

Introduction to Data Science and Machine Learning Tools : Python, Jupyter, and Google Colab

Dr.Eng. Igi Ardiyanto

Dept. of Electrical and Information Engineering
UNIVERSITAS GADJAH MADA

PROFILE



Dr. IGI ARDIYANTO

Deep Learning • Computer Vision • Automation
Robotics • Embedded Systems

CONTACT
Dept. of Electrical Engineering and Information Tech.
Faculty of Engineering, Universitas Gadjah Mada
Jl. Grafika No.2 Bulaksumur
Yogyakarta, Indonesia 55281
E-mail : igi@ugm.ac.id

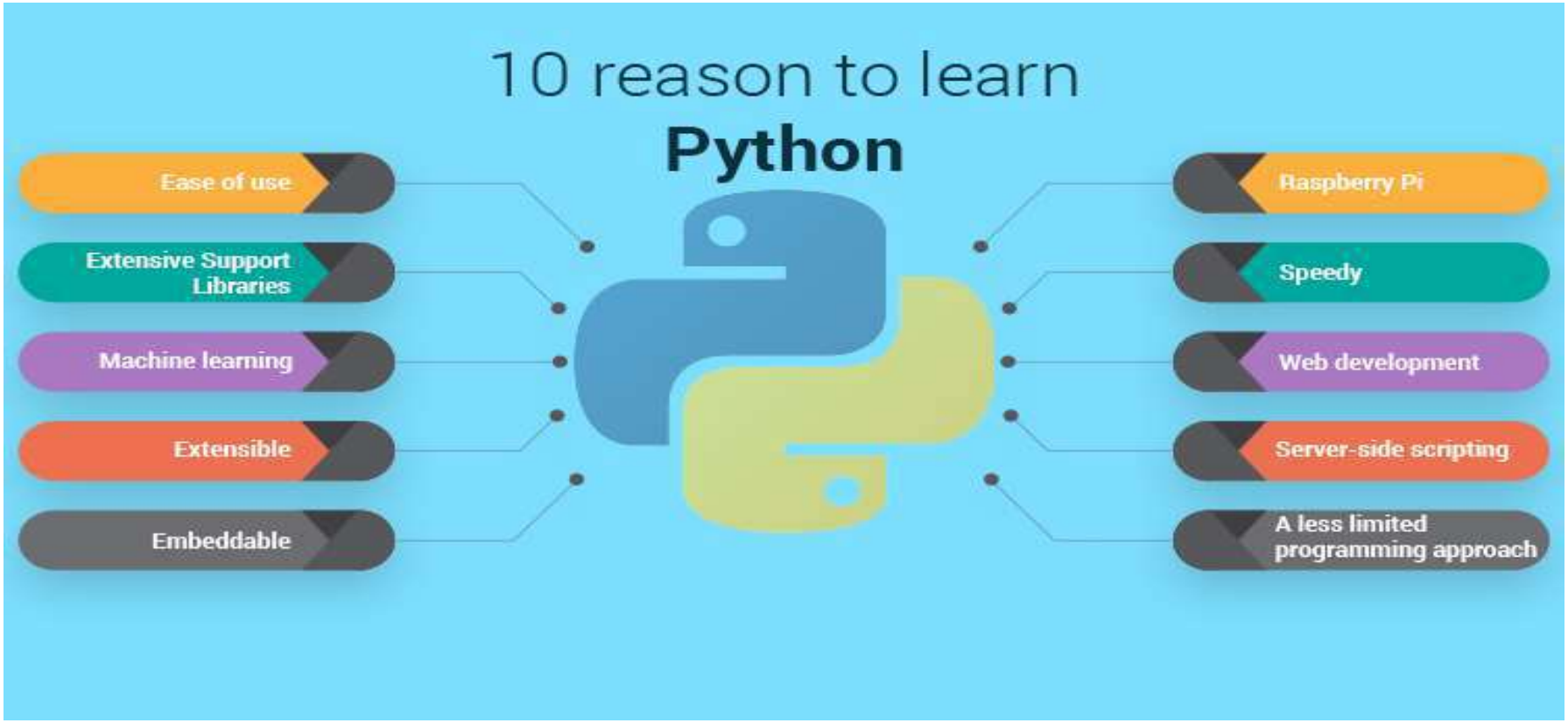
Current Position:
Head of Biomedical Engineering,
Bachelor Degree

More Information ??
http://te.ugm.ac.id/~igi

Academic :

- ❑2012 – 2015 Doctoral Degree, Toyohashi University of Technology, Japan
- ❑2010 – 2012 Master Degree, Toyohashi University of Technology, Japan

Why Python


10 reason to learn
Python
Ease of use
Extensive Support Libraries
Machine learning
Extensible
Embeddable
Raspberry Pi
Speedy
Web development
Server-side scripting
A less limited programming approach

Python Features

Elegant syntax

Easy to use language

Large standard library

Basic data types

Object-oriented programming with classes and multiple inheritance

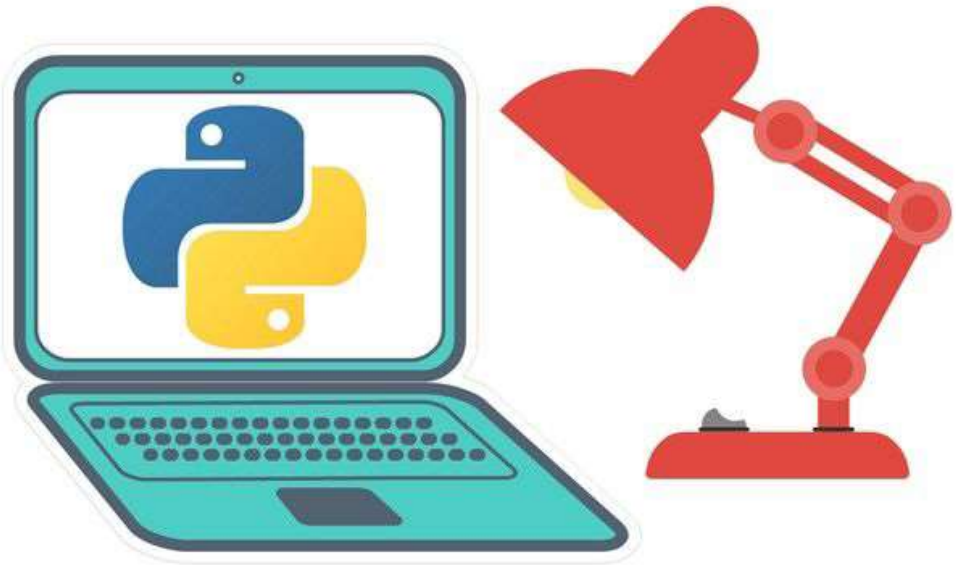
Free software

Python Applications



Python Version?

- Python 2 was started in 2000
- Python 2.7 was released in 2010
  - Will lose support in 2020
- Python 3.0 was released in 2008
  - More and more libraries are starting to support Python 3.4
- Which to use?
  - A lot more expansive support and resources for Python 2
  - Some Python 3 features are backwards compatible
  - BUT the future is looking towards Python 3

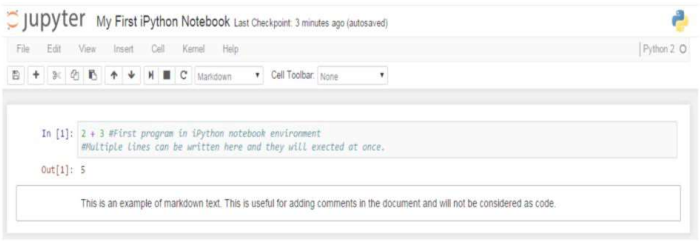


How to Install Python

Can download it from project site and install libraries individually (https://www.python.org/downloads/))

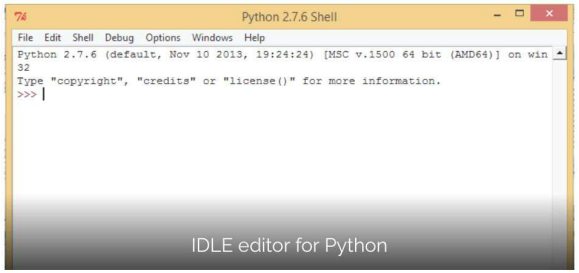
Comes pre-installed with Mac

Download Python with Anaconda distribution (https://www.anaconda.com/download/)



Development Environment

- Terminal
- IDLE editor
- Jupyter Notebook (previously called iPython Notebook)
  - try.jupyter.org



Jupyter Notebook



The browser hosts it, but it's pulling data from the directory you're running on your computer

Notebooks are downloadable as .ipynb files

Cell ➜ where you run the code

- also possible to write markdown
- `# Comments in Python`

Kernel is what your cell is running, the code that's running

Shortcuts

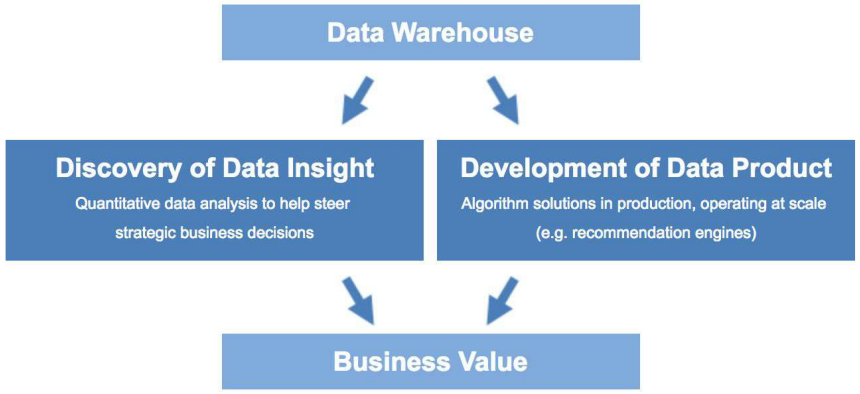
`Shift + Enter` → runs code

`Tab` → for autocomplete methods

`Shift + Tab` → expanded view of help popups

What is Data Science?

Data-driven science



Source: https://datajobs.com/what-is-data-science

Interdisciplinary field about scientific method to extract knowledge and insights from data in various forms

Includes machine learning, data mining, analytics, visualization, scraping, artificial intelligence etc

Data Science Concepts and Process

Data science relies on statistical analysis, BUT it is more than statistical analysis

Emphasis on project definition and collaboration

Data Science Project Lifecycle

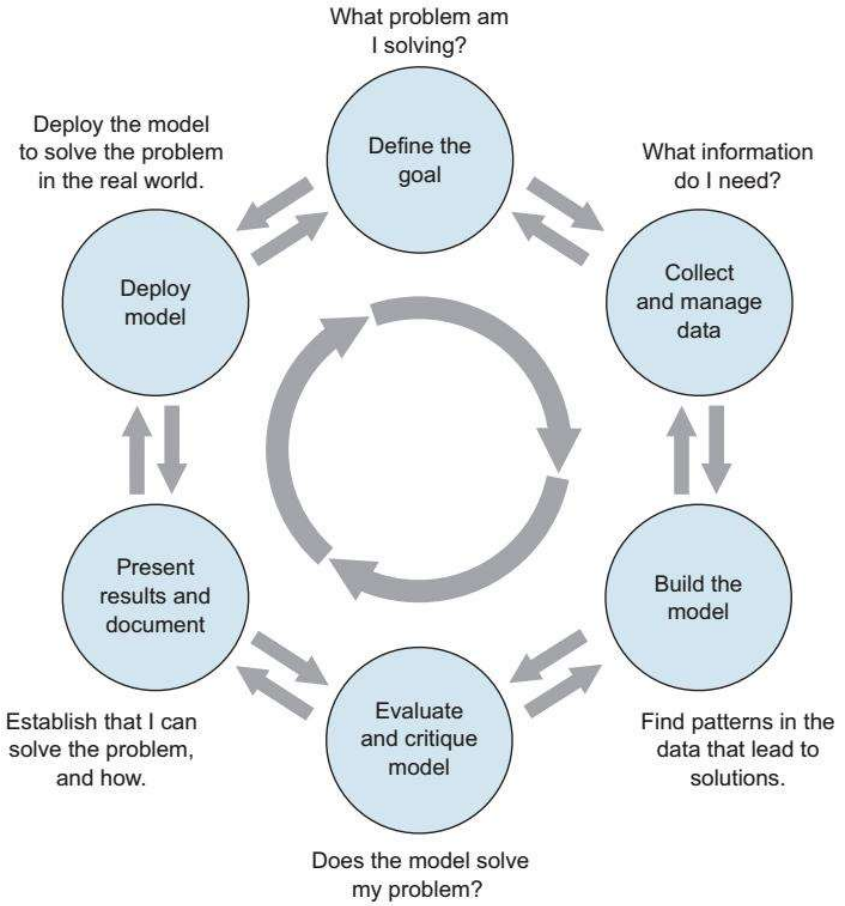
Project goal -- why are we doing this?

