**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 12 Page.no:- Class:- BCS**

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

**A ) Write a python program to plot the graph of y=x3+10x-5 for x E[-10,10] in red color.**

**-**

import matplotlib.pyplot as plt import numpy as np

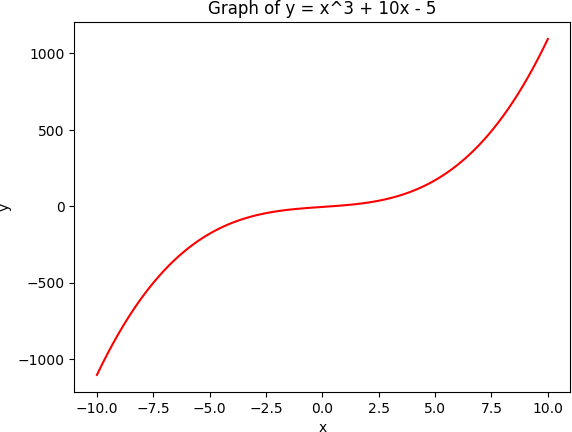
def f(x):

return x\*\*3 + 10\*x - 5

x\_values = np.linspace(-10, 10, 1000) y\_values = f(x\_values) plt.plot(x\_values, y\_values, color='red') plt.xlabel('x')

plt.ylabel('y')

plt.title('Graph of y = x^3 + 10x - 5') plt.show()



# B ) Write a python program in 3D to rotate the point (1,0,0) through XZ –plane in clockwise direction (rotation through Y-axis by an angle of 90⸰)

**-**

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D point = np.array([[1], [0], [0]])

theta = np.radians(90) cos\_theta = np.cos(theta) sin\_theta = np.sin(theta)

rotation\_matrix = np.array([[cos\_theta, 0, sin\_theta],

[0, 1, 0],

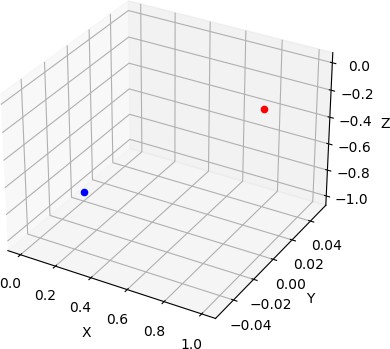
[-sin\_theta, 0, cos\_theta]])

rotated\_point = rotation\_matrix @ point fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d') ax.scatter(point[0], point[1], point[2], c='r', marker='o')

ax.scatter(rotated\_point[0], rotated\_point[1], rotated\_point[2], c='b', marker='o') ax.set\_xlabel('X')

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



# C ) Using python plot the graph of function f(x)=x2 on the interval [-2,2]

**-**

import matplotlib.pyplot as plt import numpy as np

def f(x):

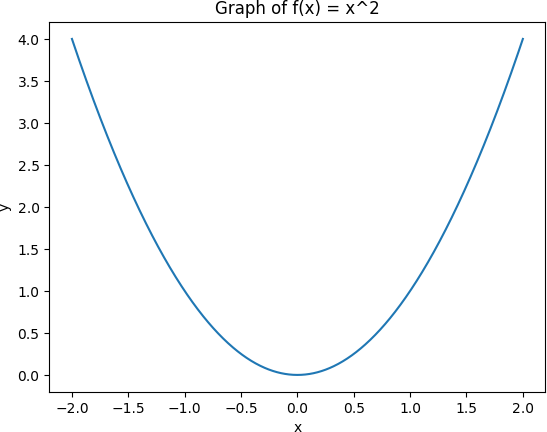
return x\*\*2

x\_values = np.linspace(-2, 2, 1000) y\_values = f(x\_values)

plt.plot(x\_values, y\_values) plt.xlabel('x')

plt.ylabel('y')

plt.title('Graph of f(x) = x^2') plt.show()



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

**A ) Write a python program to rotate the segment by 180⸰ having end points (1,0) and (2,- 1)**

**-**

import numpy as np

import matplotlib.pyplot as plt point1 = np.array([1, 0]) point2 = np.array([2, -1])

theta = np.radians(180) cos\_theta = np.cos(theta) sin\_theta = np.sin(theta)

rotation\_matrix = np.array([[cos\_theta, -sin\_theta],

[sin\_theta, cos\_theta]])

rotated\_point1 = rotation\_matrix @ point1 rotated\_point2 = rotation\_matrix @ point2

