Dr. D. Y. Patil Science and Computer Science College,

INDEX

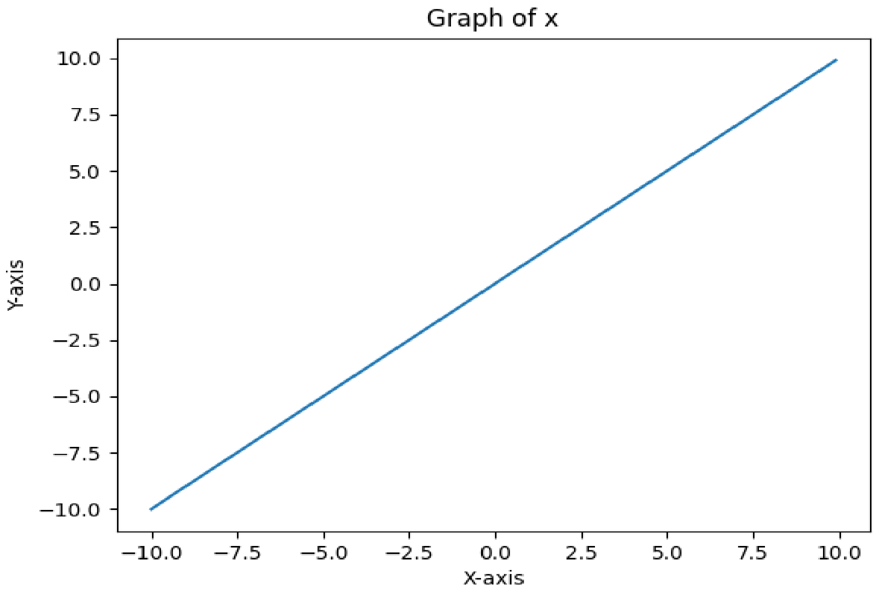
Practical No. – 1 2-D Graph Plotting-I

Q.1 Plot the graph of y=x in [-10,10] with 0.1 equal intervals.

Program :

```
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(-10,10,0.1)
y=x
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph of x')
plt.plot(x,y)
plt.show()
```

OUTPUT:

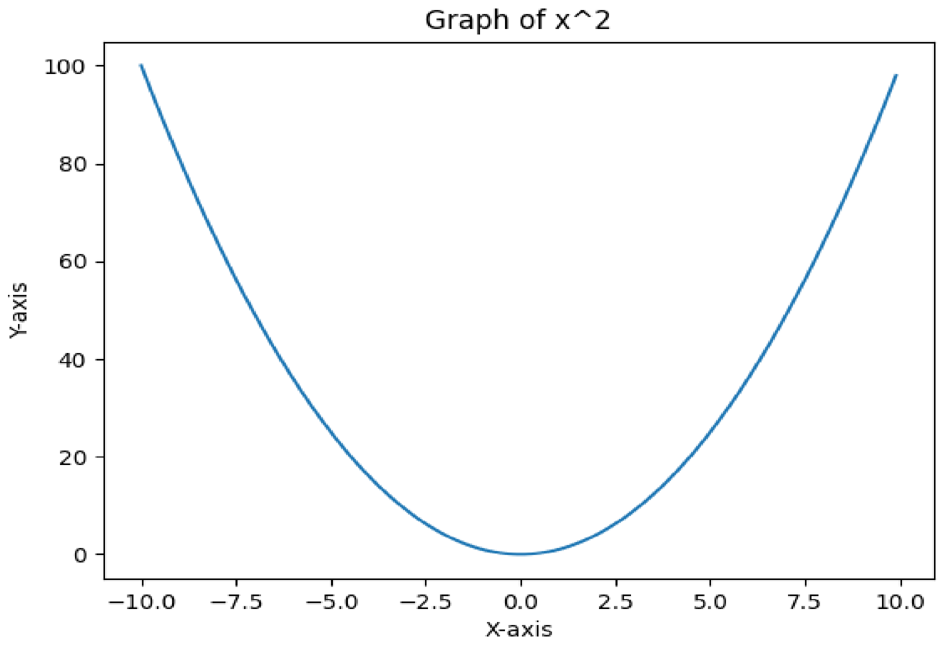


Q.2 Plot the graph of y=x^2 in [-10, 10] into 0.1 equal subintervals.

Program :

```
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(-10,10,0.1)
y=x**2
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph of x^2')
plt.plot(x,y)
plt.show()
```

OUTPUT:

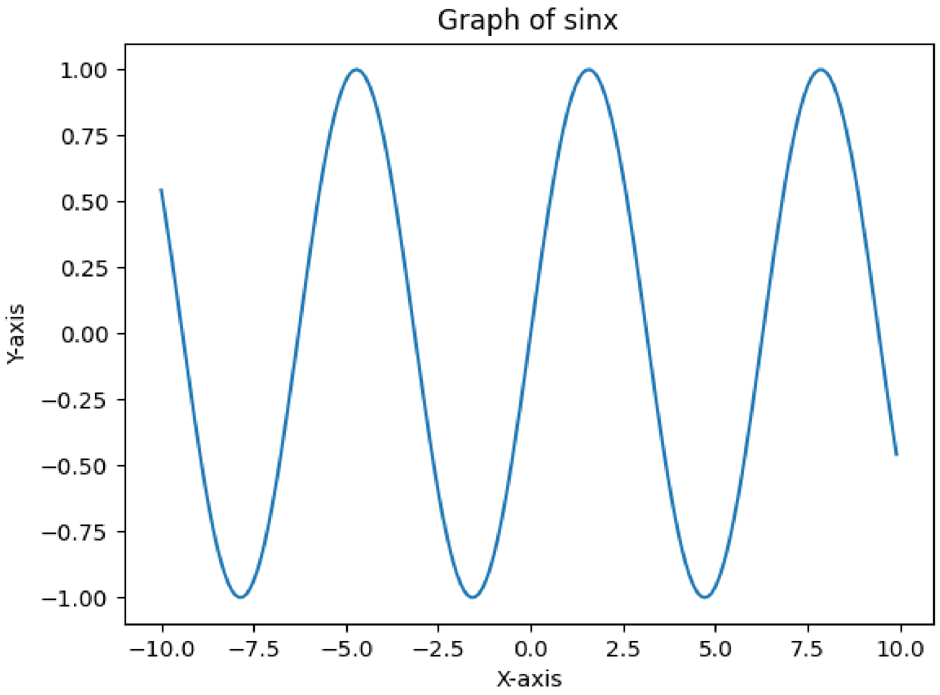


Q.3 Plot the graph of y= sin(x) in [-10, 10] into 0.1 equal subintervals.

Program :

```
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(-10, 10,0.1)
y=np.sin(x)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph of sinx')
plt.plot(x,y)
plt.show()
```

OUTPUT:

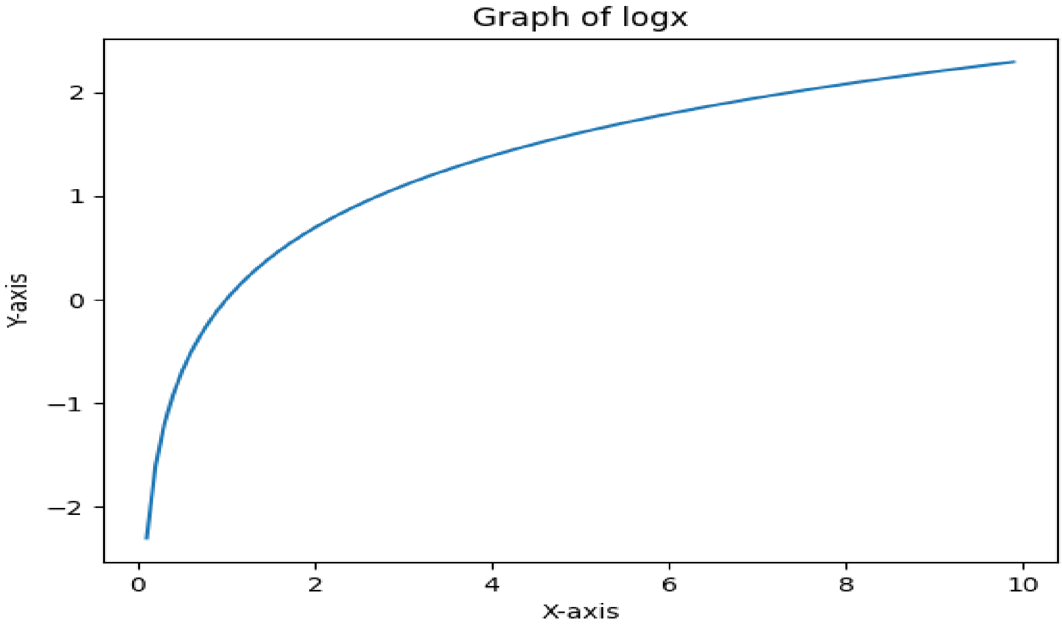


Q.4 Plot the graph of y= log(x) in [-10, 10] into 0.1 equal subintervals.

Program :

```
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(-10, 10,0.1)
y=np.log(x)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph of logxx')
plt.plot(x,y)
plt.show()
```

OUTPUT:

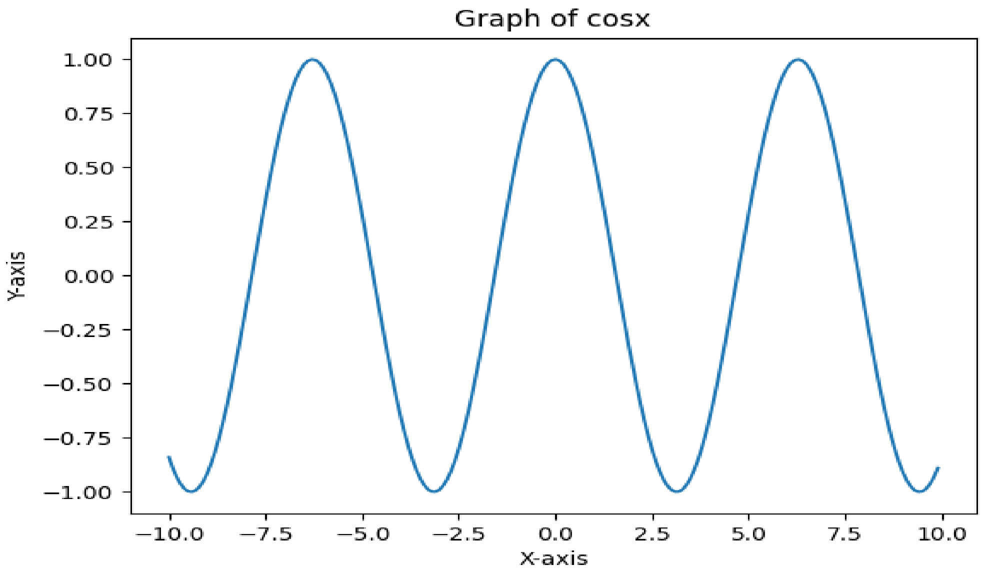


Q.5 Plot the graph of y= cos(x) in [-10, 10] into 0.1 equal subintervals.