Data collection, quality, sufficiency, and management

Exploratory analysis

Model evaluation and sufficiency

Presentation to stakeholders, project documentation, and reproducibility



Python Developer Salaries

51 Salaries

Updated Jul 17, 2017

National Avg

$92,000

Min $60k

Max $137k

New York City, NY Area Avg

$82,063*

Min $62k

Max $112k

Data Scientist Salaries

2,303 Salaries

Updated Oct 27, 2017

National Avg

$128,549*

Min $92k

Max $168k

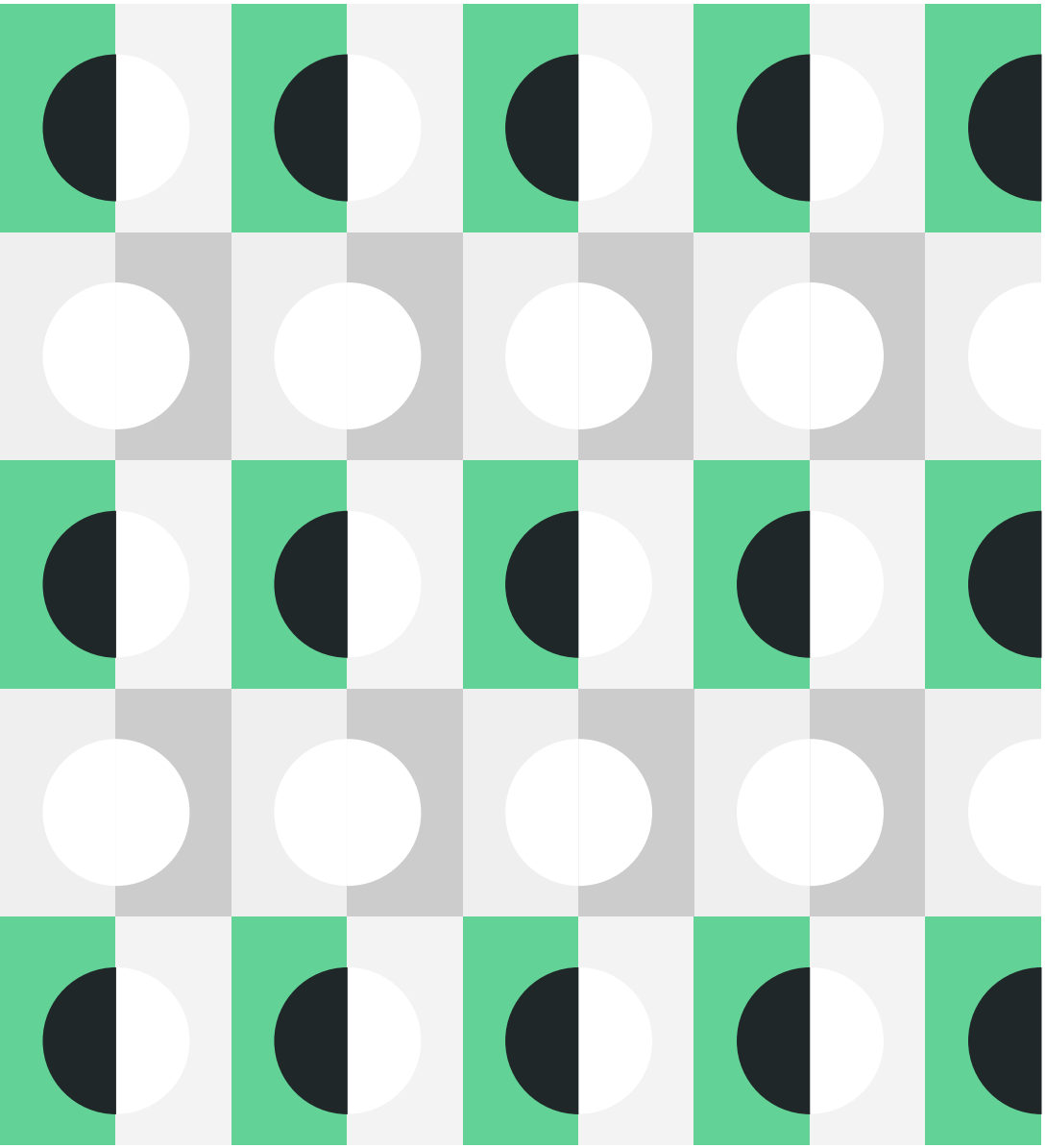
New York City, NY Area Avg

$124,917*

Min $95k

Max $159k

Source: http://www.glassdoor.com



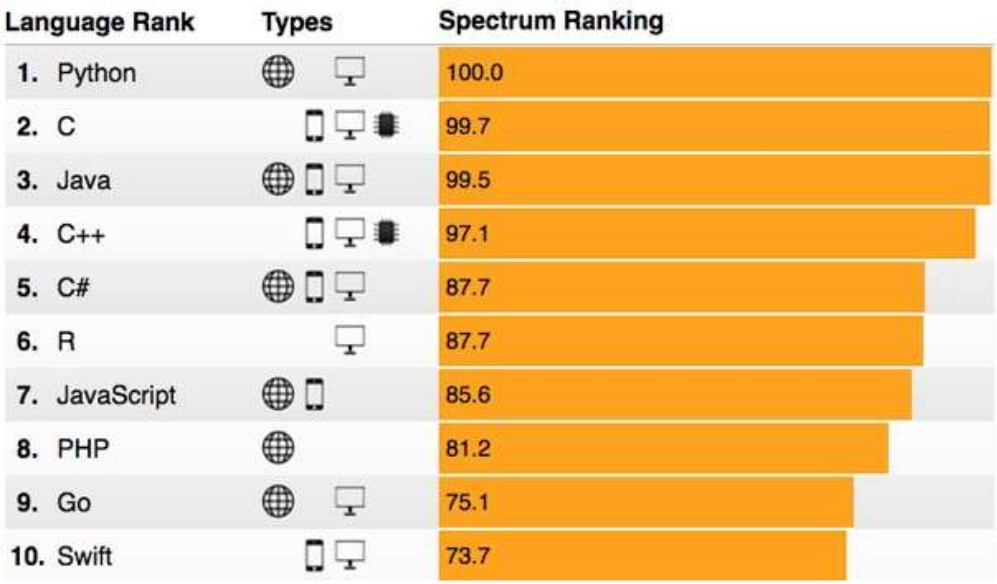
Intro to the Python Language

For Data Analysis:

- Get by with basic, key concepts
- Become familiar with libraries
- Use the technologies to your advantage

THE MOST POPULAR PROGRAMMING LANGUAGE – 2021

IEEE spectrum



Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C		99.7
3. Java		99.5
4. C++		97.1
5. C#		87.7
6. R		87.7
7. JavaScript		85.6
8. PHP		81.2
9. Go		75.1
10. Swift		73.7



Python vs Java

Java

- **Static typing →** everything must be explicitly declared
- **Verbose →** so many words!
- **Not compact**

Python

- **Dynamic typing →** an assignment statement binds a name to an object, the object can be of any type, can be later assigned to an object of a different type
- **Concise →** straight to the point!
- **Compact →** “It can all be apprehended at once in one’s head”

Differences between Python and Java

Java

```
public class HelloWorld
{
    public static void main (String[] args)
    {
        System.out.println("Hellold!");
    }
}
```

Python

```
print "Hello, world!"
```

```
print("Hello, world!") # Python version 3
```

Source: https://pythonconquerstheuniverse.wordpress.com/2009/10/03/python-java-a-side-by-side-comparison/

Differences between Python and Java

Java

```java
public static int fact(int n) {
    if (n < 0 || n > 12) {
        throw new IllegalArgumentException("The argument should be >= 0");
    } else {
        if (n == 0) {
            return 1;
        } else {
            return n * fact(n - 1);
        }
    }
}
```

Python

```python
def fact(n):
    if n==0 :
      return 1
    else :
      return n*fact(n-1);
```

```python
for i in range(100):
    print(fact(i))
```

Source: https://pythonconquerstheuniverse.wordpress.com/2009/10/03/python-java-a-side-by-side-comparison/

Numbers

- Integers, floats
- Basic arithmetic: addition, subtraction, multiplication, division
- Python 3 uses “true division” ➜ `3/2 = 1.5`
- Python 2 uses “classic division” ➜ `3/2 = 1`
  - Cast ➜ `float(3)/2 = 1.5`
  - Import python3 functions into python2 ➜

```
from __future__ import division
    3/2 = 1.5
```

- Powers ➜ `2**3 = 8`

Variable Assignment

```
In [9]: a = 5
        a + a

Out[9]: 10
```

```
[13]: income = 100
      rate = .1
      taxes = income * rate

      print(taxes)

      10.0
```

Strings

Strings use single or double quotes, depending on formatting

In [16]:
```
print ('Hello World 1!')
print ('Hello World 2!')
print ('You gotta use \n to print a new line.')
print ("Y'all also gotta be careful with how you use quotes!")
```

```
Hello World 1!
Hello World 2!
You gotta use
 to print a new line.
Y'all also gotta be careful with how you use quotes!
```

String Manipulation

Strings are *sequences* and can be indexed

```
In [22]: str = 'Hello World 1'
         print(str[6])
```

```
W
```

Grab the length of a string using `len()`

```
len(str)
```

```
13
```

Use `:` to perform slicing

```
print(str[1:])
print(str[:3])
print(str[-1])
print(str[:-1])
print(str[::2])
print(str[::-1])
```

```
ello World 1
Hel
1
Hello World
HloWrd1
1 dlroW olleH
```

```
letter = 'z'
letter * 10
```

```
'zzzzzzzzzz'
```

Strings are immutable → once created, they cannot be changed or replaced, but you can concatenate

```
print(str.upper())
print(str.lower())
print(str.split())
print(str.split('W'))
```

```
HELLO WORLD 1
hello world 1
['Hello', 'World', '1']
['Hello ', 'orld 1']
```

Lists

Lists can work similarly to strings -- they use the `len()` function and square brackets to access data

```
colors = ['red', 'blue', 'green']
print colors[0]    ## red
print colors[2]    ## green
print len(colors)  ## 3
```

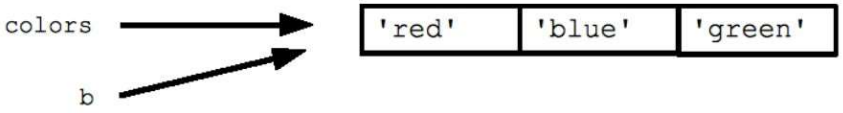



```
dc = ['beyonce', 'kelly', 'michelle']
if 'beyonce' in dc :
    print("say my name")
```

```
say my name
```

Assignment with = will not make a copy, it will make the 2 variables point to the same same list

```
b = colors   ## Does not copy the list
```




```
print("the members in destiny's child are:")
i = 0
while i < len(dc) :
    print(dc[i])
    i = i+1
```

```
the members in destiny's child are:
beyonce
kelly
michelle
```

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

Tuples

- Sequence of immutable Python objects, like lists
- Tuples cannot be changed (immutable), but lists can
- Fixed size, whereas lists are dynamic
- You cannot remove elements from a tuple (no remove or pop method)
- Faster than lists -- if you ever need to define a constant set of values to iterate through, tuples are preferable

```
# tuples
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5, 6, 7 )
print("tup1[0]: ", tup1[0]) # the item in index 0
print("tup2[1:5]: ", tup2[1:5]) # items in from index 1 to index 5
```

```
tup1[0]:  physics
tup2[1:5]:  (2, 3, 4, 5)
```

Source: https://www.tutorialspoint.com/python/python_tuples.htm

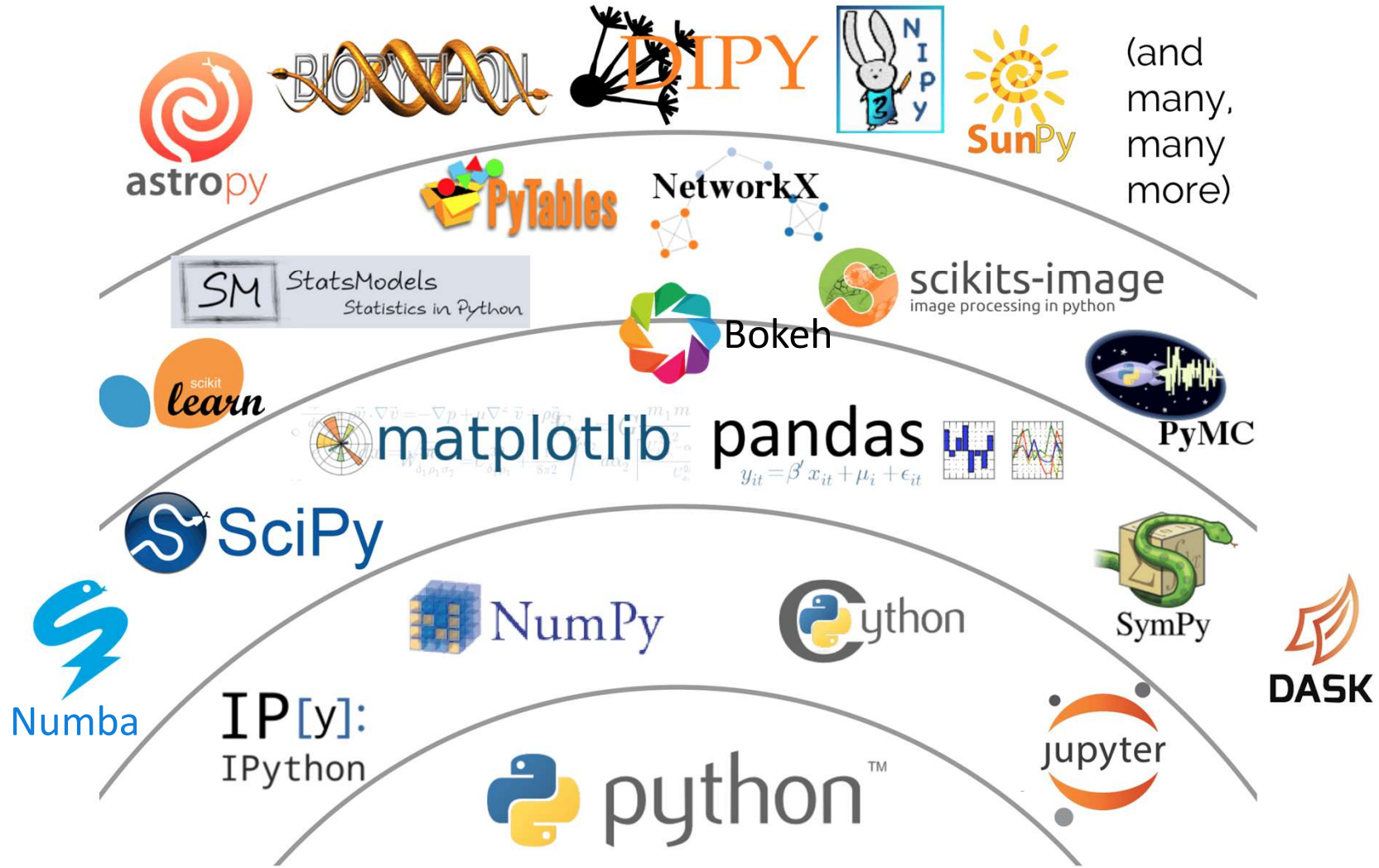
Dictionaries

- Associative array, also known as hash
- Any key in the dictionary is associated or mapped to a value
- Unordered key-value-pairs

```
# dictionaries
scoreboard = {'brady': 1204, 'jane': 4839,
              'arnold': 3999, 'pj': 2530}
scoreboard['jane']
```

```
4839
```

