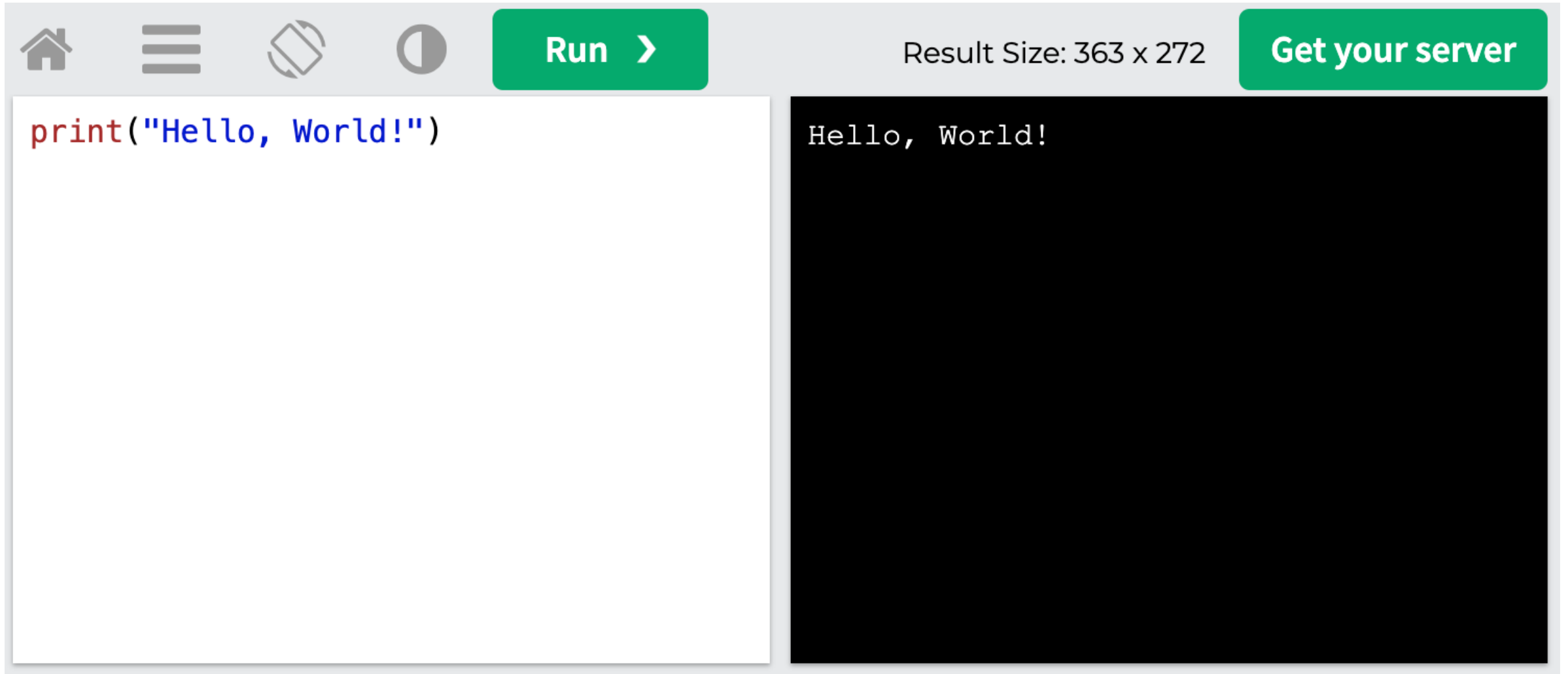
Python is a popular programming language.
Python can be used on a server to create web applications.

[Start learning Python now »](#)

Learning by Examples

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

W3Schools Python: Try Python

A screenshot of the W3Schools Python Try Python interface. The interface has a light gray header with navigation icons (home, menu, refresh, moon) and a green 'Run >' button. To the right of the 'Run' button, it says 'Result Size: 363 x 272' and a green 'Get your server' button. The main area is split into two panels: a white code editor on the left containing the Python code `print("Hello, World!")` and a black terminal window on the right displaying the output '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

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 image shows a browser window with the Google Drive interface. The address bar displays 'https://drive.google.com/drive/u/2/my-drive'. The main navigation bar includes the Drive logo, a search bar, and utility icons. On the left sidebar, the 'New' button is highlighted with a red dashed box. A dropdown menu is open, listing options like 'New folder...', 'Upload files...', 'Google Docs', 'Google Sheets', 'Google Slides', and 'More'. The 'More' option is also highlighted with a red dashed box. A second dropdown menu is open from 'More', listing 'Google Forms', 'Google Drawings', 'Google My Maps', 'Google Sites', and 'Connect more apps'. The 'Connect more apps' option is highlighted with a red dashed box. The main content area shows a 'Files' section with a 'Name' column header and an upward arrow. A storage usage indicator shows '0 bytes of 15 GB used' with an 'UPGRADE STORAGE' link. A notification at the bottom left says 'Get Backup and Sync for Mac'.

Google Colab

The screenshot shows the Google Drive web interface. A dialog box titled "Connect apps to Drive" is open, displaying a search for "colab". The search results are as follows:

App Name	Users
ZIP Extractor	307,585 users
Lumin PDF - Beautiful PDF Editor	289,310 users
CloudConvert	373,161 users
Sejda	2,131,600 users
DocHub - Edit and Sign PDF Docu...	2,131,600 users
Google Forms	4,803,614 users

Google Colab

The image shows a browser window with the Google Drive interface. A dialog box titled "Connect apps to Drive" is open in the center. The dialog has a search bar at the top with the text "colab" entered. Below the search bar, there is a list of search results. The first result is for "Colaboratory", which is highlighted with a red dashed border. The result includes the Colaboratory logo (two yellow circles), the name "Colaboratory", the URL "https://colab.research.google.com", a description: "A data analysis tool that combines code, output, and descriptive text into one collaborative document.", and a rating of five stars with "(195)" reviews. A blue button with a white plus sign and the text "+ CONNECT" is positioned to the right of the result. The background shows the Google Drive sidebar with options like "New", "My Drive", "Computers", "Shared with me", "Recent", "Starred", "Trash", "Backups", and "Storage". The top of the browser shows the address bar with the URL "https://drive.google.com/drive/u/2/my-drive".

Connect Colaboratory to Google Drive

The screenshot shows the Google Drive web interface. A dialog box titled "Connect apps to Drive" is open, displaying a search for "colab". A confirmation message from Colaboratory is shown in the center, stating "Colaboratory was connected to Google Drive." and "Make Colaboratory the default app for files it can open" with a checked checkbox. An "OK" button is highlighted with a red dashed border. The background shows the Drive sidebar with categories like "My Drive", "Computers", "Shared with me", "Recent", "Starred", "Trash", "Backups", and "Storage".

Google Colab

The image shows a browser window with the Google Drive interface. The address bar displays 'https://drive.google.com/drive/u/2/my-drive'. The main content area shows the 'New' button, which has been clicked to reveal a dropdown menu. The 'My Drive' folder is highlighted in the left sidebar. The 'New' menu is open, showing options like 'New folder...', 'Upload files...', 'Upload folder...', 'Google Docs', 'Google Sheets', 'Google Slides', 'More', 'Google Forms', 'Google Drawings', 'Google My Maps', 'Google Sites', 'Colaboratory', and 'Connect more apps'. The 'Colaboratory' option is highlighted with a red dashed box. The 'More' option in the 'New' menu is also highlighted with a red dashed box. The 'Storage' section shows '0 bytes of 15 GB used' and a 'UPGRADE STORAGE' button. The 'Files' section shows 'Store safely', 'Sync seamlessly', 'Access anywhere', and 'Share easily'.

My Drive - Google Drive

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

Drive

Search Drive

My Drive

Quick Access

New

My Drive

Computers

Shared with me

Recent

Starred

Trash

Backups

Storage

0 bytes of 15 GB used

UPGRADE STORAGE

Get Backup and Sync for Mac

New folder...

Upload files...

Upload folder...

