

## **Python Packages Installation**

### **Aim**

To download, install and explore the features of Numpy, Scipy, Jupyter, Statsmodel, and Pandas packages.

### **What is Python libraries?**

A Python library is a collection of related modules. It contains bundles of code that can be used repeatedly in different programs. It makes Python Programming simpler and convenient for the programmer.

A Python library is simply a collection of codes or modules of codes that we can use in a program for specific operations. We use libraries so that we don't need to write the code again in our program that is already available. But how it works.

### **What is NumPy?**

Python NumPy is a general-purpose array processing package that provides tools for handling n-dimensional arrays. It provides various computing tools such as comprehensive mathematical functions, linear algebra routines. NumPy provides both the flexibility of Python and the speed of well-optimized compiled C code. Its easy-to-use syntax makes it highly accessible and productive for programmers from any background.

Python is open source object oriented interpreted language. Of the many features, one of the important features that makes python a strong programming language is Python packages. A lot of external packages are written in python which you can be installed and used depending upon your requirement.

Python packages are nothing but directory of python scripts. Each script is a module which can be a function, methods or new python type created for particular functionality. numpy is one such important package created to ease array computation in python.

In this we will explain the process of downloading and installing numpy packages and how to use them in python environment on mac, windows, ubuntu and fedora operating systems. The basics of python programming language are not covered in this blog. For beginners, the basics of python programming language are covered in this Edureka blog.

All python packages are installed using pip – Package Installer for Python. You can view the details of all python packages and download them from Python Package Index (PyPI). However, pip is automatically installed when you download and install python from python.org or any other python integrated environment. Please read the blog for the best python integrated platforms which also provides loads of other functionalities. pip is the simplest way to download packages directly from PyPI from your command line.

### **NumPy Installation on Windows Operating System**

Python is not installed by default in windows operating system. You can download the required version of python from python.org. Once python is installed successfully, open command prompt and use pip to install numpy.

#### **Pre-requisites:**

The only thing that you need for installing Numpy on Windows are:

- Python
- PIP or Conda (depending upon user preference)

### **Installing Numpy on Windows:**

#### **Installing NumPy**

You can follow the steps outlined below and use the commands on most Linux, Mac, or Windows systems. Any irregularities in commands are noted along with instructions on how to modify them to your needs.

### Step 1: Check Python Version

Before you can install NumPy, you need to know which Python version you have. This programming language comes preinstalled on most operating systems (except Windows; you will need to install Python on Windows manually).

Most likely, you have Python 2 or Python 3 installed, or even both versions.

To check whether you have Python 2, run the command:

```
$ python -V
```

The output should give you a version number.

```
sofiya@sofiya-VirtualBox:~$ python -V
```

```
Python 2.7.17
```

```
sofiya@sofiya-VirtualBox:~$ python3 -V
```

```
Python 3.6.9
```

### Step 2: Install Pip

The easiest way to install NumPy is by using Pip. Pip a package manager for installing and managing Python software packages.

Unlike Python, Pip does not come preinstalled on most operating systems. Therefore, you need to set up the package manager that corresponds to the version of Python you have. If you have both versions of Python, install both Pip versions as well.

The commands below use the apt utility as we are installing on Ubuntu for the purposes of this article.

Install Pip (for Python 2) by running:

```
$ sudo apt install python-pip
```

```
sofiya@sofiya-VirtualBox:~$ pip -V
```

```
pip 9.0.1 from /usr/lib/python2.7/dist-packages (python 2.7)
```

```
sofiya@sofiya-VirtualBox:~$ pip3 -V
```

```
pip 9.0.1 from /usr/lib/python3/dist-packages (python 3.6)
```

### Step 3: Install NumPy

With Pip set up, you can use its command line for installing NumPy.

Install NumPy with Python 2 by typing:

```
$ pip install numpy
```

Pip downloads the NumPy package and notifies you it has been successfully installed.

```
sofiya@sofiya-VirtualBox:~$ pip install numpy
```

```
Collecting numpy
```

```
  Downloading https://files.pythonhosted.org/packages/3a/5f/47e578b3ae79e2624e205445ab77a1848acdaa2929a00eeef6b16eaaeb20/numpy-1.16.6-cp27-cp27mu-manylinux1_x86_64.whl (17.0MB)
```

```
100% |████████████████████████████████████████| 17.0MB 34kB/s
```

```
Installing collected packages: numpy
```

```
Successfully installed numpy-1.16.6
```

### Step 4: Verify NumPy Installation

Use the show command to verify whether NumPy is now part of your Python packages:

```
$ pip show numpy
```

```
Upgrading NumPy
```

If you already have NumPy and want to upgrade to the latest version, for Pip2 use the command:

```
$ pip install --upgrade numpy
```

## SciPy

### What is scipy?

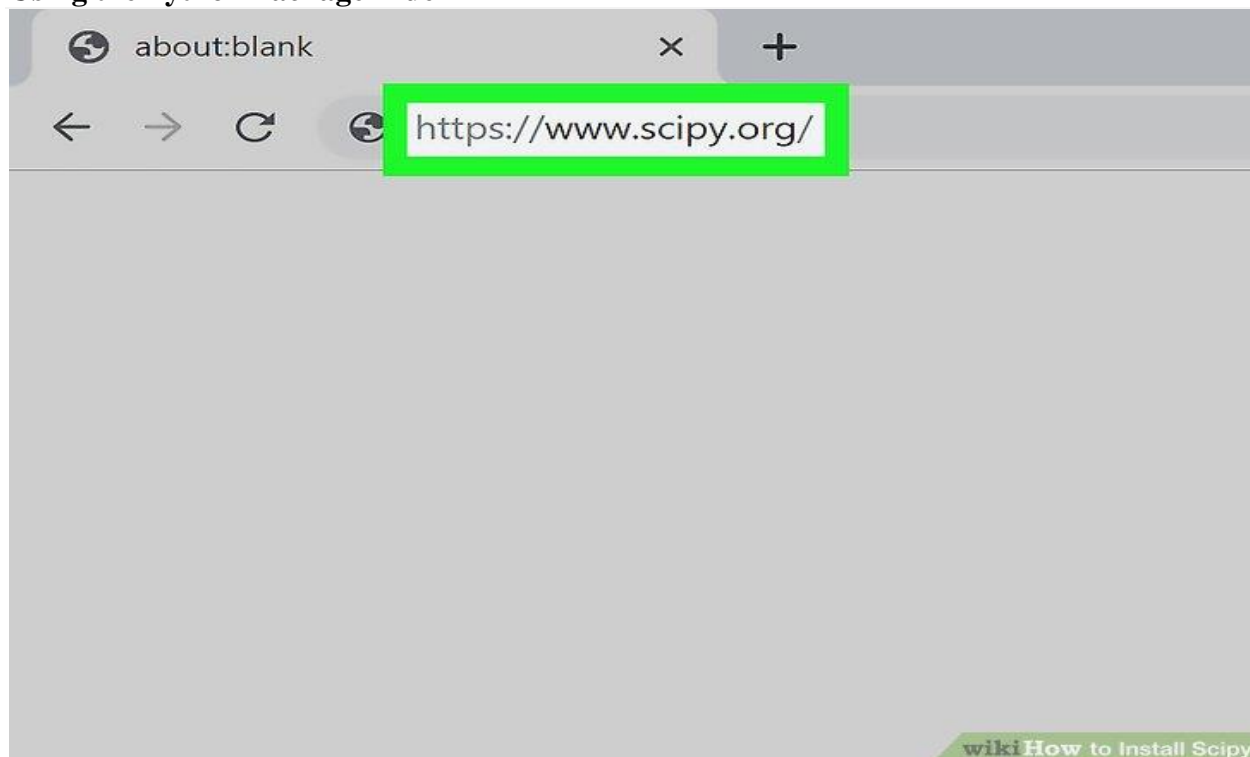
SciPy in Python is an open-source library used for solving mathematical, scientific, engineering, and technical problems. It allows users to manipulate the data and visualize the data using a wide range of high-level Python commands. SciPy is built on the Python NumPy extension.

### Why SciPy?

- SciPy contains varieties of sub packages which help to solve the most common issue related to Scientific Computation.
- SciPy package in Python is the most used Scientific library only second to GNU Scientific Library for C/C++ or Matlab's.
- Easy to use and understand as well as fast computational power.
- It can operate on an array of NumPy library.

To install the main SciPy packages from the SciPy library, using Windows, Mac or Linux. SciPy is a free and open-source Python library with packages optimized and developed for scientific and technical computing. If you have Python installed, you can use Python's standard pip package manager, and install it from the Python Package index.

### Using the Python Package Index

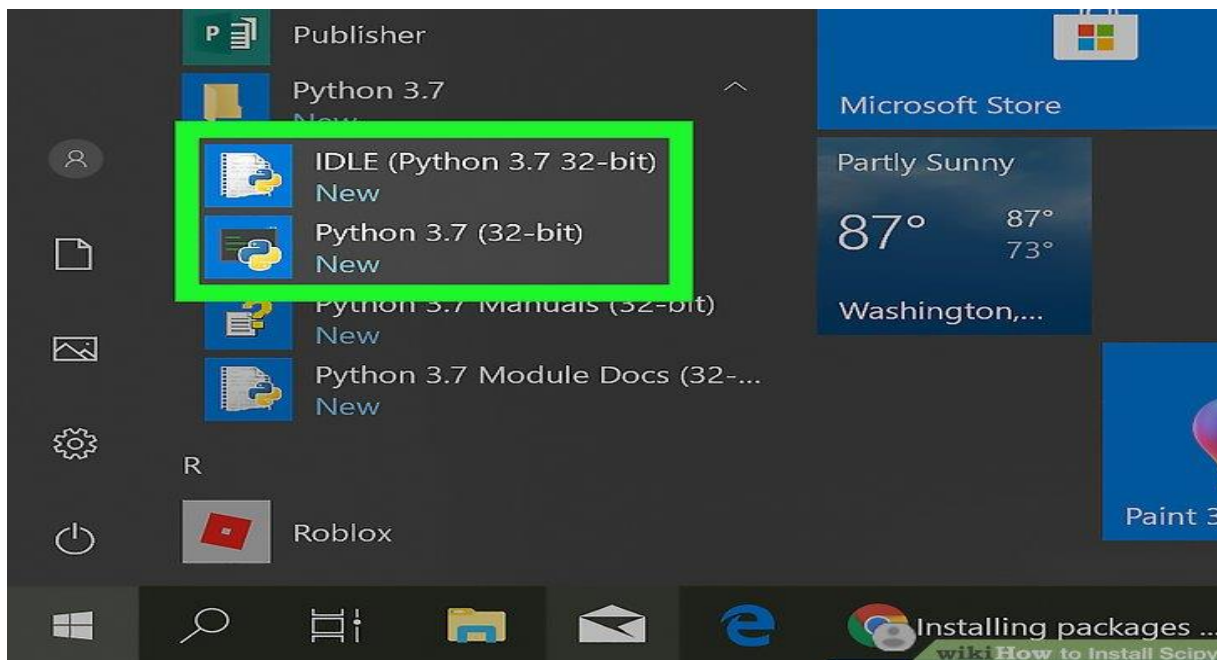


**Open the SciPy website in your internet browser.** Type or paste <https://www.scipy.org/> into the address bar, and press **Enter** or **Return** on your keyboard.



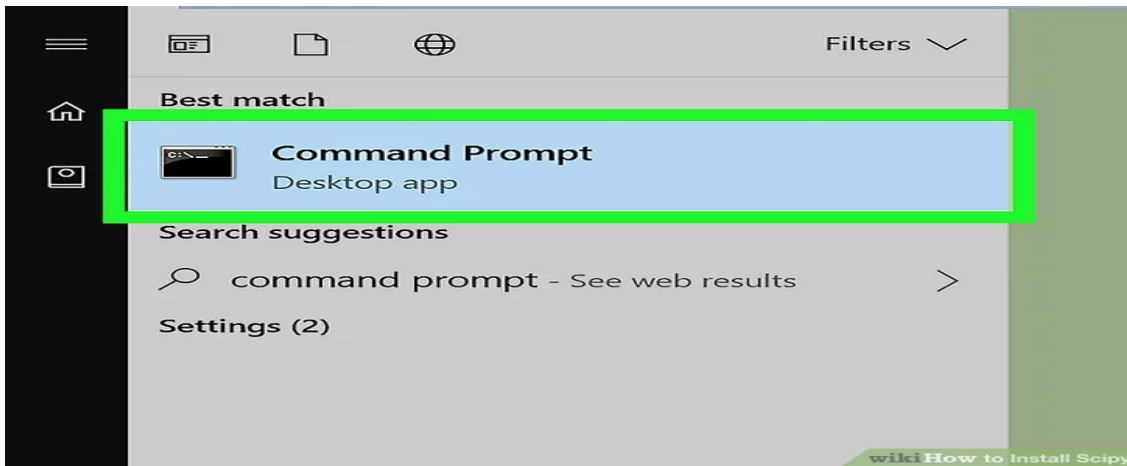
Click the **Install** button on the home page. This button looks like a downward green arrow on the blue-and-white SciPy icon. You can find it near the upper-left corner of the page.

- This will open the SciPy installation details on a new page.



**Make sure Python is installed on your computer.** SciPy is an open-source Python library, and requires a basic Python distribution installed on your system.

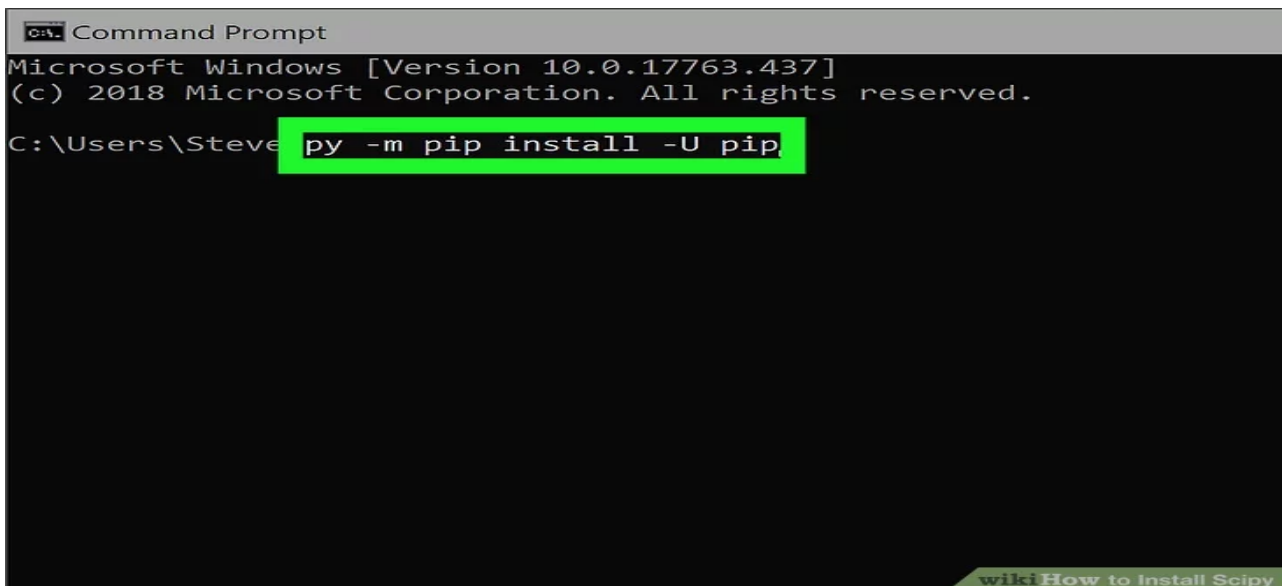
- If you don't have Python installed, you can select one of the recommended distributions under the "Scientific Python Distributions" heading, and install it to your computer.
- If you're not sure how to install Python, make sure to check out [this article](#) for detailed instructions on installing the core packages.



- **Open your computer's command prompt terminal.** You can open the Command Prompt on Windows, Terminal on Mac, or your distribution's Terminal on Linux.

**Type and run `python -m pip install -U pip`.** This command will make sure the latest pip files are installed on your system to handle package managing tasks.

- Press `↵` **Enter** or `↩` **Return** to run the command.



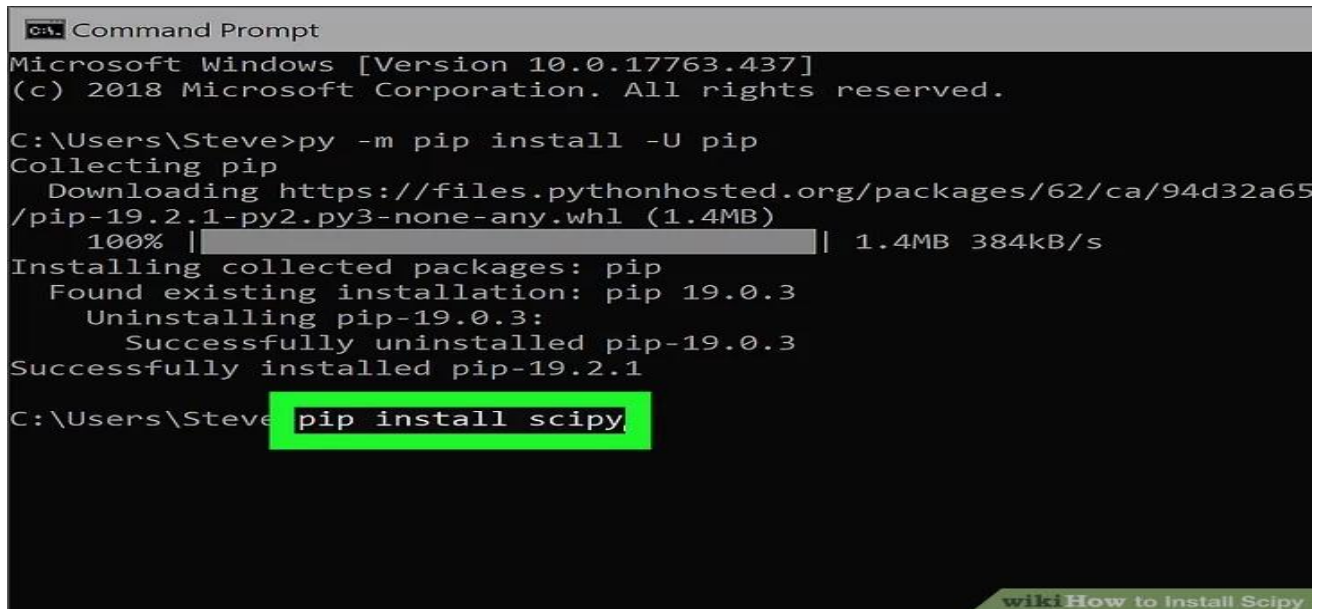
**Type and run `pip install scipy` in the command prompt.** This will use the Python Package index, and install the core SciPy packages on your computer.