Python Libraries



© 2016 Continuum Analytics - Confidential & Proprietary



SciKit-Learn

Machine learning module built on top of SciPy

Started in 2007 by David Cournapeau as a Google Summer of Code project

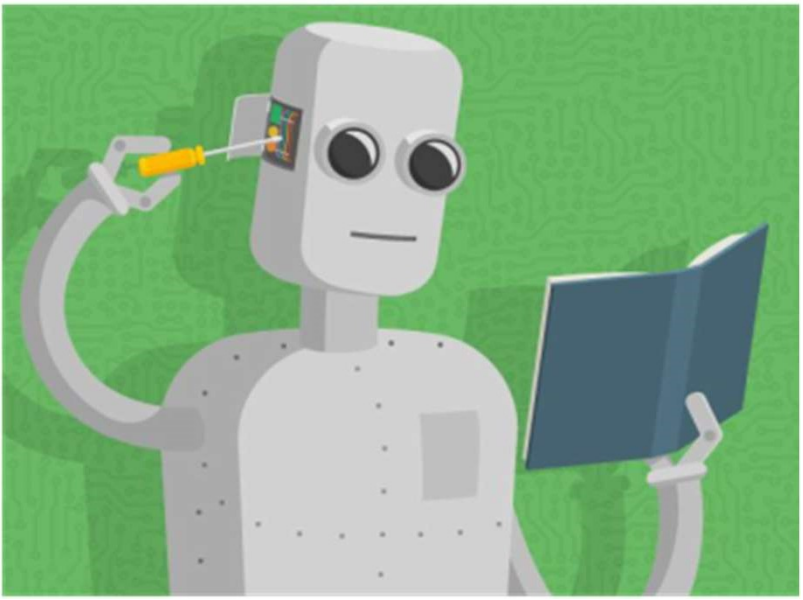
Currently maintained by volunteers

1. Install Dependency using **Python Package Manager**
    a. Package that code depends on

```
MAC: pip install -U scikit-learn

WINDOWS: python -m pip install -U pip

Or with conda:
conda install scikit-learn
```



Source: https://github.com/scikit-learn/scikit-learn, http://scikit-learn.org/stable/index.html

Breaking it Down

2. Import Dependency and sub-module ➜ tree (to build a decision tree)

3. Create data sets in lists (list of lists)

4. Store decision tree classifier initialize using fit method

5. Print to terminal

```
from sklearn import tree
# [height(cm), weight(lb), shoe size(cm)]

#list of lists
X = [[163,120,24], [185,170,28], [190,200,30], [165,130,22],
                [170,149,25], [182,177,27], [172,158,26], [180,170,26]]
Y = ['female','male', 'male', 'female', 'female', 'male', 'male',
          'female']

clf = tree.DecisionTreeClassifier()

# mapping measurements to gender
clf = clf.fit(X,Y)

prediction = clf.predict([[175,145,25]])

print(prediction)
```

pandas

Popular python package for data analysis & manipulation

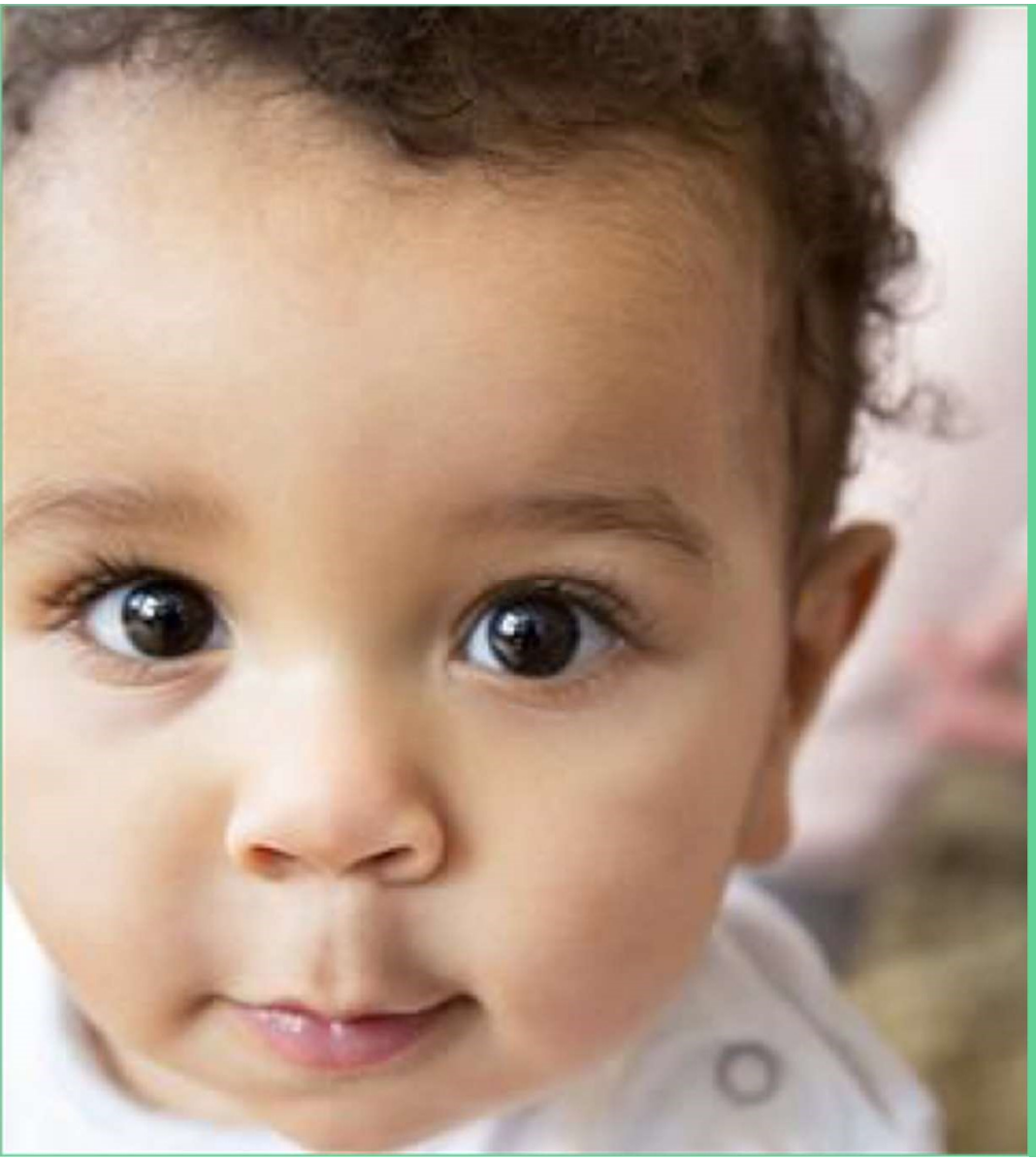
Well suited for ordered and unordered data, tabular data, arbitrary matrix data, observational/statistical data

- Python package pro
- Install using conda or pip

```
pip install pandas
```



Source: https://github.com/pandas-dev/pandas



Popular Baby Names

Using Pandas and matplotlib for Data Analysis

1. Environment Setup
2. Create data set
3. Get data → read it from text
4. Prepare data → making sure data is clean
5. Analyze data
6. Present data

Source:
http://nbviewer.jupyter.org/urls/bitbucket.org/hrojas/learn-pandas/raw/master/lessons/01%20-%20Lesson.ipynb
https://www.babycenter.com/top-baby-names-2016.htm
https://www.ssa.gov/oact/babynames/index.html

Environment Setup

```
# General syntax to import specific functions in a library:
##from (library) import (specific library function)
from pandas import DataFrame, read_csv

# General syntax to import a library but no functions:
##import (library) as (give the library a nickname/alias)
import matplotlib.pyplot as plt
import pandas as pd

# Enable inline plotting
%matplotlib inline
```

Create Data Set

```
names = ["Noah", "Emma", "Liam", "Olivia",
         "William", "Ava", "Mason", "Sophia"]
# number of births with that name in 2016
births = [19015, 19414, 18138, 19246, 15668, 16237, 15192, 16070]
```

Merge the lists together using `zip()`

```
BabyDataSet = list(zip(names, births))
BabyDataSet
```

```
[('Noah', 19015),
 ('Emma', 19414),
 ('Liam', 18138),
 ('Olivia', 19246),
 ('William', 15668),
 ('Ava', 16237),
 ('Mason', 15192),
 ('Sophia', 16070)]
```

Create Data Set ➜ Create DataFrame

```
# create a dataFrame object --> holds contents of
# BabyDataSet in a format similar
# to sql table or spreadsheet

df = pd.DataFrame(data = BabyDataSet,
                  columns=['Names', 'Births'])
df
```

	Names	Births
0	Noah	19015
1	Emma	19414
2	Liam	18138
3	Olivia	19246
4	William	15668
5	Ava	16237
6	Mason	15192
7	Sophia	16070

Create Data Set ➜ Create .csv

```
# create csv file
df.to_csv('births2016.csv',index=False,header=False)

Location = r'births2016.csv'
df = pd.read_csv(Location)
```

Make a .csv out of the DataFrame

`Location` sets where you want the .csv to be saved

- Prefacing the location string with `r` escapes the string if you output the file to a different directory

Get Data → Read .csv

```
Location = r'births2016.csv'
df = pd.read_csv(Location)
```

```
df
```

`read_csv` pulls in the data from the csv into the console

- Reads the first entry as the header

	Noah	19015
0	Emma	19414
1	Liam	18138
2	Olivia	19246
3	William	15668
4	Ava	16237
5	Mason	15192
6	Sophia	16070

Get Data ➔ Edit .csv

```
df = pd.read_csv(Location, header=None)
df
```

	0	1
0	Noah	19015
1	Emma	19414
2	Liam	18138
3	Olivia	19246
4	William	15668
5	Ava	16237
6	Mason	15192
7	Sophia	16070

```
df = pd.read_csv(Location,
                 names=['Names', 'Births'])
df
```

	Names	Births
0	Noah	19015
1	Emma	19414
2	Liam	18138
3	Olivia	19246
4	William	15668
5	Ava	16237
6	Mason	15192
7	Sophia	16070

Prepare Data ➜ Make sure it's clean

```
# Check data type of the columns
df.dtypes
```

```
Names      object
Births      int64
dtype: object
```

- Births are type ***int64*** meaning, no floats or alpha numeric characters will be present

Analyze Data

- Find the most popular baby name with highest birth rate
  - Sort the DataFrame and select the top row
  - OR use the `max()` attribute to find the max value

```
Sorted = df.sort_values(['Births'],
                        ascending=False)
Sorted.head(1)
```

