TRAINING REPORT
At
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Submitted in partial fulfillment of the requirements for the award
of Bachelor of Engineering Degree in

Computer Science and Engineering

By

Kumar Aditya (38110276)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

AUGUST 2020

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119
www.sathyabamauniversity.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Kumar Aditya (38110276)** who carried out the project entitled "**Covid – 19 Data Processing**" under my supervision from April 2020 to July 2020.

Internal Guide

Dr. Albert Mayan J, M.E., Ph.D.

Head of Department

Dr. S.VIGNESHWARI, M.E., Ph.D.,
Dr. Lakshmanan L, M.E, Ph.D.,

Submitted for Viva voce Examination held on______________________

Internal Examiner **External Examiner**

DECLARATION

I, **Kumar Aditya (38110276)** hereby declare that the Project Report entitled "**Covid – 19 Data Processing"** done by me under the guidance of **Dr. Albert Mayan J, M.E., PhD** at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

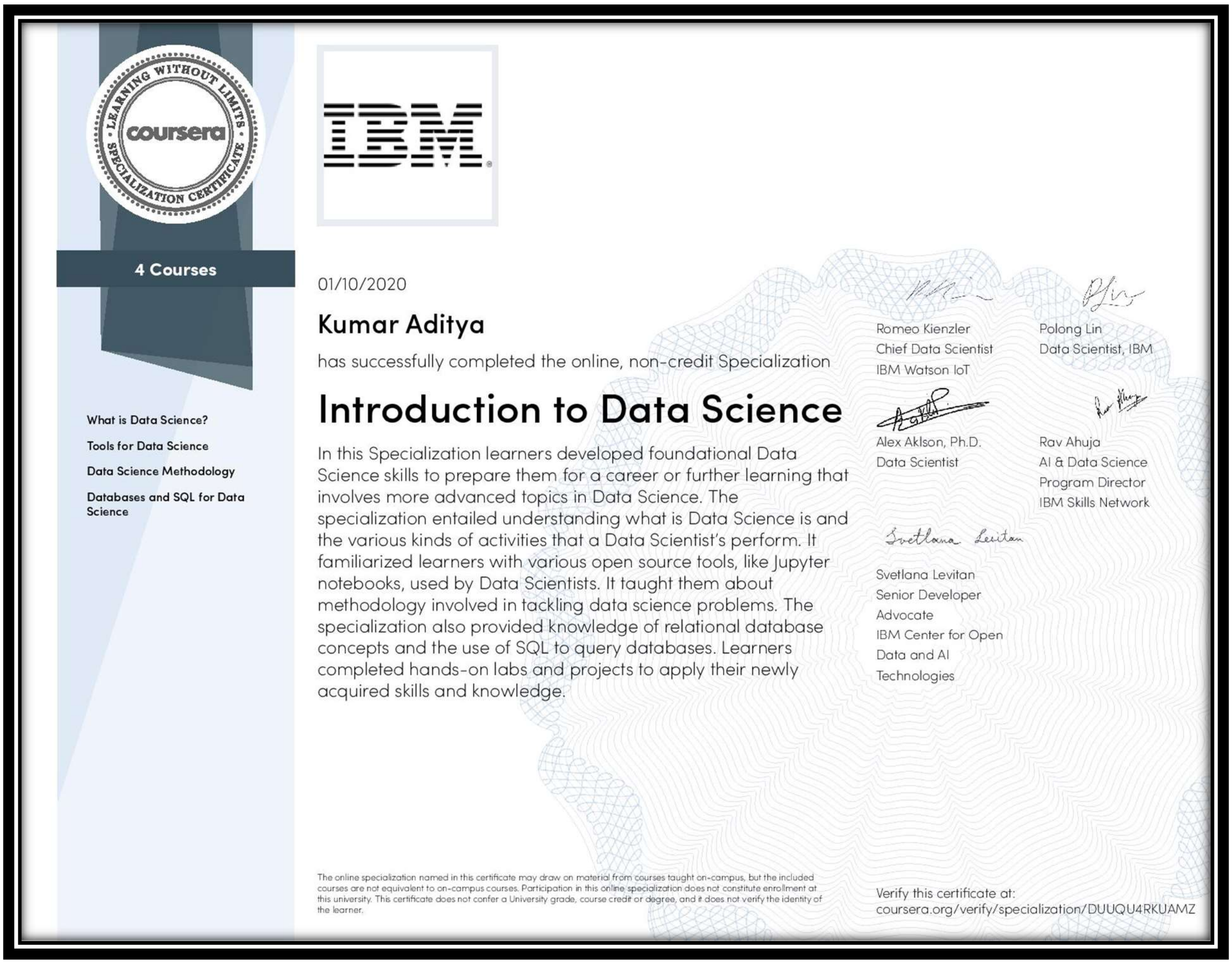
I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D.**, **Dean**, School of Computing , **Dr.S.Vigneshwari M.E., Ph.D., and Dr.L.Lakshmanan M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Albert Mayan, M.E., PhD** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



LEARNING WITHOUT LIMITS · coursera · SPECIALIZATION CERTIFICATE

4 Courses

What is Data Science?

Tools for Data Science

Data Science Methodology

Databases and SQL for Data Science

IBM

01/10/2020

Kumar Aditya

has successfully completed the online, non-credit Specialization

Introduction to Data Science

In this Specialization learners developed foundational Data Science skills to prepare them for a career or further learning that involves more advanced topics in Data Science. The specialization entailed understanding what is Data Science is and the various kinds of activities that a Data Scientist's perform. It familiarized learners with various open source tools, like Jupyter notebooks, used by Data Scientists. It taught them about methodology involved in tackling data science problems. The specialization also provided knowledge of relational database concepts and the use of SQL to query databases. Learners completed hands-on labs and projects to apply their newly acquired skills and knowledge.

Romeo Kienzler
Chief Data Scientist
IBM Watson IoT

Polong Lin
Data Scientist, IBM

Alex Aklson, Ph.D.
Data Scientist

Rav Ahuja
AI & Data Science
Program Director
IBM Skills Network

Svetlana Levitan

Svetlana Levitan
Senior Developer
Advocate
IBM Center for Open
Data and AI
Technologies

The online specialization named in this certificate may draw on material from courses taught on-campus, but the included courses are not equivalent to on-campus courses. Participation in this online specialization does not constitute enrollment at this university. This certificate does not confer a University grade, course credit or degree, and it does not verify the identity of the learner.

Verify this certificate at:
coursera.org/verify/specialization/DUUQU4RKUAMZ

ABSTRACT

Covid – 19 Data Processing is a Program which can load, merge, clean and aggregate the COVID-19 time series data. This program can give graphs and stats about any country in the world. It is designed by using Python Language, Matplotlib, Seaborn, Alter, Numpy and Pandas.

It takes the updated data from the web and show the current stats of Covid – 19 such as Confirmed, Recovered, Death and Active cases around each country for which the user requires. Further this program also compares data of newly confirmed cases between multiple countries.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

1.1 OUTLINE

This Covid – 19 Data Processing which is made using Python Language, Matplotlib, Seaborn, Alter, Numpy and Pandas can be used for creating graphs and stats for countries and also compare data of countries. Pandas is used to create DataFrame. Matplotlib, Seaborn and Alter is used to create graphs and stats table.

1.2 WHY USE PYTHON, NUMPY, PANDAS

Python is open source, interpreted, high level language and provides great approach for object-oriented programming. It is one of the best languages used by data scientist for various data science projects/application. It provides great functionality to deal with mathematics, statistics and scientific function. It provides great libraries to deals with data science application. It uses the elegant syntax; hence the programs are easier to read. The interactive mode of Python makes its simple to test codes. It allows developer to run the code anywhere, including Windows, Mac OS X, UNIX, and Linux. It is free software in a couple of categories. It does not cost anything to use or download Pythons or to add it to the application.

Numpy is Python library that provides mathematical function to handle large dimension array. It provides various method/function for Array, Metrics, and linear algebra. NumPy stands for Numerical Python. It provides lots of useful features for operations on n-arrays and matrices in Python. The library provides vectorization of mathematical operations on the NumPy array type, which enhance performance and speeds up the execution. It’s very easy to work with large multidimensional arrays and matrices using NumPy. Nearly every scientist working in Python draws on the power of NumPy. NumPy brings the computational power of languages like C and Fortran to Python, a language much easier to learn and use. With this power comes simplicity: a solution in NumPy is often clear and elegant. The core functionality of NumPy is its "ndarray", for *n*-dimensional array, data structure. These arrays are strided views on memory.

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. Pandas is one of the most popular Python library for data manipulation and analysis. Pandas provide useful functions to manipulate large amount of structured data. Pandas provide easiest method to perform analysis. It provides large data structures and manipulating numerical tables and time series data. Pandas is a perfect tool for data wrangling. Pandas is designed for quick and easy data manipulation, aggregation, and visualization. There two data structures in Pandas: Series handle and store data in one-dimensional data whereas DataFrame handle and store Two-dimensional data. It is a group by engine allowing split-apply-combine operations on data sets.

Data science is an interdisciplinary field focused on extracting knowledge from data sets, which are typically large (see big data). The field encompasses analysis, preparing data for analysis, and presenting findings to inform high-level decisions in an organization. As such, it incorporates skills from computer science, mathematics, statistics, information visualization, graphic design, complex systems, communication and business. Data science is a "concept to unify statistics, data analysis, machine learning, domain knowledge and their related methods" in order to "understand and analyze actual phenomena" with data.

1.3 WHY USE MATPLOTLIB, SEABORN, ALTAIR

Data science is a "concept to unify statistics, data analysis. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is useful Python library for Data Visualization. Descriptive analysis and visualizing data is very important for any organization. It provides various method to Visualize data in more effective way. It is a comprehensive library for creating static, animated, and interactive visualizations in Python. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. It consists of several plots like line, bar, scatter, histogram etc.

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. It is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures. It is a Specialized support for using categorical variables to show observations or aggregate statistics. It has automatic estimation and plotting of linear regression models for different kinds dependent variables. It has high-level abstractions for structuring multi-plot grids that let you easily build complex visualizations It has tools for choosing color palettes that faithfully reveal patterns in your data. Its specialized support for using categorical variables to show observations or aggregate statistics.

Altair is a declarative statistical visualization library for Python. With Altair, you can spend more time understanding your data and its meaning. Altair's API is simple, friendly and consistent and built on top of the powerful Vega-Lite visualization grammar. This elegant simplicity produces beautiful and effective visualizations with a minimal amount of code. Altair offers a powerful and concise visualization grammar that enables you to build a wide range of statistical visualizations quickly. It can be used to change several other attributes of the tables such as the name of the table and name of the columns. It can changes the structure or schema of an already created database table by adding or removing one or more columns.

1.4 LITERATURE REVIEW

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's *Benevolent Dictator For Life*, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project.

It is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta programming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML and Data Science.

In 1962, John Tukey described a field he called “data analysis,” which resembles modern data science. Later, attendees at a 1992 statistics symposium at the University of Montpellier II acknowledged the emergence of a new discipline focused on data of various origins and forms, combining established concepts and principles of statistics and data analysis with computing.

The term “data science” has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux. In 1997, C.F. Jeff Wu suggested that statistics should be renamed data science. He reasoned that a new name would help statistics shed inaccurate stereotypes, such as being synonymous with accounting, or limited to describing data. In 1998, Chikio Hayashi argued for data science as a new, interdisciplinary concept, with three aspects: data design, collection, and analysis.

During the 1990s, popular terms for the process of finding patterns in datasets (which were increasingly large) included “knowledge discovery” and “data mining. The modern conception of data science as an independent discipline is sometimes attributed to William S. Cleveland. In a 2001 paper, he advocated an expansion of statistics beyond theory into technical areas; because this would significantly change the field, it warranted a new name. Data science is a growing field. A career as a data scientist is ranked at the third best job in America for 2020 by Glassdoor, and was ranked the number one best job from 2016-2019.

As such, it incorporates skills from computer science, mathematics, statistics, information visualization, graphic design, complex systems, communication and business. Statistician Nathan Yau, drawing on Ben Fry, also links data science to human-computer interaction: users should be able to intuitively control and explore data. In 2015, the American Statistical Association identified database management, statistics and machine learning, and distributed and parallel systems as the three emerging foundational professional communities.

1.5 PROBLEM STATEMENT

To process data of Covid – 19 and plot graphs and stats using Data Visualization Tools. Such that one can get data of different countries of world and plot its graphs and stats like Confirmed, Recovered, Deaths etc. Also compare data of different countries.

1.6 OBJECTIVES

A DataFrame to create and process Covid – 19 data and give different graphs and stats of different countries with respect to Confirmed, Recovered, Death.

CHAPTER 2

AIM & SCOPE OF DATA SCIENCE

2.1 REQUIREMENTS

2.1.1 HARDWARE REQUIREMENTS

- Inter core i5 7th Gen / Ryzen 5 2500u
- 8GB of DDR4 RAM
- 128 GB SSD / HDD
- 2GB Nvidia GeForce RTX 1050 Ti

2.1.2 SOFTWARE REQUIREMENTS:

- Python installed on your system
- Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

2.2 ROLE OF JUPYTER

Jupyter is a free, open-source, interactive web tool known as a computational notebook, which researchers can use to combine software code, computational output, explanatory text and multimedia resources in a single document. Computational notebooks have been around for decades, but Jupyter in particular has exploded in popularity over the past couple of years. This rapid uptake has been aided by an enthusiastic community of user–developers and a redesigned architecture that allows the notebook to speak dozens of programming languages — a fact reflected in its name, which was inspired, according to co-founder Fernando Pérez, by the programming languages Julia (Ju), Python (Py) and R. Computational notebooks are essentially laboratory notebooks for scientific computing.