- You can also install other core packages like Numpy and Matplotlib by using the `pip install numpy` and `pip install matplotlib` commands.

```
Command Prompt
Microsoft Windows [Version 10.0.17763.437]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Steve>py -m pip install -U pip
Collecting pip
  Downloading https://files.pythonhosted.org/packages/62/ca/94d32a65
/pip-19.2.1-py2.py3-none-any.whl (1.4MB)
    100% |#####| 1.4MB 384kB/s
Installing collected packages: pip
  Found existing installation: pip 19.0.3
  Uninstalling pip-19.0.3:
    Successfully uninstalled pip-19.0.3
  Successfully installed pip-19.2.1

C:\Users\Steve>pip install scipy
```



## Pandas

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis/manipulation tool available in any language.

Pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data need not be labeled at all to be placed into a pandas data structure

The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, DataFrame provides everything that R’s data.frame provides and much more. pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

### Python installation

Pandas dataframes are some of the most useful data structures available in any library. It has uses in every data-intensive field, including but not limited to scientific computing, data science, and machine learning. The library does not come included with a regular install of Python. To use it, you must install the Pandas framework separately.

### How to Install Python Pandas on Windows?

Before you install Pandas, you must bear in mind that it supports only Python versions 3.7, 3.8, and 3.9. Therefore, if you have not installed Python on your computer or have an older version of Python installed, you must install a version that supports Pandas on your computer.

### Installing with pip

It is a package installation manager that makes installing Python libraries and frameworks straightforward. As long as you have a newer version of Python installed (> Python 3.4), pip will be installed on your computer along with Python by default.

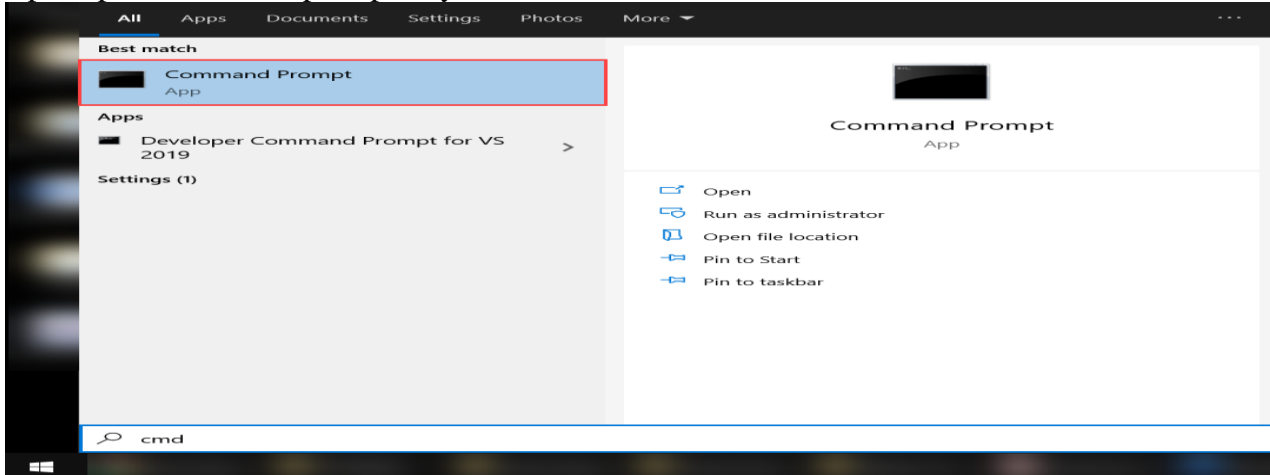


However, if you're using an older version of Python, you will need to install pip on your computer before installing Pandas. The easiest way to do this is to upgrade to the latest version of Python available on <https://www.python.org>.

### Step #1: Launch Command Prompt

Press the Windows key on your keyboard or click on the Start button to open the start menu. Type "cmd," and the Command Prompt app should appear as a listing in the start menu.

Open up the command prompt so you can install Pandas.



### Step #2: Enter the Required Command

After you launch the command prompt, the next step in the process is to type in the required command to initialize pip installation.

Enter the command "pip install pandas" on the terminal. This should launch the pip installer. The required files will be downloaded, and Pandas will be ready to run on your computer.

```
ca Command Prompt
Microsoft Windows [Version 10.0.19043.1083]
(c) Microsoft Corporation. All rights reserved.

c:\Users\>pip install pandas
Collecting pandas
  Downloading https://files.pythonhosted.org/packages/cb/e3/c0bc0f1b3835564f69094135de105a3def2eeb2689338a906bfc659c99d0/pandas-1.3.0-cp38-cp38-win_amd64.whl (10.2MB)
    | 10.2MB 3.3MB/s
Collecting pytz>=2017.3 (from pandas)
  Downloading https://files.pythonhosted.org/packages/70/94/784178ca5dd892a98f113cdd923372024dc04b8d40abe77ca76b5fb90ca6/pytz-2021.1-py2.py3-none-any.whl (510kB)
    | 512kB 6.4MB/s
Collecting python-dateutil>=2.7.3 (from pandas)
  Downloading https://files.pythonhosted.org/packages/36/7a/87837f39d0296e723bb9b62bbb257d0355c7f6128853c78955f57342a56d/python_dateutil-2.8.2-py2.py3-none-any.whl (247kB)
    | 256kB ...
Collecting numpy>=1.17.3 (from pandas)
  Downloading https://files.pythonhosted.org/packages/df/22/b74e5cedee1e3f108c986bd0b75600997d8b25def334a68f08d372db523/numpy-1.21.0-cp38-cp38-win_amd64.whl (14.0MB)
    | 14.0MB 2.2MB/s
Collecting six>=1.5 (from python-dateutil>=2.7.3->pandas)
  Downloading https://files.pythonhosted.org/packages/d9/5a/e7c31adbe875f2abbb91bd84cf2dc52d792b5a01506781dbc925c91daf11/six-1.16.0-py2.py3-none-any.whl
Installing collected packages: pytz, six, python-dateutil, numpy, pandas
Successfully installed numpy-1.21.0 pandas-1.3.0 python-dateutil-2.8.2 pytz-2021.1 six-1.16.0
WARNING: You are using pip version 19.2.3, however version 21.1.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

## Jupyter

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports:

Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

The Jupyter Notebook is not included with Python, so if you want to try it out, you will need to install Jupyter.

There are many distributions of the Python language.. The most popular is CPython, which is the reference version of Python that you can get from their website. It is also assumed that you are using Python 3.

### Installation

If so, then you can use a handy tool that comes with Python called **pip** to install Jupyter Notebook like this:

```
$ pip install jupyter
```

### Starting the Jupyter Notebook Server

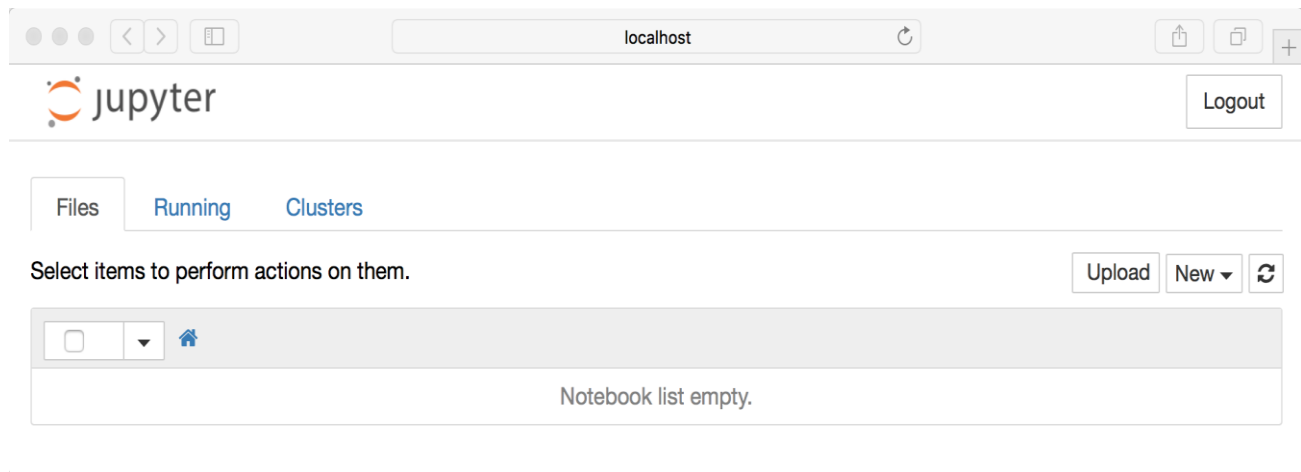
Now that you have Jupyter installed, let's learn how to use it. To get started, all you need to do is open up your terminal application and go to a folder of your choice. I recommend using something like your Documents folder to start out with and create a subfolder there called Notebooks or something else that is easy to remember.

Then just go to that location in your terminal and run the following command:

```
$ jupyter notebook
```

This will start up Jupyter and your default browser should start (or open a new tab) to the following URL: <http://localhost:8888/tree>

Your browser should now look something like this:



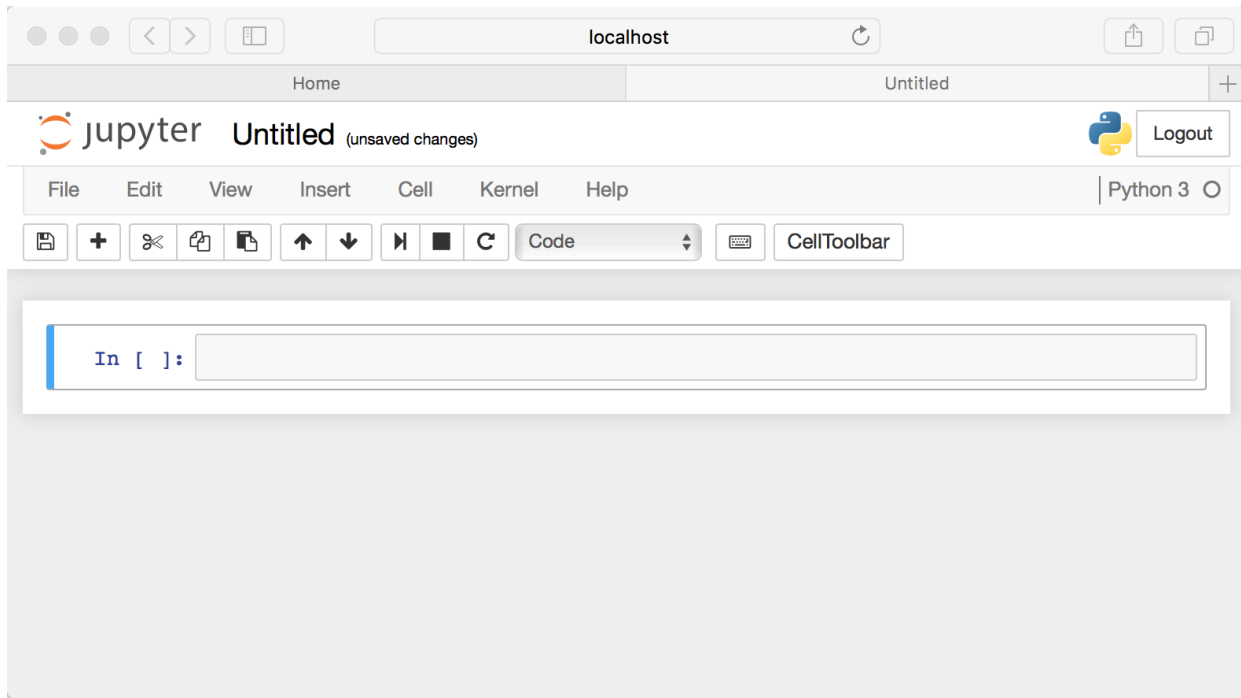
### Creating a Notebook

Now that you know how to start a Notebook server, you should probably learn how to create an actual Notebook document.

All you need to do is click on the New button (upper right), and it will open up a list of choices. On my machine, I happen to have Python 2 and Python 3 installed, so I can create a Notebook that uses either of these. For simplicity's sake, let's choose Python 3.

Your web page should now look like this:

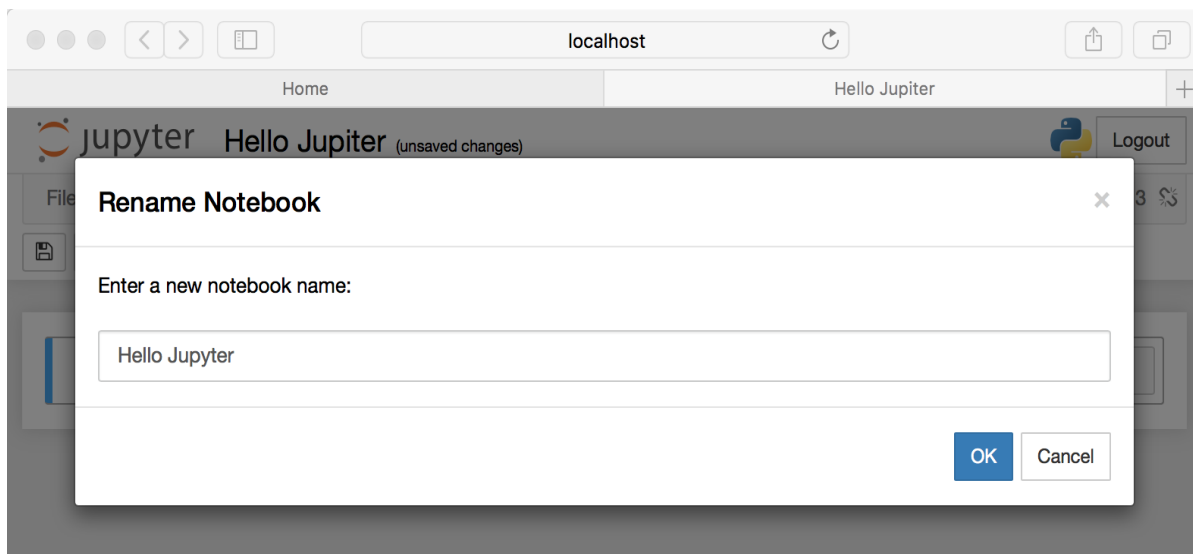




## Naming

You will notice that at the top of the page is the word *Untitled*. This is the title for the page and the name of your Notebook. Since that isn't a very descriptive name, let's change it!

Just move your mouse over the word *Untitled* and click on the text. You should now see an in-browser dialog titled *Rename Notebook*. Let's rename this one to *Hello Jupyter*:



## Running Cells

A Notebook's cell defaults to using code whenever you first create one, and that cell uses the kernel that you chose when you started your Notebook.

In this case, you started yours with Python 3 as your kernel, so that means you can write Python code in your code cells. Since your initial Notebook has only one empty cell in it, the Notebook can't really do anything.

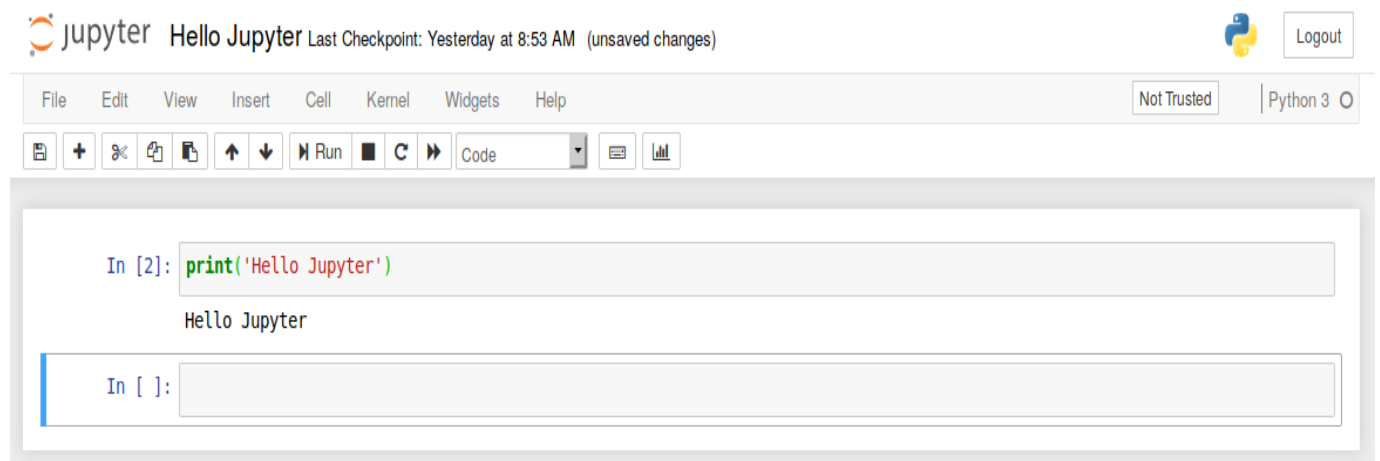
Thus, to verify that everything is working as it should, you can add some Python code to the cell and try running its contents.

Let's try adding the following code to that cell:

```
print('Hello Jupyter!')
```

Running a cell means that you will execute the cell's contents. To execute a cell, you can just select the cell and click the *Run* button that is in the row of buttons along the top. It's towards the middle. If you prefer using your keyboard, you can just press **Shift + Enter**.

When I ran the code above, the output looked like this:



If you have multiple cells in your Notebook, and you run the cells in order, you can share your variables and imports across cells. This makes it easy to separate out your code into logical chunks without needing to reimport libraries or recreate variables or functions in every cell.

When you run a cell, you will notice that there are some square braces next to the word *In* to the left of the cell. The square braces will auto fill with a number that indicates the order that you ran the cells. For example, if you open a fresh Notebook and run the first cell at the top of the Notebook, the square braces will fill with the number *1*

## Statsmodels

Statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. An extensive list of result statistics are available for each estimator. The results are tested against existing statistical packages to ensure that they are correct. The package is released under the open source Modified BSD (3-clause) license. The online documentation is hosted at [statsmodels.org](https://statsmodels.org).

As its name implies, statsmodels is a Python library built specifically for statistics. Statsmodels is built on top of NumPy, SciPy, and matplotlib, but it contains more advanced functions for statistical testing and modeling that you won't find in numerical libraries like NumPy or SciPy.

Some of the essential features of this package are-

1. It includes various models of linear regression like ordinary least squares, generalized least squares, weighted least squares, etc.
2. It provides some efficient functions for time series analysis.
3. It also has some datasets for examples and testing.
4. Models based on survival analysis are also available.
5. All the statistical tests that we can imagine for data on a large scale are present.

## Installing statsmodels

Let's have a look at the steps of installing statsmodels in Python-

1. Checking the version of Python installed in our PCs, we have discussed this already in the previous articles but let's talk about this again-  
There are two ways to check the version of Python in Windows-
  - *Using Powershell*
  - *Using Command Prompt*

## Using PowerShell

Follow the below steps to check the version of Python using PowerShell.

1. Click 'Win+R' or type 'Run' on the taskbar's search pane.
2. Type 'Powershell'
3. A window will appear on your screen named 'Windows Powershell'
4. Click on 'Enter'
5. Type *python -version* and click on 'Enter'
6. The version would be displayed in the next line.

## Using Command Prompt

Type 'Command Prompt' on the taskbar's search pane and you'll see its icon. Click on it to open the command prompt.

Also, you can directly click on its icon if it is pinned on the taskbar.

1. Once the 'Command Prompt' screen is visible on your screen.
2. Type *python -version* and click on 'Enter'.
3. The version installed in your system would be displayed in the next line.

## Result:

Thus, the python libraries Numpy, pandas, statsmodels, jupyter are downloaded and installed successfully.

Ex.No : 02

## **NumPy Arrays**

### **Aim:**

To write the python programs to establish various NumPy array operations like, Attributes, Indexing, Slicing, Reshaping, Concatenation and Splitting.

## **Array Attribute's**

### **Aim:**

To write a python program to generate an array and show its attributes like dimension, shape and size.

### **Algorithm:**

1. Start the python program.
2. Import the numpy library as np
3. Generate an array by using randint method.
4. For 2 and 3 dimensional arrays mention the number of rows and columns in size ( ).
5. Print the dimension, size, and shape of an array separately.
6. Stop the program.

### **Program:**

```
import numpy as np
np.random.seed(0)
x1 = np.random.randint(10, size=6)
x2 = np.random.randint(10, size=(3, 4))
x3 = np.random.randint(10, size=(3, 4, 5))
print("x1 ndim: ", x1.ndim)
print("x2 shape:", x2.shape)
print("x3 size: ", x3.size)
```

### **Output:**

```
x1 ndim: 1
x2 shape: (3, 4)
x3 size: 60
```

## Array Indexing

### Aim:

To write a python program to perform array indexing concept.

### Algorithm:

1. Start the python program.
2. Import the NumPy package as np.
3. Generate an array using random values.
4. Apply various indexing methods to identify elements in an array.
5. Apply negative indexing to refer element in an array from right to left.
6. Stop the program.

### Program: Single dimensional array

```
import numpy as np
x1 = np.random.randint(10, size=6)
print('array')
print(x1)
print('select an element from an array')
print(x1[0])
print('select an element by negative index')
print(x1[-3])
```

### Output:

```
array
[5 9 3 0 5 0]
select an element from an array
5
select an element by negative index
0
```

### Program: Multi-dimensional array

```
import numpy as np
x1 = np.random.randint(12, size=(3,4))
print('array')
print(x1)
print('select an element from an array')
print(x1[1,2])
```

### Output:

```
array
[[ 0 10 2 11]
 [10 7 11 2]
 [ 9 2 3 11]]
select an element from an array
11
```



## Array Slicing

### Aim:

To write a python program to perform slicing operation in an array.

### Algorithm:

1. Start the python program.
2. Import the NumPy library as np.
3. Generate an array.
4. Slice the array to specific no of starting elements or element after the specified no of elements or specified number of middle elements.
5. Slice the array with step of elements.
6. Stop the program.

### Program: Slicing single dimensional array

```
import numpy as np
print('A sequence Array')
x = np.arange(10)
print(x)
print('Slice first 3 values')
print(x[:3])
print('Slice elements after index 7')
print(x[7:])
print('slice middle values')
print(x[4:7])
print('slice every second element')
print(x[::2])
```

### Output:

```
A sequence Array
[0 1 2 3 4 5 6 7 8 9]
Slice first 3 values
[0 1 2]
Slice elements after index 7
[7 8 9]
slice middle values
[4 5 6]
Slice every second element
[0 2 4 6 8]
```

### Program: Slicing multi-dimensional array

```
import numpy as np
x = np.random.randint(10, size=(3, 4))
print('Array',x2)
print('Slice two rows and three columns')
print(x[:2,:3])
print('Slice three rows and two columns')
print(x[:3,::2])
print('Slice a particular value')
print(x[1,:1])
```

**Output:**

```
Array [[3 5 2 4]
       [7 6 8 8]
       [1 6 7 7]]
```

Slice two rows and three columns

```
[[3 2 0]
```

```
[8 3 8]]
```

Slice three rows and two columns

```
[[3 0]
```

```
[8 8]
```

```
[8 3]]
```

Slice a particular value

```
[[3]]
```

## Array Reshaping

### Aim:

To write a python program to reshape the given array.

### Algorithm:

1. Start the python program.
2. Import the NumPy array as np.
3. Generate an array.
4. Reshape the single dimensional array into multi-dimensional array.
5. Stop the program.

### Program:

```
import numpy as np
x=np.arange(1,10)
print('Array',x)
x1=x.reshape(3,3)
print('Reshaped Array')
print(x1)
```

### Output:

```
Array [1 2 3 4 5 6 7 8 9]
Reshaped Array
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

## Array Concatenation and Splitting

### Aim:

To write a python program to concatenate two arrays into a single array and to split a single array into small arrays.

### Algorithm:

1. Start the python program.
2. Import the NumPy as np
3. Generate two arrays x and y.
4. Concatenate the x and y as z.
5. Print the z array.
6. Split the z into x1,x2. The splitting point is index 3
7. Print the split arrays.
8. Stop the program.

### Program:

```
import numpy as np
x=np.array([1,2,3,4,5])
y=np.array([6,7,8,9,10])
print('X=',x)
print('Y=',y)
print('Array Concatenation')
z=np.concatenate([x,y])
print('Z=',z)
print('Array Splitting')
x1,x2=np.split(z,[3])
print('X1=',x1)
print('X2=',x2)
```

### Output:

```
X= [1 2 3 4 5]
Y= [ 6 7 8 9 10]
Array Concatenation
Z= [ 1 2 3 4 5 6 7 8 9 10]
Array Splitting
X1= [1 2 3]
X2= [ 4 5 6 7 8 9 10]
```

### Result:

Thus, the program for various array operations are written and executed successfully.

Ex.No. : 03

## **Pandas**

### **Aim:**

Write the python program to perform various actions with pandas Series and Data Frames.

### **Pandas – Series**

### **Aim:**

To write the python program to create and perform various actions in Pandas series.

### **Algorithm:**

1. Start the python program.
2. Import the pandas library as pd.
3. Create the series as pd.Series() and perform various actions.
4. Stop the program.

### **Program:**

#### **Series with default index.**

```
import pandas as pd
x=pd.Series([10,20,30,40,50])
print(x)
```

### **Output:**

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

#### **Series with explicit index**

```
import pandas as pd
x=pd.Series([10,20,30,40,50],index=['a','b','c','d','e'])
print(x)
```

### **Output:**

```
a    10
b    20
c    30
d    40
e    50
dtype: int64
```

#### **Series as specialized dictionary and display single and group of data's**

```
import pandas as pd
marks = {'Ram':80,
        'Kasim':90,
        'Faizal':85,
        'Tamil':95,
        'Sai':92}
result = pd.Series(marks)
```

```

print(result)
print('Display a single data in series')
print(result['Ram'])
print('Display a set of data in series')
print(result['Kasim':'Tamil'])

```

**Output:**

```

Ram    80
Kasim  90
Faizal 85
Tamil  95
Sai    92
dtype: int64
Display a single data in series
80
Display a set of data in series
Kasim  90
Faizal 85
Tamil  95
dtype: int64

```

## Pandas – Data Frame

**Aim:**

To write the python program to create and perform various actions in Pandas Data Frame.

**Algorithm:**

1. Start the python program.
2. Import the pandas library as pd.
3. Create the series as pd.DataFrame() and perform various actions.
4. Stop the program.

**Program: simple data frame**

```

import pandas as pd
data = {"Roll No": [100, 101, 102],
        "Percentage": [95, 80, 75]}
df = pd.DataFrame(data)
print(df)

```

**Output:**

	Roll No	Percentage
0	100	95
1	101	80
2	102	75

**Locate Row**

```
import pandas as pd
```



```
data = {"Roll No": [100, 101, 102],
        "Percentage": [95, 80, 75]}
df = pd.DataFrame(data)
print(df.loc[1])
```

### Output:

```
Roll No 101
Percentage 80
Name: 1, dtype: int64
```

### Named Indexes

```
import pandas as pd
data = {
    "Roll No": [100, 101, 102],
    "Percentage": [95, 80, 75]}
df = pd.DataFrame(data, index = ["AAA", "BBB", "CCC"])
print(df)
print('Locate a named index')
print(df.loc['BBB'])
```

### Output:

```
      Roll No Percentage
AAA      100          95
BBB      101          80
CCC      102          75
Locate a named index
Roll No 101
Percentage 80
Name: BBB, dtype: int64
```

### Checking for missing values using isnull() and notnull() :

```
import pandas as pd
import numpy as np
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}
df = pd.DataFrame(dict)
df.isnull()
```

### Output:

	First Score	Second Score	Third Score
0	False	False	True
1	False	False	False
2	True	False	False
3	False	True	False

### Filling missing values using fillna()

```
import pandas as pd
import numpy as np
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}
df = pd.DataFrame(dict)
df.fillna(0)
```

### Output:

	First Score	Second Score	Third Score
0	100.0	30.0	0.0
1	90.0	45.0	40.0
2	0.0	56.0	80.0
3	95.0	0.0	98.0

### Drop null values

```
import pandas as pd
import numpy as np
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}
df = pd.DataFrame(dict)
df.dropna()
```

### Output:

	First Score	Second Score	Third Score
1	90.0	45.0	40.0

### String functions

#### Convert upper case to lower case

```
import pandas as pd
import numpy as np
s = pd.Series(['X', 'Y', 'Z', 'Aaba', 'Baca', np.nan, 'CABA', None, 'bird', 'horse', 'dog'])
print("Original series:")
print(s)
print("\nConvert all string values to upper case:")
print(s.str.upper())
print("\nConvert all string values to lower case:")
print(s.str.lower())
print("\nLength of the string values:")
print(s.str.len())
```

Output:

**Output**

Original series:

```
0    X
1    Y
2    Z
3    Aaba
4    Baca
5    NaN
6    CABA
7    None
8    bird
9    horse
10   dog
dtype: object
```

Convert all string values to upper case:

```
0    X
1    Y
2    Z
3    AABA
4    BACA
5    NaN
6    CABA
7    None
8    BIRD
9    HORSE
10   DOG
dtype: object
```

Convert all string values to lower case:

```
0    x
1    y
2    z
3    aaba
4    baca
5    NaN
6    caba
7    None
8    bird
9    horse
10   dog
dtype: object
```

Length of the string values of the said Series:

```
0    1.0
1    1.0
2    1.0
3    4.0
4    4.0
```

```
5    NaN
6    4.0
7    NaN
8    4.0
9    5.0
10   3.0
dtype: float64
```

### **Remove whitespaces, left sided whitespaces and right sided whitespaces of the string values of a given pandas series**

```
import pandas as pd
color1 = pd.Index([' Green', 'Black ', ' Red ', 'White', ' Pink '])
print("Original series:")
print(color1)
print("\nRemove whitespace")
print(color1.str.strip())
print("\nRemove left sided whitespace")
print(color1.str.lstrip())
print("\nRemove Right sided whitespace")
print(color1.str.rstrip())
```

#### **Output:**

```
Original series:
Index([' Green', 'Black ', ' Red ', 'White', ' Pink '], dtype='object')

Remove whitespace
Index(['Green', 'Black', 'Red', 'White', 'Pink'], dtype='object')

Remove left sided whitespace
Index(['Green', 'Black ', 'Red ', 'White', 'Pink '], dtype='object')

Remove Right sided whitespace
Index([' Green', 'Black', ' Red', 'White', ' Pink'], dtype='object')
```

### **Pandas Joining and merging DataFrame along rows**

```
import pandas as pd
student_data1 = pd.DataFrame({
    'student_id': ['S1', 'S2', 'S3', 'S4', 'S5'],
    'name': ['AAA', 'BBB', 'CCC', 'DDD', 'EEE'],
    'marks': [200, 210, 190, 222, 199]})

student_data2 = pd.DataFrame({
    'student_id': ['S4', 'S5', 'S6', 'S7', 'S8'],
    'name': ['FFF', 'GGG', 'HHH', 'III', 'JJJ'],
    'marks': [201, 200, 198, 219, 201]})
```

```

print("Original DataFrames:")
print(student_data1)
print("-----")
print(student_data2)
print("\nJoin the said two dataframes along rows:")
result_data = pd.concat([student_data1, student_data2])
print(result_data)

```

### Output:

Original DataFrames:

	student_id	name	marks
0	S1	AAA	200
1	S2	BBB	210
2	S3	CCC	190
3	S4	DDD	222
4	S5	EEE	199

---

	student_id	name	marks
0	S4	FFF	201
1	S5	GGG	200
2	S6	HHH	198
3	S7	III	219
4	S8	JJJ	201

Join the said two dataframes along rows:

	student_id	name	marks
0	S1	AAA	200
1	S2	BBB	210
2	S3	CCC	190
3	S4	DDD	222
4	S5	EEE	199
0	S4	FFF	201
1	S5	GGG	200
2	S6	HHH	198
3	S7	III	219
4	S8	JJJ	201

### To join the two given data frames along columns

```

import pandas as pd
student_data1 = pd.DataFrame({
    'student_id': ['S1', 'S2', 'S3', 'S4', 'S5'],
    'name': ['AAA', 'BBB', 'CCC', 'DDD', 'EEE'],
    'marks': [200, 210, 190, 222, 199]})

student_data2 = pd.DataFrame({
    'student_id': ['S4', 'S5', 'S6', 'S7', 'S8'],
    'name': ['FFF', 'GGG', 'HHH', 'III', 'JJJ'],
    'marks': [201, 200, 198, 219, 201]})

```

```

print("Original DataFrames:")
print(student_data1)
print("-----")
print(student_data2)
print("\nJoin the said two dataframes along rows:")
result_data = pd.concat([student_data1, student_data2],axis = 1)
print(result_data)

```

### Output:

Original DataFrames:

	student_id	name	marks
0	S1	AAA	200
1	S2	BBB	210
2	S3	CCC	190
3	S4	DDD	222
4	S5	EEE	199

---

	student_id	name	marks
0	S4	FFF	201
1	S5	GGG	200
2	S6	HHH	198
3	S7	III	219
4	S8	JJJ	201

Join the said two dataframes along rows:

	student_id	name	marks	student_id	name	marks
0	S1	AAA	200	S4	FFF	201
1	S2	BBB	210	S5	GGG	200
2	S3	CCC	190	S6	HHH	198
3	S4	DDD	222	S7	III	219
4	S5	EEE	199	S8	JJJ	201

### To append rows to an existing Data Frame and display the combined data.

```

import pandas as pd
student_data1 = pd.DataFrame({
    'student_id': ['S1', 'S2', 'S3', 'S4', 'S5'],
    'name': ['AAA', 'BBB', 'CCC', 'DDD', 'EEE'],
    'marks': [200, 210, 190, 222, 199]})

s6=pd.Series(['S6','FFF',205],index=['student_id','name','marks'])

print("Original DataFrames:")
print(student_data1)
print("\n New Row(s)")
print(s6)
combined_data=student_data1.append(s6,ignore_index=True)
print("\n Combined data:")
print(combined_data)

```

### Output:



Original DataFrames:

	student_id	name	marks
0	S1	AAA	200
1	S2	BBB	210
2	S3	CCC	190
3	S4	DDD	222
4	S5	EEE	199

New Row(s)  
student\_id S6  
name FFF  
marks 205  
dtype: object

Combined data:

	student_id	name	marks
0	S1	AAA	200
1	S2	BBB	210
2	S3	CCC	190
3	S4	DDD	222
4	S5	EEE	199
5	S6	FFF	205

**Result:**

Thus, the programs to exhibit various Pandas features are written and executed successfully.

Ex. No.: 04

## Reading Data from Text, Excel and Descriptive Analysis in Data Set

### Aim:

To write python program to read the data from text file, excel file and from web exploring various commands on doing descriptive analysis on iris data set.