Google Docs

Google Sheets

Google Slides

More

Google Forms

Google Drawings

Google My Maps

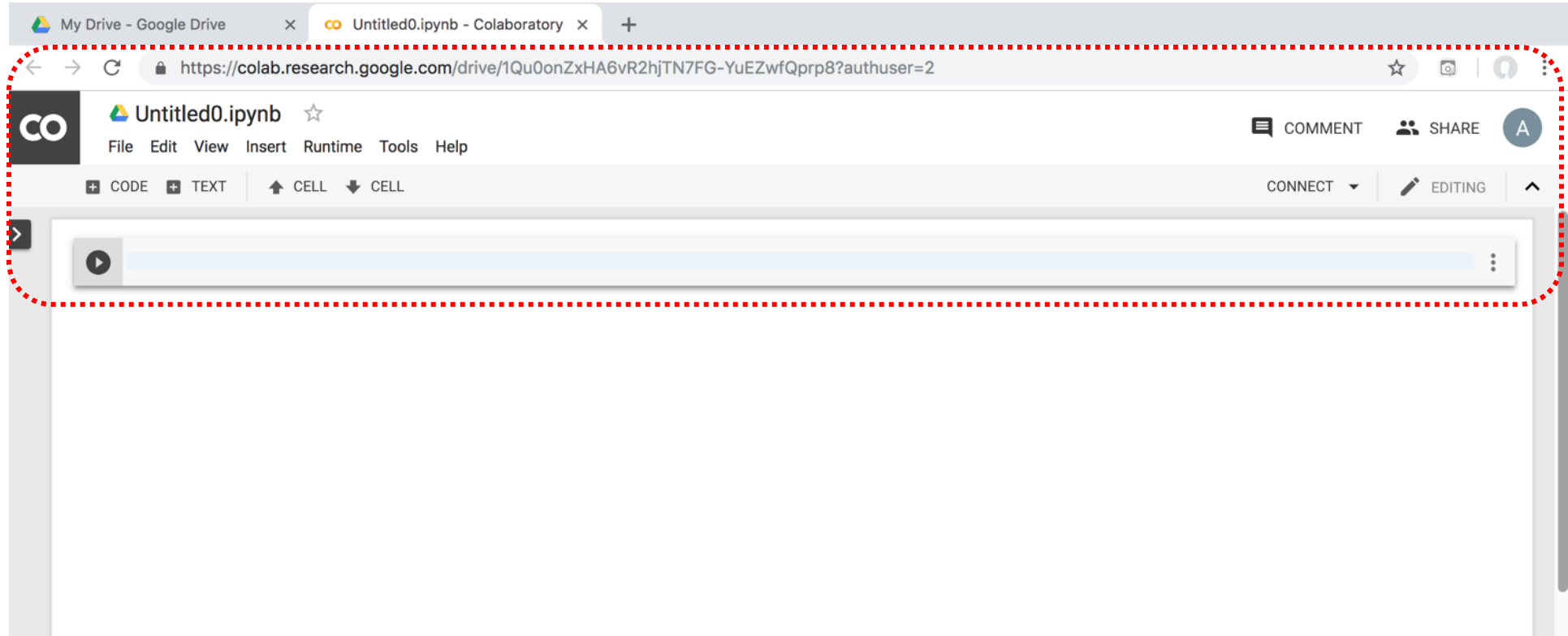
Google Sites

Colaboratory

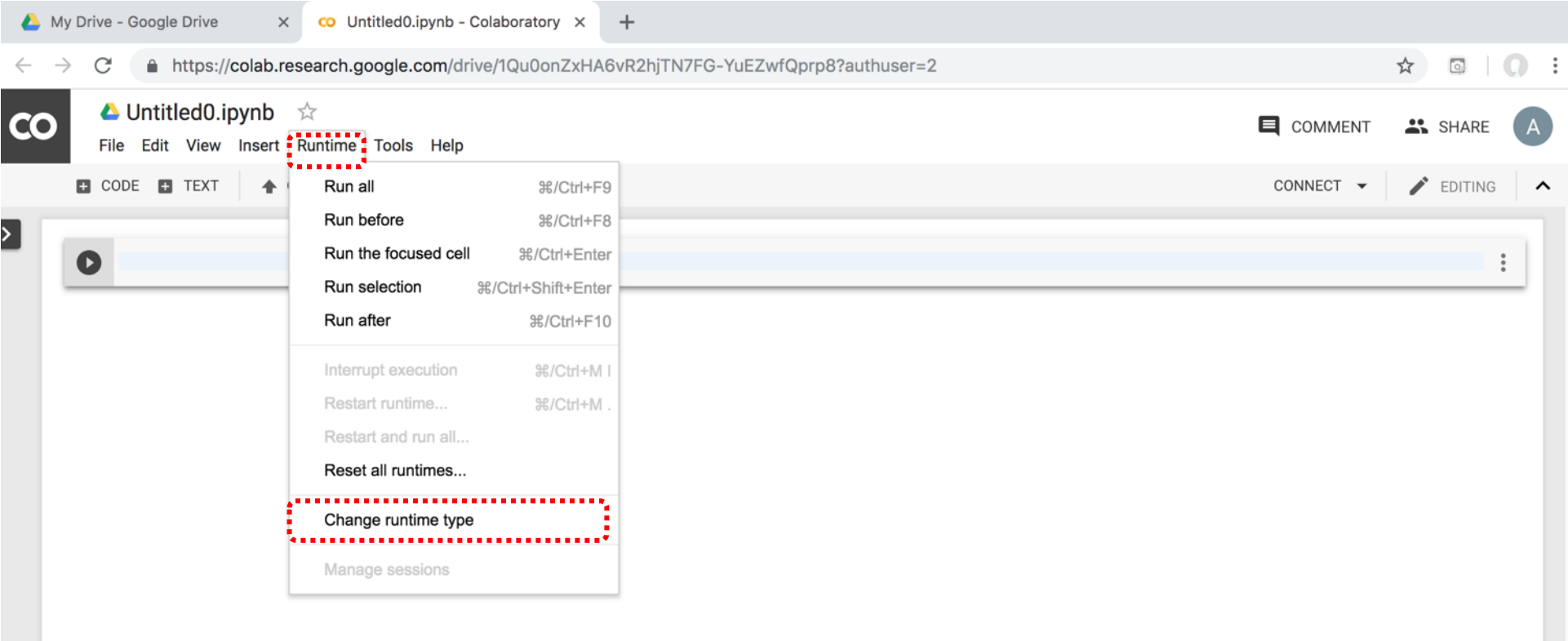
Connect more apps

Name ↑

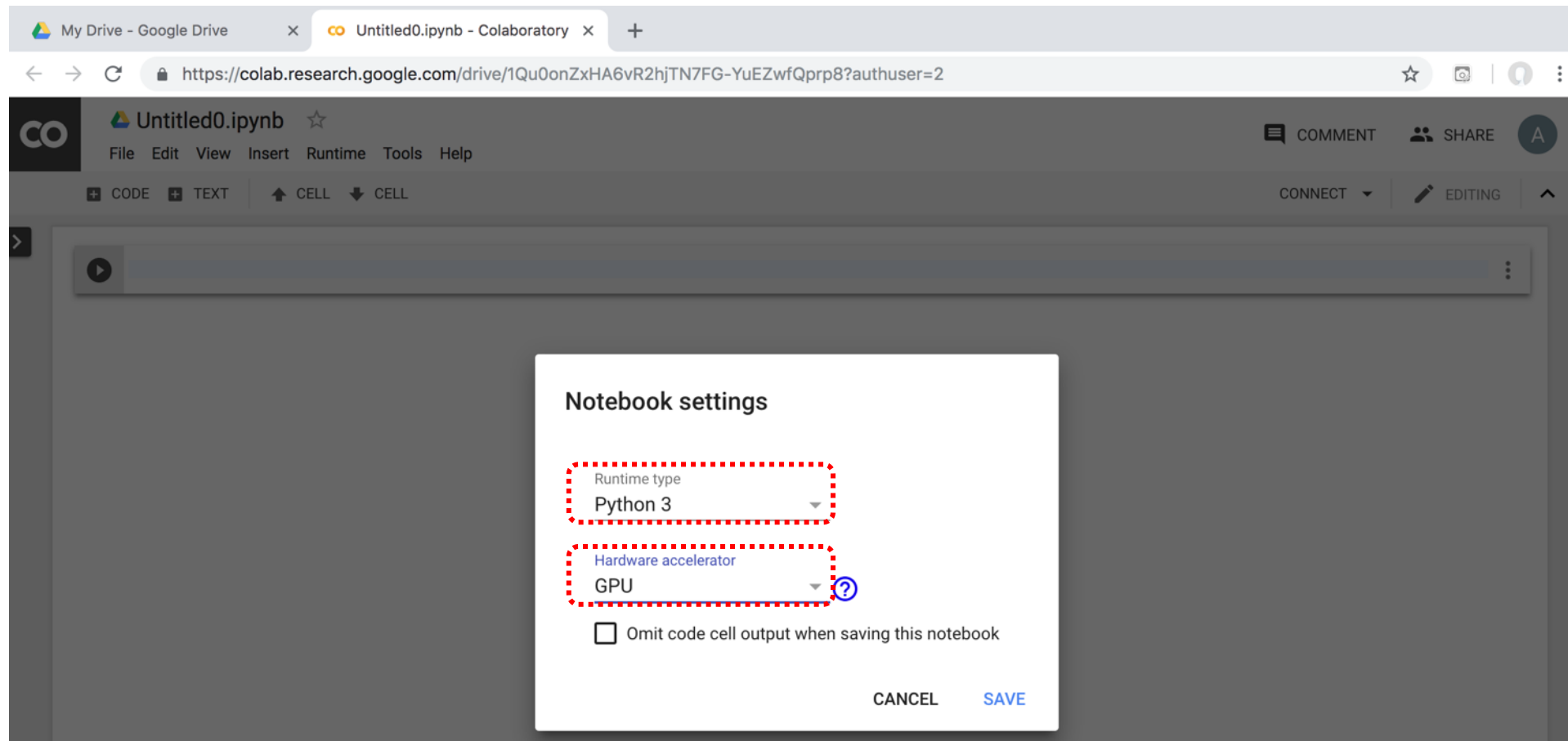
Google Colab



Google Colab



Run Jupyter Notebook Python3 GPU Google Colab



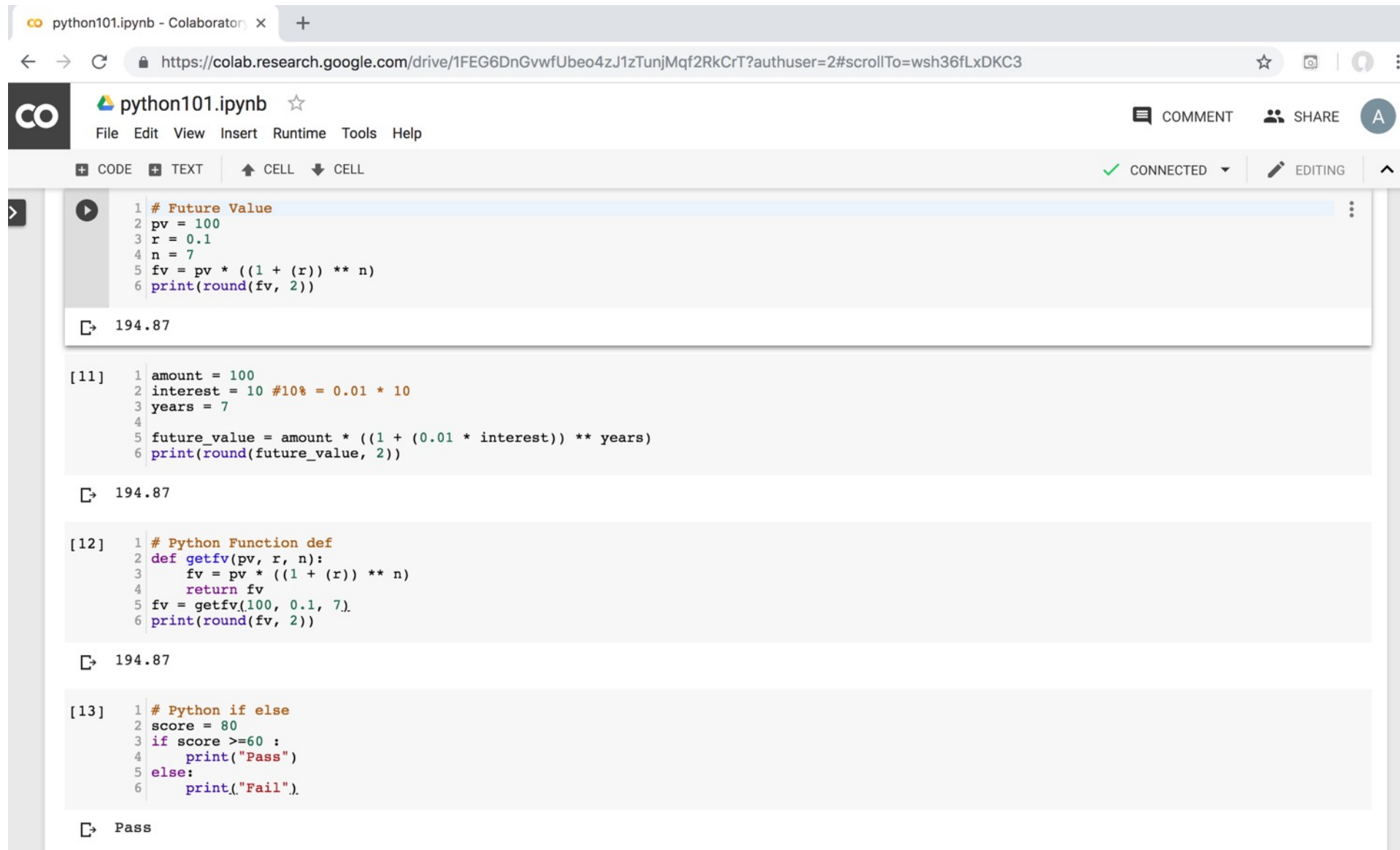
Google Colab Python Hello World

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



Python in Google Colab (Python101)

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



The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL: <https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT?authuser=2#scrollTo=wsh36fLxDKC3>. The notebook title is "python101.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with options for CODE, TEXT, CELL, and a status indicator showing "CONNECTED" and "EDITING".

The notebook contains four code cells, each followed by its output:

```
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/aintpupython101>



Anaconda
The Most Popular
Python
Data Science Platform

Download Anaconda



Products ▾

Pricing

Solutions ▾

Resources ▾

Partners ▾

Blog

Company ▾

Contact Sales

Data science technology for a better world.

Anaconda offers the easiest way to perform Python/R data science and machine learning on a single machine. Start working with thousands of open-source packages and libraries today.

Download 

Get Additional Installers



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

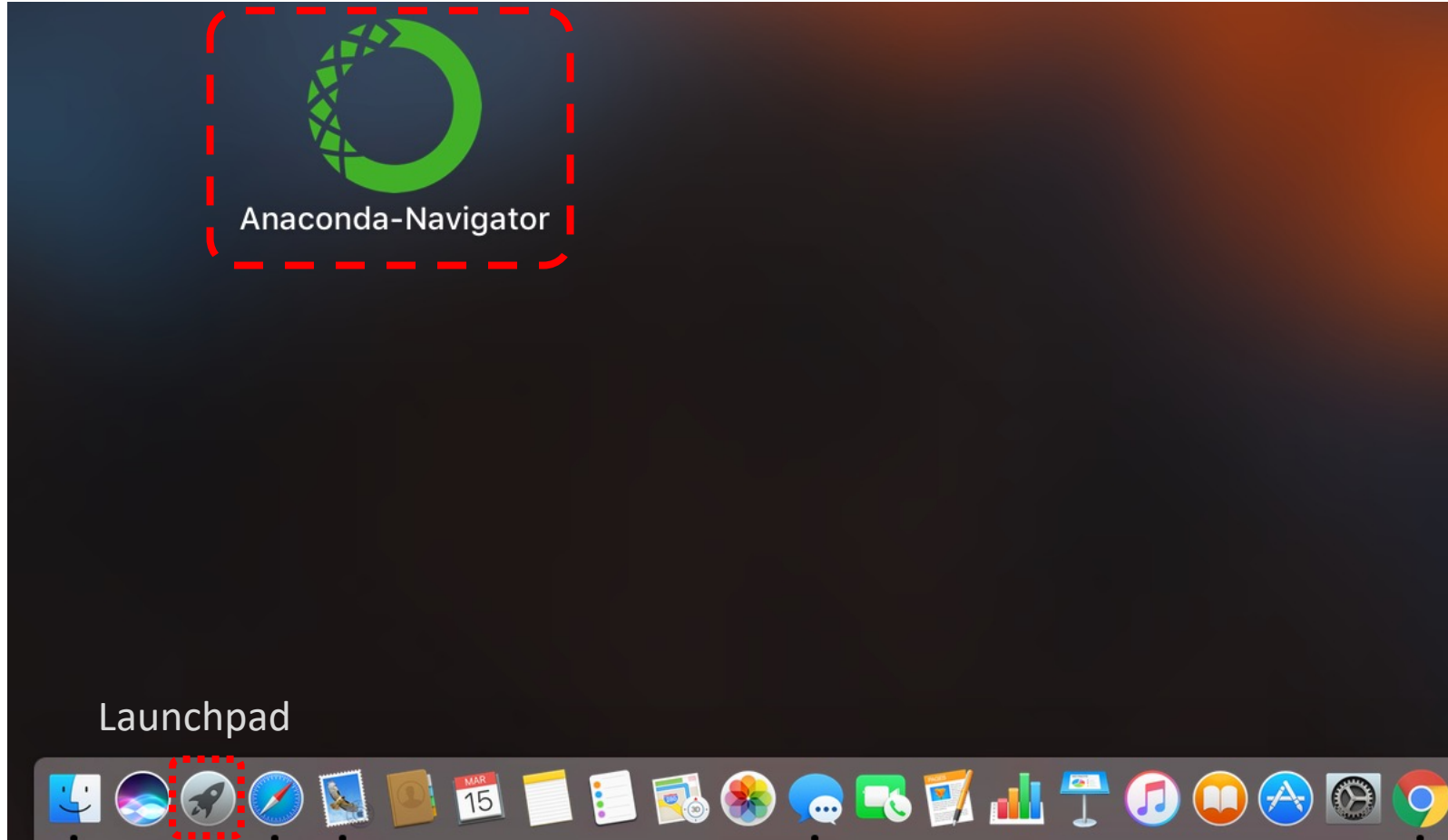




Python

HelloWorld

Anaconda-Navigator



Anaconda Navigator

The screenshot displays the Anaconda Navigator desktop application. At the top, the title bar reads "Anaconda Navigator". Below it, the application header features the "ANACONDA NAVIGATOR" logo on the left and a "Sign in to Anaconda Cloud" button on the right. A left-hand sidebar contains navigation options: "Home", "Environments", "Learning", and "Community". At the bottom of the sidebar are links for "Documentation", "Developer Blog", and "Feedback", along with social media icons for Twitter, YouTube, and GitHub.

The main content area is titled "Applications on" and shows a dropdown menu set to "base (root)" and a "Channels" button. A "Refresh" button is located in the top right of this section. The applications are arranged in a grid:

- jupyterlab** (0.31.5): An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. [Launch]
- jupyter notebook** (5.4.0): Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. [Launch]
- qtconsole** (4.3.1): PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. [Launch]
- spyder** (3.2.6): Scientific PYTHON Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features. [Launch]
- vscode** (1.22.2): Streamlined code editor with support for development operations like debugging, task running and version control. [Launch]
- glueviz** (0.12.4): Multidimensional data visualization across files. Explore relationships within and among related datasets. [Install]

The "jupyter notebook" application card is highlighted with a red dashed border, and its "Launch" button is enclosed in a solid red box.

Jupyter Notebook

Home x

localhost:8888/tree/Documents/Data/BDA

jupyter Logout

Files Running Clusters

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

<input type="checkbox"/> 0	/ Documents / Data / BDA	Name ↓	Last Modified
<input type="checkbox"/>	..		seconds ago
The notebook list is empty.			

Jupyter Notebook

New Python 3

The screenshot shows a web browser window with the Jupyter Notebook interface. The browser's address bar displays `localhost:8888/tree/Documents/Data/BDA`. The Jupyter logo and a 'Logout' button are visible at the top. Below the logo, there are tabs for 'Files', 'Running', and 'Clusters'. A message says 'Select items to perform actions on them.' To the right of this message are 'Upload', 'New', and a refresh icon. The 'New' dropdown menu is open, showing options: 'Notebook: Python 3', 'Other: Text File', 'Folder', and 'Terminal'. The 'Python 3' option is highlighted with a red box. The file browser below shows the path `/ Documents / Data / BDA` and a message 'The notebook list is empty.'

print("hello, world")

The image shows a Jupyter Notebook interface in a web browser. The browser tabs include 'Home' and 'HelloWorld'. The address bar shows the URL 'localhost:8888/notebooks/Documents/Data/BDA/HelloWorld.ipynb'. The Jupyter logo and 'HelloWorld (autosaved)' are visible in the top left, along with a 'Logout' button and a Python logo. A menu bar contains 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. Below the menu is a toolbar with icons for saving, adding, deleting, copying, pasting, and running. The 'Run' button is highlighted with a red box. The main area contains a code cell with the text 'In [1]: print("hello, world")' and its output 'hello, world'. The code line is also highlighted with a red box. Below the code cell is an empty input field labeled 'In []:'.





Python

Programming

Foundations of Python Programming

- **Python Syntax**
 - **Python Comments**
- **Python Variables**
- **Python Data Types**
 - **Python Numbers**
 - **Python Casting**
 - **Python Strings**
- **Python Operators**
- **Python Booleans**

Python Hello World

```
print("Hello World")
```

```
print("Hello World")
```

Python Syntax

comment

```
# comment
```

Python Syntax

Indentation

the spaces at the beginning of a code line
4 spaces

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

Python Variables

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

Python Variables

```
x = 2  
y = x + 1
```

python_version()

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

Python Version: 3.10.12

Python Data Types

```
x = "Hello World"    #str
x = 2                #int
x = 2.5              #float
x = 7j                #complex
```

Python Data Types

```
x = ["apple", "banana", "cherry"] #list
x = ("apple", "banana", "cherry") #tuple
x = range(6) #range
x = {"name" : "Tom", "age" : 20} #dict
x = {"apple", "banana", "cherry"} #set
x = frozenset({"apple", "banana", "cherry"})
#frozenset
```

Python Data Types

```
x = True #bool
x = b"Hello" #bytes
x = bytearray(5) #bytearray
x = memoryview(bytes(5)) #memoryview
x = None #NoneType
```

Python Casting

```
x = str(3) # x will be '3'  
y = int(3) # y will be 3  
z = float(3) # z will be 3.0  
print(x, type(x))  
print(y, type(y))  
print(z, type(z))
```

```
3 <class 'str'>  
3 <class 'int'>  
3.0 <class 'float'>
```

Python Numbers

```
x = 2 # int
y = 3.4 # float
z = 7j #complex
print(x, type(x))
print(y, type(y))
print(z, type(z))
```

```
2 <class 'int'>
3.4 <class 'float'>
7j <class 'complex'>
```

Python Arithmetic Operators

Operator	Name	Example
+	Addition	$7 + 2 = 9$
-	Subtraction	$7 - 2 = 5$
*	Multiplication	$7 * 2 = 14$
/	Division	$7 / 2 = 3.5$
//	Floor division	$7 // 2 = 3$ (Quotient)
%	Modulus	$7 \% 2 = 1$ (Remainder)
**	Exponentiation	$7 ** 2 = 49$

Python Basic Operators

```
print('7 + 2 =', 7 + 2)
print('7 - 2 =', 7 - 2)
print('7 * 2 =', 7 * 2)
print('7 / 2 =', 7 / 2)
print('7 // 2 =', 7 // 2)
print('7 % 2 =', 7 % 2)
print('7 ** 2 =', 7 ** 2)
```

7 + 2 = 9
7 - 2 = 5
7 * 2 = 14
7 / 2 = 3.5
7 // 2 = 3
7 % 2 = 1
7 ** 2 = 49

Python Booleans: True or False

```
# Python Booleans: True or False  
print(3 > 2)  
print(3 == 2)  
print(3 < 2)
```

Python BMI Calculator

```
# BMI Calculator in Python
height_cm = 170
weight_kg = 60
height_m = height_cm/100
BMI = (weight_kg/(height_m**2))

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

Your BMI is: 20.8

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

How much is your \$100 worth after 7 years?

```
# How much is your $100 worth after 7 years?  
fv = 100 * 1.1 ** 7  
print('fv = ', round(fv, 2))  
# output = 194.87
```

```
fv = 194.87
```

Future Value

```
# Future Value
pv = 100
r = 0.1
n = 7

fv = pv * ((1 + (r)) ** n)
print(round(fv, 2))
```

194.87

Future Value

```
# Future Value
amount = 100
interest = 10 #10% = 0.01 * 10
years = 7

future_value = amount * ((1 + (0.01 * interest)) ** years)
print(round(future_value, 2))
```

194.87

Python

Data Structures

Python Data Structures

- Python Lists `[]`
- Python Tuples `()`
- Python Sets `{}`
- Python Dictionaries `{k:v}`

Python Data Structures

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

Python Data Types

```
x = ["apple", "banana", "cherry"] #list
x = ("apple", "banana", "cherry") #tuple
x = {"name": "Tom", "age": 20} #dict
x = {"apple", "banana", "cherry"} #set
```

Python Collections

- **There are four collection data types in the Python programming language**
- **List []**
 - **a collection which is ordered and changeable. Allows duplicate members.**
- **Tuple ()**
 - **a collection which is ordered and unchangeable. Allows duplicate members.**
- **Set {}**
 - **a collection which is unordered, unchangeable, and unindexed. No duplicate members.**
- **Dictionary {k:v}**
 - **a collection which is ordered and changeable. No duplicate members.**

Python Dictionaries {k:v}

- **As of Python version 3.7, dictionaries are ordered.**
- **In Python 3.6 and earlier, dictionaries are unordered.**

Lists []

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

4
60
70
90

Lists []

- **len():** how many items
- **type():** data type
- **list() constructor:** creating a new list

Python List Methods

• Method	Description
• <code>append()</code>	Adds an element at the end of the list
• <code>clear()</code>	Removes all the elements from the list
• <code>copy()</code>	Returns a copy of the list
• <code>count()</code>	Returns the number of elements with the specified value
• <code>extend()</code>	Add the elements of a list (or any iterable), to the end of the current list
• <code>index()</code>	Returns the index of the first element with the specified value
• <code>insert()</code>	Adds an element at the specified position
• <code>pop()</code>	Removes the element at the specified position
• <code>remove()</code>	Removes the item with the specified value
• <code>reverse()</code>	Reverses the order of the list
• <code>sort()</code>	Sorts the list

Tuples ()

A **tuple** in Python is a collection that **cannot be modified**.
A tuple is defined using **parenthesis**.

