Jupyter Notebook Data Analysis Documentation

Introduction-

Jupyter Notebook is an open-source, web-based interactive computing environment that enables users to create and share documents with real-time code, equations, visualisations, and narrative text. It is frequently used for data analysis, data visualisation, machine learning, and other purposes.

This documentation offers a thorough tutorial on how to use Jupyter Notebook to carry out data analysis.

Installation-

Run the following commands on the terminal

In windows

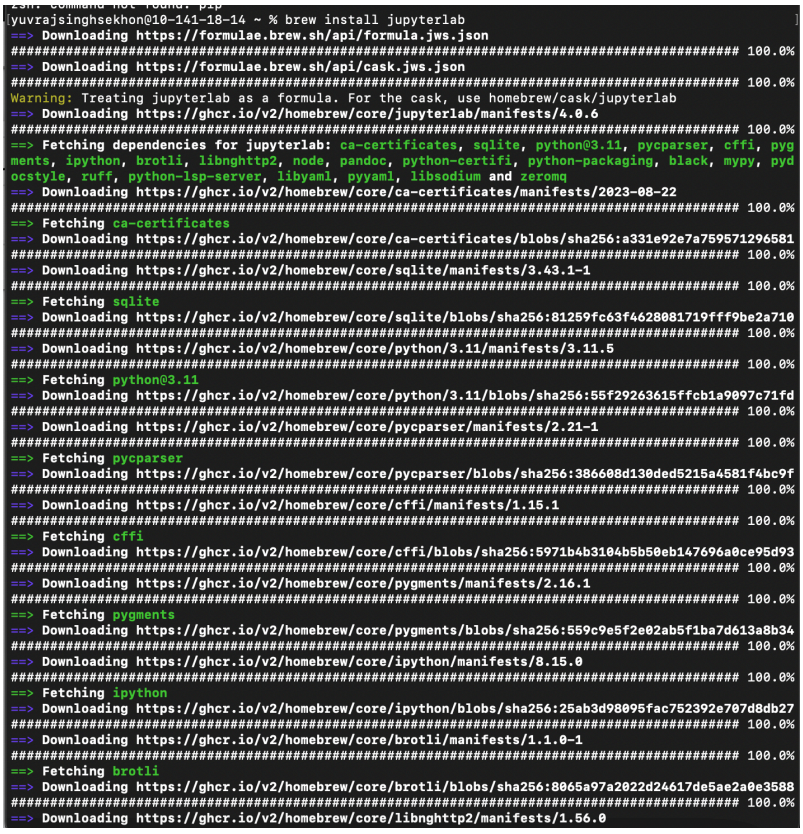
```
pip install notebook
```

And to launch

```
jupyter notebook
```

In Mac or Linux run the following command-

```
brew install jupyterlab
```

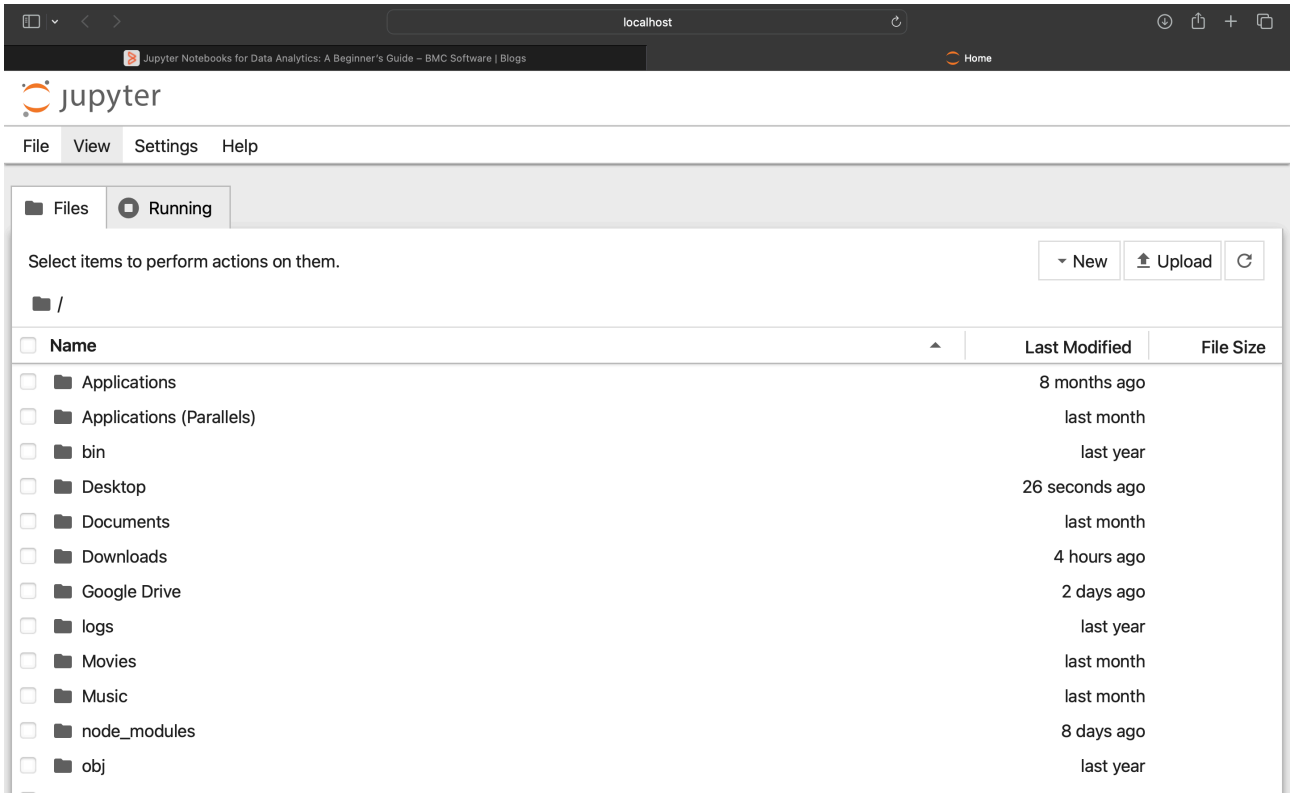


And to launch

```
jupyter notebook
```

```
yuvrajsinghsekhon@10-141-18-14 ~ % jupyter notebook
[I 2023-09-21 14:30:16.086 ServerApp] Package notebook took 0.0000s to import
[I 2023-09-21 14:30:16.101 ServerApp] Package jupyter_lsp took 0.0153s to import
[W 2023-09-21 14:30:16.101 ServerApp] A `_jupyter_server_extension_points` function was not found in
 jupyter_lsp. Instead, a `_jupyter_server_extension_paths` function was found and will be used for n
ow. This function name will be deprecated in future releases of Jupyter Server.
[I 2023-09-21 14:30:16.111 ServerApp] Package jupyter_server_terminals took 0.0101s to import