	Names	Births
1	Emma	19414

```
df['Births'].max()
```

```
19414
```

Present Data ➜ Plot the DataFrame

- Plot the ***Births*** column and label the graph to show the highest point on the graph ➜ with the table, the end user can navigate the data clearly
- `plot()` is a pandas attribute that lets you plot the data in the dataframe

```
# Create graph
df['Births'].plot()

# Maximum value in the data set
MaxValue = df['Births'].max()

# Name associated with the maximum value
MaxName = df['Names'][df['Births'] == df['Births'].max()].values

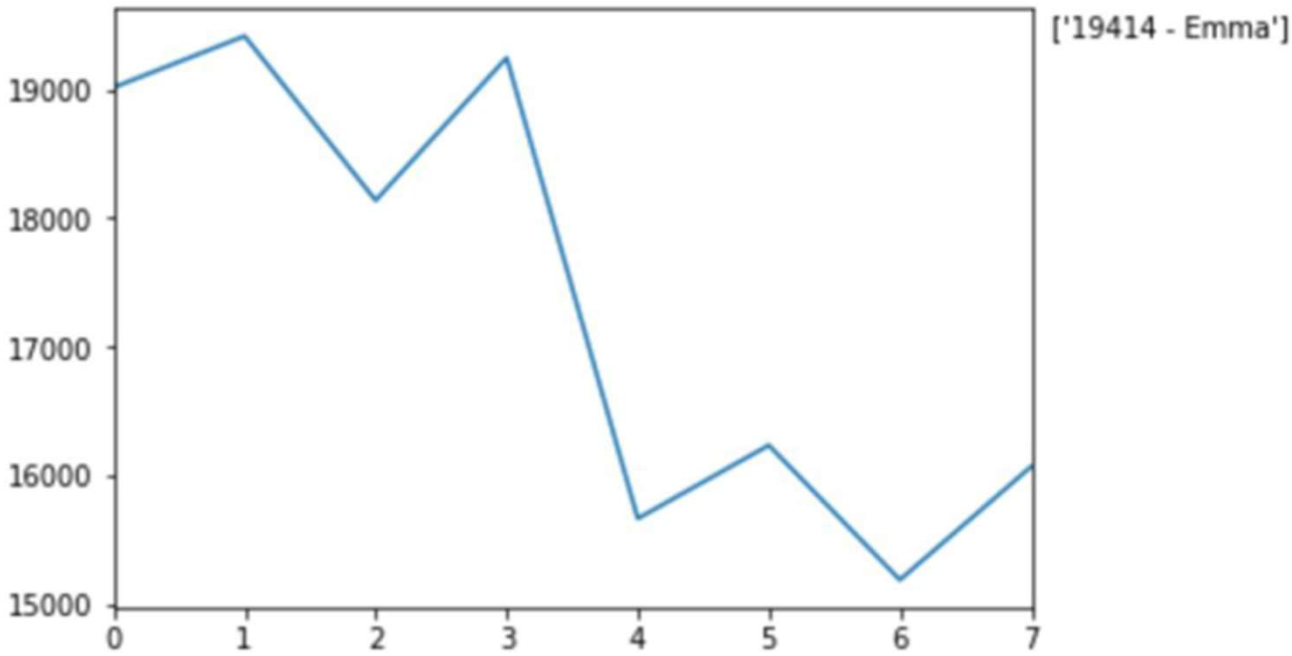
# Text to display on graph
Text = str(MaxValue) + " - " + MaxName

# Add text to graph
plt.annotate(Text, xy=(1, MaxValue), xytext=(8, 0),
                 xycoords=('axes fraction', 'data'),
             textcoords='offset points')

print("The most popular name")
df[df['Births'] == df['Births'].max()]
#Sorted.head(1) can also be used
```

The most popular name

	Names	Births
1	Emma	19414





Jupyter Notebooks

Introduction to Interactive Jupyter Notebooks



Code

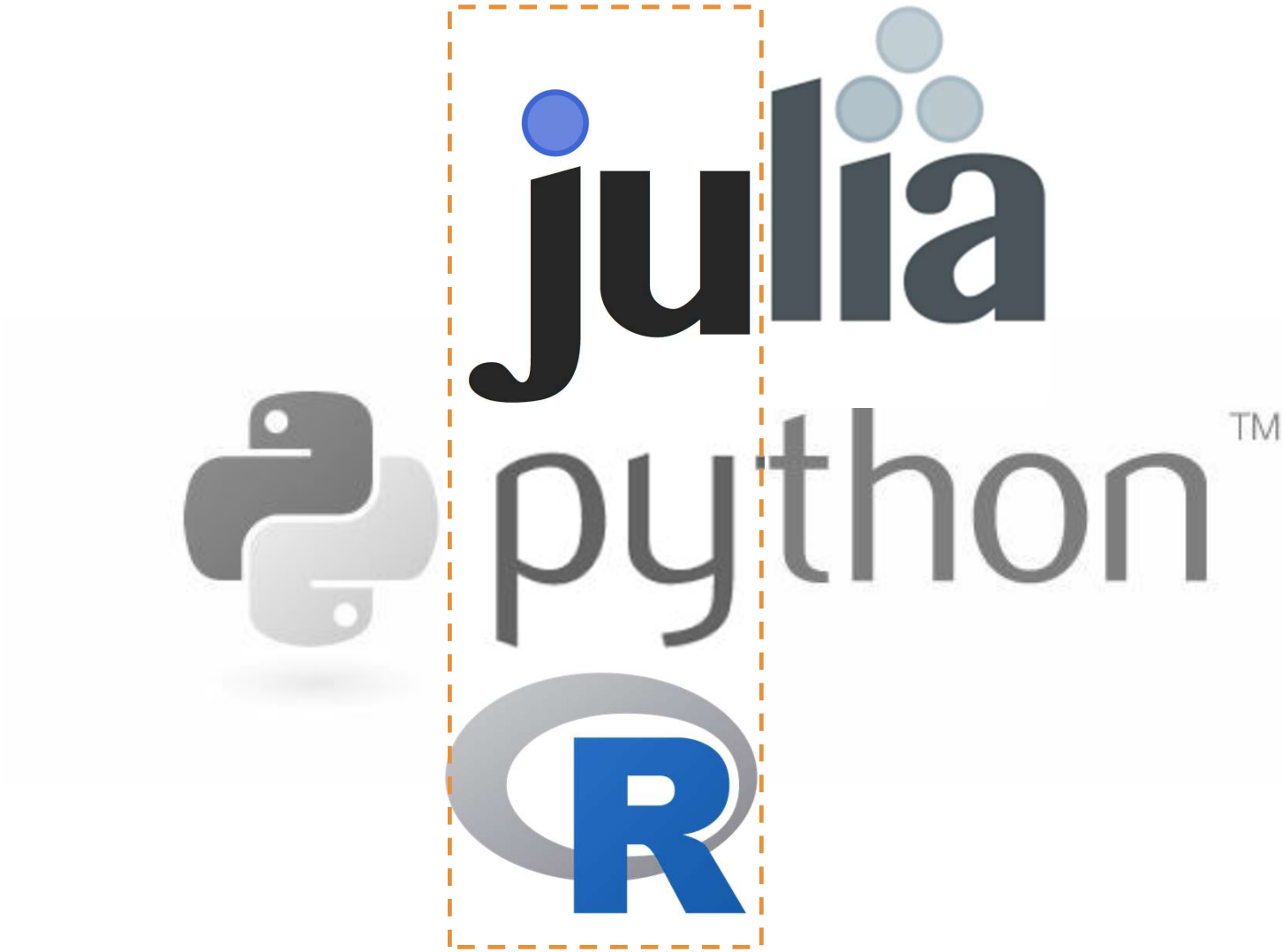


Rich Text



Terminal-like

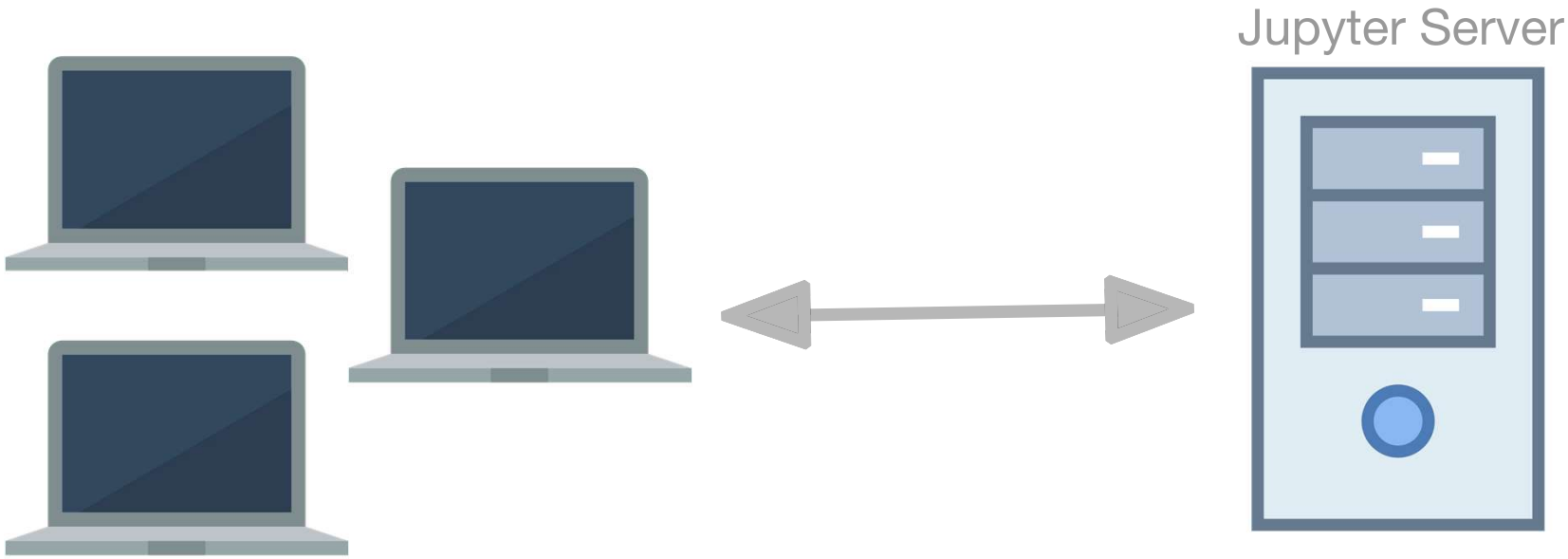
Jupyter Notebooks are documents that contain both code and rich text with some of the magic of the terminal


julia
python™
R



Get all the kernels! https://github.com/jupyter/jupyter/wiki/Jupyter-kernels

How does it work?



Or hosted locally typically on **localhost:8888**

Installation

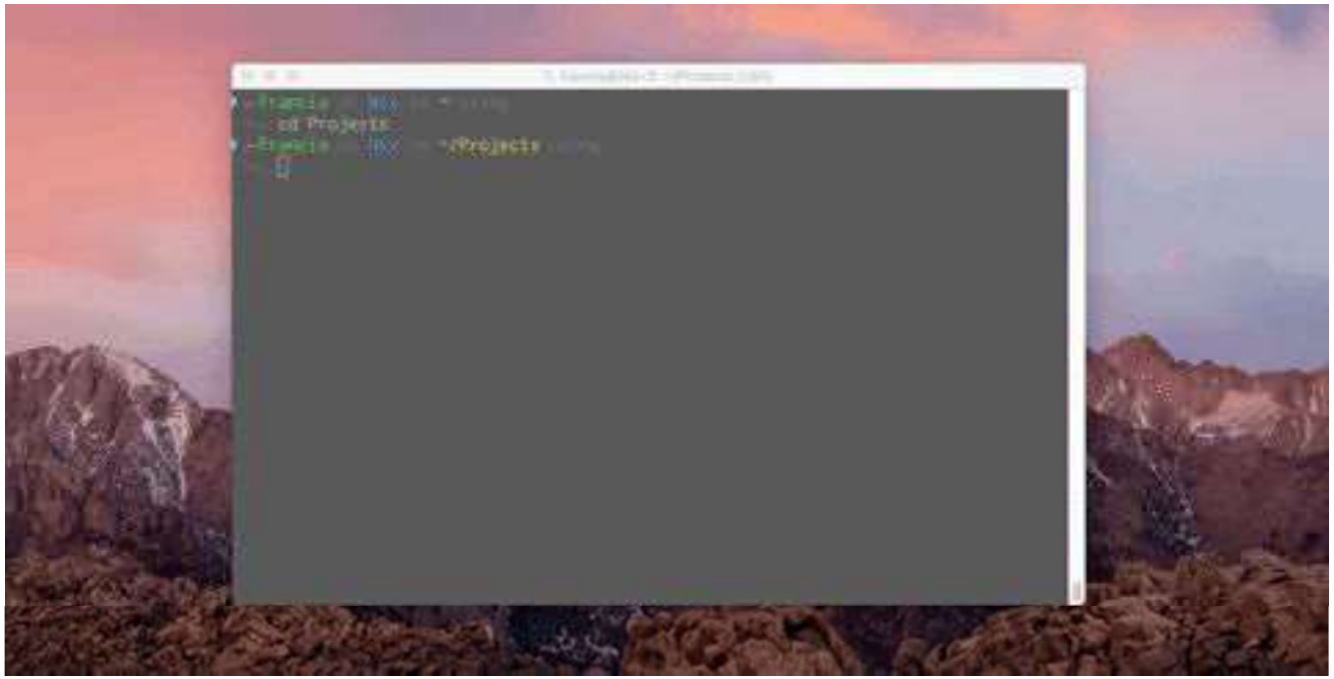
```
# On Windows
python -m pip install -U pip setuptools
# On OS X or Linux
pip install -U pip setuptools
```

```
# Python2
pip install jupyter
# Python 3
pip3 install jupyter
```