### Algorithm:

1. Start the python program.
2. Create text file, excel file and CSV file with relevant data.
3. Read the data from text file by using `pd.read_txt()` method.
4. Install packages like `xlrd`, `xlwt` and `openpyxl` to read data from excel file.
5. Read the data from the excel file by using `pd.read_excel()` method.
6. Read the data from data set and CSV file by using `pd.read_csv()` method.
7. Download the Iris data set.
8. Perform various analysis in Iris data set.
9. Visualize the results of the analysis by various plotting methods.
10. Stop the program

### Programs:

#### To read data from excel file.

```
import pandas as pd
df = pd.read_excel("D:\mark.xls")
print(df)
```

### Output:

	s.no	rollno	name	sub1	sub2	sub3
0	1	100	a	91	89	87
1	2	101	b	95	95	89
2	3	102	c	92	94	87
3	4	103	d	91	92	91
4	5	104	e	89	81	92
5	6	105	f	87	94	94
6	7	106	g	89	93	92

#### Add a column in excel file

```
import pandas as pd
df = pd.read_excel("D:\mark.xls")
df['total']=df['sub1']+df['sub2']+df['sub3']
df.to_excel("D:\mark.xls")
print(df)
```

### Output:

	Unnamed: 0	s.no	rollno	name	sub1	sub2	sub3	total
0	0	1	100	a	91	89	87	267
1	1	2	101	b	95	95	89	279
2	2	3	102	c	92	94	87	273
3	3	4	103	d	91	92	91	274
4	4	5	104	e	89	81	92	262
5	5	6	105	f	87	94	94	275
6	6	7	106	g	89	93	92	274

### Export the excel file to csv file

```
import pandas as pd
d=pd.read_excel("D:\mark.xls")
d.to_csv("D:\mark.csv")
df=pd.DataFrame(pd.read_csv("D:\mark.csv"))
print(df)
```

### Output

	Unnamed: 0.2	Unnamed: 0.1	Unnamed: 0	s.no	...	sub1	sub2	sub3	total
0	0	0	0	1	...	91	89	87	267
1	1	1	1	2	...	95	95	89	279
2	2	2	2	3	...	92	94	87	273
3	3	3	3	4	...	91	92	91	274
4	4	4	4	5	...	89	81	92	262
5	5	5	5	6	...	87	94	94	275
6	6	6	6	7	...	89	93	92	274

[7 rows x 10 columns]

### Convert Excel file to text file.

```
import pandas as pd
df=pd.read_csv("D:\mark.csv")
df.to_csv("D:\mark.txt")
print(df)
```

### Output

,Unnamed: 0.2,Unnamed: 0.1,Unnamed: 0,s.no,rollno,name,sub1,sub2,sub3,total  
0,0,0,0,1,100,a,91,89,87,267  
1,1,1,1,2,101,b,95,95,89,279  
2,2,2,2,3,102,c,92,94,87,273  
3,3,3,3,4,103,d,91,92,91,274  
4,4,4,4,5,104,e,89,81,92,262  
5,5,5,5,6,105,f,87,94,94,275  
6,6,6,6,7,106,g,89,93,92,274

## Programs on iris data set

### Program to print top 5 rows - df.head()

```
import pandas as pd
df = pd.read_csv("D:\iris.csv")
print(df.head())
```

### Output

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa

3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

### Program to print last 5 rows – df.tail()

```
import pandas as pd
df = pd.read_csv("D:\iris.csv")
print(df.tail())
```

### Output

	sepal.length	sepal.width	petal.length	petal.width	variety
145	6.7	3.0	5.2	2.3	Virginica
146	6.3	2.5	5.0	1.9	Virginica
147	6.5	3.0	5.2	2.0	Virginica
148	6.2	3.4	5.4	2.3	Virginica
149	5.9	3.0	5.1	1.8	Virginica

### Quick statistical summary of the dataset - df.describe()

```
import pandas as pd
df = pd.read_csv("D:\iris.csv")
print(df.describe())
```

### Output

	sepal.length	sepal.width	petal.length	petal.width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

### Checking Missing Values - isnull()

```
import pandas as pd
df = pd.read_csv('D:\iris.csv')
print(df.isnull().sum())
```

### Output

```
sepal.length    0
sepal.width     0
petal.length    0
petal.width     0
variety         0
dtype: int64
```

## Checking Duplicates - drop\_duplicates()

```
import pandas as pd
df = pd.read_csv('D:\iris.csv')
data = df.drop_duplicates(subset="variety",)
print(data)
```

### Output

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
50	7.0	3.2	4.7	1.4	Versicolor
100	6.3	3.3	6.0	2.5	Virginica

## Count - Series.value\_counts()

```
import pandas as pd
df = pd.read_csv('D:\iris.csv')
print(df.value_counts("variety"))
```

### Output

```
variety
Setosa    50
Versicolor 50
Virginica  50
dtype: int64
```

## Display a specific column only

```
import pandas as pd
df = pd.read_csv('D:\iris.csv')
data = df[["sepal.length"]]
print(data)
```

### Output

	sepal.length
0	5.1
1	4.9
2	4.7
3	4.6
4	5.0
..	...
145	6.7
146	6.3
147	6.5
148	6.2
149	5.9

[150 rows x 1 columns]

### Calculating sum, mean and mode of a particular column.

```
import pandas as pd
df = pd.read_csv('D:\iris.csv')
sum_data = df["sepal.length"].sum()
mean_data = df["sepal.length"].mean()
median_data = df["sepal.length"].median()
print("Sum:",sum_data, "\nMean:", mean_data, "\nMedian:",median_data)
```

### Output

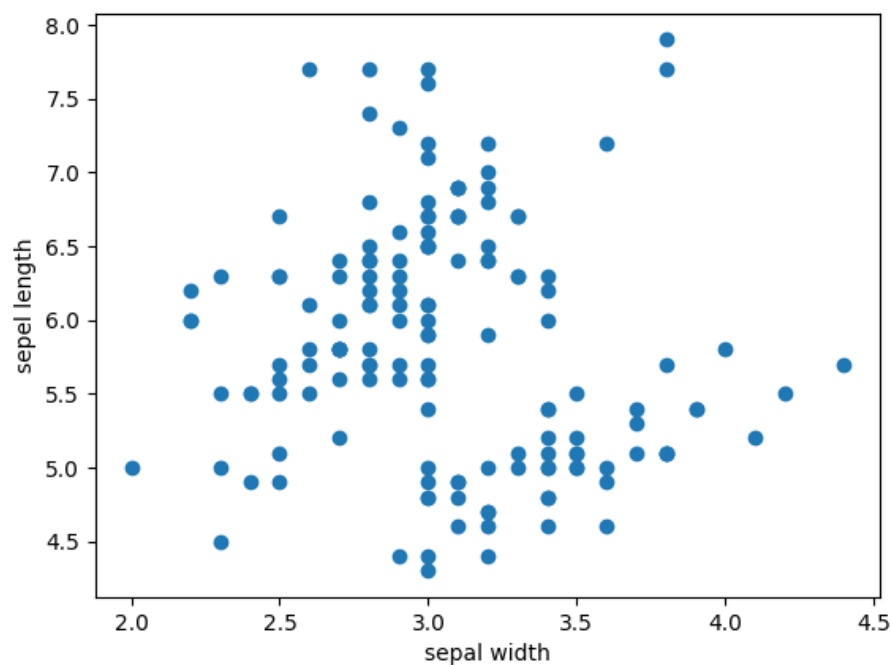
```
Sum: 876.5
Mean: 5.8433333333333334
Median: 5.8
```

### Data Visualization

#### Scatter plot of the iris data set (sepal width, sepal length)

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("D:\iris.csv")
plt.scatter(df["sepal.width"], df["sepal.length"])
plt.xlabel("sepal width")
plt.ylabel("sepal length")
plt.show()
```

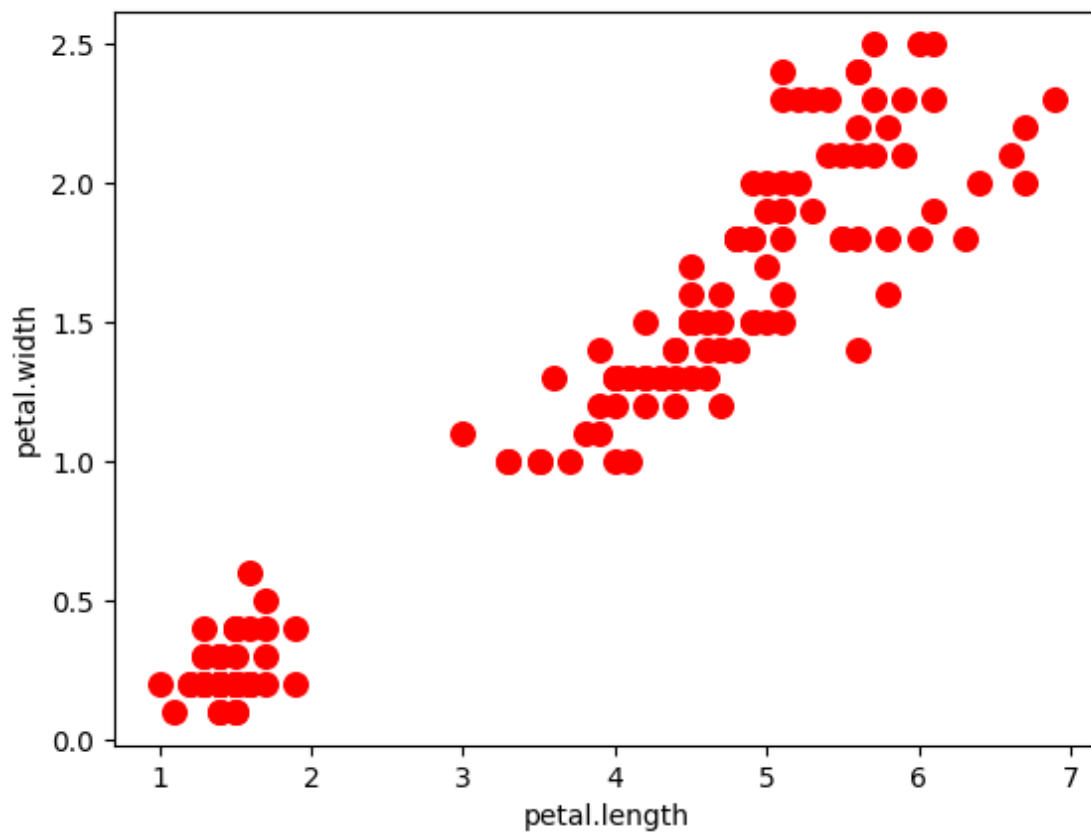
### Output



### Scatter plot of the iris data set (petal width, petal length)

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("D:\iris.csv")
df.plot(kind="scatter", x="petal.length", y="petal.width", color="red", s=70 )
plt.show()
```

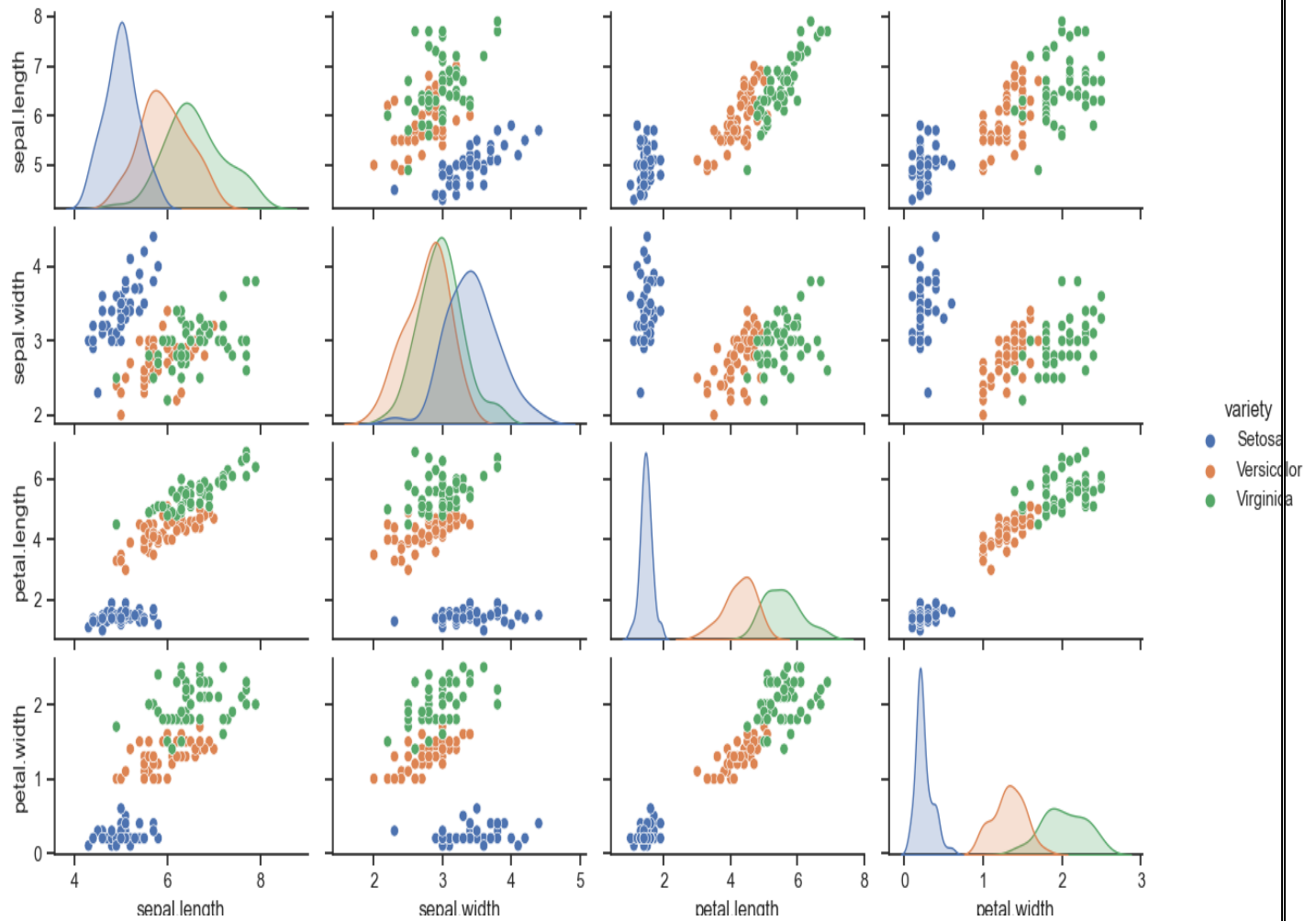
### Output



### Plot the entire values of the iris dataset using pairplot()

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
sns.set(style="ticks", color_codes=True)
df=pd.read_csv('D:\iris.csv')
g=sns.pairplot(df,hue='variety')
plt.show()
```

## Output



## Result:

Thus, the programs to read data from text file, excel file and csv file are written and executed successfully.



### Univariant, Bivariant and regression Analysis

#### Aim:

To write a python program to read Pima Indian Diabetes data set and perform various univariant, bivariant, linear regression and multi regression.

#### Algorithm:

1. Start the python program.
2. Download the Pima Indian Diabetes data set from UCI repository.
3. Read the dataset by using `pd.read_csv()` method.
4. Calculate frequency, mean, median, variance, standard deviation, correlation coefficient by using taking various columns.
5. Display the calculated values.
6. Stop the program.

#### 5.a. Univariant Analysis

(Mean, Median, Variance, Standard Deviation, Skewness, Kurtosis)

##### Read dataset

```
import pandas as pd
df=pd.read_csv("D:\diabetes.csv")
print(df)
```

##### Output

DataFrame:

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	tested_positive
1	1	85	66	29	0	26.6	0.351	31	tested_negative
2	8	183	64	0	0	23.3	0.672	32	tested_positive
3	1	89	66	23	94	28.1	0.167	21	tested_negative
4	0	137	40	35	168	43.1	2.288	33	tested_positive
..	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	tested_negative
764	2	122	70	27	0	36.8	0.340	27	tested_negative
765	5	121	72	23	112	26.2	0.245	30	tested_negative
766	1	126	60	0	0	30.1	0.349	47	tested_positive
767	1	93	70	31	0	30.4	0.315	23	tested_negative

[768 rows x 9 columns]

##### Read shape

```
import pandas as pd
df=pd.read_csv("D:\diabetes.csv")
print(df.shape)
```

##### Output

(768, 9)

##### Read data types

```
import pandas as pd
df=pd.read_csv("D:\diabetes.csv")
print(df.dtypes)
```

## Output

```
preg    int64
plas    int64
pres    int64
skin    int64
insu    int64
mass    float64
pedi    float64
age     int64
class   object
dtype: object
```

### 5.a1 Program to find frequency in diabetes dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv('D:\diabetes.csv')
freq_table = pd.crosstab(data['plas'], 'no_of_result')
print(freq_table)
```

## Output

```
col_0  no_of_result
plas
0       5
44      1
56      1
57      2
61      1
...      ...
195     2
196     3
197     4
198     1
199     1
```

[136 rows x 1 columns]

### Program to find Mean, Median, Variance, Standard Deviation

```
import pandas as pd
df = pd.read_csv('D:\diabetes.csv')
mean1 = df['plas'].mean()
sum1 = df['plas'].sum()
max1 = df['plas'].max()
min1 = df['plas'].min()
count1 = df['plas'].count()
median1 = df['plas'].median()
std1 = df['plas'].std()
var1 = df['plas'].var()
print("Mean=", mean1)
print("Sum=", sum1)
```

```

print("Maximum=",max1)
print("Minimum",min1)
print("Count=",count1)
print("Median=",median1)
print("Standard Devation=",std1)
print("Variance=",var1)

```

### Output:

```

Mean= 120.89453125
Sum= 92847
Maximum= 199
Minimum 0
Count= 768
Median= 117.0
Standard Devation= 31.97261819513622
Variance= 1022.2483142519557

```

### Program to find Mode

```

import pandas as pd
df = pd.read_csv ('D:\diabetes.csv')
mode1=df['plas'].mode()
print("Mode=",mode1)

```

### Output

```

Mode= 0    99
1    100
Name: plas, dtype: int64

```

### Program to find skew ness

```

import pandas as pd
df=pd.read_csv("D:\diabetes.csv")
dataframe=pd.DataFrame(data=df)
skewvalue=dataframe.skew(axis=1)
print("DataFrame:")
print(dataframe)
print("Skew:")
print(skewvalue)

```

### Output

```

DataFrame:
   preg  plas  pres  skin  insu  mass  pedi  age  class
0     6   148    72    35     0   33.6  0.627  50  tested_positive
1     1    85    66    29     0   26.6  0.351  31  tested_negative
2     8   183    64     0     0   23.3  0.672  32  tested_positive
3     1    89    66    23    94   28.1  0.167  21  tested_negative
4     0   137    40    35   168   43.1  2.288  33  tested_positive
..    ..   ...   ...   ...   ...   ...   ...   ...   ...
763  10   101    76    48   180   32.9  0.171  63  tested_negative
764   2   122    70    27     0   36.8  0.340  27  tested_negative
765   5   121    72    23   112   26.2  0.245  30  tested_negative

```

766	1	126	60	0	0	30.1	0.349	47	tested_positive
767	1	93	70	31	0	30.4	0.315	23	tested_negative

[768 rows x 9 columns]

Skew:

```
0    1.527988
1    0.841074
2    2.213061
3    0.531780
4    1.205725
```

...

```
763    1.168927
764    1.413922
765    0.751268
766    1.482404
767    0.988313
```

Length: 768, dtype: float64

### Program to find Kurtosis

```
import pandas as pd
import numpy as np
df=pd.read_csv("D:\diabetes.csv")
dataframe=pd.DataFrame(data=df)
kurt=dataframe.kurt(axis=1)
print("DataFrame:")
print(dataframe)
print("Kurtosis:")
print(kurt)
```

### Output

DataFrame:

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	tested_positive
1	1	85	66	29	0	26.6	0.351	31	tested_negative
2	8	183	64	0	0	23.3	0.672	32	tested_positive
3	1	89	66	23	94	28.1	0.167	21	tested_negative
4	0	137	40	35	168	43.1	2.288	33	tested_positive
..	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	tested_negative
764	2	122	70	27	0	36.8	0.340	27	tested_negative
765	5	121	72	23	112	26.2	0.245	30	tested_negative
766	1	126	60	0	0	30.1	0.349	47	tested_positive
767	1	93	70	31	0	30.4	0.315	23	tested_negative

[768 rows x 9 columns]

Kurtosis:

```
0    2.597033
1   -0.317557
2    5.134624
```

```

3    -1.531771
4     0.150578
...
763    1.651766
764    1.775973
765   -1.170875
766    2.086562
767   -0.031148
Length: 768, dtype: float64

