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

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

**A ) Write a python program to plot the function f(x,y)=-x2-y2 when -10≤x,y≤10**

**-**

import matplotlib.pyplot as plt import numpy as np

def f(x, y):

return -x\*\*2 - y\*\*2

x = np.linspace(-10, 10, 100)

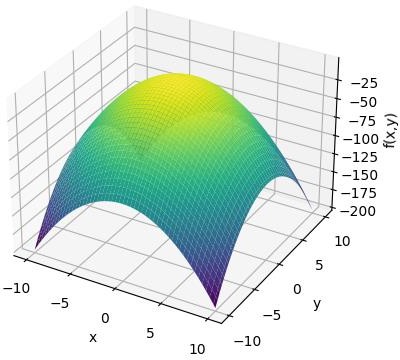
y = np.linspace(-10, 10, 100) X, Y = np.meshgrid(x, y)

Z = f(X, Y)

fig = plt.figure()

ax = fig.add\_subplot(projection='3d') ax.plot\_surface(X, Y, Z, cmap='viridis') ax.set\_xlabel('x')

ax.set\_ylabel('y') ax.set\_zlabel('f(x,y)') plt.show()



**B ) Write a python program to plot graph of the function f(x)=log(3x2) in [1,10] with black dashed points**

**-**

import matplotlib.pyplot as plt import numpy as np

def f(x):

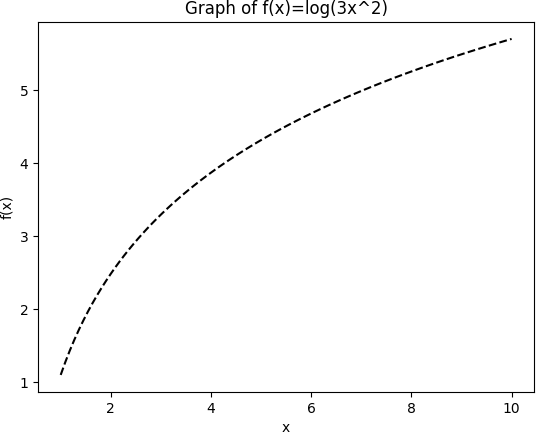
return np.log(3\*x\*\*2)

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

plt.plot(x, y, 'k--') plt.xlabel('x')

plt.ylabel('f(x)')

plt.title('Graph of f(x)=log(3x^2)') plt.show()



**C ) Write a python program to generate plot of the function f(x)=x2 in the interval [-5,5] in figure of size of 6x6 inches**

**-**

import matplotlib.pyplot as plt import numpy as np

def f(x):

return x\*\*2

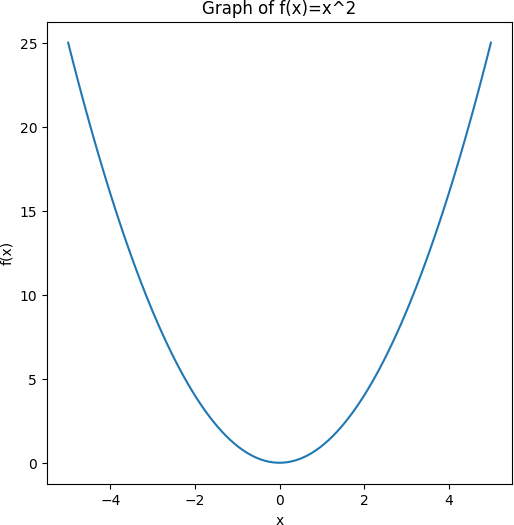
x = np.linspace(-5, 5, 100) y = f(x)

fig = plt.figure(figsize=(6, 6)) plt.plot(x, y)

plt.xlabel('x')

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

plt.show()



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

**A ) Write a python program to declare the line segment passing through the points A(0,7) , B(5,2) . also find the length and midpoint of the line segment passing through points A and B**

**-**

import math Ax, Ay = 0, 7

Bx, By = 5, 2

slope = (By - Ay) / (Bx - Ax)

y\_intercept = Ay - slope \* Ax

print("Equation of the line: y = {}x + {}".format(slope, y\_intercept))

length = math.sqrt((Bx - Ax)\*\*2 + (By - Ay)\*\*2) print("Length of the line segment: {:.2f}".format(length))

midpoint\_x = (Ax + Bx) / 2 midpoint\_y = (Ay + By) / 2

print("Midpoint of the line segment: ({}, {})".format(midpoint\_x, midpoint\_y))

output :

