# *Creating a data science environment with Jupyter and Docker*

***by***

***PL. Nagarathinam***

***Course on AWS and Devops***

## Jupyter:

**“Jupyter”** can mean a lot of things.

**“Project Jupyter”** is a large umbrella project that covers many different software offerings and tools. That includes **Jupyter Notebook** and **JupyterLab**, which are both popular notebook-editor programs. The Jupyter project, and its subprojects, all centre around providing tools and standards for interactive computing with computational notebooks.

Jupyter Notebook and JupyterLab are both open-source web applications that allow you to create and share documents containing live code, equations, visualizations, and narrative text. The primary purpose of these tools is to provide an interactive environment for data exploration, analysis, and visualization.

- **Jupyter Notebook**

  Jupyter Notebook is a widely adopted web-based interface for creating and sharing documents that contain live code, equations, visualizations, and narrative text. It was first introduced in 2014 as part of the IPython project and later evolved into a separate project under the name Jupyter. The name “Jupyter” is a combination of three programming languages: Julia, Python, and R, highlighting its support for multiple languages through the use of different kernels.

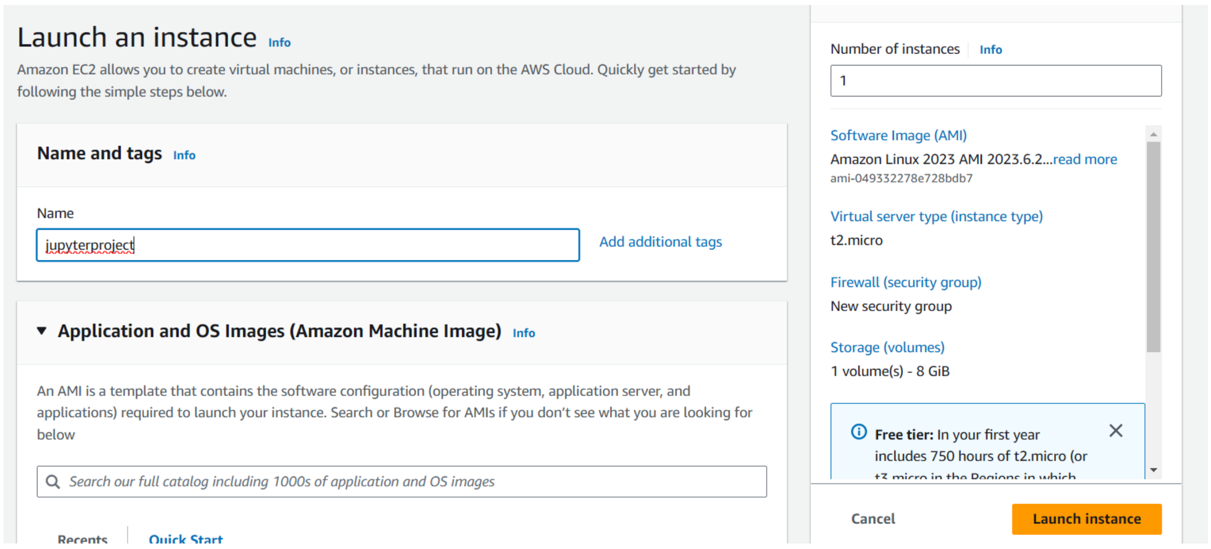
- **JupyterLab**

  JupyterLab is the next-generation web-based user interface for Project Jupyter. Launched in 2018, it offers all the familiar building blocks of the classic Jupyter Notebook (notebook, terminal, text editor, file browser, rich outputs, etc.) in a flexible and powerful user interface. JupyterLab aims to provide a more integrated and extensible environment for working with Jupyter Notebooks and other Jupyter components

**Using the docker tool to connect the Jupyter server and connect to the web server, and with the aid of AWS cloud in EC2,**

Step 1:

Open the AWS console and launch the new Instances and named it, then connect to the terminal



Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: jupyterproject

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents | Quick Start

Number of instances Info: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.6.2...read more ami-049332278e728bdb7

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or

Cancel | Launch instance

***Step 2:***

Next install the docker tool and docker compose in the instance to access the jupyter in the web browsers with basics Linux commands

```
sudo apt update

sudo apt install docker.io

sudo apt install docker-compose
```

***Step 3:***

After installation create the folder and add docker compose.yml file and give some script to access the jupyter

```
mkdir jupyterfolder

nano docker-compose.yml
```

**Insert the script**

version: '3'

services:

  jupyter:

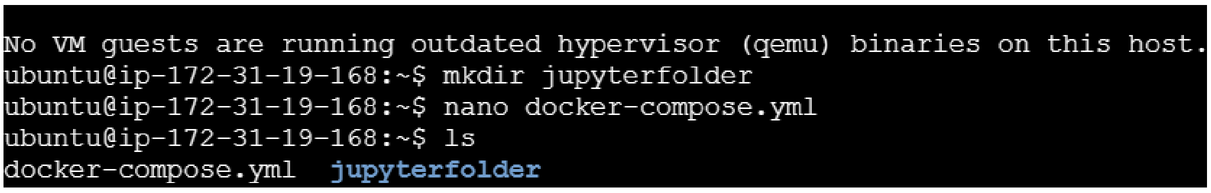
    image: jupyter/scipy-notebook

    ports:

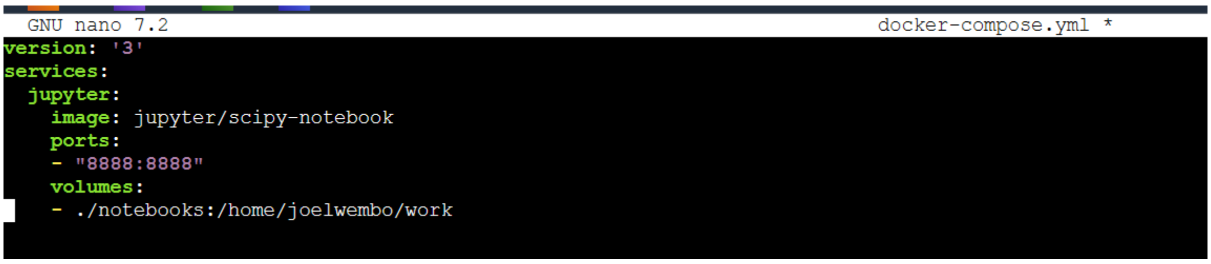
    - "8888:8888"

    volumes:

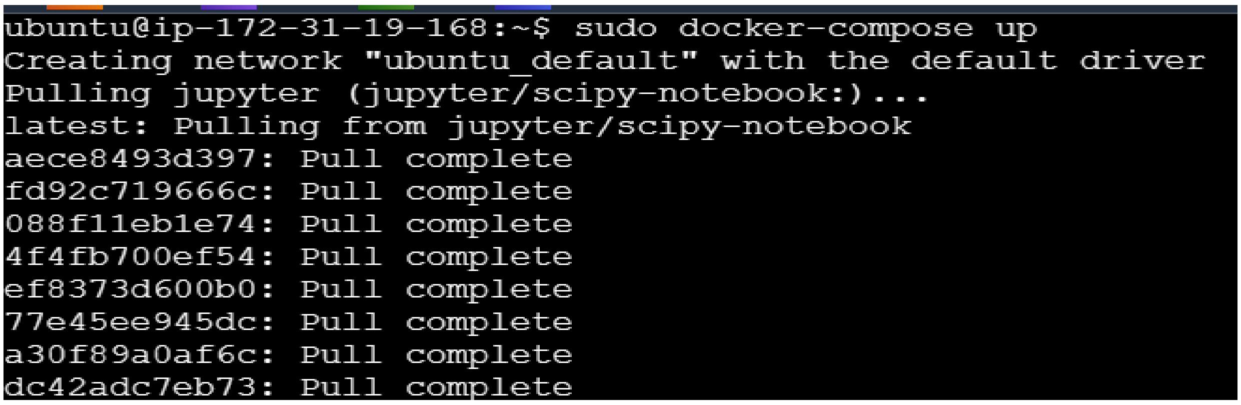
    - ./notebooks:/home/joelwembo/work



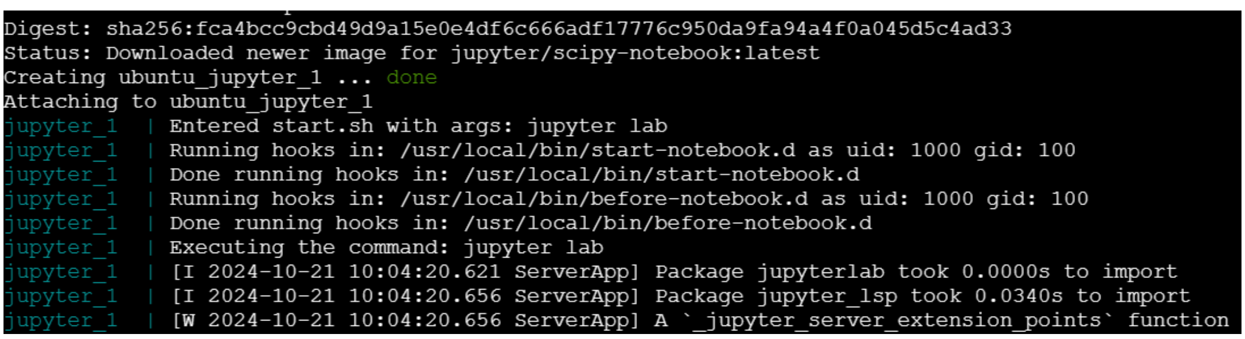
```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-19-168:~$ mkdir jupyterfolder
ubuntu@ip-172-31-19-168:~$ nano docker-compose.yml
ubuntu@ip-172-31-19-168:~$ ls
docker-compose.yml  jupyterfolder
```



```
  GNU nano 7.2                                                docker-compose.yml *
version: '3'
services:
  jupyter:
    image: jupyter/scipy-notebook
    ports:
    - "8888:8888"
    volumes:
    - ./notebooks:/home/joelwembo/work
```