```

### program to find skewness and kurtosis of a data set. (Column Wise)

```

import pandas as pd
data=pd.read_csv("D:\diabetes.csv")
df=pd.DataFrame(data)
print(df)
print("Skewness")
print(df.skew())
print("Kurtoisis")
print(df.kurtosis())

```

### Output

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	tested_positive
1	1	85	66	29	0	26.6	0.351	31	tested_negative
2	8	183	64	0	0	23.3	0.672	32	tested_positive
3	1	89	66	23	94	28.1	0.167	21	tested_negative
4	0	137	40	35	168	43.1	2.288	33	tested_positive
..	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	tested_negative
764	2	122	70	27	0	36.8	0.340	27	tested_negative
765	5	121	72	23	112	26.2	0.245	30	tested_negative
766	1	126	60	0	0	30.1	0.349	47	tested_positive
767	1	93	70	31	0	30.4	0.315	23	tested_negative

[768 rows x 9 columns]

Skewness

```

preg    0.901674
plas    0.173754
pres   -1.843608
skin    0.109372
insu    2.272251
mass   -0.428982
pedi    1.919911
age     1.129597
dtype: float64

```

Kurtoisis

```

preg    0.159220
plas    0.640780
pres    5.180157

```

```
skin -0.520072
insu  7.214260
mass  3.290443
pedi  5.594954
age   0.643159
dtype: float64
```

## 5.b Bivariant Analysis (Linear and Logistic regression modeling)

### Program to find correlations (based on a single column)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('D:\diabetes.csv')
correlations = df.corr()
print(correlations['plas'])
```

### Output

```
preg 0.129459
plas 1.000000
pres 0.152590
skin 0.057328
insu 0.331357
mass 0.221071
pedi 0.137337
age 0.263514
Name: plas, dtype: float64
```

### Program to find correlations (for entire data set)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('D:\diabetes.csv')
correlations = df.corr()
print(correlations)
```

### Output:

	preg	plas	pres	skin	insu	mass	pedi \
preg	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523
plas	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337
pres	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265
skin	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928

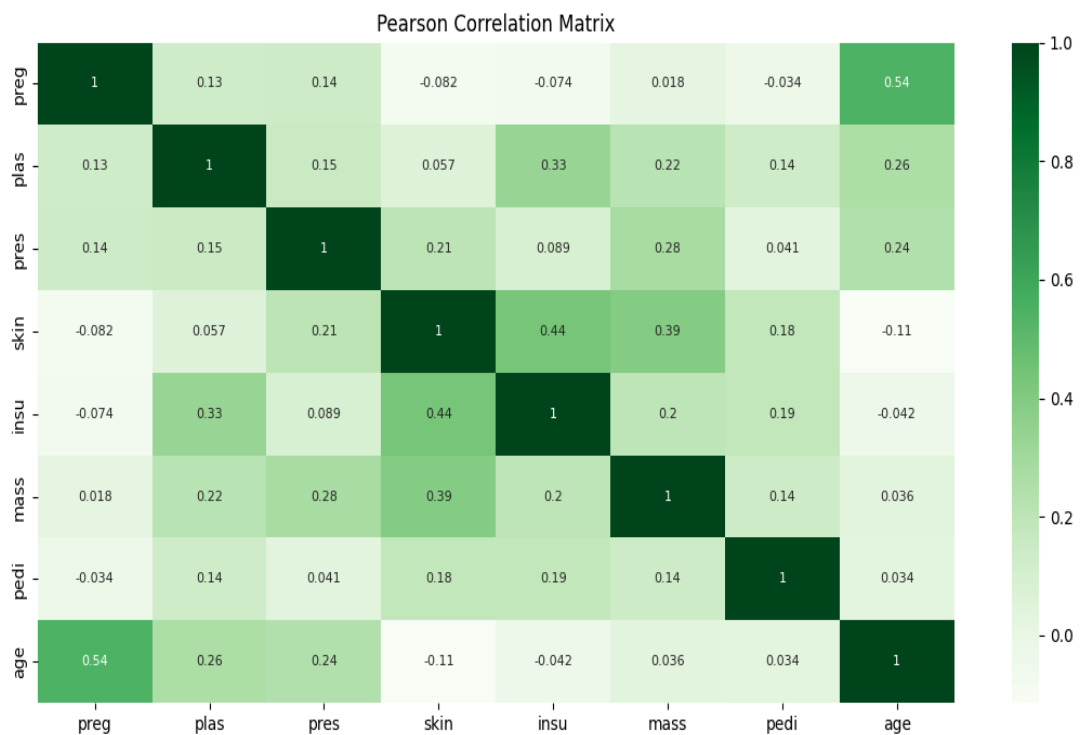
insu	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071
mass	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647
pedi	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000
age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561

	age
preg	0.544341
plas	0.263514
pres	0.239528
skin	-0.113970
insu	-0.042163
mass	0.036242
pedi	0.033561
age	1.000000

### program to generate Pearson correlation matrix

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('D:\diabetes.csv')
correlations = df.corr()
fig_1 = plt.figure(figsize=(12, 10))
sns.heatmap(correlations, annot=True, cmap='Greens', annot_kws={'size': 8})
plt.title('Pearson Correlation Matrix')
plt.show()
```

### Output:



### Program to find high correlated columns

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('D:\diabetes.csv')
correlations = df.corr()
highly_correlated_features = correlations[correlations > 0.75]
print(highly_correlated_features.fillna('-'))
```

### Output

	preg	plas	pres	skin	insu	mass	pedi	age
preg	1.0	-	-	-	-	-	-	-
plas	-	1.0	-	-	-	-	-	-
pres	-	-	1.0	-	-	-	-	-
skin	-	-	-	1.0	-	-	-	-
insu	-	-	-	-	1.0	-	-	-
mass	-	-	-	-	-	1.0	-	-
pedi	-	-	-	-	-	-	1.0	-
age	-	-	-	-	-	-	-	1.0

### Program for linear regression

```
import numpy as np
import matplotlib.pyplot as plt
def estimate_coef(x, y):
    n = np.size(x)
    m_x = np.mean(x)
    m_y = np.mean(y)
    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x
    return (b_0, b_1)

def plot_regression_line(x, y, b):
    plt.scatter(x, y, color = "m", marker = "o", s = 30)
    y_pred = b[0] + b[1]*x
    plt.plot(x, y_pred, color = "g")
    plt.xlabel('x')
    plt.ylabel('y')
    plt.show()
def main():
    x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
    y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])
    b = estimate_coef(x, y)
    print("Estimated coefficients:\nb_0 = {} \nb_1 = {}".format(b[0], b[1]))
    plot_regression_line(x, y, b)
```



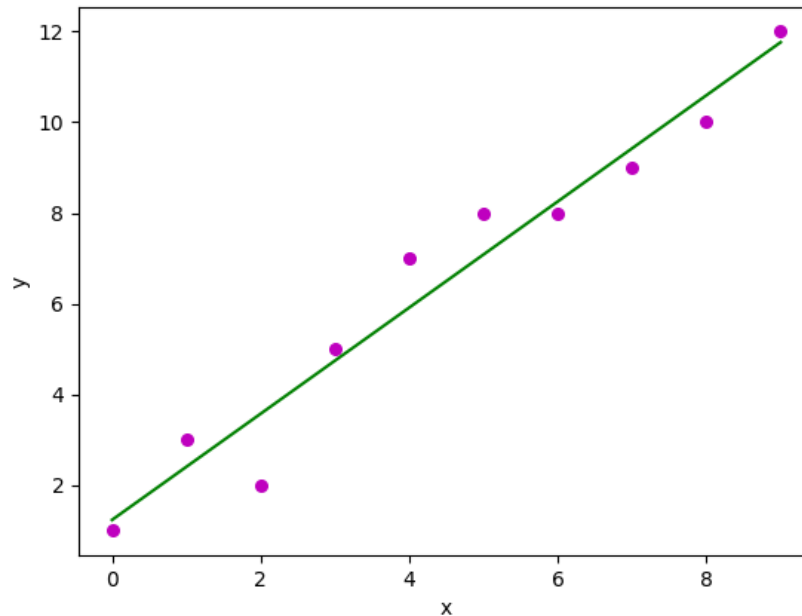
```
if __name__ == "__main__":
    main()
```

## Output

Estimated coefficients:

$b_0 = 1.2363636363636363$

$b_1 = 1.1696969696969697$



## 5.c Multi Linear Regression

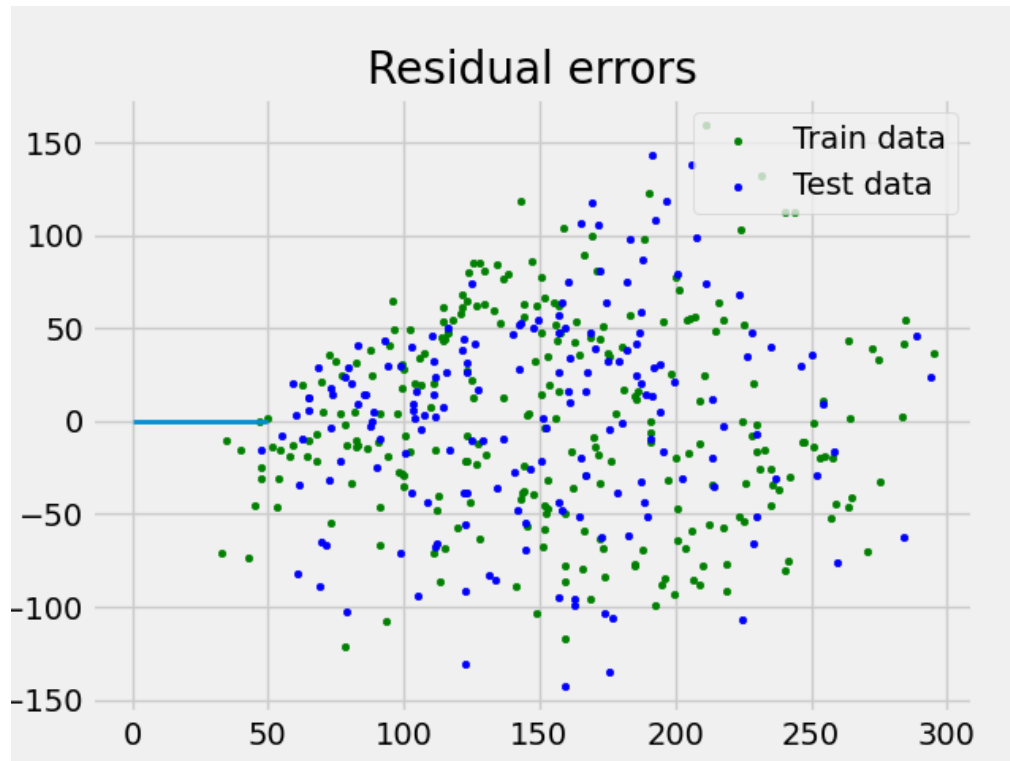
### Program for multi linear regression

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model, metrics
DD = datasets.load_diabetes(return_X_y=False)
X = DD.data
y = DD.target
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)
reg = linear_model.LinearRegression()
reg.fit(X_train, y_train)
print('Coefficients: ', reg.coef_)
print('Variance score: {}'.format(reg.score(X_test, y_test)))
plt.style.use('fivethirtyeight')
plt.scatter(reg.predict(X_train), reg.predict(X_train) - y_train,
            color="green", s=10, label='Train data')
plt.scatter(reg.predict(X_test), reg.predict(X_test) - y_test,
            color="blue", s=10, label='Test data')
plt.hlines(y=0, xmin=0, xmax=50, linewidth=2)
```

```
plt.legend(loc = 'upper right')
plt.title("Residual errors")
plt.show()
```

## Output

Coefficients: [ -59.73800266 -215.61743016 599.93653427 291.96204216 -829.65803548  
544.64726088 164.85115861 224.22784698 768.94812134 70.84955215]  
Variance score: 0.41603976183363156



## Program to find intercept, coefficient, prediction and standard error

```
import pandas as pd
from sklearn import linear_model
import statsmodels.api as sm
df=pd.read_csv('D:\diabetes.csv')
x = df[['plas','age']]
y = df['pres']
regr = linear_model.LinearRegression()
regr.fit(x, y)
print('Intercept: \n', regr.intercept_)
print('Coefficients: \n', regr.coef_)
x = sm.add_constant(x)
model = sm.OLS(y, x).fit()
predictions = model.predict(x)
print_model = model.summary()
print(print_model)
```

## Output

```
Intercept:  
50.35019628140931
```

```
Coefficients:  
[0.05820612 0.35253184]
```

```
| OLS Regression Results  
=====
```

Dep. Variable:	pres	R-squared:	0.066
Model:	OLS	Adj. R-squared:	0.064
Method:	Least Squares	F-statistic:	27.02
Date:	Tue, 18 Oct 2022	Prob (F-statistic):	4.59e-12
Time:	16:14:42	Log-Likelihood:	-3338.6
No. Observations:	768	AIC:	6683.
Df Residuals:	765	BIC:	6697.
Df Model:	2		
Covariance Type:	nonrobust		

```
=====
```

	coef	std err	t	P> t	[0.025	0.975]
-----	-----	-----	-----	-----	-----	-----
const	50.3502	2.940	17.126	0.000	44.579	56.122
plas	0.0582	0.022	2.654	0.008	0.015	0.101
age	0.3525	0.060	5.913	0.000	0.235	0.470
-----	-----	-----	-----	-----	-----	-----

Omnibus:	335.224	Durbin-Watson:	1.963
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1605.380
Skew:	-1.985	Prob(JB):	0.00
Kurtosis:	8.865	Cond. No.	563.

```
=====
```

### Notes:

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

## Program for scatter plot and multiple regression of a dataset ( using pressure and age)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('D:\diabetes1.csv')
ax1 = df[df['Class'] == 'positive'].plot(kind='scatter', x='pres', y='age', color='blue', alpha=0.5,
figsize=(10, 7))
df[df['Class'] == 'negative'].plot(kind='scatter', x='pres', y='age', color='magenta', alpha=0.5,
figsize=(10, 7), ax=ax1)
df_positive_sample = df[df['Class'] == 'positive'].sample(50)
df_negative_sample = df[df['Class'] == 'negative'].sample(50)
pos_fit = np.polyfit(df_positive_sample.pres, df_positive_sample.age, 1)
neg_fit = np.polyfit(df_negative_sample.pres, df_negative_sample.age, 1)

print(pos_fit)
print(neg_fit)
fig = plt.figure(figsize=(10, 7))
sns.regplot(x=df_positive_sample.pres, y=df_positive_sample.age, color='blue', marker='+')
```

```

sns.regplot(x=df_positive_sample.pres, y=df_negative_sample.age, color='magenta', marker='+')
plt.legend(labels=['Positive', 'Negative'])
plt.title('Relationship between Pressure and Age', size=24)
plt.xlabel('Pressure', size=18)
plt.ylabel('Age', size=18);
plt.show()

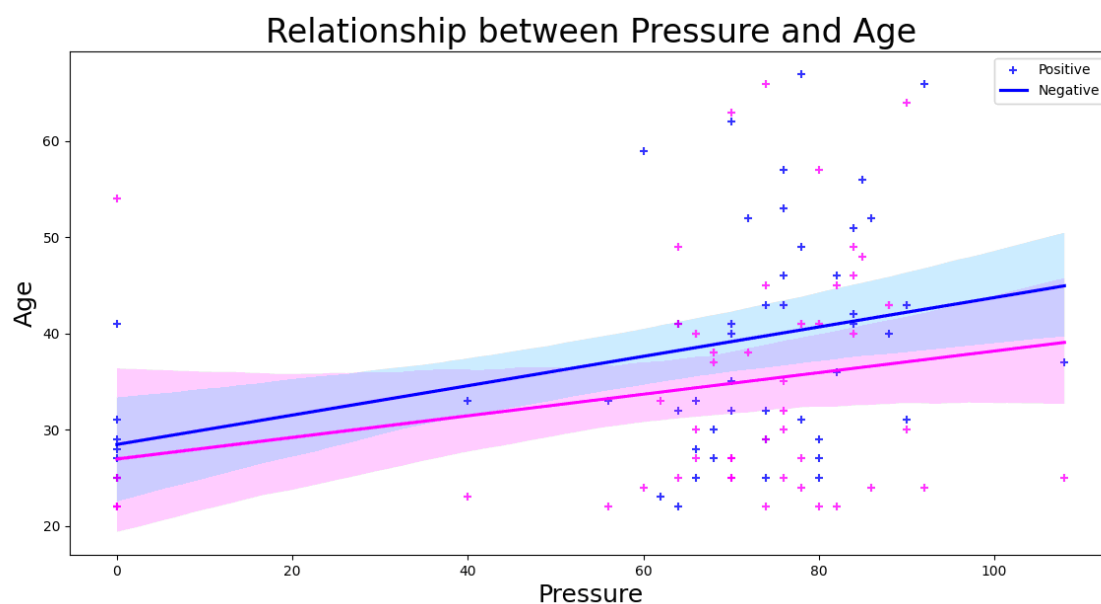
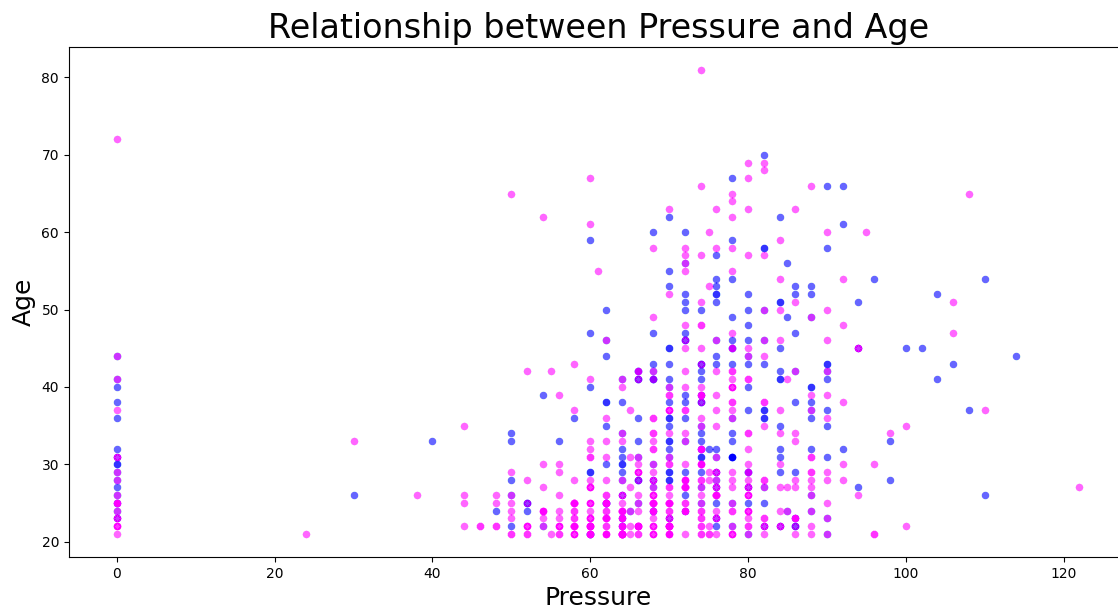
```

## Output

Coefficients

[ 0.08700875 29.78164098]

[ 0.25144586 14.1738764 ]



## Result:

Thus, the python programs for various univariant, bivariant, linear regression, logistic regression and multi linear regression with visualization is written and executed successfully.