Equation of the line: y = -1.0x + 7.0 Length of the line segment: 7.07 Midpoint of the line segment: (2.5, 4.5)

**B ) Write a python program to draw a polygon with vertices (0,0),(2,0),(2,3),(1,6) and rotate it by 90⸰**

**-**

import matplotlib.pyplot as plt import numpy as np

vertices = np.array([[0, 0], [2, 0], [2, 3], [1, 6]]) fig, ax = plt.subplots()

polygon = plt.Polygon(vertices, color='blue') ax.add\_patch(polygon)

ax.set\_xlim(-2, 8)

ax.set\_ylim(-2, 8)

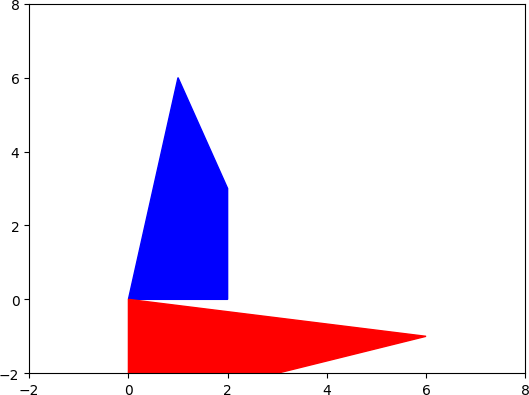
rotation\_angle = 90

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

[np.sin(np.radians(rotation\_angle)), np.cos(np.radians(rotation\_angle))]]) vertices\_rotated = np.dot(vertices, rotation\_matrix)

polygon\_rotated = plt.Polygon(vertices\_rotated, color='red') ax.add\_patch(polygon\_rotated)

plt.show()



**C) Write a python program to generate vector x in the interval [0,15] using numpy package with 100 subintervals**

**-**

import numpy as np x\_min, x\_max = 0, 15

num\_intervals = 100

x = np.linspace(x\_min, x\_max, num\_intervals) print(x)

output

|  |  |  |  |
| --- | --- | --- | --- |
| [ 0. | 0.15151515 0.3030303 0.45454545 0.60606061 0.75757576 | | |
| 0.90909091 | | 1.06060606 1.21212121 1.36363636 1.51515152 1.66666667 | |
| 1.81818182 | | 1.96969697 2.12121212 2.27272727 2.42424242 2.57575758 | |
| 2.72727273 | | 2.87878788 3.03030303 3.18181818 3.33333333 3.48484848 | |
| 3.63636364 | | 3.78787879 3.93939394 4.09090909 4.24242424 4.39393939 | |
| 4.54545455 | | 4.6969697 4.84848485 5. | 5.15151515 5.3030303 |
| 5.45454545 | | 5.60606061 5.75757576 5.90909091 6.06060606 6.21212121 | |
| 6.36363636 | | 6.51515152 6.66666667 6.81818182 6.96969697 7.12121212 | |
| 7.27272727 | | 7.42424242 7.57575758 7.72727273 7.87878788 8.03030303 | |
| 8.18181818 | | 8.33333333 8.48484848 8.63636364 8.78787879 8.93939394 | |
| 9.09090909 | | 9.24242424 9.39393939 9.54545455 9.6969697 9.84848485 | |
| 10. | 10.15151515 10.3030303 10.45454545 10.60606061 10.75757576 | | |

10.90909091 11.06060606 11.21212121 11.36363636 11.51515152 11.66666667

11.81818182 11.96969697 12.12121212 12.27272727 12.42424242 12.57575758

12.72727273 12.87878788 13.03030303 13.18181818 13.33333333 13.48484848

13.63636364 13.78787879 13.93939394 14.09090909 14.24242424 14.39393939

14.54545455 14.6969697 14.84848485 15. ]

**Q 3 ) Attempt the following**

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

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

**Subject to 3x+5y≤15**

**6x+2y≥24 X,y≥0**

**-**

import numpy as np

from scipy.optimize import linprog obj = np.array([-5, -3])

lhs\_eq = np.array([[3, 5], [-6, -2]]) rhs\_eq = np.array([15, -24])

