**Sahakar Maharshi Bhausaheb Santuji Thorat College Sangamner**

**Remark**

**Demonstrator’s Signature**

**Date:-**

**/**

**/20**

**DEPARTMENT OF COMPUTER SCIENCE**

**Sub : Mathematics**

**Name:-\_Gorde Yash Somnath Roll.No:-\_21 Date:- Title of the expt:- Slip no 14 Page.no:- Class:- BCS**

**Q1 ) Attempt any Two of the following**

**A ) Write a python program to plot 2D graph of the function f(x)=x2 and g(x)=x3 in [-1,1]**

**-**

import numpy as np

import matplotlib.pyplot as plt def f(x):

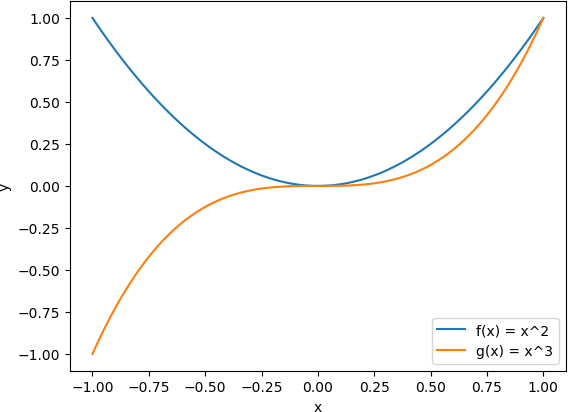
return x\*\*2

def g(x): return x\*\*3

x = np.linspace(-1, 1, 100) plt.plot(x, f(x), label='f(x) = x^2')

plt.plot(x, g(x), label='g(x) = x^3') plt.xlabel('x')

plt.ylabel('y') plt.legend() plt.show()



**B ) Write a python program to plot 3D graph of the function f(x)=e-x2 in [-5,5] with green dashed points line with upward pointing trangle**

**-**

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D def f(x, y):

return np.exp(-x\*\*2)

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

y = np.linspace(-5, 5, 100) X, Y = np.meshgrid(x, y) Z = f(X, Y)

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

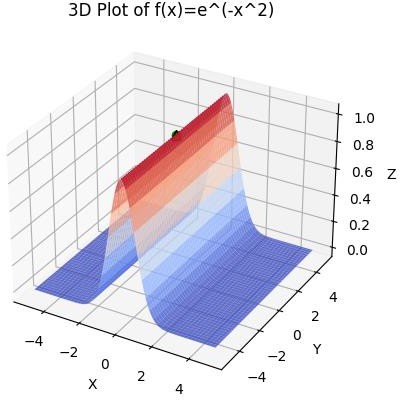
surf = ax.plot\_surface(X, Y, Z, cmap='coolwarm', alpha=0.8)

ax.scatter([0], [0], [1], s=50, c='green', marker='^', edgecolors='black', linewidths=1, alpha=1)

ax.plot([0], [0], [1], c='green', marker='o', linestyle='--') ax.set\_xlabel('X')

ax.set\_ylabel('Y') ax.set\_zlabel('Z')

ax.set\_title('3D Plot of f(x)=e^(-x^2)') plt.show()



**C ) Write a python program to generate 3D plot of the function z=sin(x)+cos(y) in**

**-5 <x,y<5**

**-**

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D def f(x, y):

return np.sin(x) + np.cos(y)

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

y = np.linspace(-5, 5, 100) X, Y = np.meshgrid(x, y) Z = f(X, Y)

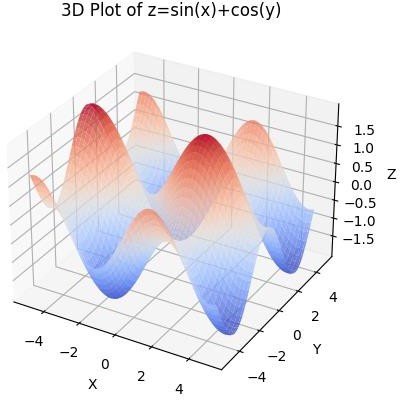
fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

surf = ax.plot\_surface(X, Y, Z, cmap='coolwarm', alpha=0.8)

ax.set\_xlabel('X') ax.set\_ylabel('Y') ax.set\_zlabel('Z')

ax.set\_title('3D Plot of z=sin(x)+cos(y)') plt.show()