Ex.No. : 06

## Plotting

### Aim:

To write the python program to apply and explore various plotting's for UCI data sets.

### Algorithm:

1. Start the python program.
2. Import the required packages like matplotlib, seaborn, scikitlearn, etc.
3. Read the data from various dataset downloaded from UCI repository.
4. Plot various plots like normal curves, scatter plot, counter plots, 3D plot etc.
5. Display the plots.
6. Stop the program.

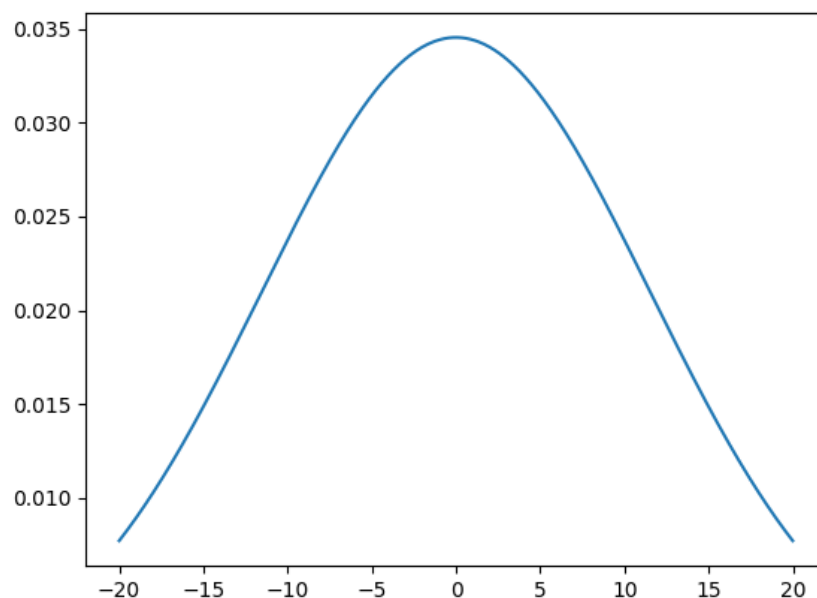
### Programs:

#### A. Normal Curve

program to plot normal curve for the given range of values.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import statistics
x_axis = np.arange(-20, 20, 0.01)
mean = statistics.mean(x_axis)
sd = statistics.stdev(x_axis)
plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
plt.show()
```

### Output



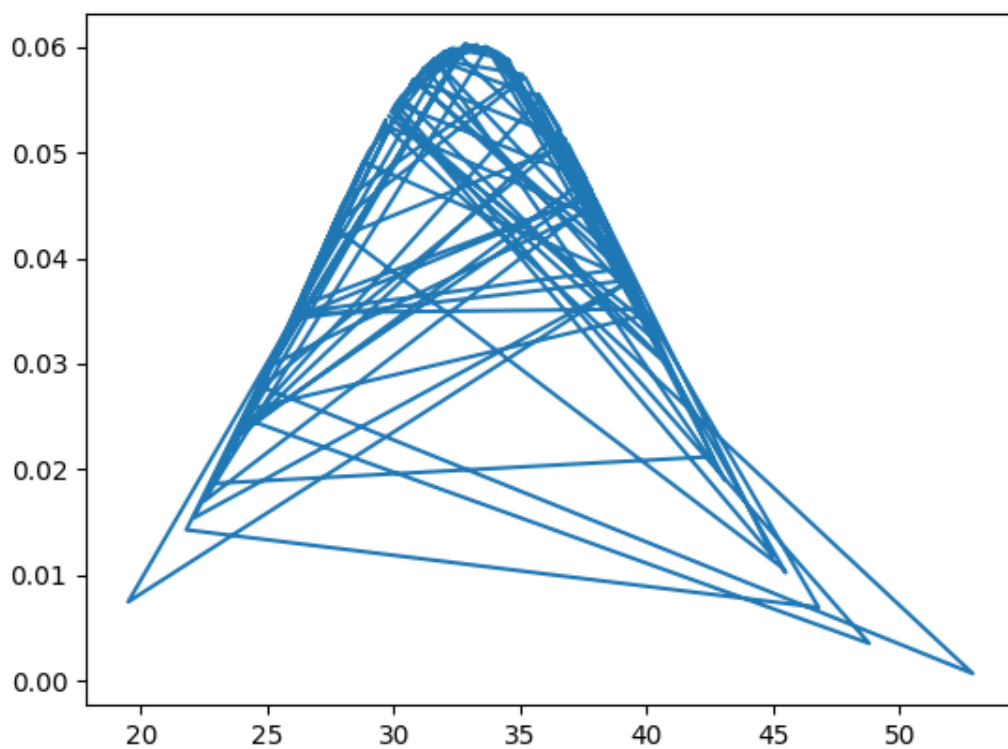
### Program to plot normal curve for the diabetes dataset.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import statistics
df=pd.read_csv('D:\diabetes.csv')
x_axis = df['mass'].sample(100)

mean = statistics.mean(x_axis)
sd = statistics.stdev(x_axis)

plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
plt.show()
```

### Output

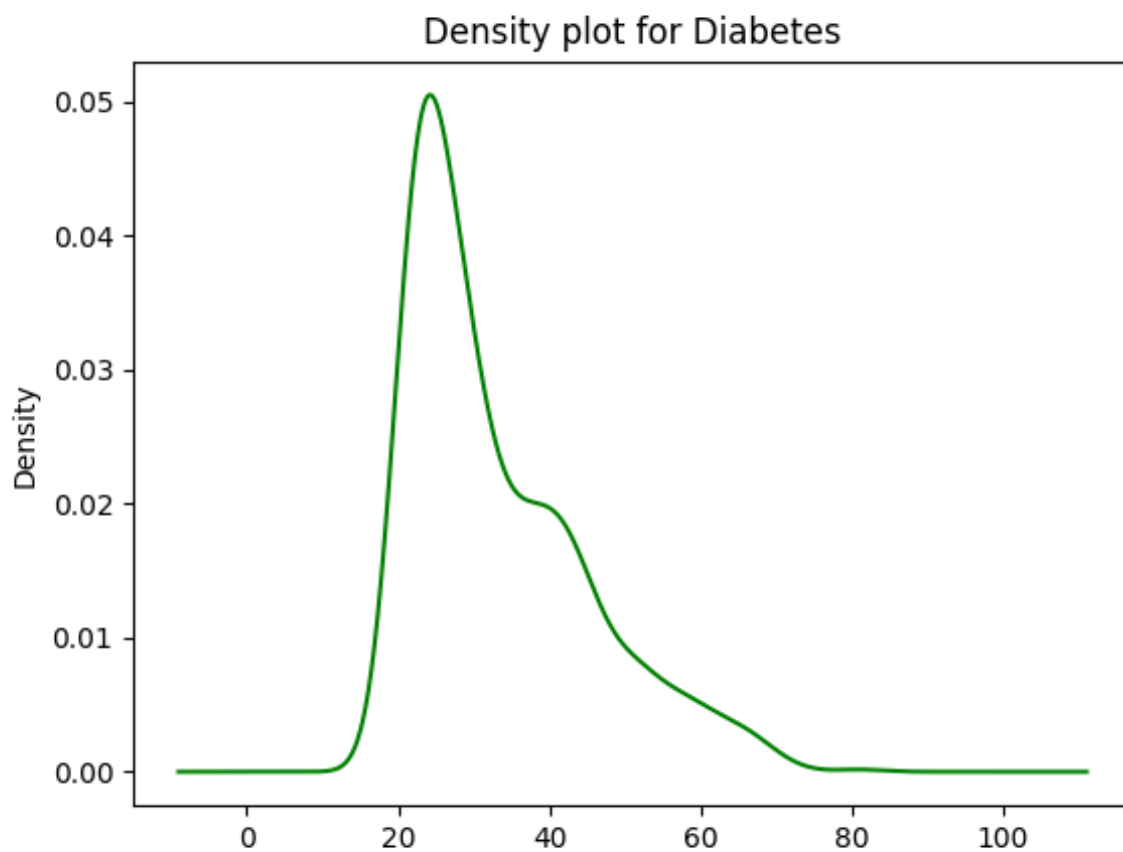


## B.Density and contour plots

### Program to plot density plot for diabetes data set

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data=pd.read_csv('D:\diabetes.csv')
data.age.plot.density(color='green')
plt.title('Density plot for Diabetes')
plt.show()
```

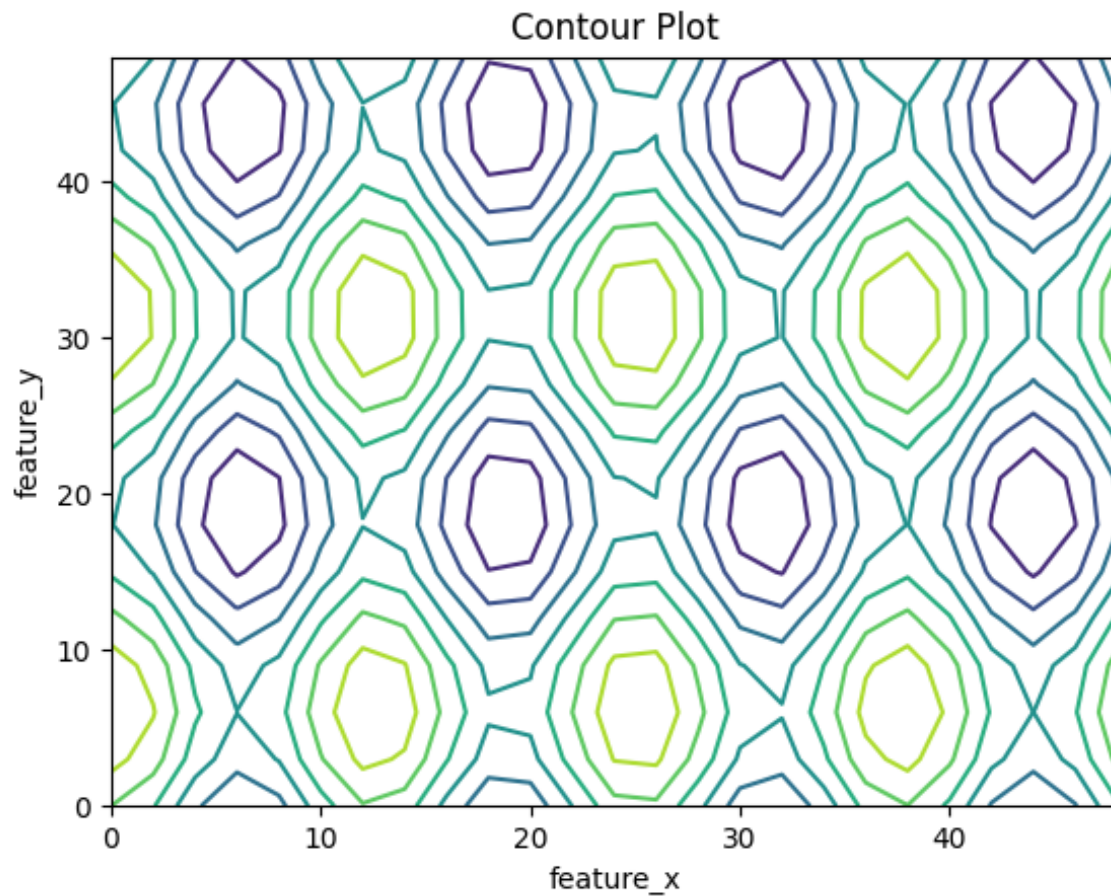
### Output



## Program for contour plot

```
import matplotlib.pyplot as plt
import numpy as np
feature_x = np.arange(0, 50, 2)
feature_y = np.arange(0, 50, 3)
[X, Y] = np.meshgrid(feature_x, feature_y)
fig, ax = plt.subplots(1, 1)
Z = np.cos(X / 2) + np.sin(Y / 4)
ax.contour(X, Y, Z)
ax.set_title('Contour Plot')
ax.set_xlabel('feature_x')
ax.set_ylabel('feature_y')
plt.show()
```

## Output

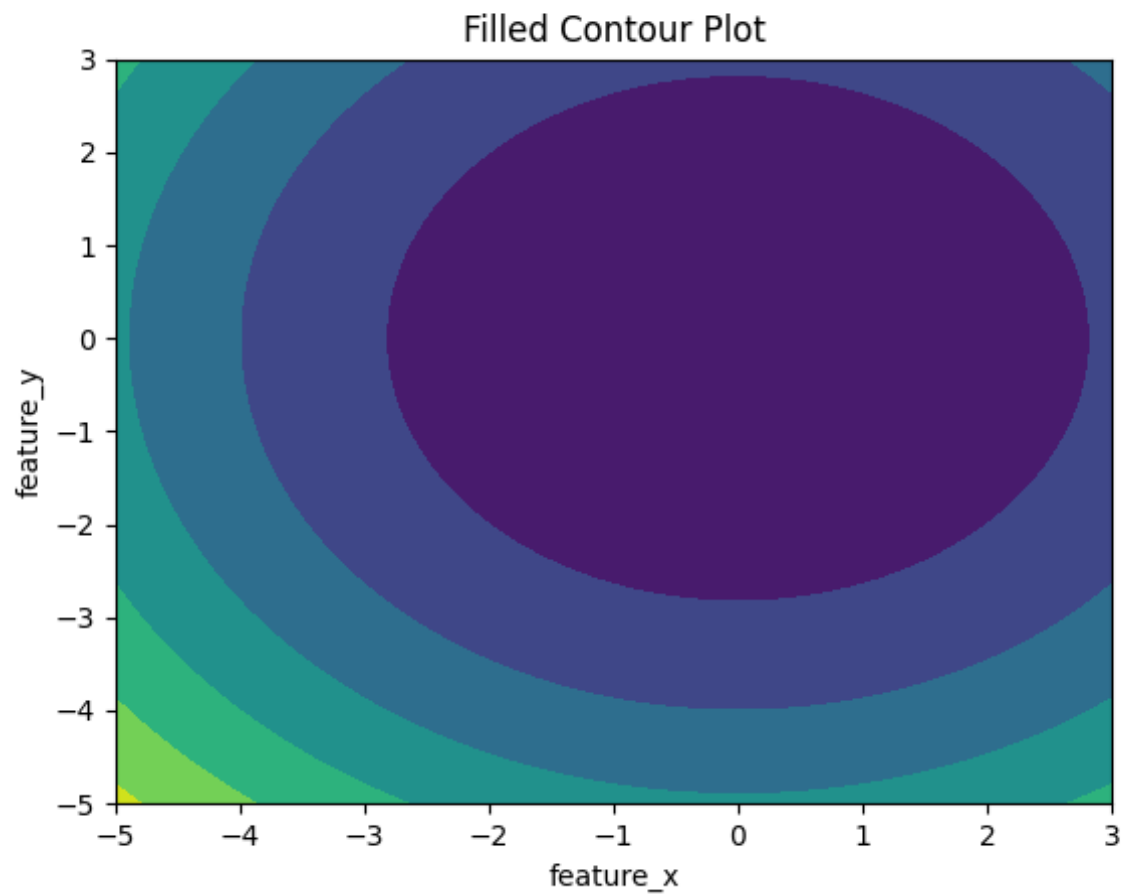




### program for filled contour plot

```
import matplotlib.pyplot as plt
import numpy as np
feature_x = np.linspace(-5.0, 3.0, 70)
feature_y = np.linspace(-5.0, 3.0, 70)
[X, Y] = np.meshgrid(feature_x, feature_y)
fig, ax = plt.subplots(1, 1)
Z = X ** 2 + Y ** 2
ax.contourf(X, Y, Z)
ax.set_title('Filled Contour Plot')
ax.set_xlabel('feature_x')
ax.set_ylabel('feature_y')
plt.show()
```

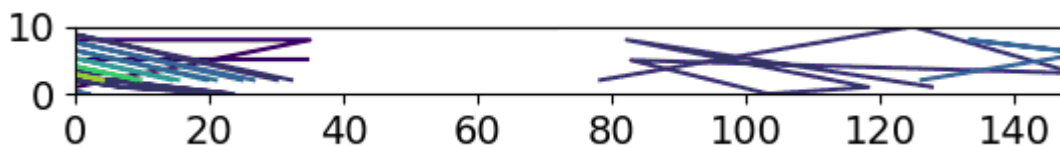
### Output



### Program to plot the contour plot for a data set

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
px_orbital = pd.read_csv("D:\diabetes1.csv")
x = px_orbital.iloc[0, 1:]
y = px_orbital.iloc[1:, 0]
px_values = px_orbital.iloc[1:, 1:]
mpl.rcParams['font.size'] = 14
mpl.rcParams['legend.fontsize'] = 'large'
mpl.rcParams['figure.titlesize'] = 'medium'
fig, ax = plt.subplots()
ax.contour(x, y, px_values)
ax.set_aspect('equal')
plt.show()
```

### Output

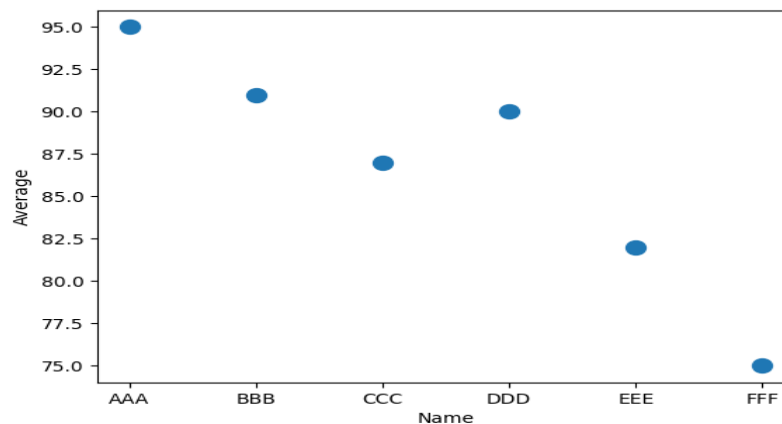


### C. Correlation and scatter plots

#### Program to plot a scatter plot using data frame

```
import pandas as pd
import matplotlib.pyplot as plt
data={'Name':['AAA', 'BBB', 'CCC', 'DDD', 'EEE', 'FFF'],
      'Average':[95, 91, 87, 90, 82, 75]}
df = pd.DataFrame(data = data);
df.plot.scatter(x = 'Name', y = 'Average', s = 100);
plt.show()
```