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

Sets {}

```
animals = {'cat', 'dog'}
print('cat' in animals)      True
print('fish' in animals)    False
animals.add('fish')
print('fish' in animals)    True
print(len(animals))         3
animals.add('cat')
print(len(animals))         3
animals.remove('cat')
print(len(animals))         2
```

Dictionary {key : value}

Python Dictionary

Key → Value

'EN' → 'English'

'FR' → 'French'

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

English

Python Data Structures

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

Python Control Logic and Loops

Python Control Logic and Loops

- Python **if else**
 - **if elif else**
 - Booleans: True, False
 - Operators: ==, !=, >, <, >=, <=, and, or, not
- Python **for** Loops
 - **for**
- Python **while** Loops
 - **While**
 - break
 - continue

Python **if...else**

- **Python if...else**
 - **if elif else**
 - **Booleans: True, False**
 - **Operators: ==, !=, >, <, >=, <=, and, or, not**

Python Conditions and **If** statements

- Python supports the usual **logical conditions** from mathematics:
 - Equals: $a == b$
 - Not Equals: $a != b$
 - Less than: $a < b$
 - Less than or equal to: $a <= b$
 - Greater than: $a > b$
 - Greater than or equal to: $a >= b$

Python Comparison Operators

Operator	Name	Example
<code>==</code>	Equal	<code>x == y</code>
<code>!=</code>	Not equal	<code>x != y</code>
<code>></code>	Greater than	<code>x > y</code>
<code><</code>	Less than	<code>x < y</code>
<code>>=</code>	Greater than or equal to	<code>x >= y</code>
<code><=</code>	Less than or equal to	<code>x <= y</code>

Python Logical Operators

Operator	Description	Example
and	Returns True if both statements are true	<code>x < 5 and x < 10</code>
or	Returns True if one of the statements is true	<code>x < 5 or x < 4</code>
not	Reverse the result, returns False if the result is true	<code>not(x < 5 and x < 10)</code>

Python if

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

Python if else

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

Python if elif else

```
score = 95
if score >= 90 :
    print("A")
elif score >=60 :
    print("Pass")
else:
    print("Fail")
```

Python if elif else

```
# Python if elif else
score = 90
grade = ""
if score >=90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "E"
print(grade)
```

Python **for** Loops

```
for i in range(1, 6):  
    print(i)
```

1
2
3
4
5

Python **for** loops

```
# for loops
for i in range(1,10):
    for j in range(1,10):
        print(i, ' * ', j, ' = ', i*j)
```

Python **while** Loops

- **while**
 - **break**
 - **continue**

Python **while** loops

```
# while loops
age = 10
while age < 20:
    print(age)
    age = age + 1
```

Python Functions and Modules

Python Functions and Modules

- Python Functions
 - **def** myfunction():
- Python Classes/Objects
 - **class** MyClass:
- Python Modules
 - mymodule.py
 - **import** mymodule

Python Functions

Python Functions

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.
- Creating a Function
 - In Python a function is defined using the **def** keyword:

Python Function def

```
# Python Function def
# indentation for blocks. four spaces
def getfv(pv, r, n):
    fv = pv * ((1 + (r)) ** n)
    return fv
fv = getfv(100, 0.1, 7)
print(round(fv, 2))
```

194.87

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

How much is your \$100 worth after 7 years?

```
# How much is your $100 worth after 7 years?  
fv = 100 * 1.1 ** 7  
print('fv = ', round(fv, 2))  
# output = 194.87
```

```
fv = 194.87
```

Future Value

```
# Future Value
pv = 100
r = 0.1
n = 7

fv = pv * ((1 + (r)) ** n)
print(round(fv, 2))
```

194.87

Future Value

```
# Future Value
amount = 100
interest = 10 #10% = 0.01 * 10
years = 7

future_value = amount * ((1 + (0.01 * interest)) ** years)
print(round(future_value, 2))
```

194.87

Python Function

`def` `getfv()` **define get future value function**

```
# Python Function def
# indentation for blocks. four spaces
def getfv(pv, r, n):
    fv = pv * ((1 + (r)) ** n)
    return fv
fv = getfv(100, 0.1, 7)
print(round(fv, 2))
```

194.87

Python
Classes/Objects
class MyClass :

Python Classes/Objects

- Python is an object oriented programming language.
- Almost everything in Python is an object, with its properties and methods.
- A Class is like an object constructor, or a "blueprint" for creating objects.
- Create a Class:
 - To create a class, use the keyword **class**:

Python Classes/Objects

class MyClass:

```
# Python class
class MyClass:
    x = 5

c1 = MyClass()
print(c1.x)
```

Python Classes/Objects

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

```
p1 = Person("Alan", 20)
```

```
print(p1.name)
```

```
print(p1.age)
```

Alan

20

Python Classes/Objects

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

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

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

Python Classes/Objects

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

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

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

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

Python Classes and Objects

```
class Vehicle:
    name = ""
    kind = "car"
    color = ""
    value = 100.00
    def description(self):
        desc_str = "%s is a %s %s worth $%.2f." %
(self.name, self.color, self.kind, self.value)
        return desc_str
```

Python Classes and Objects

```
car1 = Vehicle()
car1.name = "Fer"
car1.color = "red"
car1.kind = "convertible"
car1.value = 60000.00

car2 = Vehicle()
car2.name = "Jump"
car2.color = "blue"
car2.kind = "van"
car2.value = 10000.00

print(car1.description())
print(car1.name)
print(car2.description())
print(car2.name)
```

```
class Vehicle:
    name = ""
    kind = "car"
    color = ""
    value = 100.00
    def description(self):
        desc_str = "%s is a %s %s
worth $%.2f." % (self.name, self.color,
self.kind, self.value)
        return desc_str
```

```
Fer is a red convertible worth $60000.00.
Fer
Jump is a blue van worth $10000.00.
Jump
```

Python Modules

Python Modules

- Consider a **module** to be the same as a **code library**.
- A file containing a set of functions you want to include in your application.
- Create a Module
 - To create a **module** just save the code you want in a **file** with the file extension **.py**:
- Use a Module
 - **import** module

Python Modules

```
# mymodule.py
def greeting(name):
    print("Hello, " + name)
```

```
import mymodule
mymodule.greeting("Alan")
```

```
mymodule.py
def greeting(name):
    print("Hello, " + name)
```

Python File Input / Output

```
# Python File Input / Output
with open('myfile.txt', 'w') as file:
    file.write('Hello World\nThis is Python File Input Output')

with open('myfile.txt', 'r') as file:
    text = file.read()
    print(text)
```

Hello World This is Python File Input Output

Python File Input / Output

```
# Python File Input / Output
filename = 'mymodule.py'
with open(filename, 'w') as file:
    text = '''def greeting(name):
print("Hello, " + name)
'''
    file.write(text)

with open(filename, 'r') as file:
    text = file.read()
print(filename)
print(text)
```

```
mymodule.py
def greeting(name):
    print("Hello, " + name)
```

Python Modules

```
import mymodule
```

```
# mymodule.py  
def greeting(name):  
    print("Hello, " + name)
```

```
import mymodule  
mymodule.greeting("Alan")
```

Hello, Alan

Python `main()` function

```
#Python main() function
def main():
    print("Hello World!")

if __name__ == "__main__":
    main()
```

Files and Exception Handling

Files and Exception Handling

- **Python Files (File Handling)**
 - **open()**
 - **f = open("myfile.txt")**
- **Python Try Except (Exception Handling)**
 - **try:**
 - **except:**
 - **else:**
 - **finally:**

File Handling

- The key function for working with files in Python is the **open()** function.
- The **open()** function takes two parameters; **filename**, and **mode**.
- There are four different methods (modes) for opening a file:
 - **"r" - Read** - Default value. Opens a file for reading, error if the file does not exist
 - **"a" - Append** - Opens a file for appending, creates the file if it does not exist
 - **"w" - Write** - Opens a file for writing, creates the file if it does not exist
 - **"x" - Create** - Creates the specified file, returns an error if the file exists

Python Files (File Handling)

```
f = open("myfile.txt", "w")  
f.write("Hello World")  
f.close()
```

```
f = open("myfile.txt", "r")  
text = f.read()  
print(text)  
f.close()
```

Hello World

Python Files (File Handling)

```
# Python File Input / Output
with open('myfile.txt', 'w') as file:
    file.write('Hello World')

with open('myfile.txt', 'r') as file:
    text = file.read()
print(text)
```

Hello World

Python Files

```
# Python File Input / Output
with open('myfile.txt', 'w') as file:
    file.write('Hello World\nPython File IO')

with open('myfile.txt', 'r') as file:
    text = file.read()
print(text)
```

Hello World

Python File IO

Python Files

```
# Python File Input / Output
with open('myfile.txt', 'a+') as file:
    file.write('\n' + 'New line')

with open('myfile.txt', 'r') as file:
    text = file.read()
print(text)
```

Hello World

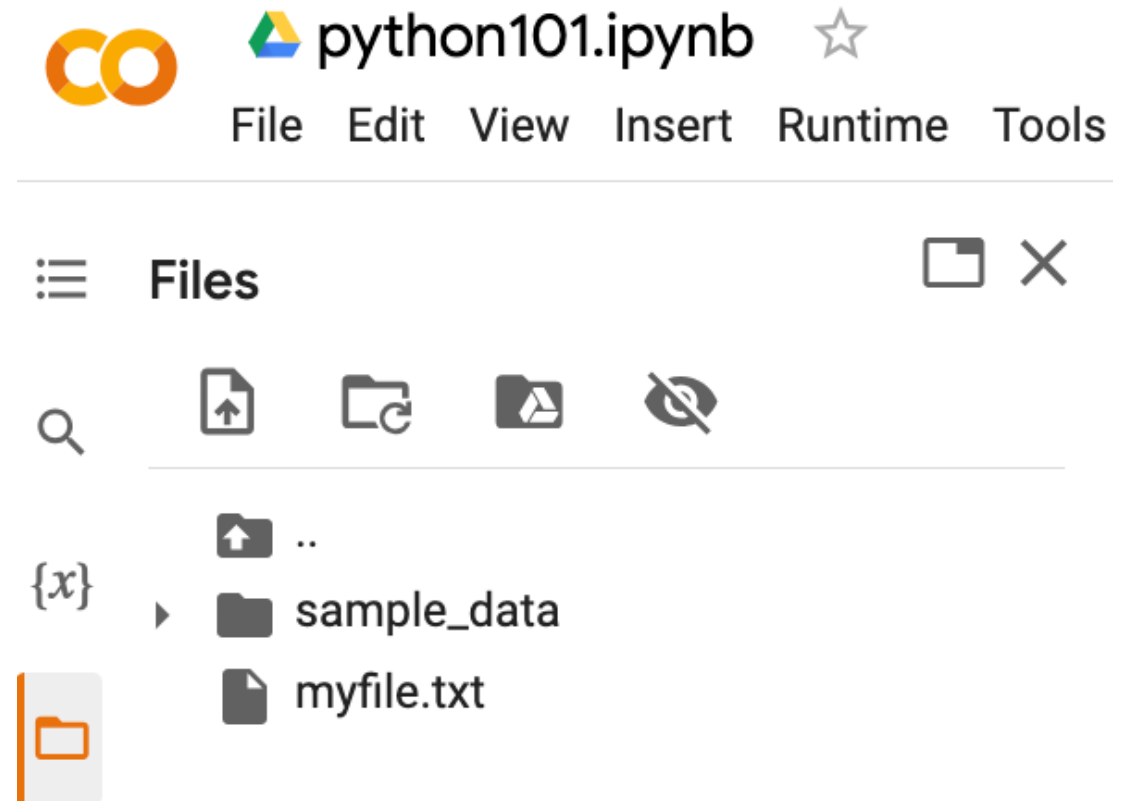
Python File IO

New line

Python Files

```
# !ls list files  
!ls
```

```
myfile.txt sample_data
```



The screenshot shows a Jupyter Notebook interface. At the top, the notebook title is "python101.ipynb" with a star icon. Below the title is a menu bar with "File", "Edit", "View", "Insert", "Runtime", and "Tools". The main area displays the output of the command `!ls`, which is `myfile.txt sample_data`. On the right side, there is a "Files" panel with a search icon and a list of files and folders: `..`, `sample_data`, and `myfile.txt`. The `sample_data` folder is expanded, showing its contents.