Program :

```
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(-10, 10,0.1)
y=np.cos(x)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph of cosx')
plt.plot(x,y)
plt.show()
```

OUTPUT:

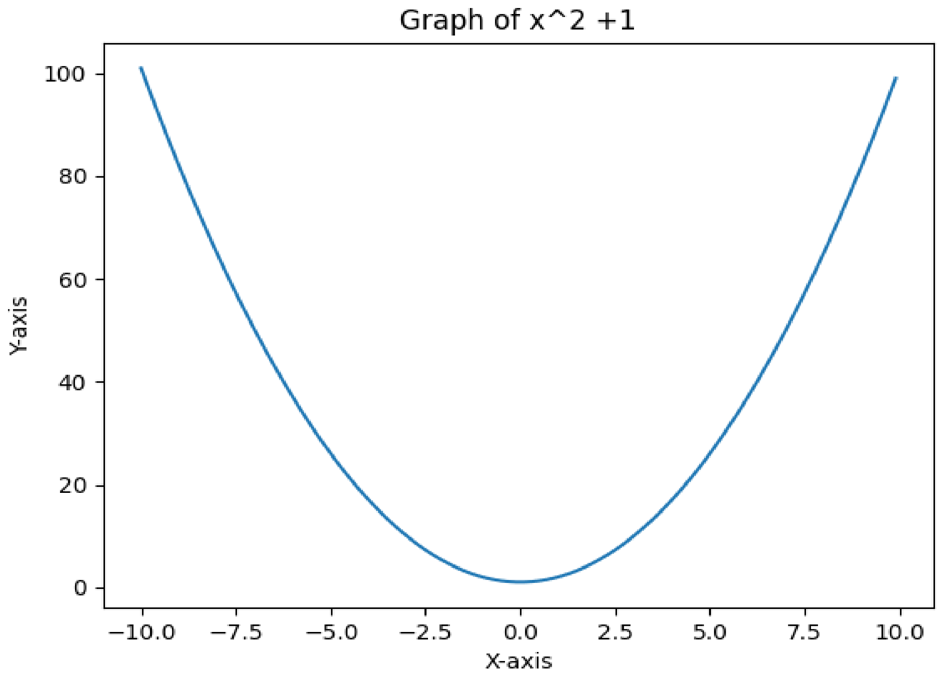


Q.6 Plot the graph of y= x^2 + 1 in [-10, 10] into 0.1 equal subintervals.

Program :

```
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(-10, 10,0.1)
y=x**2+1
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph of x^2 +1')
plt.plot(x,y)
plt.show()
```

OUTPUT:

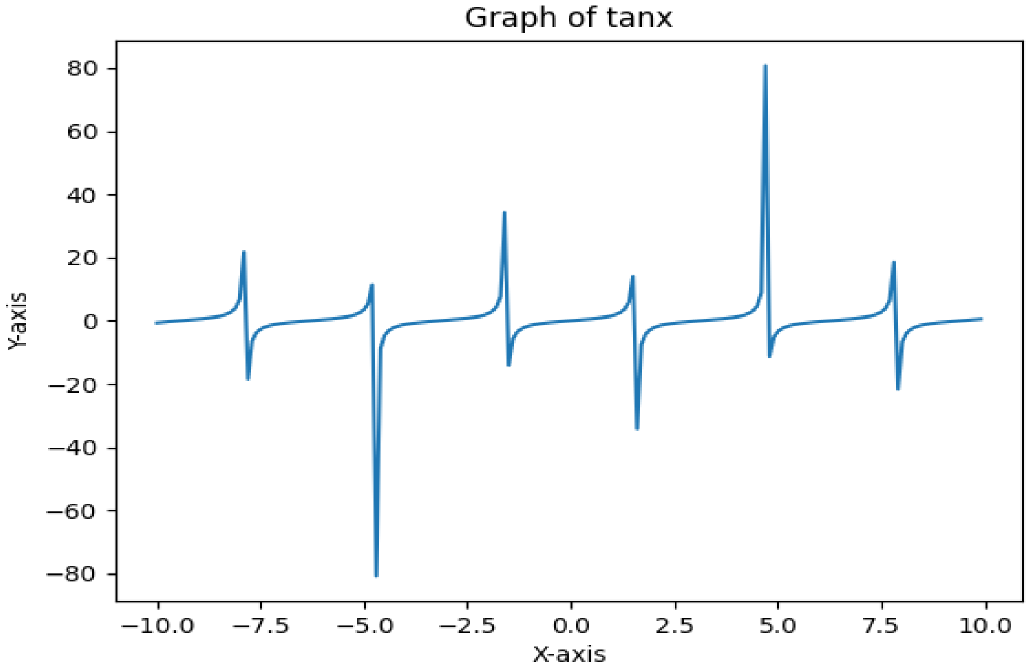


Q.7 Plot the graph of y= tan(x) in [-10, 10] into 0.1 equal subintervals.

Program :

```
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(-10, 10,0.1)
y=np.tan(x)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph of tanx')
plt.plot(x,y)
plt.show()
```

OUTPUT:

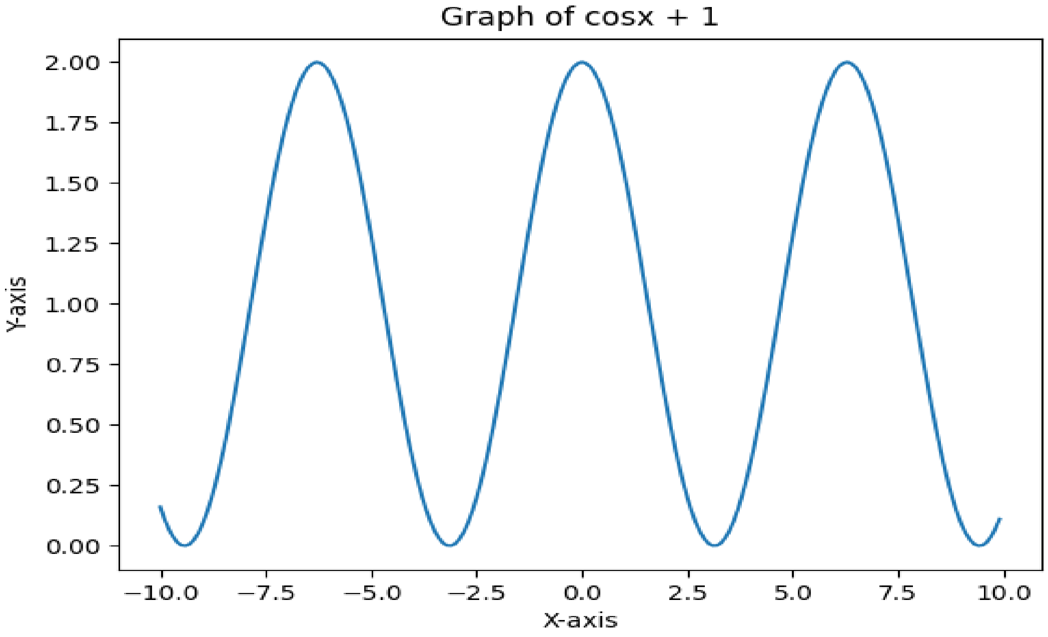


Q.8 Plot the graph of y= cos(x)+1 in [-10, 10] into 0.1 equal subintervals.

Program :

```
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(-10, 10,0.1)
y=np.cos(x)+1
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph of cosx + 1')
plt.plot(x,y)
plt.show()
```

OUTPUT:

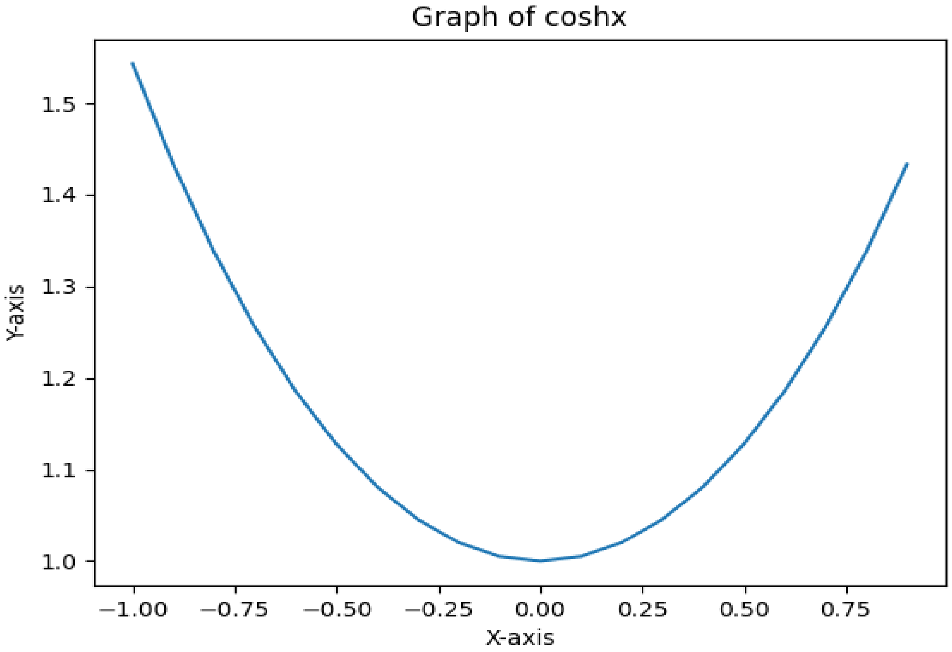


Q.9 Plot the graph of y= cosh(x) in [-1, 1] into 0.1 equal subintervals.

