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

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

**A ) Write a python program plot the graph of sin x and cos x in [0,π] in one figure with 2x1 subplots**

**-**

import numpy as np

import matplotlib.pyplot as plt x = np.linspace(0, np.pi, 100) y\_sin = np.sin(x)

y\_cos = np.cos(x)

fig, axs = plt.subplots(nrows=2, ncols=1, figsize=(6, 8))

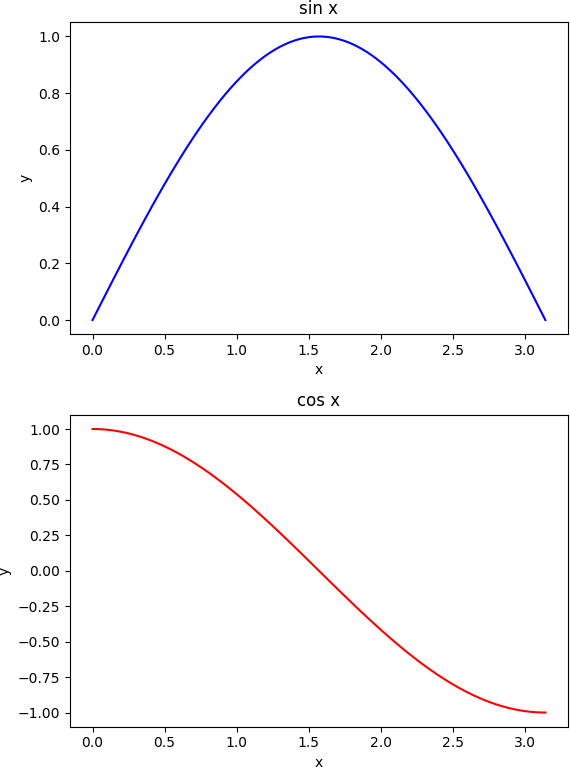
axs[0].plot(x, y\_sin, color='blue') axs[0].set\_title('sin x') axs[0].set\_xlabel('x')

axs[0].set\_ylabel('y')

axs[1].plot(x, y\_cos, color='red') axs[1].set\_title('cos x') axs[1].set\_xlabel('x')

axs[1].set\_ylabel('y')

plt.tight\_layout() plt.show()



**B ) Write a python program to plot the graph of the function in the given interval**

**I ) f(x)=x3 in [0,5]**

**II ) f(x)=x2 in [-2,2]**

**-**

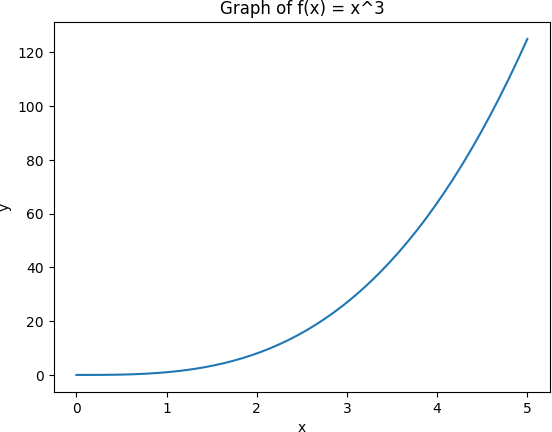
import numpy as np

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

return x\*\*3

x = np.linspace(0, 5) plt.plot(x, f(x)) plt.title('Graph of f(x) = x^3') plt.xlabel('x')

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



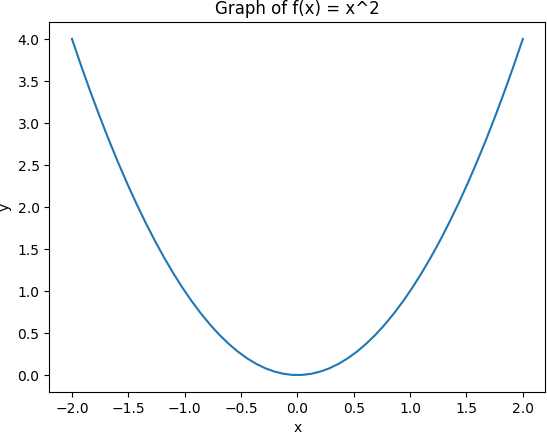
import numpy as np

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

return x\*\*2

x = np.linspace(-2, 2) plt.plot(x, f(x)) plt.title('Graph of f(x) = x^2') plt.xlabel('x')

