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

# Q1. Attempt any of the following

**A ) Write a python program to plot 2D graph of the function f(x)=log10(x) in the interval [0,10]**



import numpy as np

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

return np.log10(x)

x = np.linspace(0.1, 10, 100) y = f(x)

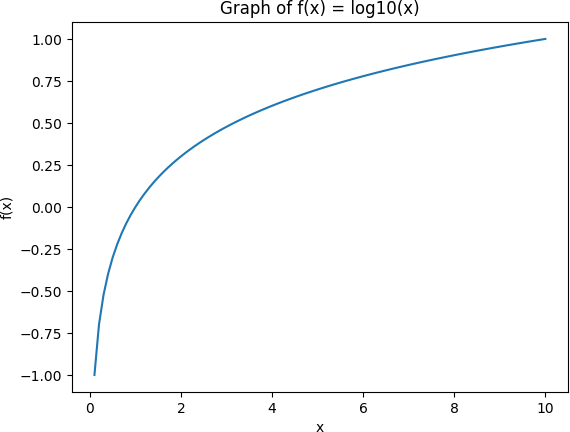
plt.plot(x, y)

plt.xlabel('x')

plt.ylabel('f(x)')

plt.title('Graph of f(x) = log10(x)') plt.show()

output :



# B ) Using Python plot the graph of function f(x)=sin-1(x) on the interval [-1,1]

-

import numpy as np

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

return np.arcsin(x)

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

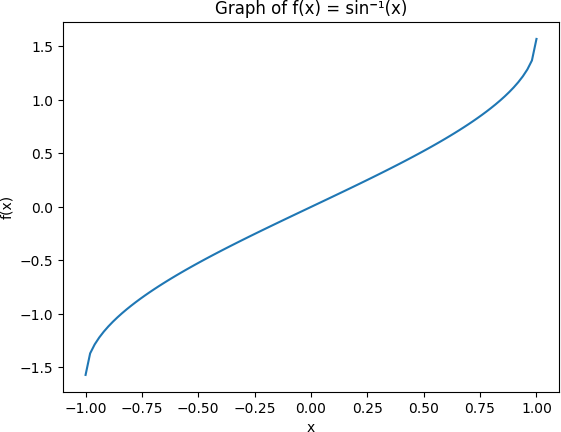
plt.plot(x, y)

plt.xlabel('x')

plt.ylabel('f(x)')

plt.title('Graph of f(x) = sin⁻¹(x)') plt.show()

output :



# C ) Using python plot the surface plot of parabola z=x2+y2 in -6<x, y<6

->

import numpy as np

import matplotlib.pyplot as plt

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

return x\*\*2 + y\*\*2

x = np.linspace(-6, 6, 100)

y = np.linspace(-6, 6, 100) X, Y = np.meshgrid(x, y) Z = f(X, Y)

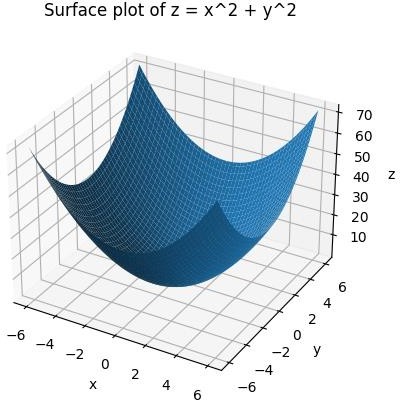
fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d') ax.plot\_surface(X, Y, Z)

ax.set\_xlabel('x') ax.set\_ylabel('y') ax.set\_zlabel('z')

ax.set\_title('Surface plot of z = x^2 + y^2') plt.show()

output :



**Q2 Attempt any TWO of the following**

**A . 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 matplotlib.pyplot as plt import numpy as np

vertices = np.array([(0,0), (2,0), (2,3), (1,6)]) plt.plot(vertices[:,0], vertices[:,1], color='blue') theta = np.pi

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

[np.sin(theta), np.cos(theta)]]) rotated\_vertices = vertices.dot(rotation\_matrix) plt.plot(rotated\_vertices[:,0], rotated\_vertices[:,1], color='red')

plt.xlim([-2, 7])

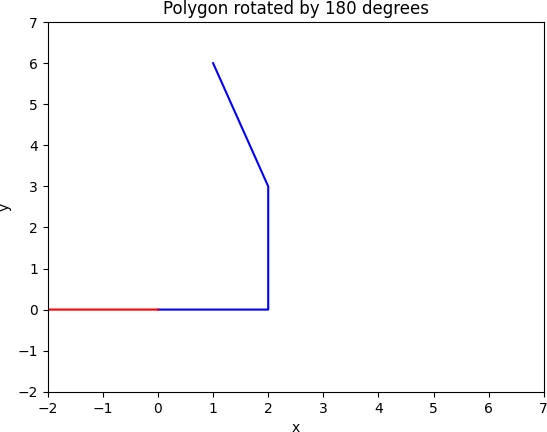
plt.ylim([-2, 7])

plt.xlabel('x')

plt.ylabel('y')

plt.title('Polygon rotated by 180 degrees') plt.show()

output :



**B ) Using python generate line passing thorugh points (2,3) and (4,3) and find equation of the line**

-

import matplotlib.pyplot as plt x1, y1 = 2, 3

x2, y2 = 4, 3

slope = (y2 - y1) / (x2 - x1) y\_intercept = y1 - slope \* x1

x = [0, 5]