For data scientists, that format can drive exploration. Notebooks, Barba says, are a form of interactive computing, an environment in which users execute code, see what happens, modify and repeat in a kind of iterative conversation between researcher and data. They aren’t the only forum for such conversations — IPython, the interactive Python interpreter on which Jupyter’s predecessor, IPython Notebook, was built, is another. But notebooks allow users to document those conversations, building “more powerful connections between topics, theories, data and results”.

The Jupyter notebook has two components. Users input programming code or text in rectangular cells in a front-end web page. The browser then passes that code to a back-end ‘kernel’, which runs the code and returns the results.

Jupyter’s newest variant is JupyterLab, which launched as a beta in January 2018 and is available (like the Jupyter notebook) either as a stand-alone package or as part of the free Anaconda scientific-computing environment.

2.2.1 LANGUAGES USED FOR PROJECT

Machine Learning , data science and artificial Intelligence-based projects are obviously what the future holds . We want better personalization, smarter recommendations, and improved search functionality. Our apps can see, hear, and respond-that’s what artificial intelligence (AI) has brought , enhancing the user experience and creating value across many industries.

FEATURES OF PYTHON:

Following are some of the important features that make Python the first choice of Data Science Engineers.

Simple and consistent:

Python offers concise and readable code. While complex algorithms and versatile workflows stand behind data science, machine learning and AI, Python’s simplicity allows developers to write reliable systems. Python code is understandable by humans, which makes it easier to build models for machine learning.

Extensive selection of libraries and frameworks:

Implementing Data Science and ML algorithms can be tricky and requires a lot of time. It's vital to have a well-structured and well tested environment to enable developers to come up with the best coding solutions.

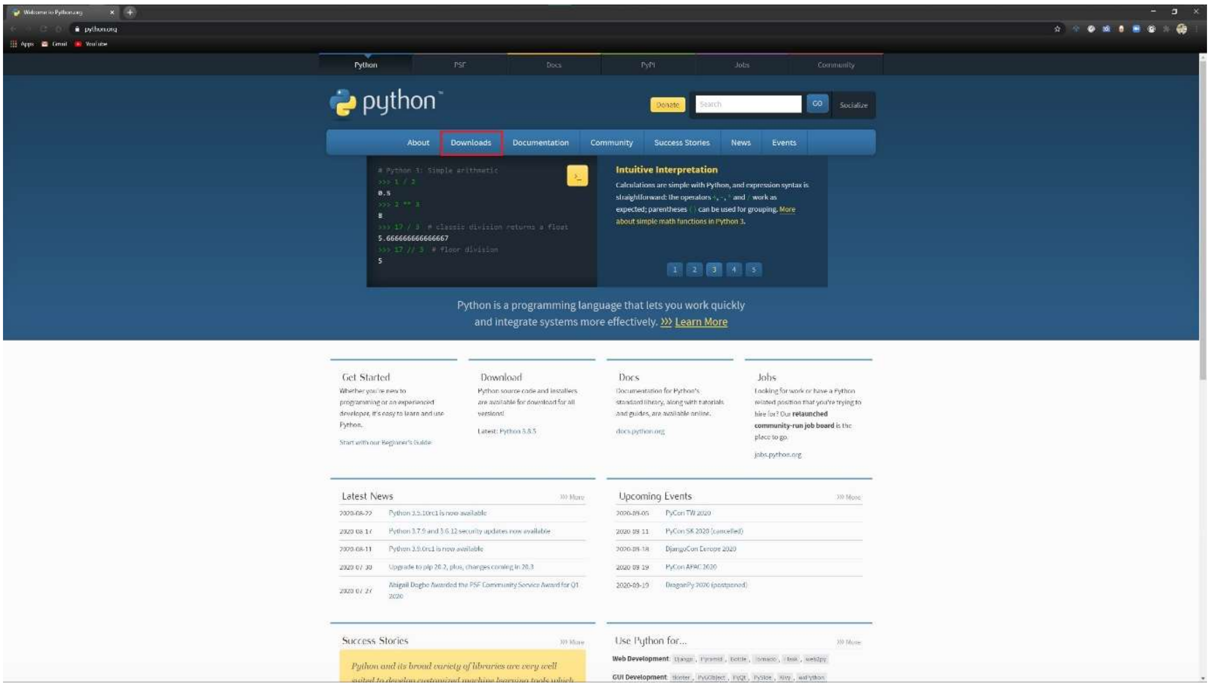
- Numpy and Pandas for general purpose data analysis.
- Matplotlib for creating static, animated, and interactive visualizations.
- Seaborn for high-level interface for drawing attractive and informative statistical graphics.
- Altair for understanding data and its meaning.

Platform independence:

Python software can be easily distributed and used on that operating system without a Python interpreter. Great Community and Popularity: In the Developer survey 2018 by Stack Overflow, Python was among the most popular programming language, which ultimately mean that you can find easily a development company.

2.3 HOW TO INSTALL PYTHON

From Website:



```
# Python 3: Simple arithmetic
>>> 1 / 2
0.5
>>> 2 ** 3
8
>>> 17 / 3  # classic division returns a float
5.666666666666667
>>> 17 // 3  # floor division
5
```

Intuitive Interpretation

Calculations are simple with Python, and expression syntax is straightforward: the operators +, -, * and / work as expected; parentheses ( ) can be used for grouping. More about simple math functions in Python 3.

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Get Started

Whether you're new to programming or an experienced developer, it's easy to learn and use Python.

Start with our Beginner's Guide

Download

Python source code and installers are available for download for all versions!

Latest: Python 3.8.5

Docs

Documentation for Python's standard library, along with tutorials and guides, are available online.

docs.python.org

Jobs

Looking for work or have a Python related position that you're trying to hire for? Our **relaunched community-run job board** is the place to go.

jobs.python.org

Latest News >>> More

2020-08-22	Python 3.5.10rc1 is now available
2020-08-17	Python 3.7.9 and 3.6.12 security updates now available
2020-08-11	Python 3.9.0rc1 is now available
2020-07-30	Upgrade to pip 20.2, plus, changes coming in 20.3
2020-07-27	Abigail Dogbe Awarded the PSF Community Service Award for Q1 2020

Upcoming Events >>> More

2020-09-05	PyCon TW 2020
2020-09-11	PyCon SK 2020 (cancelled)
2020-09-18	DjangoCon Europe 2020
2020-09-19	PyCon APAC 2020
2020-09-19	DragonPy 2020 (postponed)

Success Stories >>> More

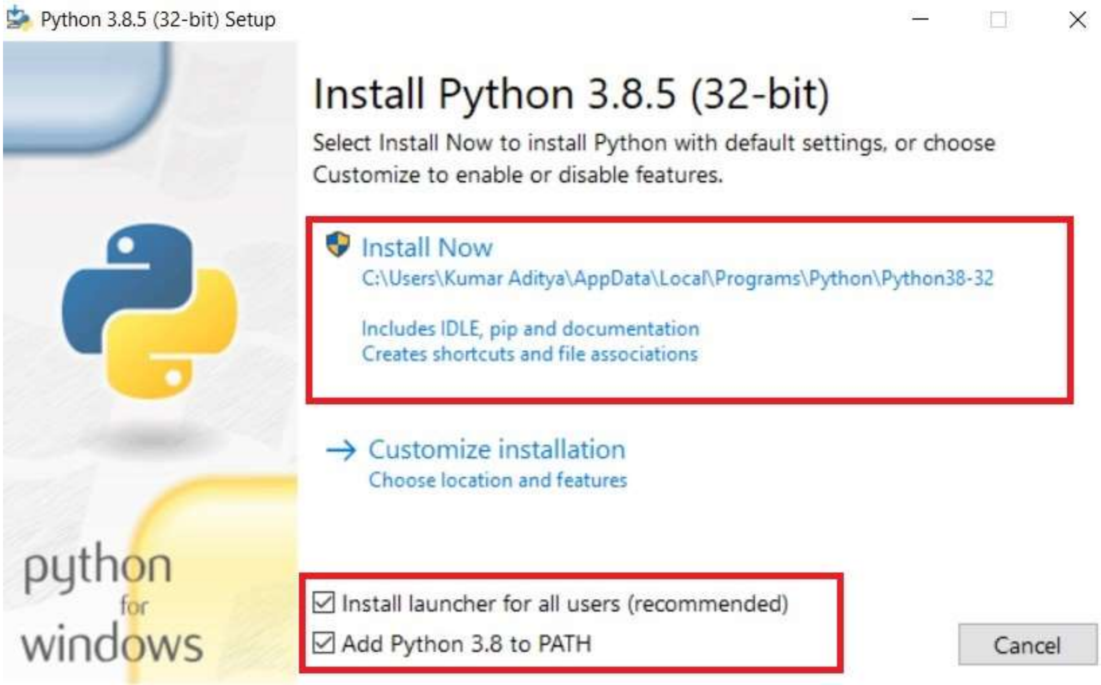
Python and its broad variety of libraries are very well suited to develop customized machine learning tools which

Use Python for... >>> More

Web Development: Django, Pyramid, Bottle, Tornado, Flask, web2py

GUI Development: tkInter, PyGObject, PyQt, PySide, Kivy, wxPython

You can download Python from their website directly to your system for development purposes.



Python 3.8.5 (32-bit) Setup

Install Python 3.8.5 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\Kumar Aditya\AppData\Local\Programs\Python\Python38-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

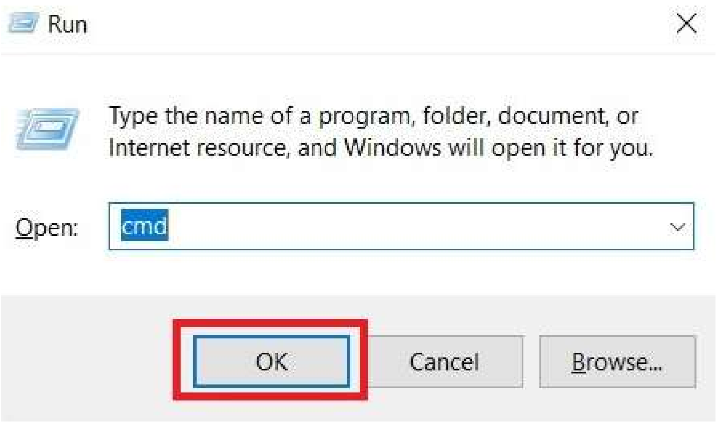
→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.8 to PATH

Do not forget to “Add Python to PATH”

❖ After installation is complete you will be able to see Python in Command Prompt.

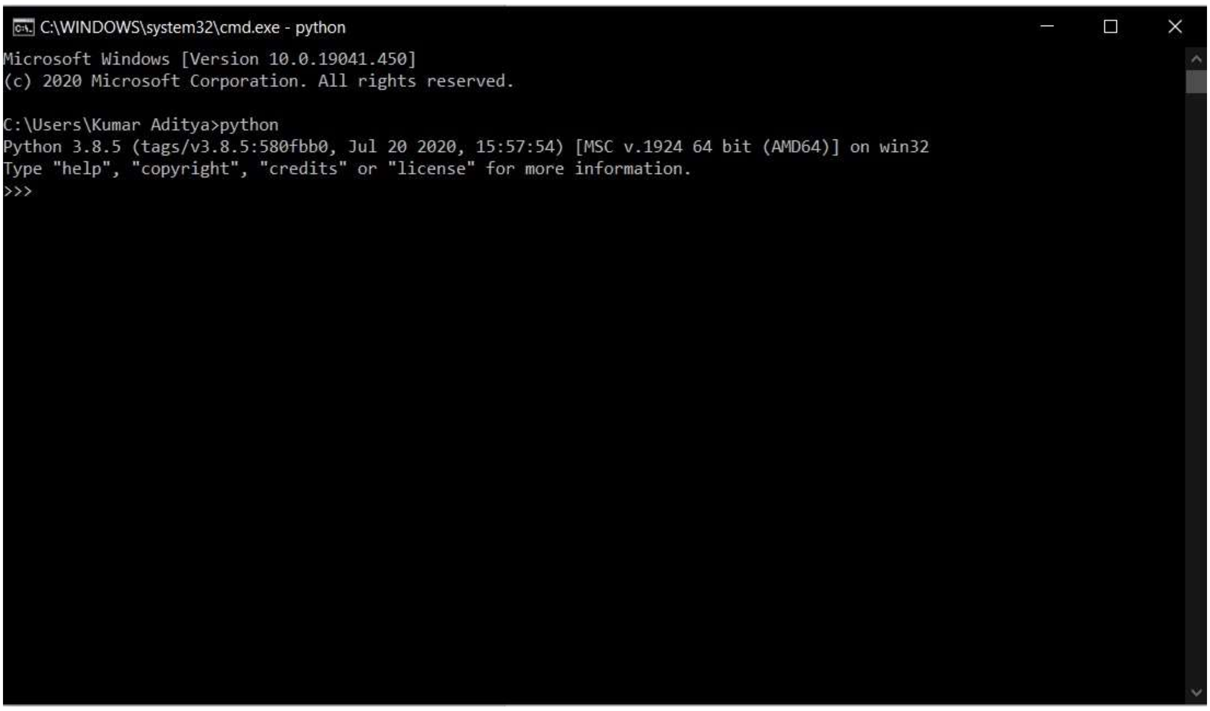
Run -> cmd



Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: cmd

❖ You can see information about your version and switch to python by typing “python” in Command Prompt.