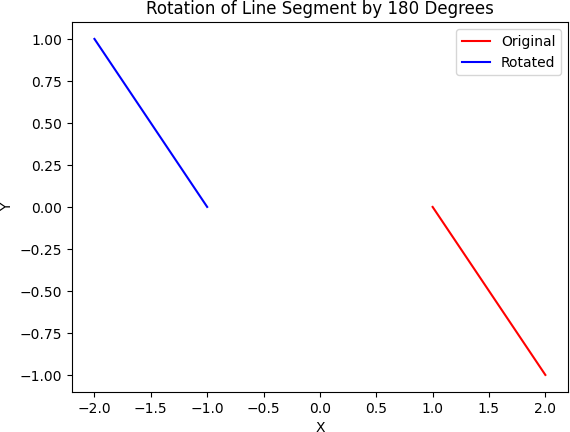
plt.plot([point1[0], point2[0]], [point1[1], point2[1]], 'r', label='Original') plt.plot([rotated\_point1[0], rotated\_point2[0]], [rotated\_point1[1], rotated\_point2[1]], 'b', label='Rotated')

plt.xlabel('X')

plt.ylabel('Y')

plt.title('Rotation of Line Segment by 180 Degrees') plt.legend()

plt.show()



**B ) Write a python program to draw a polygon with 8 sides having radius 5 centered at origin and find its area and perimeter**

**-**

import matplotlib.pyplot as plt import numpy as np

num\_sides = 8

radius = 5

angles = np.linspace(0, 2 \* np.pi, num\_sides + 1)[:-1] x\_coords = radius \* np.cos(angles)

y\_coords = radius \* np.sin(angles)

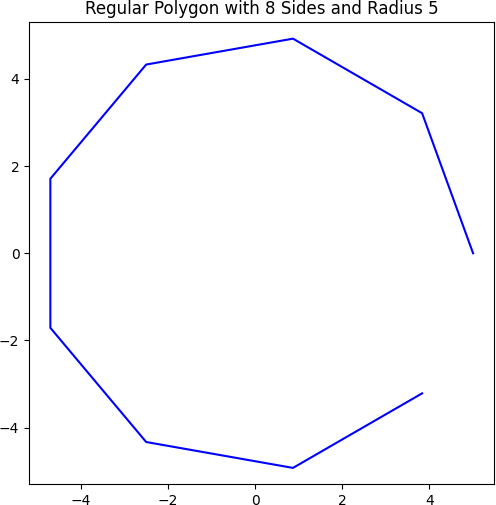
plt.figure(figsize=(6, 6)) plt.plot(x\_coords, y\_coords, 'b') plt.axis('equal')

plt.title('Regular Polygon with 8 Sides and Radius 5')

side\_length = 2 \* radius \* np.sin(np.pi / num\_sides) perimeter = num\_sides \* side\_length

area = (num\_sides \* radius \*\* 2 \* np.sin(2 \* np.pi / num\_sides)) / 2

print("Area:", area) print("Perimeter:", perimeter) plt.show()



**C ) Write a python program to find the area and perimeter of the ∆XYZ , where X(1,2), Y(2,-2),Z(-1,2)**

**-**

import math X = (1, 2)

Y = (2, -2)

Z = (-1, 2)

XY = math.sqrt((Y[0] - X[0]) \*\* 2 + (Y[1] - X[1]) \*\* 2)

XZ = math.sqrt((Z[0] - X[0]) \*\* 2 + (Z[1] - X[1]) \*\* 2)

YZ = math.sqrt((Z[0] - Y[0]) \*\* 2 + (Z[1] - Y[1]) \*\* 2)

perimeter = XY + XZ + YZ s = (XY + XZ + YZ) / 2

area = math.sqrt(s \* (s - XY) \* (s - XZ) \* (s - YZ))

print("Area:", area) print("Perimeter:", perimeter)

output :

