23261A6660 24/07/2025

EXPERIMENT – 1

AIM :

Compute Central Tendency Measures and Measure of Dispersion

```
import statistics as st
from math import isnan
from itertools import filterfalse

data = [20.7,float('NaN'), 19.2, 18.3, float('NaN'), 14.4]
print(data)
clean = list(filterfalse(isnan, data))
print("Cleaned data:", clean)
print('median =', st.median(clean))
print('median low =', st.median_low(clean))
print('median high =', st.median_high(clean))
print(f'{st.mode([1, 1, 2, 3, 3, 3, 4])}')
print(f'{st.mode(["red", "blue", "blue", "red", "green", "red", "red"])}')
print(f'{st.multimode('aabbbbbccddddeeeffffgg')}')

data1 = [6,7,10,13,14,14,18,19,22,24]
print("variance =", st.variance(data))
mean = st.mean(data1)
print('variance=',st.variance(data1,mean))
X = [3,5,6,6,7,8,12,14,15,19]
Y = [6,7,7,13,16,15,17,20,24,27]
print('covariance = ',st.covariance(X,Y))
print('correlation = ',st.correlation(X,Y))
```

```
[20.7, nan, 19.2, 18.3, nan, 14.4]
Cleaned data: [20.7, 19.2, 18.3, 14.4]
median = 18.75
median low = 18.3
median high = 19.2
3
red
['b', 'd']
variance = nan
variance= 36.67777777777778
covariance =  35.333333333333336
correlation =  0.9443522018799386
```

```
import numpy as np
import statistics as st

x=[33,34,35,35,36,37,38,39,40]
y=[41,42,43,44,45,46,47,48,49]
print('covarience=', np.cov(x,y))

x1=[3.63, 3.82, 3.82, 3.42, 3.59, 2.87, 3.03, 3.46, 3.36, 3.3]
y1=[58.2, 49.4, 48.45, 54.28, 54.9, 43.7, 47.2, 45.2, 54.4, 50.4]
print("correlation=", np.correlate(x1,y1))
```

```
covarience= [[5.5   6.375]
 [6.375 7.5  ]]
correlation= [1741.7126]
```

EXPERIMENT – 2

AIM:

Study of Python Basic Libraries: Statistics, Math, NumPy, and SciPy

```
import numpy as np
arr = np.array([11,42,35])
print("Array with Rank 1: \n",arr)
arr2 = np.array([[1,2,3],[4,5,6],[7,8,9]])
print("Array with Rank 2: \n",arr2)
arr3 = np.array((1,6,4))
print("Array created using passed tuple:\n",arr3)
```

```
Array with Rank 1:
 [11 42 35]
Array with Rank 2:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Array created using passed tuple:
 [1 6 4]
```

```
arr = np.array([[-2, 3, -4, 1],
 [ 4, -1, 2, 0],
 [ 3, -3, 1, -2],
 [-1, 5, 0, 4]])
print("Initial array:\n",arr)
sliced_arr = arr[:2,::2]
print("Array with first 2 rows and alternate columns(0 and 2):\n",sliced_arr)
Index_arr = arr[[1, 2, 0, 3],[2,1,0,1]]
print("Elements at indices(1,2),(2,1),(0,0),(3,1):\n",Index_arr)
```

```
Initial array:
 [[-2  3 -4  1]
 [ 4 -1  2  0]
 [ 3 -3  1 -2]
 [-1  5  0  4]]
Array with first 2 rows and alternate columns(0 and 2):
 [[-2 -4]
 [ 4  2]]
Elements at indices(1,2),(2,1),(0,0),(3,1):
 [ 2 -3 -2  5]
```

```
a = np.array([[1,2],[3,4]])
b = np.array([[5,7],[9,8]])
print("Adding 1 to every element:\n",a+1)
print("Subtracting 2 from each element:\n",b-2)
```

```
Adding 1 to every element:
 [[2 3]
 [4 5]]
Subtracting 2 from each element:
 [[3 5]
 [7 6]]
```

[4]:
```
s = "MACHINE LEARNING"
print(s)
print(s[6])
print(s[-1])
print(s[-3])
print(s[1:4])
print(s[-5:-1])
print(s[-2:-7:-2])
```

```
MACHINE LEARNING
E
G
I
ACH
RNIN
NNA
```

```
x = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
a1 = x.flatten()
a1[2] = 33
print(x)
print(a1)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[ 1  2 33  4  5  6  7  8  9 10 11 12]
```

```
a2 = x.ravel()
a2[0] = 33
print(x)
print(a2)
```

```
[[33  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[33  2  3  4  5  6  7  8  9 10 11 12]
```

```
print(np.zeros((2,4)))
print(np.ones((3,4,2)))
print(np.empty((3,4)))
np.arange(10,30,5)
```

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]]
[[[1. 1.]
  [1. 1.]
  [1. 1.]
  [1. 1.]]

 [[1. 1.]
  [1. 1.]
  [1. 1.]
  [1. 1.]]

 [[1. 1.]
  [1. 1.]
  [1. 1.]
  [1. 1.]]]
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
array([10, 15, 20, 25])
```

```
a = np.array([20,30,40,50])
b = np.arange(4)
print('a =',a)
print('b =',b)
c = a - b
print('c =',c)
print('b**2 =',b**2)
print("a<35 =",a<35)
```

```
a = [20 30 40 50]
b = [0 1 2 3]
c = [20 29 38 47]
b**2 = [0 1 4 9]
a<35 = [ True  True False False]
```

```
a = np.arange(8)
print(a)
b = np.arange(12).reshape(4,3)
print(b)
c = np.arange(24).reshape(2,3,4)
print(c)
```

```
[0 1 2 3 4 5 6 7]
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

```
A = np.array([[1,1],[0,1]])
B = np.array([[2,0],[3,4]])
print(A*B)
print(A@B)
```

```
[[2 0]
 [0 4]]
[[5 4]
 [3 4]]
```

#SciPy Implementation

```
from scipy import special as sp
a = sp.exp10(2)
b = sp.exp10(3)
c = sp.sindg(90)
print("a=", a)
print("b=", b)
print("c=", c)
```

```
a= 100.0
b= 1000.0
c= 1.0
```

```python
from scipy import linalg
import numpy as np
matrix = np.array([[1, 2, 3], [3, 5, 5], [6, 4, 7]])
print(matrix)
linalg.det(matrix)
linalg.inv(matrix)
arr = np.array([[5, 4], [6, 3]])
eg_val, eg_vect = linalg.eig(arr)
print("eg_val:", eg_val)
print("eg_vect:\n", eg_vect)
```

```
[[1 2 3]
 [3 5 5]
 [6 4 7]]
eg_val: [ 9.+0.j -1.+0.j]
eg_vect:
 [[ 0.70710678 -0.5547002 ]
 [ 0.70710678  0.83205029]]
```

```python
from scipy import integrate
import numpy as np
from math import sqrt
a = lambda x: x**2
inte = integrate.quad(a, 0, 1)
print("inte\n", inte)
f = lambda x, y: 64 * x * y
p = lambda x: 0
q = lambda y: sqrt(1 - 2 * y**2)
integration = integrate.dblquad(f, 0, 2 / 4, p, q)
print("integration\n", integration)
```

```
inte
 (0.3333333333333333, 3.700743415417189e-15)
integration
 (3.0, 9.657432734515774e-14)
```

23261A6660 07/08/2025

EXPERIMENT – 3

AIM:

Study of Python Libraries for ML Applications: Pandas and Matplotlib

```
import pandas as pd
s = pd.Series(["Jpn","Eng","Frc","Spn"])
type(s)
```

```
pandas.core.series.Series
```

```
s = pd.Series(["Jpn","Eng","Frc","Spn"],index = ["a","b","c","d"])
s
```

```
a    Jpn
b    Eng
c    Frc
d    Spn
dtype: object
```

```
EmployeeData = {
    'ID' : [101,102,103],
    'name' : ['Daniel','Butler','Morgan'],
    'age' : [34,32,29],
    'city' : ['Goa','Agra','Delhi']
}
empDB = pd.DataFrame(EmployeeData)
empDB
```

	ID	name	age	city
0	101	Daniel	34	Goa
1	102	Butler	32	Agra
2	103	Morgan	29	Delhi

```
empDB = pd.DataFrame(EmployeeData,index = ['a','b','c'])
print(empDB)
```

```
    ID    name  age   city
a  101  Daniel   34    Goa
b  102  Butler   32   Agra
c  103  Morgan   29  Delhi
```

```
ColumnData = ['ID','Emp','Salary','Exp']
df = pd.DataFrame(columns = ColumnData)
df
```

ID	Emp	Salary	Exp

```
df1 = pd.DataFrame()
df1['id'] = [1,2,3]
df1['emp'] = ['A','B','C']
df1['salary'] = [1000,2000,3000]
df1['exp'] = [2,3,4]
df1
```

	id	emp	salary	exp
0	1	A	1000	2
1	2	B	2000	3
2	3	C	3000	4

```
df3 = pd.DataFrame(columns = ['emp','salary','grade'],index = ['a','b','c'])
df3
```

	emp	salary	grade
a	NaN	NaN	NaN
b	NaN	NaN	NaN
c	NaN	NaN	NaN

```
df3['emp'] = ['Michael','Lorenzo','Alex']
df3
```

	emp	salary	grade
a	Michael	NaN	NaN
b	Lorenzo	NaN	NaN
c	Alex	NaN	NaN

```
df = pd.read_csv("Iris.csv")
df.head(4)
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa

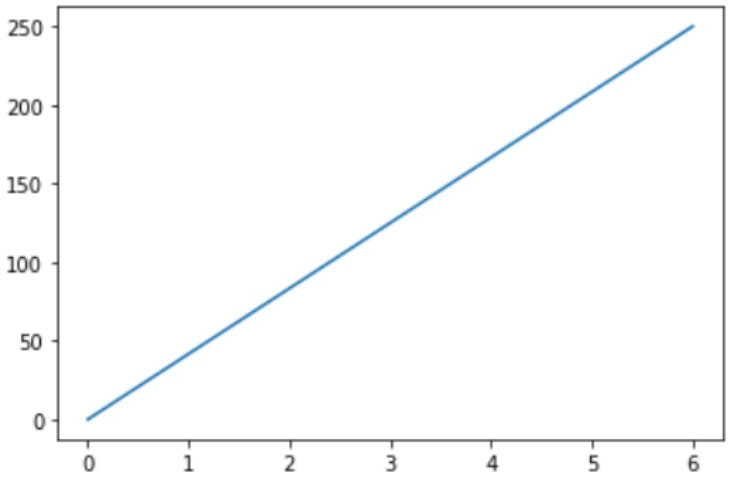
23261A6660 14/08/2025

EXPERIMENT – 3

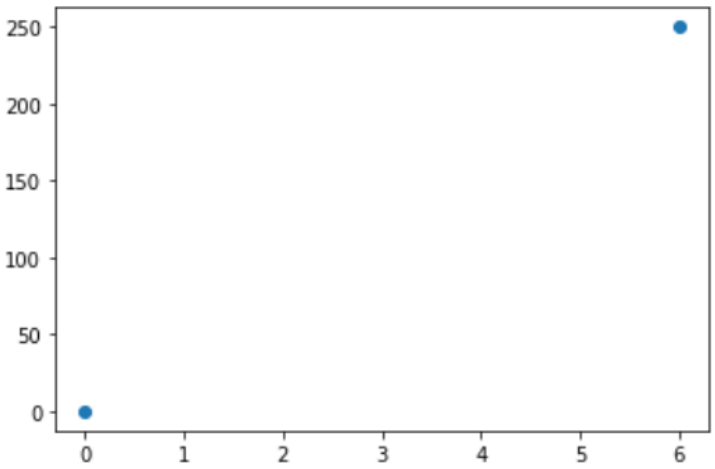
AIM:

Study of Python Libraries for ML Applications: Pandas and Matplotlib

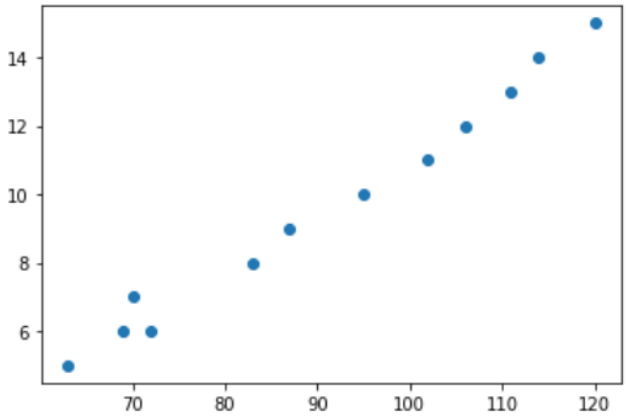
[14]:
```
xpoints = np.array([0,6])
ypoints = np.array([0,250])
plt.plot(xpoints,ypoints)
plt.show()
```



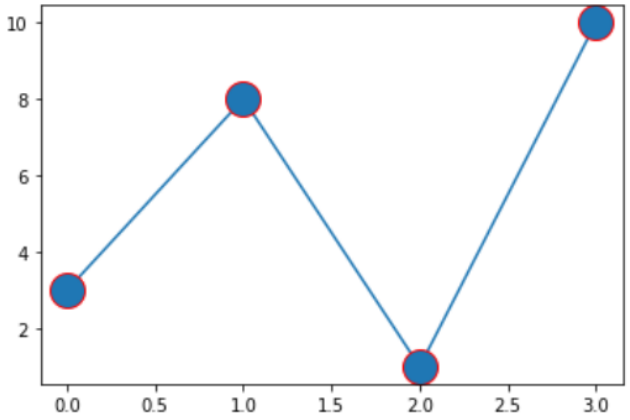
[15]:
```
xpoints = np.array([0,6])
ypoints = np.array([0,250])
plt.plot(xpoints,ypoints,'o')
plt.show()
```



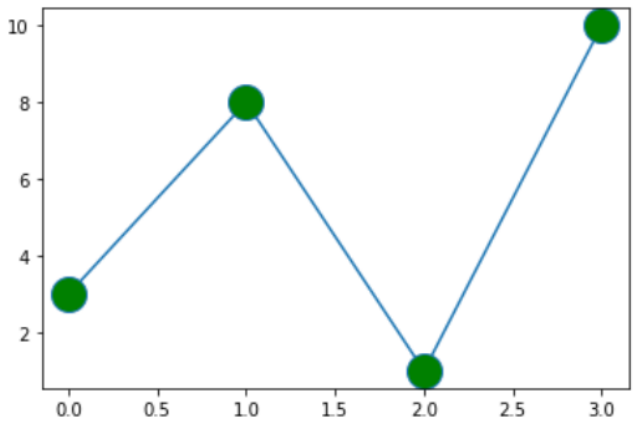
[16]:
```
xpoints = np.array([63, 69, 72, 70, 83, 87, 95, 102, 106, 111, 114, 120])
ypoints = np.array([5, 6, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])
plt.plot(xpoints,ypoints,'o')
plt.show()
```



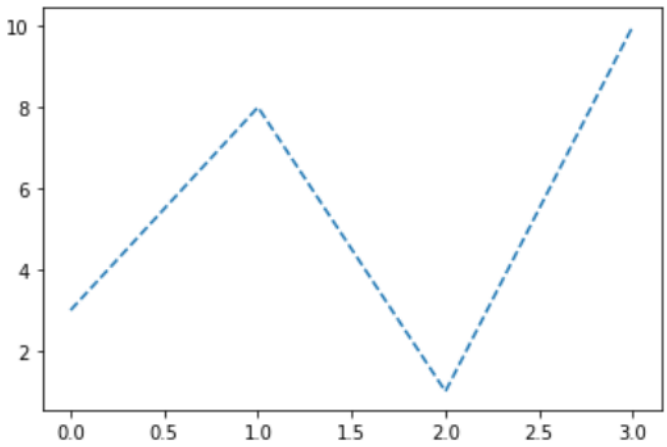
[19]:
```
plt.plot(ypoints,marker = 'o',ms = 20,mec = 'r')
plt.show()
```



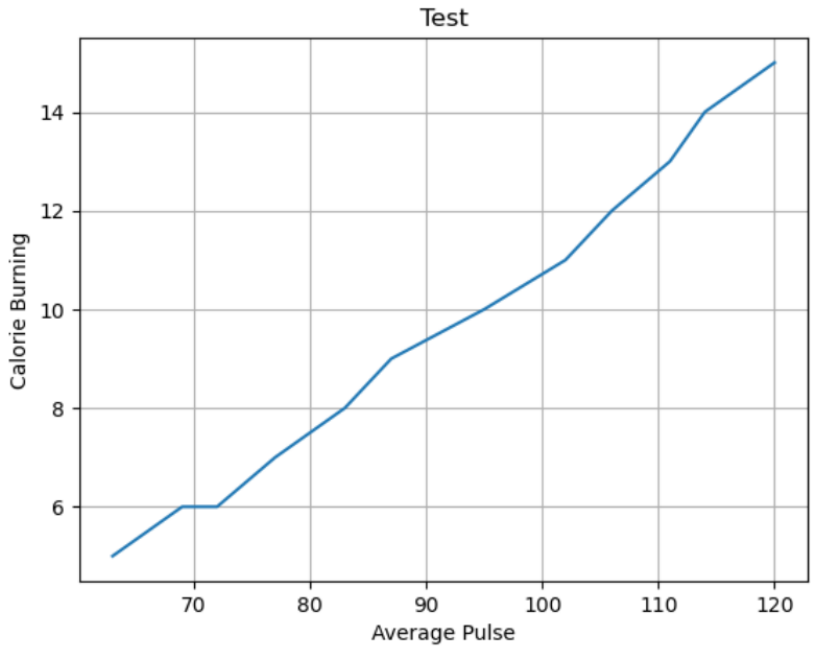
[20]:
```
plt.plot(ypoints,marker = 'o',ms = 20,mfc = 'g')
plt.show()
```



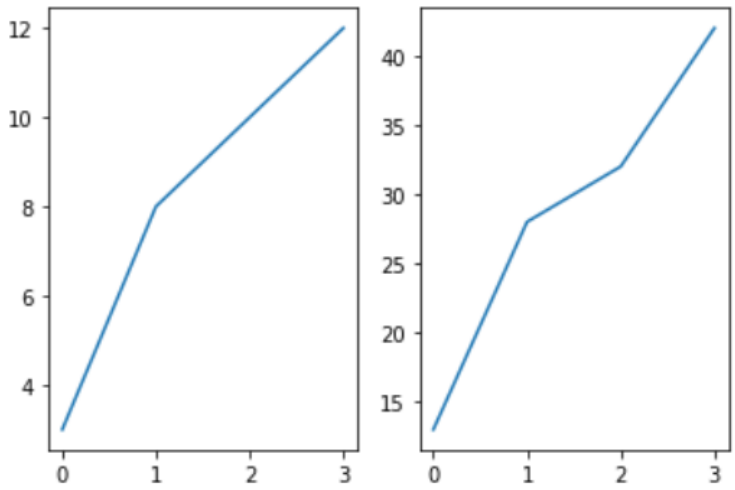
[22]:
```
plt.plot(ypoints,linestyle = 'dashed')
plt.show()
```



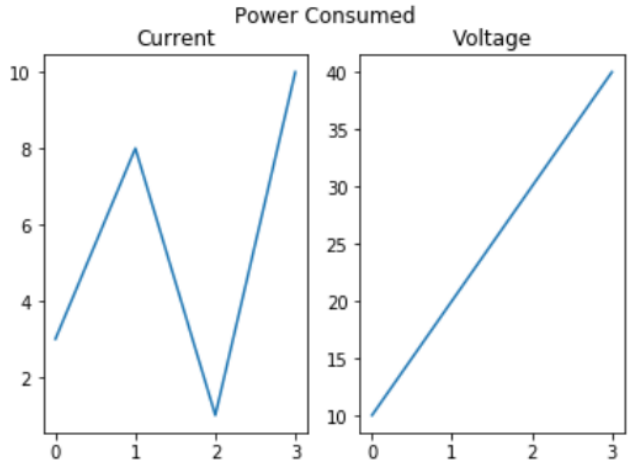
[27]:
```
y = np.array([63, 69, 72, 77, 83, 87, 95, 102, 106, 111, 114, 120])
x = np.array([5, 6, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burning")
plt.title("Test")
plt.plot(y,x)
plt.grid()
plt.show()
```



```
x = np.array([0,1,2,3])
y = np.array([3,8,10,12])
plt.subplot(1,2,1)
plt.plot(x,y)
x = np.array([0,1,2,3])
y = np.array([13,28,32,42])
plt.subplot(1,2,2)
plt.plot(x,y)
plt.show()
```

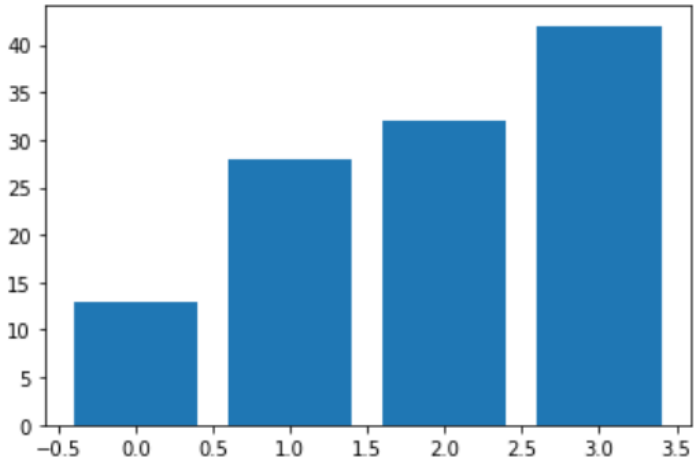



```
x1 = np.array([0, 1, 2, 3])
y1 = np.array([3, 8, 1, 10])
plt.subplot(1, 2, 1)
plt.plot(x1, y1)
plt.title("Current")
x2 = np.array([0, 1, 2, 3])
y2 = np.array([10, 20, 30, 40])
plt.subplot(1, 2, 2)
plt.plot(x2, y2)
plt.title("Voltage")
plt.suptitle("Power Consumed")
plt.show()
```



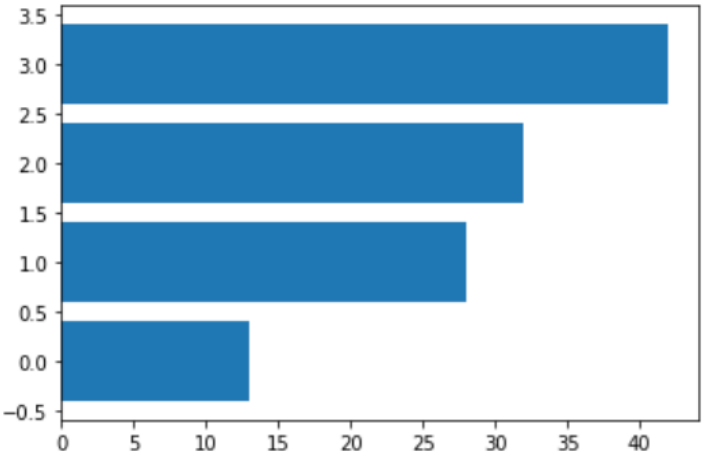
```
plt.bar(x,y)
```

```
<BarContainer object of 4 artists>
```



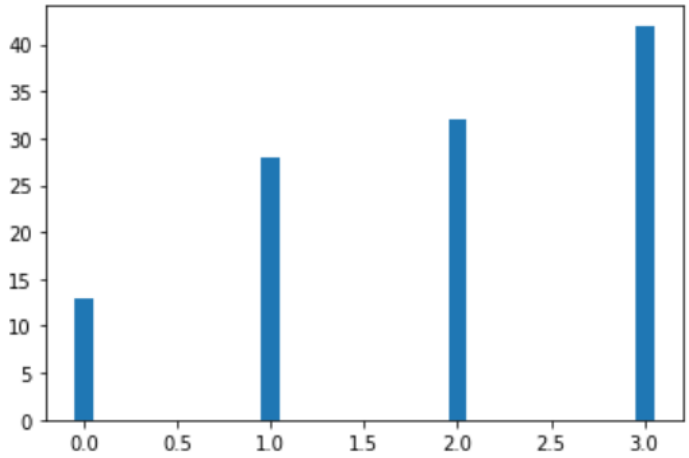

```
plt.barh(x,y)
```

```
<BarContainer object of 4 artists>
```



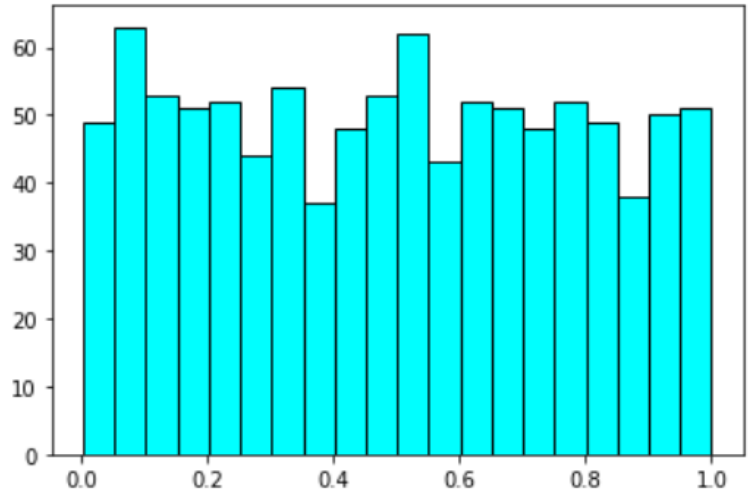

```
plt.bar(x,y,width = 0.1)
```

```
<BarContainer object of 4 artists>
```



```
data = np.random.random(1000)
plt.hist(data,bins = 20,color = "cyan", edgecolor = "black")
```

```
(array([49., 63., 53., 51., 52., 44., 54., 37., 48., 53., 62., 43., 52.,
        51., 48., 52., 49., 38., 50., 51.]),
 array([0.00218999, 0.05207762, 0.10196525, 0.15185287, 0.2017405 ,
        0.25162813, 0.30151576, 0.35140339, 0.40129102, 0.45117865,
        0.50106628, 0.55095391, 0.60084154, 0.65072917, 0.7006168 ,
        0.75050443, 0.80039206, 0.85027969, 0.90016732, 0.95005495,
        0.99994257]),
 <a list of 20 Patch objects>)
```




```
plt.pie(y)
```

```
([<matplotlib.patches.Wedge at 0x1d92eb78248>,
  <matplotlib.patches.Wedge at 0x1d92eb78b88>,
  <matplotlib.patches.Wedge at 0x1d92eb7c048>,
  <matplotlib.patches.Wedge at 0x1d92eb7cc88>],
 [Text(1.0313589124935578, 0.382490252973989, ''),
  Text(0.10501489634580775, 1.094975740162986, ''),
  Text(-1.0995895670519726, 0.030046364679530268, ''),
  Text(0.451980867449008, -1.0028525791261855, '')])
```

23261A66660 21/08/2025

EXPERIMENT – 4

AIM:

Implementation of Simple Linear Regression

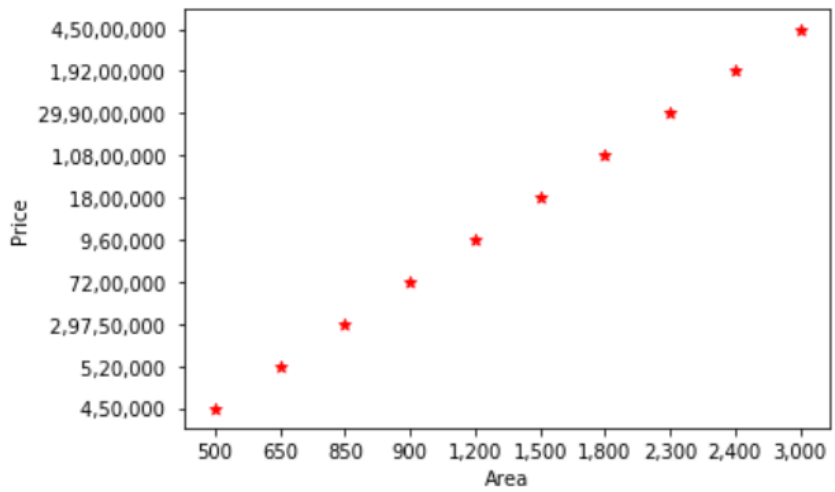
```
import pandas as pd
df = pd.read_csv('areaprice.csv')
```

```
df
```

	Area	Price
0	500	450000
1	650	520000
2	850	29750000
3	900	7200000
4	1200	9600000
5	1500	18000000
6	1800	10800000
7	2300	299000000
8	2400	19200000
9	3000	45000000

```
plt.xlabel('Area')
plt.ylabel('Price')
plt.scatter(df['Area'],df['Price'],color = 'r',marker="*")
```

```
<matplotlib.collections.PathCollection at 0x1d932447cc8>
```



```
from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit(df[['Area']], df['Price'])
```

```
LinearRegression()
```

```
print(reg.coef_)
```

```
[46716.11842105]
```

```
print(reg.predict(np.array([[10000]])))
```

```
[4.38087845e+08]
```

```
reg.intercept_
```

```
-29073338.81578946
```

```
d = pd.read_csv('House Price India.csv')
```

```
d.head(10)
```

```
d.head(10)
```

	grade of the house	Area of the house(excluding basement)	Area of the basement	Price
0	10	3370	280	2380000
1	8	1910	1010	1400000
2	8	2910	0	1200000
3	9	3310	0	838000
4	8	1880	830	805000
5	9	1700	900	790000
6	10	3660	0	785000
7	8	1550	690	750000
8	8	1440	950	750000
9	7	1300	900	698000

```
dataset = pd.read_csv('House Price India.csv')
X = dataset.iloc[:,:-1]
y = dataset.iloc[:,1].values
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=1/3,random_state = 0)
```

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
y_pred = regressor.predict(X_test)
```

```
y_pred
```

```
array([1440., 4270., 1010., ..., 1380., 1380., 2240.])
```