[I 2023-09-21 14:30:16.112 ServerApp] Package jupyterlab took 0.0000s to import
[I 2023-09-21 14:30:17.536 ServerApp] Package notebook_shim took 0.0000s to import
[W 2023-09-21 14:30:17.536 ServerApp] A `_jupyter_server_extension_points` function was not found in
 notebook_shim. Instead, a `_jupyter_server_extension_paths` function was found and will be used for
 now. This function name will be deprecated in future releases of Jupyter Server.
[I 2023-09-21 14:30:17.537 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2023-09-21 14:30:17.539 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2023-09-21 14:30:17.541 ServerApp] jupyterlab | extension was successfully linked.
[I 2023-09-21 14:30:17.543 ServerApp] notebook | extension was successfully linked.
[I 2023-09-21 14:30:17.544 ServerApp] Writing Jupyter server cookie secret to /Users/yuvrajsinghsekh
on/Library/Jupyter/runtime/jupyter_cookie_secret
[I 2023-09-21 14:30:18.058 ServerApp] notebook_shim | extension was successfully linked.
[I 2023-09-21 14:30:18.137 ServerApp] notebook_shim | extension was successfully loaded.
[I 2023-09-21 14:30:18.138 ServerApp] jupyter_lsp | extension was successfully loaded.
[I 2023-09-21 14:30:18.139 ServerApp] jupyter_server_terminals | extension was successfully loaded.
[I 2023-09-21 14:30:18.140 LabApp] JupyterLab extension loaded from /opt/homebrew/Cellar/jupyterlab/
4.0.6/libexec/lib/python3.11/site-packages/jupyterlab