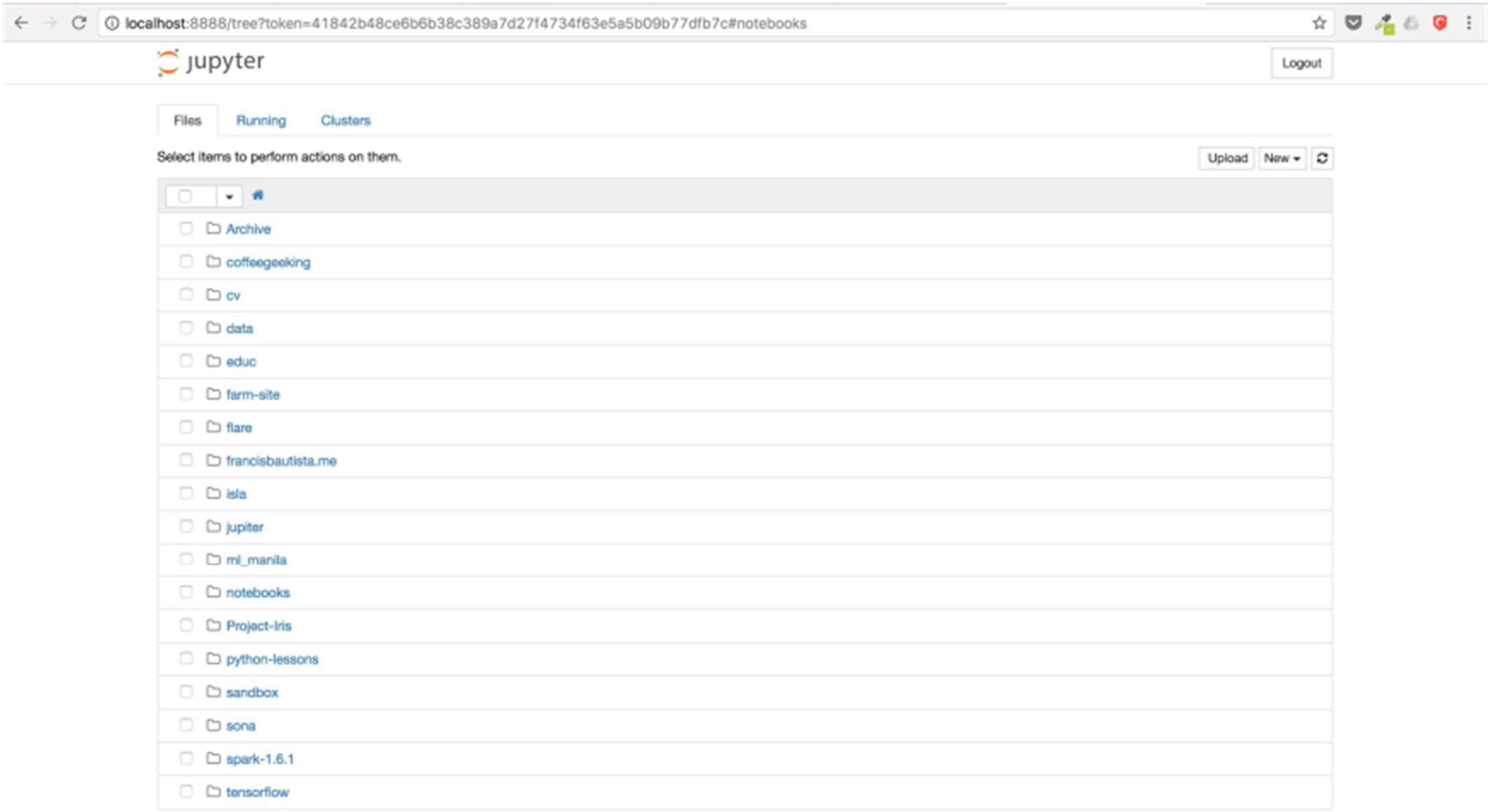
Installation



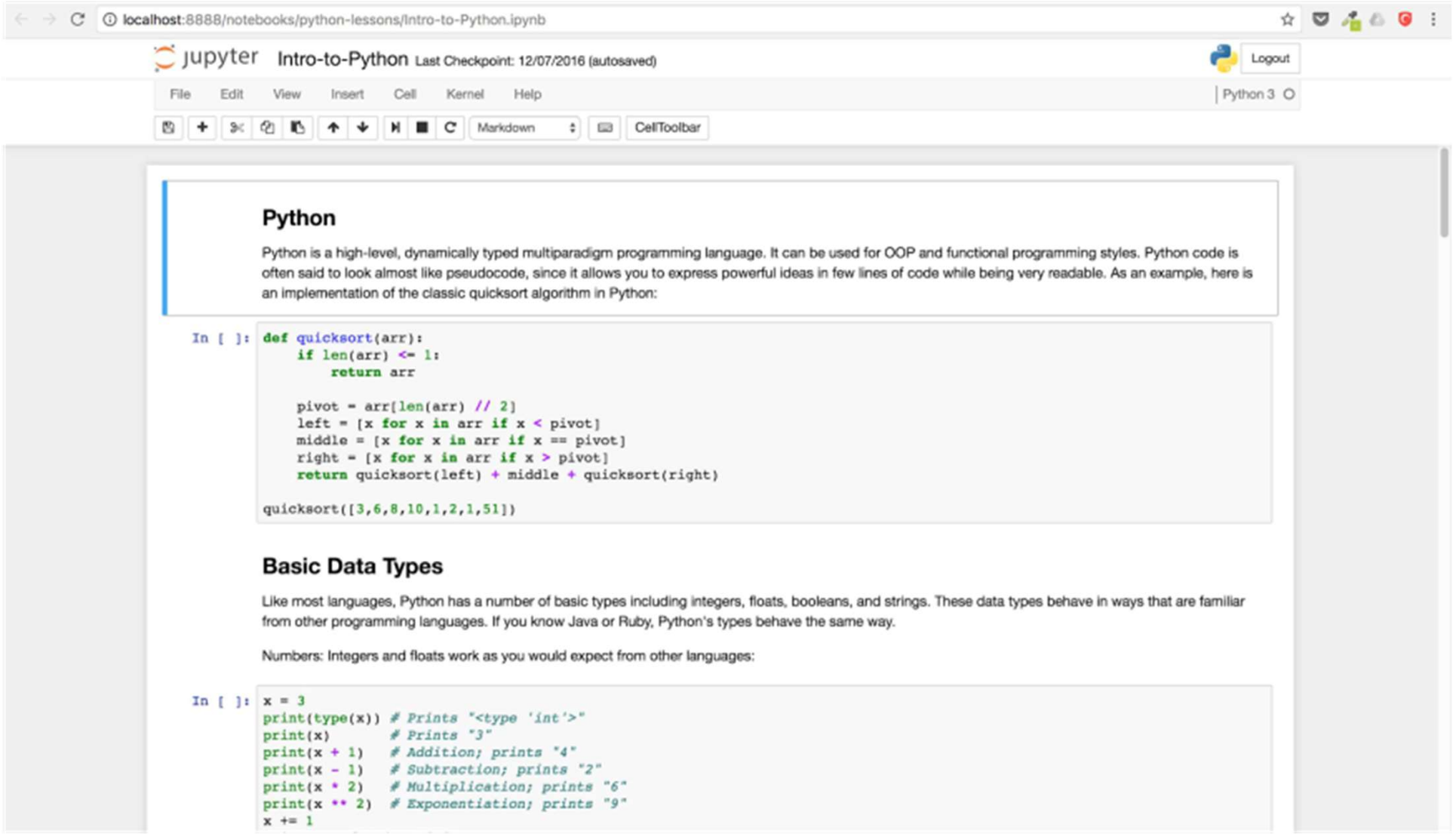
If you'd rather avoid the hassle of installing packages individually, just download and install Anaconda. It hosts all of the most popular scientific Python libraries and is generally more stable.

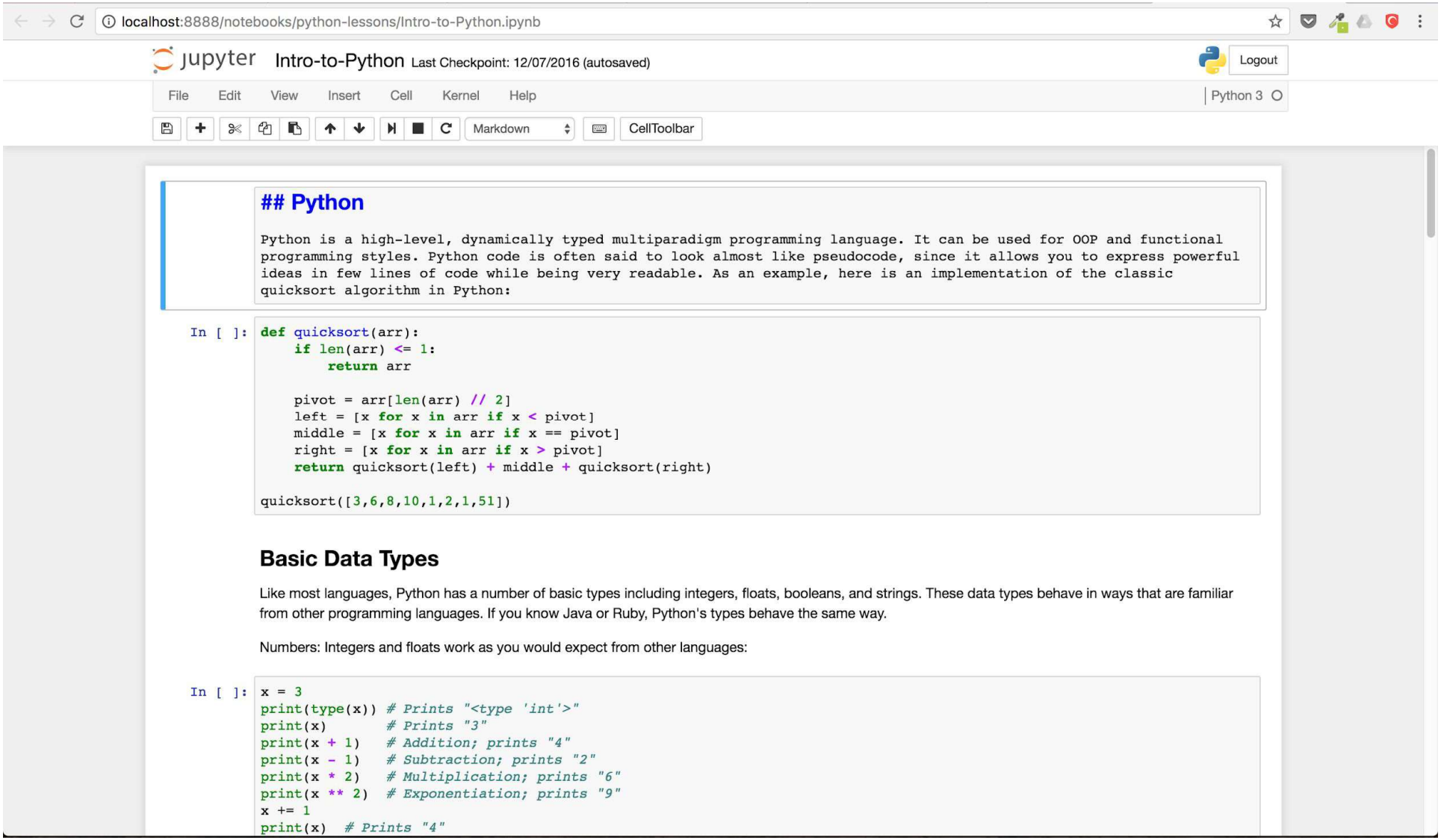


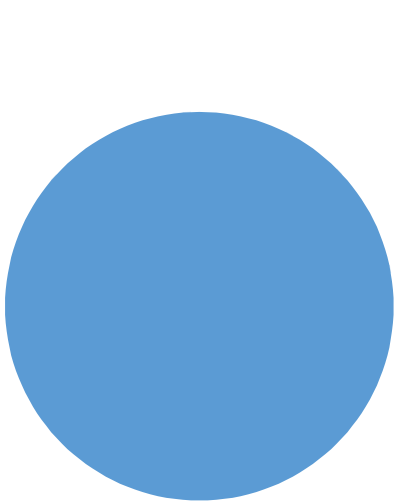
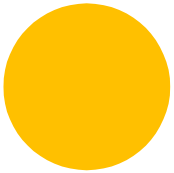
Type `$ jupyter notebook` in a terminal window to launch


localhost:8888/tree?token=41842b48ce6b6b38c389a7d27f4734f63e5a5b09b77dfb7c#notebooks
jupyter
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
Archive
coffeegeeking
cv
data
educ
farm-site
flare
francisbautista.me
isla
jupiter
ml_manila
notebooks
Project-Iris
python-lessons
sandbox
sona
spark-1.6.1
tensorflow


localhost:8888/tree/python-lessons
jupyter
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
/ python-lessons
..
data
Intro-to-Python.ipynb
Matplotlib.ipynb
Numpy.ipynb
Pandas + Folium Demo.ipynb
Pandas Demo.ipynb
Pandas-Intro.ipynb
coffeeshops.csv
README.md


localhost:8888/notebooks/python-lessons/Intro-to-Python.ipynb
jupyter
Intro-to-Python
Last Checkpoint: 12/07/2016 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Python 3
Markdown
CellToolbar
Python
Python is a high-level, dynamically typed multiparadigm programming language. It can be used for OOP and functional programming styles. Python code is often said to look almost like pseudocode, since it allows you to express powerful ideas in few lines of code while being very readable. As an example, here is an implementation of the classic quicksort algorithm in Python:
In [ ]:
def quicksort(arr):
if len(arr) <= 1:
return arr
pivot = arr[len(arr) // 2]
left = [x for x in arr if x < pivot]
middle = [x for x in arr if x == pivot]
right = [x for x in arr if x > pivot]
return quicksort(left) + middle + quicksort(right)
quicksort([3,6,8,10,1,2,1,51])
Basic Data Types
Like most languages, Python has a number of basic types including integers, floats, booleans, and strings. These data types behave in ways that are familiar from other programming languages. If you know Java or Ruby, Python's types behave the same way.
Numbers: Integers and floats work as you would expect from other languages:
In [ ]:
x = 3
print(type(x)) # Prints "<type 'int'>"
print(x) # Prints "3"
print(x + 1) # Addition; prints "4"
print(x - 1) # Subtraction; prints "2"
print(x * 2) # Multiplication; prints "6"
print(x ** 2) # Exponentiation; prints "9"
x += 1