Python OS, IO, files, and Google Drive

```
import os

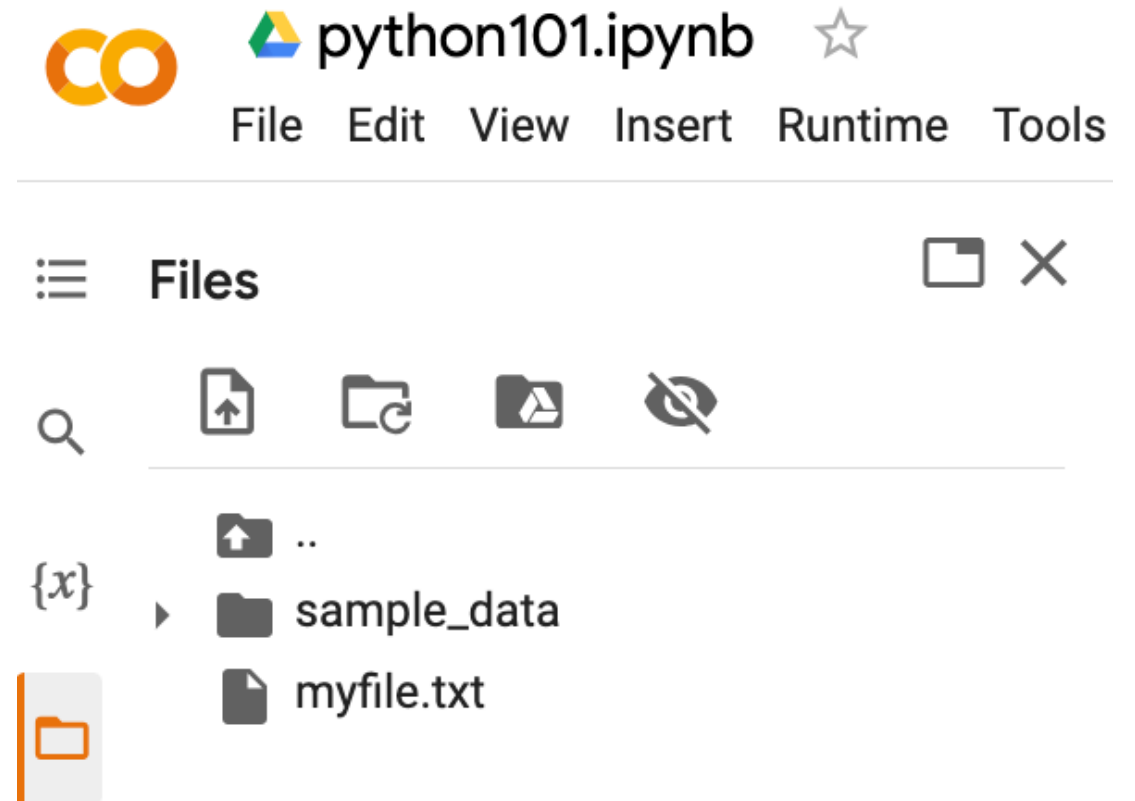
cwd = os.getcwd()
print(cwd)
```

/content

os.listdir()

```
os.listdir(cwd)
```

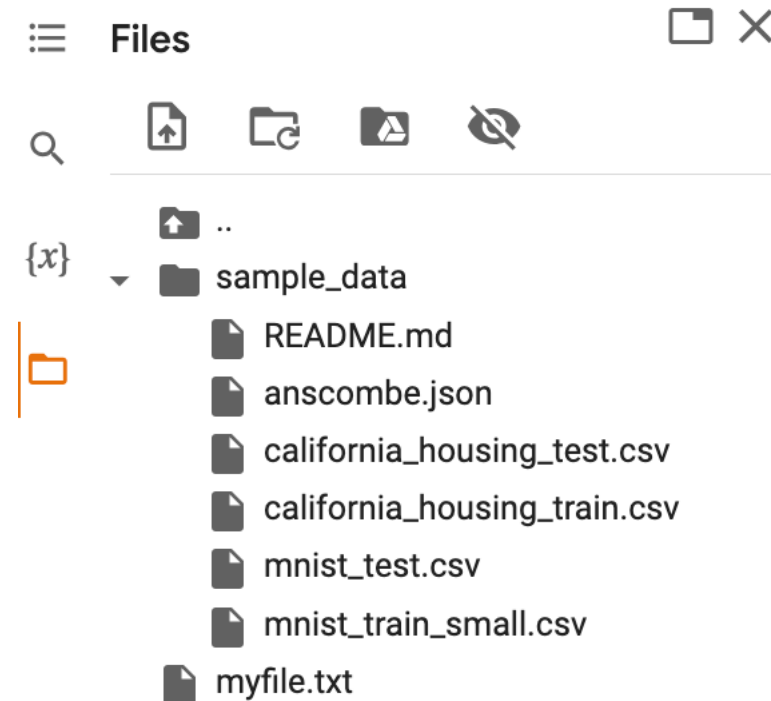
```
['.config',  
'myfile.txt',  
'sample_data']
```



os.path.join()

```
path = os.path.join(cwd, 'sample_data')
print(path)
os.listdir(path)
```

```
/content/sample_data
['README.md', 'anscombe.json',
'mnist_train_small.csv',
'mnist_test.csv',
'california_housing_train.csv',
'california_housing_test.csv']
```



from google.colab import files

```
from google.colab import files

with open('io_file_myday.txt', 'w') as f:
    f.write('Google Colab File Write Text some content Myday')

import time
time.sleep(1) # time sleep 1 second

files.download('io_file_myday.txt')
print('downloaded')
```

downloaded

Python Files

```
from google.colab import files
uploaded = files.upload()

for fn in uploaded.keys():
    print('User uploaded file "{name}"
with length {length} bytes'.format(
name=fn, length=len(uploaded[fn])))
```

User uploaded file "io_file_myday2.txt" with length 47 bytes

os.remove()

```
import os
if os.path.exists("myfile.txt"):
    os.remove("myfile.txt")
    print("myfile.txt removed")
else:
    print("The file does not exist")
```

myfile.txt removed

```
os.mkdir("myfolder1")  
os.rmdir("myfolder1")
```

```
import os  
os.listdir()  
os.mkdir("myfolder1")  
os.listdir()  
os.rmdir("myfolder1")  
os.listdir()
```

Python Try Except

- The **try** block lets you test a block of code for errors.
- The **except** block lets you handle the error.
- The **else** block lets you execute code when there is no error.
- The **finally** block lets you execute code, regardless of the result of the try- and except blocks.

Python Try Except (Exception Handling)

try: **except:**

```
#Python try except
try:
    print(x)
except:
    print("Exception Error")
```

Python try: except: finally:

```
#Python try except finally
try:
    print("Hello")
except:
    print("Exception Error")
finally:
    print("Finally process")
```

Hello
Finally process

Python try: except: else:

```
#Python try except else
try:
    print("Hello")
except:
    print("Exception Error")
else:
    print("No exception")
```

Hello

No exception

Python try: except: else: finally:

```
try:  
    print("Hello")  
except:  
    print("Exception Error")  
else:  
    print("No exception")  
finally:  
    print("Finally process")
```

Hello

No exception

Finally process

Python try: except: else: finally:

```
try:
    price = float(input("Enter the price of the stock (e.g. 10):"))
    shares = int(input("Enter the number of shares (e.g. 2):"))
    total = price * shares
except Exception as e:
    print("Exception error:", str(e))
else:
    print("The total value of the shares is:", total)
finally:
    print("Thank you.")
```

```
Enter the price of the stock (e.g. 10):10
Enter the number of shares (e.g. 2):2
The total value of the shares is: 20.0
Thank you.
```

Python try: except: else: finally:

```
try:  
    file = open("myfile.txt")  
    file.write("Python write file")  
    print("file saved")  
except:  
    print("Exception file Error")
```

Exception file Error

Python try: except: else: finally:

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

```
Exception file Error
Finally process
```

Python try: except: else: finally:

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

```
file saved  
Finally process
```

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

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:

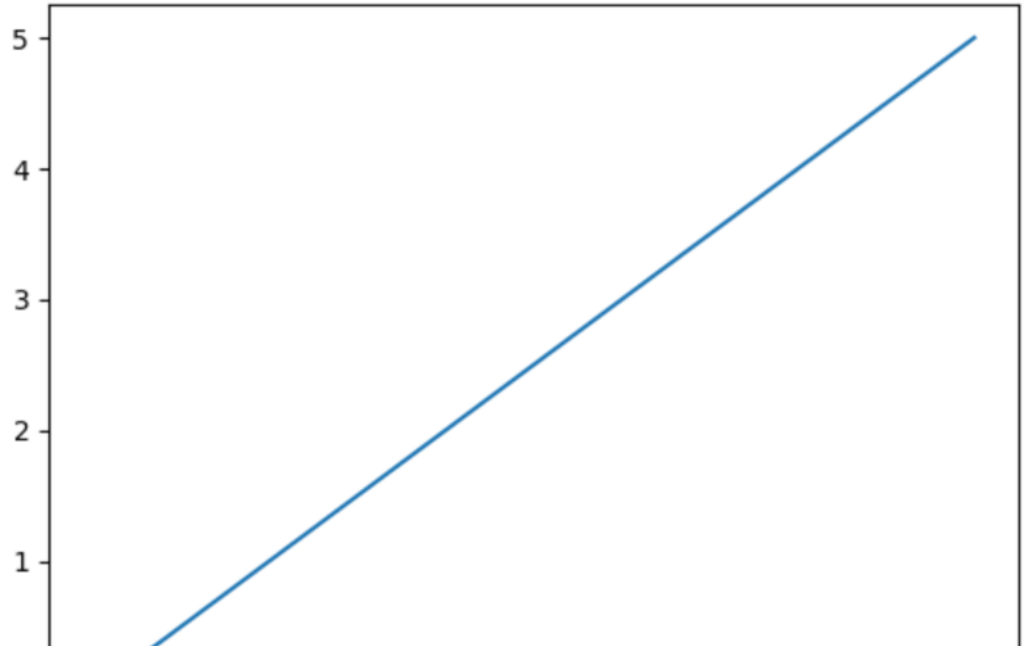


- Python Modules
 - NumPy Tutorial
 - Pandas Tutorial
 - SciPy Tutorial
 - Django Tutorial
- Python Matplotlib
 - Matplotlib Intro**
 - Matplotlib Get Started
 - Matplotlib Pyplot
 - Matplotlib Plotting
 - Matplotlib Markers
 - Matplotlib Line
 - Matplotlib Labels
 - Matplotlib Grid
 - Matplotlib Subplot
 - Matplotlib Scatter
 - Matplotlib Bars
 - Matplotlib Histograms
 - Matplotlib Pie Charts

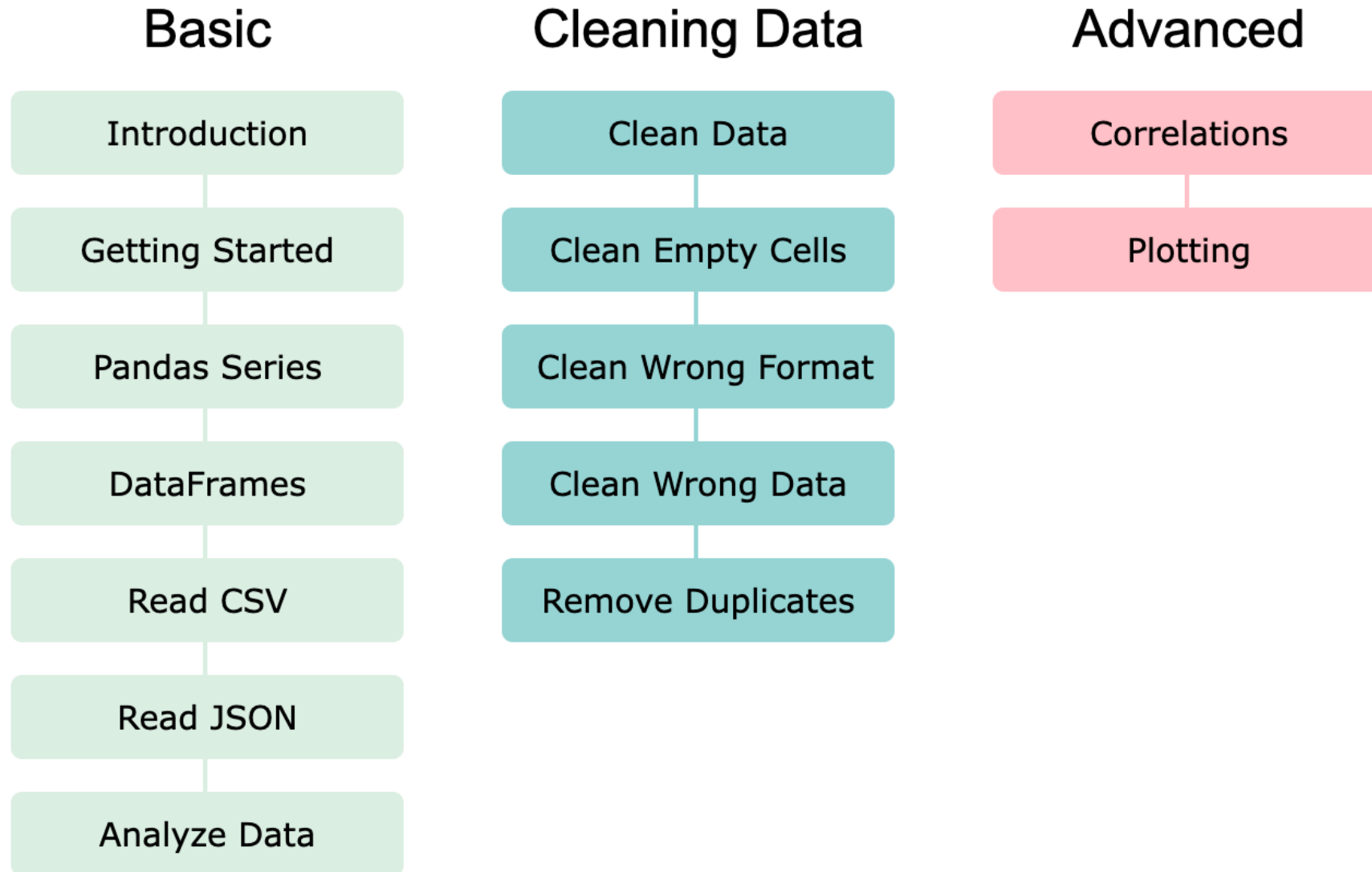
Matplotlib Tutorial

[← 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-book Public

Notifications Fork 14.1k Star 19k

Code Issues 2 Pull requests 1 Actions Projects Wiki Security Insights

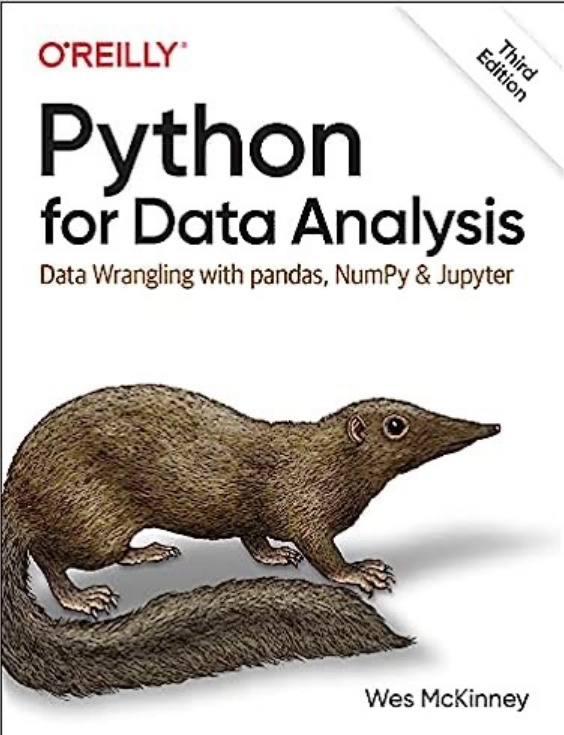
3rd-edition 3 branches 0 tags

Go to file Code

About

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

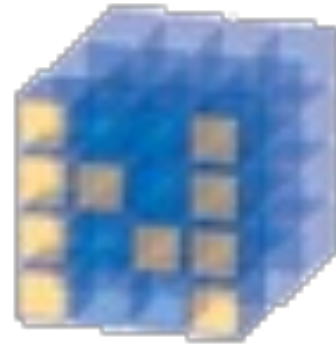
wesm	Upload cleaner notebook files without internal build toolchai...	f1757b8	3 days ago	70 commits
datasets	Add fec.parquet			10 months ago
examples	Simplifying datasets			10 months ago
.gitignore	Add gitignore			8 years ago
COPYING	Update COPYING			4 months ago
README.md	Update notebooks in advance 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
Python
for Data Analysis
Data Wrangling with pandas, NumPy & Jupyter
Third Edition
Wes McKinney

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

NumPy



NumPy

Base

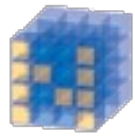
N-dimensional array
package

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

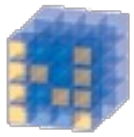
NumPy ndarray

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
```

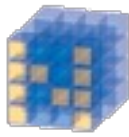
```
2 * v
```

```
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])
```

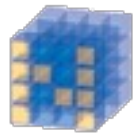
Python Data Structures