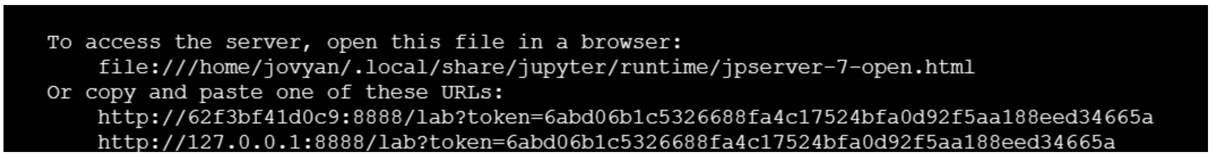
```
ubuntu@ip-172-31-19-168:~$ sudo docker-compose up
Creating network "ubuntu_default" with the default driver
Pulling jupyter (jupyter/scipy-notebook:)...
latest: Pulling from jupyter/scipy-notebook
aece8493d397: Pull complete
fd92c719666c: Pull complete
088f11eb1e74: Pull complete
4f4fb700ef54: Pull complete
ef8373d600b0: Pull complete
77e45ee945dc: Pull complete
a30f89a0af6c: Pull complete
dc42adc7eb73: Pull complete
```



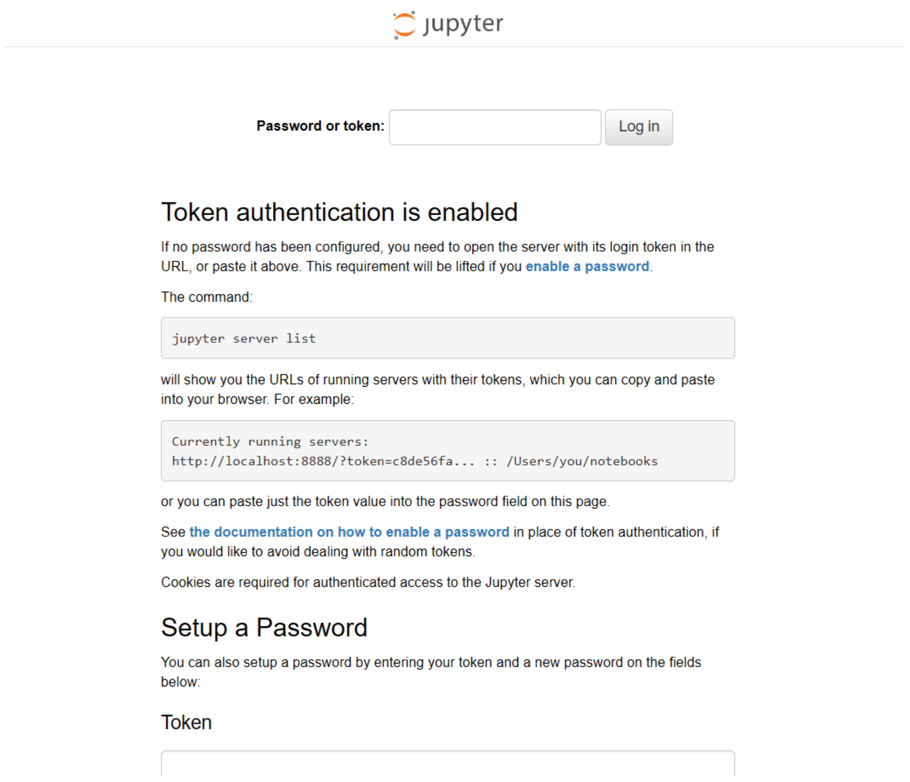
```
Digest: sha256:fca4bcc9cbd49d9a15e0e4df6c666adf17776c950da9fa94a4f0a045d5c4ad33
Status: Downloaded newer image for jupyter/scipy-notebook:latest
Creating ubuntu_jupyter_1 ... done
Attaching to ubuntu_jupyter_1
jupyter_1  | Entered start.sh with args: jupyter lab
jupyter_1  | Running hooks in: /usr/local/bin/start-notebook.d as uid: 1000 gid: 100
jupyter_1  | Done running hooks in: /usr/local/bin/start-notebook.d
jupyter_1  | Running hooks in: /usr/local/bin/before-notebook.d as uid: 1000 gid: 100
jupyter_1  | Done running hooks in: /usr/local/bin/before-notebook.d
jupyter_1  | Executing the command: jupyter lab
jupyter_1  | [I 2024-10-21 10:04:20.621 ServerApp] Package jupyterlab took 0.0000s to import
jupyter_1  | [I 2024-10-21 10:04:20.656 ServerApp] Package jupyter_lsp took 0.0340s to import
jupyter_1  | [W 2024-10-21 10:04:20.656 ServerApp] A `_jupyter_server_extension_points` function
```