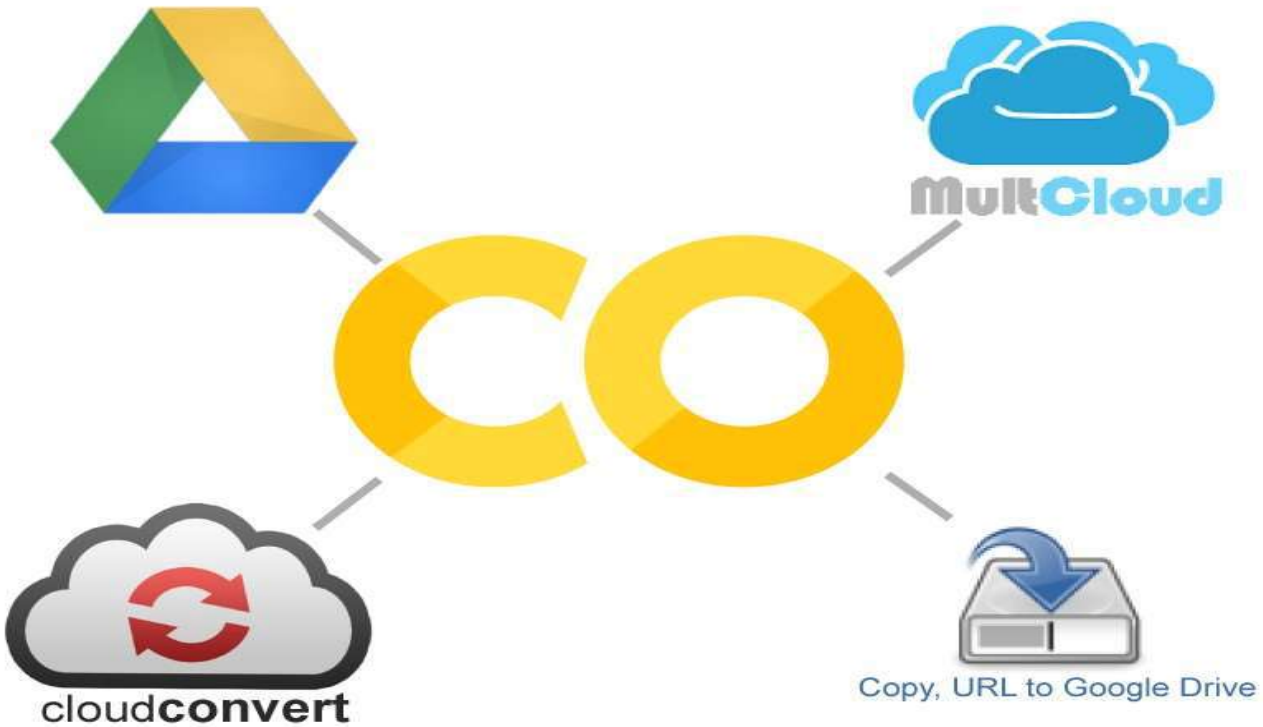

localhost:8888/notebooks/python-lessons/Intro-to-Python.ipynb
jupyter
Intro-to-Python
Last Checkpoint: 12/07/2016 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Python 3
Markdown
CellToolbar
Python
Python is a high-level, dynamically typed multiparadigm programming language. It can be used for OOP and functional programming styles. Python code is often said to look almost like pseudocode, since it allows you to express powerful ideas in few lines of code while being very readable. As an example, here is an implementation of the classic quicksort algorithm in Python:
In [ ]:
def quicksort(arr):
if len(arr) <= 1:
return arr
pivot = arr[len(arr) // 2]
left = [x for x in arr if x < pivot]
middle = [x for x in arr if x == pivot]
right = [x for x in arr if x > pivot]
return quicksort(left) + middle + quicksort(right)
quicksort([3,6,8,10,1,2,1,51])
Basic Data Types
Like most languages, Python has a number of basic types including integers, floats, booleans, and strings. These data types behave in ways that are familiar from other programming languages. If you know Java or Ruby, Python's types behave the same way.
Numbers: Integers and floats work as you would expect from other languages:
In [ ]:
x = 3
print(type(x)) # Prints "<type 'int'>"
print(x) # Prints "3"
print(x + 1) # Addition; prints "4"
print(x - 1) # Subtraction; prints "2"
print(x * 2) # Multiplication; prints "6"
print(x ** 2) # Exponentiation; prints "9"
x += 1
print(x) # Prints "4"



Python Programming with Google Colab

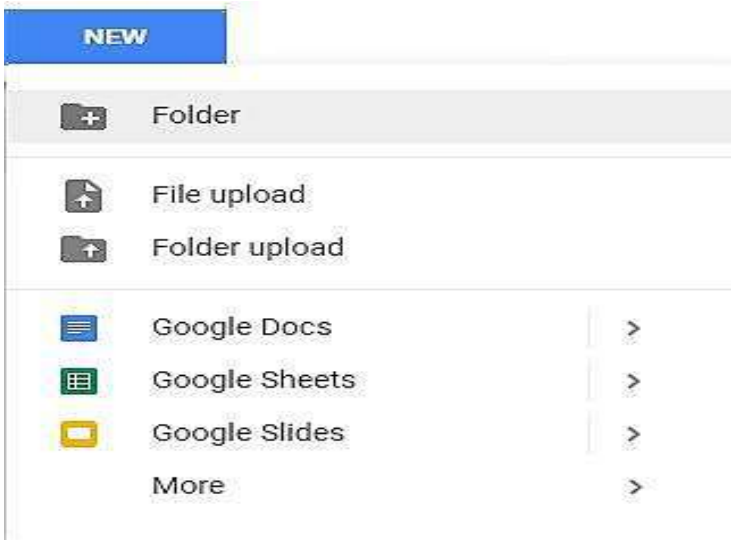
Google Colab

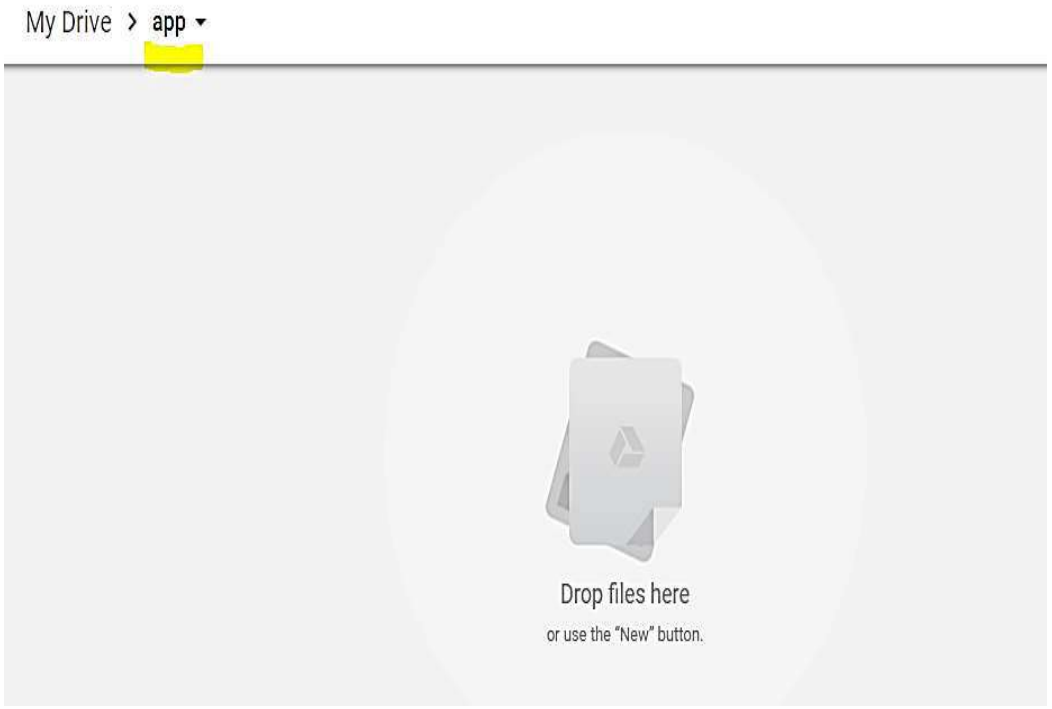


Why Google Colab

- Google Colab comes with collaboration backed in the product.
- It also runs on Google servers
- Don’t need to install anything
- Notebooks are saved to your Google Drive account
- Google Colab recently added support for Tensor Processing Unit (TPU) apart from its existing GPU and CPU instances. So, it’s a big deal for all deep learning people

