


## Practical – 1

**AIM:** Study of numpy library of Python. Write a program to demonstrate use of various functions.


### Numpy:

- In Python we have lists that serve the purpose of arrays, but they are slow to process.
- NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
- Arrays are very frequently used in data science, where speed and resources are very important.

#### ❖ Installation of NumPy:

- If you have Python and PIP already installed on a system, it will install the numpy.
- Command:  
 **pip install numpy**

#### ❖ Import NumPy:

- Once NumPy is installed, import it in your applications by adding the import keyword.
- Create an alias with the as keyword while importing:
- Command:  
 **import numpy as np**

### Programs:

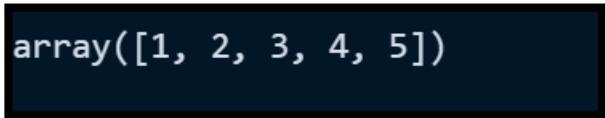
#### 1. Create an Array:

- We can create a NumPy ndarray object by using the array() function.
- We can create multi-dimensional array by using numpy.

#### CODE:

```
##create an 1D array  
a = np.array([1,2,3,4,5])  
a
```

#### OUTPUT:



```
array([1, 2, 3, 4, 5])
```

**CODE:**

```
# create an 2D array of multidimensional  
b = np.array([[1,2,3],[4,5,6]])  
b
```

**OUTPUT:**

```
array([[1, 2, 3],  
       [4, 5, 6]])
```

**2. Shape of an Array:**

- The shape of an array is the number of elements in each dimension.
- NumPy arrays have an attribute called shape that returns a tuple with each index having the number of corresponding elements.

**CODE:**

```
# return the row and columns of the array  
arr3.shape
```

**OUTPUT:**

```
(3, 2)
```

**3. Size of an Array:**

- Numpy arrays have attribute called size that returns a total no. of elements are available in the array.

**CODE:**

```
# return the total no. of elements in the array  
a.size
```

**OUTPUT:**

```
5
```

**CODE:**

```
x = np.size(a)
x
```

**OUTPUT:****4. Dimension of an Array:**

- NumPy Arrays provides the ndim attribute that returns an integer that tells us how many dimensions the array have.

**CODE:**

```
arr3.ndim
```

**OUTPUT:****5. Sorting of numpy array:**

- The NumPy ndarray object has a function called sort(), that will sort a specified array.
- You can also sort arrays of strings, or any other data type.

**CODE:**

```
# create an 1D array
arr5 = np.array([9,0,11,12,0,2,43,35,6,21])
arr5

# 1D array sort simply
np.sort(arr5)
```

**OUTPUT:**

```
# create an 1D array
arr5 = np.array([9,0,11,12,0,2,43,35,6,21])
arr5

array([ 9,  0, 11, 12,  0,  2, 43, 35,  6, 21])

# 1D array sort simply
np.sort(arr5)

array([ 0,  0,  2,  6,  9, 11, 12, 21, 35, 43])
```

**6. Zeros function of numpy:**

- The numpy.zeros() function returns a new array of given shape and type, with zeros.

**CODE:**

```
x = np.zeros(2)
x
```

**OUTPUT:**

```
array([0., 0.])
```

**7. Ones function of numpy:**

- The `numpy.ones()` function returns a new array of given shape and type, with ones.

**CODE:**

```
y = np.ones((2,3))  
  
y
```

**OUTPUT:**

```
array([[1., 1.],  
       [1., 1.]])
```

**8. Arange function of numpy:**

- The `arange([start,] stop[, step,][, dtype])` : Returns an array with evenly spaced elements as per the interval. The interval mentioned is half-opened i.e. [Start, Stop)

**CODE:**

```
z = np.arange(10)  
print(z)  
  
p = np.arange(2,8)  
print(p)  
  
q = np.arange(5,10,2)  
print(q)  
  
r = np.arange(10,5,-2)  
print(r)
```

**OUTPUT:**

```
[0 1 2 3 4 5 6 7 8 9]  
[2 3 4 5 6 7]  
[5 7 9]  
[10 8 6]
```

**9. Concatenate of numpy arrays:**

- For concatenate the dimension or shape should be same for the both the arrays.

**CODE:**

```
np.concatenate((np.array([1,2,3]),a),axis=0)
```

**OUTPUT:**

```
array([1, 2, 3, 1, 2, 3, 4, 5])
```

**10. Random class of numpy:**

- The random is a module present in the NumPy library. This module contains the functions which are used for generating random numbers.
- This module contains some simple random data generation methods, some permutation and distribution functions, and random generator functions.

**CODE:**

```
print(np.random.rand(4))  
print(np.random.rand(2,5,2))  
print(np.random.rand(1,5))
```

**OUTPUT:**

```
[0.86507345 0.12504403 0.52014018 0.8168395 ]  
[[[0.41558978 0.25991226]  
 [0.52479828 0.11488848]  
 [0.22069653 0.65400866]  
 [0.11233719 0.93027937]  
 [0.79941694 0.94825028]]  
  
 [[0.74847756 0.00108545]  
 [0.52406901 0.45411413]  
 [0.45173186 0.82146872]  
 [0.16894577 0.4472261 ]  
 [0.90364501 0.97268656]]]  
[[0.87829515 0.51187129 0.76336302 0.50376698 0.19342203]]
```

**11. Linespace function of numpy:**

- The NumPy.linspace() function returns an array of evenly spaced values within the specified interval [start, stop].
- It is similar to NumPy.arange() function but instead of a step, it uses a sample number.

**CODE:**

```
c = np.linspace(10,20,num=3)
```

```
c
```

**OUTPUT:**

```
array([10., 15., 20.])
```

**12. Indexing in array:**

- Array indexing is the same as accessing an array element.
- You can access an array element by referring to its index number.
- The indexes in NumPy arrays start with 0, meaning that the first element has index 0, and the second has index 1 etc.