**Q2 )Attempt any TWO of the following**

**A ) Write a python program to reflect the line segment joining the points A[5,3] and B[1,4] through the line y=x+1**

**-**

import numpy as np import matplotlib.pyplot as plt

A = np.array([5, 3])

B = np.array([1, 4])

x = np.linspace(-10, 10, 100) y = x + 1

M = (A + B) / 2 v = A - M

n = np.array([-1, 1]) p = np.dot(v, n) \* n Ar = A - 2 \* p

Br = B - 2 \* p

plt.plot([A[0], B[0]], [A[1], B[1]], 'bo-', label='AB')

plt.plot([Ar[0], Br[0]], [Ar[1], Br[1]], 'go-', label='A\'B\'') plt.plot(x, y, 'r--', label='Reflection Line')

plt.xlim(-10, 10)

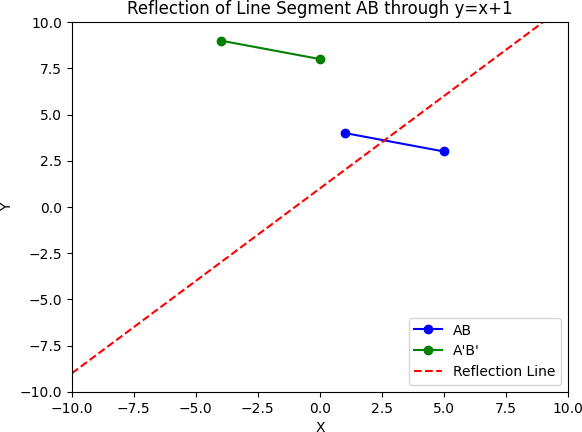
plt.ylim(-10, 10)

plt.xlabel('X')

plt.ylabel('Y')

plt.title('Reflection of Line Segment AB through y=x+1') plt.legend()

plt.show()



B**) Write a python program to draw a polygon with vertices (0,0),(2,0),(2,3) and (1,6) and rotate it by 180⸰**

**-**

import numpy as np

import matplotlib.pyplot as plt

vertices = np.array([[0, 0], [2, 0], [2, 3], [1, 6], [0, 0]])

plt.plot(vertices[:, 0], vertices[:, 1], 'b-', label='Original') theta = np.pi # 180 degrees in radians

R = np.array([[np.cos(theta), -np.sin(theta)],

[np.sin(theta), np.cos(theta)]]) rotated\_vertices = vertices @ R

plt.plot(rotated\_vertices[:, 0], rotated\_vertices[:, 1], 'g-', label='Rotated')

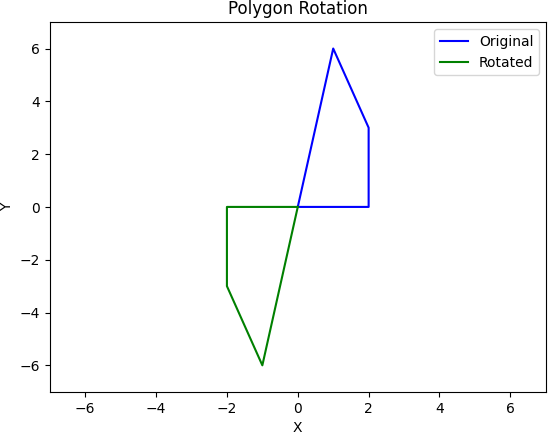
plt.xlim(-7, 7)

plt.ylim(-7, 7)

plt.xlabel('X')

plt.ylabel('Y') plt.title('Polygon Rotation') plt.legend()

plt.show()



**C ) Write a python program to find the area and perimeter of the tringle ABC where A[0,0],B[5,0],C[3,3]**

**-**

import math

A = [0, 0]

B = [5, 0]

C = [3, 3]

a = math.dist(B, C) # Length of side a (opposite vertex A)

b = math.dist(A, C) # Length of side b (opposite vertex B)

c = math.dist(A, B) # Length of side c (opposite vertex C) perimeter = a + b + c

s = perimeter / 2

area = math.sqrt(s \* (s - a) \* (s - b) \* (s - c))

print(f"The area of the triangle ABC is {area:.2f} square units") print(f"The perimeter of the triangle ABC is {perimeter:.2f} units")

output :

The area of the triangle ABC is 7.50 square units The perimeter of the triangle ABC is 12.85 units

**Q3 ) Attempt the following**

**A ) Attempt any ONE of the following**

**I ) Write a python program to solve the following LPP : Max Z=150x+75y**

**Subject to 4x+6y≤24**

**5x+3y≤15 X,y≥0**

**-**

import numpy as np

from scipy.optimize import linprog c = np.array([-150, -75])

A = np.array([[4, 6], [5, 3]])

b = np.array([24, 15]) x\_bounds = (0, None) y\_bounds = (0, None)

res = linprog(c, A\_ub=A, b\_ub=b, bounds=[x\_bounds, y\_bounds])

print("Status: ", res.message)

print("Optimal values:\n x =", res.x[0], "\n y =", res.x[1]) print("Optimal value of the objective function: Z =", -res.fun)

output :

Status: Optimization terminated successfully. (HiGHS Status 7: Optimal) Optimal values:

x = 3.0

y = 0.0

Optimal value of the objective function: Z = 450.0

**II) Write a python program to solve the following LPP : Min Z=x+y**

**Subject to x≥6**

**y≥6 x+y≤11 x,y≥0**

**-**

from scipy.optimize import linprog obj = [1, 1]

lhs = [[-1, 0], [0, -1], [1, 1]]

rhs = [-6, -6, 11]

bounds = [(0, None), (0, None)]

result = linprog(c=obj, A\_ub=lhs, b\_ub=rhs, bounds=bounds, method='simplex')

print("Status:", result.message) print("Optimal Solution:", result.fun) print("Optimal Decision Variables:", result.x)

output :

Optimal Solution: 12.0

Optimal Decision Variables: [6. 6.]

**B ) Attempt any ONE of the following**

**I ) Apply each of the following transformation on the point P[2,-3]**

**A ) Reflection through X-axis**

**B ) Scaling in X-cordinate by factor 2**

**C ) Scaling in Y-coordinate by factor 1.5**

**D ) Reflection through the line y=x**

**-**

import numpy as np P = np.array([2, -3])

A = np.array([[1, 0], [0, -1]]) P1 = A.dot(P)

print("Reflection through X-axis:", P1)

B = np.array([[2, 0], [0, 1]]) P2 = B.dot(P)

print("Scaling in X-coordinate by factor 2:", P2)

C = np.array([[1, 0], [0, 1.5]]) P3 = C.dot(P)

print("Scaling in Y-coordinate by factor 1.5:", P3)

D = np.array([[0, 1], [1, 0]]) P4 = D.dot(P)

print("Reflection through the line y=x:", P4)

output :

Reflection through X-axis: [2 3]

Scaling in X-coordinate by factor 2: [ 4 -3]

Scaling in Y-coordinate by factor 1.5: [ 2. -4.5] Reflection through the line y=x: [-3 2]

**II ) Apply each of the following transformation on the points P[3,-1]**

**A ) Shering in Y-direction by 2 units**

**B ) Scaling in X and Y direction by ½ and 3 units respectively**

**C ) Shering in both X and Y direction by -2 and 4 units respectively**

**D ) Rotation about origin by an angle 30 degree**

**-**

import numpy as np import math

P = np.array([3, -1])

A = np.array([[1, 2], [0, 1]])

P1 = A.dot(P)

print("Sherring in Y-direction by 2 units:", P1)

B = np.array([[0.5, 0], [0, 3]]) P2 = B.dot(P)

print("Scaling in X and Y direction by ½ and 3 units respectively:", P2)

C = np.array([[1, -2], [-4, 1]]) P3 = C.dot(P)

print("Sherring in both X and Y direction by -2 and 4 units respectively:", P3)

angle = 30

theta = math.radians(angle)

D = np.array([[math.cos(theta), -math.sin(theta)], [math.sin(theta), math.cos(theta)]]) P4 = D.dot(P)

print("Rotation about origin by an angle 30 degree:", P4)

output :

Sherring in Y-direction by 2 units: [ 1 -1]

Scaling in X and Y direction by ½ and 3 units respectively: [ 1.5 -3. ] Sherring in both X and Y direction by -2 and 4 units respectively: [ 5 -13] Rotation about origin by an angle 30 degree: [3.09807621 0.6339746 ]