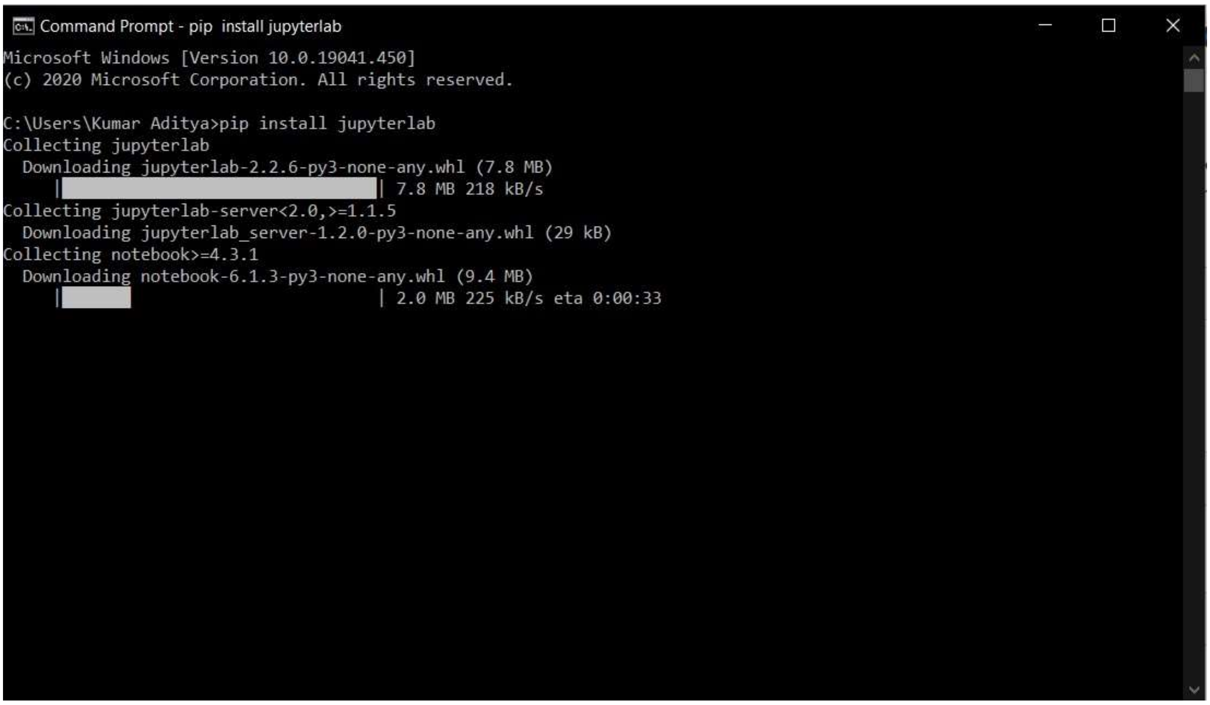
```
Microsoft Windows [Version 10.0.19041.450]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Kumar Aditya>python
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

2.4 HOW TO INSTALL JUPYTER NOTEBOOK

Using pip:

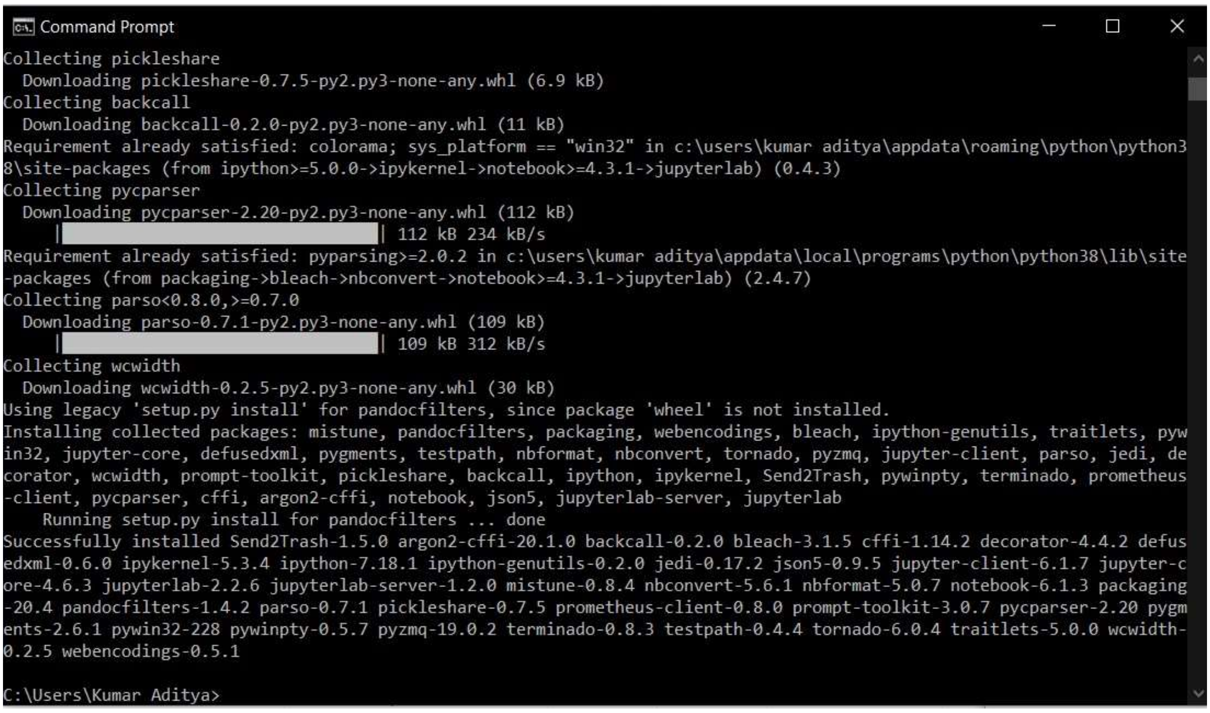
If you use pip, you can install it with: (In Command Prompt)

```
pip install jupyterlab
```



```
Microsoft Windows [Version 10.0.19041.450]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Kumar Aditya>pip install jupyterlab
Collecting jupyterlab
  Downloading jupyterlab-2.2.6-py3-none-any.whl (7.8 MB)
     |████████████████████████████████| 7.8 MB 218 kB/s
Collecting jupyterlab-server<2.0,>=1.1.5
  Downloading jupyterlab_server-1.2.0-py3-none-any.whl (29 kB)
Collecting notebook>=4.3.1
  Downloading notebook-6.1.3-py3-none-any.whl (9.4 MB)
     |███████                         | 2.0 MB 225 kB/s eta 0:00:33
```

❖ After the completion of setup:

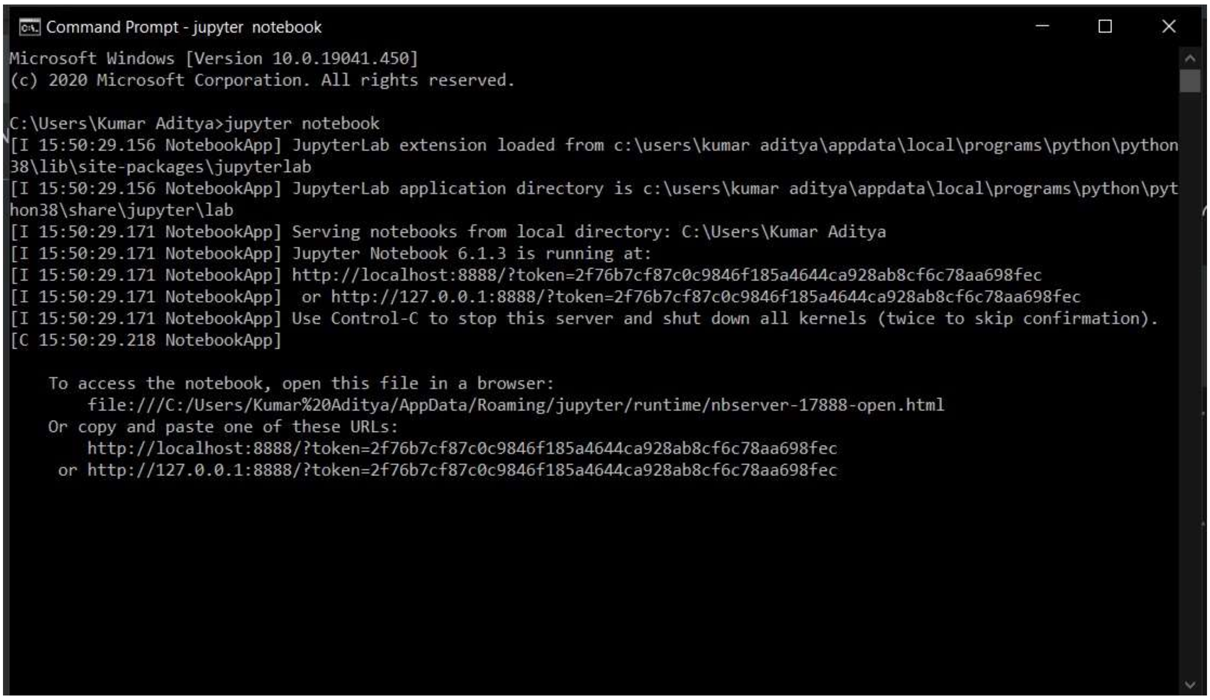


```
Collecting pickleshare
  Downloading pickleshare-0.7.5-py2.py3-none-any.whl (6.9 kB)
Collecting backcall
  Downloading backcall-0.2.0-py2.py3-none-any.whl (11 kB)
Requirement already satisfied: colorama; sys_platform == "win32" in c:\users\kumar aditya\appdata\roaming\python\python3
8\site-packages (from ipython>=5.0.0->ipykernel->notebook>=4.3.1->jupyterlab) (0.4.3)
Collecting pycparser
  Downloading pycparser-2.20-py2.py3-none-any.whl (112 kB)
     |████████████████████████████████| 112 kB 234 kB/s
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\kumar aditya\appdata\local\programs\python\python38\lib\site
-packages (from packaging->bleach->nbconvert->notebook>=4.3.1->jupyterlab) (2.4.7)
Collecting parso<0.8.0,>=0.7.0
  Downloading parso-0.7.1-py2.py3-none-any.whl (109 kB)
     |████████████████████████████████| 109 kB 312 kB/s
Collecting wcwidth
  Downloading wcwidth-0.2.5-py2.py3-none-any.whl (30 kB)
Using legacy 'setup.py install' for pandocfilters, since package 'wheel' is not installed.
Installing collected packages: mistune, pandocfilters, packaging, webencodings, bleach, ipython-genutils, traitlets, pyw
in32, jupyter-core, defusedxml, pygments, testpath, nbformat, nbconvert, tornado, pyzmq, jupyter-client, parso, jedi, de
corator, wcwidth, prompt-toolkit, pickleshare, backcall, ipython, ipykernel, Send2Trash, pywinpty, terminado, prometheus
-client, pycparser, cffi, argon2-cffi, notebook, json5, jupyterlab-server, jupyterlab
    Running setup.py install for pandocfilters ... done
Successfully installed Send2Trash-1.5.0 argon2-cffi-20.1.0 backcall-0.2.0 bleach-3.1.5 cffi-1.14.2 decorator-4.4.2 defus
edxml-0.6.0 ipykernel-5.3.4 ipython-7.18.1 ipython-genutils-0.2.0 jedi-0.17.2 json5-0.9.5 jupyter-client-6.1.7 jupyter-c
ore-4.6.3 jupyterlab-2.2.6 jupyterlab-server-1.2.0 mistune-0.8.4 nbconvert-5.6.1 nbformat-5.0.7 notebook-6.1.3 packaging
-20.4 pandocfilters-1.4.2 parso-0.7.1 pickleshare-0.7.5 prometheus-client-0.8.0 prompt-toolkit-3.0.7 pycparser-2.20 pygm
ents-2.6.1 pywin32-228 pywinpty-0.5.7 pyzmq-19.0.2 terminado-0.8.3 testpath-0.4.4 tornado-6.0.4 traitlets-5.0.0 wcwidth-
0.2.5 webencodings-0.5.1

C:\Users\Kumar Aditya>
```

2.4.1 Opening Jupyter Notebook

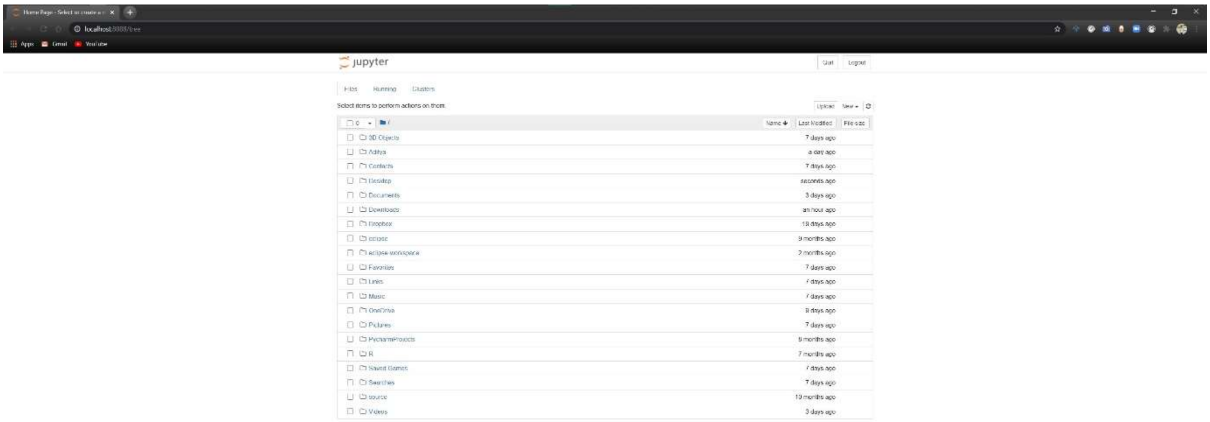
You can type following command in cmd to open notebook.

