Sahakar Maharshi Bhausaheb Santuji Thorat College Sangamner

DEPARTMENT OF COMPUTER SCIENCE

MATHEMATICS

Remark
Demonstrators
Signature
Date :- / /2023

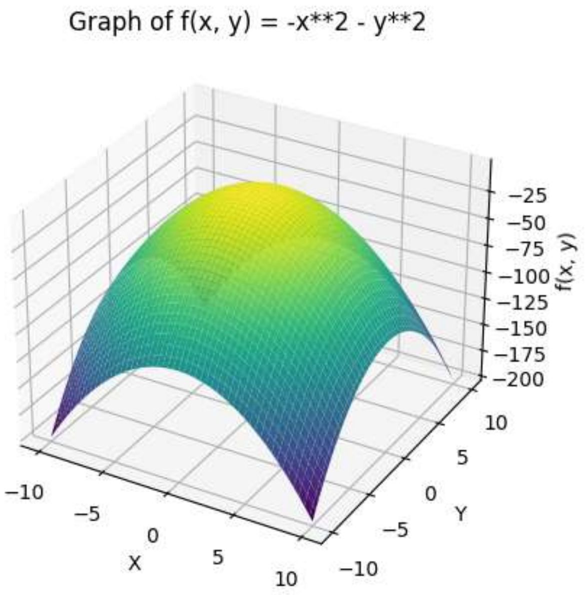
Name :- Prem Vijay Vajare **Batch No. :- D** **Roll No:- 75** **Date:-** / /2023

Title of the:- Practical 16 **Expt. No .**16 **Class :-** S.Y.BCS

Q.1) Write a Python program to plot graph of the function f (x, y) = -x*2 – y**2 when -10<=x,y<= 10.

Syntax:

```
import matplotlib.pyplot as plt

import numpy as np

# Define the function

def f(x, y):

    return -x**2 - y**2

# Generate x and y values within the range of -10 to 10

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

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

# Create a grid of x and y values

X, Y = np.meshgrid(x, y)

# Compute the values of f(x, y) for each (x, y) in the grid

Z = f(X, Y)

# Plot the surface using matplotlib

fig = plt.figure()

ax = fig.add_subplot(111, projection='3d')

ax.plot_surface(X, Y, Z, cmap='viridis')

ax.set_xlabel('X')

ax.set_ylabel('Y')

ax.set_zlabel('f(x, y)')

ax.set_title('Graph of f(x, y) = -x**2 - y**2')

plt.show()
```

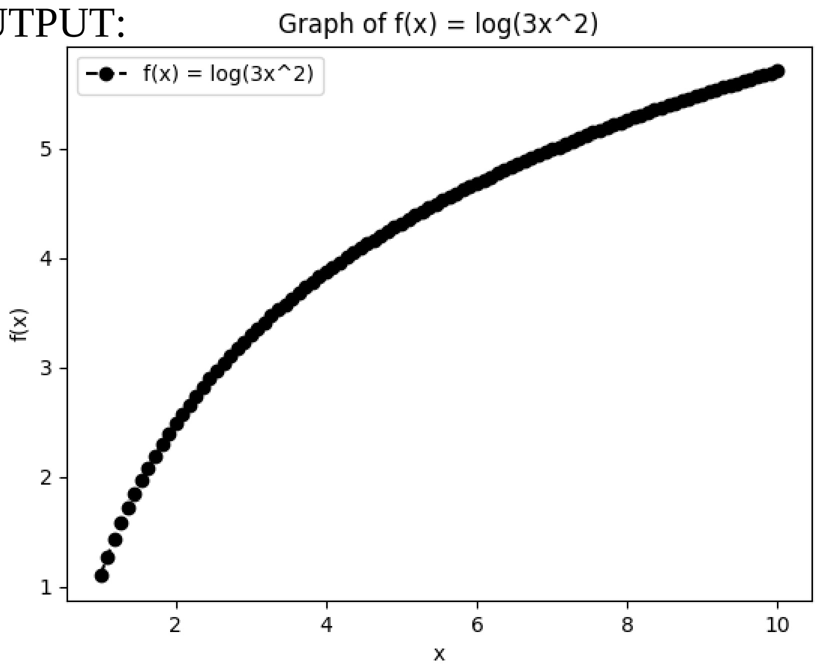
OUTPUT: 