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



**C ) Write a python program to plot 3D surface plot of the function z=cos(|x|+|Y|) in**

**-1<x,y<1**

**-**

import matplotlib.pyplot as plt import numpy as np

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

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

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

y = np.linspace(-1, 1, 100) X, Y = np.meshgrid(x, y) Z = f(X, Y)

fig = plt.figure()

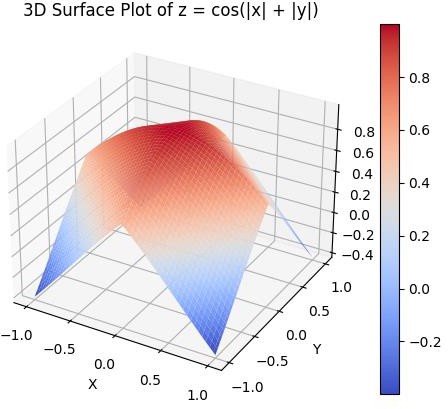
ax = fig.add\_subplot(111, projection='3d')

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

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

ax.set\_title('3D Surface Plot of z = cos(|x| + |y|)')

fig.colorbar(surf) plt.show()



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

**A ) Write a python program to draw regular with 20 sides and radius 1 ceneterd at (0,0)**

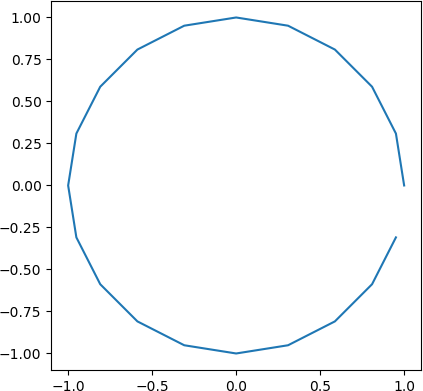
**-**

import matplotlib.pyplot as plt import numpy as np

theta = np.linspace(0, 2\*np.pi, 20, endpoint=False) x = np.cos(theta)

y = np.sin(theta)

fig, ax = plt.subplots() ax.plot(x, y) ax.set\_aspect('equal') plt.show()



**B ) Write a python program to draw a polygon with vertices (0,0),(1,0),(2,2),(1,4), also find area of polygon**

**-**

import matplotlib.pyplot as plt vertices = [(0,0), (1,0), (2,2), (1,4)]

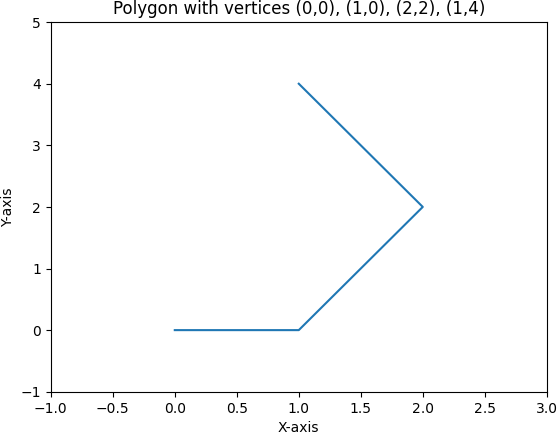
x = [vertex[0] for vertex in vertices] y = [vertex[1] for vertex in vertices]

plt.plot(x, y)

plt.xlim(-1, 3)

plt.ylim(-1, 5) plt.xlabel('X-axis') plt.ylabel('Y-axis')

plt.title('Polygon with vertices (0,0), (1,0), (2,2), (1,4)') plt.show()



**C )Write a python program to find area and perimeter of triangle ABC where A[0,1],B[-5,0] and C[-3,3]**

**-**

import math

A = [0, 1]

B = [-5, 0]

C = [-3, 3]

AB = math.sqrt((B[0]-A[0])\*\*2 + (B[1]-A[1])\*\*2)

BC = math.sqrt((C[0]-B[0])\*\*2 + (C[1]-B[1])\*\*2)

CA = math.sqrt((A[0]-C[0])\*\*2 + (A[1]-C[1])\*\*2)

perimeter = AB + BC + CA s = perimeter / 2

area = math.sqrt(s \* (s-AB) \* (s-BC) \* (s-CA))

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

output :

