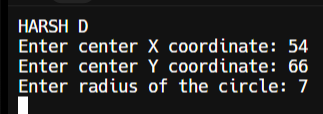
PRACTICAL – 3

PROGRAM -1

AIM- WAPP for Bresenham Circle Drawing Algorithm

CODE-

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  def bresenham\_circle(center\_x, center\_y, radius):  x = radius  y = 0  d = 3 - 2 \* radius  points = []  while x >= y:  # Add points in all octants  points.append((center\_x + x, center\_y + y))  points.append((center\_x + y, center\_y + x))  points.append((center\_x - y, center\_y + x))  points.append((center\_x - x, center\_y + y))  points.append((center\_x - x, center\_y - y))  points.append((center\_x - y, center\_y - x))  points.append((center\_x + y, center\_y - x))  points.append((center\_x + x, center\_y - y))  # Mid-point inside or on the perimeter  if d < 0:  d = d + 4 \* y + 6  else:  d = d + 4 \* (y - x) + 10  x -= 1  y += 1  return points  # Get center coordinates and radius from user  center\_x = int(input("Enter center X coordinate: "))  center\_y = int(input("Enter center Y coordinate: "))  radius = int(input("Enter radius of the circle: "))  # Generate points for the circle  circle\_points = bresenham\_circle(center\_x, center\_y, radius)  # Plot the circle  x\_values, y\_values = zip(\*circle\_points)  plt.scatter(x\_values, y\_values)  plt.gca().set\_aspect('equal', adjustable='box')  plt.title('Bresenham Circle Algorithm')  plt.xlabel('X')  plt.ylabel('Y')  plt.grid(True)  plt.show() |



OUTPUT-