[I 2023-09-21 14:30:18.140 LabApp] JupyterLab application directory is /opt/homebrew/Cellar/jupyterl
ab/4.0.6/libexec/share/jupyter/lab
[I 2023-09-21 14:30:18.140 LabApp] Extension Manager is 'pypi'.
[I 2023-09-21 14:30:18.141 ServerApp] jupyterlab | extension was successfully loaded.
[I 2023-09-21 14:30:18.143 ServerApp] notebook | extension was successfully loaded.
[I 2023-09-21 14:30:18.143 ServerApp] Serving notebooks from local directory: /Users/yuvrajsinghsekh
on
[I 2023-09-21 14:30:18.143 ServerApp] Jupyter Server 2.7.3 is running at:
[I 2023-09-21 14:30:18.143 ServerApp] http://localhost:8888/tree?token=25f6a7859d649fdc99481dafe6a6f
0c609f1e6a4fead8625
[I 2023-09-21 14:30:18.143 ServerApp]     http://127.0.0.1:8888/tree?token=25f6a7859d649fdc99481dafe
6a6f0c609f1e6a4fead8625
[I 2023-09-21 14:30:18.143 ServerApp] Use Control-C to stop this server and shut down all kernels (t
wice to skip confirmation).
[C 2023-09-21 14:30:18.148 ServerApp]

    To access the server, open this file in a browser:
        file:///Users/yuvrajsinghsekhon/Library/Jupyter/runtime/jpserver-33481-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/tree?token=25f6a7859d649fdc99481dafe6a6f0c609f1e6a4fead8625
        http://127.0.0.1:8888/tree?token=25f6a7859d649fdc99481dafe6a6f0c609f1e6a4fead8625
[I 2023-09-21 14:30:18.417 ServerApp] Skipped non-installed server(s): bash-language-server, dockerf
```