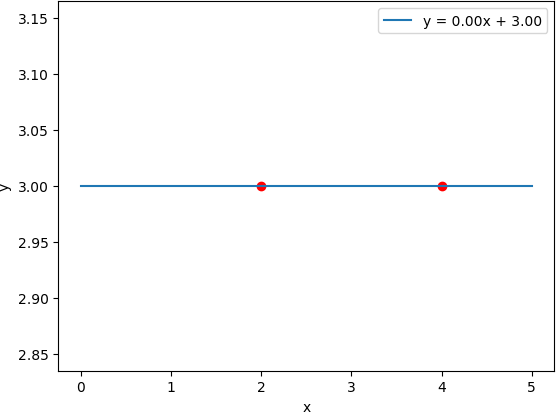
y = [slope \* xi + y\_intercept for xi in x]

plt.plot(x, y, label=f"y = {slope:.2f}x + {y\_intercept:.2f}") plt.scatter([x1, x2], [y1, y2], color='red')

plt.xlabel("x")

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

output :



**Q3 ) Attempt any ONE of the following**

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

**I )Write a pyhon program to solve the following LPP**

**MAX Z=150x+75y**

**Subject to 4x+6y≤24**

**5x+3y≤15 x,y≥0**



from pulp import \*

problem = LpProblem("LP Problem", LpMaximize) x = LpVariable('x', lowBound=0, cat='Continuous') y = LpVariable('y', lowBound=0, cat='Continuous')

problem += 150 \* x + 75 \* y problem += 4 \* x + 6 \* y <= 24 problem += 5 \* x + 3 \* y <= 15 status = problem.solve()

print(f"Status: {LpStatus[status]}") print(f"x = {value(x):.2f}")

print(f"y = {value(y):.2f}")

print(f"Z = {value(problem.objective):.2f}")

1. **Write a python to display the following LPP by using pulp module and simplex method.Find Its optimal Solution if exist**

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

**Subject to 4x+6y-5z+2w≤-20**

**-8x-3y+3z+2w≤20 X+y≤11 X,y,z,w≥0**



from pulp import \*

problem = LpProblem("LP Problem", LpMaximize) x = LpVariable('x', lowBound=0, cat='Continuous')

y = LpVariable('y', lowBound=0, cat='Continuous')

z = LpVariable('z', lowBound=0, cat='Continuous') w = LpVariable('w', lowBound=0, cat='Continuous')

problem += 4 \* x + y + 3 \* z + 5 \* w

problem += 4 \* x + 6 \* y - 5 \* z + 2 \* w <= -20 problem += -8 \* x - 3 \* y + 3 \* z + 2 \* w <= 20 problem += x + y <= 11 problem.solve(solvers.PULP\_CBC\_CMD(msg=0))

print(f"Status: {LpStatus[problem.status]}")

print(f"x = {value(x):.2f}")

print(f"y = {value(y):.2f}")

print(f"z = {value(z):.2f}")

print(f"w = {value(w):.2f}")

print(f"Z = {value(problem.objective):.2f}")

**b ) Attempt any of the following**

**1 ) plot 3D axes with labels as X-asix and z-Axis and also plot following points with given coordinates in one graph**

**I ) (70,-25,15) as a diamond in black color**

**II ) (50 , 72, -45) as a \* in green color**

**III ) (58,-82,65) as a dot in green color**

**IV ) (20,72,-45) as a \* in red color**



import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d') ax.set\_xlabel('X-axis')

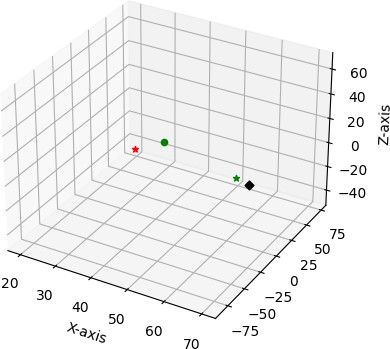
ax.set\_zlabel('Z-axis') x1, y1, z1 = 70, -25, 15

x2, y2, z2 = 50, 72, -45

x3, y3, z3 = 58, -82, 65

x4, y4, z4 = 20, 72, -45

ax.scatter(x1, y1, z1, marker='D', c='black') ax.scatter(x2, y2, z2, marker='\*', c='green') ax.scatter(x3, y3, z3, marker='o', c='green') ax.scatter(x4, y4, z4, marker='\*', c='red') plt.show()



**II ) Find the combined transformation of the line segment between the points A[4,-1] & B[3,0] by using Python program for the following sequence of transformation**

1. **) Shering in X direction by 9 units**
2. **) Rotation about origin through an angel π**
3. **) Scaling in X-coordinate by 2 units**
4. **) Reflection through the line y=x**



import numpy as np

import matplotlib.pyplot as plt

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

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

T1 = np.array([[1/9, 0], [0, 1]])

T2 = np.array([[-1, 0], [0, -1]])

T3 = np.array([[2, 0], [0, 1]])

T4 = np.array([[0, 1], [1, 0]])

AB = B - A

AB\_T1 = T1 @ AB AB\_T2 = T2 @ AB\_T1 AB\_T3 = T3 @ AB\_T2 AB\_T4 = T4 @ AB\_T3

A\_T = A + AB\_T4 B\_T = B + AB\_T4

plt.plot([A[0], B[0]], [A[1], B[1]], 'b', label='Original line segment')

plt.plot([A\_T[0], B\_T[0]], [A\_T[1], B\_T[1]], 'r', label='Transformed line segment')

plt.xlim(-10, 10)

plt.ylim(-10, 10) plt.xlabel('X-axis') plt.ylabel('Y-axis')

plt.legend() plt.show()