```
jupyter notebook
```



```
Microsoft Windows [Version 10.0.19041.450]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Kumar Aditya>jupyter notebook
[I 15:50:29.156 NotebookApp] JupyterLab extension loaded from c:\users\kumar aditya\appdata\local\programs\python\python
38\lib\site-packages\jupyterlab
[I 15:50:29.156 NotebookApp] JupyterLab application directory is c:\users\kumar aditya\appdata\local\programs\python\pyt
hon38\share\jupyter\lab
[I 15:50:29.171 NotebookApp] Serving notebooks from local directory: C:\Users\Kumar Aditya
[I 15:50:29.171 NotebookApp] Jupyter Notebook 6.1.3 is running at:
[I 15:50:29.171 NotebookApp] http://localhost:8888/?token=2f76b7cf87c0c9846f185a4644ca928ab8cf6c78aa698fec
[I 15:50:29.171 NotebookApp]  or http://127.0.0.1:8888/?token=2f76b7cf87c0c9846f185a4644ca928ab8cf6c78aa698fec
[I 15:50:29.171 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 15:50:29.218 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/Kumar%20Aditya/AppData/Roaming/jupyter/runtime/nbserver-17888-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=2f76b7cf87c0c9846f185a4644ca928ab8cf6c78aa698fec
     or http://127.0.0.1:8888/?token=2f76b7cf87c0c9846f185a4644ca928ab8cf6c78aa698fec
```

After this your notebook will be opened in your default browser:

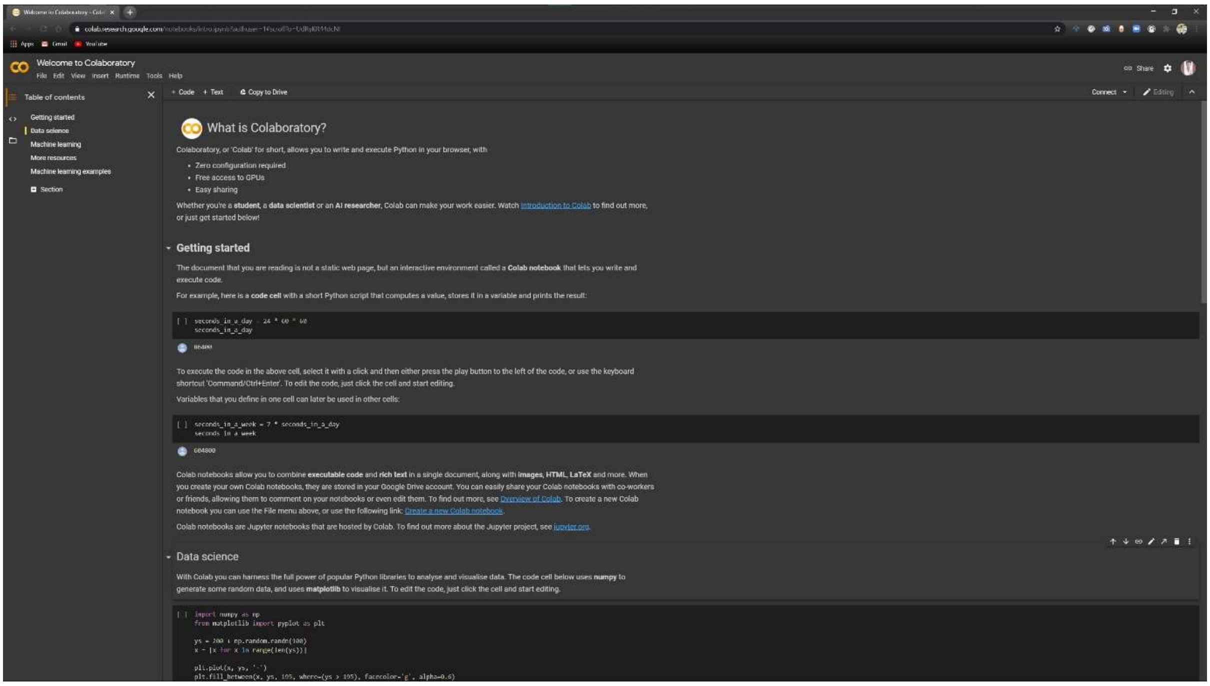


Select items to perform actions on them.

□ 0 ▾ /	Name ↓	Last Modified	File size
□ 3D Objects		7 days ago	
□ Aditya		a day ago	
□ Contacts		7 days ago	
□ Desktop		seconds ago	
□ Documents		3 days ago	
□ Downloads		an hour ago	
□ Dropbox		19 days ago	
□ eclipse		9 months ago	
□ eclipse-workspace		2 months ago	
□ Favorites		7 days ago	
□ Links		7 days ago	
□ Music		7 days ago	
□ OneDrive		8 days ago	
□ Pictures		7 days ago	
□ PycharmProjects		8 months ago	
□ R		7 months ago	
□ Saved Games		7 days ago	
□ Searches		7 days ago	
□ source		10 months ago	
□ Videos		3 days ago	

2.5 Google Colaboratory

With Colab you can harness the full power of popular Python libraries to analyse and visualize data. To edit the code, just click the cell and start editing. It is one of the best alternatives for Jupyter Notebook.



What is Colaboratory?

Colaboratory, or 'Colab' for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to find out more, or just get started below!

Getting started

The document that you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable and prints the result:

```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

```
86400
```

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut 'Command/Ctrl+Enter'. To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

```
seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

```
604800
```

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them. To find out more, see Overview of Colab. To create a new Colab notebook you can use the File menu above, or use the following link: Create a new Colab notebook.

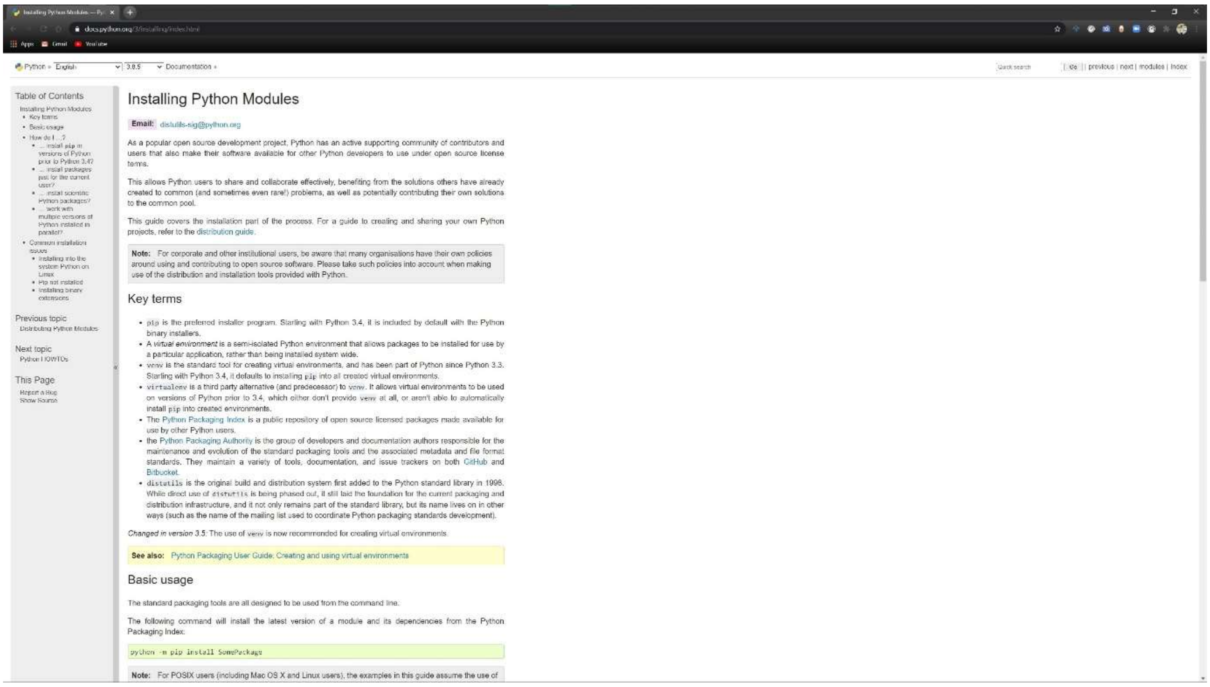
Colab notebooks are Jupyter notebooks that are hosted by Colab. To find out more about the Jupyter project, see jupyter.org.

Data science

With Colab you can harness the full power of popular Python libraries to analyse and visualise data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualise it. To edit the code, just click the cell and start editing.

```
import numpy as np
from matplotlib import pyplot as plt

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)
```

2.6 INSTALLING PYTHON MODULES:


Python » English 3.8.5 Documentation »
Quick search
previous | next | modules | index
Table of Contents
Installing Python Modules
Key terms
Basic usage
How do I …?
… install pip in versions of Python prior to Python 3.4?
… install packages just for the current user?
… install scientific Python packages?
… work with multiple versions of Python installed in parallel?
Common installation issues
Installing into the system Python on Linux
Pip not installed
Installing binary extensions
Previous topic
Distributing Python Modules
Next topic
Python HOWTOs
This Page
Report a Bug
Show Source

Installing Python Modules

Email: distutils-sig@python.org

As a popular open source development project, Python has an active supporting community of contributors and users that also make their software available for other Python developers to use under open source license terms.

This allows Python users to share and collaborate effectively, benefiting from the solutions others have already created to common (and sometimes even rare!) problems, as well as potentially contributing their own solutions to the common pool.

This guide covers the installation part of the process. For a guide to creating and sharing your own Python projects, refer to the distribution guide.

Note: For corporate and other institutional users, be aware that many organisations have their own policies around using and contributing to open source software. Please take such policies into account when making use of the distribution and installation tools provided with Python.

Key terms

- `pip` is the preferred installer program. Starting with Python 3.4, it is included by default with the Python binary installers.
- A *virtual environment* is a semi-isolated Python environment that allows packages to be installed for use by a particular application, rather than being installed system wide.
- `venv` is the standard tool for creating virtual environments, and has been part of Python since Python 3.3. Starting with Python 3.4, it defaults to installing `pip` into all created virtual environments.
- `virtualenv` is a third party alternative (and predecessor) to `venv`. It allows virtual environments to be used on versions of Python prior to 3.4, which either don't provide `venv` at all, or aren't able to automatically install `pip` into created environments.
- The Python Packaging Index is a public repository of open source licensed packages made available for use by other Python users.
- the Python Packaging Authority is the group of developers and documentation authors responsible for the maintenance and evolution of the standard packaging tools and the associated metadata and file format standards. They maintain a variety of tools, documentation, and issue trackers on both GitHub and Bitbucket.
- `distutils` is the original build and distribution system first added to the Python standard library in 1998. While direct use of `distutils` is being phased out, it still laid the foundation for the current packaging and distribution infrastructure, and it not only remains part of the standard library, but its name lives on in other ways (such as the name of the mailing list used to coordinate Python packaging standards development).

Changed in version 3.5: The use of `venv` is now recommended for creating virtual environments.

See also: Python Packaging User Guide: Creating and using virtual environments

Basic usage

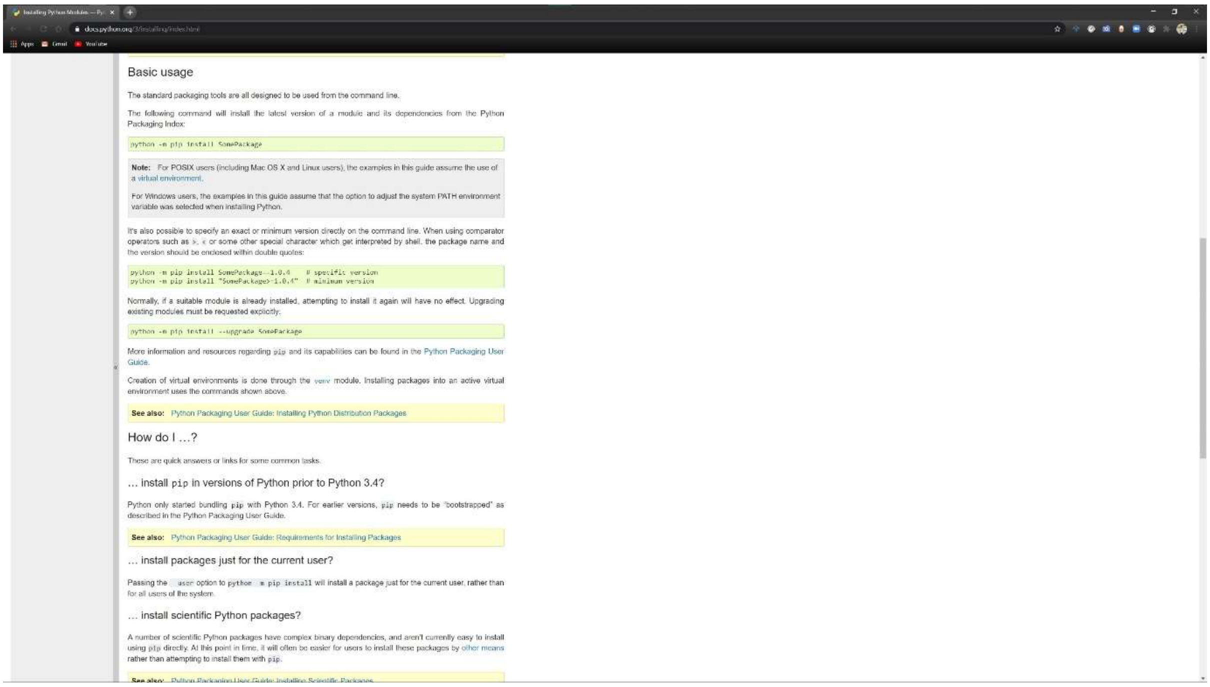
The standard packaging tools are all designed to be used from the command line.

The following command will install the latest version of a module and its dependencies from the Python Packaging Index:

```
python -m pip install SomePackage
```

Note: For POSIX users (including Mac OS X and Linux users), the examples in this guide assume the use of

```
python -m pip install SomePackage
```



Installing Python Modules — Py
docs.python.org/3/installing/index.html
Apps
Gmail
YouTube

Basic usage

The standard packaging tools are all designed to be used from the command line.

The following command will install the latest version of a module and its dependencies from the Python Packaging Index:

```
python -m pip install SomePackage
```

Note: For POSIX users (including Mac OS X and Linux users), the examples in this guide assume the use of a virtual environment.

For Windows users, the examples in this guide assume that the option to adjust the system PATH environment variable was selected when installing Python.

It's also possible to specify an exact or minimum version directly on the command line. When using comparator operators such as `>`, `<` or some other special character which get interpreted by shell, the package name and the version should be enclosed within double quotes:

```
python -m pip install SomePackage==1.0.4    # specific version
python -m pip install "SomePackage>=1.0.4"  # minimum version
```

Normally, if a suitable module is already installed, attempting to install it again will have no effect. Upgrading existing modules must be requested explicitly:

```
python -m pip install --upgrade SomePackage
```

More information and resources regarding `pip` and its capabilities can be found in the Python Packaging User Guide.

Creation of virtual environments is done through the `venv` module. Installing packages into an active virtual environment uses the commands shown above.

See also: Python Packaging User Guide: Installing Python Distribution Packages

How do I …?

These are quick answers or links for some common tasks.

… install `pip` in versions of Python prior to Python 3.4?

Python only started bundling `pip` with Python 3.4. For earlier versions, `pip` needs to be "bootstrapped" as described in the Python Packaging User Guide.

See also: Python Packaging User Guide: Requirements for Installing Packages

… install packages just for the current user?

Passing the `--user` option to `python -m pip install` will install a package just for the current user, rather than for all users of the system.

… install scientific Python packages?

A number of scientific Python packages have complex binary dependencies, and aren't currently easy to install using `pip` directly. At this point in time, it will often be easier for users to install these packages by other means rather than attempting to install them with `pip`.

See also: Python Packaging User Guide: Installing Scientific Packages

CHAPTER 3

METHODS AND MATERIAL USED

3.1 MATERIAL USED

- Jupyter Notebook/ Google Colaboratory

3.2 METHODS

3.2.1 DESIGNING THE PROJECT

Confirmed, deaths and recovered are kept in different CSV files. That makes difficult for plotting them in the same data visualization.

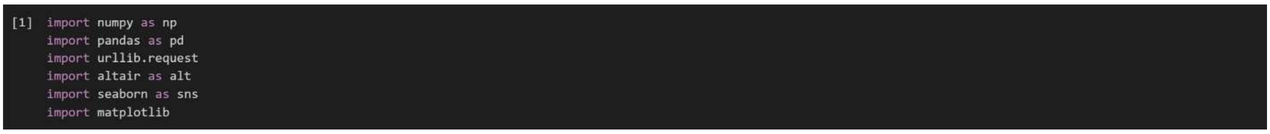
This COVID-19 Data processing runs the following steps:

1. Download raw CSV dataset from JHU CSSE public Github page.
2. Load raw CSV dataset and extract the common date list.
3. Merges the raw confirmed, deaths, and recovered CSV data into one DataFrame.
4. Performs data cleanings due to missing values, wrong datatypes and cases from cruise ships.
5. Data Aggregation: Add an active case column *Active*, which is calculated by active_case = confirmed — deaths — recovered. Aggregate data into Country/Region wise and group them by Date and Country/Region. After that, add day wise New cases, New deaths and New recovered by deducting the corresponding cumulative data on the previous day.

There are 3 tasks we would like to do

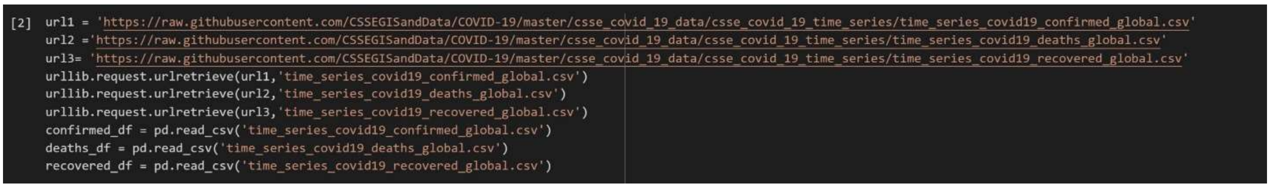
1. Converting Date from string to datetime
2. Replacing missing value NaN
3. Coronavirus cases reported from 3 cruise ships should be treated differently.

Importing Libraries:



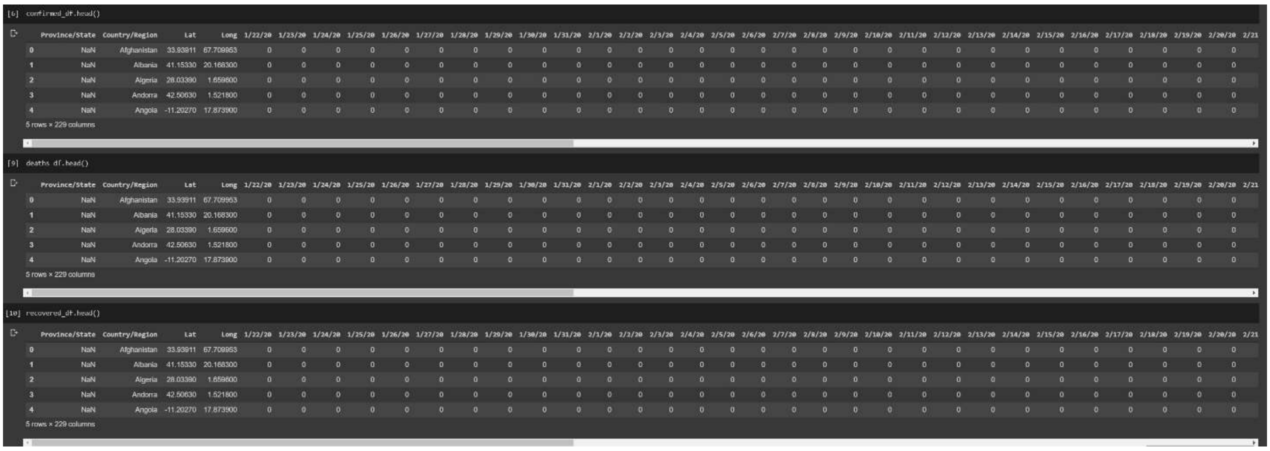
```
[1]  import numpy as np
     import pandas as pd
     import urllib.request
     import altair as alt
     import seaborn as sns
     import matplotlib
```

Getting data from URL:



```
[2]  url1 = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv'
     url2 ='https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_global.csv'
     url3= 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_recovered_global.csv'
     urllib.request.urlretrieve(url1,'time_series_covid19_confirmed_global.csv')
     urllib.request.urlretrieve(url2,'time_series_covid19_deaths_global.csv')
     urllib.request.urlretrieve(url3,'time_series_covid19_recovered_global.csv')
     confirmed_df = pd.read_csv('time_series_covid19_confirmed_global.csv')
     deaths_df = pd.read_csv('time_series_covid19_deaths_global.csv')
     recovered_df = pd.read_csv('time_series_covid19_recovered_global.csv')
```

Initial Data:



```
[8]  confirmed_df.head()
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	2/1/20	2/2/20	2/3/20	2/4/20	2/5/20	2/6/20	2/7/20	2/8/20	2/9/20	2/10/20	2/11/20	2/12/20	2/13/20	2/14/20	2/15/20	2/16/20	2/17/20	2/18/20	2/19/20	2/20/20	2/21
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	NaN	Albania	41.15330	20.168300	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	NaN	Andorra	42.50630	1.521800	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	NaN	Angola	-11.20270	17.873900	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

5 rows × 229 columns

```
[9]  deaths_df.head()
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	2/1/20	2/2/20	2/3/20	2/4/20	2/5/20	2/6/20	2/7/20	2/8/20	2/9/20	2/10/20	2/11/20	2/12/20	2/13/20	2/14/20	2/15/20	2/16/20	2/17/20	2/18/20	2/19/20	2/20/20	2/21
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	NaN	Albania	41.15330	20.168300	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	NaN	Andorra	42.50630	1.521800	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	NaN	Angola	-11.20270	17.873900	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

5 rows × 229 columns

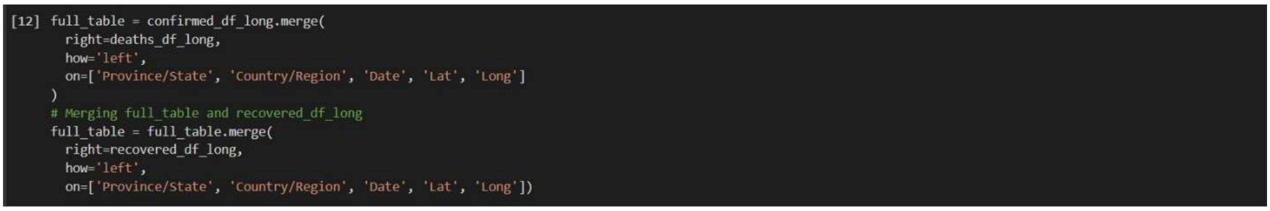
```
[10] recovered_df.head()
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	2/1/20	2/2/20	2/3/20	2/4/20	2/5/20	2/6/20	2/7/20	2/8/20	2/9/20	2/10/20	2/11/20	2/12/20	2/13/20	2/14/20	2/15/20	2/16/20	2/17/20	2/18/20	2/19/20	2/20/20	2/21
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	NaN	Albania	41.15330	20.168300	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	NaN	Andorra	42.50630	1.521800	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	NaN	Angola	-11.20270	17.873900	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

5 rows × 229 columns

Confirmed, death and recovered are in separate DataFrame. That makes it difficult to visualize data together.