Using the supplied URL (https://localhost:8888), we can reach the JupyterLab installation after starting the JupyterLab.

Install via Anaconda data science toolkit

Anaconda is a flexible platform made for scientific computing and data science. It streamlines the installation process for Python-based projects by providing a carefully curated selection of pre-installed libraries and packages. Anaconda's Conda package manager makes environment management simple and enables users to build segregated workspaces with certain requirements. This ensures efficiency and repeatability in jobs involving data analysis and development.

Benefits of using Jupyter Lab via Anaconda-

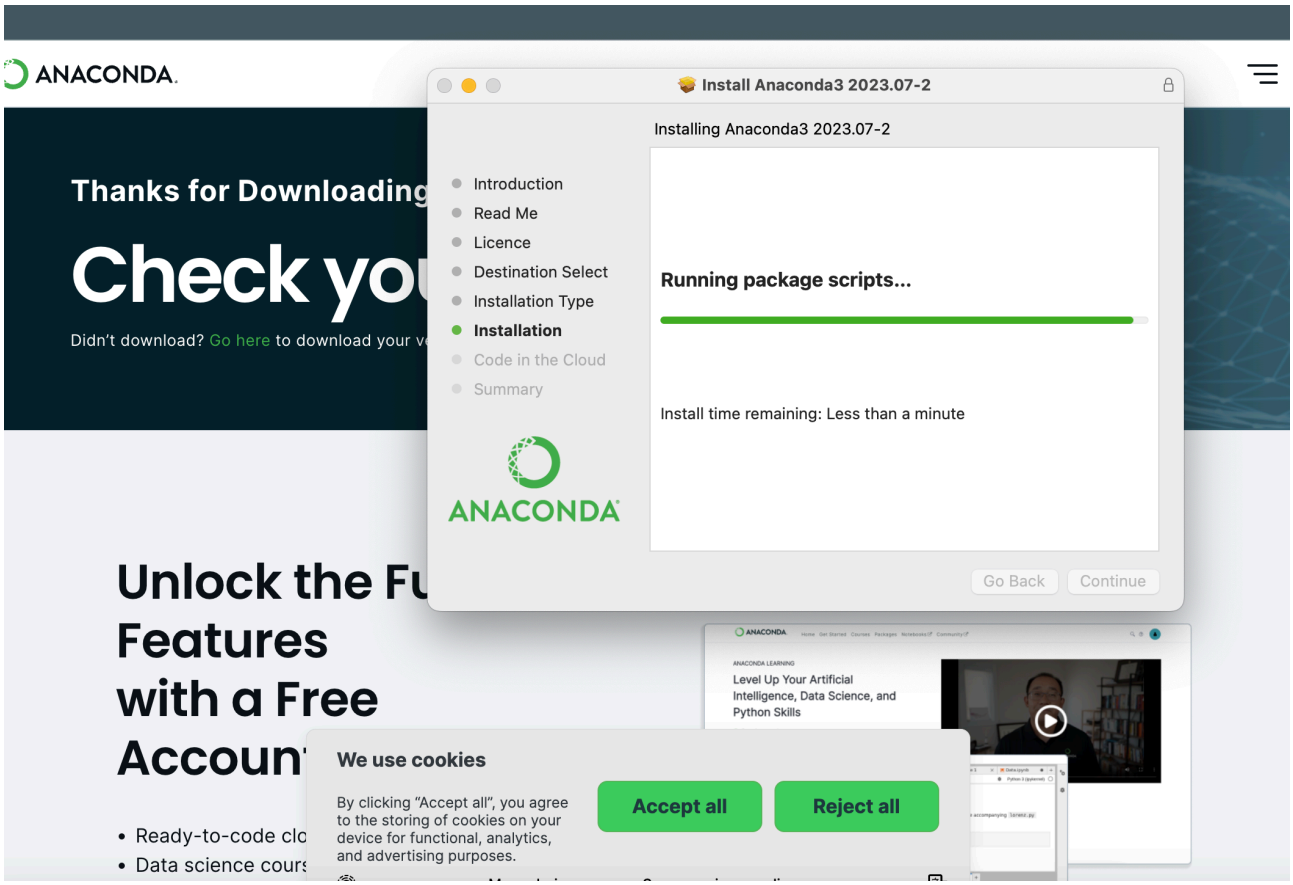
- Easy Package Management with pre-packaged data science libraries.
- Conda Environment Management for isolated and reproducible environments.
- Simplified Installation and Setup for Jupyter Lab and dependencies.
- Cross-platform compatibility across Windows, macOS, and Linux.
- Integrated Development Environment (IDE) for efficient work.
- Rich Text and Multimedia Support with Markdown capabilities.
- File Management and Version Control with Git integration.
- Extensions and Customization for tailored workflows.
- Efficient Resource Use with isolated environments.

- Active Community and Support for assistance and resources.

To launch Jupyter Lab via Anaconda, follow these steps:

1. Install Anaconda:
Download anaconda through their official website.

2.Open Anaconda Navigator:
The Anaconda Navigator should be opened after installation. We can look for it in your programmes or in the search bar on your computer.