## *Step 4:*

The Jupyter server application installed successfully in the terminal, the copy the public ip address in the instance and paste in the web brower. The Jupyter web page will display…To Enter the Token id in the jupyter web page: In the EC2 Instance after docker compose we get the jupyter installed successful will token id copy that and paste onto it.



```
    To access the server, open this file in a browser:
        file:///home/jovyan/.local/share/jupyter/runtime/jpserver-7-open.html
    Or copy and paste one of these URLs:
        http://62f3bf41d0c9:8888/lab?token=6abd06b1c5326688fa4c17524bfa0d92f5aa188eed34665a
        http://127.0.0.1:8888/lab?token=6abd06b1c5326688fa4c17524bfa0d92f5aa188eed34665a
```



Token authentication is enabled

If no password has been configured, you need to open the server with its login token in the URL, or paste it above. This requirement will be lifted if you **enable a password**.

The command:

```
jupyter server list
```

will show you the URLs of running servers with their tokens, which you can copy and paste into your browser. For example:

```
Currently running servers:
http://localhost:8888/?token=c8de56fa... :: /Users/you/notebooks
```

or you can paste just the token value into the password field on this page.

See **the documentation on how to enable a password** in place of token authentication, if you would like to avoid dealing with random tokens.

Cookies are required for authenticated access to the Jupyter server.

Setup a Password

You can also setup a password by entering your token and a new password on the fields below:

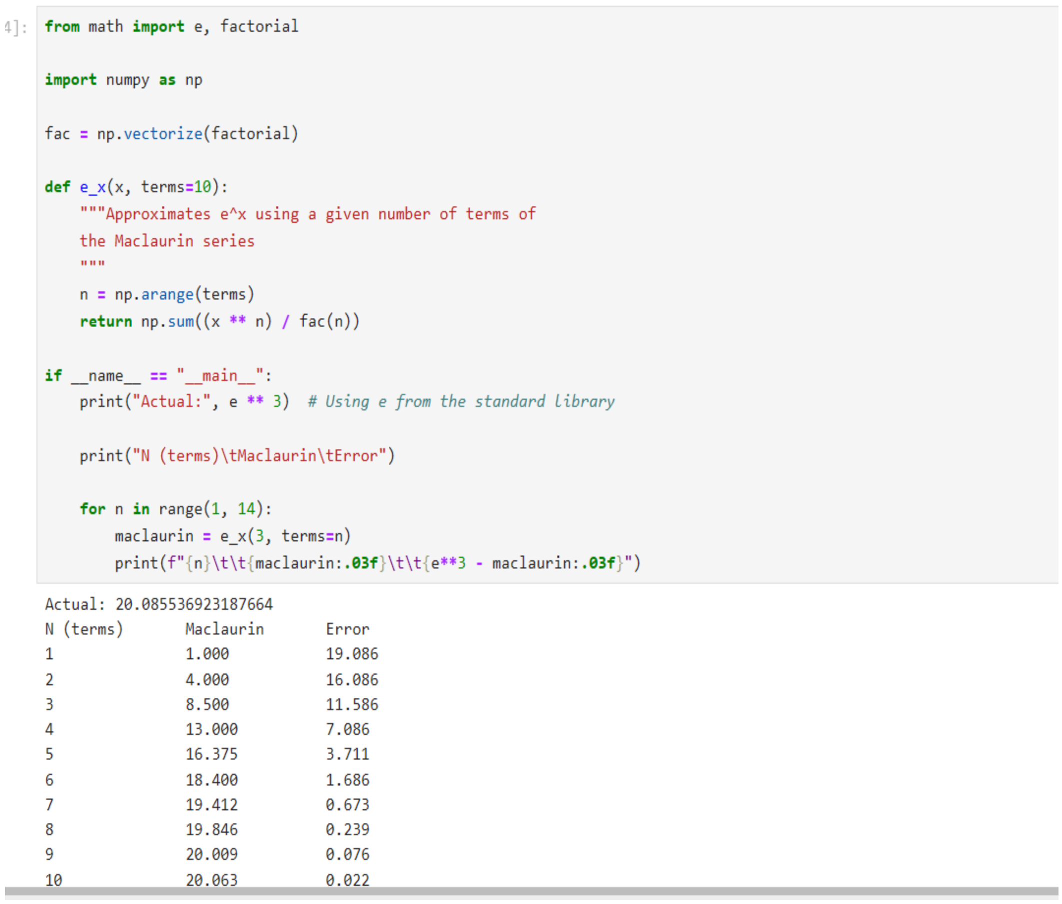
Token

***Step 5:***

In Jupyter environment we go to do data science using python libraries like Numpy and pandas

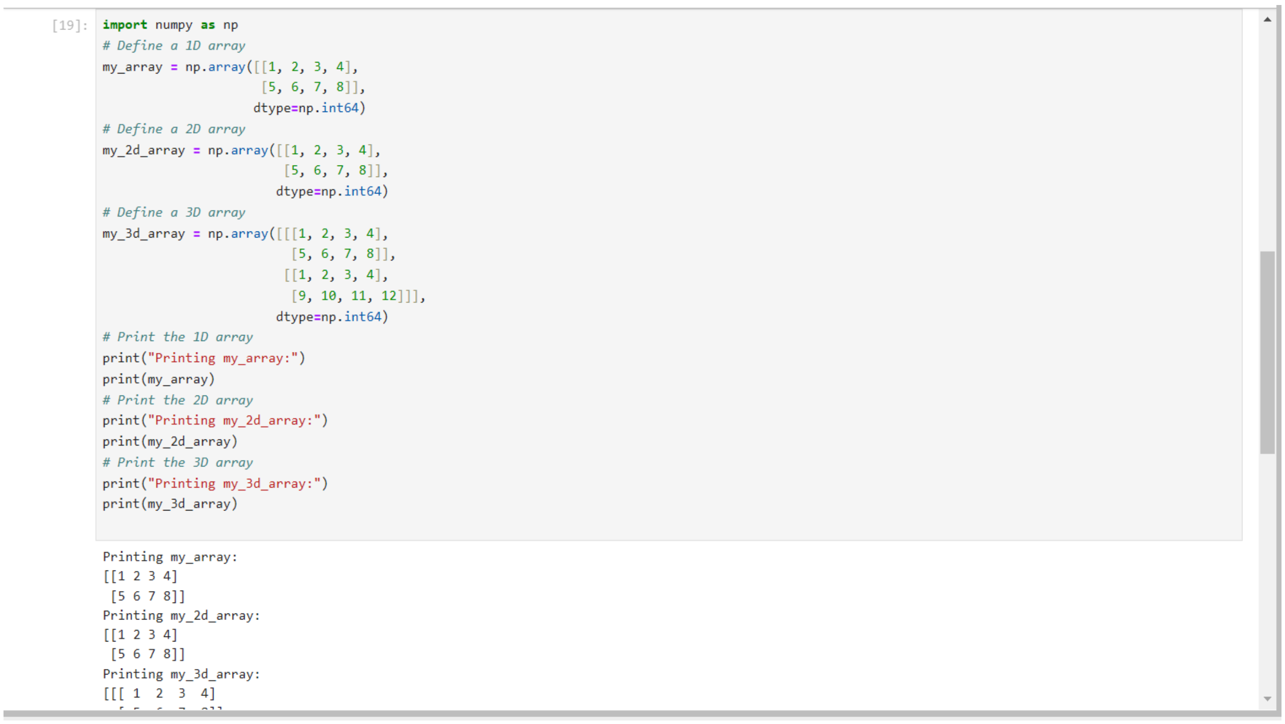
- **Numpy Library in python**

NumPy is a general-purpose array-processing Python library which provides handy methods/functions for working n-dimensional arrays. NumPy is a short form for “Numerical Python“. It provides various computing tools such as comprehensive mathematical functions, and linear algebra routines.