```
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



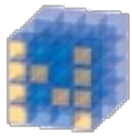
NumPy

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
```

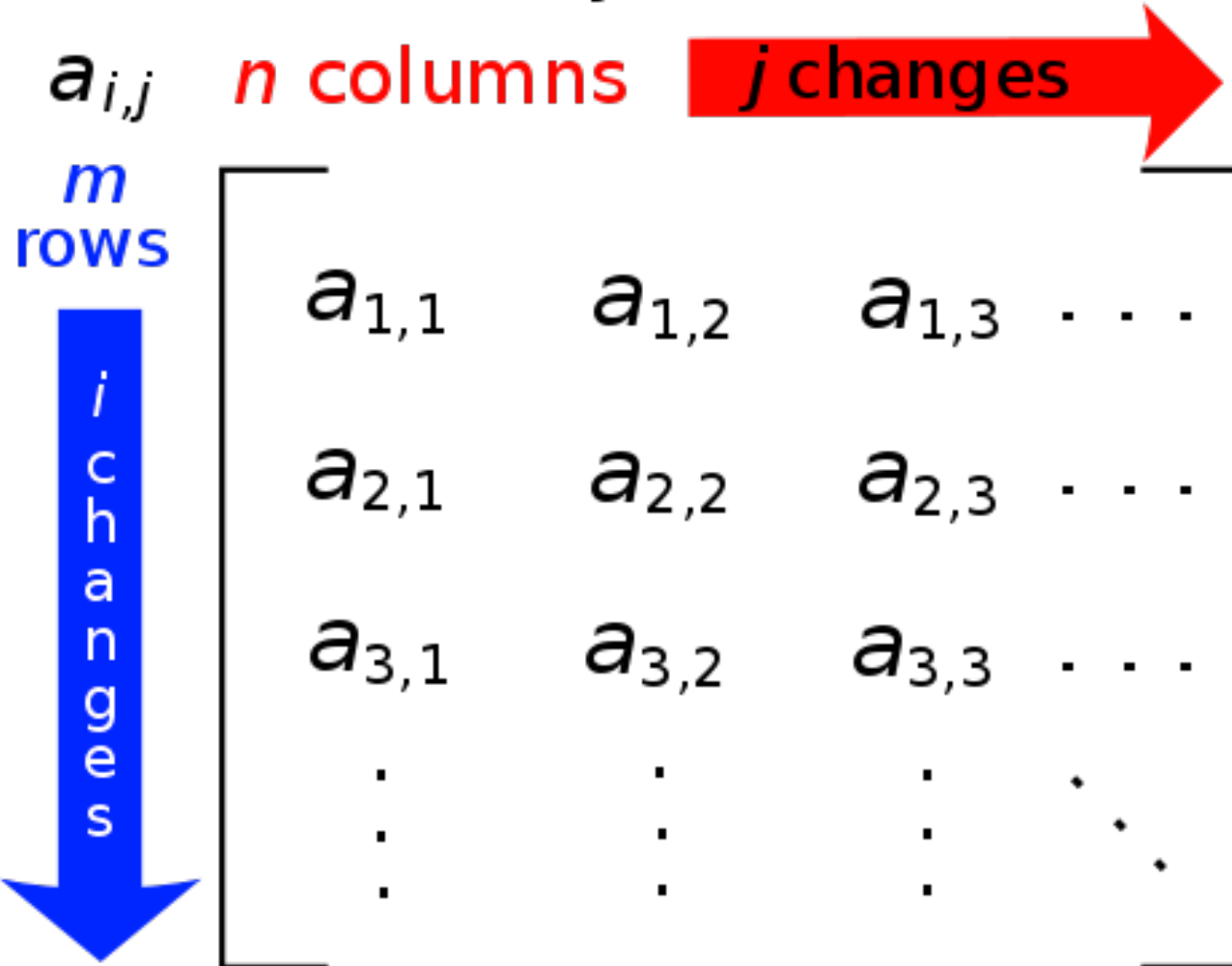
```
2
```

```
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

0

0,0

0,1

0,2

axis 0

1

1,0

1,1

1,2

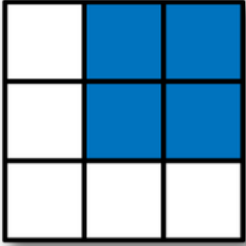
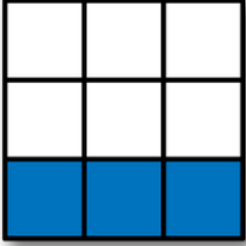
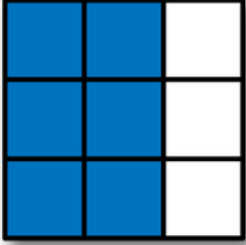
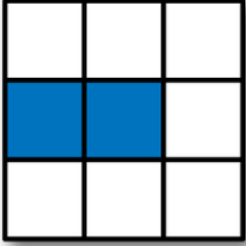
2

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

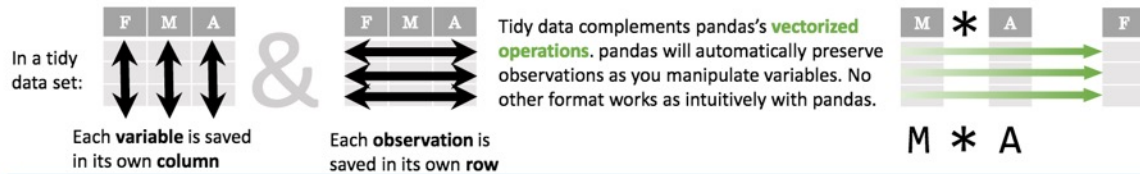
Comparison with SAS

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.
```

n	v	a	b	c
d	1	4	7	10
e	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

```
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	
'\.'	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'