3. Launch Jupyter Lab:

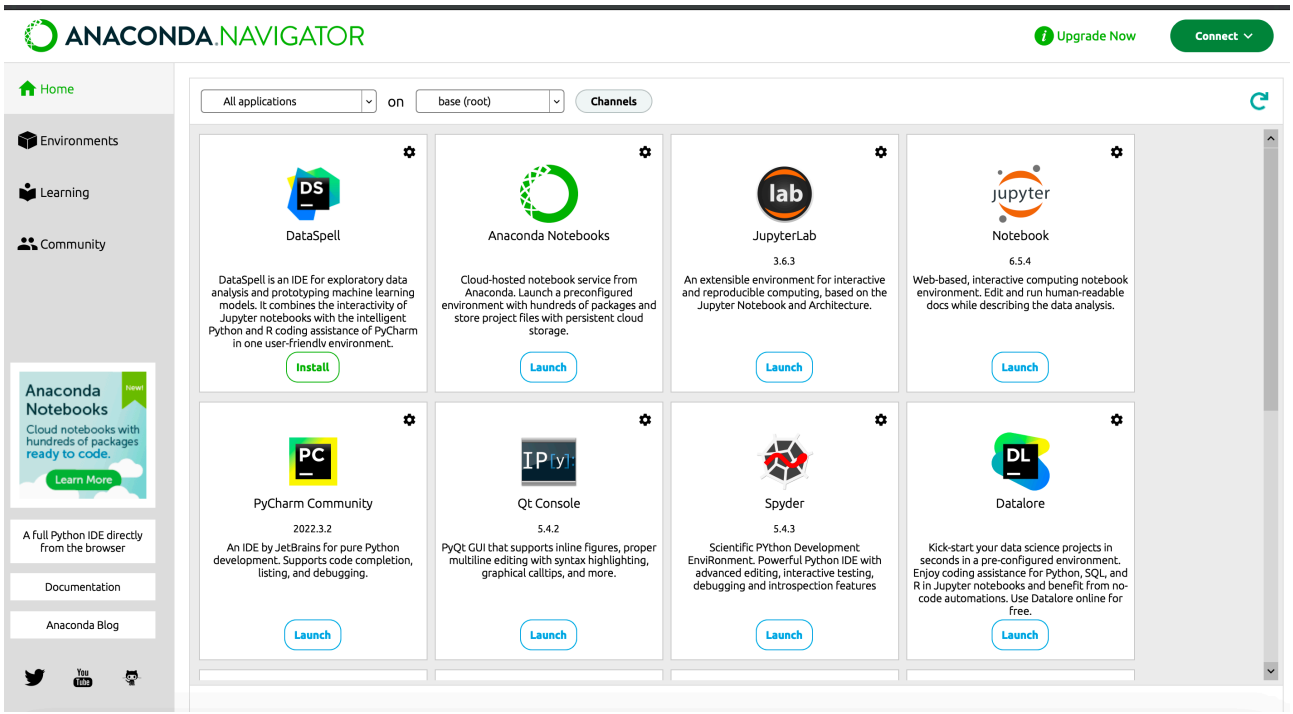
On the left side of the Anaconda Navigator, there is a list of installed environments. Locate the workspace you want to use (such as the base environment or any other custom workspace you've developed). then select that environment's "Home" button.

Next, select JupyterLab and then click the "Launch" button. This will launch the Jupyter Lab interface in a new tab in your default web browser.

4. Navigating Jupyter Lab:
A file browser, which resembles a file explorer, will be shown on the left once Jupyter Lab has been launched. You may search through your files and add new ones, such as notebooks.

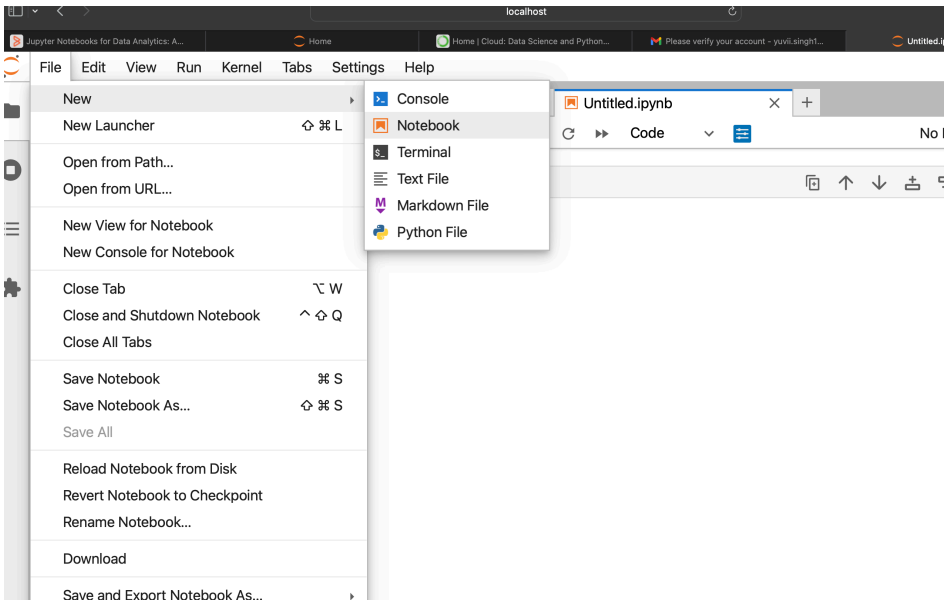
The workspace is the major section of the interface, where you can open and use text files, consoles, notebooks, and other tools.