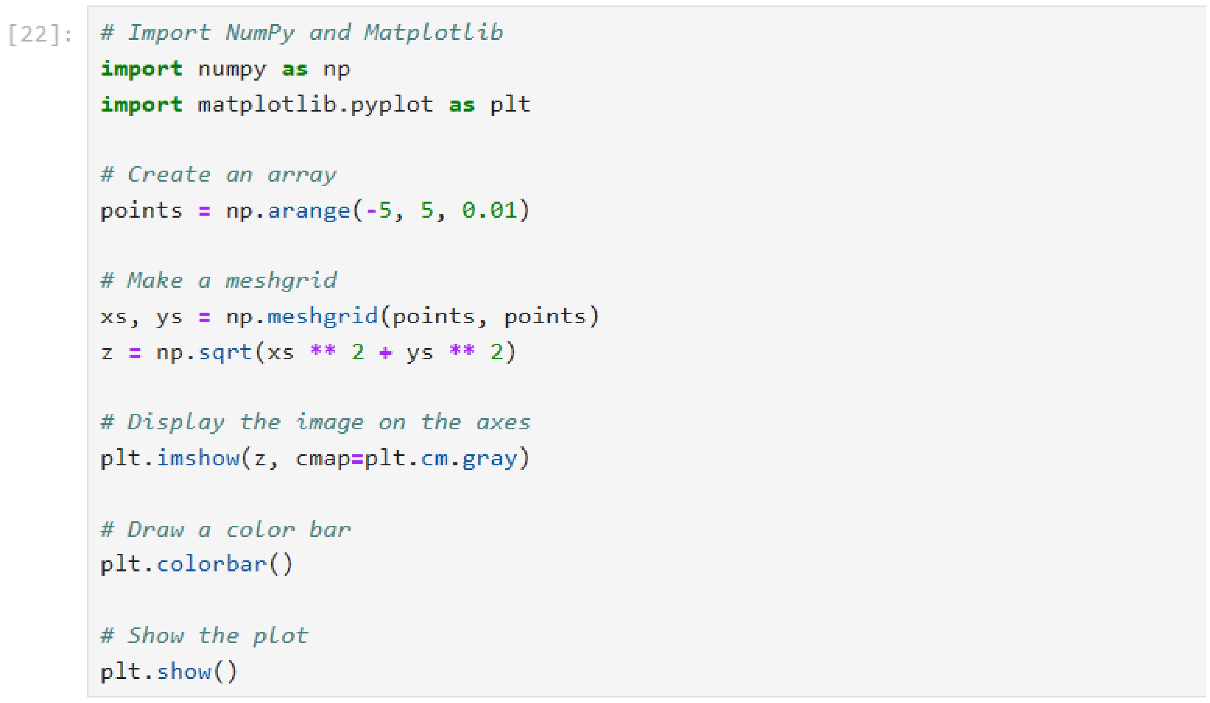
```
4]: from math import e, factorial

import numpy as np

fac = np.vectorize(factorial)

def e_x(x, terms=10):
    """Approximates e^x using a given number of terms of
    the Maclaurin series
    """
    n = np.arange(terms)
    return np.sum((x ** n) / fac(n))

if __name__ == "__main__":
    print("Actual:", e ** 3)  # Using e from the standard library

    print("N (terms)\tMaclaurin\tError")

    for n in range(1, 14):
        maclaurin = e_x(3, terms=n)
        print(f"{n}\t\t{maclaurin:.03f}\t\t{e**3 - maclaurin:.03f}")
```

```
Actual: 20.085536923187664
N (terms)       Maclaurin       Error
1               1.000           19.086
2               4.000           16.086
3               8.500           11.586
4               13.000          7.086
5               16.375          3.711
6               18.400          1.686
7               19.412          0.673
8               19.846          0.239
9               20.009          0.076
10              20.063          0.022
```