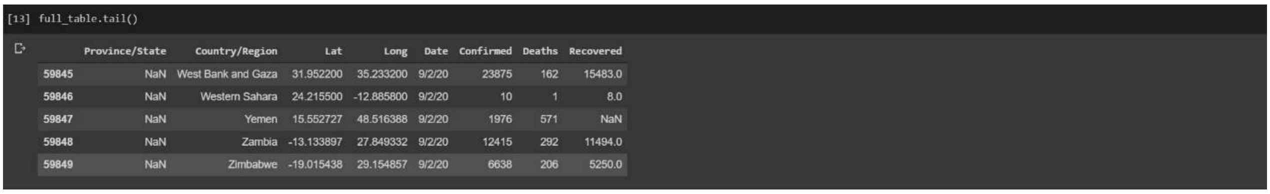
Merging Data:



```
[12] full_table = confirmed_df_long.merge(
       right=deaths_df_long,
       how='left',
       on=['Province/State', 'Country/Region', 'Date', 'Lat', 'Long']
     )
     # Merging full_table and recovered_df_long
     full_table = full_table.merge(
       right=recovered_df_long,
       how='left',
       on=['Province/State', 'Country/Region', 'Date', 'Lat', 'Long'])
```

Merged Data:

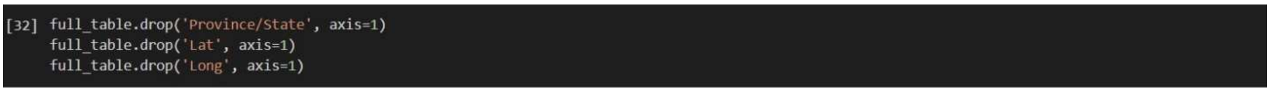
- Confirmed, Deaths, Recovered and Active are cumulative data.
- New cases, New deaths and New Recovered are day wise data.
- This DataFrame is ordered by Date and Country/Region.



```
[13] full_table.tail()
```

```
       Province/State   Country/Region        Lat        Long    Date  Confirmed  Deaths  Recovered
59845             NaN  West Bank and Gaza  31.952200   35.233200  9/2/20      23875     162    15483.0
59846             NaN    Western Sahara  24.215500  -12.885800  9/2/20         10       1        8.0
59847             NaN             Yemen  15.552727   48.516388  9/2/20       1976     571        NaN
59848             NaN            Zambia -13.133897   27.849332  9/2/20      12415     292    11494.0
59849             NaN          Zimbabwe -19.015438   29.154857  9/2/20       6638     206     5250.0
```

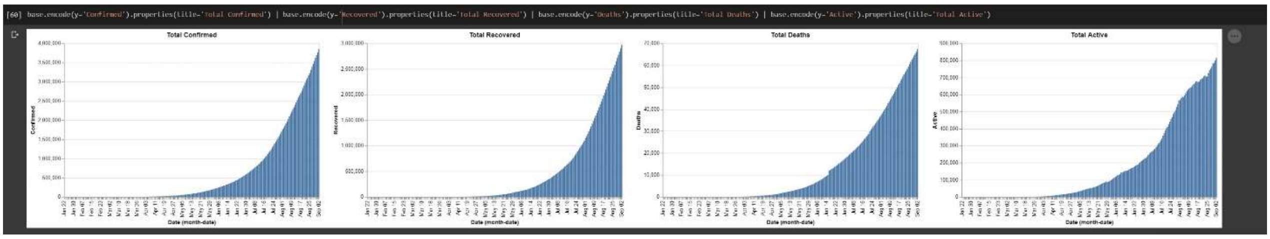
Drop columns which are not required:



```
[32] full_table.drop('Province/State', axis=1)
     full_table.drop('Lat', axis=1)
     full_table.drop('Long', axis=1)
```

Plotting graphs:

For simplicity, let’s use Python Data Visualization library Altair to create some simple visualizations. Altair is a declarative statistical visualization library for Python, based on Vega and Vega-Lite. Altair offers a powerful and concise visualization grammar that enables you to build a wide range of statistical visualizations quickly.



```
[60] base.encode(y='Confirmed').properties(title='Total Confirmed') | base.encode(y='Recovered').properties(title='Total Recovered') | base.encode(y='Deaths').properties(title='Total Deaths') | base.encode(y='Active').properties(title='Total Active')
```

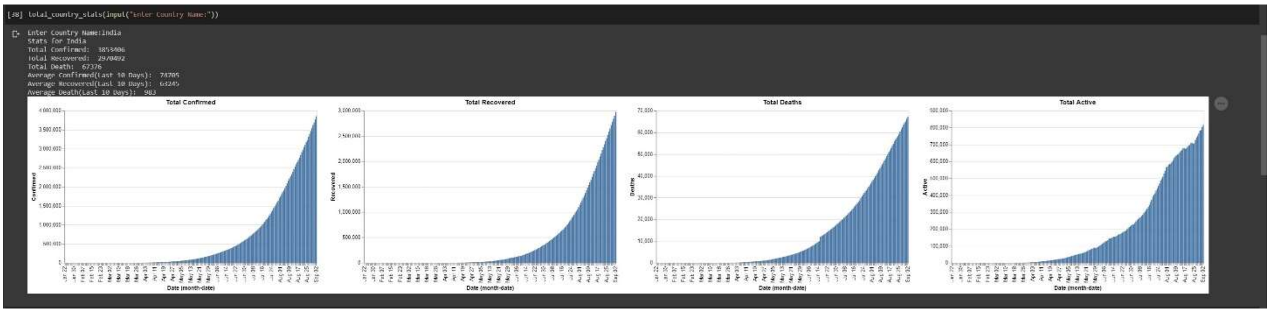
CHAPTER 4

RESULT AND DISCUSSION

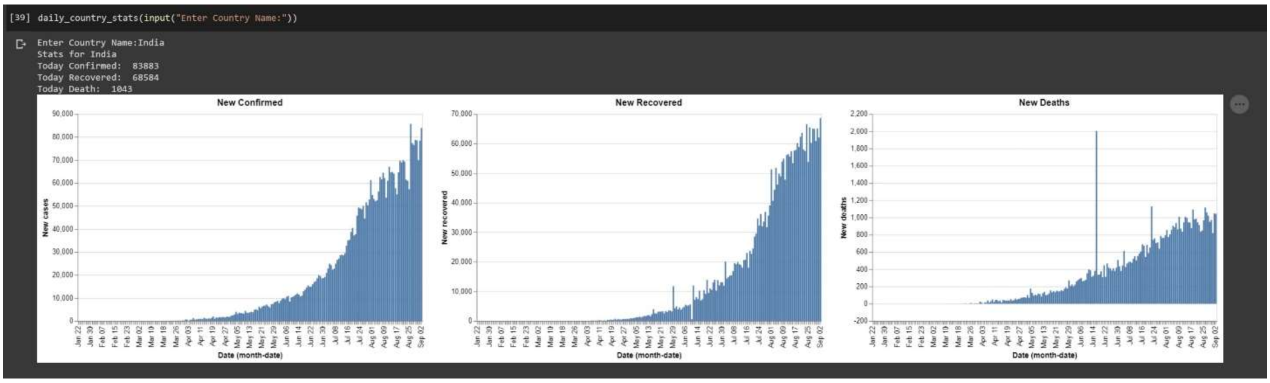
RESULT

Now we have connected everything together and now we will try our data by calling the functions.

- Stats for India:

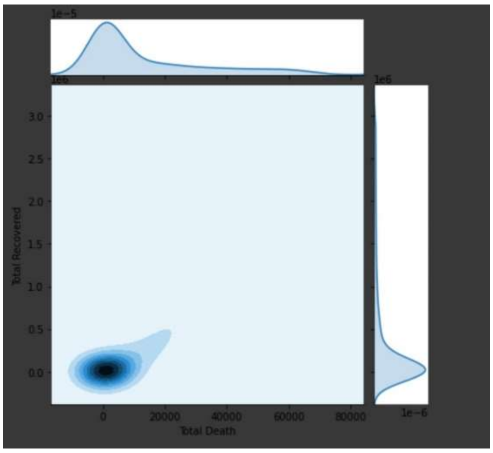


```
[38] total_country_stats(input("Enter Country Name:"))

Enter Country Name:India
Stats for India
Total Confirmed:  3853406
Total Recovered:  2970492
Total Death:  67376
Average Confirmed(Last 10 Days):  74705
Average Recovered(Last 10 Days):  63245
Average Death(Last 10 Days):  983
```

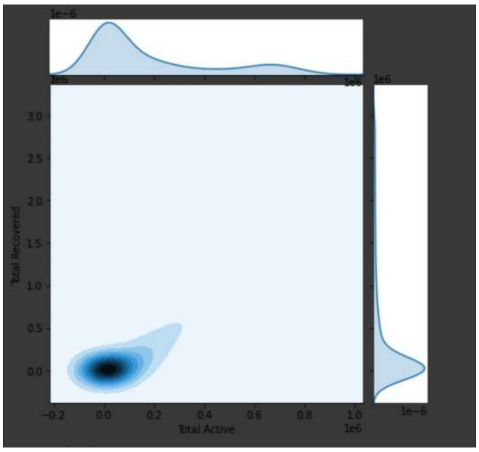


```
[39] daily_country_stats(input("Enter Country Name:"))

Enter Country Name:India
Stats for India
Today Confirmed:  83883
Today Recovered:  68584
Today Death:  1043
```

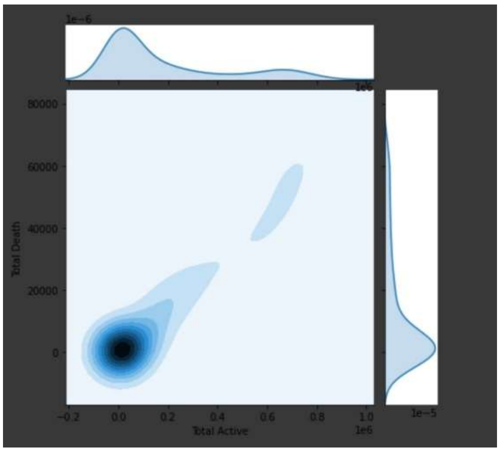
- Density Graphs:



Total Death vs Total Recovered

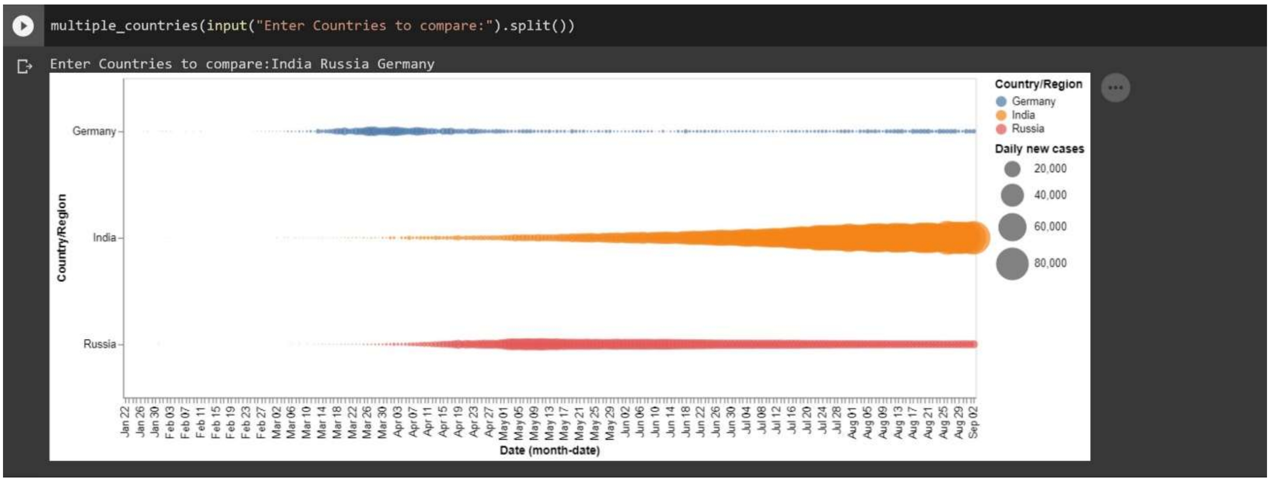


Total Active vs Total Recovered



Total Active vs Total Death

- Comparing different countries (India, Russia, Germany) on Daily New Cases:



```
multiple_countries(input("Enter Countries to compare:").split())
```

```
Enter Countries to compare:India Russia Germany
```

DISCUSSION

Different government of different countries are working hard to control the spread of COVID -19. While US (Population: 32.82 crores (2019)) saw huge spikes in deaths during the start of this pandemic (approx. 2000 per day) while the country with much more populations like India (135.26 crores (2018)) have only recorded 2000 deaths one day and an average of 1000 deaths per day. Aim of this project is to process and help to analyze effect of measures by different countries.

I have used Python as my basic language because it is easy to understand and very efficient and capable of doing computational work. Libraries like Matplotlib, Seaborn and Alter are very much capable of visualizing data efficiently and libraries like pandas are efficient for manipulating data.

CHAPTER 5

WHY DATA PROCESSING?

The main reason behind data processing is that data almost never comes in a form that is ready for us. A large amount of time spent on a data science project is on manipulating data.

Importance of data processing includes increased productivity and profits, better decisions, more accurate and reliable. Further cost reduction, ease in storage, distributing and report making followed by better analysis and presentation are other advantages. The need to process data is now widely realized and reflected in every field of work. Let the work be done in a business atmosphere or for educational research purpose, data management systems are used by every business. It is a multidimensional process which is involved in almost every field of human life. Generally speaking, the term “Data Processing” is used where you have to collect innumerable data files from different sources. You have to arrange them in a way that can be practically beneficial for the purpose you have gathered all that material. It is a task of synchronizing collected data from different sources and convert it to an organized form. This makes it easy to understand and retrieve the specific information anytime. There are various data processing methods which include manual data processing, mechanical data processing and electronic data processing. Data processing is one of the most important daily tasks especially when dealing with big data and performing data mining. All those fields where we can expect a huge data available to settle down like education, banking or transportation now realizes the importance of data processing.

The invention of computer technology was one of the most important events of all time. With the improvement of computer technology and ease of use, it has become a popular technology in the hands of many. Data processing has also become popular with computer systems making it easier to be handled. In the current times of multiple industries ruling the economy of the many countries of the world, data processing is a field that has numerous applications in most fields like business, education, healthcare, research and more. The importance is increasing with the increase in advancement in areas like data science, machine learning, artificial intelligence, data quality and data security etc.

The invention of computer technology was one of the most important events of all time. With the improvement of computer technology and ease of use, it has become a popular technology in the hands of many. Data processing has also become popular with computer systems making it easier to be handled. In the current times of multiple industries ruling the economy of the many countries of the world, data processing is a field that has numerous applications in most fields like business, education, healthcare, research and more. The importance is increasing with the increase in advancement in areas like data science, machine learning, artificial intelligence, data quality and data security etc.

Data is being collected by almost everyone either knowingly or unknowingly. Collection of data is the first step but processing of data is another vital activity. Companies, institutes & various groups all over the world are engaged in the work of data processing. While talking about the importance of data processing it is equally important to be aware about the related aspects right starting from the methods of data collection, data processing, data processing cycle, information processing cycle, methods of data processing, types of data processing, data presentation and analysis till the data management best practices.

