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

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

**A ) Write a python program to plot 3D axes with labels as X-axis ,Y-axis and Z-axis and also pack following point with given Coordination in the same graph :(70,-25,15) as a diamond in black color**

**-**

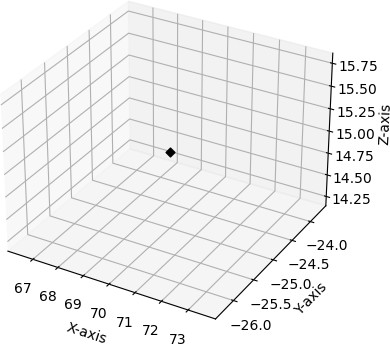
import matplotlib.pyplot as plt

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

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

ax.set\_xlabel('X-axis') ax.set\_ylabel('Y-axis') ax.set\_zlabel('Z-axis') x, y, z = (70,-25,15)

ax.scatter(x, y, z, marker='D', color='black') plt.show()



**B ) Plot the graph of y=e-x2 in [-5,5] with red dashed-points line with Upward Pointing triangle**

**-**

import numpy as np

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

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

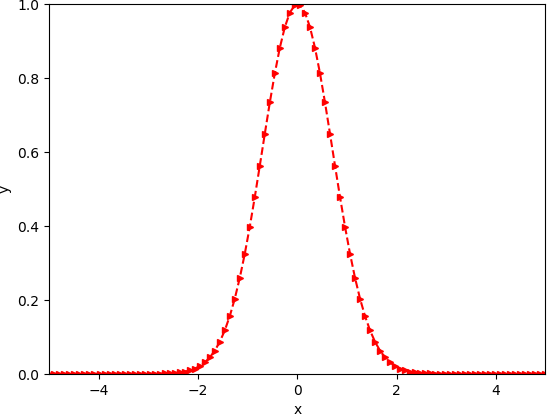
x = np.linspace(-5, 5, 100) y = f(x)

fig, ax = plt.subplots() ax.set\_xlim([-5, 5])

ax.set\_ylim([0, 1])

ax.plot(x, y, 'r-->', markersize=5) ax.set\_xlabel('x') ax.set\_ylabel('y')

plt.show()



**C ) Draw a bar graph in GREEN colour to reprenset the data below :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Subject** | **Maths** | **Science** | **English** | **Marathi** | **Hindi** |
| **Percentage of passing** | **68** | **90** | **70** | **85** | **91** |

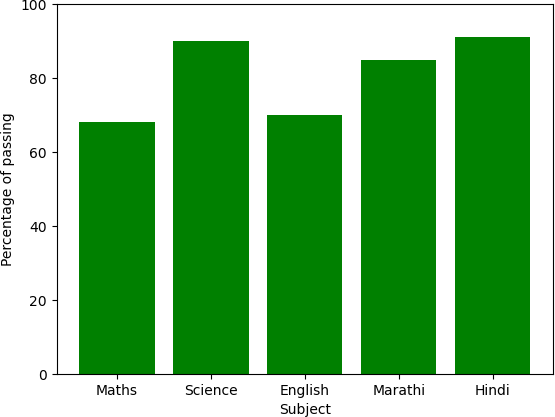
**-**

import matplotlib.pyplot as plt

subjects = ['Maths', 'Science', 'English', 'Marathi', 'Hindi'] percentages = [68, 90, 70, 85, 91]

fig, ax = plt.subplots() ax.set\_ylim([0, 100])

ax.bar(subjects, percentages, color='green') ax.set\_xlabel('Subject') ax.set\_ylabel('Percentage of passing') plt.show()



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

**I ) Write a python program to reflect the ∆ABC through the line Y=3 where A(1,0),B(2,-1),C(-1,3)**

**-**

import numpy as np

import matplotlib.pyplot as plt

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

B = np.array([2, -1])