Logic in Python (and pandas)		
<	Less than	!=
>	Greater than	df.column.isin(values)
==	Equals	pd.isnull(obj)
<=	Less than or equals	pd.notnull(obj)
>=	Greater than or equals	&, , ~, ^, df.any(), df.all()
		Logical and, or, not, xor, any, all

```
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

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 Data Analysis and Visualization

 pandas

 plotly

matplotlib

 bokeh

 seaborn

Python

Pandas



Python
matplotlib
matplotlib

Python

seaborn



seaborn

Python

plotly



Python

bokeh

bokeh

Python matplotlib



Installation Documentation Examples Tutorials Contributing

[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.

<h3>Create</h3> <ul style="list-style-type: none">• Develop publication quality plots with just a few lines of code• Use interactive figures that can zoom, pan, update...	<h3>Customize</h3> <ul style="list-style-type: none">• Take full control of line styles, font properties, axes properties...• Export and embed to a number of file formats and interactive environments	<h3>Extend</h3> <ul style="list-style-type: none">• 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](#)

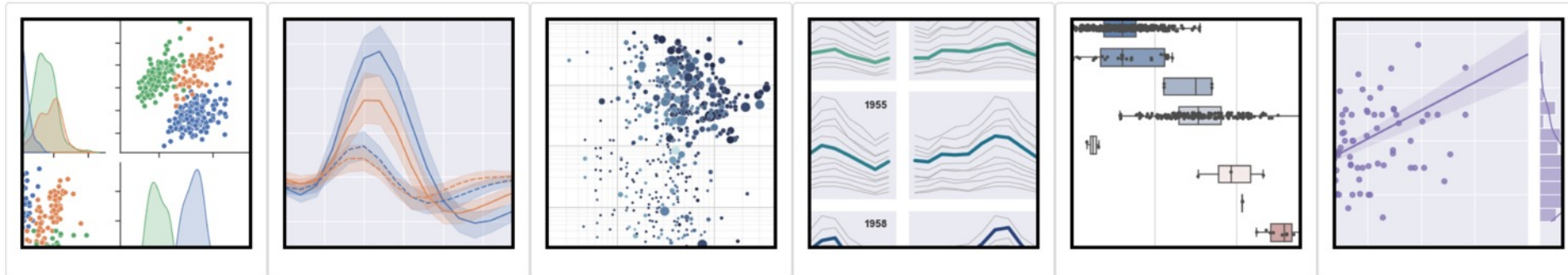
Last release for Python 2
2.2.5: [docs](#) | [changelog](#)

Development version
[docs](#)





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

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.

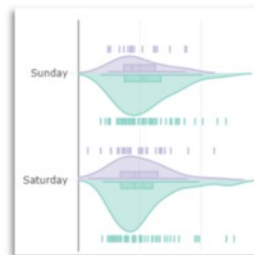
Plotly.py is [free and open source](#) and you can [view the source](#), [report issues](#) or [contribute on GitHub](#).

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.](#)

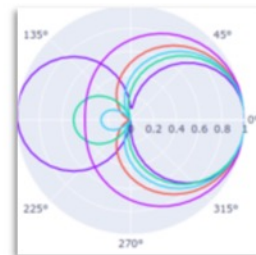
[Install Dash Enterprise on Azure](#) | [Install Dash Enterprise on AWS](#)

Fundamentals

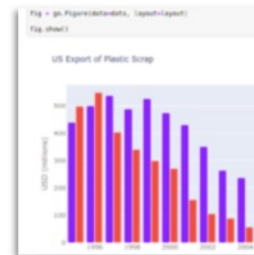
[More Fundamentals »](#)



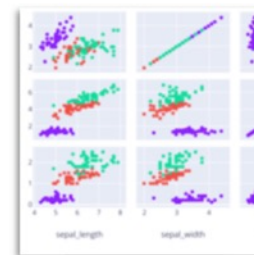
The Figure Data Structure



Creating and Updating Figures



Displaying Figures



Plotly Express

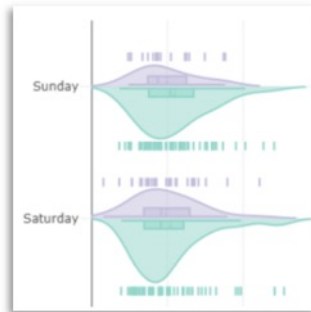


Analytical Apps with Dash

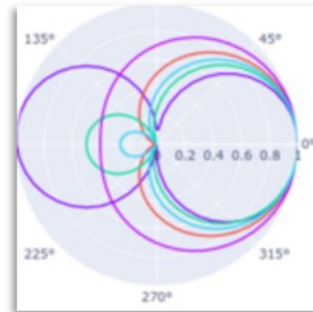
Python Plotly Graphing Library

Fundamentals

[More Fundamentals »](#)



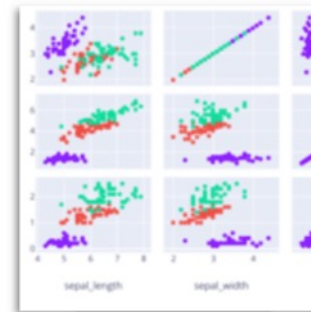
The Figure Data Structure



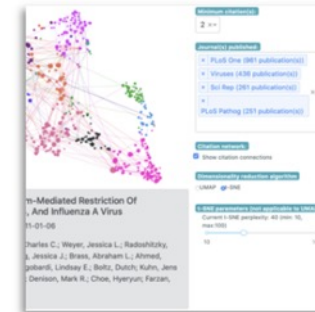
Creating and Updating Figures



Displaying Figures



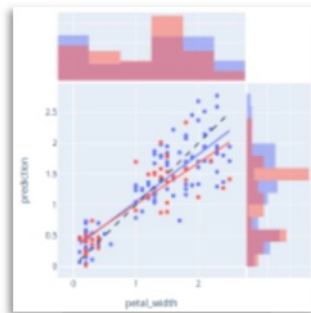
Plotly Express



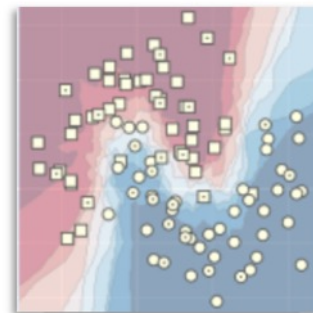
Analytical Apps with Dash

Artificial Intelligence and Machine Learning

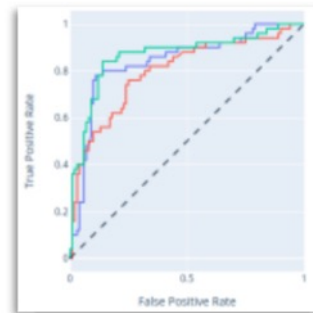
[More AI and ML »](#)



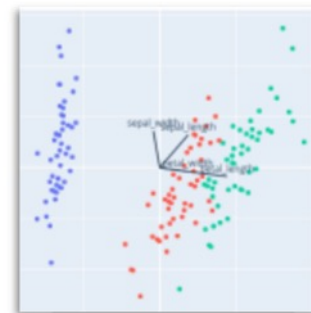
ML Regression



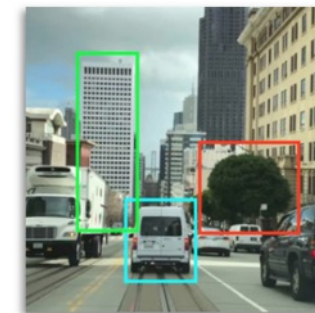
kNN Classification



ROC and PR Curves



PCA Visualization

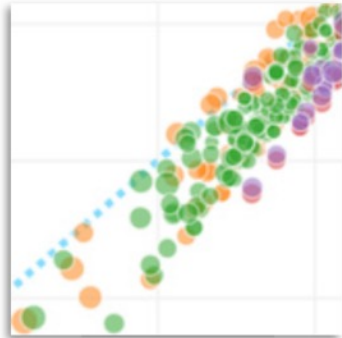


AI/ML Apps with Dash

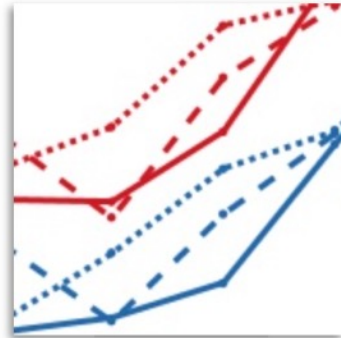
Python Plotly Graphing Library

Basic Charts

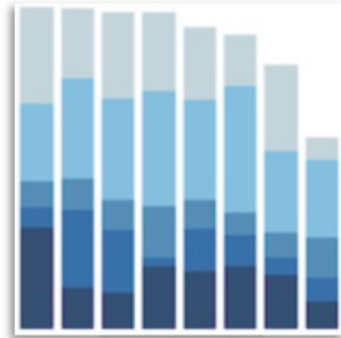
[More Basic Charts »](#)



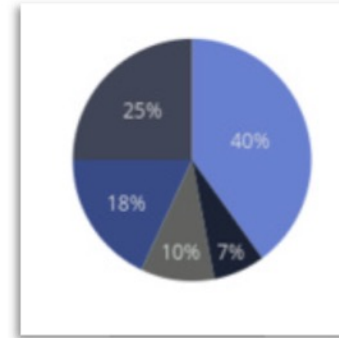
Scatter Plots



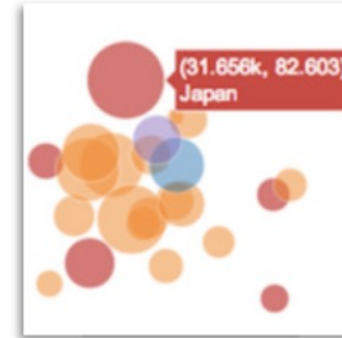
Line Charts



Bar Charts



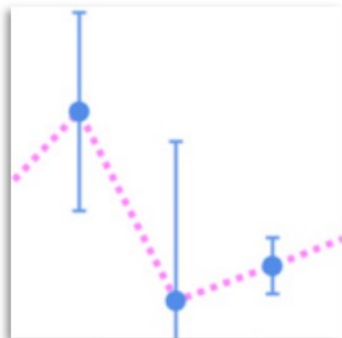
Pie Charts



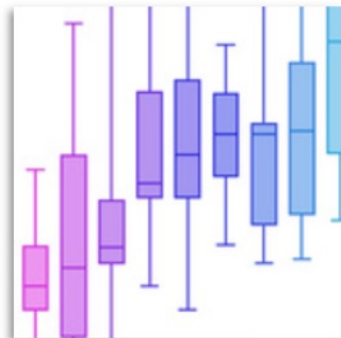
Bubble Charts

Statistical Charts

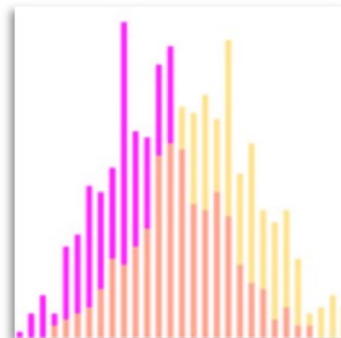
[More Statistical Charts »](#)



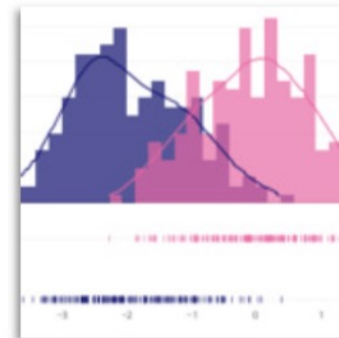
Error Bars



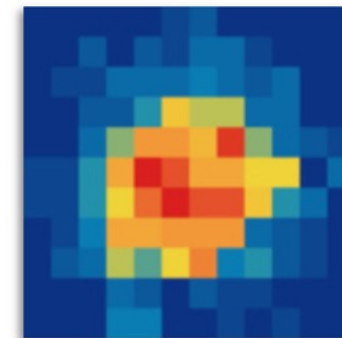
Box Plots



Histograms



Distplots

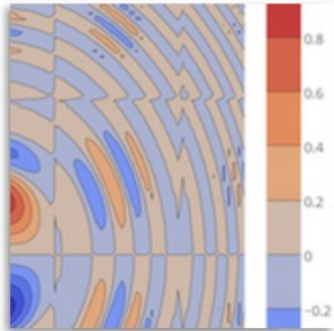


[2D Histograms](#)

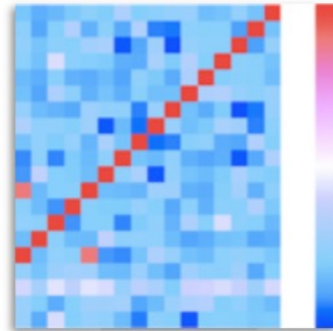
Python Plotly Graphing Library

Scientific Charts

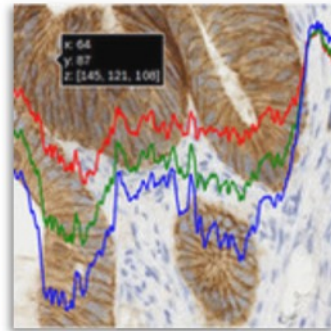
[More Scientific Charts »](#)



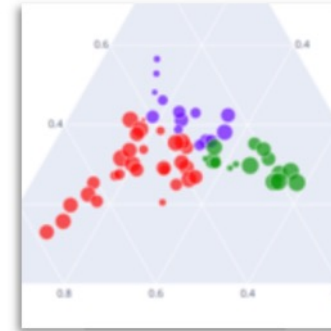
Contour Plots



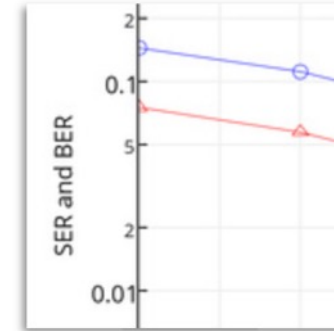
Heatmaps



Imshow



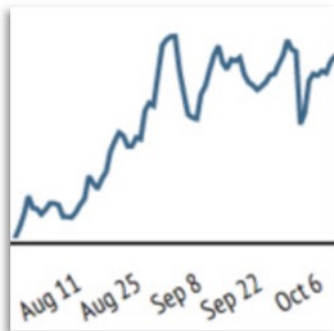
Ternary Plots



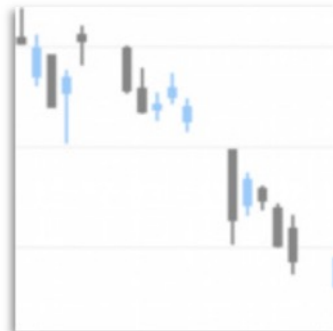
Log Plots

Financial Charts

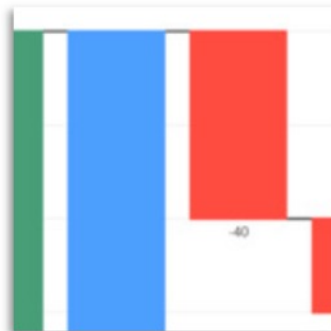
[More Financial Charts »](#)



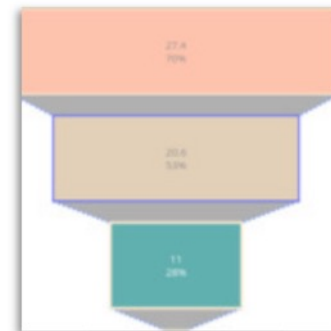
Time Series and Date Axes



Candlestick Charts



Waterfall Charts



Funnel Chart

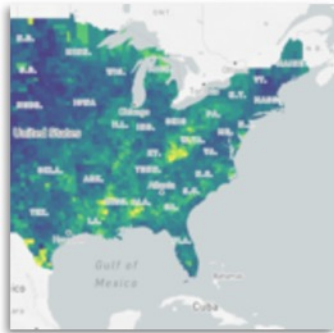


[OHLC Charts](#)

Python Plotly Graphing Library

Maps

[More Maps >](#)



Mapbox Choropleth
Maps



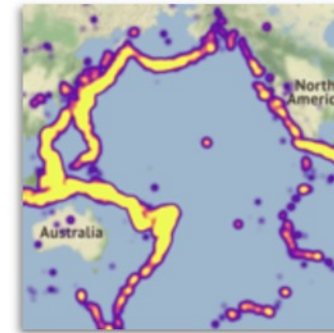
Lines on Mapbox



Filled Area on Maps



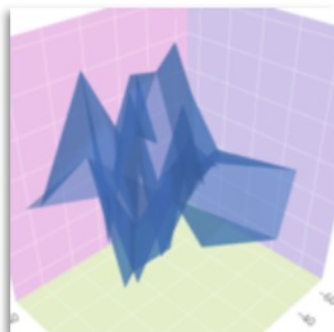
Bubble Maps



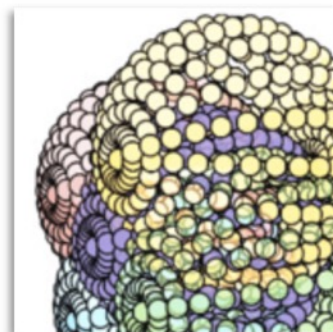
Mapbox Density
Heatmap

3D Charts

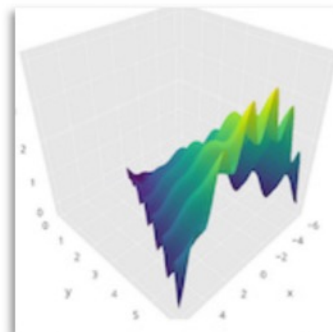
[More 3D Charts >](#)



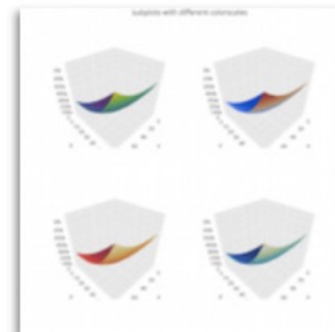
3D Axes



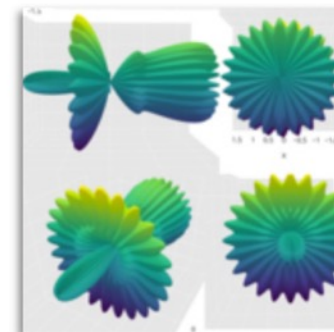
3D Scatter Plots



3D Surface Plots



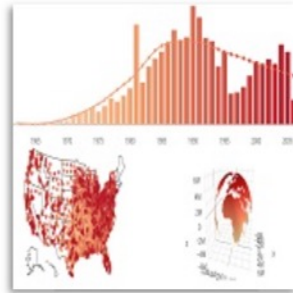
3D Subplots



[3D Camera Controls](#)

Python Plotly Graphing Library

Subplots



Mixed Subplots



Map Subplots

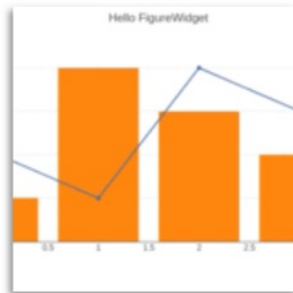


Table and Chart Subplots

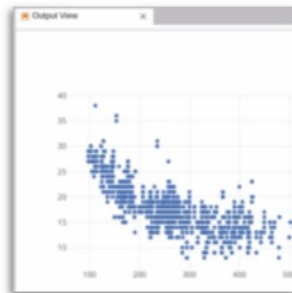


Figure Factory Subplots

Jupyter Widgets Interaction



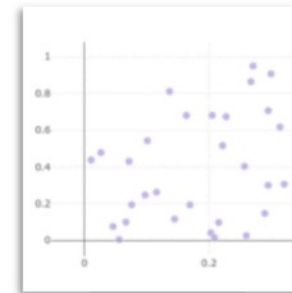
Plotly FigureWidget Overview



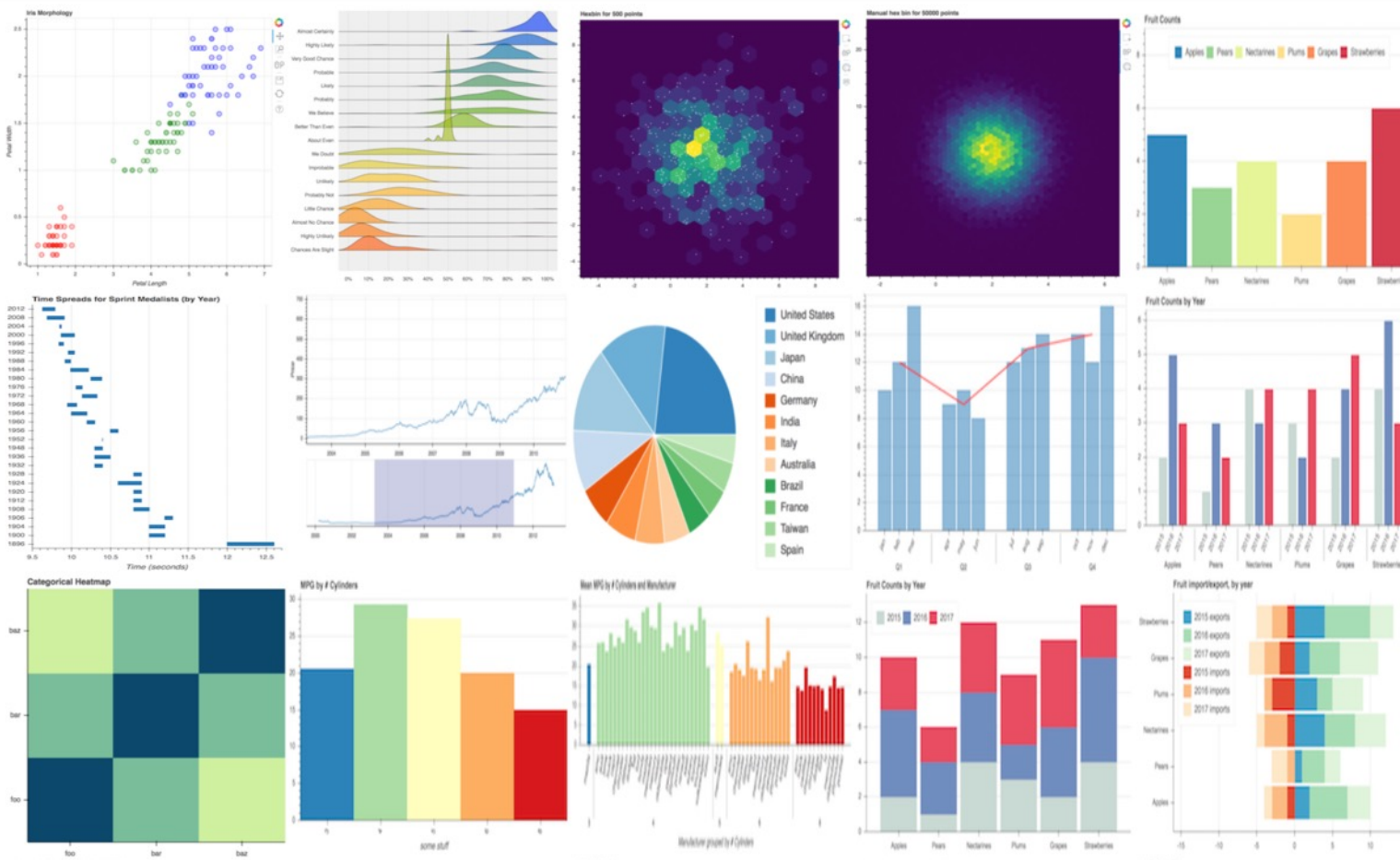
Jupyter Lab with FigureWidget



Interactive Data Analysis with FigureWidget ipywidgets



Click Events



Iris flower data set

setosa



versicolor



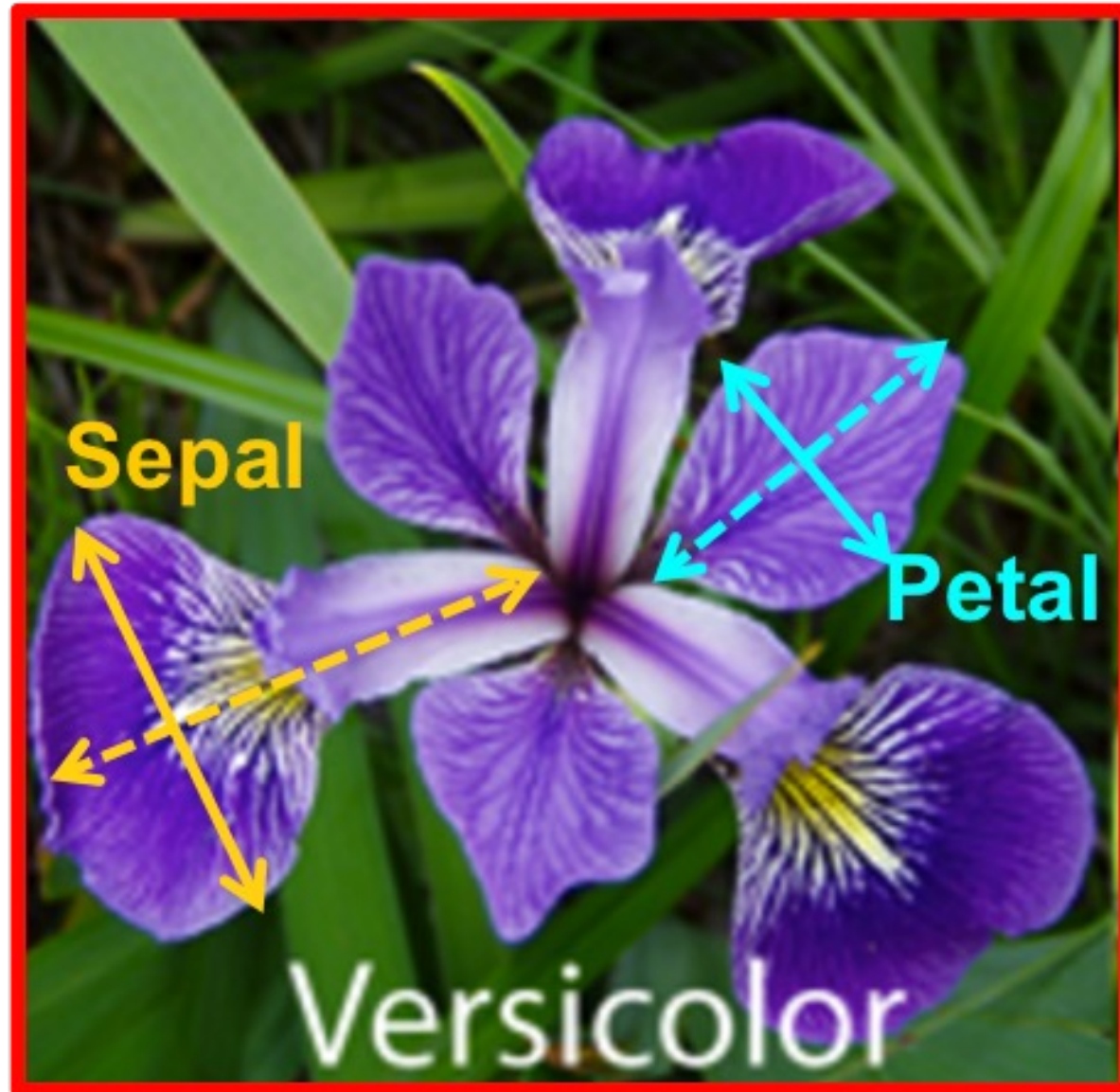
virginica



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



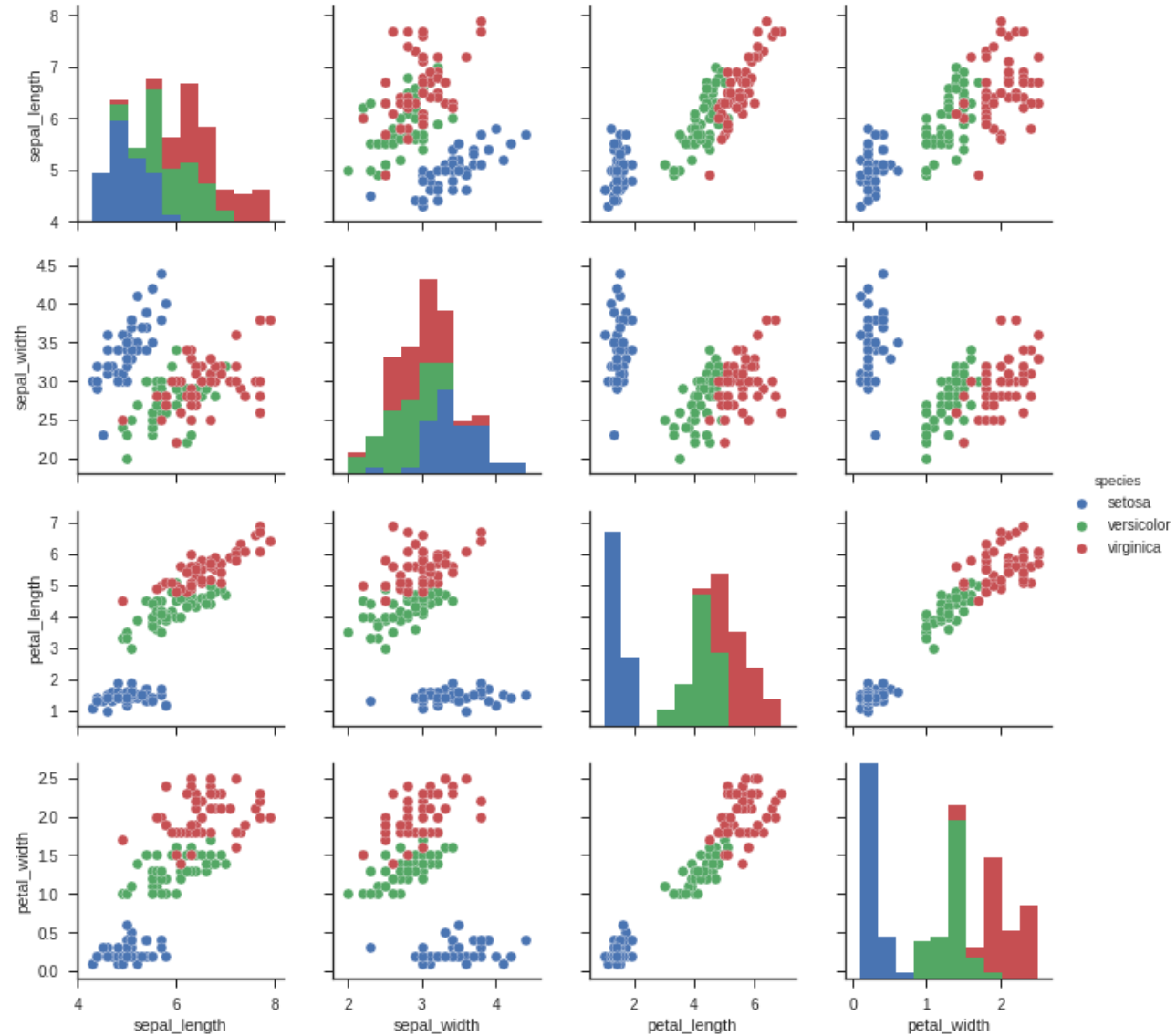
virginica



versicolor



Iris Data Visualization



Data Visualization in Google Colab

python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share Settings A

RAM Disk Editing

Table of contents

- 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

+ Code + Text