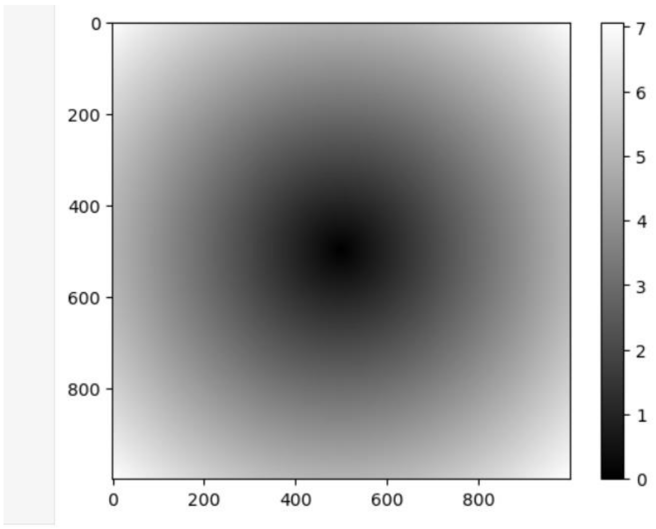


```
[19]: import numpy as np
      # Define a 1D array
      my_array = np.array([[1, 2, 3, 4],
                           [5, 6, 7, 8]],
                          dtype=np.int64)
      # Define a 2D array
      my_2d_array = np.array([[1, 2, 3, 4],
                              [5, 6, 7, 8]],
                             dtype=np.int64)
      # Define a 3D array
      my_3d_array = np.array([[[1, 2, 3, 4],
                               [5, 6, 7, 8]],
                              [[1, 2, 3, 4],
                               [9, 10, 11, 12]]],
                             dtype=np.int64)
      # Print the 1D array
      print("Printing my_array:")
      print(my_array)
      # Print the 2D array
      print("Printing my_2d_array:")
      print(my_2d_array)
      # Print the 3D array
      print("Printing my_3d_array:")
      print(my_3d_array)
```

```
Printing my_array:
[[1 2 3 4]
 [5 6 7 8]]
Printing my_2d_array:
[[1 2 3 4]
 [5 6 7 8]]
Printing my_3d_array:
[[[ 1  2  3  4]
```



```
[22]: # Import NumPy and Matplotlib
      import numpy as np
      import matplotlib.pyplot as plt

      # Create an array
      points = np.arange(-5, 5, 0.01)

      # Make a meshgrid
      xs, ys = np.meshgrid(points, points)
      z = np.sqrt(xs ** 2 + ys ** 2)

      # Display the image on the axes
      plt.imshow(z, cmap=plt.cm.gray)

      # Draw a color bar
      plt.colorbar()

      # Show the plot
      plt.show()
```



```
# Import `numpy` as `np`
import numpy as np
x = np.array([[1, 2, 3], [3, 4, 5]])
y = np.array([6,7,8])

# Add `x` and `y`
z = np.add(x,y)
print("Addition of x and y:\n", z)

# Subtract `x` and `y`
z = np.subtract(x,y)
print("Subtraction of y from x:\n", z)

# Multiply `x` and `y`
z = np.multiply(x,y)
print("Element-wise multiplication of x and y:\n", z)
```

```
Addition of x and y:
 [[ 7  9 11]
 [ 9 11 13]]
Subtraction of y from x:
 [[-5 -5 -5]
 [-3 -3 -3]]
Element-wise multiplication of x and y:
 [[ 6 14 24]
 [18 28 40]]
```

- **Pandas Library in Python**

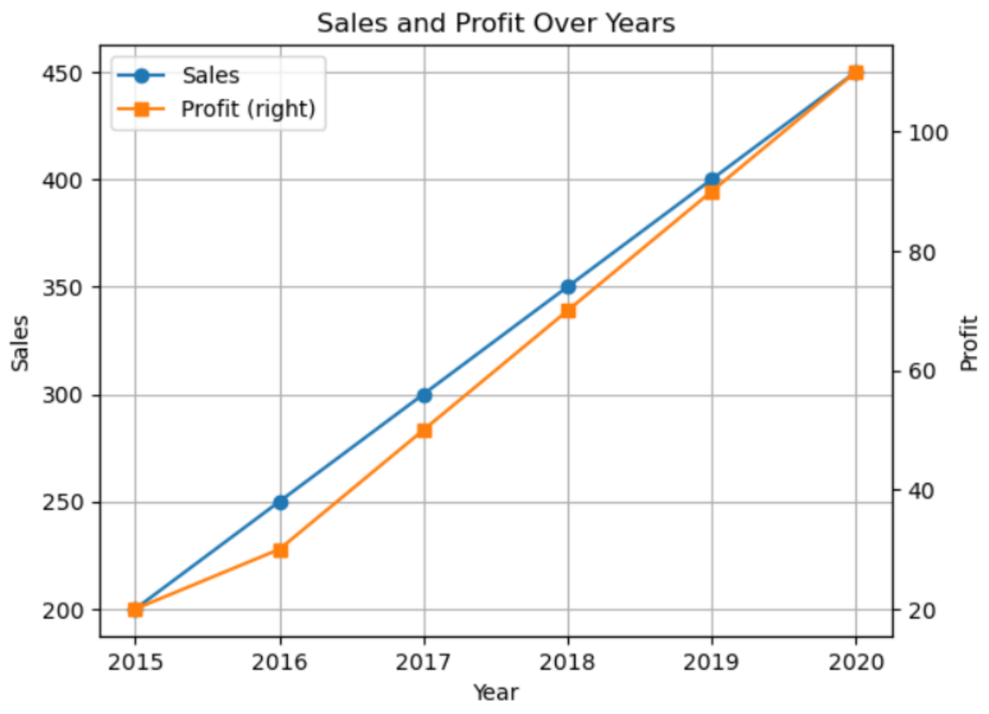
Pandas is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance & productivity for users.