Q.2) Write a Python program to plot graph of the function f(x) = log(3x^2) in [1,10] with black dashed points

Syntax:

```
import matplotlib.pyplot as plt
import numpy as np
# Define the function
def f(x):
    return np.log(3 * x**2)
# Generate x values within the range of [1, 10]
x = np.linspace(1, 10, 100)
# Compute the values of f(x) for each x in the range
y = f(x)
# Plot the graph with black dashed points
plt.plot(x, y, 'o--', color='black', label='f(x) = log(3x^2)')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Graph of f(x) = log(3x^2)')
plt.legend()
plt.show()
```

OUTPUT:



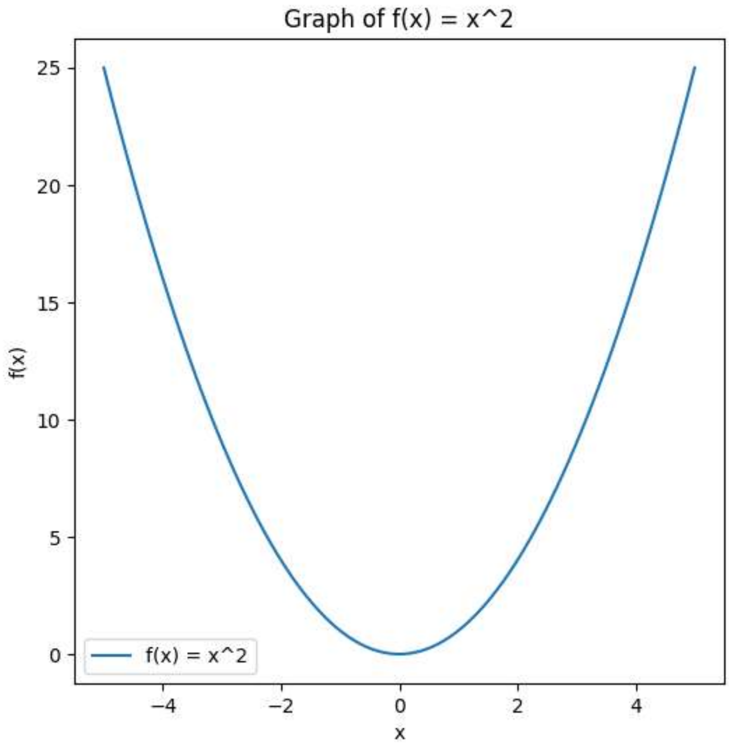
Q.3) Write python program to generate plot of the function f(x) = x^2, in the interval [-5,5] in figure of size 6X6 inches

Syntax:

```
import matplotlib.pyplot as plt
import numpy as np
# Define the function
def f(x):
    return x**2
# Generate x values within the range of [-5, 5]
x = np.linspace(-5, 5, 100)
# Compute the values of f(x) for each x in the range
y = f(x)
# Create a figure with size 6x6 inches
fig = plt.figure(figsize=(6, 6))
# Plot the graph of the function
plt.plot(x, y, label='f(x) = x^2')
plt.xlabel('x')
plt.ylabel('f(x)')
```

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

OUTPUT:



Q.4) 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.

Syntax:

```
import math
# Define the coordinates of points A and B
xA, yA = 0, 7
xB, yB = 5, 2
# Calculate the length of the line segment using the distance formula
length = math.sqrt((xB - xA)**2 + (yB - yA)**2)
# Calculate the midpoint of the line segment
midpoint_x = (xA + xB) / 2
midpoint_y = (yA + yB) / 2
# Print the equation of the line passing through A and B
print("The equation of the line passing through A and B is: ")
```

```
print(f"y - {yA} = {(yB - yA) / (xB - xA)}(x - {xA})")
# Print the length and midpoint of the line segment
print(f"Length of the line segment: {length}")
print(f"Midpoint of the line segment: ({midpoint_x}, {midpoint_y})")
```

Output:
The equation of the line passing through A and B is:
y - 7 = -1.0(x - 0)
Length of the line segment: 7.0710678118654755
Midpoint of the line segment: (2.5, 4.5)