```
[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")
```

sepal_length

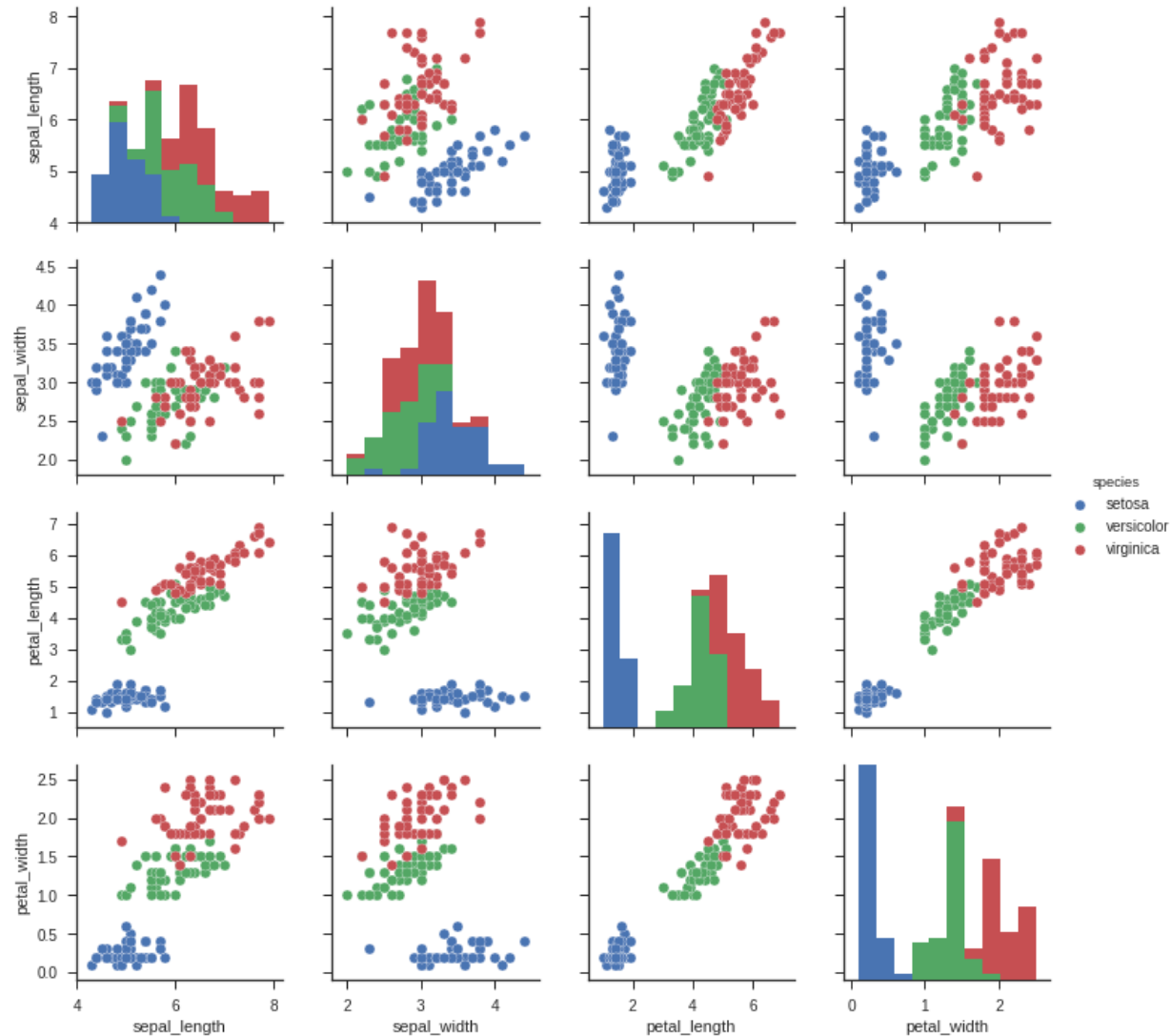
sepal_width

petal_width

species

- setosa
- versicolor
- virginica

```
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)
```

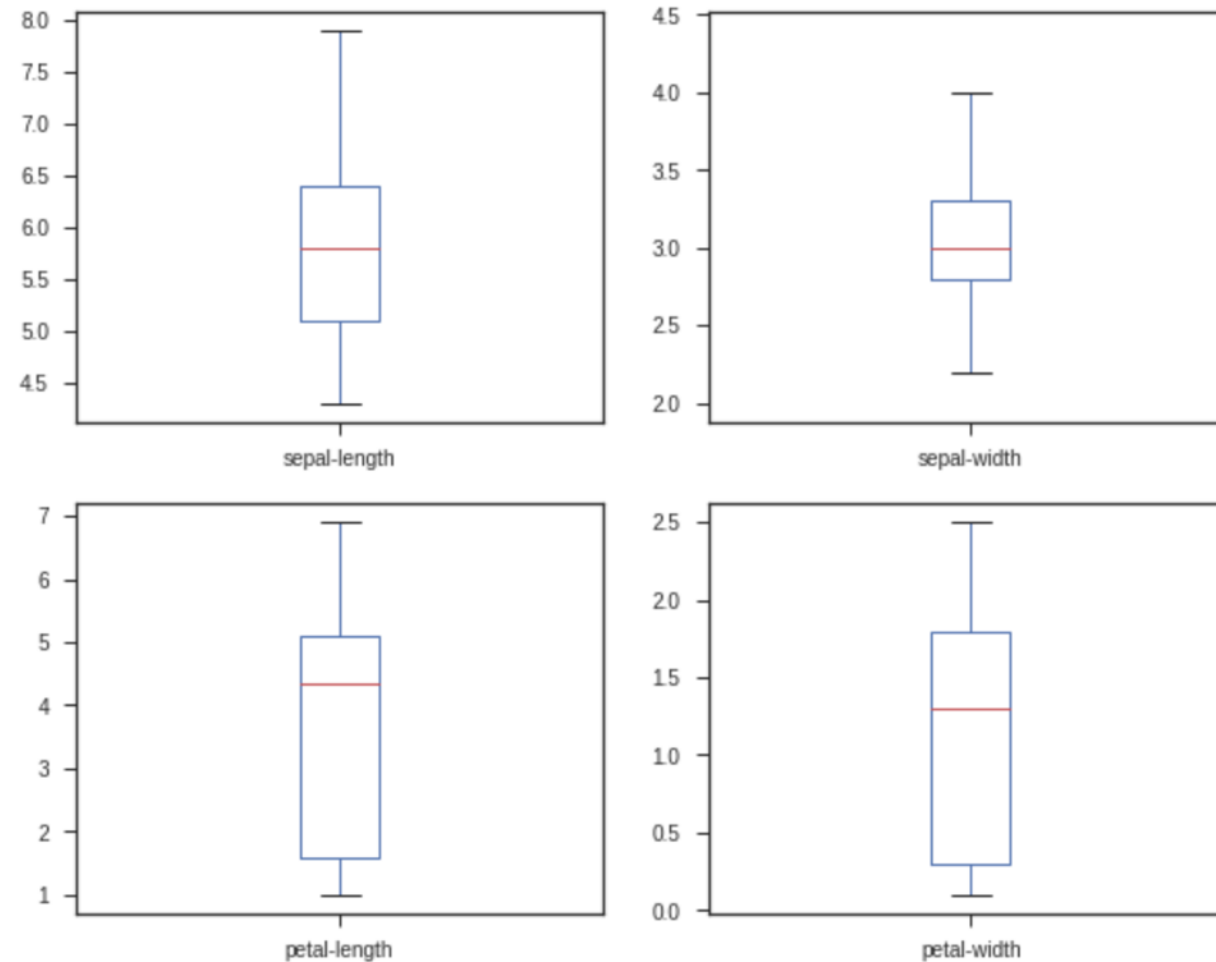
```
df.groupby('class').size()
```

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

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

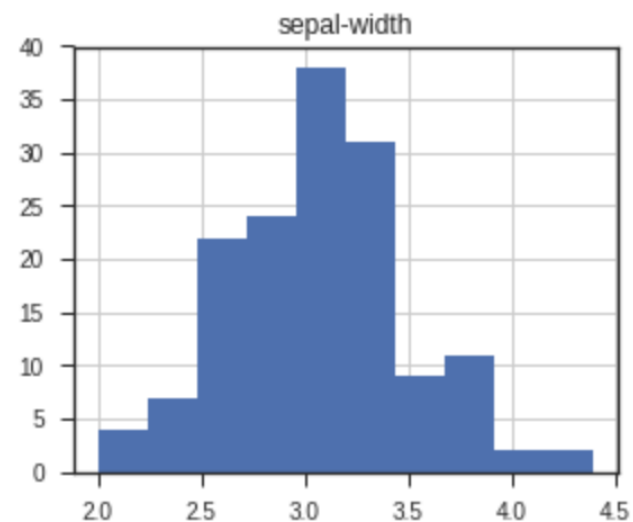
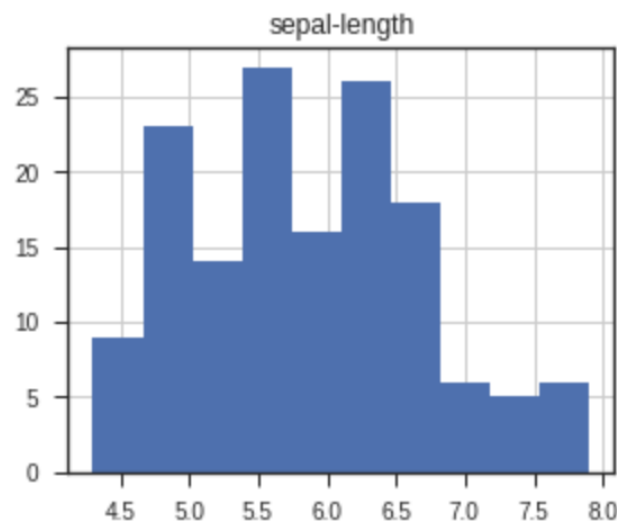
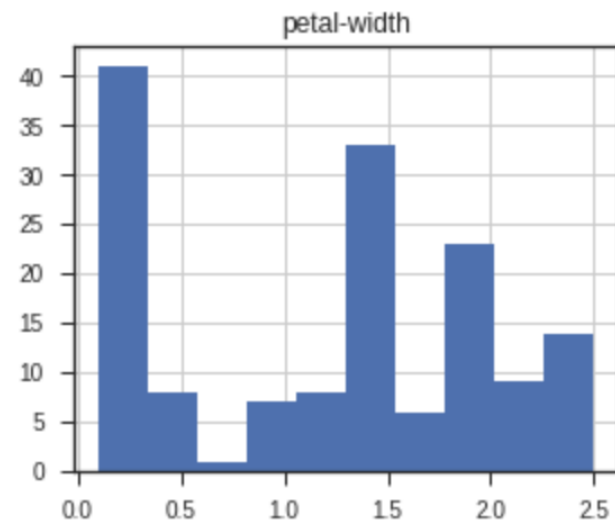
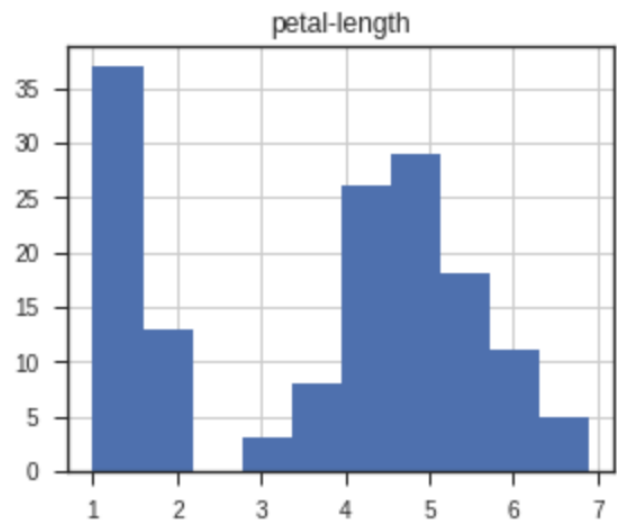
```
plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```