[2]:
```python
# importing pandas module
import pandas as pd

# making data frame
data = pd.read_csv("https://media.geeksforgeeks.org/wp-content/uploads/nba.csv")

# calling head() method
# storing in new variable
data_top = data.head()

# display
data_top
```

[2]:

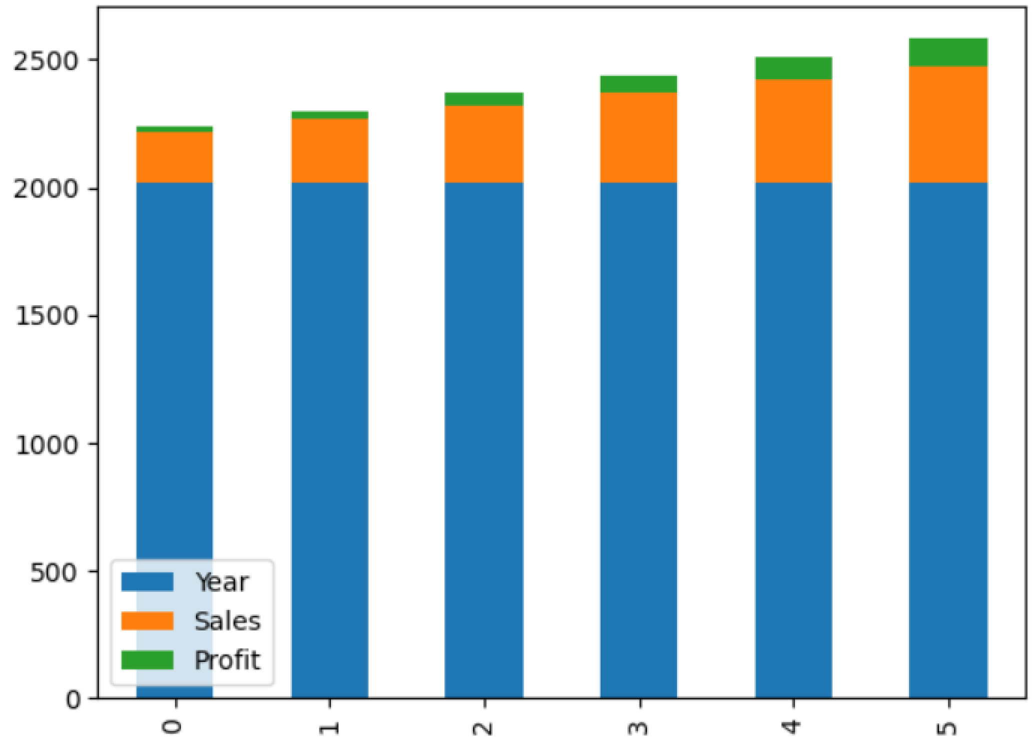
| | Name | Team | Number | Position | Age | Height | Weight | College | Salary |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Avery Bradley | Boston Celtics | 0.0 | PG | 25.0 | 6-2 | 180.0 | Texas | 7730337.0 |
| **1** | Jae Crowder | Boston Celtics | 99.0 | SF | 25.0 | 6-6 | 235.0 | Marquette | 6796117.0 |
| **2** | John Holland | Boston Celtics | 30.0 | SG | 27.0 | 6-5 | 205.0 | Boston University | NaN |
| **3** | R.J. Hunter | Boston Celtics | 28.0 | SG | 22.0 | 6-5 | 185.0 | Georgia State | 1148640.0 |
| **4** | Jonas Jerebko | Boston Celtics | 8.0 | PF | 29.0 | 6-10 | 231.0 | NaN | 5000000.0 |

```python
import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Year': [2015, 2016, 2017, 2018, 2019, 2020],
    'Sales': [200, 250, 300, 350, 400, 450],
    'Profit': [20, 30, 50, 70, 90, 110]
}

df = pd.DataFrame(data)

# Plotting both Sales and Profit over the years
ax = df.plot(x='Year', y='Sales', kind='line', marker='o', title='Sales and Profit Over Years')
df.plot(x='Year', y='Profit', kind='line', marker='s', ax=ax, secondary_y=True)

# Adding labels and grid
ax.set_ylabel('Sales')
ax.right_ax.set_ylabel('Profit')
ax.grid(True)

plt.show()
```



```python
df.plot.bar(stacked=True)
```

```
<Axes: >
```


Year
Sales
Profit