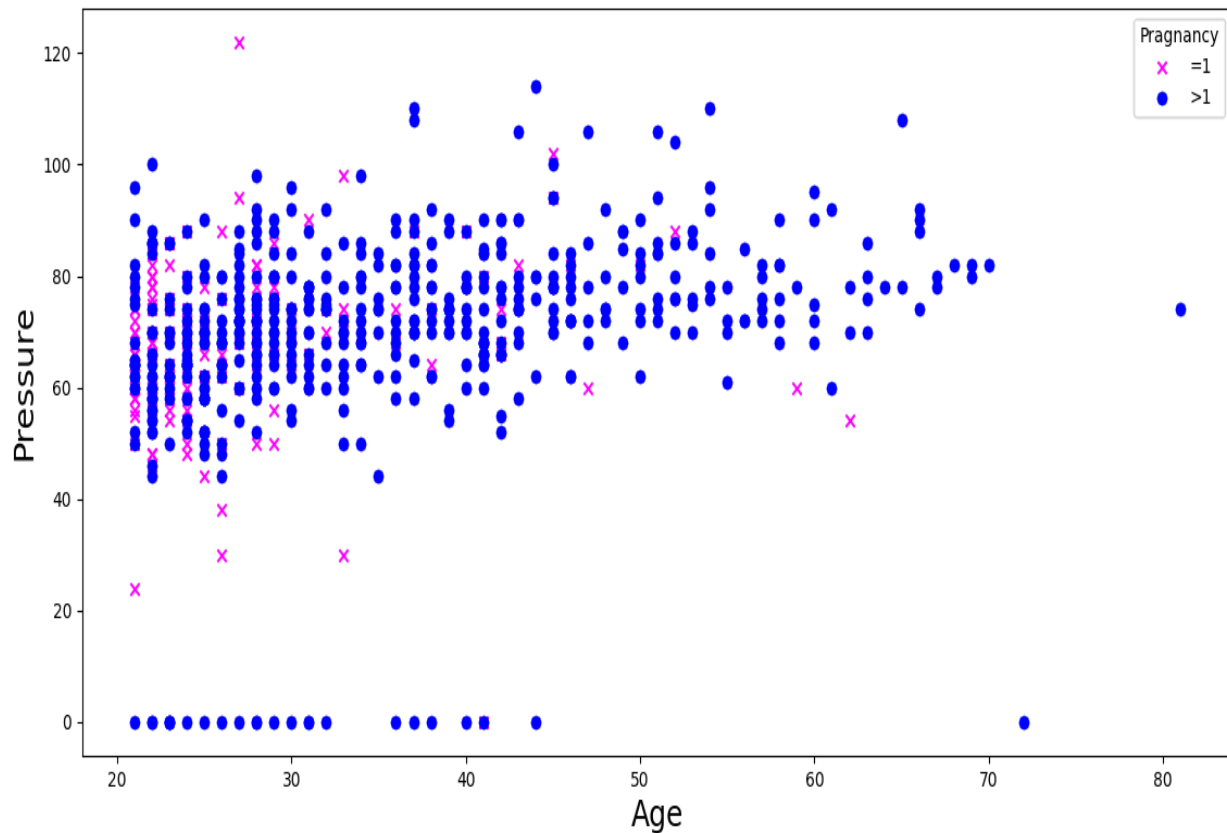
### Output



## Program to plot scatter plot for a dataset (age and pressure based pregnancy)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv("D:\diabetes.csv")
plt.figure(figsize=(10,7))
plt.scatter('age', 'pres', data=data[data.preg==1], marker = 'x', c = 'magenta')
plt.scatter('age', 'pres', data=data[data.preg>1], marker = 'o', c = 'blue')
plt.legend(('=1', '>1'), title='Pragnancy')
plt.xlabel('Age', size=18)
plt.ylabel('Pressure', size=18);
plt.show()
```

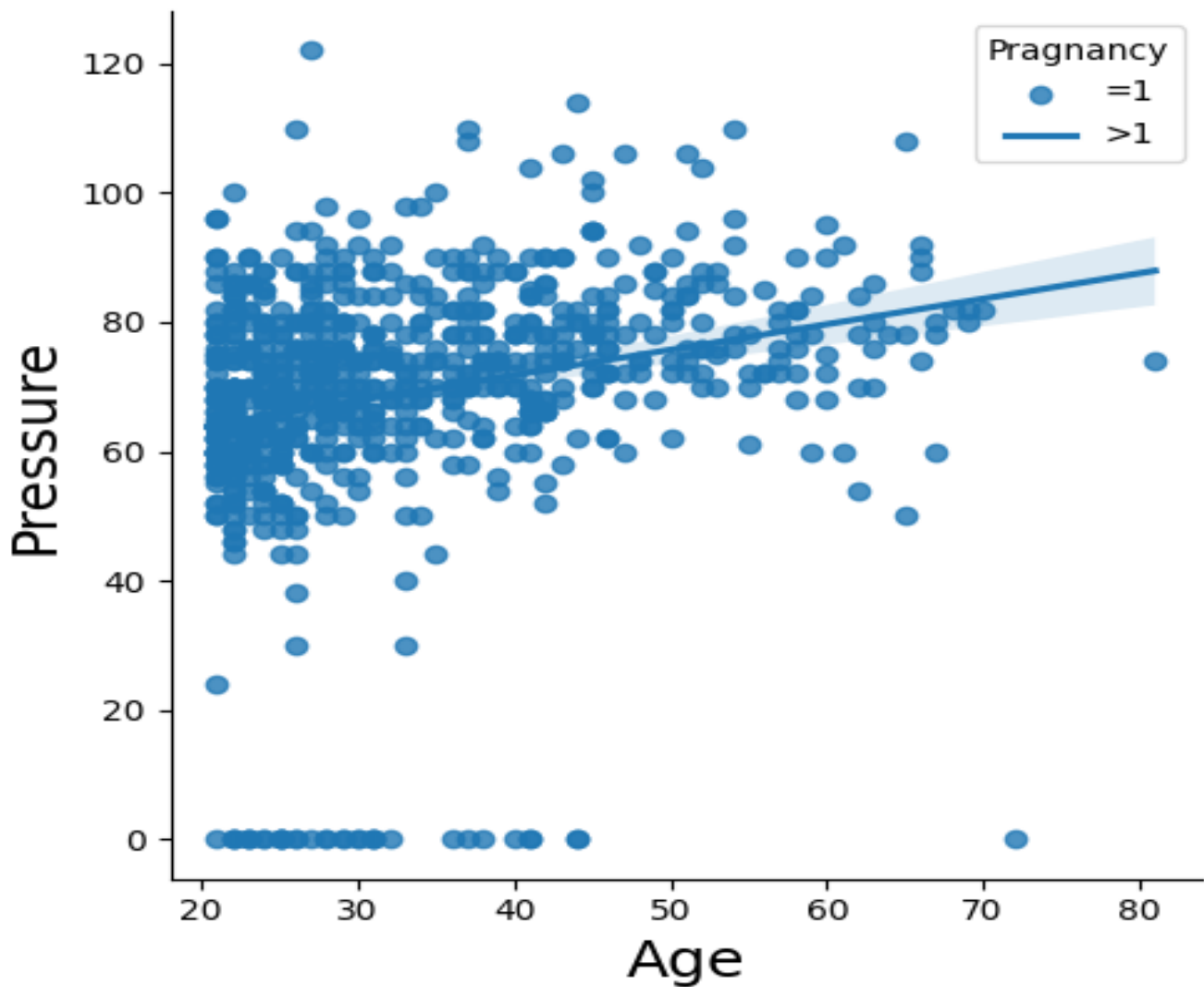
## Output



### Program to plot scatter plot with regression line for a dataset (age and pressure based pregnancy)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv("D:\diabetes.csv")
plt.figure(figsize=(10,7))
sns.lmplot(x = "age", y = "pres", data=data)
plt.legend(('=1', '>1'), title='Pregnancy')
plt.xlabel('Age', size=18)
plt.ylabel('Pressure', size=18);
plt.show()
```

### Output

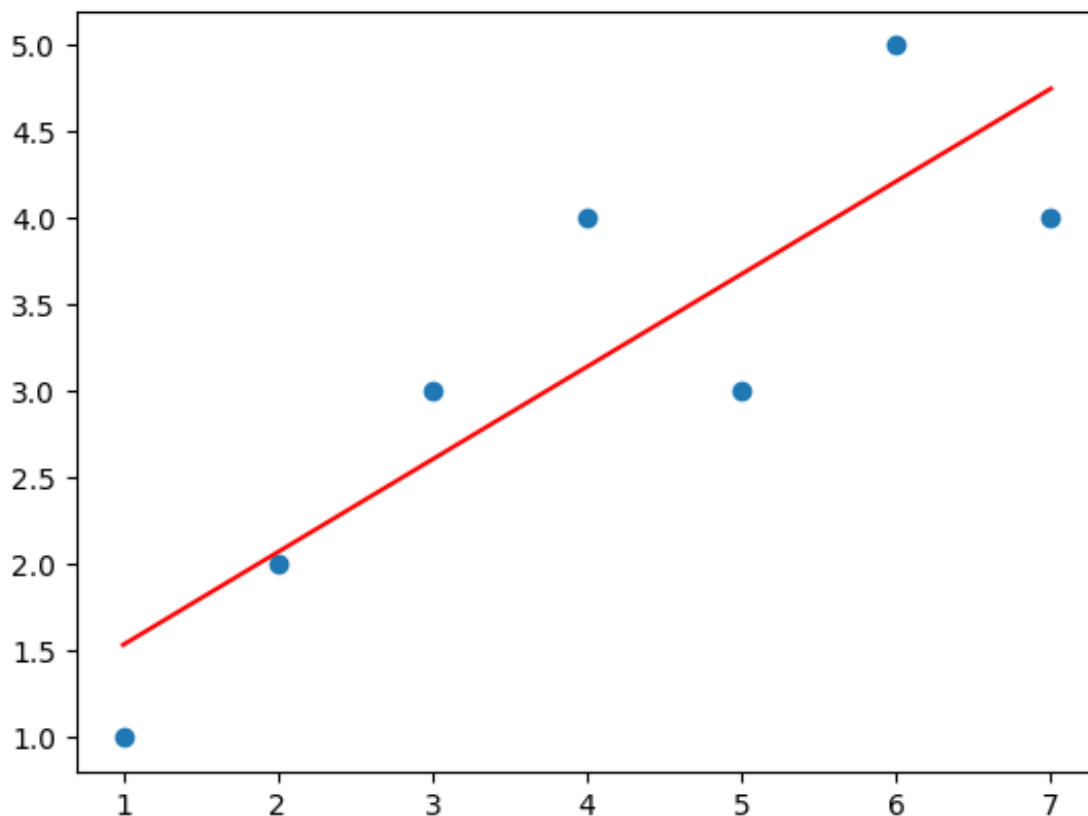


### Program to find correlation and plot the graph for set of data's

```
import sklearn
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
y = pd.Series([1, 2, 3, 4, 3, 5, 4])
x = pd.Series([1, 2, 3, 4, 5, 6, 7])
correlation = y.corr(x)
print(correlation)
plt.scatter(x, y)
plt.plot(np.unique(x), np.poly1d(np.polyfit(x, y, 1))
        (np.unique(x)), color='red')
plt.show()
```

### Output

0.8603090020146067

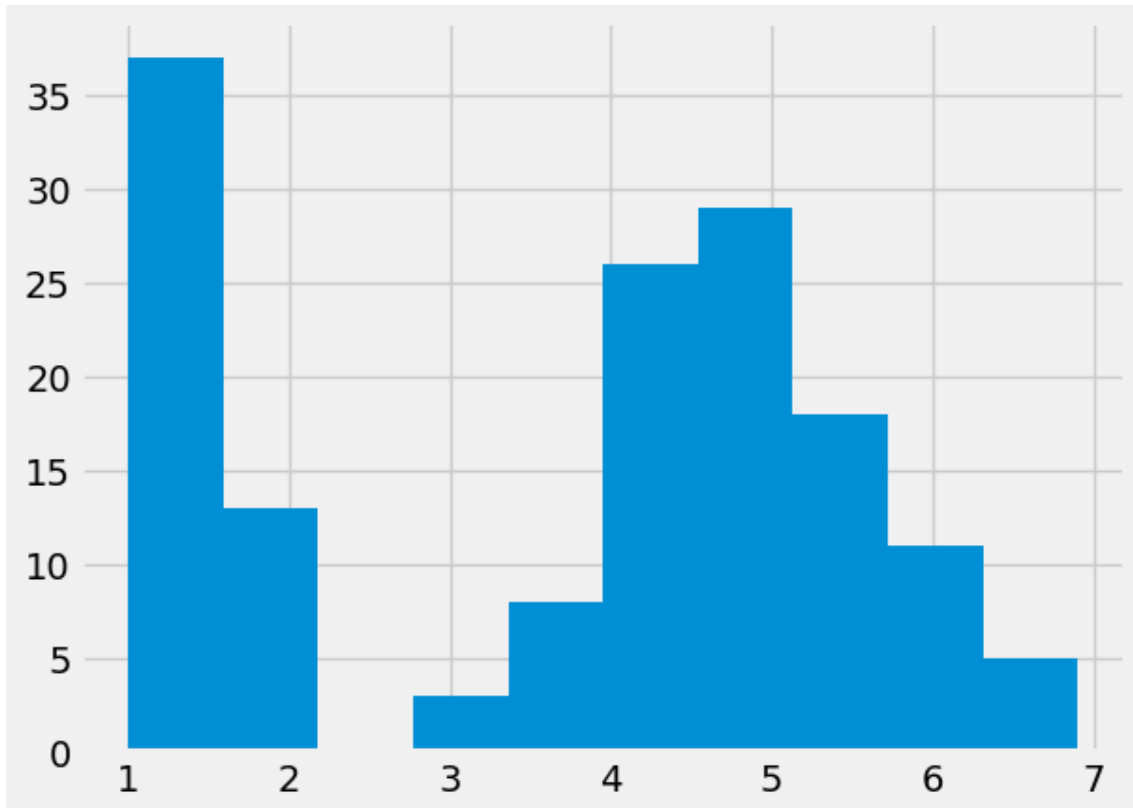


## D. Histograms

### Program to draw Histogram for a data set

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('D:\iris.csv')
plt.hist(df['petal.length'])
plt.show()
```

### Output:

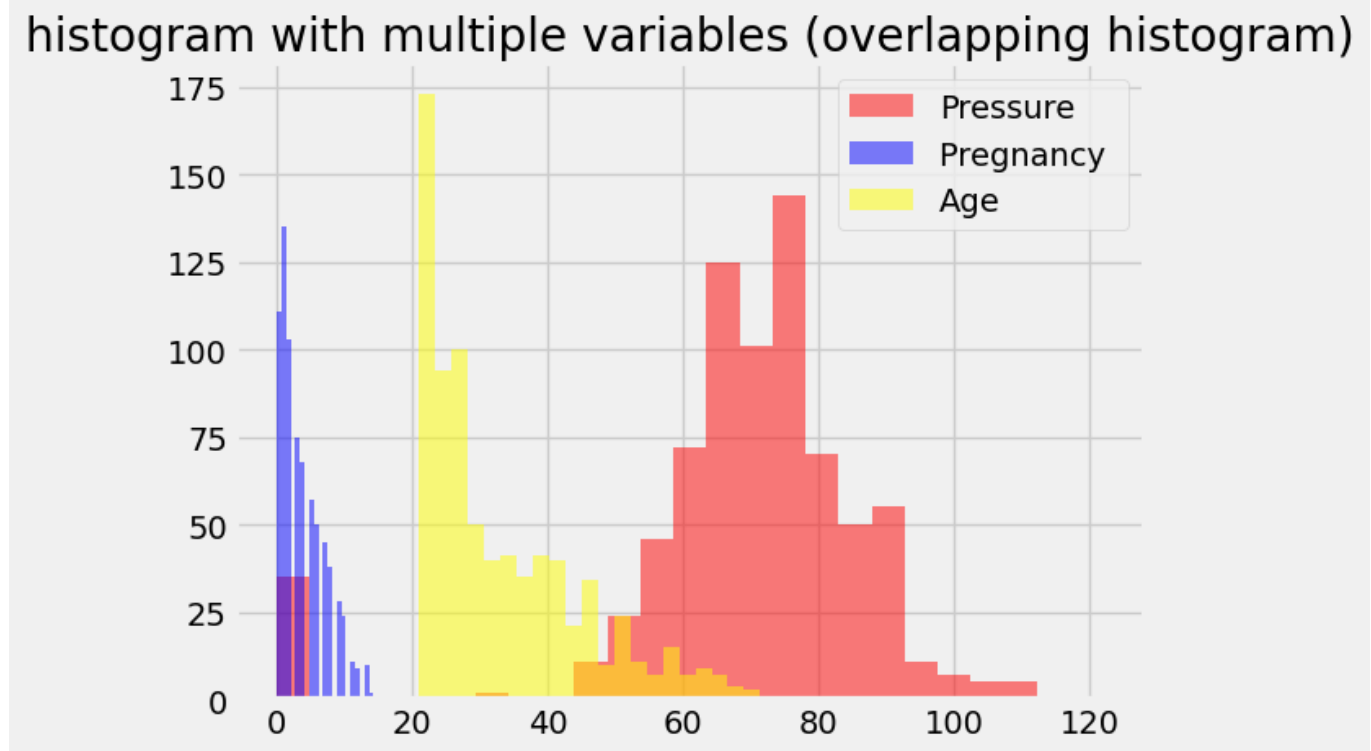


### Program to plot Histograms for various fields

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df = pd.read_csv('D:\diabetes.csv')
plt.hist(df['pres'],bins = 25, alpha = 0.5,
         color = 'red')
plt.hist(df['preg'],bins = 25, alpha = 0.5,
         color = 'blue')
plt.hist(df['age'],bins = 25, alpha = 0.5,
         color = 'yellow')
plt.title("histogram with multiple \variables (overlapping histogram)")
plt.legend(['Pressure','Pregnancy ','Age'])

plt.show()
```