There are tabs for open files, active terminals, and other utilities on the right side.



5. Creating a New Notebook:

Click the "+ Notebook" button in the file browser or select File > New > Notebook to start a new Jupyter Notebook. The workspace will then launch a new notepad.



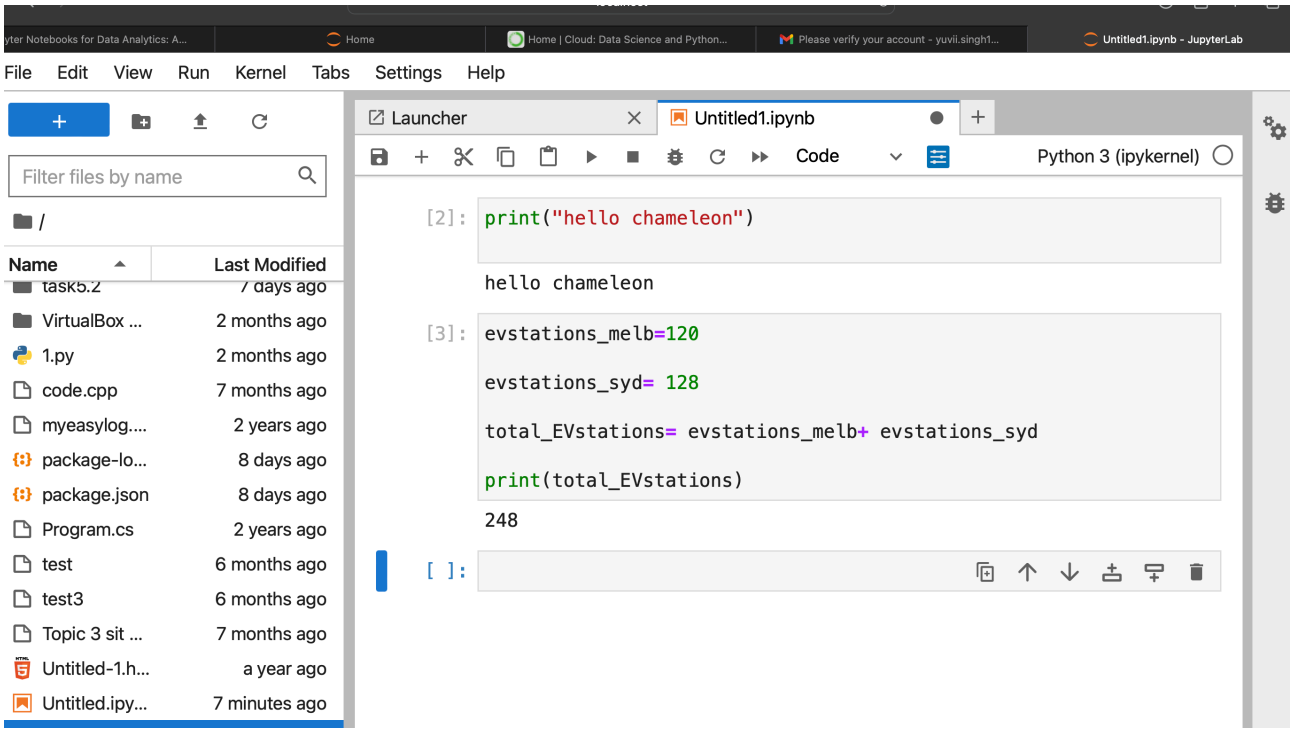
6. Using Jupyter Lab:

Jupyter Lab can be used in the same way as a Jupyter Notebook. You can use markdown cells to generate documentation, write and run code in code cells, and use other capabilities like file management and code completion.

To save your work, click the disc icon in the toolbar or press Ctrl + S on a computer running Windows or Linux or Cmd + S on a Mac.