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

triangle = np.array([A, B, C, A]) line = np.array([[-5, 3], [5, 3]])

translation\_matrix = np.array([[1, 0, 0], [0, 1, -3], [0, 0, 1]])

reflection\_matrix = np.array([[1, 0, 0], [0, -1, 0], [0, 0, 1]])

reflected\_triangle = np.dot(reflection\_matrix, np.concatenate((triangle.T, np.ones((1, 4))), axis=0)).T[:, :2]

inverse\_translation\_matrix = np.array([[1, 0, 0], [0, 1, 3], [0, 0, 1]])

reflected\_triangle = np.dot(inverse\_translation\_matrix, np.concatenate((reflected\_triangle.T, np.ones((1, 4))), axis=0)).T[:, :2]

fig, ax = plt.subplots()

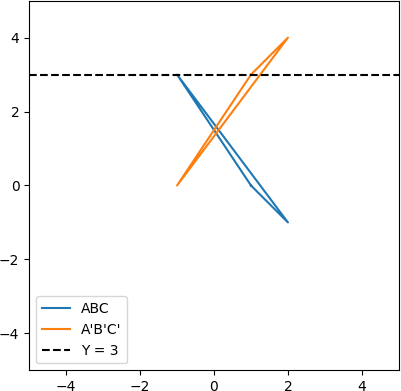
ax.plot(triangle[:, 0], triangle[:, 1], label='ABC')

ax.plot(reflected\_triangle[:, 0], reflected\_triangle[:, 1], label='A\'B\'C\'')

ax.plot(line[:, 0], line[:, 1], 'k--', label='Y = 3')

ax.set\_xlim([-5, 5])

ax.set\_ylim([-5, 5]) ax.set\_aspect('equal') ax.legend() plt.show()



1. **) Write a python program to rotate the ∆ABC by 90⸰ where A(1,2),B(2,-2),C(-1,2)**

**-**

import numpy as np

import matplotlib.pyplot as plt

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

B = np.array([2, -2])

C = np.array([-1, 2])

triangle = np.array([A, B, C, A])

angle = np.radians(90)

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

rotated\_triangle = np.dot(rotation\_matrix, triangle.T).T

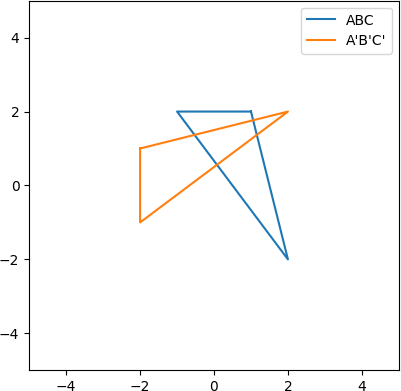
fig, ax = plt.subplots()

ax.plot(triangle[:, 0], triangle[:, 1], label='ABC')

ax.plot(rotated\_triangle[:, 0], rotated\_triangle[:, 1], label='A\'B\'C\'')

ax.set\_xlim([-5, 5])

ax.set\_ylim([-5, 5]) ax.set\_aspect('equal') ax.legend() plt.show()



1. **) Write a python program to draw a polygon with 6 sides and radius 1 centered at (1,2) and find its area and perimeter**

**-**

import matplotlib.pyplot as plt import numpy as np

x\_center = 1

y\_center = 2

n\_sides = 6

radius = 1

angles = np.linspace(0, 2 \* np.pi, n\_sides, endpoint=True) x = x\_center + radius \* np.cos(angles)

y = y\_center + radius \* np.sin(angles) plt.plot(x, y)

plt.axis('equal') plt.show()

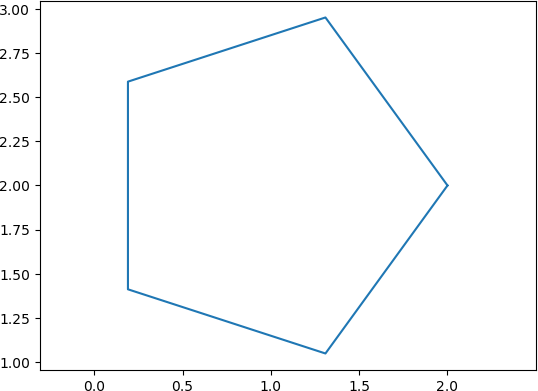
side\_length = 2 \* radius \* np.sin(np.pi/n\_sides) perimeter = n\_sides \* side\_length

area = (n\_sides \* radius\*\*2 \* np.sin(2 \* np.pi / n\_sides))/2 print("Perimeter:", perimeter)

print("Area:", area) output :