SPEED, ACCURATE AND RELIABLE

Data processing is important to make sure that all of the work that is done through that collected facts and figures is done quite speedily and without making any errors. When a data is collected and figured through computers, there are no or negligible chance of errors. It is almost guaranteed that the further processes will be done with maximum possible accuracy. If the input data is accurate then the output is always accurate. Processing can be done at a greater speed and with higher accuracy of right set and combination of software’s are used. Another importance of data processing is a major advantage when working in a competitive environment. It is not uncommon to have access to same data. Data and information with higher degree of accuracy is more reliable. Predictive modelling, data cleaning, data validation batch processing is necessary for accurate data.

COST IS REDUCED

Data once collected acts as asset for any group and having it stored provides easy access to when required. This eliminates the need to collect data again and again. Moreover, it is very easy and convenient to make copied of the stored data when stored in digital form. Sending or transferring the data is also much easier. This directly helps in cost reduction. The cost or loss which a company might incur because of lack of information is also drastically reduced. This is so as processed data enables it to take a wise and informed decision thus again saving on huge cost.

Other benefits and merits of Data Processing are:

1. Data processing makes it easier to validate actions and changes and transactions easily and reduce dependence on computational power for collecting them on demand from a basic form.
2. Insurance claims can be easily handled and settled with properly processed data and make it time-saving for the police authorities as well. Keeping and managing health records, creating electronic health records is now possible due to batch processing, powerful & reliable data warehouses.
3. Data processing can also be made to include image processing and make it easier to present any data to users in a readable format that is liked by them.
4. Invoices can be easily generated for services which have been used and make the customer experience better.
5. Data processing in the form of word processing can help in making documents which are readable and likeable by readers and be made even more engaging.

CHAPTER 6

CONCLUSION & FUTURE WORK

SUMMARY:

The intension of this project is to process different data sets of COVID – 19 such as confirmed, recovered and death and plots statics graph to help analyze the approach and its effect on different countries.

FUTURE WORK:

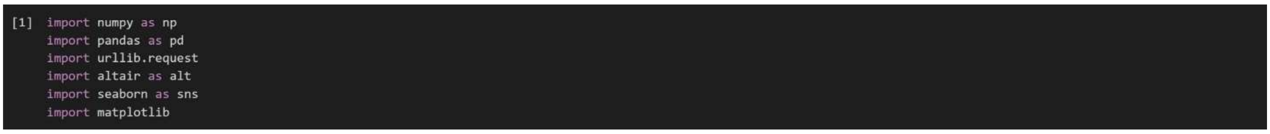
In future projects I will improve the design of my program by giving User Interface and more graph options. Also, would put more extra features like prediction modelling of countries.

CONCLUSION:

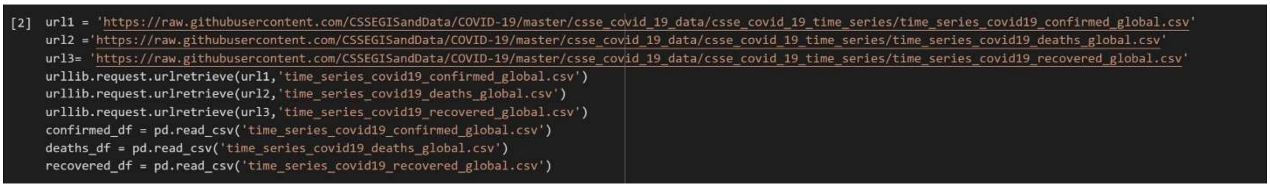
Open the Jupyter Notebook and run all the cell. When prompted enter country names or a set of country name as necessary for the analysis.

CHAPTER 7

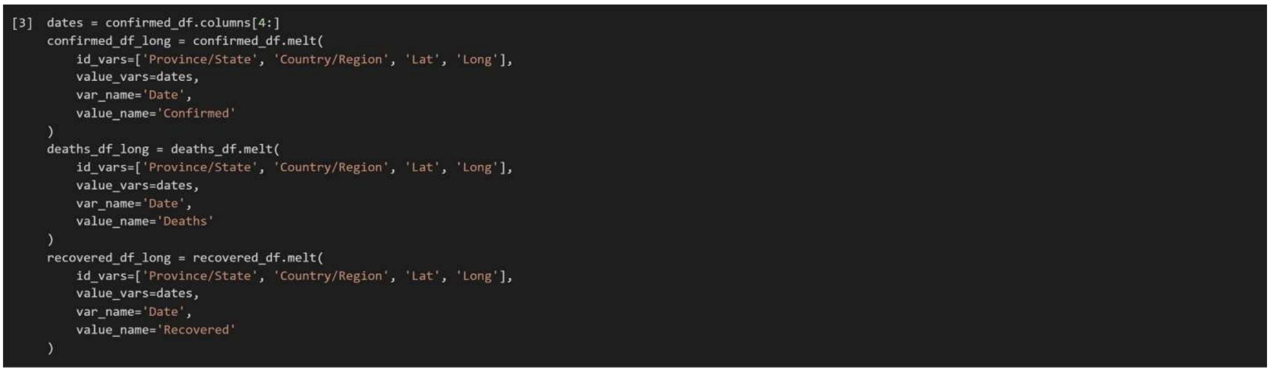
SOURCE CODE:



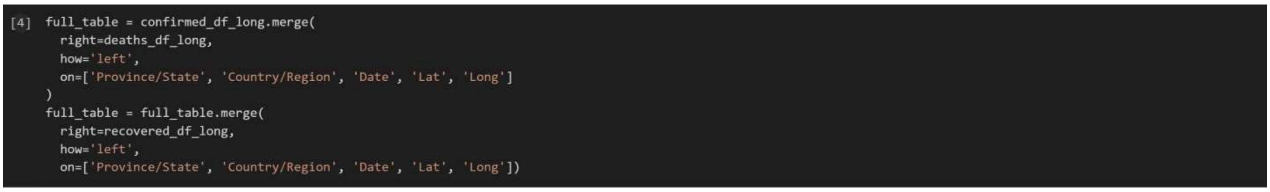
```
[1]  import numpy as np
     import pandas as pd
     import urllib.request
     import altair as alt
     import seaborn as sns
     import matplotlib
```



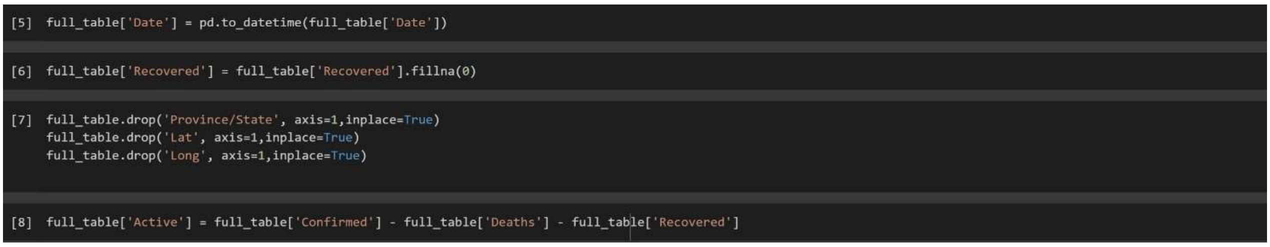
```
[2]  url1 = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv'
     url2 ='https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_deaths_global.csv'
     url3= 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_recovered_global.csv'
     urllib.request.urlretrieve(url1,'time_series_covid19_confirmed_global.csv')
     urllib.request.urlretrieve(url2,'time_series_covid19_deaths_global.csv')
     urllib.request.urlretrieve(url3,'time_series_covid19_recovered_global.csv')
     confirmed_df = pd.read_csv('time_series_covid19_confirmed_global.csv')
     deaths_df = pd.read_csv('time_series_covid19_deaths_global.csv')
     recovered_df = pd.read_csv('time_series_covid19_recovered_global.csv')
```



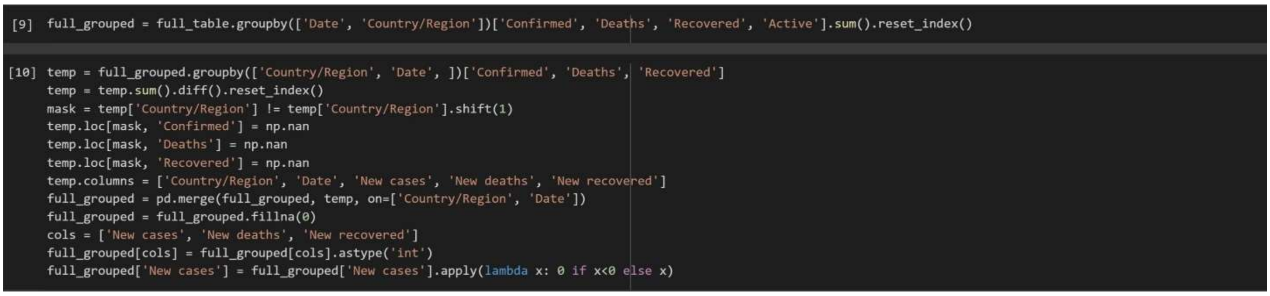
```
[3]  dates = confirmed_df.columns[4:]
     confirmed_df_long = confirmed_df.melt(
         id_vars=['Province/State', 'Country/Region', 'Lat', 'Long'],
         value_vars=dates,
         var_name='Date',
         value_name='Confirmed'
     )
     deaths_df_long = deaths_df.melt(
         id_vars=['Province/State', 'Country/Region', 'Lat', 'Long'],
         value_vars=dates,
         var_name='Date',
         value_name='Deaths'
     )
     recovered_df_long = recovered_df.melt(
         id_vars=['Province/State', 'Country/Region', 'Lat', 'Long'],
         value_vars=dates,
         var_name='Date',
         value_name='Recovered'
     )
```



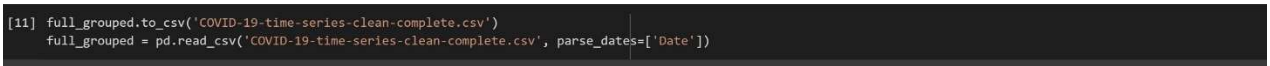
```
[4]  full_table = confirmed_df_long.merge(
       right=deaths_df_long,
       how='left',
       on=['Province/State', 'Country/Region', 'Date', 'Lat', 'Long']
     )
     full_table = full_table.merge(
       right=recovered_df_long,
       how='left',
       on=['Province/State', 'Country/Region', 'Date', 'Lat', 'Long'])
```



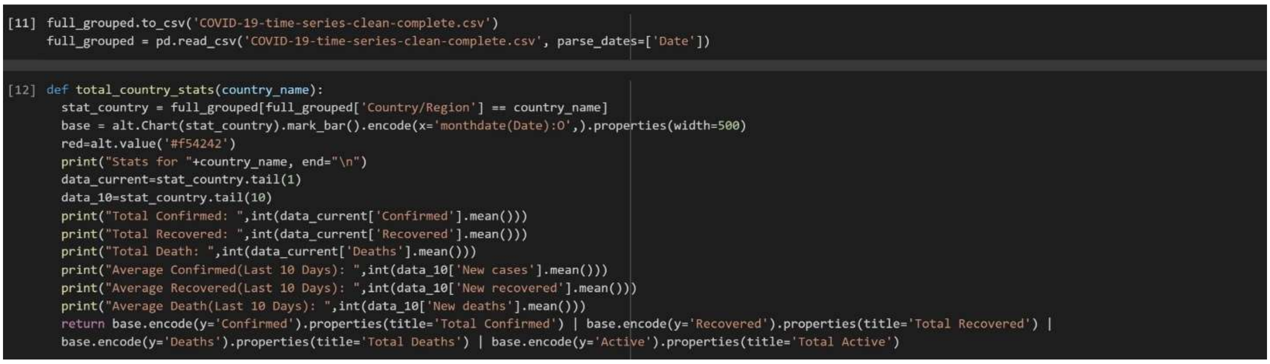
```
[5]  full_table['Date'] = pd.to_datetime(full_table['Date'])

[6]  full_table['Recovered'] = full_table['Recovered'].fillna(0)

[7]  full_table.drop('Province/State', axis=1,inplace=True)
     full_table.drop('Lat', axis=1,inplace=True)
     full_table.drop('Long', axis=1,inplace=True)

[8]  full_table['Active'] = full_table['Confirmed'] - full_table['Deaths'] - full_table['Recovered']
```