Components of Jupyter notebook

(i) Jupyter Notebook files, also known as **.ipynb** files, are a type of document that incorporate live code, visuals, and narrative text. By allowing users to run code cells individually, see instant results, and incorporate rich media like graphics and equations, it facilitates interactive computing. Ipynb files, which are widely used in data science and research, provide a flexible and cooperative framework for coding and documentation.



(ii) The computational engine that runs the code in a Jupyter Notebook is called the **notebook kernel.** The chosen programming language for the notebook is interpreted and processed by it, keeping the current state of the computations throughout the session.

(iii) A "cell" in a Jupyter Notebook refers to a distinct element of the notebook's user interface. The building blocks of your document are cells, each of which can contain either code or text. There are two primary cell types:

code cells:

For writing and running code, use code cells. Typically, they are connected to a certain programming language, such as Python, R, Julia, etc. These cells allow you to enter and execute code, and the results will be shown below the cell.

```
def values(a,b):
    return a*b

jattpower = values(90,78)
print(jattpower)

print("this is a code cell")
print("hi mates")
```

```
7020
this is a code cell
hi mates
```

Markdown Cells:

The Markdown syntax is used in cells to add structured text, headings, lists, links, and more. They enable you to give clarifications, produce documentation, and give your code context. The text in a Markdown cell gets converted into formatted material when it is run.

heading1

This is an example of a markdown cell

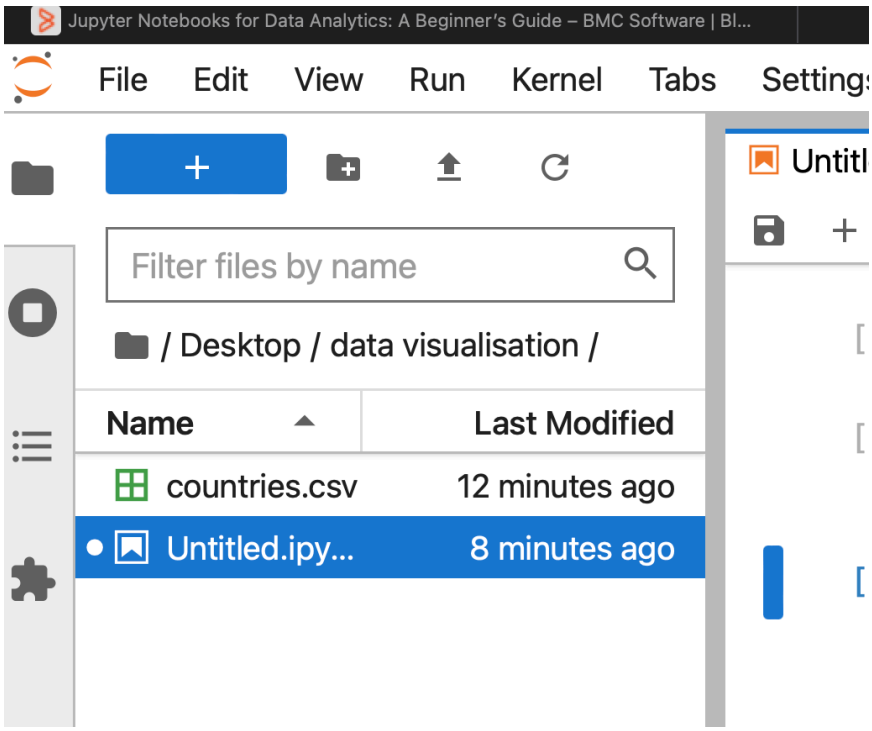
heading 2

it is nice to document the work right!!!!

Data Analysis through jupyter notebook

Follow the given steps below to do data analysis with the help of jupyter notebook-

1) create a folder- Create a folder on the desktop and add the dataset into it for which you want to visualise the data.
2) Navigating that folder in jupyter notebook- Open Jupiter Notebook and then follow this path => File -> Desktop -> folder



3) Commands-

I)

```
import pandas as pd
```

In this step, pandas is imported with a nickname pd and we will use pandas to import and utilize data from.csv files.