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

res = linprog(c=obj, A\_eq=lhs\_eq, b\_eq=rhs\_eq, bounds=bnd, method='simplex')

print("x = ", res.x[0])

print("y = ", res.x[1])

print("Z = ", -res.fun)

output :

x = 3.75

y = 0.75

Z = 21.0

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

**Subject to x+y≥5**

**x≥5 y≤2 x,y≥0**

**-**

import numpy as np

from scipy.optimize import linprog obj = np.array([3.5, 2])

lhs\_eq = np.array([[-1, -1], [-1, 0], [0, 1]])

rhs\_eq = np.array([-5, -5, 2])

bnd = [(5, None), (0, 2)]

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

print("x = ", res.x[0])

print("y = ", res.x[1])

print("Z = ", res.fun)

output :

x = 5.0

y = 0.0

Z = 17.5

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

**I ) Write a python program to plot the tringle with vertices at [4,3],[6,3],[6,5] and its reflections through 1) X-axis 2) Y-axis . All the figures must be in different colors , also plot the two axes**

**-**

import matplotlib.pyplot as plt import numpy as np

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

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

ry = np.array([[-1, 0], [0, 1]]) tri\_rx = np.dot(triangle, rx) tri\_ry = np.dot(triangle, ry)

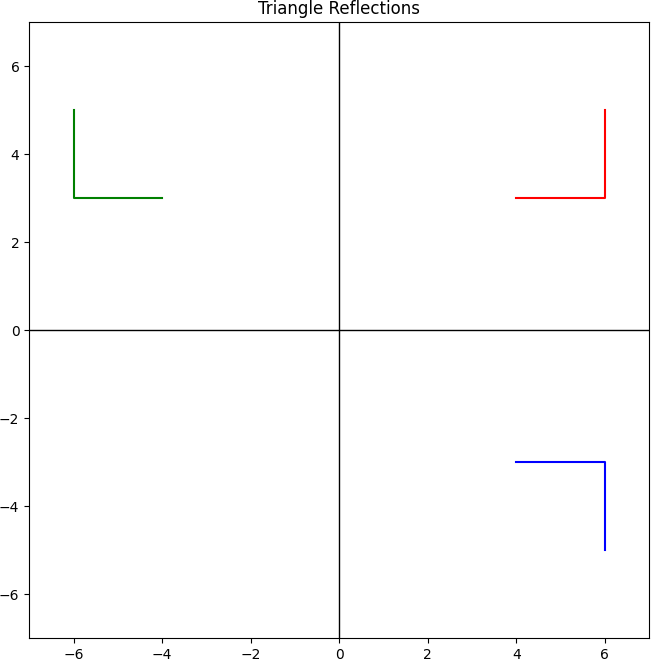
fig, ax = plt.subplots(figsize=(8, 8))

ax.axhline(0, color='black', lw=1) ax.axvline(0, color='black', lw=1) ax.plot(triangle[:,0], triangle[:,1], color='red')

ax.plot(tri\_rx[:,0], tri\_rx[:,1], color='blue')

ax.plot(tri\_ry[:,0], tri\_ry[:,1], color='green') ax.set\_xlim(-7, 7)

ax.set\_ylim(-7, 7) ax.set\_title('Triangle Reflections') plt.show()



**II ) Write a python program to plot the triangle with vertices at [3,3],[3,6],[0,6] and its reflection through line y=x and y-axis. Also plot the mirror lines**

**-**

import matplotlib.pyplot as plt import numpy as np

triangle = np.array([[3, 3], [3, 6], [0, 6]])

y\_x = np.array([[0, 1], [1, 0]])

y\_axis = np.array([[-1, 0], [0, 1]]) tri\_y\_x = np.dot(triangle, y\_x)

tri\_y\_axis = np.dot(triangle, y\_axis)

x\_eq\_y = np.linspace(0, 8, 100) y\_eq\_0 = np.zeros\_like(x\_eq\_y) y\_eq\_x = x\_eq\_y

fig, ax = plt.subplots(figsize=(8, 8))

ax.plot(x\_eq\_y, y\_eq\_0, color='black', linestyle='--', linewidth=1) ax.plot(y\_eq\_x, color='black', linestyle='--', linewidth=1) ax.plot(triangle[:,0], triangle[:,1], color='red')

ax.plot(tri\_y\_x[:,0], tri\_y\_x[:,1], color='blue') ax.plot(tri\_y\_axis[:,0], tri\_y\_axis[:,1], color='green') ax.set\_xlim(-2, 8)

ax.set\_ylim(-2, 8) ax.set\_title('Triangle Reflections') plt.show()