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



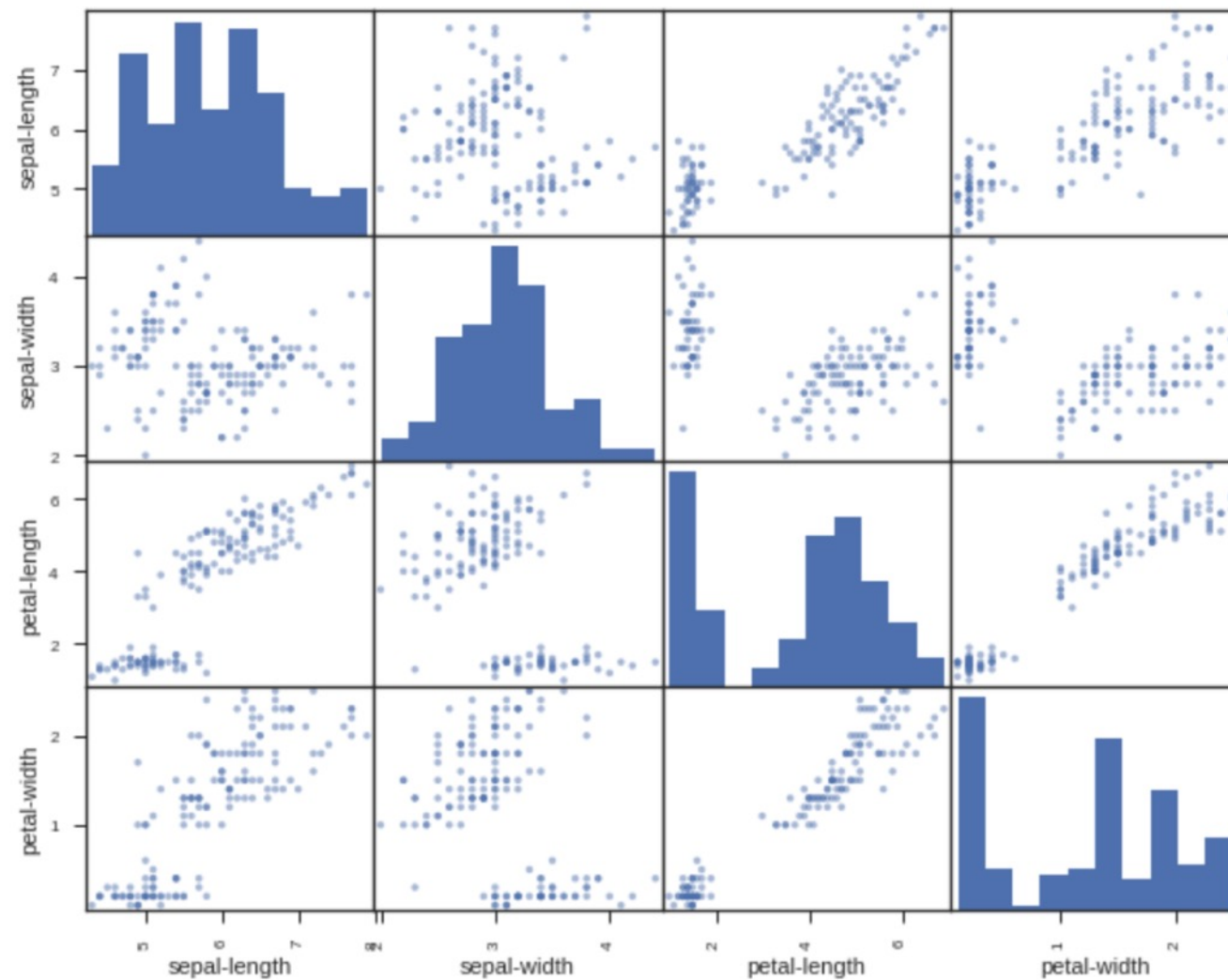
```
df.hist()  
plt.show()
```

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



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

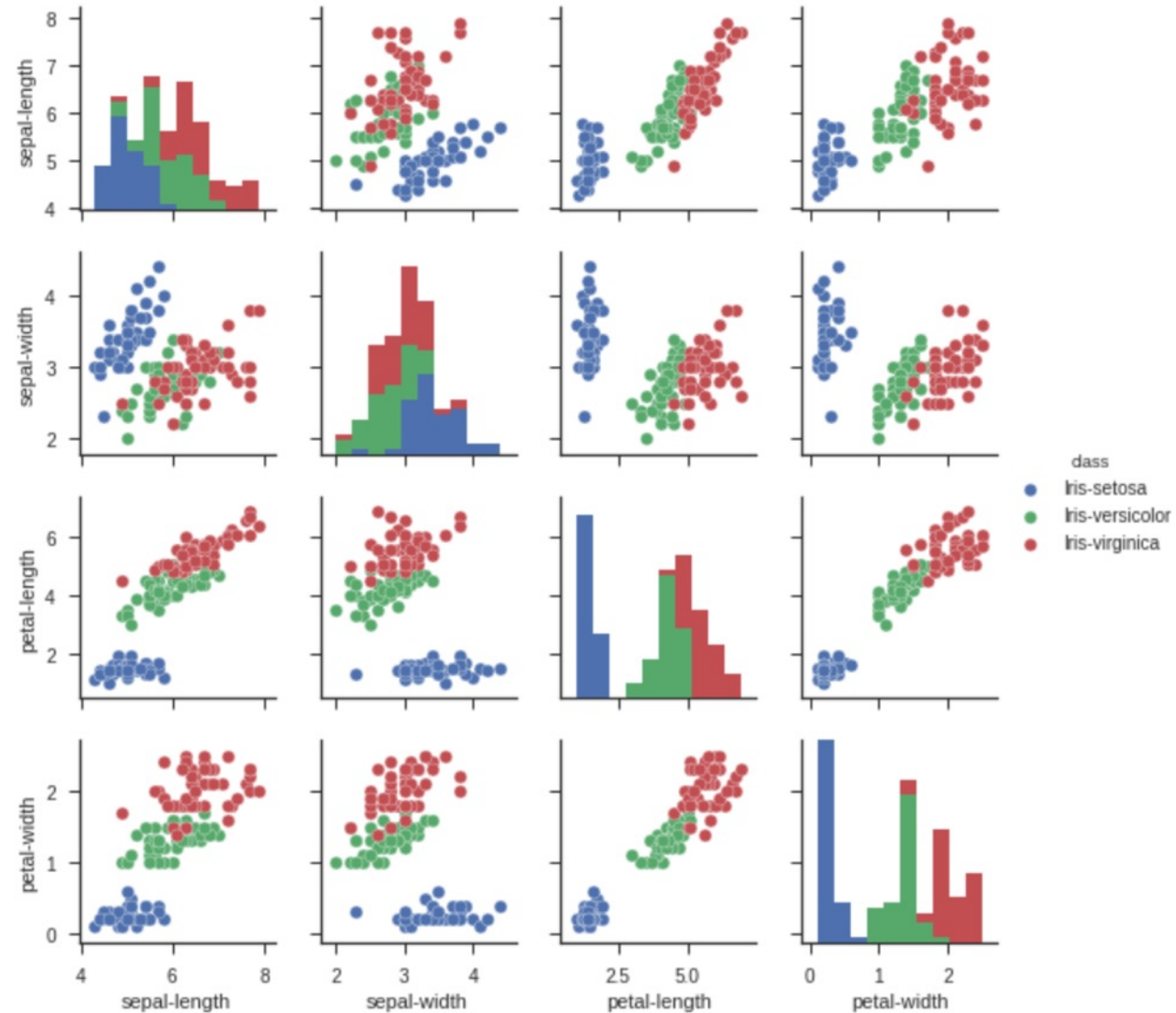
```
scatter_matrix(df)  
plt.show(.)
```



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

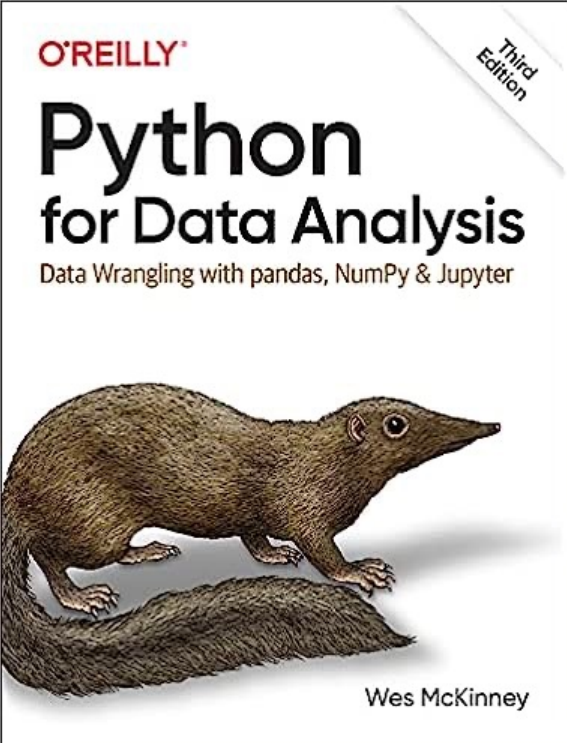
3rd-edition 3 branches 0 tags

Go to file Code

About

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

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
Python
for Data Analysis
Data Wrangling with pandas, NumPy & Jupyter
Third Edition
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

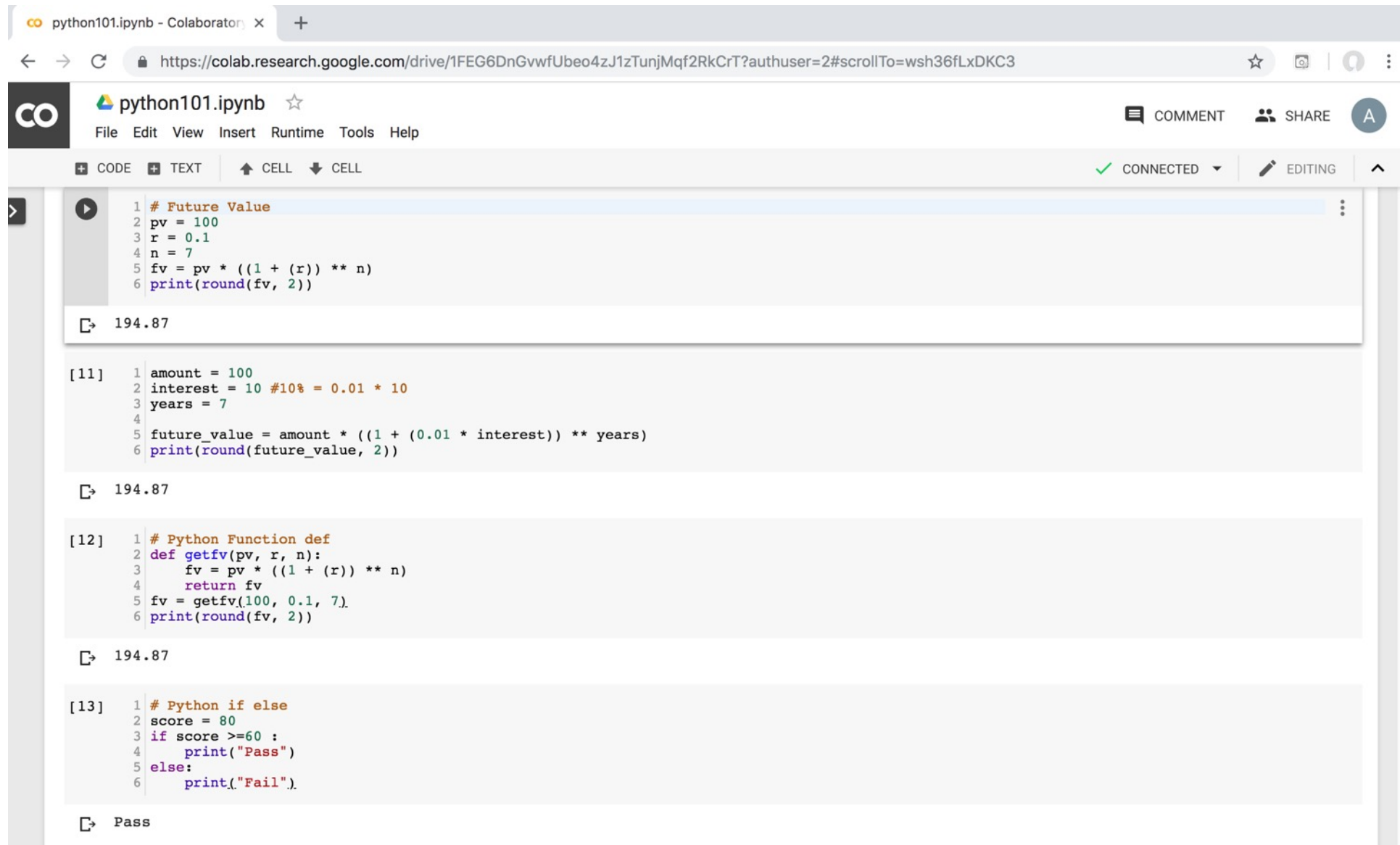
my_arr = np.arange(1_000_000)
my_list = list(range(1_000_000))
```

```
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
```

Python in Google Colab (Python101)

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



The screenshot shows a Google Colab notebook titled "python101.ipynb". The interface includes a browser address bar with the URL <https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT?authuser=2#scrollTo=wsh36fLxDKC3>. The notebook has a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". The top right shows "COMMENT", "SHARE", and a user profile icon. The notebook is in "EDITING" mode and is "CONNECTED".

The notebook contains four code cells:

- Cell 1:** A code cell with the following Python code:

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

The output is "194.87".
- Cell [11]:** A code cell with the following Python code:

```
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))
```

The output is "194.87".
- Cell [12]:** A code cell with the following Python code:

```
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))
```

The output is "194.87".
- Cell [13]:** A code cell with the following Python code:

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

The output is "Pass".

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab notebook interface. At the top, the notebook is titled "python101.ipynb" and has a star icon. The menu bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", with a status indicator "All changes saved". On the right, there are icons for "Comment", "Share", "Settings", and a user profile "A". Below the menu bar, there are RAM and Disk usage indicators, and a toolbar with icons for "Editing", "Up", "Down", "Link", "Comment", "Copy", "Paste", "Delete", and "More".

The left sidebar contains a "Table of contents" with a search icon and a list of 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" (highlighted), "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", and "Efficient Frontier Portfolio Optimisation in Python".

The main content area shows a code cell with the following Python code:

```
[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")
```

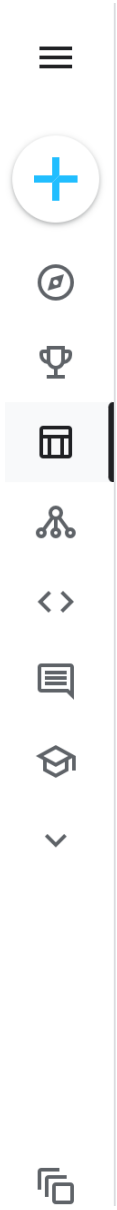
Below the code, a pairplot is displayed. The plot is a 4x4 grid of subplots. The diagonal elements are histograms showing the distribution of "sepal_length", "sepal_width", and "th" for each species. The off-diagonal elements are scatter plots showing the relationship between pairs of variables: "sepal_length" vs "sepal_width", "sepal_length" vs "th", and "sepal_width" vs "th". The data points are colored by species: blue for "setosa", orange for "versicolor", and green for "virginica". A legend in the bottom right corner identifies the species colors.

<https://tinyurl.com/aintpupython101>

kaggle™

**Kaggle Datasets
for
Data Science**

Kaggle Datasets for Machine Learning



Datasets

+ New Dataset

Your Work

 Filters

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


89 Datasets

Hotness ▾

	S&P 500 ESG Risk Ratings Pritish Dugar · Updated 4 months ago Usability 10.0 · 1 File (CSV) · 252 kB	50 Bronze ...
	Public Company ESG Ratings Dataset Alistair King · Updated 8 months ago Usability 10.0 · 1 File (CSV) · 43 kB	67 Gold ...
	US Funds dataset from Yahoo Finance Stefano Leone · Updated 3 years ago Usability 10.0 · 7 Files (CSV) · 371 MB	246 Silver ...

Kaggle Datasets for Machine Learning



Search 



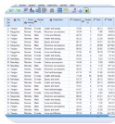



Datasets

[+ New Dataset](#) [Your Work](#)

[Filters](#)

- All datasets X
- 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 ...

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

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

Updated 5y ago
657 comments · Credit Card Fraud Detection

▲ 5453

Gold ...



Outlier!!! The Silent Killer

Updated 3y ago
138 comments · Titanic - Machine Learning from Disaster +16

▲ 1070

Gold ...



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

Updated 8y ago
125 comments · Credit Card Fraud Detection

▲ 796

Gold ...



Credit Card Fraud Detection Predictive Models

Updated 4y ago
41 comments · Credit Card Fraud Detection

▲ 489

Gold ...

Kaggle Code: Credit Card Fraud Detection



JANIO MARTINEZ BACHMANN · 5Y AGO · 922,344 VIEWS

▲ 5453

Edit My Copy 12207



Credit Fraud | | Dealing with Imbalanced Datasets

Python · Credit Card Fraud Detection

Notebook Input Output Logs Comments (657)

Run
861.1s

🕒 Version 70 of 70

- Finance
- Banking
- Data Visualization
- Classification
- Dimensionality Reduction

Credit Fraud Detector



Note: There are still aspects of this kernel that will be subjected to changes. I've noticed a recent increase of interest towards this kernel so I will focus more on the steps I took and why I took them to make it clear why I took those steps.

Before we Begin:

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

Computer Vision



▶ See all 1415 tasks

Natural Language Processing



▶ 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.
- 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=EHiORDZ31VA>
- Min-Yuh Day (2024), Python 101, <https://tinyurl.com/aintpupython101>