ii)

```
from matplotlib import pyplot as plt
```

This will import matplotlib and help in creating charts

iii)

```
data=pd.read_csv('countries.csv')
```

Here **data** is a variable and **pd.read_csv** is a command which will fetch the data from the file which is mentioned in the brackets.
Note- the .csv file should be in the same folder. As that of .ipynb file

We can see the data by just typing the name of the variable which is also data in this case

```
[4]: data
```

[4]:

	country	year	population
0	Afghanistan	1952	8425333
1	Afghanistan	1957	9240934
2	Afghanistan	1962	10267083
3	Afghanistan	1967	11537966
4	Afghanistan	1972	13079460
...	...	...	...
1699	Zimbabwe	1987	9216418
1700	Zimbabwe	1992	10704340
1701	Zimbabwe	1997	11404948
1702	Zimbabwe	2002	11926563
1703	Zimbabwe	2007	12311143

1704 rows × 3 columns

IV)

```
afg= data[data.country== 'Afghanistan']
```

Now to use the data set we will first declare a variable which in this case is afg and then use the command data[data.country==‘Afghanistan’]

Here,
data = name of the data file which we earlier declared
Data.country= the row from which we will take the data
‘afghanistan’=the country name of which values will be taken up

```
[5]: afg= data[data.country== 'Afghanistan']
```

```
[6]: afg
```

[6]:

	country	year	population
0	Afghanistan	1952	8425333
1	Afghanistan	1957	9240934
2	Afghanistan	1962	10267083
3	Afghanistan	1967	11537966
4	Afghanistan	1972	13079460
5	Afghanistan	1977	14880372
6	Afghanistan	1982	12881816
7	Afghanistan	1987	13867957
8	Afghanistan	1992	16317921
9	Afghanistan	1997	22227415

Similar commands were used to extract China's data

```
[9]: china= data[data.country== 'China']
```

```
[10]: china
```

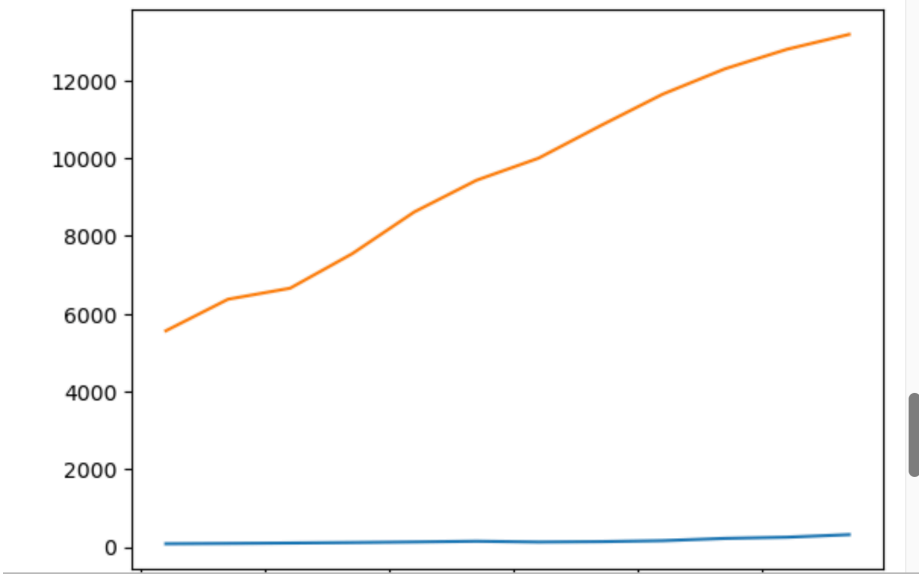
[10]:

	country	year	population
288	China	1952	556263527
289	China	1957	637408000
290	China	1962	665770000
291	China	1967	754550000
292	China	1972	862030000
293	China	1977	943455000
294	China	1982	1000281000
295	China	1987	1084035000
296	China	1992	1164970000

Then to make graphs following command were used-

We divided the population by 10 to the power 5 to make it a smaller number and so that comparison can become easy.

```
]: plt.plot(afg.year, afg.population/ 10**5)
   plt.plot(china.year,china.population/ 10**5)
   plt.show()
```



This was a detailed example to demonstrate data analysis through jupyter notebook.

Now you can go to the official website to check more commands and play with them through different datasets.