Area: 4.000000000000003

Perimeter: 11.123105625617661

**Q 3 ) Attempt the following**

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

**I ) Write a program to solve the following LPP : Min Z=3.5x+2y**

**Subject to x+y≥5**

**x≥4 y≤2 x,y≥0**

**-**

from scipy.optimize import linprog c = [3.5, 2]

A = [[1, 1],

[-1, 0],

[0, 1]]

b = [5, -4, 2]

x\_bounds = (0, None) y\_bounds = (0, None)

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

print("Optimal value:", round(result.fun, 2)) print("Optimal point:", tuple(round(x, 2) for x in result.x))

output :

Optimal value: 14.0

Optimal point: (4.0, 0.0)

**II ) Write a python program to solve the LPP and find optimal solution if exist Max Z=3x+5y+4z**

**Subject to 2x+3y≤8**

**2y+5z≤10**

**3x+2y+4z≤15 X,y,z≥0**

**-**

from scipy.optimize import linprog c = [-3, -5, -4]

A = [[2, 3, 0],

[0, 2, 5],

[3, 2, 4]]

b = [8, 10, 15]

x\_bounds = (0, None) y\_bounds = (0, None) z\_bounds = (0, None)

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

if result.success:

print("Optimal value:", round(-result.fun, 2)) print("Optimal point:", tuple(round(x, 2) for x in result.x))

else:

print("Optimization failed. The problem may be infeasible or unbounded.")

output :

Optimal value: 18.66

Optimal point: (2.17, 1.22, 1.51)

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

**I ) Write a python program to apply the following transformation on the point [-2,4]**

**A ) Reflection through Reflection through –axis**

**B ) Scaling in X-coordinate by factor 6**

**C ) Scaling in Y-coordinate by factor 4.1**

**D ) Shering in X-direction by** 𝟕

𝟐

**-**

import numpy as np point = np.array([-2, 4])

reflection\_x = np.array([[1, 0], [0, -1]]) reflected\_point\_x = np.dot(reflection\_x, point)

scaling\_x = np.array([[6, 0], [0, 1]]) scaled\_point\_x = np.dot(scaling\_x, point) scaling\_y = np.array([[1, 0], [0, 4.1]]) scaled\_point\_y = np.dot(scaling\_y, point)

shearing\_x = np.array([[1, 7/2], [0, 1]]) sheared\_point\_x = np.dot(shearing\_x, point)

print("Original point: ", point)

print("Reflection through x-axis: ", reflected\_point\_x) print("Scaling in x-coordinate by factor 6: ", scaled\_point\_x) print("Scaling in y-coordinate by factor 4.1: ", scaled\_point\_y) print("Shearing in x-direction by 7/2: ", sheared\_point\_x)

output :

Original point: [-2 4]

Reflection through x-axis: [-2 -4]

Scaling in x-coordinate by factor 6: [-12 4]

Scaling in y-coordinate by factor 4.1: [-2. 16.4] Shearing in x-direction by 7/2: [12. 4.]

**II ) Write a python program to find the combined transformation on the line segment between the points A[4,1] & B[-3,0] for the following sequence of transformation**

**A ) Rotation about origin through an angle**𝝅

𝟒

**B ) Uniform scaling by 7.3 units**

**C ) Scaling in X-coordinate by 3 units**

**D ) Shering in X direction by** 𝟏 **units**

𝟐

**-**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.transforms import Affine2D

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

B = np.array([-3, 0])

AB = np.vstack((A, B))

T1 = Affine2D().rotate(np.pi/4) # Rotation about origin through an angle π/4 T2 = Affine2D().scale(7.3, 7.3) # Uniform scaling by 7.3 units

T3 = Affine2D().scale(3, 1) # Scaling in X-coordinate by 3 units T4 = Affine2D().skew(0.5, 0) # Shering in X direction by 1/2 units

T\_combined = T1 + T2 + T3 + T4 AB\_transformed = T\_combined.transform(AB) fig, ax = plt.subplots()

ax.plot(AB[:, 0], AB[:, 1], 'b', label='Original')

ax.plot(AB\_transformed[:, 0], AB\_transformed[:, 1], 'r', label='Transformed') ax.legend()

ax.set\_aspect('equal') plt.show()