Perimeter: 5.999999999999999

Area: 2.598076211353316



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

1. **) Attempt any ONE of the following**

**A ) Solve LPP by using Python : Min Z=x+y**

**Subject to x≥6**

**y≥6 x+y≥11 x,y≥0**

**-**

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

A = [[-1, 0], [0, -1], [-1, -1]]

b = [-6, -6, -11]

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

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

print('Optimal value:', res.fun) print('x:', res.x[0])

print('y:', res.x[1])

output :

Optimal value: 12.0

x: 6.0

y: 6.0

**B) Write a python program to solve the following LPP and to find the optimal solution if exists**

**MAX Z=3x+5y +4z**

**Subject to 2x+3y≤8**

**2y+5z≤10**

**3x+2y+4z≤15 X,y,z≥0**

**-**

import pulp

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

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

z = pulp.LpVariable('z', lowBound=0, cat='Continuous') prob += 3\*x + 5\*y + 4\*z

prob += 2\*x + 3\*y <= 8 prob += 2\*y + 5\*z <= 10

prob += 3\*x + 2\*y + 4\*z <= 15 status = prob.solve()

print("Status:", pulp.LpStatus[status]) print("Optimal Solution:")

print("x =", pulp.value(x))

print("y =", pulp.value(y))

print("z =", pulp.value(z))

print("Optimal Value: Z =", pulp.value(prob.objective))

1. **) Attempt Any one of the following**
   1. **Write a python program to apply the following transformation on the point (-2,4) :**
      1. **Reflection through X-axis**
      2. **Scaling in X-coordinate by factor 6**
      3. **Scaling in x-direction by 4 units**
      4. **Rotate about origin through an angle 30⸰**

**-**

import math point = [-2, 4]

reflected\_point = [point[0], -1\*point[1]]

print("I) Reflection through X-axis:", reflected\_point)

scaled\_point = [6\*point[0], point[1]]

print("II) Scaling in X-coordinate by factor 6:", scaled\_point)

shifted\_point = [point[0] + 4, point[1]]

print("III) Scaling in x-direction by 4 units:", shifted\_point)

angle = math.radians(30)

rotated\_point = [point[0]\*math.cos(angle) - point[1]\*math.sin(angle), point[0]\*math.sin(angle) + point[1]\*math.cos(angle)]

print("IV) Rotate about origin through an angle 30⸰:", rotated\_point)

Output :

1. Reflection through X-axis: [-2, -4]
2. Scaling in X-coordinate by factor 6: [-12, 4]
3. Scaling in x-direction by 4 units: [2, 4]
4. Rotate about origin through an angle 30⸰: [-3.732050807568877, 2.464101615137755]

**II )Write a python program to find the combined transformation between the points for the following sequence of transformation**

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

𝟐

**B ) uniform scaling by 3.5 units**

**C ) Scaling in X & Y direction by 3 & 5 units respectively**

**D ) shering in X direction by 6 units**

**-**

import math

import numpy as np point = np.array([2, 5]) theta = math.pi/2

rotation\_matrix = np.array([[math.cos(theta), -1\*math.sin(theta)], [math.sin(theta), math.cos(theta)]])

point = rotation\_matrix.dot(point)

print("After A) Rotation about origin through an angle π/2: ", point)

scaling\_factor = 3.5

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

[0, scaling\_factor]]) point = scaling\_matrix.dot(point)

print("After B) Uniform scaling by 3.5 units: ", point)

scaling\_factors = np.array([3, 5]) scaling\_matrix = np.diag(scaling\_factors) point = scaling\_matrix.dot(point)

print("After C) Scaling in X & Y direction by 3 & 5 units respectively: ", point)

shearing\_factor = 6

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

[0, 1]])

point = shearing\_matrix.dot(point)

print("After D) Shering in X direction by 6 units: ", point)

output :

After A) Rotation about origin through an angle π/2: [-5. 2.] After B) Uniform scaling by 3.5 units: [-17.5 7. ]

After C) Scaling in X & Y direction by 3 & 5 units respectively: [-52.5 35. ] After D) Shering in X direction by 6 units: [157.5 35. ]