Program :

```
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(-1, 1,0.1)
y=np.cosh(x)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph of coshx')
plt.plot(x,y)
plt.show()
```

OUTPUT:

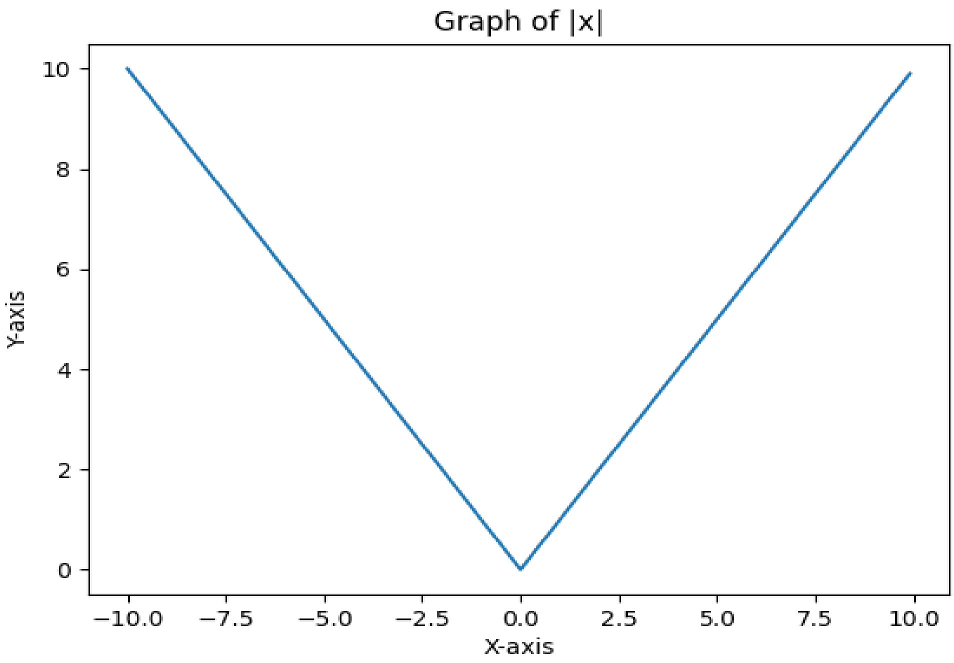


Q10 Plot the graph of y= |x| in [-10, 10] into 0.1 equal subintervals.

Program :

```
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(-10, 10,0.1)
y=np.absolute(x)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Graph of |x|')
plt.plot(x,y)
plt.show()
```

OUTPUT:


Graph of |x|
10
8
6
4
2
0
Y-axis
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
X-axis

Practical No. – 2 2-D Graph Plotting-II

Q.1 Draw line graph of the following data:

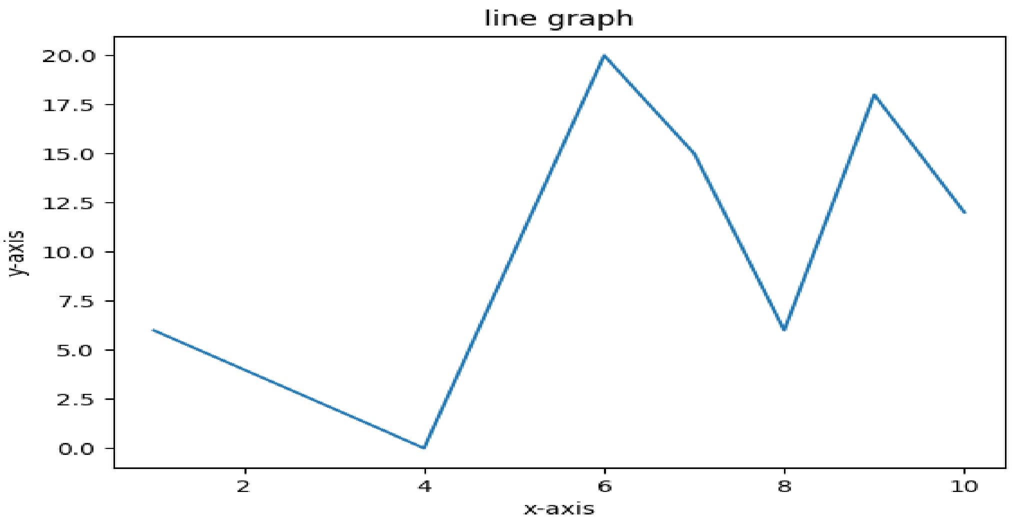
Program :

Color(X)	1	2	3	4	5
Number(Y)	16	20	30	26	34

Match(X)	1	2	3	4	5	6	7	8	9	10
Run(Y)	6	4	2	0	10	20	15	6	18	12

```
import matplotlib.pyplot as plt
import numpy as np
Match= [1,2,3,4,5,6,7,8,9,10]
run=[6,4,2,0,10,20,15,6,18,12]
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('line graph')
plt.plot(Match, run)
plt.show()
```

OUTPUT:

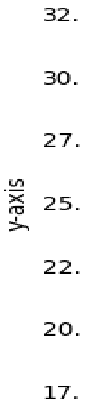


Q.2 Draw line graph of the following data:

Program :

```
import matplotlib.pyplot as plt
import numpy as np
color= [1,2,3,4,5]
number=[16, 20,30,26,34]
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('line graph')
plt.plot(color, number)
plt.show()
```

OUTPUT:



X1	1	2	3	4	5	6	7	8	9	10
Y1	6	4	2	0	10	20	15	6	18	12

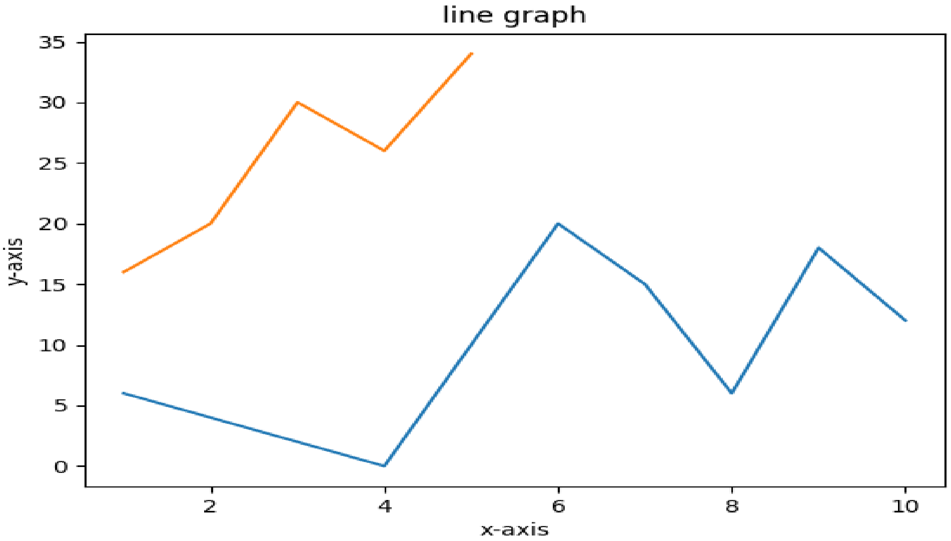
X2	1	2	3	4	5
Y2	16	20	30	26	34

Q.3 Draw the combination of line graph with the following data:

Program :

```
import matplotlib.pyplot as plt
import numpy as np
X1= [1,2,3,4,5,6,7,8,9,10]
Y1=[6,4,2,0,10,20,15,6,18,12]
plt.plot(X1,Y1,label='line 1')
X2= [1,2,3,4,5]
Y2=[16, 20,30,26,34]
plt.plot(X2,Y2,label='line 2')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('line graph')
plt.show()
```

OUTPUT:



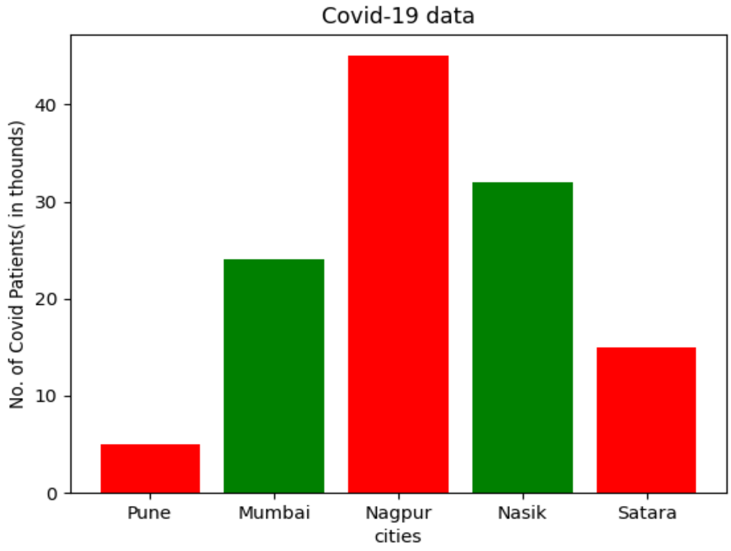
Q.4 Draw a bar graph using following data:

Cities	Pune	Mumbai	Nagpur	Nasik	Satara
No.of Covid Patients	5	24	45	32	15

Program :

```
import matplotlib.pyplot as plt
left= [1,2,3,4,5]
height=[5,24,45,32,15]
tick_label=[5,24,45,32,15]
tick_label=['Pune', 'Mumbai', 'Nagpur', 'Nasik', 'Satara']
plt.bar(left, height, tick_label=tick_label, width=0.8, color=['red','green'])
plt.xlabel('cities')
plt.ylabel('No. of Covid Patients( in thounds)')
plt.title('Covid-19 data')
plt.show()
```

OUTPUT:



Q.5 Draw a bar graph using following data:

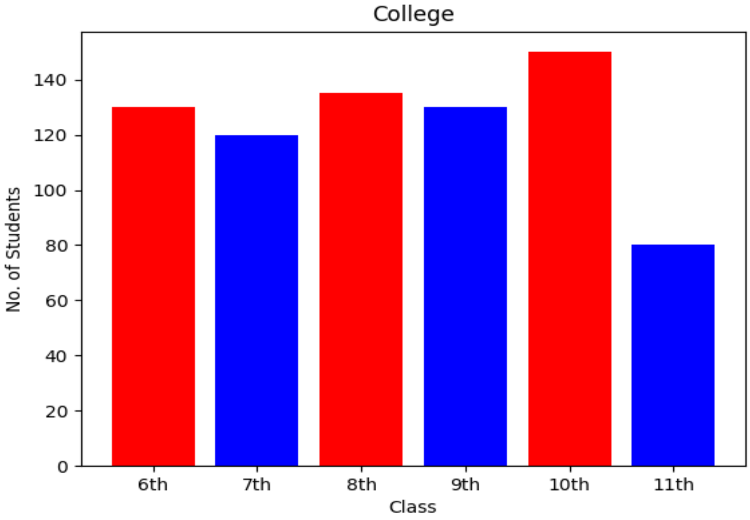
Program :

```
import matplotlib.pyplot as plt
x= [1,2,3,4,5,6]
y=[130,120,135,130,150,80]
tick_label=['mon','tues', 'wed','thurs','fri','sat']
tick_label=['6th','7th','8th','9th','10th','11th']
plt.bar(x, y, tick_label=tick_label, width=0.8, color=['red','blue'])
```

Class	6th	7th	8th	9th	10th	11th
No. of Students	130	120	135	130	150	80

```
plt.xlabel('Class')
plt.ylabel('No. of Students')
plt.title('College')
plt.show()
```

OUTPUT:



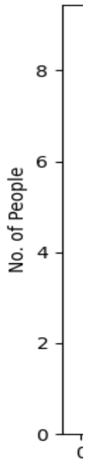
Q.6 Draw the Histogram for the following data:

Ages=2,5,70,40,30,45,50,45,432,40,44,60,713,57,18,90,77,32,21,20,40,45,32,38

Program :

```
import matplotlib.pyplot as plt
Ages=[2,5,70,40,30,45,50,45,432,40,44,60,713,57,18,90,77,32,21,20,40,45,32,38]
range=(0,100)
bins=5
plt.hist(Ages,bins,range,color='blue',histtype='bar',rwidth=0.8)
plt.xlabel('Ages')
plt.ylabel('No. of People')
plt.title('histogram plot')
plt.show()
```

OUTPUT:



Cities	Pune	Mumbai	Nagpur	Nasik	Satara
No.of Covid	5	24	45	32	15

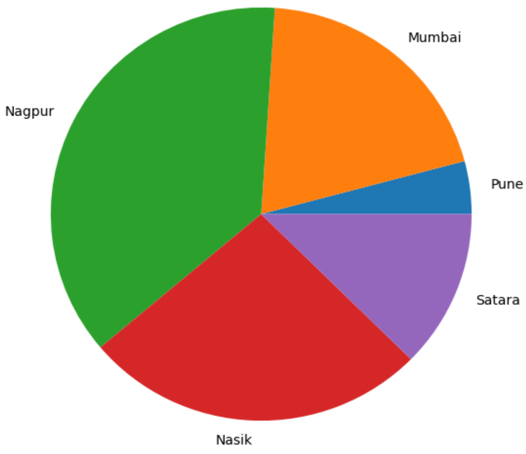
Q.7Draw Pie Chart using following Data:

Patients					

Program:

```
import matplotlib.pyplot as plt
left=[1,2,3,4,5]
height=[5,24,45,32,15]
tick_label=['Pune', 'Mumbai', 'Nagpur', 'Nasik', 'Satara']
fig=plt.figure(figsize=(10,7))
plt.pie(height, labels=tick_label)
plt.show()
```

OUTPUT:

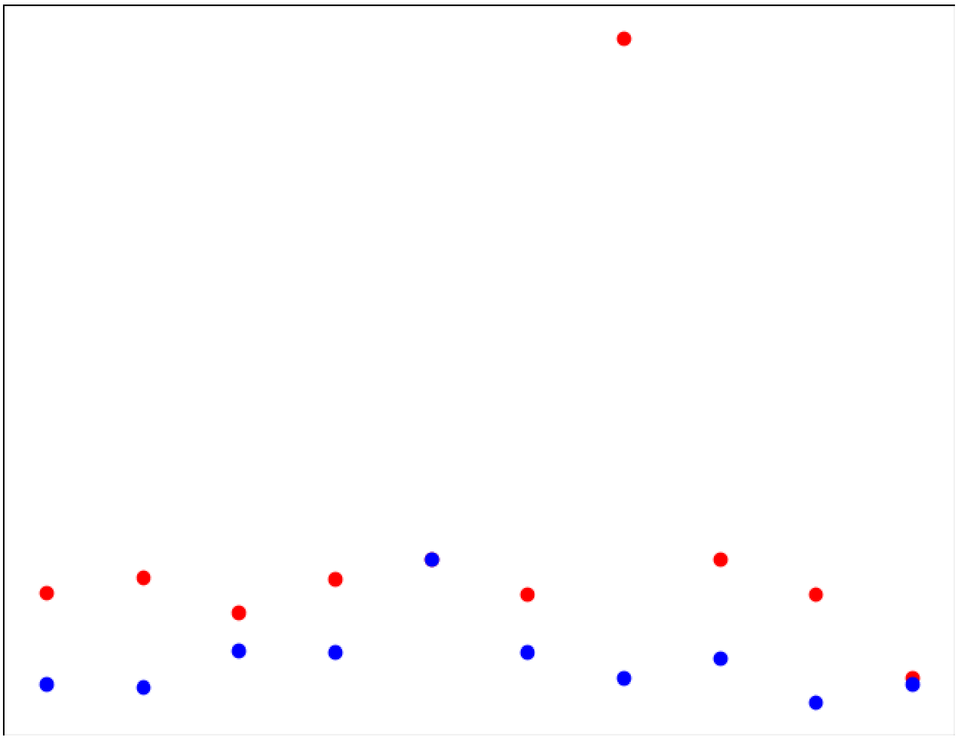


Q.8 Draw Scatter Plots using following Data:
Girls Score= 81, 90, 70, 89, 100, 80, 90, 100, 80, 34
Boys Score= 30, 29, 49, 48,100, 48, 34, 45, 20, 30

Program:

```
import matplotlib.pyplot as plt
girls_scores=[81,90,70,89,100,80,390,100,80,34]
boys_scores=[30,29,49,48,100,48,34,45,20,30]
grades_range=[10,20,30,40,50,60,70,80,90,100]
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
ax.scatter(grades_range,girls_scores,color='r')
ax.scatter(grades_range, boys_scores, color='b')
ax.set_xlabel('Grades Range')
ax.set_ylabel('Grades Scored')
ax.set_title('scatter plot')
plt.show()
```

OUTPUT:



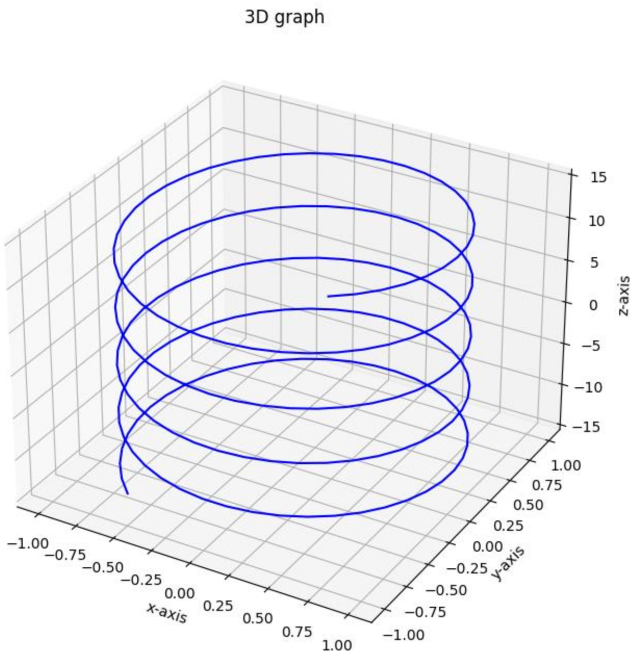
Practical No. 3 3D Graph Plotting

Q.1 Draw Helix x=sin(z), y=cos(z), $-15 \leq z \leq 15$

Program:

```
from numpy import*
from matplotlib.pyplot import*
ax=axes(projection='3d')
z=linspace(-15,15,200)
x=sin(z)
y=cos(z)
ax.plot3D(x,y,z,'blue')
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_title('3D graph')
show()
```

OUTPUT:

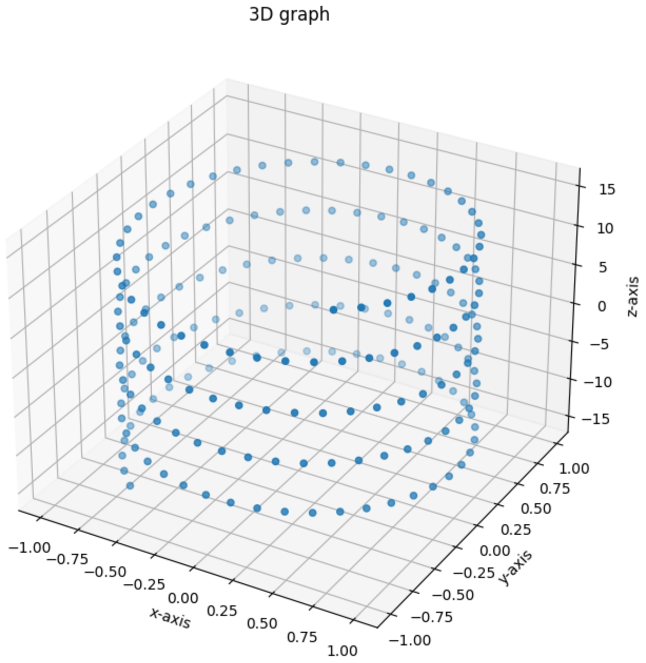


Que.2 Draw the Helix Scatter x=sin(z), y=cos(z), $-15 \leq z \leq 15$

Program:

```
from numpy import*
from matplotlib.pyplot import*
ax=axes(projection='3d')
z=linspace(-15,15,200)
x=sin(z)
y=cos(z)
ax.scatter(x,y,z,cmap='Reds')
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_title('3D graph')
show()
```

OUTPUT:



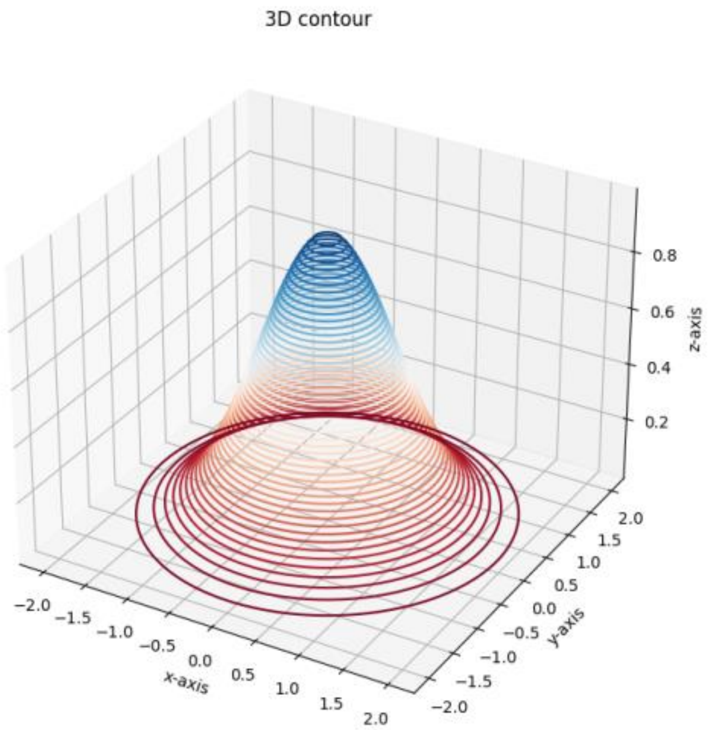
Q.3 Draw the contour $z = -x^2 - y^2$

Program:

```
from numpy import*
from matplotlib.pyplot import*
ax=axes(projection='3d')
def f(x,y):
    return exp(-x**2-y**2)
x=linspace(-2,2,100)
y=linspace(-2,2,100)
X,Y=meshgrid(x,y)
Z=f(X,Y)
ax.contour3D(X,Y,Z,50,cmap='RdBu')

ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_title('3D contour')
show()
```

OUTPUT:



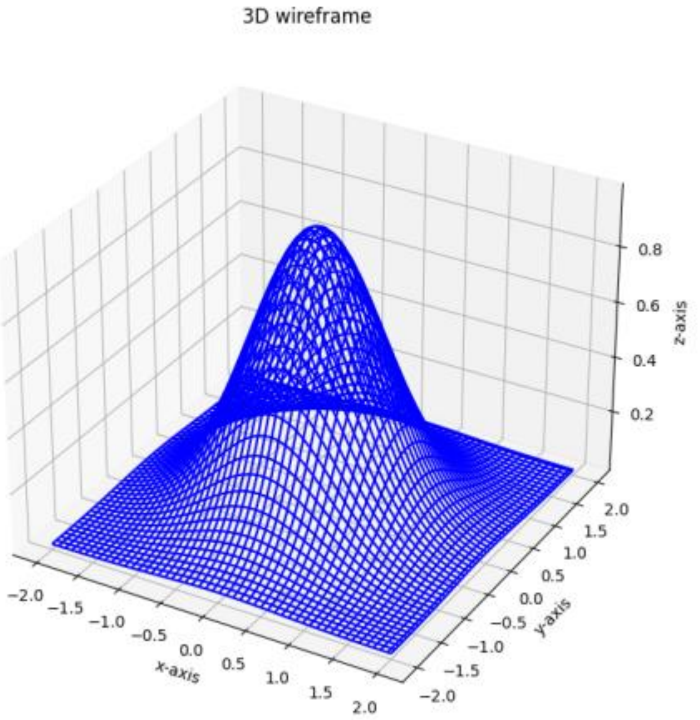
Q.4 Draw the wireframe $z = -x^2 - y^2$

Program:

```
from numpy import*
from matplotlib.pyplot import*
ax=axes(projection='3d')

def f(x,y):
    return exp(-x**2-y**2)
x=linspace(-2,2,100)
y=linspace(-2,2,100)
X,Y=meshgrid(x,y)
Z=f(X,Y)
ax.plot_wireframe(X,Y,Z,color='blue')
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_title('3D wireframe')
show()
```

OUTPUT:



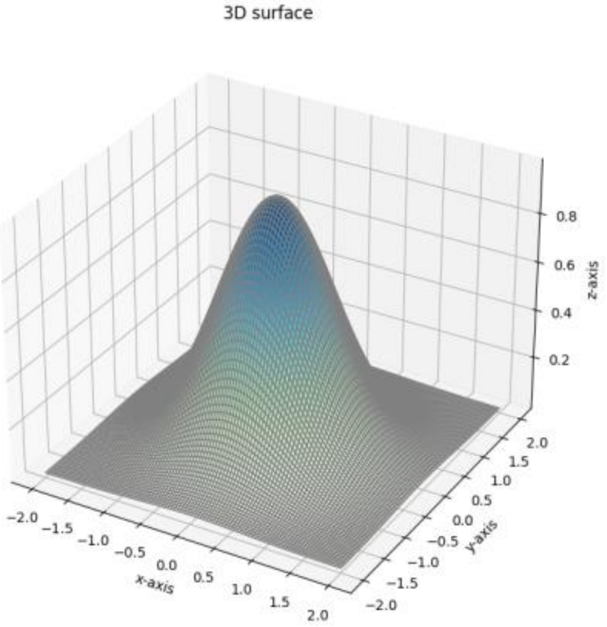
Q.5 Draw the surface $z = -x^2 - y^2$

Program:

```
from numpy import*
from matplotlib.pyplot import*
ax=axes(projection='3d')

def f(x,y):
   return exp(-x**2-y**2)
x=linspace(-2,2,100)
y=linspace(-2,2,100)
X,Y=meshgrid(x,y)
Z=f(X,Y)
ax.plot_surface(X,Y,Z,rstride=1,cstride=1,cmap='GnBu',edgecolor='grey')
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_title('3D surface')
show()
```

OUTPUT:



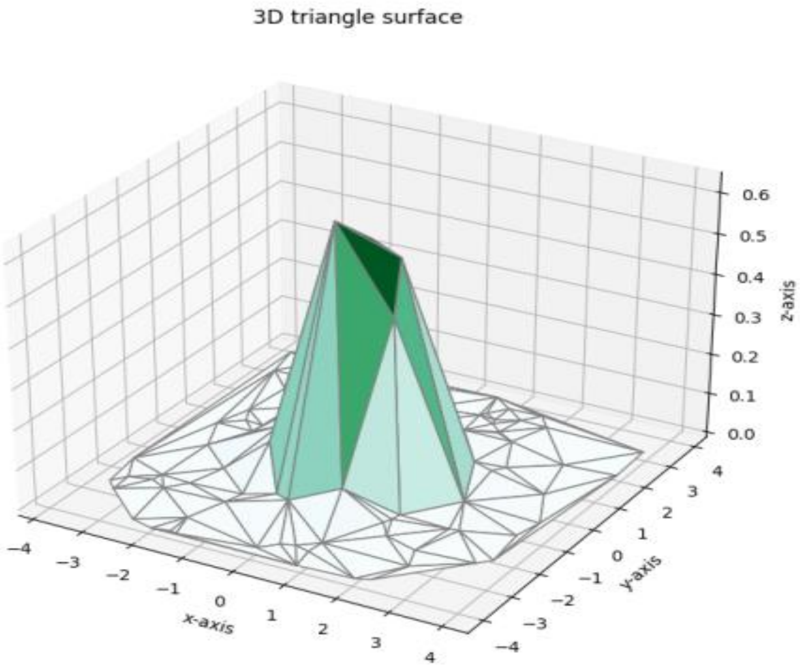
Q.6. Draw the triangle surface $z = -x^2 - y^2$

Program:

```
from numpy import*
from matplotlib.pyplot import*
ax=axes(projection='3d')

def f(x,y):
   return exp(-x**2-y**2)
x=random.uniform(low=-4,high=4,size=100)
y=random.uniform(low=-4,high=4,size=100)
z=f(x,y)
ax.plot_trisurf(x,y,z,cmap='BuGn',edgecolor='grey')
ax.set_xlabel('x-axis')
ax.set_ylabel('y-axis')
ax.set_zlabel('z-axis')
ax.set_title('3D triangle surface')
show()
```

OUTPUT:


3D triangle surface
x-axis
−4
−3
−2
−1
0
1
2
3
4
y-axis
−4
−3
−2
−1
0
1
2
3
4
z-axis
0.0
0.1
0.2
0.3
0.4
0.5
0.6

Practical No. 4 2D- Transformation

Q1.Apply transformation on a point B(5,0)

1)scaling in x co-ordinate by factor 2.

Program:

```
from sympy import*
p=Point(5,0)
 m=Matrix([[2,0,0],[0,1,0], [0,0,1]])
 P1=p.transform(m)
print(P1)
```

OUTPUT

```
Point2D(10, 0)
```

2)scaling in y co-ordinate by factor 3.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix([[1,0,0], [0,3,0], [0,0,1]])
 P1=p.transform(m)
 Print(P1)
```

OUTPUT:

```
Point2D(5, 0)
```

3)scaling in x and y co-ordinate by factor 2 and 3.

Program:

```
from sympy import*
p=Point(5,0)

m=Matrix([[2,0,0], [0,3,0], [0,0,1]])
 P1=p.transform(m)
print(P1)
```

OUTPUT:

```
OUTPUT Point2D(10, 0)
```

4)uniform sacling by factor 3.

```
from sympy import*
p=Point(5,0)

m=Matrix([[3,0,0],[0,3,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(15,0)
```

5)reflection through the the line y=0.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix[[[1,0,0),[0,-1,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(5,0)
```

6)reflection through the line x=0.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix([[-1,0,0],[0,1,0], [0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(-5, 0)
```

7)reflection through the line x-y=0.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix[[[0,1,0],[1,0,0],[0,0,1]])
 P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(0,5)
```

8)reflection through the line x+y=0

Program:

```
from sympy import*
 p=Point(5,0)
m=Matrix([[0,-1,0],[-1,0,0],[0,0,1]]]
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(0, -5)
```

9)reflection through the line origin.

```
from sympy import*
p=Point(5,0)
m=Matrix(-1,0,0],[0,-1,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(-5,0)
```

10)shearing in x co-ordinate by factor 3.

Program:

```
from sympy import*
 p=Point(5,0)
m=Matrix([[1,0,0],[3,1,0],[0,0,1]])
 P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(5, 0)
```

11)shearing in y co-ordinate by factor 2.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix([[1,2,0],[0,1,0], [0,0,1]])
 P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(5, 10)
```

12)shearing in x and y co-ordinate by factor 3 and 2.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix([[1,2,0],[3,1,0],[0,0,1]])
 P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(5, 10)
```

13)rotation about origin by angle 90

Program:

```
from sympy import*
 p=Point(5,0)
m=Matrix([[0,1,0),(-1,0,0],[0,0,1])
```

```
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
 Point2D(0,5)
```

14)rotation about origin by angle -90.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix[[[0,1,0],[-1,0,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(0,5)
```

15)translation in x direction by factor 2.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix([[1,0,0],[0,1,0],[2,0,1]])
 P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(7,0)
```

16)translation in y direction by factor 3.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix([[1,0,0],[0,1,0],[0,3,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(5, 3)
```

17)translation in x and y direction by factor 2 and 3.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix([[1,0,0], [0,1,0],[2,3,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
 Point2D(7,3)
```

Q2.Apply each of the transformation on a line p(4,5)

1)reflection through y-axis.

Program:

```
from sympy import*
p=Point(4,5)
m=Matrix[[-1,0,0],[0,1,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(-4,5)
```

2)scaling in x co-ordinate by factor 3.

Program:

```
from sympy import*
p=Point(4,5)
m=Matrix([[3,0,0],[0,1,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(12,5)
```

3)scaling in y co-ordinate by factor 2

Program:

```
 from sympy import*
p=Point(4,5)
m=Matrix[[[1,0,0],[0,2,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(4,10)
```

4)shearing in y direction by 3 unit

Program:

```
from sympy import*
p=Point(4,5)
m=Matrix([[1,3,0], [0,1,0], [0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(4, 17)
```

5)scaling in both x and y direction by 5/3 and 2 unit resp.

Program:

```
from sympy import*
p=Point(4,5)
m=Matrix([[1,2,0],[5/3,1,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(37/3, 13)
```

6)shearing in x and y direction by -3 and 1 resp.

Program:

```
from sympy  import*
p=Point(4,5)
m=Matrix([[1,1,0],[-3,1,0], [0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(-11, 9)
```

7)rotation about origin by angle 45.

Program:

```
from sympy import*
p=Point(4,5)
m=Matrix[[[0.7071,0.7071,0],[-0.7071,0.7071,0),(0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(-7071/10000, 63639/10000)
```

Practical No. 5 3D Transformation

Q1.Apply transformation on a point B(5,0)

1)scaling in x co-ordinate by factor 2.

Program:

```
from sympy import*
p=Point(5,0)
 m=Matrix([[2,0,0],[0,1,0], [0,0,1]])
 P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(10, 0)
```

2)scaling in y co-ordinate by factor 3.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix([[1,0,0], [0,3,0], [0,0,1]])
 P1=p.transform(m)
 Print(P1)
```

OUTPUT:

```
Point2D(5, 0)
```

3)scaling in x and y co-ordinate by factor 2 and 3.

Program:

```
from sympy import*
p=Point(5,0)

m=Matrix([[2,0,0], [0,3,0], [0,0,1]])
 P1=p.transform(m)
print(P1)
```

OUTPUT

```
OUTPUT Point2D(10, 0)
```

4)uniform sacling by factor 3.

```
from sympy import*
p=Point(5,0)

m=Matrix([[3,0,0],[0,3,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(15,0)
```

5)reflection through the the line y=0.

```
from sympy import*
p=Point(5,0)
m=Matrix[[[1,0,0),[0,-1,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(5,0)
```

6)reflection through the line x=0.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix([[-1,0,0],[0,1,0], [0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(-5, 0)
```

7)reflection through the line x-y=0.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix[[[0,1,0],[1,0,0],[0,0,1]])
 P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(0,5)
```

8)reflection through the line x+y=0

Program:

```
from sympy import*
 p=Point(5,0)
m=Matrix([[0,-1,0],[-1,0,0],[0,0,1]]]
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(0, -5)
```

9)reflection through the line origin.

Program:

```
from sympy import*
```

```
p=Point(5,0)
m=Matrix(-1,0,0],[0,-1,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(-5,0)
```

10)shearing in x co-ordinate by factor 3.

Program:

```
from sympy import*
 p=Point(5,0)
m=Matrix([[1,0,0],[3,1,0],[0,0,1]])
 P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(5, 0)
```

11)shearing in y co-ordinate by factor 2.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix([[1,2,0],[0,1,0], [0,0,1]])
 P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(5, 10)
```

12)shearing in x and y co-ordinate by factor 3 and 2.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix([[1,2,0],[3,1,0],[0,0,1]])
 P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(5, 10)
```

13)rotation about origin by angle 90

Program:

```
from sympy import*
 p=Point(5,0)
m=Matrix([[0,1,0),(-1,0,0],[0,0,1])
P1=p.transform(m)
```

```
print(P1)
```

OUTPUT:

```
 Point2D(0,5)
```

14)rotation about origin by angle -90.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix[[[0,1,0],[-1,0,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(0,5)
```

15)translation in x direction by factor 2.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix([[1,0,0],[0,1,0],[2,0,1]])
 P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(7,0)
```

16)translation in y direction by factor 3.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix([[1,0,0],[0,1,0],[0,3,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(5, 3)
```

17)translation in x and y direction by factor 2 and 3.

Program:

```
from sympy import*
p=Point(5,0)
m=Matrix([[1,0,0], [0,1,0],[2,3,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
 Point2D(7,3)
```

Q2.Apply each of the transformation on a line p(4,5)

1)reflection through y-axis.

Program:

```
from sympy import*
p=Point(4,5)
m=Matrix[[-1,0,0],[0,1,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(-4,5)
```

2)scaling in x co-ordinate by factor 3.

Program:

```
from sympy import*
p=Point(4,5)
m=Matrix([[3,0,0],[0,1,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(12,5)
```

3)scaling in y co-ordinate by factor 2

Program:

```
 from sympy import*
p=Point(4,5)
m=Matrix[[[1,0,0],[0,2,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(4,10)
```

4)shearing in y direction by 3 unit

Program:

```
from sympy import*
p=Point(4,5)
m=Matrix([[1,3,0], [0,1,0], [0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(4, 17)
```

5)scaling in both x and y direction by 5/3 and 2 unit resp.

Program:

```
from sympy import*
```

```
p=Point(4,5)
m=Matrix([[1,2,0],[5/3,1,0],[0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(37/3, 13)
```

6)shearing in x and y direction by -3 and 1 resp.

Program:

```
from sympy  import*
p=Point(4,5)
m=Matrix([[1,1,0],[-3,1,0], [0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(-11, 9)
```

7)rotation about origin by angle 45.

Program:

```
from sympy import*
p=Point(4,5)
m=Matrix[[[0.7071,0.7071,0],[-0.7071,0.7071,0),(0,0,1]])
P1=p.transform(m)
print(P1)
```

OUTPUT:

```
Point2D(-7071/10000, 63639/10000)
```

Practical No. 6 LPP

Q.1. Solve the LPP:

Minimize Z = x_1-3x_2+2x_3

Subject to, 3x_1-x_2+2x_3<=7
3x_1-1 x_2+2 x_3<=7
-2x_1+4x_2<=12
-4x_1+3x_2+ 8x_3<=10
x_1>=0, x_2>=0, x_3>=0

Program:

```
from pulp import*
lpp=LpProblem(‘Problem’,LpMinimize)
x1=LpVariable(‘x1’,lowBound=0)
x2=LpVariable(‘x2’,lowBound=0)
x3=LpVariable(‘x3’,lowBound=0)
lpp+=x1-3*x2+2*x3
lpp+=3*x1-x2+2*x3<=7
lpp+=-2*x1+4*x2<=12
lpp+=-4*x1+3*x2+8*x3<=10
print(lpp)
lpp.solve()
print(“\nx1=”,value(x1),“\nx2=”,value(x2),“\nx3=”,value(x3),”Zmin=”,value(lpp.obje
ctive))
```

OUTPUT:

```
Problem:
MINIMIZE
1*x1 + -3*x2 + 2*x3 + 0
SUBJECT TO
_C1: 3 x1 - x2 + 2 x3 <= 7
_C2: - 2 x1 + 4 x2 <= 12
_C3: - 4 x1 + 3 x2 + 8 x3 <= 10
VARIABLES
x1 Continuous
x2 Continuous
x3 Continuous
x1= 4.0
x2= 5.0
x3= 0.0
Zmin= -11.0
```

Q.2. Solve the LPP:

Minimize Z = 2x_1-3x_2

Subject to, -3x_1+4x_2<=12
x_2<=2
x_1>=0, x_2>=0

Program:

```
from pulp import*
lpp=LpProblem(‘Problem’,LpMinimize)
```

```
x1=LpVariable(‘x1’,lowBound=0)
x2=LpVariable(‘x2’,lowBound=0)
lpp+=2*x1-3*x2
lpp+=-3*x1+4*x2<=12
lpp+=x2<=2
print(lpp)
lpp.solve()
print(“\nx1=”,value(x1),“\nx2=”,value(x2), “Zmin=”,value(lpp.objective))
```

OUTPUT:

```
Problem:
MINIMIZE
2*x1 + -3*x2 + 0
SUBJECT TO
_C1: - 3 x1 + 4 x2 <= 12
_C2: x2 <= 2
VARIABLES
x1 Continuous
x2 Continuous
x1= 0.0
x2= 2.0
Zmin= -6.0
```

Q.3. Solve the LPP:
Minimize $Z = -2x_1-2x_2$
Subject to, $-1x_1+x_2<=3$
$-1x_1+3x_2<=12$
$1x_1-4x_2<=4$
$x_1>=0,\ x_2>=0,\ x_3>=0$

Program:

```
from pulp import*
lpp=LpProblem(‘Problem’,LpMinimize)
x1=LpVariable(‘x1’,lowBound=0)
x2=LpVariable(‘x2’,lowBound=0)
lpp+=-2*x1-2*x2
lpp+=-x1+x2<=3
lpp+=-x1+3*x2<=12
lpp+=x1-4*x2<=4
print(lpp)
lpp.solve()
print(“\nx1=”,value(x1),“\nx2=”,value(x2),“\nx3=”,value(x3),”Zmin=”,value(lpp.objective))
```

OUTPUT:

```
Problem:
MINIMIZE
-2*x1 + -2*x2 + 0
```

```
SUBJECT TO
_C1: - x1 + x2 <= 2
_C2: - x1 + 3 x2 <= 12
_C3: x1 - 4 x2 <= 4
VARIABLES
x1 Continuous
x2 Continuous
x1= 3.0
x2= 5.0
Zmin= -16.0
```

Q.4. Solve the LPP:

Minimize $Z = -3x_1-3x_2-2x_3$

Subject to, $x_1+x_2+2x_3<=20$

$2x_1+x_2+4\ x_3<=32$

$-4x_1+3x_2+\ 8x_3<=10$

$x_1>=0,\ x_2>=0,\ x_3>=0$

Program:

```
from pulp import*
lpp=LpProblem(‘Problem’,LpMinimize)
x1=LpVariable(‘x1’,lowBound=0)
x2=LpVariable(‘x2’,lowBound=0)
x3=LpVariable(‘x3’,lowBound=0)
lpp+=-3*x1-3*x2-2*x3
lpp+=x1+x2+2*x3<=20
lpp+=2*x1+x2+4*x3<=32
print(lpp)
lpp.solve()
print(“\nx1=”,value(x1),“\nx2=”,value(x2),“\nx3=”,value(x3),”Zmin=”,value(lpp.objective))
```

OUTPUT:

```
Problem:
MINIMIZE
-3*x1 + -3*x2 + -2*x3 + 0
SUBJECT TO
_C1: x1 + x2 + 2 x3 <= 20
_C2: 2 x1 + x2 + 4 x3 <= 32
VARIABLES
x1 Continuous
x2 Continuous
x3 Continuous
x1= 12.0
x2= 8.0
x3= 0.0
Zmin= -60.0
```

Q.5. Solve the LPP:

Minimize $Z = -20x_1-x_2-2x_3$

Subject to, $x_1+4x_2-x_3<=20$

$x_1+x_2<=10$

$3x_1+5x_2-3x_3<=50$

$x_1>=0, x_2>=0, x_3>=0$

Program:

```
from pulp import*
lpp=LpProblem('Problem',LpMinimize)
x1=LpVariable('x1',lowBound=0)
x2=LpVariable('x2',lowBound=0)
x3=LpVariable('x3',lowBound=0)
lpp+=-20*x1-x2-2*x3
lpp+=x1+4*x2-x3<=20
lpp+=x1+x2<=10
lpp+=3*x1+5*x2-3*x3<=50
print(lpp)
lpp.solve()
print("\nx1=",value(x1),"\nx2=",value(x2),"\nx3=",value(x3),"Zmin=",value(lpp.objective))
```

OUTPUT:

```
Problem:
MINIMIZE
-20*x1 + -1*x2 + -2*x3 + 0
SUBJECT TO
_C1: x1 + 4 x2 - x3 <= 20
_C2: x1 + x2 <= 10
_C3: 3 x1 + 5 x2 - 3 x3 <= 50
VARIABLES
x1 Continuous
x2 Continuous
x3 Continuous
x1= 10.0
x2= 0.0
x3= 0.0
 Zmin= -200.0
```

Q.6. Solve the LPP:

Maximize $Z = 2x_1+3x_2+4x_3$

Subject to, $3x_1-2x_2<=41$

$2x_1+ x_2+3 x_3<=35$

$2x_1+3x_2<=30$

$x_1>=0, x_2>=0, x_3>=0$

Program:

```
from pulp import*
lpp=LpProblem('Problem',LpMaximize)
```

```
x1=LpVariable('x1',lowBound=0)
x2=LpVariable('x2',lowBound=0)
x3=LpVariable('x3',lowBound=0)
lpp+=2*x1+3*x2+4*x3
lpp+=3*x1-2*x3<=41
lpp+=2*x1+x2+x3<=35
lpp+=2*x2+3*x3<=30
print(lpp)
lpp.solve()
print("\nx1=",value(x1),"\nx2=",value(x2),"\nx3=",value(x3),"Zmax=",value(lpp.objective))
```

OUTPUT:

```
Problem:
MAXIMIZE
50*x1 + 80*x2 + 0
SUBJECT TO
_C1: x1 + 2 x2 <= 32
_C2: 3 x1 + 4 x2 <= 84
VARIABLES
x1 Continuous
x2 Continuous
x1= 20.0
x2= 6.0 Zmax= 1480.0
```

Q.7. Solve the LPP:

Maximize Z = $50x_1 + 80x_2$

Subject to, $x_1+2x_2<=32$

$3x_1+4x_2<=84$

$x_1>=0$, $x_2>=0$

Program:

```
from pulp import*
lpp=LpProblem('Problem',LpMaximize)
x1=LpVariable('x1',lowBound=0)
x2=LpVariable('x2',lowBound=0)
lpp+=50*x1+80*x2
lpp+=x1+2*x2<=32
lpp+=3*x1+4*x2<=84
print(lpp)
lpp.solve()
print("\nx1=",value(x1),"\nx2=",value(x2),"Zmax=",value(lpp.objective))
```

OUTPUT:

```
Problem:
MAXIMIZE
2*x1 + 3*x2 + 4*x3 + 0
SUBJECT TO
_C1: 3 x1 - 2 x3 <= 41
```

```
_C2: 2 x1 + x2 + x3 <= 35
_C3: 2 x2 + 3 x3 <= 30
VARIABLES
x1 Continuous
x2 Continuous
x3 Continuous
x1= 10.0
x2= 15.0
x3= 0.0 Zmax= 65.0
```

Q.8. Solve the LPP:

Maximize Z =$6x_1+3x_2$

Subject to, $-2x_1+3x_2<=9$

$-x_1+3x_2 <=12$

$x_1>=0$, $x_2>=0$

Program:

```
from pulp import*
lpp=LpProblem('Problem',LpMaximize)
x1=LpVariable('x1',lowBound=0)
x2=LpVariable('x2',lowBound=0)
lpp+=6*x1+3*x2
lpp+=-2*x1+3*x2<=9
lpp+=-x1+3*x2<=12
print(lpp)
lpp.solve()
print("\nx1=",value(x1),"\nx2=",value(x2),"Zmax=",value(lpp.objective))
```

OUTPUT:

```
Problem:
MAXIMIZE
6*x1 + 3*x2 + 0
SUBJECT TO
_C1: - 2 x1 + 3 x2 <= 9
_C2: - x1 + 3 x2 <= 12
VARIABLES
x1 Continuous
x2 Continuous
x1= 0.0
x2= 0.0
Zmax= 0.0
```

Q.9. Solve the LPP:

Maximize Z = $3x_1+2x_2+5x_3$

Subject to, $x_1+x_2+x_3<=9$

$2x_1+3x_2+5x_3<=30$

$2x_1-x_2-x_3<=8$

$x_1>=0$, $x_2>=0$, $x_3>=0$

Program:

```
from pulp import*
lpp=LpProblem('Problem',LpMaximize)
x1=LpVariable('x1',lowBound=0)
x2=LpVariable('x2',lowBound=0)
x3=LpVariable('x3',lowBound=0)
lpp+=3*x1+2*x2+5*x3
lpp+=x1+x2+x3<=9
lpp+=2*x1+3*x2+5*x3<=30
lpp+=2*x1-x2-x3<=8
print(lpp)
lpp.solve()
print("\nx1=",value(x1),"\nx2=",value(x2),"\nx3=",value(x3),"Zmax=",value(lpp.obje
ctive))
```

OUTPUT:

```
Problem:
MAXIMIZE
3*x1 + 2*x2 + 5*x3 + 0
SUBJECT TO
_C1: x1 + x2 + x3 <= 9
_C2: 2 x1 + 3 x2 + 5 x3 <= 30
_C3: 2 x1 - x2 - x3 <= 8
VARIABLES
x1 Continuous
x2 Continuous
x3 Continuous
x1= 5.0
x2= 0.0
x3= 4.0
Zmax= 35.0
```

Q.10. Solve the LPP:

Maximize Z = $3x_1+3x_2+2x_3$

Subject to, $x_1+x_2+2x_3<=20$

$2x_1+ x_2+4x_3<=32$

$x_1>=0, x_2>=0, x_3>=0$

Program:

```
from pulp import*
lpp=LpProblem('Problem',LpMaximize)
x1=LpVariable('x1',lowBound=0)
x2=LpVariable('x2',lowBound=0)
x3=LpVariable('x3',lowBound=0)
lpp+=3*x1+3*x2+2*x3
lpp+=x1+x2+2*x3<=20
lpp+=2*x1+x2+4*x3<=32
```

```
print(lpp)
lpp.solve()
print("\nx1=",value(x1),"\nx2=",value(x2),"\nx3=",value(x3),"Zmax=",value(lpp.objective))
```

OUTPUT:

```
Problem:
MAXIMIZE
3*x1 + 3*x2 + 2*x3 + 0
SUBJECT TO
_C1: x1 + x2 + 2 x3 <= 20
_C2: 2 x1 + x2 + 4 x3 <= 32
VARIABLES
x1 Continuous
x2 Continuous
x3 Continuous
x1= 12.0
x2= 8.0
x3= 0.0
Zmax= 60.0
```

Practical No. 7 2D Transformation(Line , ray, segment)

Q.1. Check whether the given points are collinear: (1,1),(2,2),(5,5),(3,5).

Program:

```
from sympy import*
a=Point(1,1)
b=Point(2,2)
c=Point(5,5)
d=Point(3,5)
p=Point.is_collinear(a,b,c)
print(p)
p1=Point.is_collinear(a,c,d)
print(p1)
```

OUTPUT:

```
True
False
```

Q.2. Check whether the given points are coplanar: (1,0,0),(0,1,0),(0,0,1),(0,0,0).

Program:

```
from sympy import*
a=Point(1,0,0)
b=Point(0,1,0)
c=Point(0,0,1)
d=Point(0,0,0)
p=Point.are_coplanar(a,b,c)
print(p)
p1=Point.are_coplanar(a,b,c,d)
print(p1)
```

OUTPUT:

```
True
False
```

Q.3. Find the distance between the points (1,1) and (2,2).

Program:

```
from sympy import*
a=Point(1,1)
b=Point(2,2)
d=a.distance(b)
print(d)
```

OUTPUT:

```
sqrt(2)
```

Que. 4 Find the coefficients of line passing through 2 points (1,2), (3,4)

Program:

```
from sympy import*
l1=Line(Point(1,2),Point(3,4))
a=l1.coefficients
print(a)
```

OUTPUT:

```
(-2, 2, -2)
```

Que.5 Find the equation of line passing through point (1,0) with slope -1

Program:

```
from sympy import*
l2=Line(Point(1,0),slope=-1)
b=l2.equation()
print(b)
```

OUTPUT:

```
x + y – 1
```

Que.6 Find the coefficients of line x+y-1=0

Program:

```
from sympy import*
x,y=symbols('x,y')
l3=Line(x+y-1)
c=l3.coefficients
print(c)
```

OUTPUT:

```
(1, 1, -1)
```

Que. 7 A] Define 1) Line passing through (2,2),(3,3)
2) Ray passing through (0,0), (1,1)
3) Segment passing through (0,0), (1,0)

B] Find angle between them
C] Find point of intersection
D] Find length
E] Find the distance of point P(1,2) from line, ray, segment
F] Find the slopes
G] Find the midpoint of segment
H] Write any 2 points on line, segment & ray
I] Rotate the line by angle $\pi/2$, segment by angle $\pi/2$ and ray by angle $3\pi/2$

Program:

A]

```
 from sympy import*
l=Line(Point(2,2),Point(3,3))
s=Segment(Point(0,0),Point(1,0))
r=Ray(Point(0,0),Point(1,1))
print(l)
print(s)
print(r)
```

OUTPUT:

```
Line2D(Point2D(2, 2), Point2D(3, 3))
Segment2D(Point2D(0, 0), Point2D(1, 0))
Ray2D(Point2D(0, 0), Point2D(1, 1))
```

B]

Program:

```
a=l.angle_between(s)
b=l.angle_between(r)
```

```
c=s.angle_between(r)
print(a)
print(b)
print(c)
```

OUTPUT:

```
pi/4
0
pi/4
```

C]

Program:

```
x=l.intersection(s)
y=l.intersection(r)
z=s.intersection(r)
print(x)
print(y)
print(z)
```

OUTPUT:

```
[Point2D(0, 0)]
[Ray2D(Point2D(0, 0), Point2D(1, 1))]
[Point2D(0, 0)]
```

D]

Program:

```
o=l.length
A=s.length
V=r.length
print(o)
print(A)
print(V)
```

OUTPUT:

```
oo
1
Oo
```

E]

Program:

```
p=Point(1,2)
q=l.distance(p)
w=r.distance(p)
e=s.distance(p)
print(q)
print(w)
print(e)
```

OUTPUT:

```
sqrt(2)/2
sqrt(2)/2
2
```

F]

Program:

```
f=l.slope
g=s.slope
h=r.slope
print(f)
print(g)
print(h)
```

OUTPUT:

```
1
0
1
```

G]

Program:

```
y=s.midpoint
print(y)
```

OUTPUT:

```
Point2D(1/2, 0)
```

H]

Program:

```
b=l.points
n=s.points
m=r.points
print(b)
print(n)
print(m)
```

OUTPUT:

```
(Point2D(2, 2), Point2D(3, 3))
(Point2D(0, 0), Point2D(1, 0))
(Point2D(0, 0), Point2D(1, 1))
```

I]

Program:

```
j=l.rotate(pi/2)
k=r.rotate(3*pi/2)
v=s.rotate(pi/2)
print(j)
print(k)
print(v)
```

OUTPUT:

```
Line2D(Point2D(-2, 2), Point2D(-3, 3))
Ray2D(Point2D(0, 0), Point2D(1, -1))
Segment2D(Point2D(0, 0), Point2D(0, 1))
```

Q.8 Rotate the line passing through the points A[0,0], B[0,1] about the origin trough an angle $\pi/6$

Program:

```
from sympy import*
l=Line(Point(0,0),Point(0,1))
```

```
L=l.rotate(pi/6)
a=L.equation()
print(a)
```

OUTPUT:

```
-sqrt(3)*x/2 - y/2
```

Q.9 Rotate the segment passing through the points A[1,0], B[2,-1] about the origin trough an angle π

Program:

```
from sympy import*
l=Segment(Point(1,0),Point(2,-1))
L=l.rotate(pi)
print(L)
```

OUTPUT:

```
Segment2D(Point2D(-1, 0), Point2D(-2, 1))
```

Q.10 Rotate the ray passing through the points A[0,0], B[4,4] about the origin trough an angle $\pi/2$

Program:

```
from sympy import*
l=Ray(Point(0,0),Point(4,4))
L=l.rotate(pi/2)
print(L)
```

OUTPUT:

```
Ray2D(Point2D(0, 0), Point2D(-4, 4))
```

Q.11 Reflect the line 4x+3y-5=0 through the line x+y

Program:

```
from sympy import*
x,y=symbols('x,y')
l1=Line(4*x+3*y-5)
l2=Line(x+y)
a=l1.reflect(l2)
b=a.equation()
print(b)
```

OUTPUT:

```
x + 4*y/3 + 5/3
```

Q.12 Reflect the segment passing through [2,3], [4,6] through the line 7x+6y-3=0

Program:

```
from sympy import*
x,y=symbols('x,y')
l1=Segment(Point(2,3),Point(4,6))
l2=Line(7*x+6*y-3)
a=l1.reflect(l2)
print(a)
```

OUTPUT:

```
Segment2D(Point2D(-236/85, -93/85), Point2D(-514/85, -222/85))
```

Q.13 Let A[2,1], B[4,-1] and $[T] = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ Find equation of transformed line A’B’

Program:

```
from sympy import*
A=Point(1,1)
B=Point(-4,-1)
a=A.transform(Matrix([[1,-2,0],[-2,1,0],[0,0,1]]))
b=B.transform(Matrix([[1,-2,0],[-2,1,0],[0,0,1]]))
l=Segment(a,b)
print(l)
```

OUTPUT:

```
Segment2D(Point2D(-1, -1), Point2D(-2, 7))
```

Q.14 Let A[2,5], B[4,-13] be transformed to A’ and B’ under $[T] = \begin{bmatrix} 2 & 3 & 0 \\ 4 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Find the midpoint of segment A’B’

Program:

```
from sympy import*
A=Point(2,5)
B=Point(4,-13)
a=A.transform(Matrix([[2,3,0],[4,1,0],[0,0,1]]))
b=B.transform(Matrix([[2,3,0],[4,1,0],[0,0,1]]))
l=Segment(a,b)
m=l.midpoint
print(m)
```

OUTPUT:

```
Point2D(-10, 5)
```

Practical No. 8 Polygon and Triangle

Q1.Drown a polygon with vertices (0,0),(1,0),(2,2),(1,4) and find its area and perimeter.

Program:

```
>>> from sympy import*
>>> A = Point(0,0)
>>> B=Point(1,0)
>>> C=Point(2,2)
>>> D=Point(1,4)
>>>p=Polygon (A,B,C,D)
>>> p
Polygon(Point2D(0, 0), Point2D(1, 0), Point2D(2, 2), Point2D (1, 4) )
>>>p.area
4
>>>p.perimeter
1 + sqrt(17) +2^ * sqrt(5)
>>>
```

Q2.Drown a regular polygon with 4 sides and radius 6 centered at origin and find its area andperimeter.

Program:

```
>>> A=Point(0,0)
>>>P=Polygon(A,6,n=4)
>>> P
Regular Polygon(Point2D(0, 0), 6, 4, 0)>>>P.area
72
>>>P.perimeter
24*sqrt(2)
```

Q3.Drown a regular polygon with 8 sides and radius 2 centered at (-1,2) and find its area and perimeter.

Program:

```
>>> a=Point(-1,2)
>>> P=Polygon(a,2,n=8)
>>> P
Regular Polygon( 2D(- 1, 2) 2, 8, 0)>>>P.area
(64-32* sqrt(2))/(-4+4*sqrt(2))
>>>P.perimeter
16^ * sqrt(2 - sqrt(2))
```

Q4.Draw a regular polygon with 7 sides and radius 6 centered at (-2,2) and reflect it through line x-2y=5

Program:

```
>>> A=Point(-2,2)
>>>P=Polygon (A,6,n=7)
Regular Polygon(Point2D(-2, 2), 6, 7, 0)
>>>x,y symbols('x,y')
>>> b=Line(x-2*y-5)
>>> c=P.reflect(b)
RegularPolygon(Point2D(12/5, -34/5), -6, 7, -2*pi/7+ atan(4/3))
```

Q5.Drown a polygon with vertices (0,0),(-2,0), (5,5),(1,-6) and rotate by 180 degres and find internal angle at each vertex.

Program:

```
>>> A=Point(0,0)
>>>B=Paint(-2,0)
>>> C=Point(5,5)
>>> D=Point(1,-6)
>>> P=Polygon(A,B,C,D)
>>> P
```

```
Polygon(Point2D(0, 0), Point2D(-2, 0), Point2D(5, 5), Point2D(1, -6))
>>>P.rotate(pi)

Polygon(Point2D(0, 0), Point2D(2, 0), Point2D(-5, -5), Point2D(-1, 6))
>>>P.angles[A]
acos(-sqrt(37)/37)
>>>P.angles[B]
-acos(7* sqrt(74)/74)+2*pi
>>>P.angles[C]
-acos(83* sqrt(10138)/10138) + 2*pi
>>>P.angles[D]
-acos(62*sqrt(5069)/5069)+2*pi
```

Q6.Reflect the triangle ABC through the line y=-3, where A[1,1],B[2,-3), C[-1,5]
Program:

```
>>> A=Paint(1,1)
>>> B=Point(2,-3)
>>> C=Point(-1,5)
>>>T=Triangle(A,B,C)
>>>T
Triangle(Point2D(1, 1), Point2D(2, -3), Point2D(-1, 5))
>>> a=Line(Point(0,-3),Point(1,-3))
>>>a.equation()
y+3
>>> b=T.reflect(a)
>>> b
Triangle(Point2D(1, -7), Point2D(2, -3), Point2D(-1, -11))
```

Q7.Rotate the triangle ABC by 90,where A[1,-2], B[4,-6], C[-1,4].
Program:

```
>>> A=Point(1,-2)
>>> B=Point(4,-6)
>>> C=Point(-1,4)
>>>T=Triangle(A,B,C)
>>>T.rotate(pi/2)
Triangle(Point2D(2, 1), Point2D(6, 4), Point2D(-4, -1))
```

Q8. Find the area and perimeter of the triangle ABC,where A[0,1], B[-5,0], C[3,-3].
Program:

```
>>> A=Point(0,1)
>>> B=Point(-5,0)
>>> C=Point(3,-3)
>>>T=Triangle(A,B,C)
```

```
>>> T
Triangle(Point2D(0, 1), Point2D(-5, 0), Point2D(3,
>>>T.area
23/2
>>>T.perimeter
5 + sqrt(26) + sqrt(73)
```

Q9.Find the angle at each vertices of the triangle ABC, where A[1, 1] B[1, 2] C[0,1].

Program:

```
>>> A=Point(1,1)
>>> B=Point(1,2)
>>> C=Point(0,1)
>>>T=Triangle(A,B,C)
>>>T
Triangle(Point2D(11), Point2D(1, 2), Point2D(0, 1))
>>>T.angles[A]
pi/2
>>>T.angles[B]
pi/4
>>>T.angles[C]
pi/4
```

Q10. Reflect the triangle ABC through the line y = x + 3 where A[- 1, 0] 1,0], B[2, - 1] ,C[1,3].

Program:

```
>>> A=Point(-1,0)
>>> B=Point(2,-1)
>>> C=Point(1,3)
>>>T=Triangle(A,B,C) >>>
x,y=symbols('x,y')
>>> a=Line(x-y+3)
>>> b=T.reflect(a)
>>> b
Triangle(Point2D(-3, 2), Point2D(-4, 5)Point2D . D(0, 4) )
```

Q11.Rotate the triangle ABC by 270, where A[- 1, 2] B[2, - 5] C(- 1,7)

```
>>> A=Point(-1,2)
>>> B=Point(2,-5)
>>> C=Point(-1,7)
>>>T=Triangle(A,B,C)
>>>T.rotate (3 * pi / 2 )
Triangle ( Point2D(2, 1) , Point2D(-5, -2)2D(7, 1) )
```

Q12.Find the area and perimeter of the triangle ABC, where A[0, 1] , B[- 5, 0] C(-3,3)

Program:

```
>>> A=Point(0,1)
>>> B=Point(-5,0)
>>>C=Point(-3,3)
>>>T=Triangle(A,B,C)
>>>T
Triangle(Point2D(0, 1), Point2D(-5, 0), Point2D(-3, 3))
>>>T.area
-13/2
>>>T.perimeter
sqrt(26)+2*sqrt(13)
```

Q13.Find the angle at each vertices of the triangle PQR, where p[1,0], Q[2,3], R[0,-2]

Program:

```
>>> A=Point(1,0)
>>> P=Point(1,0)
>>> Q=Point(2,3)
>>> R=Paint(0,-2)
>>>T=Triangle(P,Q,R)
>>>T
Triangle(Point2D(1, 0), Point2D(2, 3), Point2D(0, -2))
>>> Tangles[P]
acos(-7* sqrt(2)/10)
>>> Tangles[Q]
acos(17* sqrt(290)/290)
>>>T.angles[R]
acos(12* sqrt(145)/145)
```