**Output:**



### E. Three Dimensional Plots

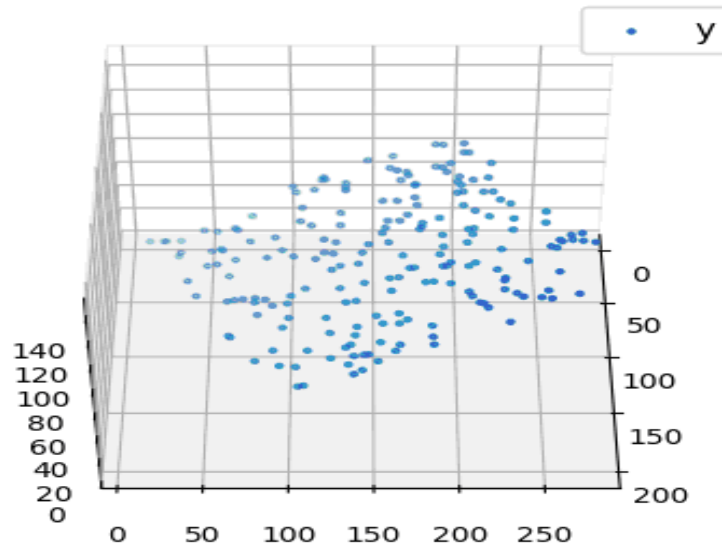
**Program to generate 3D Plot.**

```
import numpy as np
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

def generate_dataset(n):
    x = []
    y = []
    random_x1 = np.random.rand()
    random_x2 = np.random.rand()
    for i in range(n):
        x1 = i
        x2 = i/2 + np.random.rand()*n
        x.append([1, x1, x2])
        y.append(random_x1 * x1 + random_x2 * x2 + 1)
    return np.array(x), np.array(y)

x, y = generate_dataset(200)
mpl.rcParams['legend.fontsize'] = 12
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
ax.scatter(x[:, 1], x[:, 2], y, label='y', s=10)
ax.legend()
ax.view_init(45, 0)
plt.show()
```

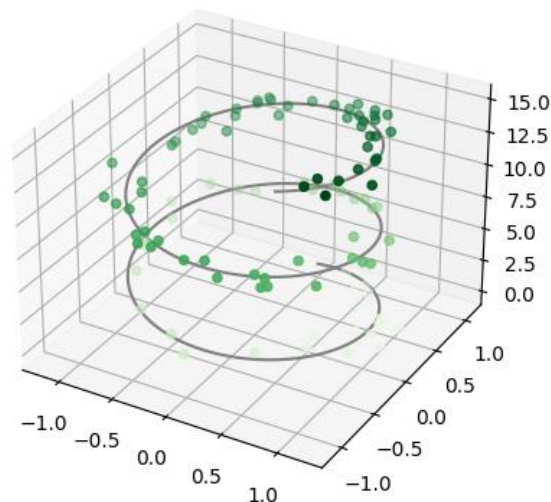
## Output



### Program for Three-Dimensional Points and Lines

```
import numpy as np
import matplotlib.pyplot as plt
ax = plt.axes(projection='3d')
zline = np.linspace(0, 15, 1000)
xline = np.sin(zline)
yline = np.cos(zline)
ax.plot3D(xline, yline, zline, 'gray')
# Data for three-dimensional scattered points
zdata = 15 * np.random.random(100)
xdata = np.sin(zdata) + 0.1 * np.random.randn(100)
ydata = np.cos(zdata) + 0.1 * np.random.randn(100)
ax.scatter3D(xdata, ydata, zdata, c=zdata, cmap='Greens');
```

### Output:

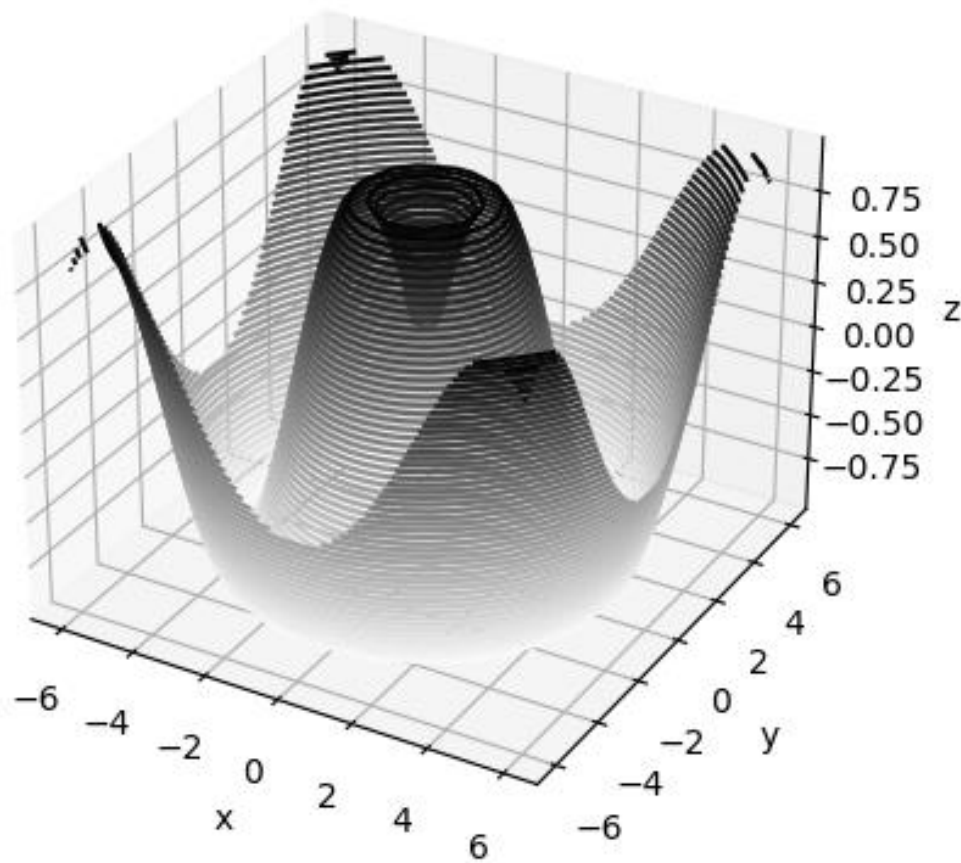




## Program for Three-Dimensional Contour Plots

```
import numpy as np
import matplotlib.pyplot as plt
def f(x, y):
    return np.sin(np.sqrt(x ** 2 + y ** 2))
x = np.linspace(-6, 6, 30)
y = np.linspace(-6, 6, 30)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.contour3D(X, Y, Z, 50, cmap='binary')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z');
```

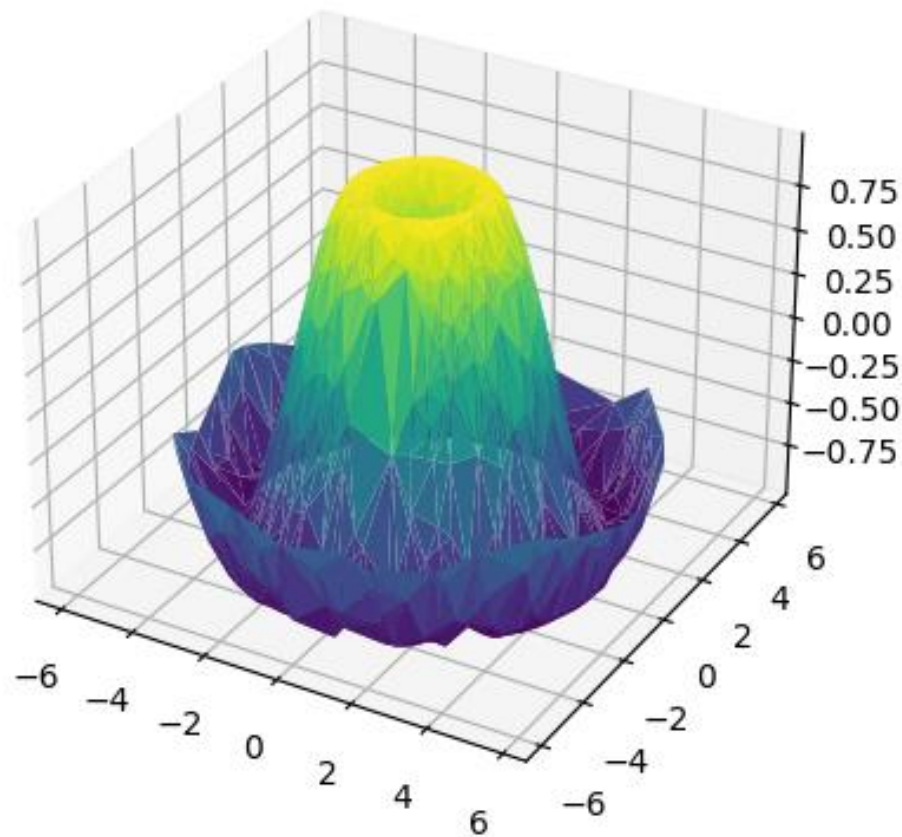
**Output:**



## Surface Triangulations

```
import numpy as np
import matplotlib.pyplot as plt
r = np.linspace(0, 6, 20)
theta = 2 * np.pi * np.random.random(1000)
r = 6 * np.random.random(1000)
x = np.ravel(r * np.sin(theta))
y = np.ravel(r * np.cos(theta))
z = f(x, y)
ax = plt.axes(projection='3d')
ax.plot_trisurf(x, y, z,
cmap='viridis', edgecolor='none');
```

### Output:



### Result:

Thus the python program for normal curve, scatter plot, counter plot, density plot, histogram and 3D plots are written and executed successfully.

Ex.No, : 07

### **Visualizing Geographic Data with Basemap**

#### **Aim:**

To write python program to visualize geographic data's by using basemap package and display the map in various projections.

#### **Algorithm:**

1. Start the python program.
2. Install the basemap package using pip command.
3. Install the pillow package using pip command.
4. Import basemap package and use mpl\_toolkits.
5. Give the required latitude and longitude values.
6. Enter the name of the place representing the latitude and longitude values.
7. Display the map in different forms of projections.
8. Stop the program.

#### **Plotting data and labels on a map**

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
fig = plt.figure(figsize=(8, 8))
m = Basemap(projection='lcc', resolution=None,
width=8E6, height=8E6,
lat_0=45, lon_0=-100,)
m.etopo(scale=0.5, alpha=0.5)
x, y = m(-122.3, 47.6)
plt.plot(x, y, 'ok', markersize=5)
plt.text(x, y, ' Seattle', fontsize=12);
```

**Output:**

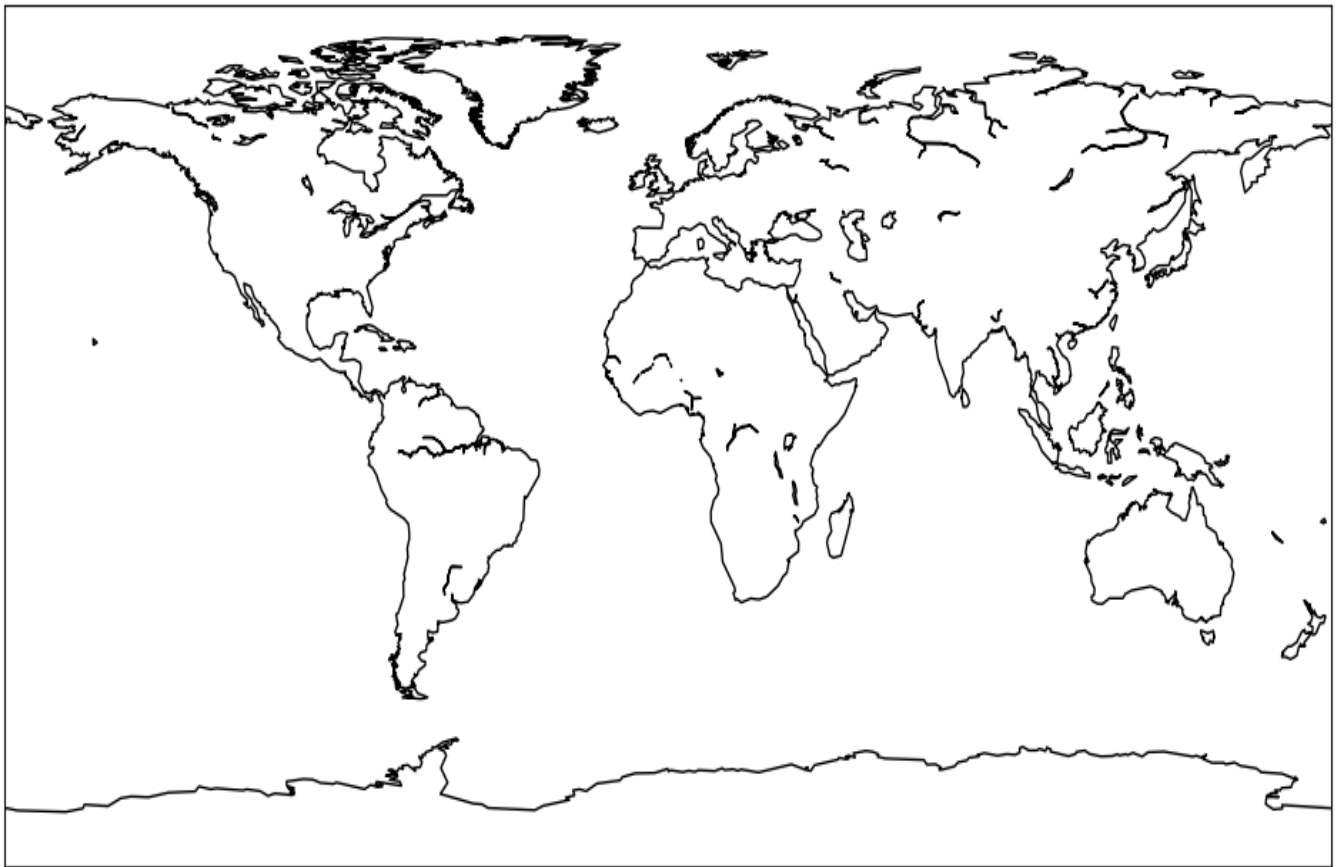


**Program for costal lines**

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
fig = plt.figure(figsize = (12,12))
m = Basemap()
m.drawcoastlines()
plt.title("Coastlines", fontsize=20)
plt.show()
```

**Output:**

Coastlines



## Program for Map projection – Cylindrical Projection

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
from itertools import chain
def draw_map(m, scale=0.2):
    m.shadedrelief(scale=scale)
    lats = m.drawparallels(np.linspace(-90, 90, 13))
    lons = m.drawmeridians(np.linspace(-180, 180, 13))
    lat_lines = chain(*(tup[1][0] for tup in lats.items()))
    lon_lines = chain(*(tup[1][0] for tup in lons.items()))
    all_lines = chain(lat_lines, lon_lines)
    line.set(linestyle='-', alpha=0.3, color='w')
fig = plt.figure(figsize=(8, 6), edgecolor='w')
m = Basemap(projection='cyl', resolution=None,
llcrnrlat=-90, urcrnrlat=90,
llcrnrlon=-180, urcrnrlon=180, )
draw_map(m)
```

## Output:

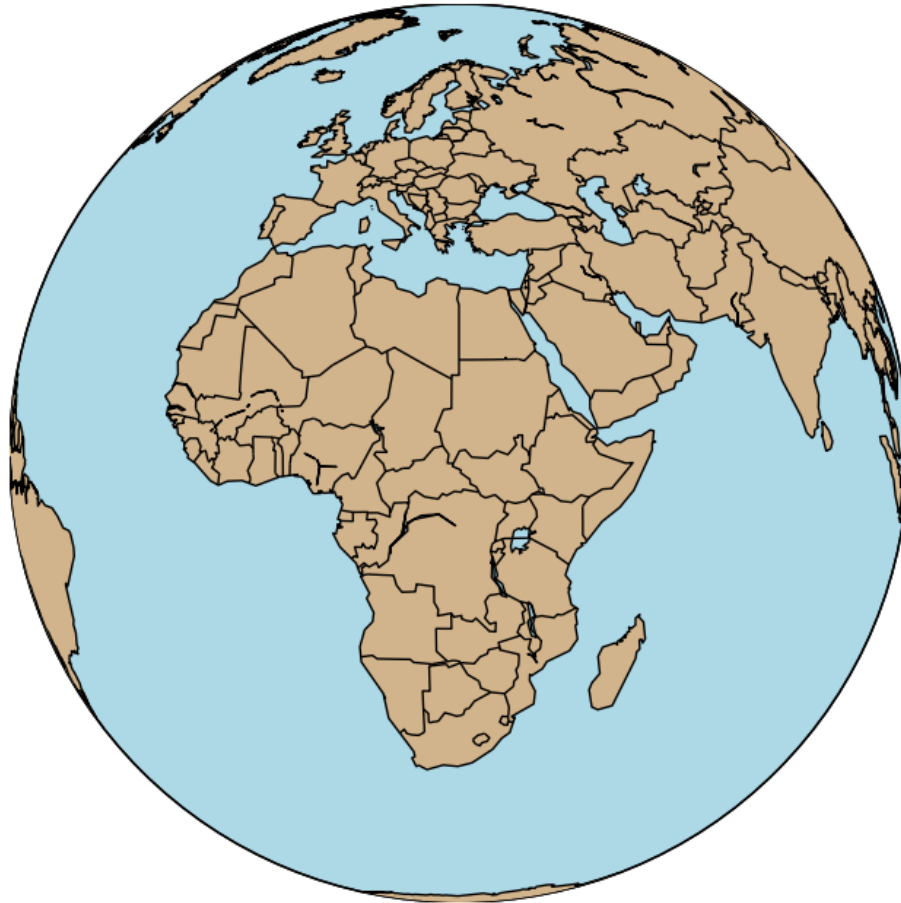


### Program for Map projection – Orthographic Projection

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
fig = plt.figure(figsize = (10,8))
m = Basemap(projection='ortho', lon_0 = 25, lat_0 = 10)
m.drawcoastlines()
m.fillcontinents(color='tan',lake_color='lightblue')
m.drawcountries(linewidth=1, linestyle='solid', color='k' )
m.drawmapboundary(fill_color='lightblue')
plt.title("Orthographic Projection", fontsize=18)
```

**Output;**

Orthographic Projection



**Result:**

Thus, the python program to geographic data and various projections are written and executed successfully.