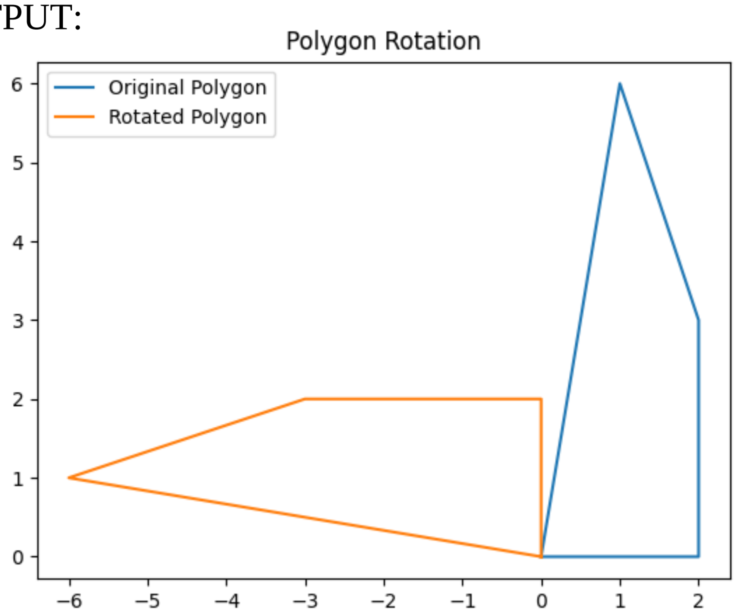
Q.5) Write a Python program to draw a polygon with vertices (0, 0), (2, 0), (2, 3) and (1, 6) and rotate it by 90°.

Syntax:

```
import matplotlib.pyplot as plt
import numpy as np
# Define the vertices of the polygon
vertices = np.array([[0, 0], [2, 0], [2, 3], [1, 6], [0, 0]])
# Plot the original polygon
plt.plot(vertices[:, 0], vertices[:, 1], label='Original Polygon')
# Define the rotation angle in degrees
rotation_angle = 90
# Convert the rotation angle to radians
theta = np.radians(rotation_angle)
# Create the rotation matrix
rotation_matrix = np.array([[np.cos(theta), -np.sin(theta)],[np.sin(theta), np.cos(theta)]])
# Apply the rotation matrix to the vertices of the polygon
rotated_vertices = np.dot(vertices, rotation_matrix.T)
# Plot the rotated polygon
plt.plot(rotated_vertices[:, 0], rotated_vertices[:, 1], label='Rotated Polygon')
# Set the aspect ratio to 'equal' for a square plot
plt.axis('equal')
# Add legend and title
plt.legend()
plt.title('Polygon Rotation')
```

# Show the plot
plt.show()

OUTPUT:



Q.6) Write a Python program to Generate vector x in the interval [0, 15] using numpy package with 100 subintervals. import numpy as np

import numpy as np

Define the start and end values of the interval

start = 0

end = 15

Define the number of subintervals

num_subintervals = 100

Generate the vector x with equally spaced values in the interval [0, 15]

x = np.linspace(start, end, num=num_subintervals+1)

Print the generated vector x

print("Generated vector x:")

print(x)

OUTPUT:

Generated vector x:

```
[ 0.    0.15  0.3   0.45  0.6   0.75  0.9   1.05  1.2   1.35  1.5   1.65
  1.8   1.95  2.1   2.25  2.4   2.55  2.7   2.85  3.    3.15  3.3   3.45
  3.6   3.75  3.9   4.05  4.2   4.35  4.5   4.65  4.8   4.95  5.1   5.25
  5.4   5.55  5.7   5.85  6.    6.15  6.3   6.45  6.6   6.75  6.9   7.05
  7.2   7.35  7.5   7.65  7.8   7.95  8.1   8.25  8.4   8.55  8.7   8.85
  9.    9.15  9.3   9.45  9.6   9.75  9.9  10.05 10.2  10.35 10.5  10.65
 10.8  10.95 11.1  11.25 11.4  11.55 11.7  11.85 12.   12.15 12.3  12.45
 12.6  12.75 12.9  13.05 13.2  13.35 13.5  13.65 13.8  13.95 14.1  14.25
 14.4  14.55 14.7  14.85 15.  ]
```

Q.7) write a Python program to solve the following LPP

Max Z = 3.5x +2 y

Subjected to

x + y >= 5

x >= 4

y <= 2

x > 0 , y > 0

Syntax:

```
import numpy as np

from scipy.optimize import linprog

# Coefficients of the objective function

c = [-3.5, -2]

# Coefficients of the inequality constraints

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

b = [-5, -4, 2]

# Bounds on the variables

x_bounds = (0, None)

y_bounds = (0, None)

# Solve the linear programming problem
```

```
result = linprog(c, A_ub=A, b_ub=b, bounds=[x_bounds, y_bounds])
if result.success:
    print("Optimal solution found:")
    print("x =", result.x[0])
    print("y =", result.x[1])
    print("Maximum value of Z =", -result.fun)
else:
    print("Optimal solution not found.")
```

OUTPUT:

Optimal solution not found.