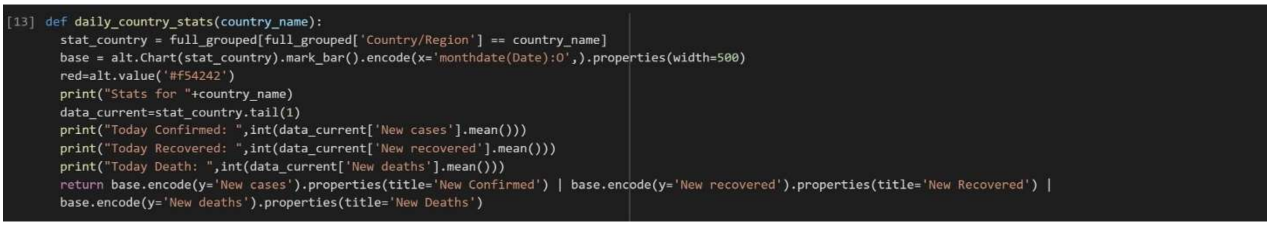
```
[9]  full_grouped = full_table.groupby(['Date', 'Country/Region'])['Confirmed', 'Deaths', 'Recovered', 'Active'].sum().reset_index()

[10] temp = full_grouped.groupby(['Country/Region', 'Date', ])['Confirmed', 'Deaths', 'Recovered']
     temp = temp.sum().diff().reset_index()
     mask = temp['Country/Region'] != temp['Country/Region'].shift(1)
     temp.loc[mask, 'Confirmed'] = np.nan
     temp.loc[mask, 'Deaths'] = np.nan
     temp.loc[mask, 'Recovered'] = np.nan
     temp.columns = ['Country/Region', 'Date', 'New cases', 'New deaths', 'New recovered']
     full_grouped = pd.merge(full_grouped, temp, on=['Country/Region', 'Date'])
     full_grouped = full_grouped.fillna(0)
     cols = ['New cases', 'New deaths', 'New recovered']
     full_grouped[cols] = full_grouped[cols].astype('int')
     full_grouped['New cases'] = full_grouped['New cases'].apply(lambda x: 0 if x<0 else x)
```



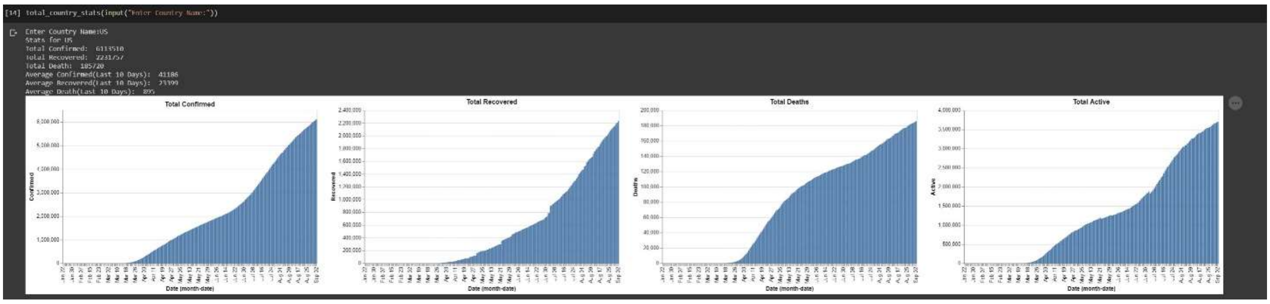
```
[11] full_grouped.to_csv('COVID-19-time-series-clean-complete.csv')
     full_grouped = pd.read_csv('COVID-19-time-series-clean-complete.csv', parse_dates=['Date'])
```



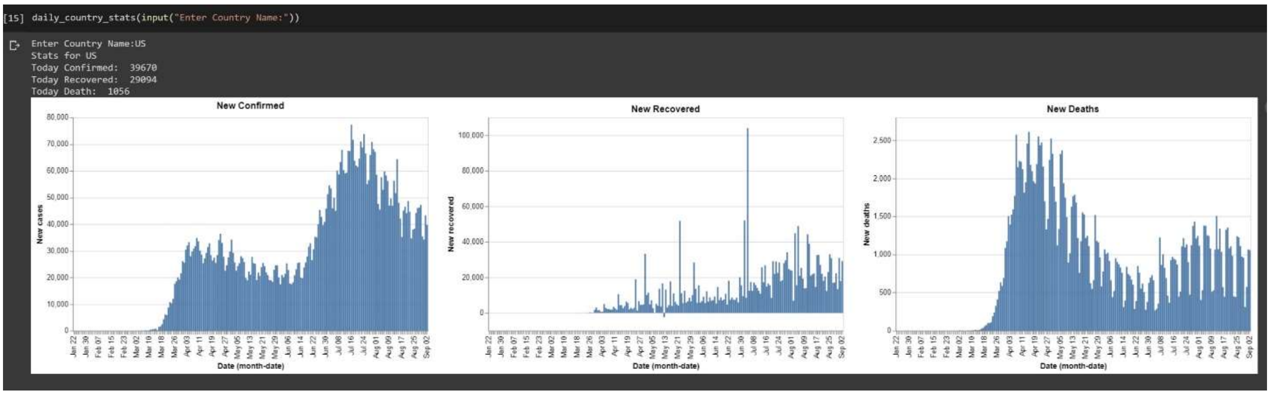
```
[12] def total_country_stats(country_name):
       stat_country = full_grouped[full_grouped['Country/Region'] == country_name]
       base = alt.Chart(stat_country).mark_bar().encode(x='monthdate(Date):O',).properties(width=500)
       red=alt.value('#f54242')
       print("Stats for "+country_name, end="\n")
       data_current=stat_country.tail(1)
       data_10=stat_country.tail(10)
       print("Total Confirmed: ",int(data_current['Confirmed'].mean()))
       print("Total Recovered: ",int(data_current['Recovered'].mean()))
       print("Total Death: ",int(data_current['Deaths'].mean()))
       print("Average Confirmed(Last 10 Days): ",int(data_10['New cases'].mean()))
       print("Average Recovered(Last 10 Days): ",int(data_10['New recovered'].mean()))
       print("Average Death(Last 10 Days): ",int(data_10['New deaths'].mean()))
       return base.encode(y='Confirmed').properties(title='Total Confirmed') | base.encode(y='Recovered').properties(title='Total Recovered') |
       base.encode(y='Deaths').properties(title='Total Deaths') | base.encode(y='Active').properties(title='Total Active')
```



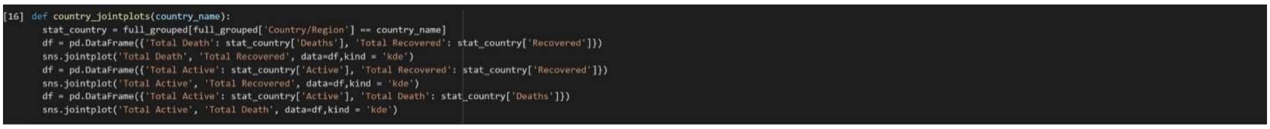
```
[13] def daily_country_stats(country_name):
       stat_country = full_grouped[full_grouped['Country/Region'] == country_name]
       base = alt.Chart(stat_country).mark_bar().encode(x='monthdate(Date):O',).properties(width=500)
       red=alt.value('#f54242')
       print("Stats for "+country_name)
       data_current=stat_country.tail(1)
       print("Today Confirmed: ",int(data_current['New cases'].mean()))
       print("Today Recovered: ",int(data_current['New recovered'].mean()))
       print("Today Death: ",int(data_current['New deaths'].mean()))
       return base.encode(y='New cases').properties(title='New Confirmed') | base.encode(y='New recovered').properties(title='New Recovered') |
       base.encode(y='New deaths').properties(title='New Deaths')
```



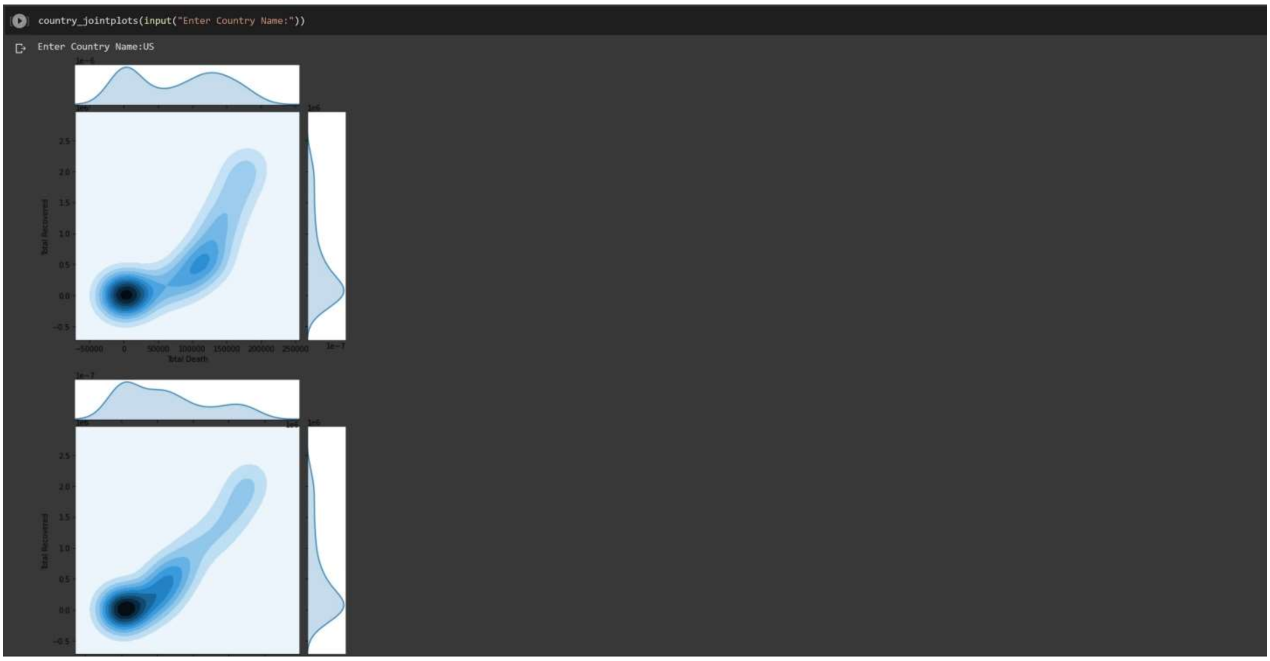
```
[14] total_country_stats(input("Enter Country Name:"))

  Enter Country Name:US
  Stats for US
  Total Confirmed:  6113510
  Total Recovered:  2231757
  Total Death:  185720
  Average Confirmed(Last 10 Days):  41186
  Average Recovered(Last 10 Days):  23399
  Average Death(Last 10 Days):  895
```



```
[15] daily_country_stats(input("Enter Country Name:"))

  Enter Country Name:US
  Stats for US
  Today Confirmed:  39670
  Today Recovered:  29094
  Today Death:  1056
```

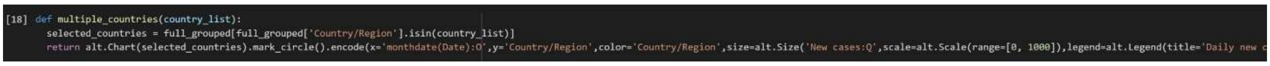


```
[16] def country_jointplots(country_name):
       stat_country = full_grouped[full_grouped['Country/Region'] == country_name]
       df = pd.DataFrame({'Total Death': stat_country['Deaths'], 'Total Recovered': stat_country['Recovered']})
       sns.jointplot('Total Death', 'Total Recovered', data=df,kind = 'kde')
       df = pd.DataFrame({'Total Active': stat_country['Active'], 'Total Recovered': stat_country['Recovered']})
       sns.jointplot('Total Active', 'Total Recovered', data=df,kind = 'kde')
       df = pd.DataFrame({'Total Active': stat_country['Active'], 'Total Death': stat_country['Deaths']})
       sns.jointplot('Total Active', 'Total Death', data=df,kind = 'kde')
```

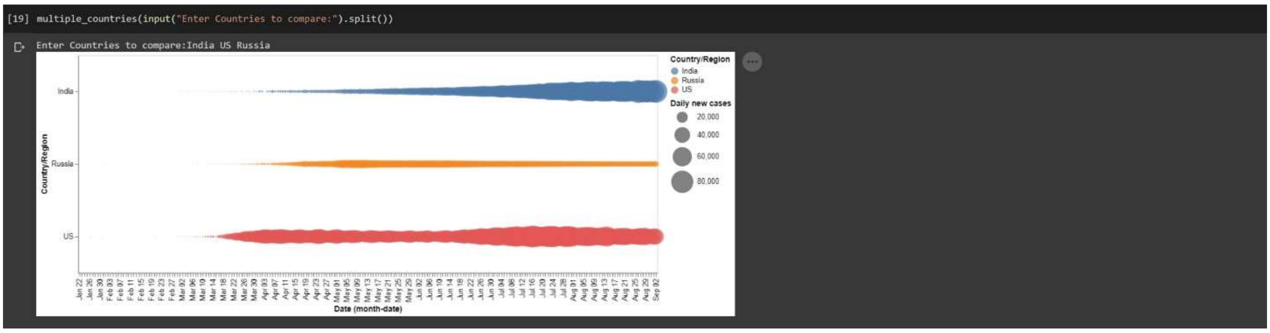


```
country_jointplots(input("Enter Country Name:"))
```

```
Enter Country Name:US
```



```
[18] def multiple_countries(country_list):
       selected_countries = full_grouped[full_grouped['Country/Region'].isin(country_list)]
       return alt.Chart(selected_countries).mark_circle().encode(x='monthdate(Date):O',y='Country/Region',color='Country/Region',size=alt.Size('New cases:Q',scale=alt.Scale(range=[0, 1000]),legend=alt.Legend(title='Daily new c
```



```
[19] multiple_countries(input("Enter Countries to compare:").split())
```

```
Enter Countries to compare:India US Russia
```

CHAPTER 8

REFERENCES:

I. IBM Data Science Professional Coursera

II. Geeks for Geeks

III. Stack Over Flow

IV. Krish Naik Youtube