**CODE + OUTPUT:**

```
a = np.array([1,2,3,4,5])
print(a)

22]
.. [1 2 3 4 5]

print(a[0],a[1])
print(a[-1],a[-2])

23]
.. 1 2
   5 4

b = np.array([[1,2,3],[4,5,6]])
print(b[0][1])

24]
.. 2
```

**13. Slicing in array:**

- Slicing in python means taking elements from one given index to another given index.
- We pass slice instead of index like this: [start:end].
- We can also define the step, like this: [start:end:step].
- If we don't pass start its considered 0
- If we don't pass end its considered length of array in that dimension
- If we don't pass step its considered 1



**CODE + OUTPUT:**

```
a = np.array([[1,2],[4,5],[7,8]])
a[:, :]
5]
array([[1, 2],
       [4, 5],
       [7, 8]])

a[0][:]
0]
array([1, 2])

a[1][:]
2]
array([4, 5])

a[-1][0]
7]
7
```

14. Arithmetic Operation on array:**CODE + OUTPUT:**

```
c = np.array([5,6,7,8,9])
c
5]
array([5, 6, 7, 8, 9])

print(a)
1]
[1 2 3 4 5]

print(a+c)
6]
[ 6  8 10 12 14]

print(a-c)
7]
[-4 -4 -4 -4 -4]

print(a*c)
8]
[ 5 12 21 32 45]

print(a/c)
9]
[0.2      0.33333333 0.42857143 0.5      0.55555556]

print(a//c)
0]
[0 0 0 0 0]
```

**15. Max,Min,Mean and Std function of array:**

- NumPy has both a package-level function and an ndarray method named max(). They work in the same way, though the package function np.max() requires the target array name as its first parameter.

**CODE + OUTPUT:**

```
#min  
np.min(a)
```

0

```
#max  
np.max(a)
```

4

```
#mean  
np.mean(a)
```

2.0

```
#std  
np.std(a)
```

1.4142135623730951

16. Comparison of array:**CODE + OUTPUT:**

```
print(a[a<c])
```

```
[1 2 3 4 5]
```

```
print(a == c)
```

```
[False False False False False]
```

```
print(c[c>=6])
```

```
[6 7 8 9]
```

```
print(a[a%2 == 0])
```

```
[2 4]
```

```
print(c[(c>5) & (c<9)])
```

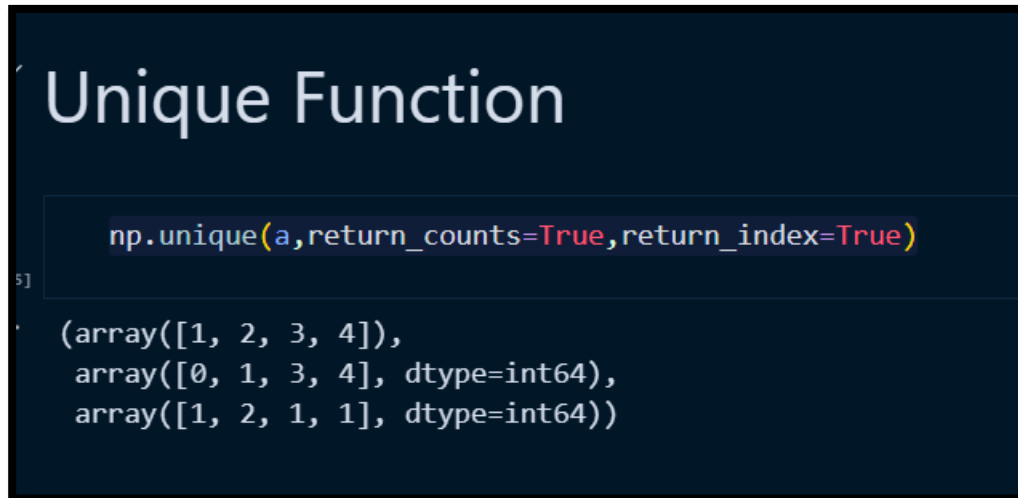
```
[6 7 8]
```

```
print(c[(c>5) | (c<9)])
```

```
[5 6 7 8 9]
```

**17. Unique in array:**

- The transpose of a matrix is found by interchanging its rows into columns or columns into rows.

**CODE + OUTPUT:**

```
np.unique(a,return_counts=True,return_index=True)

(array([1, 2, 3, 4]),
 array([0, 1, 3, 4], dtype=int64),
 array([1, 2, 1, 1], dtype=int64))
```

**18. Transpose of matrix:****CODE + OUTPUT:**

## Transpose and Reshape Matrix

```
np.transpose(b)
[6]
· array([[1, 2, 3],
        [2, 3, 4],
        [3, 5, 6]])

b.transpose()
[7]
· array([[1, 2, 3],
        [2, 3, 4],
        [3, 5, 6]])

b.T
[8]
· array([[1, 2, 3],
        [2, 3, 4],
        [3, 5, 6]])

d = np.array([10,11,12,13,14,15])
d.reshape((2,3))
[2]
· array([[10, 11, 12],
        [13, 14, 15]])
```

**19. Flip method of array:**

- The `numpy.flip()` function reverses the order of array elements along the specified axis, preserving the shape of the array.

**CODE + OUTPUT:**

```
Reverse array

np.flip(a)
array([4, 3, 2, 2, 1])

np.flip(b,axis=1)
array([[3, 2, 1],
       [5, 3, 2],
       [6, 4, 3]])

np.flip(b,axis=0)
array([[3, 4, 6],
       [2, 3, 5],
       [1, 2, 3]])

b[:1] = np.flip(b[:1])

b
array([[6, 4, 3],
       [3, 2, 1],
       [6, 4, 3]])
```

## Practical – 2

**AIM:** Study of matplotlib library of Python. Write a program to demonstrate use of various functions.

### Matplotlib:

- Matplotlib is a low level graph plotting library in python that serves as a visualization utility.
- Matplotlib was created by John D. Hunter.
- Matplotlib is open source and we can use it freely.

#### ❖ Installation of Matplotlib:

- If you have Python and PIP already installed on a system, it will install the Matplotlib.
- Command:

```
➤ pip install matplotlib
```

#### ❖ Import Matplotlib:

- Once Matplotlib is installed, import it in your applications by adding the import keyword.
- Create an alias with the as keyword while importing:
- Command:

```
➤ import matplotlib
```

#### ❖ PyPlot:

- Import Matplotlib:
- Once Matplotlib is installed, import it in your applications by adding the import keyword.
- Create an alias with the as keyword while importing:
- Command:

```
➤ import matplotlib.pyplot
```

### Programs:

#### 1. Line Graph:

##### CODE:

```
import matplotlib.pyplot as plt
```

```
x = [1,2,3]
```

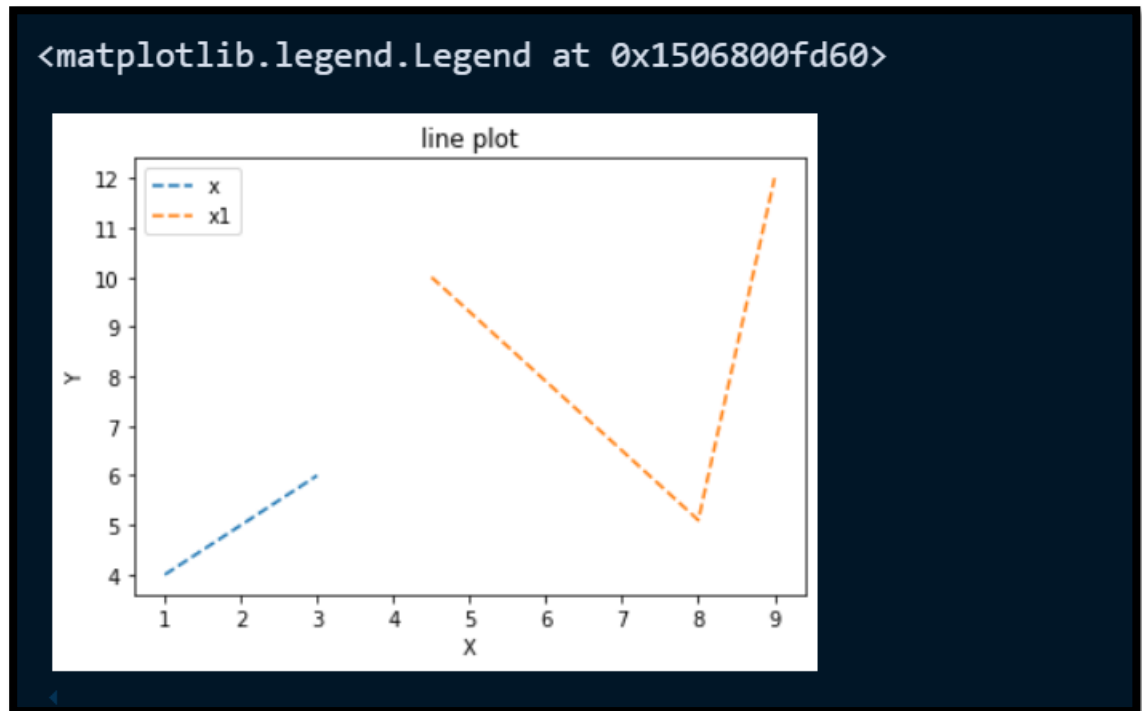
```
y = [4,5,6]
```

```
x1 = [4.5,8,9]
```

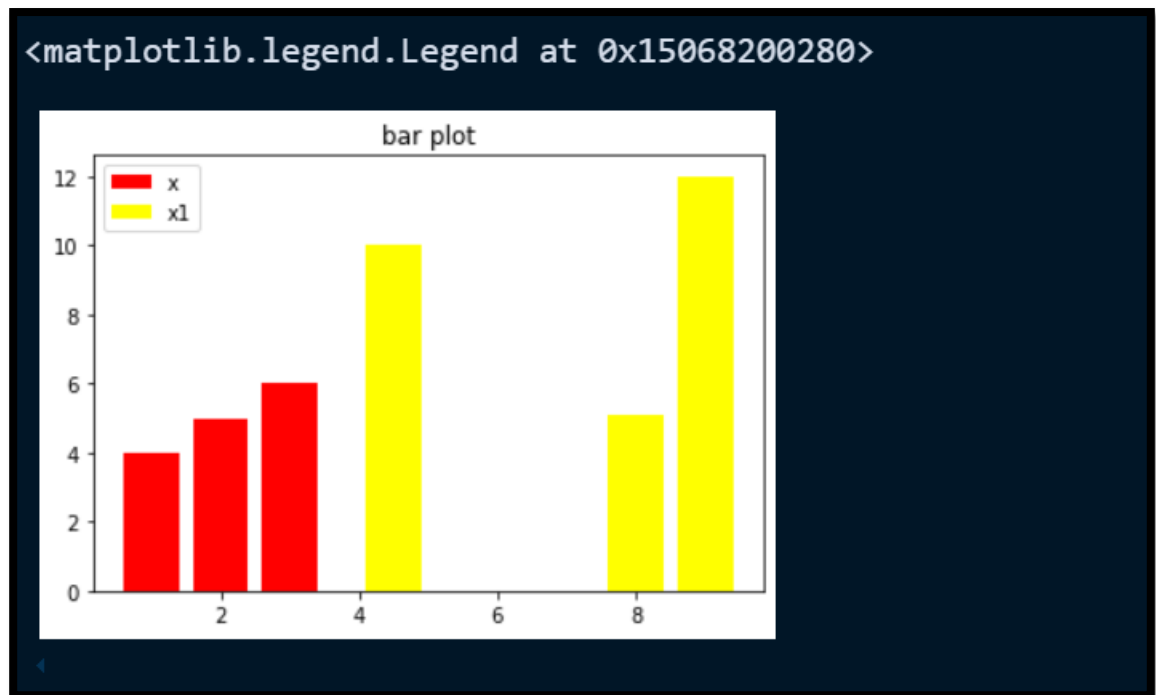
```
y1 = [10,5.1,12]
```



```
plt.plot(x,y,"--",x1,y1,"--")  
plt.title("line plot")  
plt.xlabel("X")  
plt.ylabel("Y")  
plt.legend(["x","x1"])
```

**OUTPUT:****2. Bars:****CODE:**

```
plt.bar(x,y,width=.8,color="red")  
plt.bar(x1,y1,width=0.8,color='yellow')  
plt.title("bar plot")  
plt.legend(["x","x1"])
```

**OUTPUT:****3. Histogram:**

- A histogram is a graph showing frequency distributions.
- It is a graph showing the number of observations within each given interval.

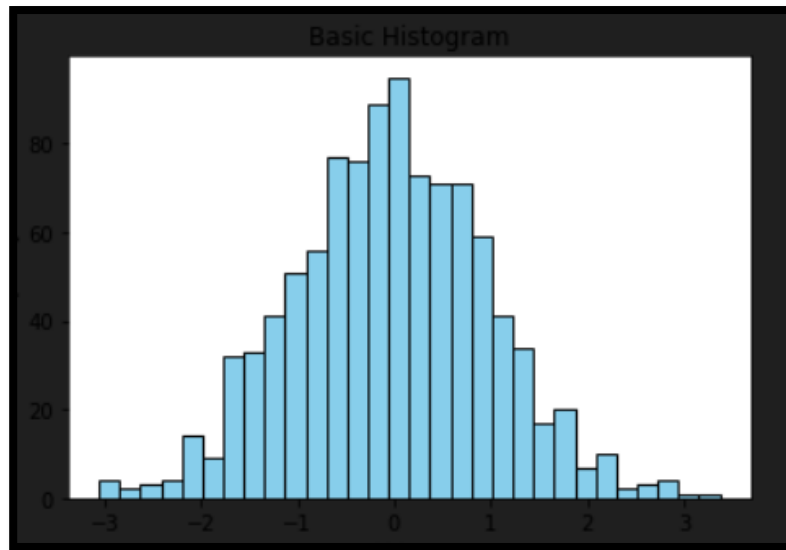
**CODE:**

```
# histogram char in python

data = np.random.randn(1000)

plt.hist(data, bins=30, color='skyblue', edgecolor='black')

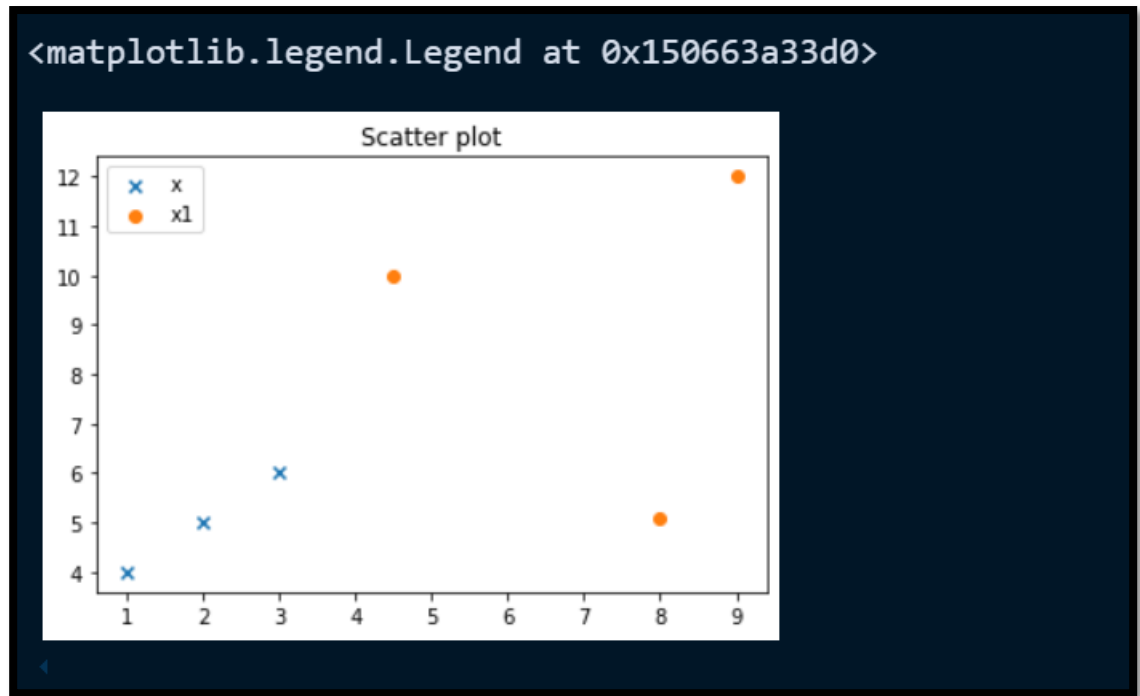
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('Basic Histogram')
plt.show()
```

**OUTPUT:****4. Scatter Plot:**

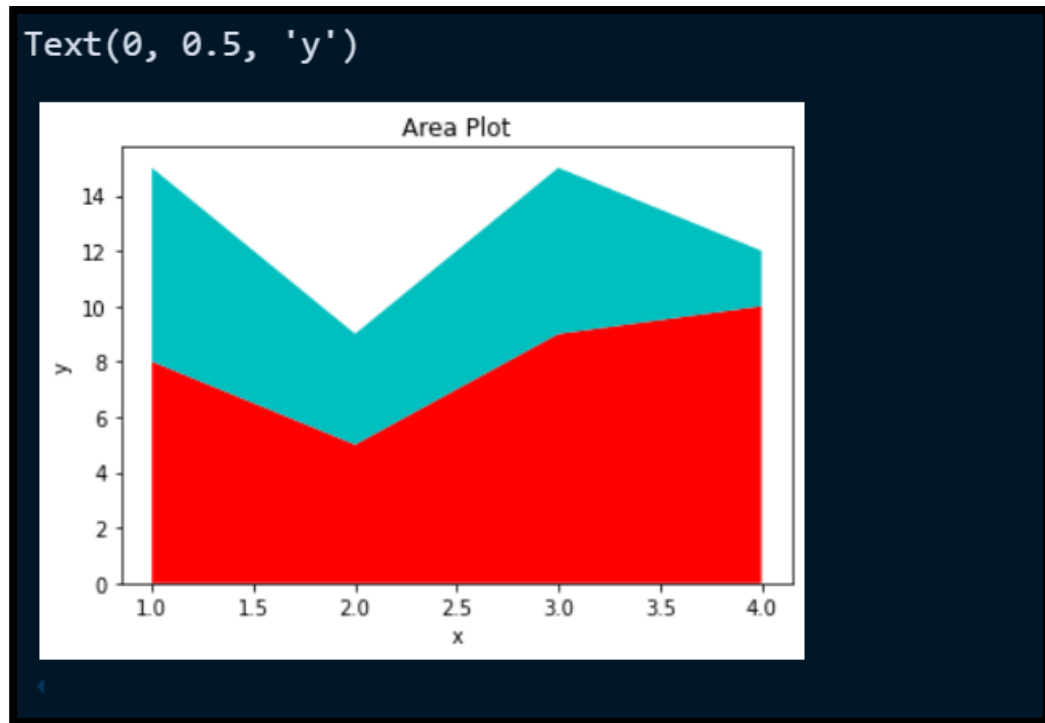
- With Pyplot, you can use the `scatter()` function to draw a scatter plot.
- The `scatter()` function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis:

**CODE:**

```
plt.scatter(x,y,marker="x")  
  
plt.scatter(x1,y1)  
  
plt.title("Scatter plot")  
  
plt.legend(["x","x1"])
```

**OUTPUT:****5. Area Chart:****CODE:**

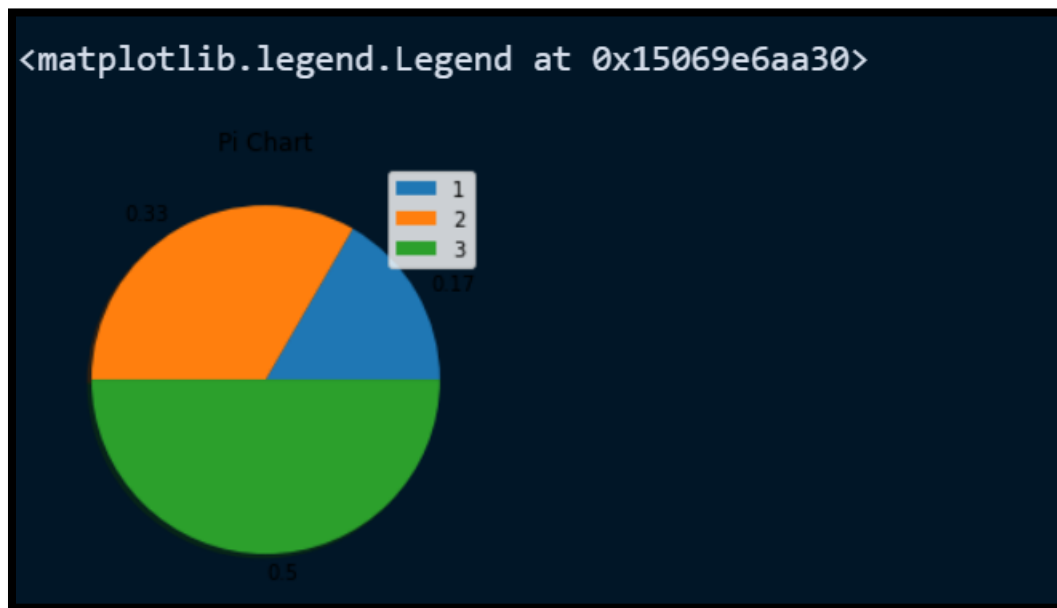
```
days = [1,2,3,4]
study = [8,5,9,10]
play = [7,4,6,2]
plt.stackplot(days,study,play,colors=['r','c'])
plt.title("Area Plot")
plt.xlabel("x")
plt.ylabel("y")
```

**OUTPUT:****6. Pie Chart:**

- With Pyplot, you can use the `pie()` function to draw pie charts:

**CODE:**

```
plt.pie(x,shadow=True,labels=[round(1/sum(x),2),round(2/sum(x),2),round(3/sum(x),2)])
plt.title("Pi Chart")
plt.legend([1,2,3])
```

**OUTPUT:**

## Practical – 3

**AIM:** Study of Pandas library of Python. Write a program to demonstrate its use.

### Pandas:

- Pandas is a Python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.
- Pandas allows us to analyze big data and make conclusions based on statistical theories.
- Pandas can clean messy data sets, and make them readable and relevant.

#### ❖ Installation of Pandas:

- If you have Python and PIP already installed on a system, it will install the pandas.
- Command:

```
 pip install pandas
```

#### ❖ Import Pandas:

- Once Pandas is installed, import it in your applications by adding the import keyword.
- Create an alias with the as keyword while importing:
- Command:

```
 import pandas as pd
```

### Programs:

#### 1. Read CSV file using pandas:

- A simple way to store big data sets is to use CSV files (comma separated files).
- CSV files contains plain text and is a well know format that can be read by everyone including Pandas.

#### CODE:

```
# read csv file in python
import pandas as pd

df = pd.read_csv('./LAB2CSV.csv')

# read top 5 samples
print("Top 5 rows of the dataset\n-----")
df.head()
```

**OUTPUT:**

Top 5 rows of the dataset

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProte
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	

5 rows x 21 columns

- Read\_csv function take one argument as **nrows** by which we can read the specific no. of rows(samples) from the dataset(CSV file).

**CODE:**

```
print("Get DataFrame just of 50 columns\n-----")
spe_df = pd.read_csv('./LAB2CSV.csv', nrows=50)

print("\n\nTop 5 rows of the dataset\n-----")
print(spe_df.head())

print("\n\nShape of the new dataframe\n-----")
spe_df.shape
```



**OUTPUT:**

```

Get DataFrame just of 50 columns
-----

Top 5 rows of the dataset
-----
   customerID  gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  \
0  7590-VHVEG  Female              0      Yes           No         1           No
1  5575-GNVDE   Male              0      No            No        34           Yes
2  3668-QPYBK   Male              0      No            No         2           Yes
3  7795-CFOCW   Male              0      No            No        45           No
4  9237-HQITU   Female            0      No            No         2           Yes

   MultipleLines  InternetService  OnlineSecurity  ...  DeviceProtection  \
0  No phone service              DSL              No  ...              No
1                No              DSL              Yes  ...              Yes
2                No              DSL              Yes  ...              No
3  No phone service              DSL              Yes  ...              Yes
4                No  Fiber optic              No  ...              No

   TechSupport  StreamingTV  StreamingMovies  ...  Contract  PaperlessBilling  \
0           No           No              No  ...  Month-to-month              Yes
1           No           No              No  ...    One year              No
2           No           No              No  ...  Month-to-month              Yes
3          Yes           No              No  ...    One year              No
...

Shape of the new dataframe
-----
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
(50, 21)

```

- Read\_csv function take one argument as **usecols** by which we can just read the specific columns from the dataset(CSV file).

**CODE:**

```

print("Get DataFrame just of 50 columns & 5 columns\n-----
---")
spe_df = pd.read_csv('./LAB2CSV.csv', nrows=50, usecols=['customerID', 'gender',
'Partner', 'InternetService', 'Contract'])

```

```
print("\n\nTop 5 rows of the dataset\n-----")
print(spe_df.head())

print("\n\nShape of the new dataframe\n-----")
spe_df.shape
```

**OUTPUT:**

```
Get DataFrame just of 50 columns & 5 columns
-----

Top 5 rows of the dataset
-----
```

	customerID	gender	Partner	InternetService	Contract
0	7590-VHVEG	Female	Yes	DSL	Month-to-month
1	5575-GNVDE	Male	No	DSL	One year
2	3668-QPYBK	Male	No	DSL	Month-to-month
3	7795-CFOCW	Male	No	DSL	One year
4	9237-HQITU	Female	No	Fiber optic	Month-to-month

```

Shape of the new dataframe
-----

(50, 5)
```

**2. Shape of dataset(dataframe):**

- The shape property returns a tuple containing the shape of the DataFrame.
- The shape is the number of rows and columns of the DataFrame

**CODE:**

```
print("Check shape of the dataset\n-----")
df.shape
```

**OUTPUT:**

```
Check shape of the dataset
```

```
-----
```

```
(7043, 21)
```

**3. Get Columns of dataset(dataframe):**

- This attribute does not require any parameters to be passed. When called on a data frame using the syntax **DataFrame.columns**, it returns the names of the columns present in that data frame.

**CODE:**

```
print("Get Columns of the dataset\n-----")
df.columns
```

**OUTPUT:**

```
Get Columns of the dataset
```

```
-----
```

```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
      'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
      'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
      'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
      'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

**4. Head function on dataset(dataframe):**

- The head() method returns a specified number of rows, string from the top.
- The head() method returns the first 5 rows if a number is not specified.

**CODE:**

```
print("Get DataFrame just of 5 columns")
spe_df = pd.read_csv('./LAB2CSV.csv', usecols=['customerID', 'gender', 'Partner',
      'InternetService', 'Contract'])
```

```
print("\n\nTop 5 rows of the dataset\n-----")
print(spe_df.head())

print("\n\nShape of the new dataframe\n-----")
spe_df.shape
```

**OUTPUT:**

Get DataFrame just of 5 columns

Top 5 rows of the dataset

```
-----
      customerID  gender  Partner  InternetService      Contract
0  7590-VHVEG   Female    Yes          DSL  Month-to-month
1  5575-GNVDE    Male     No          DSL    One year
2  3668-QPYBK    Male     No          DSL  Month-to-month
3  7795-CFOCW    Male     No          DSL    One year
4  9237-HQITU   Female    No  Fiber optic  Month-to-month
```

Shape of the new dataframe

```
-----
(7043, 5)
```

**5. Drop function of pandas:**

- The drop() method removes the specified row or column.
- By specifying the column axis (axis='columns'), the drop() method removes the specified column.
- By specifying the row axis (axis='index'), the drop() method removes the specified row.

**CODE:**

```
print("Get shape of th data frame\n-----")
new_df = df.drop(['gender', 'Partner'], axis = 1) # axis 1 : for delete column
# new_df = df.drop(['gender', 'Partner'], axis = 0) # axis 0 : for delete rows
# new_df = df.drop(Partner', axis = 1) # delete the single column for multiple column
can pass list
```

```
# new_df = df.drop(['gender', 'Partner'], axis = 1, inplace = True) # inplace True :
# means original data frame update by default False
new_df
```

**OUTPUT:**

```
Get DataFrame just of 500 columns & 5 columns
```

```
-----
```

	customerID	gender	Partner	tenure	InternetService
0	7590-VHVEG	Female	Yes	1	DSL
1	5575-GNVDE	Male	No	34	DSL
2	3668-QPYBK	Male	No	2	DSL
3	7795-CFOCW	Male	No	45	DSL
4	9237-HQITU	Female	No	2	Fiber optic
...	...	...	...	...	...
495	8205-OTCHB	Male	No	22	DSL
496	4134-BSXLX	Male	Yes	28	DSL
497	0505-SPOOW	Female	Yes	70	No
498	6235-VDHOM	Female	No	5	DSL
499	7783-YKGDV	Female	No	12	Fiber optic

```
500 rows × 5 columns
```

```
Get shape of th data frame
```

```
-----
```

```
(500, 5)
```

Get shape of th data frame

---

	customerID	tenure	InternetService
0	7590-VHVEG	1	DSL
1	5575-GNVDE	34	DSL
2	3668-QPYBK	2	DSL
3	7795-CFOCW	45	DSL
4	9237-HQITU	2	Fiber optic
...	...	...	...
495	8205-OTCHB	22	DSL
496	4134-BSXLX	28	DSL
497	0505-SPOOW	70	No
498	6235-VDHOM	5	DSL
499	7783-YKGDV	12	Fiber optic

500 rows × 3 columns

#### 6. Sample function of pandas:

- Pandas sample() is used to generate a sample random row or column from the function caller data frame.

#### CODE:

```
print("Get 10 samples from data frame\n-----")
df.sample(n = 10)
```

**OUTPUT:**

```
Get 10 samples from data frame
```

---

	<b>customerID</b>	<b>gender</b>	<b>Partner</b>	<b>tenure</b>	<b>InternetService</b>
115	3071-VBYPO	Male	Yes	3	Fiber optic
372	6122-EFVKN	Male	No	24	DSL
27	8665-UTDHZ	Male	Yes	1	DSL
418	0578-SKVMF	Female	Yes	22	Fiber optic
274	5940-AHUHD	Male	No	1	Fiber optic
325	4983-CLMLV	Female	Yes	52	Fiber optic
176	2656-FMOKZ	Female	No	15	Fiber optic
369	3520-FJGCV	Male	Yes	72	Fiber optic
142	1095-WGNNG	Female	Yes	61	Fiber optic
44	4080-IIARD	Female	Yes	13	DSL

- Here one more argument can be pass as frac it is optional argument in .sample().
- It return fraction of rows, like 0.5 for 50% of the rows

**CODE:**

```
print("Get 0.1 fractional samples of the data frame\n-----")
----")
df.sample(frac = 0.1)
```

**OUTPUT:**

```
Get 0.1 fractional samples of the data frame
-----
```

	customerID	gender	Partner	tenure	InternetService
333	0122-OAHPZ	Female	No	7	Fiber optic
160	8992-VONJD	Female	No	13	DSL
211	4195-NZGTA	Female	No	1	DSL
278	8645-KWHJO	Male	No	14	DSL
201	8544-GOQSH	Female	No	14	Fiber optic
190	7100-FQPRV	Male	Yes	43	DSL
227	2232-DMLXU	Female	Yes	1	DSL
90	4767-HZZHQ	Male	Yes	30	Fiber optic
300	0895-LMRSF	Male	No	23	DSL
441	5628-RKIFK	Female	No	49	Fiber optic
329	2739-CACDQ	Female	No	17	Fiber optic
21	1680-VDCWW	Male	Yes	12	No
444	1910-FMXJM	Female	Yes	36	Fiber optic
455	0242-NDQIA	Female	No	24	Fiber optic

**7. Count function of the pandas:**

- The count() method counts the number of not empty values for each row, or column if you specify the axis parameter as axis='columns', and returns a Series object with the result for each row (or column).

**CODE:**

```
print("Get 0.1 fractional samples(50 samples) of the data frame\n-----")
df.sample(frac = 0.1).count()
```



**OUTPUT:**

```
Get 0.1 fractional samples(50 samples) of the data frame
-----
customerID      50
gender          50
Partner         50
tenure          50
InternetService 50
dtype: int64
```

**8. Isna function of the pandas:**

- Pandas dataframe.isna() function is used to detect missing values. It return a boolean same-sized object indicating if the values are NA. NA values, such as None or numpy.NaN, gets mapped to True values.
- Everything else gets mapped to False values. Characters such as empty strings "" or numpy.inf are not considered NA values.

**CODE:**

```
print("Get the null values in this data frame\n-----")
df.isna()
```

**OUTPUT:**

Get the null values in this data frame

-----

	customerID	gender	Partner	tenure	InternetService
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
495	False	False	False	False	False
496	False	False	False	False	False
497	False	False	False	False	False
498	False	False	False	False	False
499	False	False	False	False	False

500 rows × 5 columns

**9. Sum function of pandas:**

- The sum() method adds all values in each column and returns the sum for each column.
- By specifying the column axis (axis='columns'), the sum() method searches column-wise and returns the sum of each row.

**CODE:**

```
print("For Better visualization we count total null values in column\n-----")
df.isna().sum()
```

**OUTPUT:**

```

For Better visualization we count total null values in column
-----

customerID      0
gender          0
Partner         0
tenure          0
InternetService 0
dtype: int64

```

**CODE:**

```

print("Total null values from the full data frame\n-----")
df.isna().sum().sum()

```

**OUTPUT:**

```

Total null values from the full data frame
-----

0

```

**10. Loc[] of pandas:**

- Pandas DataFrame.loc attribute accesses a group of rows and columns by label(s) or a boolean array in the given DataFrame.

**CODE:**

```

import numpy as np

print("For access sample value to add null by label data frame\n-----")

missing_val = np.random.randint(500, size=10)

```

```
df.loc[missing_val, ['gender', 'tenure']] = np.nan
df.isna().sum()
```

**OUTPUT:**

```
For access sample value to add null by label data frame
```

```
-----

customerID      0
gender          10
Partner         0
tenure          10
InternetService 0
dtype: int64
```

**11. Iloc[] of pandas:**

- The iloc property gets, or sets, the value(s) of the specified indexes.
- Specify both row and column with an index.
- To access more than one row, use double brackets and specify the indexes, separated by commas:
- `df.iloc[[0, 2]]`
- Specify columns by including their indexes in another list:
- `df.iloc[[0, 2], [0, 1]]`
- You can also specify a slice of the DataFrame with from and to indexes, separated by a colon:
- `df.iloc[0:2]`

**CODE:**

```
print("For access sample value to add null by index of data frame\n-----")

missing_val = np.random.randint(500, size=10)
df.iloc[missing_val, -1] = np.nan
df.isna().sum()
```

**OUTPUT:**

```
For access sample value to add null by index of data frame
```

```
-----
customerID      0
gender          10
Partner         0
tenure          10
InternetService 10
dtype: int64
```

**CODE:**

```
print("Total null values from the full data frame\n-----
-----")
df.isna().sum().sum()
```

**OUTPUT:**

```
Total null values from the full data frame
```

```
-----
30
```

**12. Value counts of pandas:**

- Pandas Index.value\_counts() function returns object containing counts of unique values. The resulting object will be in descending order so that the first element is the most frequently-occurring element. Excludes NA values by default.

**CODE:**

```
print("Find that values and occurence in column of data frame\n-----
-----")
df['gender'].value_counts()
```

**OUTPUT:**

```
Find that values and occurrence in column of data frame
-----
Female      251
Male        239
Name: gender, dtype: int64
```

**13. Fillna function of pandas:**

- The fillna() method replaces the NULL values with a specified value.
- The fillna() method returns a new DataFrame object unless the inplace parameter is set to True, in that case the fillna() method does the replacing in the original DataFrame instead.

**CODE:**

```
print("Replace the null values of column with specific value in data frame\n-----")
mode = df['gender'].value_counts()[0]
df['gender'].fillna(mode, inplace = True)

print("\n\nFor Better visualization we count total null values in column\n-----")
df.isna().sum()
```

**OUTPUT:**

```
Replace the null values of column with specific value in data frame
-----

For Better visualization we count total null values in column
-----

customerID      0
gender          0
Partner         0
tenure          10
InternetService 10
dtype: int64
```

**14. Mean function of pandas:**

- The mean() method returns a Series with the mean value of each column.
- By specifying the column axis (axis='columns'), the mean() method searches column-wise and returns the mean value for each row.

**CODE:**

```
print("Find that values and occurence in column of data frame\n-----")
df['tenure'].value_counts().mean()
```

**OUTPUT:**

```
Find that values and occurence in column of data frame
-----

6.805555555555555
```

**CODE:**

```
print("Replace the null values of column with specific value in data frame\n-----  
-----")  
mode = df['tenure'].value_counts().mean()  
df['tenure'].fillna(mode, inplace = True)  
  
print("\n\nFor Better visualization we count total null values in column\n-----  
-----")  
df.isna().sum()
```

**OUTPUT:**

```
Replace the null values of column with specific value in data frame  
-----  
  
For Better visualization we count total null values in column  
-----  
  
customerID      0  
gender          0  
Partner         0  
tenure          0  
InternetService 10  
dtype: int64
```

**CODE:**

```
print("Find that values and occurence in column of data frame\n-----  
-----")  
df['InternetService'].value_counts()
```

**OUTPUT:**

```
Find that values and occurence in column of data frame  
-----  
  
Fiber optic    224  
DSL            172  
No             94  
Name: InternetService, dtype: int64
```



**15. Dropna function of pandas:**

- Pandas is one of the packages that makes importing and analyzing data much easier. Sometimes CSV file has null values, which are later displayed as NaN in Pandas DataFrame. Pandas dropna() method allows the user to analyze and drop Rows/Columns with Null values in different ways.

**CODE:**

```
print("Drop null value contained samples from data frame\n-----")

df.dropna(axis = 0, how = 'any', inplace = True)
df.isna().sum()
```

**OUTPUT:**

```
Drop null value contained samples from data frame
-----

customerID      0
gender           0
Partner         0
tenure           0
InternetService 0
dtype: int64
```

**16. Condition access in dataframe of pandas:****CODE:**

```
print("Condition access dataframe samples\n-----")

new_val = df[(df['tenure'] > 60) & (df['tenure'] < 65)]
new_val
```

**OUTPUT:**

Condition access dataframe samples					
	customerID	gender	Partner	tenure	InternetService
9	6388-TABGU	Male	No	62.0	DSL
43	4671-VJLCL	Female	No	63.0	DSL
56	8769-KKTPH	Female	Yes	63.0	Fiber optic
72	1891-QRQSA	Male	Yes	64.0	Fiber optic
73	8028-PNXHQ	Male	Yes	62.0	No
114	5256-SKJGO	Female	Yes	64.0	DSL
142	1095-WGNNG	Female	Yes	61.0	Fiber optic
143	2636-SJDOU	Male	No	64.0	Fiber optic
179	3712-PKXZA	Male	Yes	61.0	No
209	1347-KTTTA	Male	Yes	64.0	Fiber optic

**17. Groupby function of pandas:**

- Pandas dataframe.groupby() function is used to split the data into groups based on some criteria. Pandas objects can be split on any of their axes. The abstract definition of grouping is to provide a mapping of labels to group names.

**CODE:**

```
new_val = df.groupby(['gender', 'tenure'])
new_val.first()
```

**OUTPUT:**

		customerID	Partner	InternetService
gender	tenure			
251	6.805556	3192-NQECA	Yes	Fiber optic
Female	0.000000	4472-LVYGI	Yes	DSL
	1.000000	7590-VHVEG	Yes	DSL
	2.000000	9237-HQITU	No	Fiber optic
	3.000000	5122-CYFXA	No	DSL
...	...	...	...	...
Male	68.000000	4322-RCYMT	Yes	DSL
	69.000000	6345-FZOQH	Yes	No
	70.000000	4686-GEFRM	Yes	Fiber optic
	71.000000	9959-WOFKT	No	Fiber optic
	72.000000	5248-YGIJN	Yes	DSL

133 rows × 3 columns




## Practical – 3

**AIM:** Write a program to demonstrate working of Linear Regression Algorithm without using scikit learn library. Implement the same algorithm using scikit learn and compare the results.

### Linear Regression Algorithm:

- Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation to observed data.
- When there is only one independent feature, it is known as Simple Linear Regression, and when there are more than one feature, it is known as Multiple Linear Regression.

#### ❖ Import Libraries:

- Command:  
 **import numpy as np**  
 **import pandas as pd**  
 **import matplotlib.pyplot as plt**

### Programs:

#### 1. Without Scikit learn:

##### CODE:

```
def estimate_coe(x, y):  
    n = np.size(x)  
  
    mean_x = np.mean(x)  
    mean_y = np.mean(y)  
  
    print("Mean of the X values : ", mean_x)  
    print("Mean of the Y values : ", mean_y)  
  
    sum_xy = 0  
    sum_x = 0  
  
    for i in range(n):  
        sum_xy += ((x[i] - mean_x) * (y[i] - mean_y))  
        sum_x += (x[i] - mean_x)**2  
  
    print("\n(x - x')(y - y') : ", sum_xy)
```

```
print("(x - x\') : ", sum_x)

slop = (sum_xy / sum_x)
constant = mean_y - slop * mean_x

return (slop, constant)

def plot_graph(x, y, ans):
    plt.scatter(x, y, color="m", marker = 'o', s = 30)

    predicted_y = ans[1] + ans[0]*x

    plt.plot(x, predicted_y, color='g')

    plt.xlabel('X Values')
    plt.ylabel('Y Values')

    plt.show()

def main():
    print("Linear Regression\n-----\n\n")
    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])

    print("Input Values\n-----")
    print("x : ", x)
    print("y : ", y)

    print("\n\nCorrelation Estimation\n-----")
    ans = estimate_coe(x, y)

    print("\nSlop of the input : ", ans[0])
    print("Constant of the input : ", ans[1])

    print("\n-----\nPlot Line :\n-----")
    plot_graph(x, y, ans)

main()
```

**OUTPUT:****Linear Regression**

-----

**Input Values**

-----

x : [0 1 2 3 4 5 6 7 8 9]

y : [ 1 3 2 5 7 8 8 9 10 12]

**Correlation Estimation**

-----

Mean of the X values : 4.5

Mean of the Y values : 6.5

 $(x - x')(y - y') : 96.5$  $(x - x') : 82.5$ 

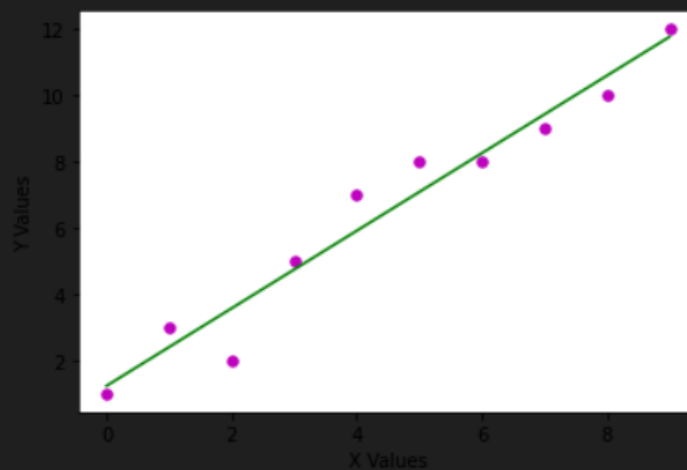
Slop of the input : 1.1696969696969697

Constant of the input : 1.2363636363636363

-----

**Plot Line :**

-----



## 2. With Scikit learn built in methods:

### CODE:

```
# linear regression with using built-in function of sklearn

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

print("Linear Regression\n-----\n\n")
df = pd.read_csv("./salary_data.csv")

print("Data From CSV File\n-----")
print(df.head(3))

x = df.iloc[:, :-1].values
y = df.iloc[:, 1].values

print("\n\nSplit data into 2 parts(Train Data, Test Data)\n-----")
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=1/3, random_state=0)

regressor = LinearRegression()

regressor.fit(x_train, y_train)

y_pred = regressor.predict(x_test)
y_train_pred = regressor.predict(x_train)

plt.scatter(x_train, y_train, color="g")
plt.scatter(x_train, y_test_pred, color="red")
plt.title("Training Dataset")
plt.xlabel('Years Of Experience')
plt.ylabel('Salary(In Rypees)')

plt.show()
```

**OUTPUT:**

## Linear Regression

### Data From CSV File

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0

### Split data into 2 parts(Train Data, Test Data)