Q.8) Write a python program to display the following LPP by using pulp module and simplex method. Find its optimal solution if exist.

Min Z = 5x+3y
subject to
x+y >= 5
x >= 4
y <= 2
x>= 0, y>= 0

Syntax:

```
from pulp import *
# Create the LP problem
problem = LpProblem("LPP", LpMinimize)
# Define the variables
x = LpVariable("x", lowBound=0)
y = LpVariable("y", lowBound=0)
# Define the objective function
problem += 5*x + 3*y
# Define the constraints
problem += x + y >= 5
problem += x >= 4
problem += y <= 2
# Solve the LP problem
problem.solve()
# Print the status of the solution
```

```
print("Status:", LpStatus[problem.status])
# If the solution is optimal, print the optimal values of x, y, and Z
if problem.status == 1:
    print("Optimal Solution:")
    print("x =", value(x))
    print("y =", value(y))
    print("Z =", 5*value(x) + 3*value(y))
else:
    print("No Optimal Solution Found.")
```

OUTPUT:

```
Status: Optimal
Optimal Solution:
x = 4.0
y = 1.0
Z = 23.0
```

Q.9) Write a python program to plot the Triangle 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.

Syntax:

```
import matplotlib.pyplot as plt
import numpy as np
# Define the vertices of the original triangle
triangle_vertices = np.array([[4, 3], [6, 3], [6, 5], [4, 3]])
# Reflect the triangle through the x-axis
x_reflected_vertices = np.array([triangle_vertices[:, 0], -triangle_vertices[:,
1]]).T
# Reflect the triangle through the y-axis
y_reflected_vertices = np.array([-triangle_vertices[:, 0], triangle_vertices[:,
1]]).T
# Plot the original triangle in red color
plt.plot(triangle_vertices[:, 0], triangle_vertices[:, 1], 'r', label='Original
Triangle')
# Plot the x-reflected triangle in blue color
plt.plot(x_reflected_vertices[:, 0], x_reflected_vertices[:, 1], 'b', label='X-
Reflected Triangle')
# Plot the y-reflected triangle in green color
plt.plot(y_reflected_vertices[:, 0], y_reflected_vertices[:, 1], 'g', label='Y-
Reflected Triangle')
```

```
# Set the axis labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Triangle and Its Reflections')
plt.legend()
# Show the plot
plt.show()
```

OUTPUT:

Q.10) Write a python program to plot the Triangle with vertices at [3, 3], [3, 6], [0, 6] and its reflections through, line y = x and y-axis. Also plot the mirror lines.

Syntax:

```
import matplotlib.pyplot as plt
import numpy as np
# Triangle vertices
triangle_vertices = np.array([[3, 3], [3, 6], [0, 6], [3, 3]])
# Reflection through y = x
reflection_y_equals_x = np.dot(triangle_vertices, np.array([[0, 1], [1, 0]]))
# Reflection through y-axis
reflection_y_axis = np.dot(triangle_vertices, np.array([[-1, 0], [0, 1]]))
# Plotting the triangle and its reflections
plt.plot(triangle_vertices[:, 0], triangle_vertices[:, 1], 'r-', label='Triangle')
plt.plot(reflection_y_equals_x[:, 0], reflection_y_equals_x[:, 1], 'g-',
label='Reflection (y = x)')
plt.plot(reflection_y_axis[:, 0], reflection_y_axis[:, 1], 'b-', label='Reflection (y-
axis)')
# Plotting the mirror lines
plt.axhline(0, color='k', linestyle=':', label='Mirror Line (y-axis)')
plt.plot(np.array([0, 6]), np.array([0, 6]), 'm:', label='Mirror Line (y = x)')
```

```
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Triangle and Its Reflections')
plt.legend()
plt.grid(True)
plt.show()
```

OUTPUT: