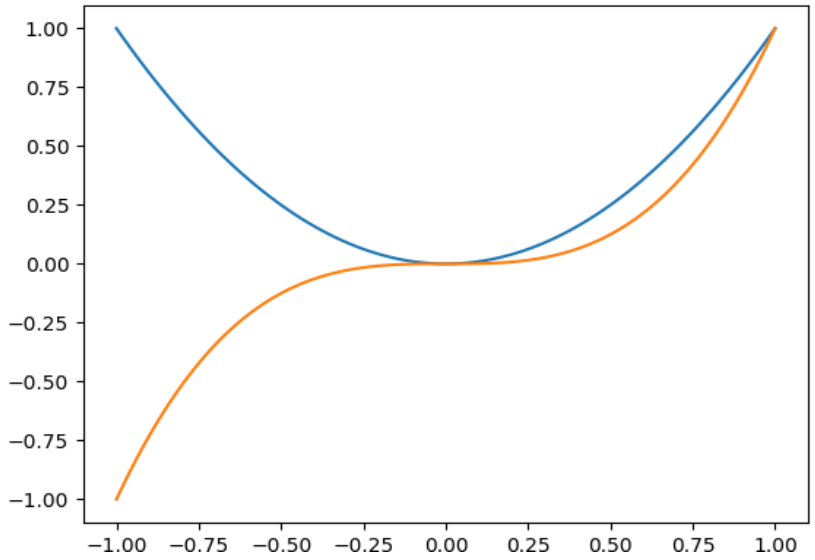
/***
Asdfjk Asdfjk
Class: S.Y.Bsc.Computer Science Asdfjk
Practical No:01
Practical Name: Graph Plotting (2 Dimensional)
***/

```
from pylab import*
import matplotlib.pyplot as plt
import numpy as np
```

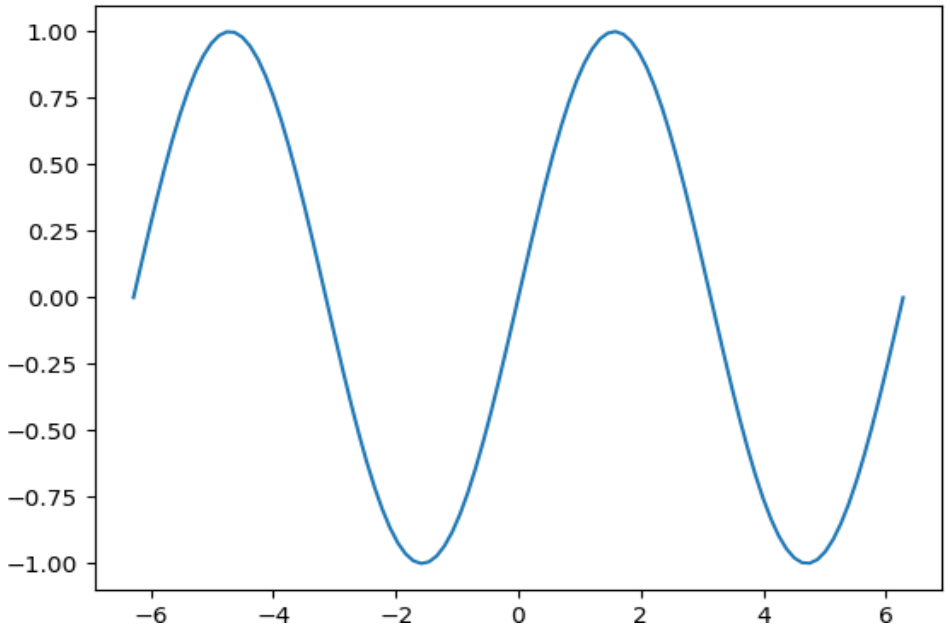
Q.1 Plot the graph og $f(x)=x^2$ and $g(x)=x^3$ in [-1,1]

```
x=np.linspace(-1,1,100)
f=x**2
g=x**3
plot(x,f)
plot(x,g)
show()
x=np.linspace(-2*pi,2*pi,100)
f=np.sin(x)
plot(x,f)
show()
# Output
```



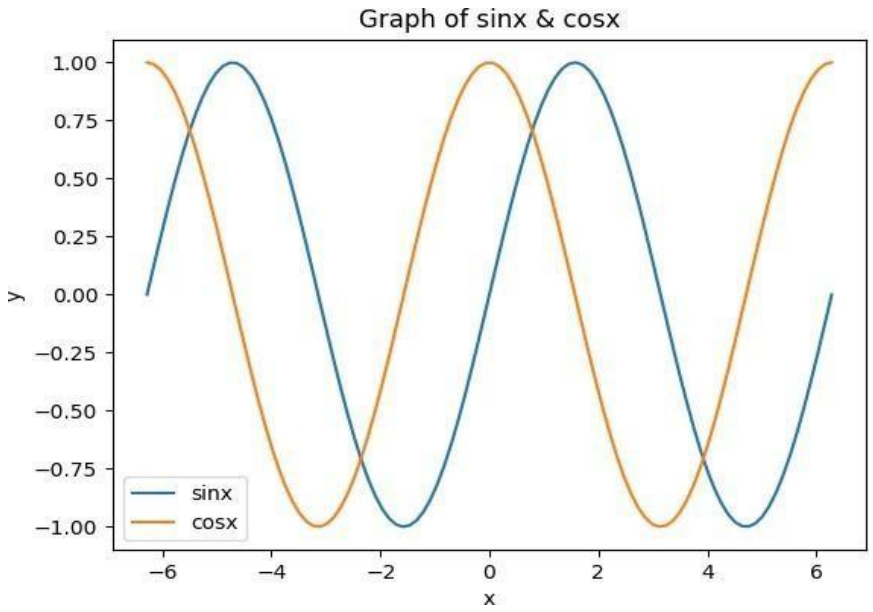
Q.2 Plot the graph of f(x) = sin(x) in [-2π,2 π]

```
x=np.linspace(-2*pi,2*pi,100)

f=np.sin(x)

g=np.cos(x)

plot(x,f,label='sinx')

plot(x,g,label='cosx')

xlabel('x')

ylabel('y')

title('Graph of sinx &cosx')

legend()

show()
#Output
```



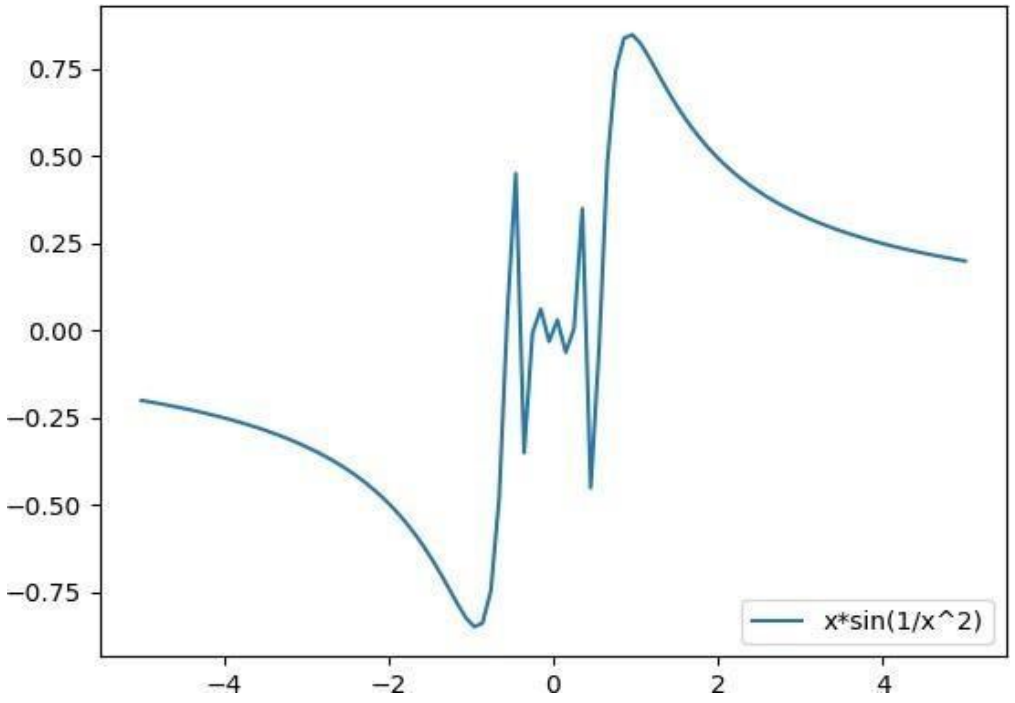
Q.3 Plot the graph of f(x)=sin(x) and g(x)=cos(x) in [-2π,2 π].

```
x=np.linspace(-2*pi,2*pi,100)
f=np.sin(x)
g=np.cos(x)
plot(x,f,label='sinx')
plot(x,g,label='cosx')
xlabel('x')
ylabel('y')
title('Graph of sinx &cosx')
legend()
show()
#Output
```



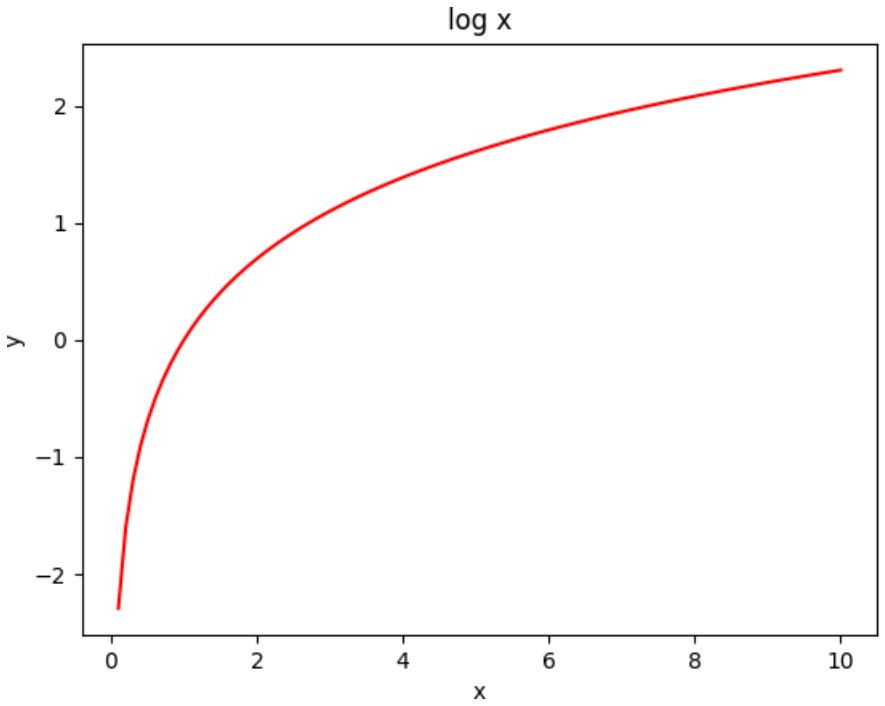
Q.4 Plot the graph of f(x)= xsin(1/x²) in [-5,5].

```
x=np.linspace(-5,5,100)
y=x*np.sin(1/(x**2))
plot(x,y,label="x*sin(1/x^2)")
legend(loc=4)
show()
#Output
```



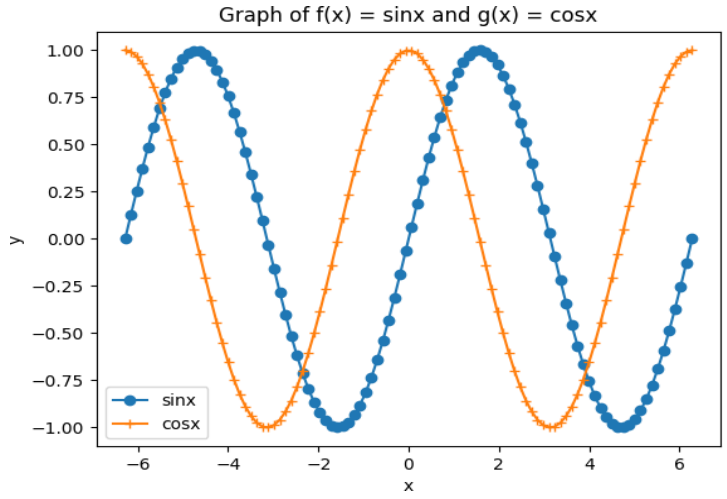
Q.5 Plot the graph of f(x) = logx in [0,10].

```
x=np.linspace(0,10,100)
y=np.log(x)
plot(x,y,'r')
xlabel('x')
ylabel('y')
title('log x')
show()
#Output
```



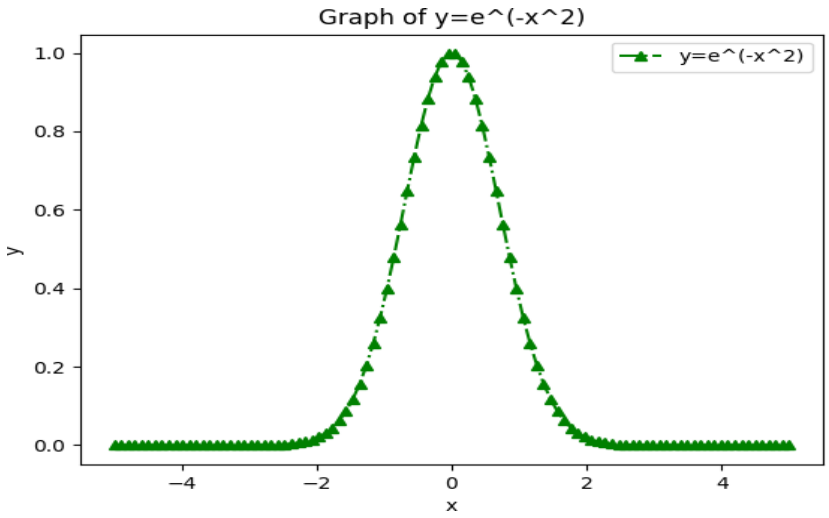
Q.6 Plot the graph of f(x)=sinx and g(x)=cosx with red dashed circle markers.

```
x=np.linspace(-2*pi,2*pi,100)
f=np.sin(x)
g=np.cos(x)
plot(x,f,label='sinx',marker='o')
plot(x,g,label='cosx',marker='+')
xlabel('x')
ylabel('y')
title('Graph of f(x) = sinx and g(x) = cosx')
legend()
show()
#Output
```



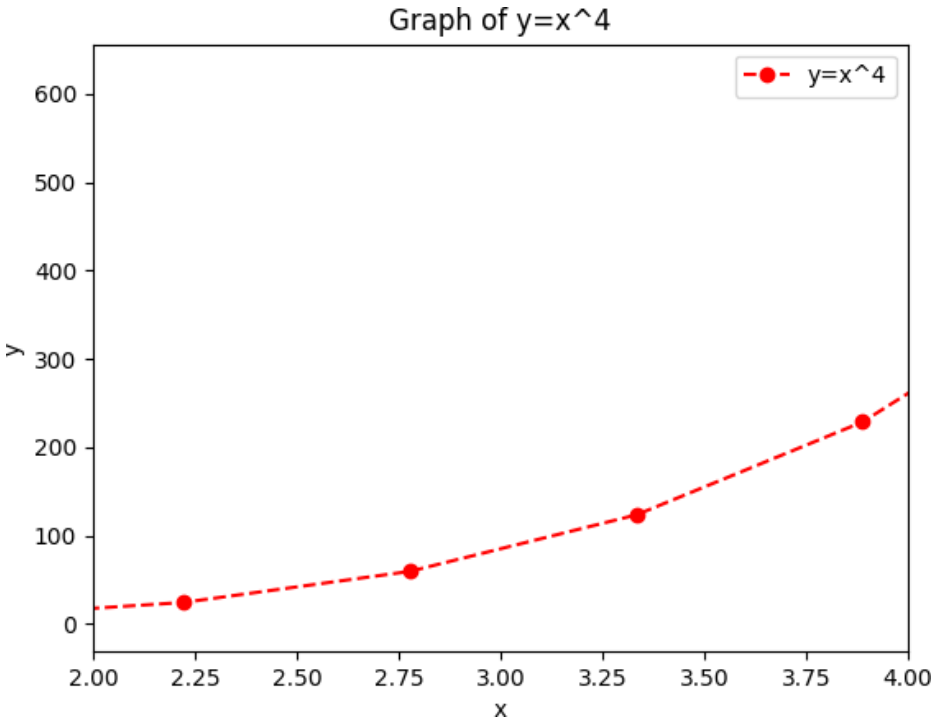
Q.7 Plot the graph of f(x) = e^{-x^2} in [-5,5] with green dashed-points line with Upward-pointing triangle.

```
x=np.linspace(-5,5,100)

y = np.exp(-x**2)

plot(x,y,"-.^g",label="y=e^(-x^2)")

xlabel('x')

ylabel('y')

title('Graph of y=e^(x^2)')

legend()

show()

# Output
```



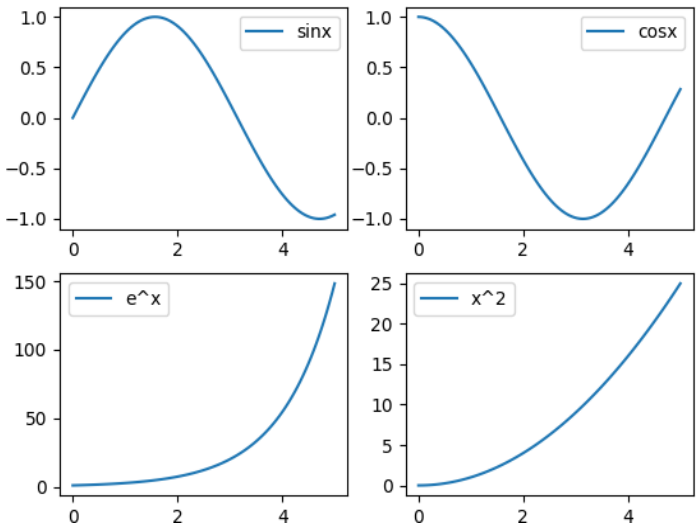
Q.8 Plot the graph of $f(x)=x^4$ in [0,5].

```
x = np.linspace(0,5,10)
y=x**4
plot(x,y,"--or",label="y=x^4")
xlabel('x')
ylabel('y')
xlim([2,4])
title('Graph of y=x^4')
legend()
show()
# Output
```



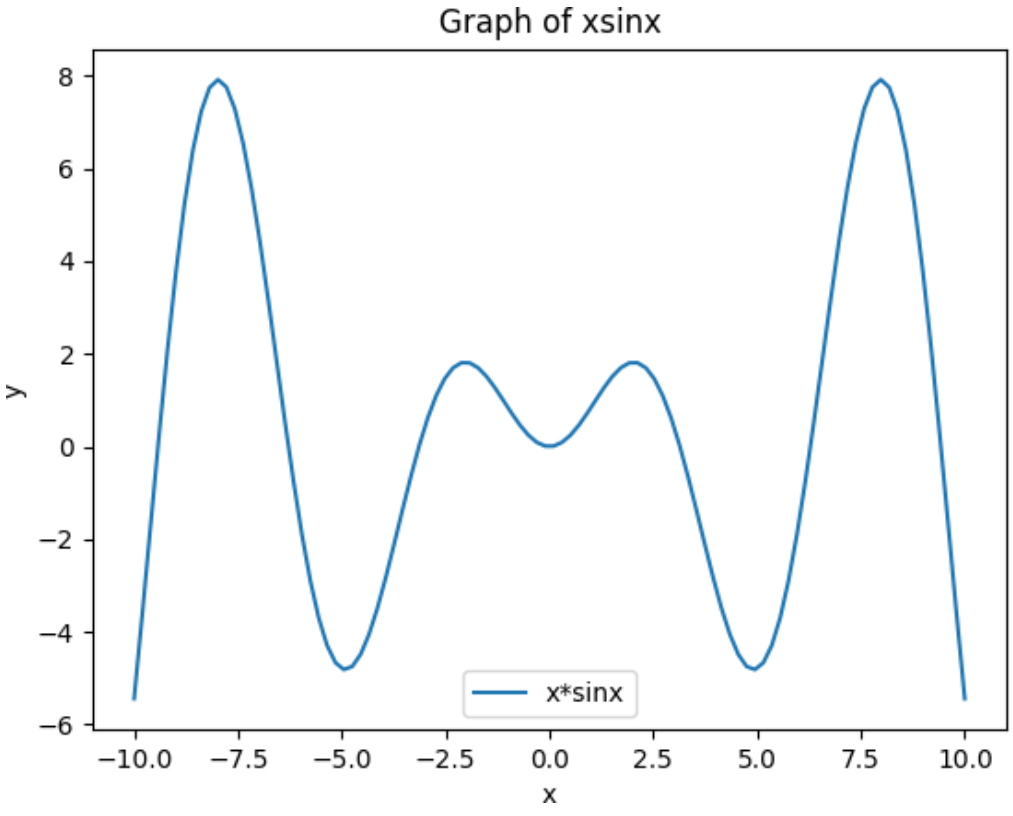
Q.9 Plot the graph of sinx, cosx, e^x and x^2 in [0,5] in one figure with (2X2) subplots.

```
x = np.linspace(0,5,100)
y1=np.sin(x)
y2=np.cos(x)
y3=np.exp(x) y4=x**2
subplot(2,2,1)
plot(x,y1,label="sinx")
legend()
subplot(2,2,2)
plot(x,y2,label="cosx")
legend()
subplot(2,2,3)
plot(x,y3,label="e^x")
legend()
subplot(2,2,4)
plot(x,y4,label="x^2")
legend()
show()
#Output
```



Q.10 Plot the graph of f(x)=xsinx in [-10,10].

```
x = np.linspace(-10,10,100)
y = x*np.sin(x)
plot(x,y,label="x*sinx")
xlabel('x')
ylabel('y')
title('Graph of xsinx')
legend()
show()
#Output
```

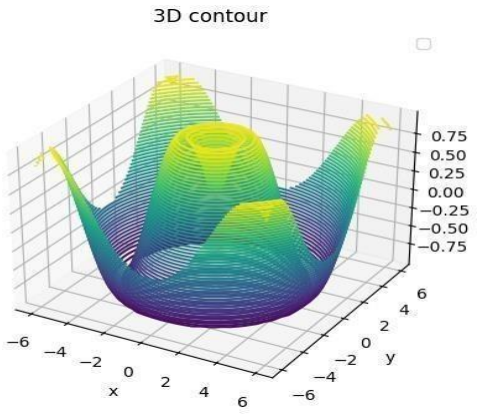


/***
Asdfjk Asdfjk
Class: S.Y.Bsc.Computer Science Asdfjk
Practical No:02
Practical Name: Graph Plotting (3 Dimensional)
***/

```
from mpl_toolkits import mplot3d

import numpy as np

from pylab import *
```

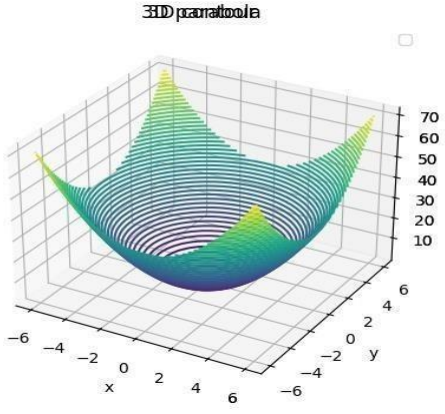
Q.1 Plot the graph of f(x) = sin(x²+y²) in -6 < x,y < 6

```
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)

ax =axes(projection='3d')

ax.contour(X,Y,Z,50)

xlabel('x')

ylabel('y')

title('3D contour')

legend()

show()

#Output
```



Q.2 Plot the parabola $z = x^2 + y^2$ in $-6 < x,y < 6$.

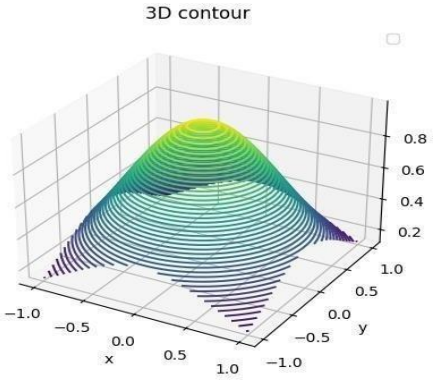
```
def f(x,y):
    return x**2+y**2
x = np.linspace(-6,6,30)
y = np.linspace(-6,6,30)
X,Y  =  np.meshgrid(x,y)
Z = f(X,Y)
ax =axes(projection='3d')
ax.contour3D(X,Y,Z,50)
xlabel('x')
ylabel('y')
title('3D parabola')
legend()
show()
#Output
```



Q.3 Plot the graph of f(x) = $e^{-x^2-y^2}$ in $-1 < x,y < 1$.

```
def f(x,y):
    return np.exp(-x**2-y**2)
x = np.linspace(-1,1,30)
```

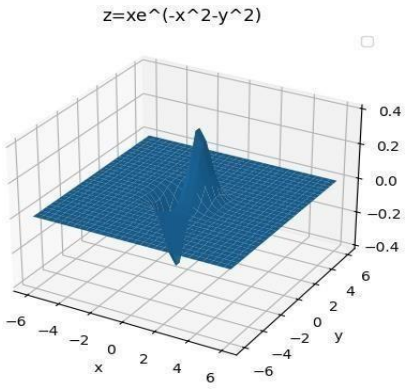
```
y = np.linspace(-1,1,30)
X,Y  =  np.meshgrid(x,y)
Z = f(X,Y)
ax =axes(projection='3d')
ax.contour3D(X,Y,Z,50)
xlabel('x')
ylabel('y')
Text(0.5, 0.5,'y')
title('3D contour')
legend()
show()
#Output
```



Q.4 Plot the function z = x$e^{-x^2-y^2}$ in -6 < x,y < 6.

```
def f(X,Y):
    return X*np.exp(-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)
ax = axes(projection='3d')
```

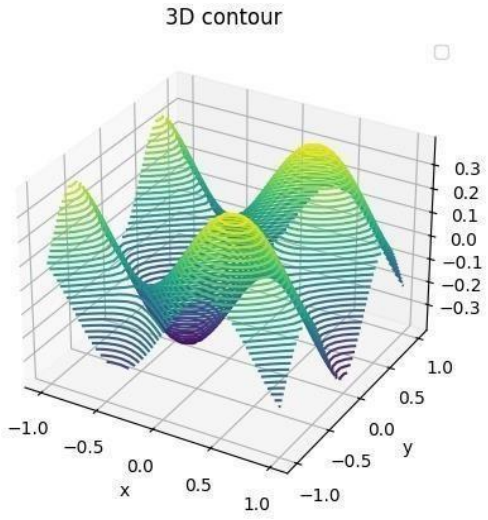
```
ax.plot_surface(X,Y,Z)
xlabel('x')
ylabel('y')
title('z=xe^(-x^2-y^2)')
legend()
show()
#Output
```



Q.5 Plot the graph of f(x) $\frac{\sin(4x)-\cos\ (5y)}{5}$ in -1<x,y<1.

```
= def f(x,y):
    return (np.sin(4*x)-np.cos(5*y))/5
x = np.linspace(-1,1,30)
y = np.linspace(-1,1,30)
X,Y = np.meshgrid(x,y)
Z = f(X,Y)
ax =axes(projection='3d')
ax.contour3D(X,Y,Z,50)
xlabel('x')
ylabel('y')
title('3D contour')
```

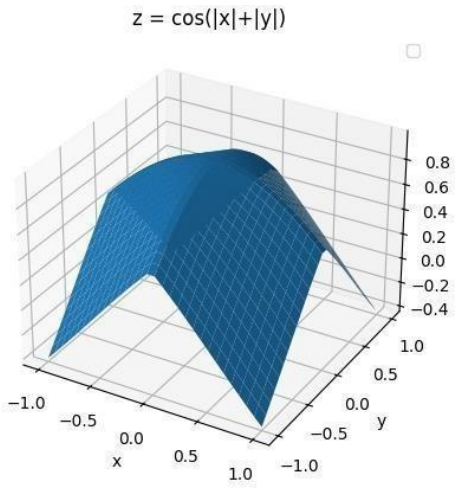
```
legend()
show()
#Output
```



Q.6 Plot the function z = cos(|x|+|y|) in -1<x,y<1.

```
def f(X,Y):
    return np.cos(abs(X)+abs(Y))
x = np.linspace(-1,1,30)
y = np.linspace(-1,1,30)
X,Y  =  np.meshgrid(x,y)
Z = f(X,Y)
ax =axes(projection='3d')
ax.plot_surface(X,Y,Z)
xlabel('x')
ylabel('y')
title('z = cos(|x|+|y|)')
legend()
show()
```

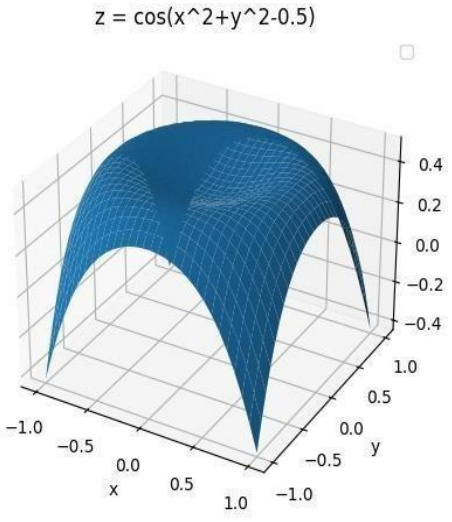
#Output



Q.7 Plot the function $z = \cos(x^2+y^2-0.5)-0.5$ in $-1<x,y<1$.

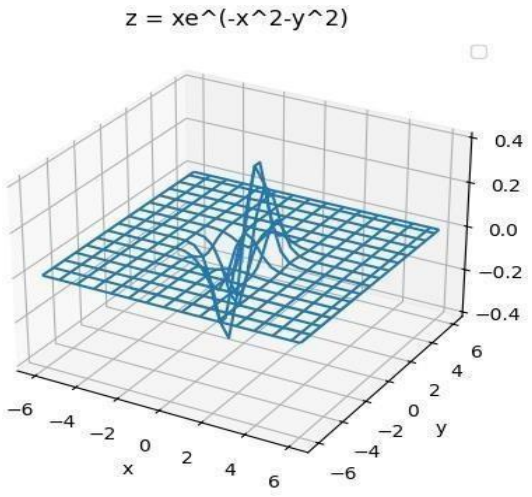
```
def f(X,Y):
    return np.cos(X**2+Y**2-0.5)-0.5
x = np.linspace(-1,1,30)
y = np.linspace(-1,1,30)
X,Y = np.meshgrid(x,y)
Z = f(X,Y)
ax =axes(projection='3d')
ax.plot_surface(X,Y,Z)
xlabel('x')
ylabel('y')
title('z = cos(x^2+y^20.5)')
legend()
show()
```

#Output



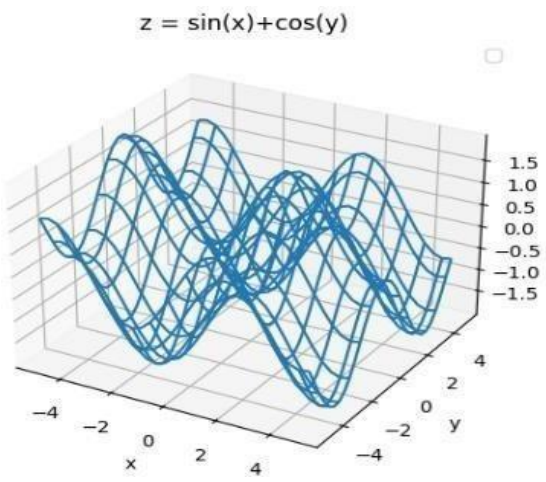
Q.8 Plot the function z = x$e^{-x^2-y^2}$ in -6<x,y<6.

```
def f(X,Y):
    return X*np.exp(-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)
ax = axes(projection='3d')
ax.plot_wireframe(X,Y,Z,rstride=2,cstride=2)
xlabel('x')
ylabel('y')
title('z = xe^(-x^2-y^2)')
legend()
show()
#Output
```



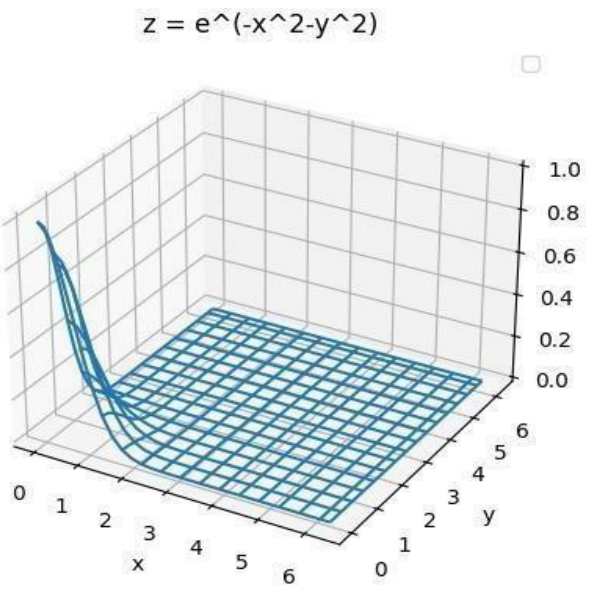
Q.9 Plot the function z = sin(x)+cos(y) in -5<x,y<5.

```
def f(X,Y):
   return np.sin(X)+np.cos(Y)

 x = np.linspace(-5,5,30)
y = np.linspace(-5,5,30)
X,Y =  np.meshgrid(x,y)
Z = f(X,Y)
ax = axes(projection='3d')
ax.plot_wireframe(X,Y,Z,rstride=2,cstride=2)
xlabel('x')
ylabel('y')
title('z = sin(x)+cos(y)')
legend()
show()
#Output
```



Q.10 Plot the function z = $e^{-x^2-y^2}$ for xϵ[0,2π], yϵ[0,2π].

```
def f(X,Y):
    return np.exp(-X**2-Y**2)
x = np.linspace(0,2*pi,30)
y = np.linspace(0,2*pi,30)
X,Y = np.meshgrid(x,y)
Z = f(X,Y)
ax = axes(projection='3d')
ax.plot_wireframe(X,Y,Z,rstride=2,cstride=2)
xlabel('x')
ylabel('y')
title('z = e^(-x^2y^2)')
legend()
show()
#Output
```


z = e^(-x^2-y^2)
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
x
0
1
2
3
4
5
6
y

/**

Name:Asdfjk Asdfjk

Class:S.Y.B.Sc Computer Science Asdfjk

Practical No:3

Practical Name: **Application to computaional Geometry - I**

**/

```
from sympy import *
```

Q.1 Check whether the points are collinear or not

```
x =  Point(0,0)
y  = Point(2,2)
z = Point(-1,-1)
w = Point(3,4)
Point.is_collinear(x,y,z)
```

#output

```
True
```

```
Point.is_collinear(x,y,w)
```

#output

```
False
```

Q.2 check whether the points are coplanar or not

```
x=Point(0,0,0)
y=Point(2,2,2)
z = Point(-1,-1,-1)
w = Point(3,4,-7)
Point.are_coplanar(x,y,z,w)
```

#output

```
True
```

Q.3 Find the distance between points x,y; y,w and x,z if x=[0,0], y=[2,2], z=[-1,-1] and w =[3,4]

```
x =  Point(0,0)
y  = Point(2,2)
z = Point(-1,-1)
w = Point(3,4)
x.distance(y)
```

#output

22—√22

```
y.distance(w)
```

#output

5—√5

```
y.distance(z)
```

#output
2—√2

Transformation of a point

```
x = Point(3,4)
x.scale(2,2)
```

#output
Point2D(6,8)Point2D(6,8)

```
x.scale(1,0)
```

#output
Point2D(3,0)Point2D(3,0)

```
x.scale(1,0)
```

#output
Point2D(3,0)Point2D(3,0)

```
x.scale(3,7)
```

#output
Point2D(9,28)Point2D(9,28)

Reflection

```
x =  Point(3,4)
y = Point(-4,-8)
x.transform(Matrix([[-1,0,0],[0,1,0],[0,0,1]]))
```

#output
Point2D(−3,4)Point2D(−3,4)

```
y.transform(Matrix([[-1,0,0],[0,1,0],[0,0,1]]))
```

#output
Point2D(4,−8)Point2D(4,−8)

```
x = Point(1,4)
y = Point(-4,3)
x.transform(Matrix([[1,0,0],[0,-1,0],[0,0,1]]))
```

#output
Point2D(1,−4)Point2D(1,−4)

```
y.transform(Matrix([[1,0,0],[0,-1,0],[0,0,1]]))
```

#output

Point2D(−4,−3)Point2D(−4,−3)

```
x = Point(2.3,4)
y = Point(3,3)
x.transform(Matrix([[-1,0,0],[0,-1,0],[0,0,1]]))
```

#output
Point2D(−2310,−4)Point2D(−2310,−4)

```
y.transform(Matrix([[-1,0,0],[0,-1,0],[0,0,1]]))
```

#output
Point2D(−3,−3)Point2D(−3,−3)

```
x = Point(5,4)
y = Point(3,-2)
x.transform(Matrix([[0,1,0],[1,0,0],[0,0,1]]))
```

#output
Point2D(4,5)Point2D(4,5)

```
y.transform(Matrix([[0,-1,0],[-1,0,0],[0,0,1]]))
```

#output
Point2D(2,−3)Point2D(2,−3)

Q.4 Reflect the given points through respective lines.

1. (3,6), x+y = 0

2. (2,6), 2x+y = -1

3. (0,-2), x+y = 5

4. (1.5,3.6), x-2y = 1

5. (-5,-6), -4x+3y = 11

```
x,y = symbols('x y')
p = Point(3,6)
p.reflect(Line(x+y))
```

#output
Point2D(−6,−3)Point2D(−6,−3)

```
x,y = symbols('x y')
p = Point(2,6)
p.reflect(Line(2*x+y+1))
```

#output
Point2D(−345,85)Point2D(−345,85)

```
x,y = symbols('x y')
p = Point(0,-2)
p.reflect(Line(x+y-5))
```

#output
Point2D(7,5)Point2D(7,5)

```
x,y = symbols('x y')
p = Point(1.5,3.6)
p.reflect(Line(x-2*y-1))
```

#output
Point2D(20950,−4425)Point2D(20950,−4425)

```
x,y = symbols('x y')
p = Point(-5,-6)
p.reflect(Line(-4*x+3*y-11))
```

#output
Point2D(−19725,−9625)Point2D(−19725,−9625)

Q.4 Reflect the point P[3 6] through the line x-2y+4 = 0

```
x,y = symbols('x y')
p = Point(3,6)
p.reflect(Line(x-2*y+4))
```

#output
Point2D(5,2)Point2D(5,2)

Shearing

```
x = Point(3,-4)
y = Point(3,-1)
# shearing int he X direction by 3 units.
x.transform(Matrix([[1,3,0],[0,1,0],[0,0,1]]))
```

#output
Point2D(3,5)Point2D(3,5)

```
# shearing in the Y direction by 7 units.

y.transform(Matrix([[1,0,0],[7,1,0],[0,0,1]]))
```

#output
Point2D(−4,−1)Point2D(−4,−1)

Rotation

```
x = Point(1,4)
y = Point(-4,7)


x.rotate(pi/2)
```

#output
Point2D(−4,1)Point2D(−4,1)

```
y.rotate(pi/3)
```

#output
Point2D(−73—√2−2,72−23—√)Point2D(−732−2,72−23)

Q.5 Apply each of the following transformations on the point P = [4,3]

1. Reflection through Y-axis

```
P = Point(4,3)
P.transform(Matrix([[-1,0,0],[0,1,0],[0,0,1]]))
```

#output
Point2D(−4,3)Point2D(−4,3)

2. Scaling in X-coordinate by factor 3.

```
P.scale(3,0)
```

#output
Point2D(12,0)Point2D(12,0)

3. Scaling in Y-coordinate by factor 3.2

```
P.scale(0,3.2)
```

#output
Point2D(0,485)Point2D(0,485)

4. Reflection through the line y = -x

```
x,y = symbols('x y')
P.reflect(Line(x+y+0))
```

#output
Point2D(−3,−4)Point2D(−3,−4)

5. Shearing in Y direction by 3 units

```
P.transform(Matrix([[1,0,0],[3,1,0],[0,0,1]]))
```

#output
Point2D(13,3)Point2D(13,3)

6. Scaling in X and Y direction by 3/2 and 2 units respectively

```
P.scale(3/2,2)
```

#output
Point2D(6,6)Point2D(6,6)

7. Shearing in both X and Y direction by -3 and 1 with respectively

```
P.transform(Matrix([[1,-3,0],[1,1,0],[0,0,1]]))
```

#output
Point2D(7,−9)Point2D(7,−9)

8. Rotation about origin by an angle 45 degrees.

```
P.rotate(pi/4)
```

#output
Point2D(2—√2,72—√2)

/**

Name:Asdfjk Asdfjk

Class:S.Y.B.Sc Computer Science Asdfjk

Practical No:4

Practical Name: **Application to computaional Geometry - II**

**/

Lines

```
from sympy import *
L = Line(Point(2,3), Point(4,1))
x,y = symbols('x y')
# Line having equation 2x+3y=4
L = Line(2*x+3*y-4)
L
```

#output

Line2D(Point2D(0,43),Point2D(1,23))Line2D(Point2D(0,43),Point2D(1,23))

```
# Line with point (1,4) and slope = 1
L = Line(Point(1,4), slope=1)
L
```

#output

Line2D(Point2D(1,4),Point2D(2,5))Line2D(Point2D(1,4),Point2D(2,5))

equation of line

```
L.equation()
```

#output

$-x+y-3$

```
L.coefficients
```

#output

```
(-1, 1, -3)
```

Line segment

```
s = Segment((0,0),(0,1))
s
```

#output

Segment2D(Point2D(0,0),Point2D(0,1))Segment2D(Point2D(0,0),Point2D(0,1))

Ray

```
r = Ray((0,0),(3,1))
r
```

#output

Ray2D(Point2D(0,0),Point2D(3,1))Ray2D(Point2D(0,0),Point2D(3,1))

Functions related to lines, rays and segments

```
# Angle between two  linear entities(lines,rays,segments)


l = Line((0,0),(0,3))
s = Segment((0,0),(0,1))
r = Ray((0,0),(8,8))
#angle between line l and segment s
l.angle_between(s)
```

0

```
#angle between line l and segment r
l.angle_between(r)
```

#output

$\pi/4$

```
### Intersection points of two linear entities
l = Line((0,8),(8,0))
s = Segment((0,0),(10,10))
r = Ray((0,3),(3,0))
# Intersection point of line l and segment s
l.intersection(s)
```

#output

```
[Point2D(4, 4)]
#Intersection point of line l and segment r
l.intersection(r)
```

#output

```
[]
```

Length linear entities

```
l = Line ((0,8),(8,0))
s = Segment((0,0),(10,10))
r = Ray((0,3),(3,0))
l.length
```

#output

∞

```
s.length
```

#output

$102—\sqrt{102}$

```
r.length
```

#output

∞

Distance

```
l = Line((0,8),(8,0))
s = Segment((0,0),(10,10))
r = Ray((0,3),(3,0))
```

```
P = Point(10,10)
l.distance(P)
```

#output
$6*\sqrt{2}$

```
s.distance(P)
```

#output
0

```
r.distance(P)
```

#output
$17*\sqrt{2}/2$

Slope of linear entities

```
L = Line(Point(0,0),Point(2,4))
#Slope of line L
L.slope
```

#output
2

```
S = Segment(Point(1,0),Point(2,-1))
#Slope of segment S
S.slope
```

#output
-1

```
R = Ray(Point(0,0),Point(4,4))
#Slope of ray R
R.slope
```

#output
1

Midpoint of segment

```
S = Segment(Point(1,0),Point(2,-1))
#Midpoint of segment S
S.midpoint
```

#output
Point2D(32,−12)Point2D(32,−12)

```
S1 = Segment(Point(3,2),Point(2,-3))
#Midpoint of segment S1
S1.midpoint
```

#output
Point2D(52,−12)Point2D(52,−12)

Points of linear entities

```
L = Line(x+y-5)
L.points
```

#output

```
(Point2D(0, 5), Point2D(1, 4))
```

```
R=Ray(Point(0,0),Point(0,1))
R.points
```

#output

```
(Point2D(0, 0), Point2D(0, 1))
```

Rotation of linear entities

Q.1 Rotate the line by 30 degrees having two points (0,0) and (0,1). Also, find its equation after applying rotation.

```
L  = Line(Point(0,0),Point(0,1))
# Rotation of line L by 30 degrees
L.rotate(pi/6)
```

#output

$$\text{Line2D(Point2D(0,0),Point2D}(-1/2,3/\sqrt{2}))$$

```
# Equation of line after rotation
L1 = L.rotate(pi/6)
L1.equation()
```

#output

$$-3*\sqrt{x}/2-y/2$$

Q.2 Rotate the segment by 180 degrees having end points (1,0) and (2,-1)

```
S = Segment(Point(1,0),Point(2,-1))
#Rotation of segment S by 180 degrees
S.rotate(pi)
```

#output

$$\text{Segment2D(Point2D}(-1,0),\text{Point2D}(-2,1))$$

Q.3 Rotate the ray by 90 degrees having starting point (0,0) in the direction of (4,4)

```
R = Ray(Point(0,0),Point(4,4))
#Rotation of ray R by 90 degrees
R.rotate(pi/2)
```

#output

$$\text{Ray2D(Point2D}(0,0),\text{Point2D}(-4,4))$$

Q.4 Rotate the ray by 90 degrees clockwise having starting point (0,0) in the direction of (4,4)

```
R = Ray(Point(0,0),Point(4,4))
#Rotation of ray R in clockwise by 90 degrees
R.rotate(-pi/2)
```

#output
Ray2D(Point2D(0,0),Point2D(4,−4))

Q.5 Rotate the ray by 180 degrees clockwise having starting point (0,0) in the direction of (4,4)

```
R = Ray(Point(0,0),Point(4,4))
#Rotation of ray R by 180 degrees
R.rotate(pi)
```

#output
Ray2D(Point2D(0,0),Point2D(−4,−4))

Q.6 Reflect the line 4x+3y = 5 through line x+y = 0 and find the equation of reflected line.

```
L = Line(4*x+3*y-5)
L1 = Line(x+y)
# Reflection of L through L1
Line = L.reflect(L1)
Line.equation()
```

#output
$x+4*y/3+5/3$

Q.7 Reflect the segment having two endpoints (2,3), (4,6) through line 7X+6y=3

```
P = Point(2,3)
Q = Point(4,6)
S = Segment(P,Q)
x,y = symbols('x y')
L = Line(7*x+6*y-3)
# Reflection of S through L
S.reflect(L)
```

#output
Segment2D(Point2D(−236/85,−93/85),Point2D(−514/85,−222/85))

Q.8 Reflect the line segmentt having starting point (0,0) in the direction of (2,4) through line x-2y =3

```
P = Point(0,0)
Q = Point(2,4)
R = Ray(P,Q)
```

```
x,y = symbols('x y')
L = Line(x-2*y-3)
#Reflection ofR through L
R.reflect(L)
```

#output
Ray2D(Point2D(6/5,−12/5),Point2D(28/5,−16/5))

Q.9

#If the line with A[2 1], B[4 -1] is transformed by the transformation matrix, [T] = $1\begin{pmatrix} 2 & \\ 2 & 1 \end{pmatrix}$ *, then find the* equation of transformed line

```
A = Point(2,1)
B = Point(4,-1)
A1 = A.transform(Matrix([[1,2,0],[2,1,0],[0,0,1]]))
B1 = B.transform(Matrix([[1,2,0],[2,1,0],[0,0,1]]))
L=Line(A1,B1)
L.equation()
```

#output
$-2x-2y+18$

Q.10

#If the line segment with endpoints A[1 1], B[-4 -1] is transformed by the transformation matrix, [T] = $\begin{pmatrix} 1 & -2 \\ -2 & 1 \end{pmatrix}$ *then find the points of transformed line*

```
A = Point(1,1)
B = Point(-4,-1)
A1 = A.transform(Matrix([[1,-2,0],[-2,1,0],[0,0,1]]))
B1 = B.transform(Matrix([[1,-2,0],[-2,1,0],[0,0,1]]))
L=Line(A1,B1)
L.points
```

#output
```
(Point2D(-1, -1), Point2D(-2, 7))
```

/**

Name:Asdfjk Asdfjk

Class:S.Y.B.Sc Computer Science Asdfjk

Practical No:5

Practical Name: **Application to computaional Geometry**

**/

Q.1 Rotate the line passing through points A[1 1] and B[5 5] about origin through an angle 90 degree.

```
from sympy import *

L = Line((1,1),(5,5))
L.rotate(pi/2)
```

#output
Line2D(Point2D(−1,1),Point2D(−5,5))

Q.2 If the line segment joining the points A[2 5], B[4 -13] is transformed to the line segment A*B* by the transformation matrix, [T]= $\begin{bmatrix} 2 & 3 \\ 4 & 1 \end{bmatrix}$, then find the midpoint of A*B*.

```
A = Point(2,5)
B = Point(4,-13)
A1 = A.transform(Matrix([[2,3,0],[4,1,0],[0,0,1]]))
B1 = B.transform(Matrix([[2,3,0],[4,1,0],[0,0,1]]))
L=Segment(A1,B1)
L.midpoint
```

#output
Point2D(−10,5)

Q.3 Reflect the line segment joining the points A[5 3] and B[1 4] through the line y = x+1

```
x,y = symbols('x y')
A = Point(5,3)
B = Point(1,4)
S = Segment(A,B)
S.reflect(Line(x-y+1))
```

#output
Segment2D(Point2D(2,6),Point2D(3,2))

Q.4 Suppose that the line segment between the points A[1 4] and B[3 6] is transformed to the line segment A*B* using the transformation matrix [T]= $\begin{bmatrix} 2 & -1 \\ 1 & 3 \end{bmatrix}$. Find slope of the transformed line segment A*B*.

```
A = Point(1,4)
B = Point(3,6)
A1 = A.transform(Matrix([[2, -1, 0],[1,3,0],[0,0,1]]))
B1 = B.transform(Matrix([[2,-1,0],[1,3,0],[0,0,1]]))
L = Segment(A1,B1)
L.slope
```

#output
2/3

Q.5 if the two lines 2x-y=5 and x+3y=-1 are transformed using the transformation matrix [T]= $\begin{bmatrix} -2 & 3 \\ 1 & 1 \end{bmatrix}$, then the point of intersection of the transformed lines.

```
x,y = symbols('x y')
l1 = Line(2*x-y-5)
l2 = Line(x+3*y+1)
p = l1.intersection(l2)
p=p[0]
p.transform(Matrix([[-2,3,0],[1,1,0],[0,0,1]]))
```

#output
Point2D(−5,5)

Q.6 If we apply shearing on the line 2x+y=3 in x and y directions by 2 and -3 uints resp. then find the equation of the resulting line

```
x,y = symbols('x y')
l=Line(2*x+y-3)
points=l.points
p=points[0]
q=points[1]
p1=p.transform(Matrix([[1,-3,0],[2,1,0],[0,0,1]]))
q1=q.transform(Matrix([[1,-3,0],[2,1,0],[0,0,1]]))
l1 = Line(p1,q1)
l1.equation()
```

#output
$5x-3y-21$

Q.7 If a 2x2 transformation matrix [T]= 1 3 -2 2 is used to transform a line L,then the equation of transformed line is y* = x*+4. Find the equation of original line.

```
x,y= symbols('x y')
l = Line(x-y+4)
points=l.points
p=points[0]
```

```
q=points[1]
M=Matrix([[1,3,0],[-2,2,0],[0,0,1]])
N=M.inv()
p1=p.transform(N)
q1=q.transform(N)
l1=Line(p1,q1)
l1.equation()
```

#output
$x/4+y/2-1/2$

Polygon

Q.8 Drown a polygon with vertices (0,0),(1,0),(2,2),(1,4) and find its area and perimeter.

```
A = Point(0,0)
B = Point(1,0)
C = Point(2,2)
D = Point(1,4)
P=Polygon(A,B,C,D)
P.area
```

#output
4

```
P.perimeter
```

#output
1+sqrt(17)+2*sqrt(5)

Q.9 Drown a regular polygon with 8 sides and radis 5 centered at origin and find the area and perimeter

```
P=Polygon((0,0),5,n=8)
P.area
```

#output
$400-200*\sqrt{(2)}/(-4+4\sqrt{2})$

```
P.perimeter
```

#output
$40*\sqrt{(2-(\sqrt{2}))}$

Q.10 Drown a regular polygon with 6 sides and readis 1 centered at (1,2) and find its area and perimeter.

```
P = Polygon((1,2),1,n=6)
P.area
```

#output
$3*\sqrt{3}/2$

P.perimeter

#output
6

/**

Name:Asdfjk Asdfjk

Class:S.Y.B.Sc Computer Science Asdfjk

Practical No:6

Practical Name: **Application to computaional Geometry**

**/

Q.1 Drown a regular polygon with 7 sides and radius 1.5 centered at (2,2) and reflect it through line x-y=5

```
from sympy import *

x,y = symbols('x, y')
P=Polygon((2,2),1.5,n=7)
T.reflect(Line(x-y-5))
```

#output
Triangle(Point2D(5,−4),Point2D(4,−3),Point2D(8,−6))

Q.2 Drown a polygon with vertices (0,0),(2,0),(2,3),(1,6) and rotate by 180 degrees and find internal angle at each vertex.

```
A  =Point(0,0)
B  =Point(2,0)
C  =Point(2,3)
D =Point(1,6)
P=Polygon(A,B,C,D)
P.rotate(pi)
```

#output
Polygon(Point2D(0,0),Point2D(−2,0),Point2D(−2,−3),Point2D(−1,−6))

Q.3 Reflect the pol ABC through the line y=3, where A[1 0],B[2 -1], C[-1 3].

```
x,y = symbols('x y')
A =Point(1,0)
B =Point(2,-1)
C =Point(-1,3)
T=Triangle(A,B,C)
P=Point(0,3)
Q=Point(1,3)
L=Line(P,Q)
T.reflect(L)
```

#output
Triangle(Point2D(1,6),Point2D(2,7),Point2D(−1,3))

Q.4 Rotate the triangle ABC by 90 degree, where A[1 2], B[2 -2], C[-1 2] .

```
x,y = symbols('x y')
A =Point(1,2)
B =Point(2,-2)
C =Point(-1,2)
T=Triangle(A,B,C)
T.rotate(pi/2)
```

#output
Triangle(Point2D(−2,1),Point2D(2,2),Point2D(−2,−1))

Q.5 Find the area and perimeter of the triangle ABC, where A[0 0], B[5 0], C[3 3].

```
x,y = symbols('x y')
A =Point(0,0)
B =Point(5,0)
C =Point(3,3)
T=Triangle(A,B,C)
T.area
```

15/2

```
T.perimeter
```

#output
Sqrt(13)+3*√2+5

Q.6 Find the angle at each vetices of the triangle ABC, where A[0 0], B[2 2], C[0 2].

```
x,y = symbols('x y')
A =Point(0,0)
B =Point(2,2)
C =Point(0,2)
T=Triangle(A,B,C)
T.angles[A]
```

#output
$\pi/4$

```
T.angles[B]
```

#output
$\pi/4$

```
T.angles[C]
```

#output
$\pi/2$

/**
Asdfjk Asdfjk
Class:S.Y.B.Sc Computer Science Asdfjk
Practical No:7
Practical Name: Study of graphical aspects of Two dimensional transformationmatrix using matplotlib
**/

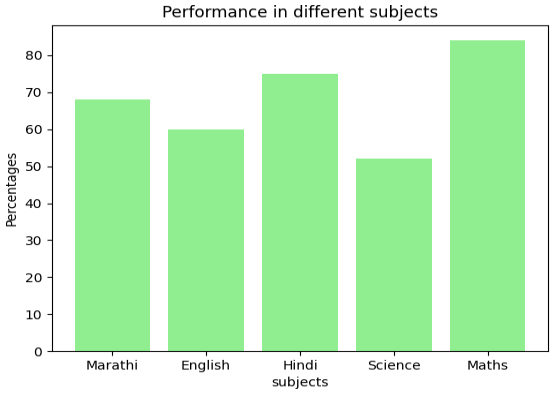
Q.1 Using python draw a bar graph in light green colour to represent the data below

Subject	Marathi	English	Hindi	Science	Maths
Percentages	68	60	75	52	84

Input:

```
subjects=['Marathi','English','Hindi','Science','Maths']

Percentages=[68,60,75,52,84]

plt.bar(subjects,Percentages,color='lightgreen')

plt.xlabel('subjects')

plt.ylabel('Percentages')

plt.title('Performance in different subjects')

plt.show()
```

Output:



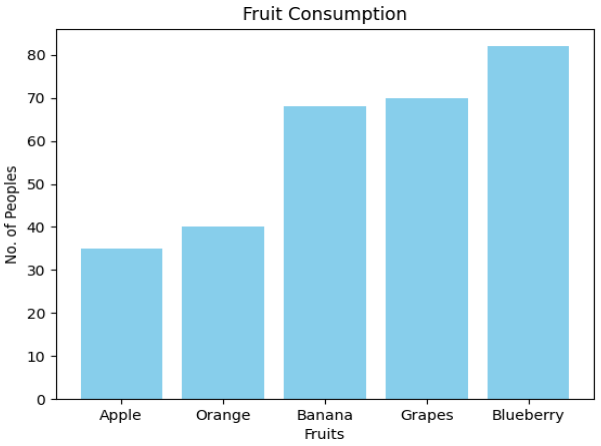
Q.2 Using python draw a bar graph in skyblue colour to represent the data below

Fruits	Apple	Orange	Banana	Grapes	Blueberry
No. of Peoples	35	40	68	70	82

Input:

```
fruits=['Apple','Orange','Banana','Grapes','Blueberry']
no_of_peoples=[35,40,68,70,82]
plt.bar(fruits,no_of_peoples,color='skyblue')
plt.xlabel('Fruits')
plt.ylabel('No. of Peoples')
plt.title('Fruit Consumption')
plt.show()
```

Output:



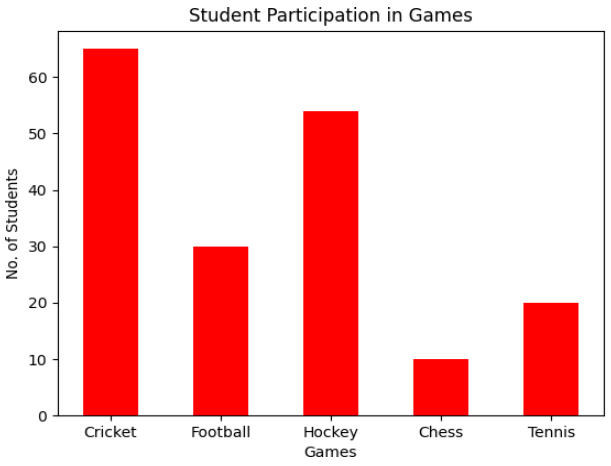
Q.3 Using python draw a bar graph in red colour and take width of 0.5 inches to represent the data below

Games	Cricket	Football	Hockey	Chess	Tennis
No. of Students	65	30	54	10	20

Input:

```
games=['Cricket','Football','Hockey','Chess','Tennis']
no_of_students=[65,30,54,10,20]
plt.bar(games,no_of_students,color='red',width=0.5)
plt.xlabel('Games')
plt.ylabel('No. of Students')
plt.title('Student Participation in Games')
plt.show()
```

Output:



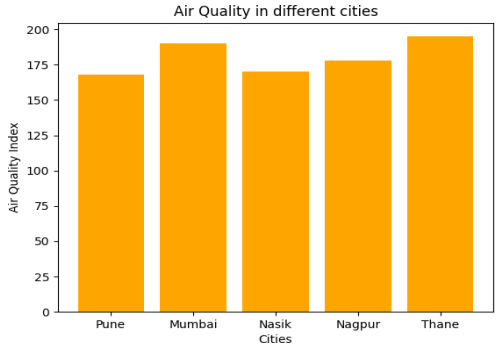
Q.4 Using python draw a horizontal graph in orange colour to represent the data below

City	Pune	Mumbai	Nashik	Nagpur	Thane
Air Quality Index	168	190	170	178	195

Input:

```
Cities=['Pune','Mumbai','Nasik','Nagpur','Thane']

air_quality=[168,190,170,178,195]

plt.bar(cities,air_quality,color='orange')

plt.ylabel('Air Quality Index')

plt.xlabel('Cities')

plt.title('Air Quality in different cities')

plt.show()
```

Output:



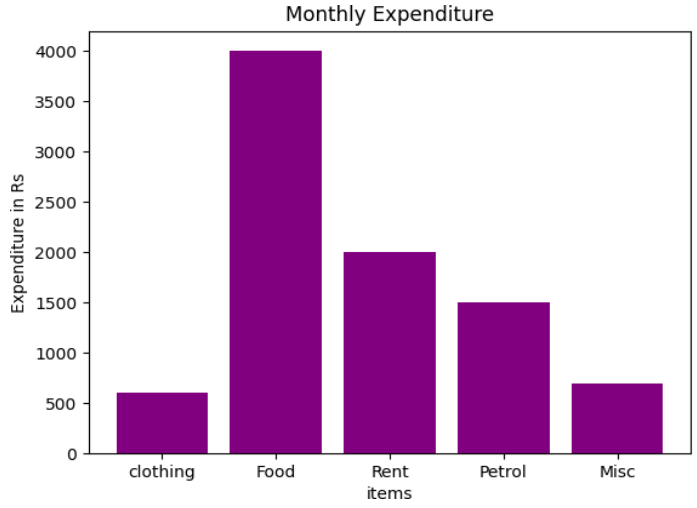
Q.5 Using python draw a bar graph in purple colour to represent the data below

Item	Clothing	Food	Rent	Petrol	Misc
Expenditure in Rs	600	4000	2000	1500	700

Input:

```
items=['clothing','Food','Rent','Petrol','Misc']

expenditure=[600,4000,2000,1500,700]

plt.bar(items,expenditure,color='purple')

plt.xlabel('items')

plt.ylabel('Expenditure in Rs')

plt.title('Monthly Expenditure')

plt.show()
```

Output:



/**
Name:Asdfjk Asdfjk
Class:S.Y.B.Sc Computer Science Asdfjk
Practical No:8
Practical Name: **Study of Operational Research in Python**
**/

```
from pulp import *
```

Q.1 Solve the following LLP:

Max Z = 150x+75y

subject to,

$4x+6y \leq 24$

$5x+3y \leq 15$

$x \geq 0, y \geq 0$

```
model  =  LpProblem(name="small-problem",  sense=LpMaximize)
x = LpVariable(name="x",lowBound=0)
y = LpVariable(name="y",lowBound=0)
model +=(4*x+6*y<=24)
model +=(5*x+3*y<=15)
model += 150*x+75*y

model
```

#output

```
small-problem:
MAXIMIZE
150*x + 75*y + 0
SUBJECT TO
_C1: 4 x + 6 y <= 24

_C2: 5 x + 3 y <= 15

VARIABLES
x Continuous
y Continuous
```

```
model.solve()
```

#output

```
1
```

```
model.objective.value()
```

#output

```
450.0
```

x.value()

#output

```
3.0
```

y.value()

#output

```
0.0
```

Q.2 Solve the following LPP

Min Z = 3.5x+2y

subject to,

x+y ≥5

x≥4

y≤2

x≥0, y≥0

```
model = LpProblem(sense=LpMinimize)
x = LpVariable(name="x",lowBound=0)
y = LpVariable(name="y",lowBound=0)
model += (x+y>=5)
model += (x>=4)
model += (y<=2)
model += 3.5*x+2*y
model
```

#output

```
NoName:
MINIMIZE
3.5*x + 2*y + 0.0
SUBJECT TO
_C1: x + y >= 5

_C2: x >= 4

_C3: y <= 2

VARIABLES
```

```
x Continuous
y Continuous
```

model.solve()

#output
```
1
```

model.objective.value()

#output
```
16.0
```

x.value()

#output
```
4.0
```
y.value()

#output
```
1.0
```

Q.3 Solve the following LLP:

Max z = 3x+5y+4z

Subject to,

$2x+3y \leq 8$

$2y+5z \leq 10$

$3x+2y+4z \leq 15$

$x \geq 0, y \geq 0, z \geq 0$

```
model = LpProblem(sense=LpMaximize)
x = LpVariable(name="x",lowBound=0)
y = LpVariable(name="y",lowBound=0)
z = LpVariable(name="z",lowBound=0)
model += (2*x+3*y<=8)
model += (2*y+5*z<=10)
model += (3*x+2*y+4*z<=15)
model += 3*x+5*y+4*z
model
```

#output
```
NoName:
MAXIMIZE
3*x + 5*y + 4*z + 0
```

```
SUBJECT TO
_C1: 2 x + 3 y <= 8

_C2: 2 y + 5 z <= 10

_C3: 3 x + 2 y + 4 z <= 15

VARIABLES
x Continuous
y Continuous
z Continuous
```

model.solve()

#output
```
1
```

model.objective.value()

#output
```
18.658536500000004
```

x.value()

#output
```
2.1707317
```

y.value()

#output
```
1.2195122
```

z.value()

#output
```
1.5121951
```

/***
Name:Asdfjk Asdfjk
Class: S.Y.Bsc.Computer Science Asdfjk
Practical No:09
Practical Name: Study of Operational Research in Python.
***/

```
from pulp import *
```

Q.1 Solve the following LLP:

Max Z = x+2y+z

subject to,

$x+\frac{1}{2}y+\frac{1}{2}z \leq 1$

$\frac{3}{2}x+2y+z \geq 8$

$x \geq 0, y \geq 0, z \geq 0$

```
model = LpProblem(sense=LpMaximize)
x = LpVariable(name="x",lowBound=0)
y = LpVariable(name="y",lowBound=0)
z = LpVariable(name="z",lowBound=0)
model +=(x+0.5*y+0.5*z<=1)
model +=(1.5*x+2*y+z>=8)
model += x+2*y+z
model
#output
NoName:
MAXIMIZE
1*x + 2*y + 1*z + 0
SUBJECT TO
C1: x + 0.5 y + 0.5 z <= 1
C2: 1.5 x + 2 y + z >= 8

VARIABLES
x Continuous
y Continuous
z Continuous
```

```
model.solve()
#output
-1
```

solve() returns the integer of the sloution is -1,i.e. solution is invesible.

Q.2 Solve the following LPP

Min Z = x+y

subject to,

$x \geq 6$

$y \geq 6$

$x+y \leq 11$

$x \geq 0, y \geq 0$

```
model = LpProblem(sense=LpMinimize)
x = LpVariable(name="x",lowBound=0)
y = LpVariable(name="y",lowBound=0)
model += (x>=6)
model += (y>=6)
model += (x+y<=11)
model += x+y
model
```

```
#output
NoName:
MINIMIZE
1*x + 1*y + 0
SUBJECT TO
_C1: x >= 6

_C2: y >= 6

_C3: x + y <= 11

_C4: x >= 6

_C5: y >= 6

_C6: x + y <= 11

VARIABLES
x Continuous
y Continuous
```

```
model.solve()
```

```
#output
-1
# solve() returns the integer status of the solution, which is -1, i.e. solution is infeasible.
```

Q.3 Solve the Following LPP:

Max z = x+y

Subject to,

x-y≥ 1

x+y≥2

x≥0, y≥0

```
model = LpProblem(sense=LpMaximize)
x = LpVariable(name="x",lowBound=0)
y = LpVariable(name="y",lowBound=0)
model += (x-y>=1)
model += (x+y>=2)
model += x+y
model
```

#output

```
NoName:
MAXIMIZE
1*x + 1*y + 0
SUBJECT TO
_C1: x - y >= 1

_C2: x + y >= 2

VARIABLES
x Continuous
y Continuous
```

```
model.solve()
```

#output

-2

 # solve() returns the integer status of the solution, which is -2, i.e. solution is unbounded

Q.4 Solve the following LPP:

Max z = 4x+y+3z+5w

Subject to,

4x+6y-5z-4x≥-20

-3x-2y+4z+w≤10

-8x-3y+3z+2w≤20

x≥ 0, y≥0, z≥0, w≥0

```
model = LpProblem(sense=LpMaximize)
x = LpVariable(name="x",lowBound=0)
y = LpVariable(name="y",lowBound=0)
z = LpVariable(name="z",lowBound=0)
w =
LpVariable(name="w",lowBound=0)
model += (4*x+6*y-5*z-4*x>=-20)
model += (-3*x-2*y+4*z+w<=10)
model += (-8*x-3*y+3*z+2*w<=20)
model += 4*x+y+3*z+5*w
model output

#output
NoName:
MAXIMIZE
5*w + 4*x + 1*y + 3*z + 0
SUBJECT TO
_C1: 0 x + 6 y - 5 z >= -20

_C2: w - 3 x - 2 y + 4 z <= 10

_C3: 2 w - 8 x - 3 y + 3 z <= 20

VARIABLES
w Continuous
x  Continuous
y Continuous
z Continuous

model.solve()

#output
-2
# solve() returns the integer status of the solution, which is -2, i.e. solution is unbounded
```

/**
Asdfjk Asdfjk
Class:S.Y.B.Sc Computer Science Asdfjk
Practical No:10
Practical Name: **Study of Operational Research in Python**
**/

```
from pulp import *
```

Q.1 Write a Python program to display the following LPP by using pulp modul e and simplex method. Find it's optimal solution if exist.

Max : Z = 3x + 2y + 5z
Subject to,
x + 2y + z ≤ 430
3x + 2y ≤ 460
x + 4y ≤ 120
x, y, z ≥0

```
model = LpProblem(sense=LpMaximize)
x = LpVariable(name="x",lowBound=0)
y = LpVariable(name="y",lowBound=0)
z = LpVariable(name="z",lowBound=0)
model += (x+2*y+z<=430)
model += (3*x+2*y<=460)
model += (x+4*y<=120)
model += 3*x+2*y+5*z
model
```

```
#output
NoName:
MAXIMIZE
3*x + 2*y + 5*z + 0
SUBJECT TO
_C1: x + 2 y + z <= 430

_C2: 3 x + 2 y <= 460

_C3: x + 4 y <= 120

VARIABLES
x Continuous
y Continuous
z Continuous
```

```
model.solve()

#output
```

```
1
```

```
print("The value of objective function is",model.objective.value())
print("The value of x is", x.value())
print("The value of y is", y.value())
print("The value of z is", z.value())

#output
The value of objective function is 2150.0
The value of x is 0.0
The value of y is 0.0
The value of z is 430.0
```

Q.2 Write a Python program to display the following LPP by using pulp module and simplex method. Find it's optimal solution if exist.

Max : Z = -2x -4y - z

Subject to,

x + 2y- z ≤ 5

2x –y + 2z ≥ 2

-x + 2y + 2z ≥ 1

x, y, z ≥0

```
model = LpProblem(sense=LpMaximize)
x = LpVariable(name="x",lowBound=0)
y = LpVariable(name="y",lowBound=0)
z = LpVariable(name="z",lowBound=0)
model += (x+2*y+z<=5)
model += (2*x-y+2*z>=2)
model += (-x+2*y+2*z>=1)
model += -2*x-4*y-z
model

#output
NoName:
MAXIMIZE
-2*x + -4*y + -1*z + 0
SUBJECT TO
_C1: x + 2 y + z <= 5

_C2: 2 x - y + 2 z >= 2

_C3: - x + 2 y + 2 z >= 1

VARIABLES
x Continuous
y Continuous
z Continuous
```

```
model.solve()

#output
1

print("The value of objective function is",model.objective.value())
print("The value of x is", x.value())
print("The value of y is", y.value())
print("The value of z is", z.value())

#output
The value of objective function is -1.0
The value of x is 0.0
The value of y is 0.0
The value of z is 1.0
```

Q.3 Write a Python program to display the following LPP by using pulp module and simplex method. Find it's optimal solution if exist.

Min: Z = 3x + 2y– 4z + w

Subject to,

6x + 3y- 2z + w ≥ 22

-2x + 3y + 5z + 3w ≤ 15

-8x + 2y -6z + 5w ≤ 18

x, y, z, w ≥0

```
model = LpProblem(sense=LpMaximize)
x = LpVariable(name="x",lowBound=0)
y = LpVariable(name="y",lowBound=0)
z = LpVariable(name="z",lowBound=0)
w = LpVariable(name="w",lowBound=0)
model += (6*x+3*y-2*z-w>=22)
model += (-2*x+3*y+5*z+3*w<=15)
model  += (-8*x+2*y-6*z+5*w<=18)
model += 3*x+2*y-4*z+w
model

#output
NoName:
MAXIMIZE
1*w + 3*x + 2*y + -4*z + 0
SUBJECT TO
_C1: - w + 6 x + 3 y - 2 z >= 22

_C2: 3 w - 2 x + 3 y + 5 z <= 15

_C3: 5 w - 8 x + 2 y - 6 z <= 18
```

```
VARIABLES
x Continuous
y Continuous
z Continuous
w Continuous
```

model.solve()

#output

```
-2
```

 # solve() returns the integer status of the solution, which is -2, i.e. solution is unbounded