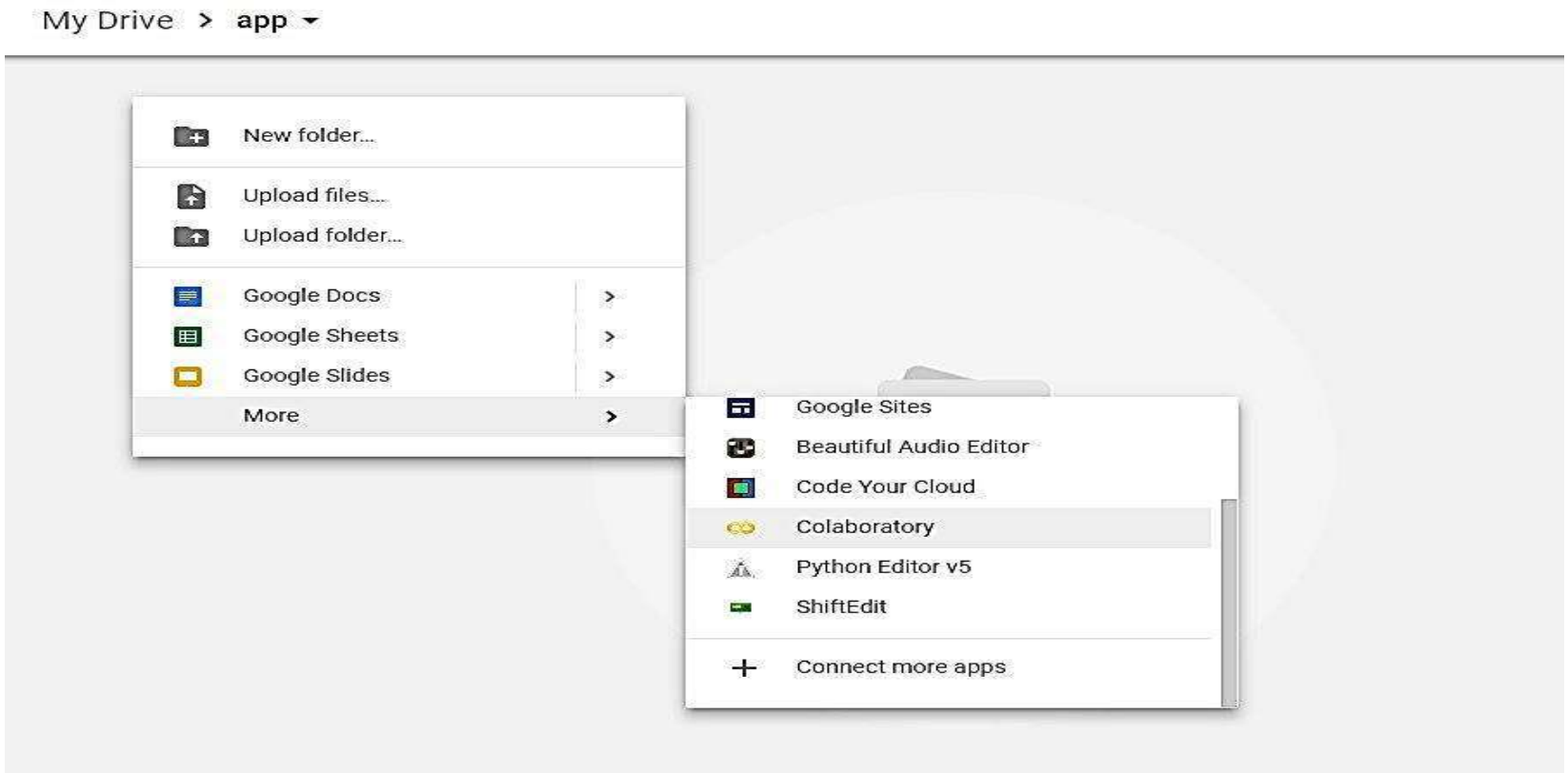
Creating Folder on Google Drive





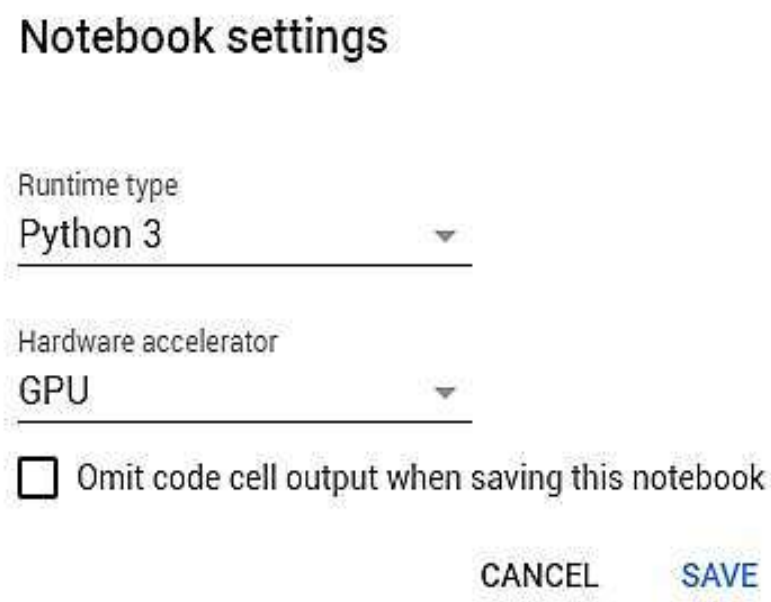
Creating New Colab Notebook

Create a new notebook via **Right click > More > Colaboratory**

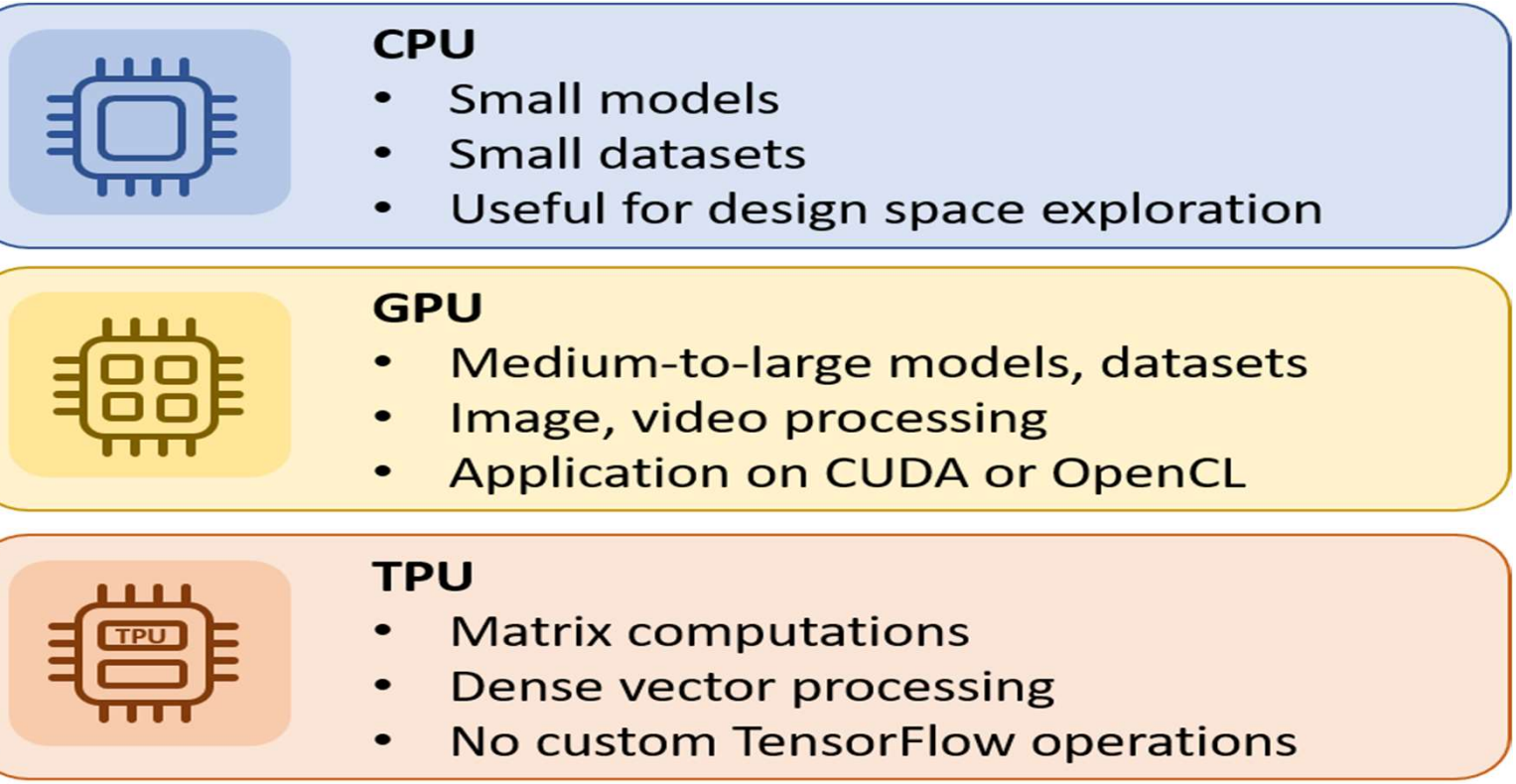


Notebook Settings

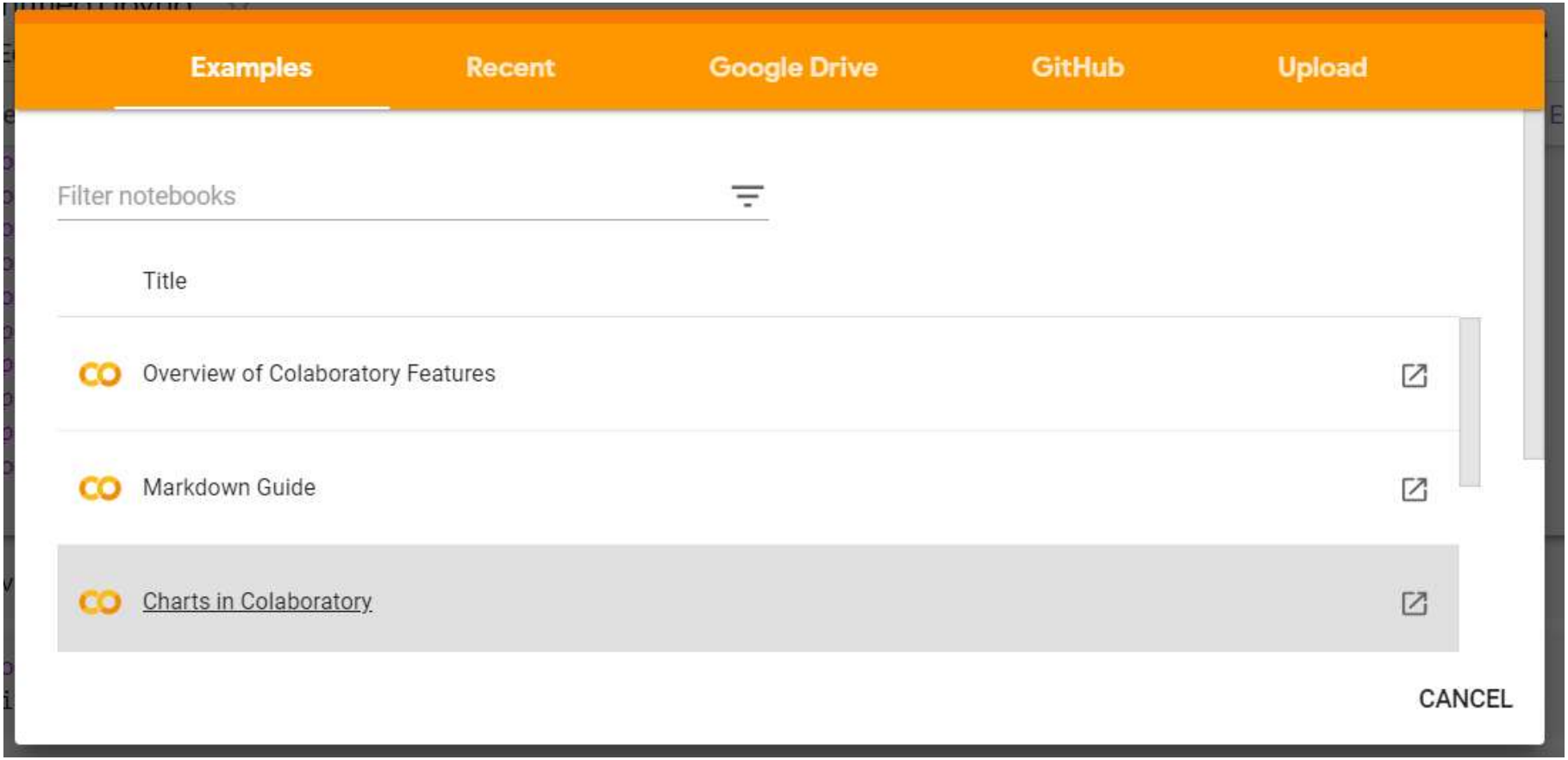
Edit > Notebook settings or **Runtime>Change runtime type** and **select GPU** as **Hardware accelerator**



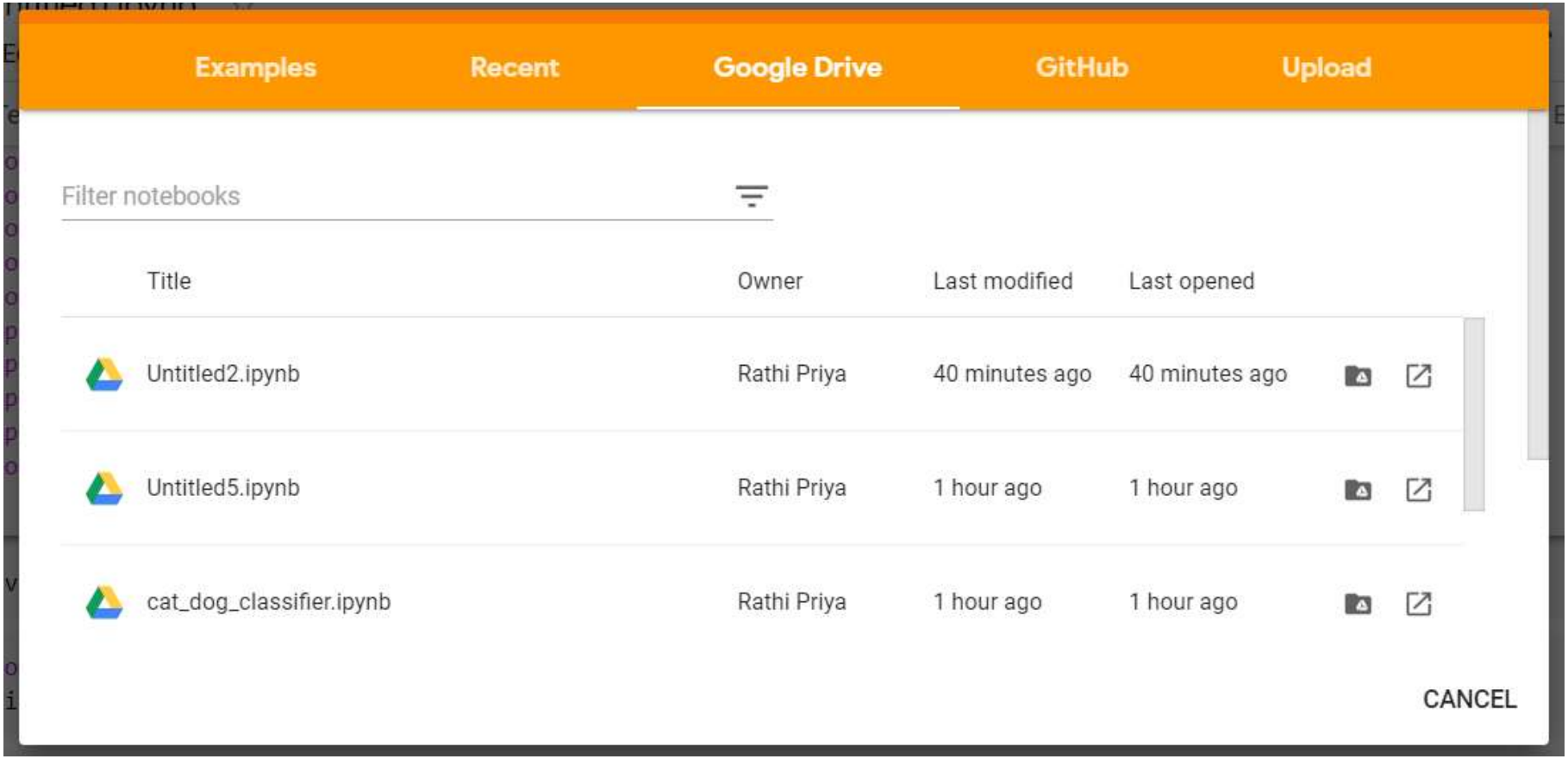
Hardware accelerator



Open python notebook in different ways



Google Drive



Running Basic Python Codes with Google Colab




```
[1]  x = 3

[2]  print(type(x)) # Prints "<class 'int'>"

     <type 'int'>

[3]  print(x)       # Prints "3"

     3

     print(x + 1)   # Addition; prints "4"

     4
```