Perimeter = 12.310122064520764

Area = 6.500000000000002

**Q3 ) Attempt the following**

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

1. **) Write a python program to solve the following LPP : Max Z=3x+5y+4z**

**Subjecct to 2x+3y≤8**

**2x+5y≤10**

**3x+2y+4z≤15 X,y,z≥0**

**-**

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

A = [[2, 3, 0], [2, 5, 0], [3, 2, 4]]

b = [8, 10, 15]

x0\_bounds = (0, None) x1\_bounds = (0, None) x2\_bounds = (0, None)

res = linprog(c, A\_ub=A, b\_ub=b, bounds=[x0\_bounds, x1\_bounds, x2\_bounds], method='simplex')

print('Optimal value:', round(res.fun \* -1, 2)) print('x:', res.x)

output :

Optimal value: 21.0

x: [0. 2. 2.75]

1. **) Write a python program to solve the following LPP : Min Z=3x+5y+4z**

**Subject to 2x+2y≤12**

**2x+2y≤10**

**5x+2y≤10 X,y≥0**

**-**

from scipy.optimize import linprog obj\_coeff = [3, 5, 4]

lhs\_ineq\_coeff = [ [2, 2, 0],

[2, 2, 0],

[5, 2, 0]

]

rhs\_ineq\_value = [12, 10, 10]

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

res = linprog(c=obj\_coeff, A\_ub=lhs\_ineq\_coeff, b\_ub=rhs\_ineq\_value, bounds=bounds, method='simplex')

print("Optimal value of Z:", round(res.fun, 2)) print("Optimal values of x, y, z:", res.x)

output :

Optimal value of Z: 0.0

Optimal values of x, y, z: [0. 0. 0.]

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

1. **) Write the python program to apply each of the following transformation on the point p[3,-1]**

**A ) Reflection through X axis**

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

**C ) Scalling in Y coordinate by factor 8**

**D ) Shering in X direction by 2 units**

**-**

import numpy as np p = np.array([3, -1])

p\_reflected = np.array([p[0], -p[1]]) angle = 30\*np.pi/180

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

[np.sin(angle), np.cos(angle)]]) p\_rotated = rotation\_matrix @ p

scaling\_matrix = np.array([[1, 0],

[0, 8]])

p\_scaled = scaling\_matrix @ p

shearing\_matrix = np.array([[1, 0],

[2, 1]])

p\_sheared = shearing\_matrix @ p

# printing the results print("Original point: ", p)

print("Reflection through X axis: ", p\_reflected) print("Rotation about origin by 30 degrees: ", p\_rotated) print("Scaling in Y coordinate by a factor of 8: ", p\_scaled) print("Shearing in X direction by 2 units: ", p\_sheared)

output :

Original point: [ 3 -1]

Reflection through X axis: [3 1]

Rotation about origin by 30 degrees: [3.09807621 0.6339746 ] Scaling in Y coordinate by a factor of 8: [ 3 -8]

Shearing in X direction by 2 units: [3 5]

1. **) Write a python program to apply the each of the following transformation on the point P[-2,4]**

**A ) Reflection through the line y=x+2**

**B ) Scaling in Y-coordination by factor 2**

**C ) Shering in X direction by units**

**D ) Rotation about origin by an angle 60 degrees**

**-**

import math P = [-2, 4]

m = 1

c = 2

x\_new = (P[1] - c + m\*P[0])/(1 + m\*\*2)

y\_new = (m\*P[1] + m\*\*2\*P[0] + c)/(1 + m\*\*2) P\_reflected = [x\_new, y\_new]

print("Reflection through the line y = x + 2:", P\_reflected)

k = 2

x\_new = P[0] y\_new = k\*P[1]

P\_scaled = [x\_new, y\_new]

print("Scaling in Y-coordinate by a factor of 2:", P\_scaled)

k = 2

x\_new = P[0] + k\*P[1]

y\_new = P[1]

P\_sheared = [x\_new, y\_new]

print("Shearing in X-direction by 2 units:", P\_sheared)

theta = math.radians(60)

x\_new = P[0]\*math.cos(theta) - P[1]\*math.sin(theta) y\_new = P[0]\*math.sin(theta) + P[1]\*math.cos(theta) P\_rotated = [x\_new, y\_new]

print("Rotation about the origin by an angle of 60 degrees:", P\_rotated)

output :

Reflection through the line y = x + 2: [0.0, 2.0] Scaling in Y-coordinate by a factor of 2: [-2, 8] Shearing in X-direction by 2 units: [6, 4]

Rotation about the origin by an angle of 60 degrees: [-4.464101615137754, 0.26794919243112325]