Running or Importing .py Files with Google Colab

```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?c

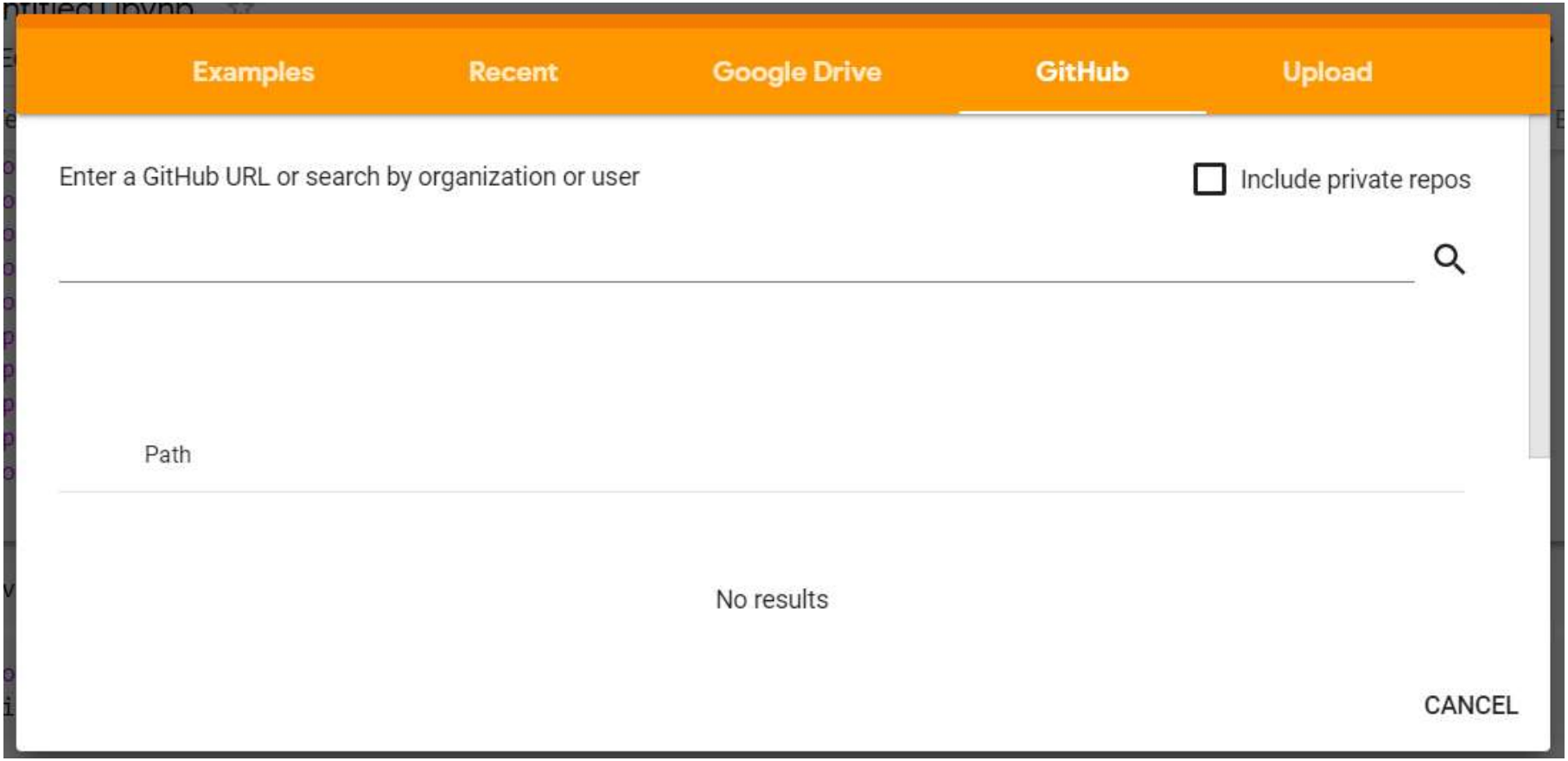
Enter your authorization code:
```

```
from google.colab import drive
drive.mount('/content/drive/')
```

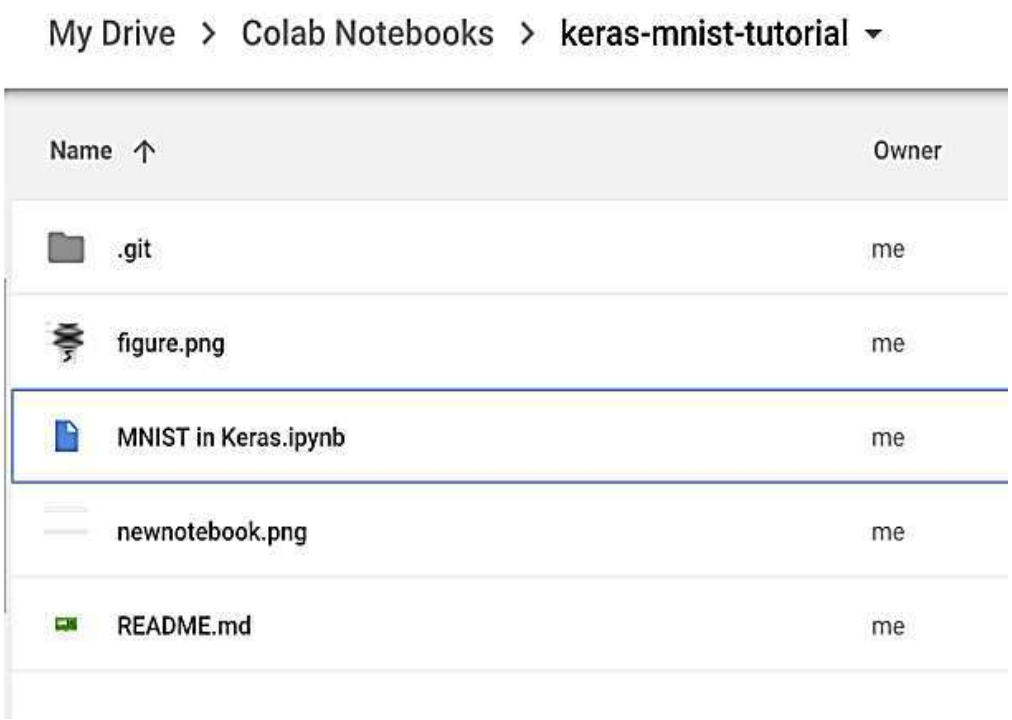
```
Go to this URL in a browser: https://accounts.google.com/o/oaut

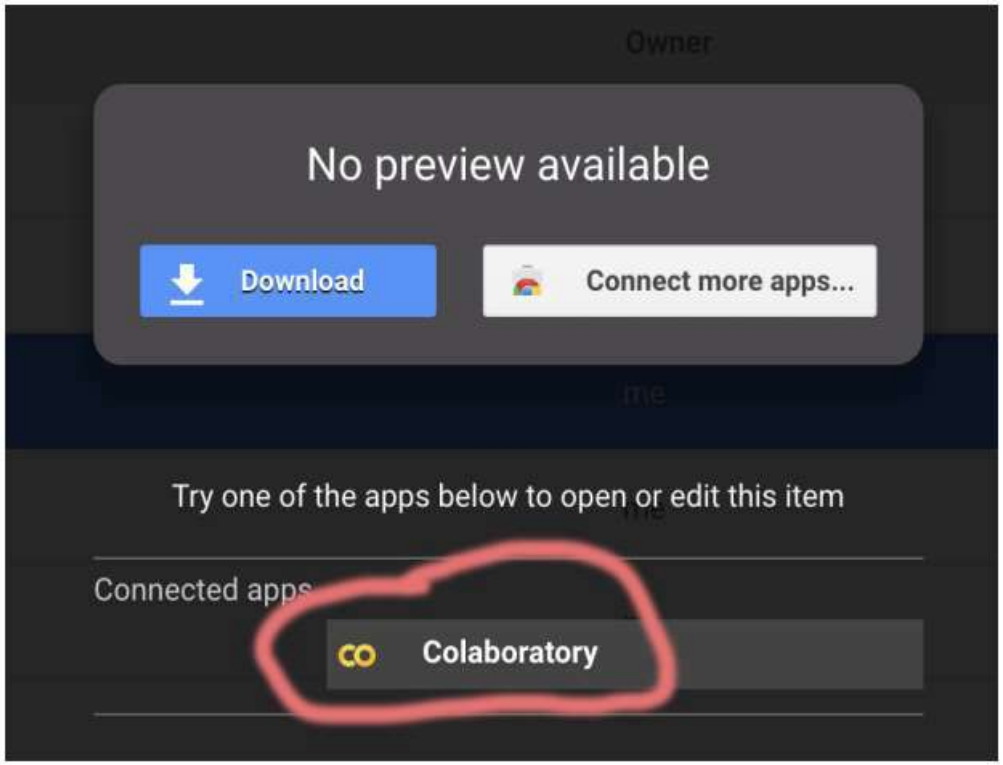
Enter your authorization code:
··········
Mounted at /content/drive/
```

Clone GitHub repos

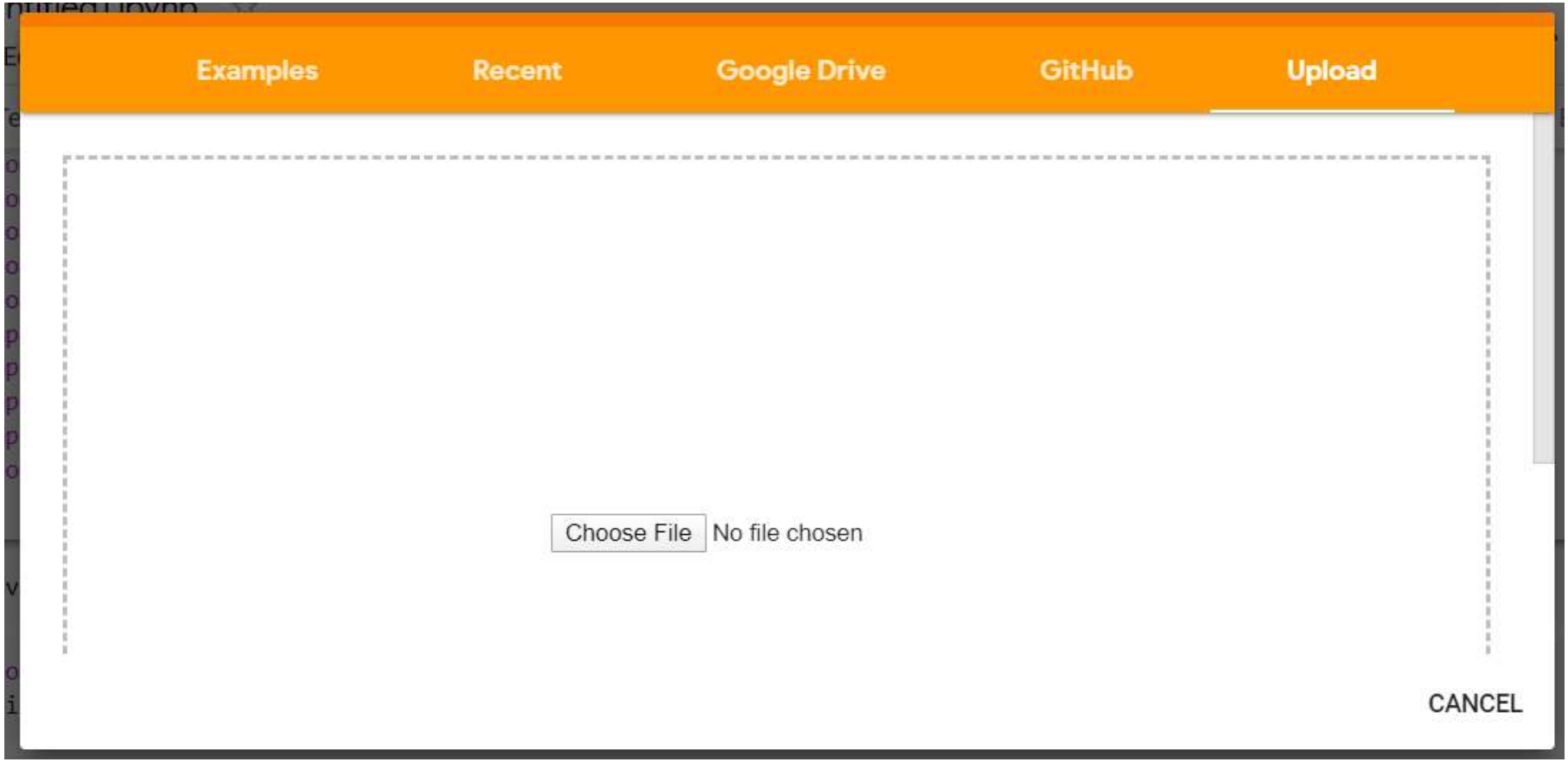


GitHub Repo


My Drive > Colab Notebooks > keras-mnist-tutorial
Name
Owner
.git
me
figure.png
me
MNIST in Keras.ipynb
me
newnotebook.png
me
README.md
me


No preview available
Download
Connect more apps...
Try one of the apps below to open or edit this item
Connected apps
Colaboratory

Upload Python Code



Thank you