NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY (NSUT)

COMPUTER GRAPHICS
PRACTICAL FILE

INFORMATION TECHNOLOGY

(NETWORK AND INFORMATION SECURITY)

SEMESTER 3 (2022-23)

SUBMITTED TO: Mrs DEEPIKA KUKREJA

SUBMITTED BY: SAUMYA (Roll No.- 2022UIN3369)

INDEX

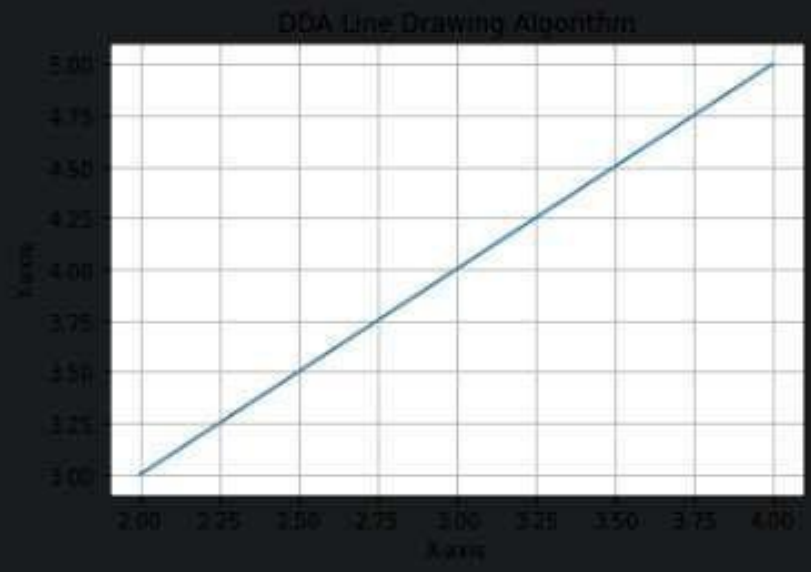
Q1. Program to draw a line using DDA algorithm.

CODE:

In [1]:
```
import matplotlib.pyplot as plt

def dda_line(x1, y1, x2, y2):
    dx = x2 - x1
    dy = y2 - y1
    steps = max(abs(dx), abs(dy))

    x_increment = dx / steps
    y_increment = dy / steps

    x, y = x1, y1
    points = [(x, y)]

    for _ in range(steps):
        x += x_increment
        y += y_increment
        points.append((int(x), int(y)))

    return points

def main():
    while True:
        print("DDA Line Drawing Algorithm Menu:")
        print("1. Draw a Line")
        print("2. Quit")
        choice = input("Enter your choice: ")

        if choice == "1":
            x1 = int(input("Enter x1: "))
            y1 = int(input("Enter y1: "))
            x2 = int(input("Enter x2: "))
            y2 = int(input("Enter y2: "))

            points = dda_line(x1, y1, x2, y2)

            x_values, y_values = zip(*points)
            plt.plot(x_values, y_values)
            plt.xlabel('X-axis')
            plt.ylabel('Y-axis')
            plt.title('DDA Line Drawing Algorithm')
            plt.grid()
            plt.show()

        elif choice == "2":
            print("Exiting the program.")
            break

        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

OUTPUT:

```
DDA Line Drawing Algorithm Menu:
1. Draw a Line
2. Quit
Enter your choice: 1
Enter x1: 2
Enter y1: 3
Enter x2: 4
Enter y2: 5
```




```
DDA Line Drawing Algorithm Menu:
1. Draw a Line
2. Quit
Enter your choice: 2
Exiting the program.
```

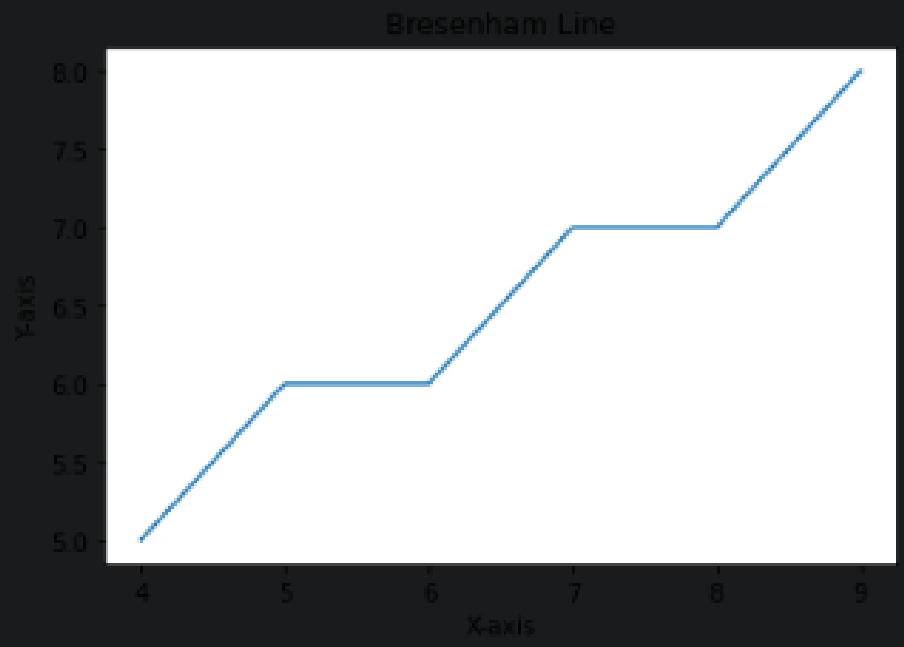
Q2. Program to draw a line using Bresenham algorithm.

CODE:

In [2]:
```
import matplotlib.pyplot as plt

def draw_line_bresenham(x1, y1, x2, y2):
    dx = abs(x2 - x1)
    dy = abs(y2 - y1)
    slope = dy > dx

    if slope:
        x1, y1 = y1, x1
        x2, y2 = y2, x2

    if x1 > x2:
        x1, x2 = x2, x1
        y1, y2 = y2, y1

    dx = abs(x2 - x1)
    dy = abs(y2 - y1)

    p = 2 * dy - dx

    y = y1

    points = [(x1, y1)]

    for x in range(x1, x2 + 1):
        if slope:
            points.append((y, x))
        else:
            points.append((x, y))

        if p >= 0:
            if y1 < y2:
                y += 1
            else:
                y -= 1
            p -= 2 * dx
        if y1 < y2:
            p += 2 * dy

    return points

def main():
    while True:
        print("Menu:")
        print("1. Draw a line using Bresenham algorithm")
        print("2. Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            x1 = int(input("Enter x1: "))
            y1 = int(input("Enter y1: "))
            x2 = int(input("Enter x2: "))
            y2 = int(input("Enter y2: "))

            points = draw_line_bresenham(x1, y1, x2, y2)

            x_values, y_values = zip(*points)

            plt.plot(x_values, y_values)
            plt.title('Bresenham Line')
            plt.xlabel('X-axis')
            plt.ylabel('Y-axis')
            plt.show()

        elif choice == '2':
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

OUTPUT:

```
Menu:
1. Draw a line using Bresenham algorithm
2. Exit
Enter your choice: 1
Enter x1: 4
Enter y1: 5
Enter x2: 9
Enter y2: 8
```




```
Menu:
1. Draw a line using Bresenham algorithm
2. Exit
Enter your choice: 2
```

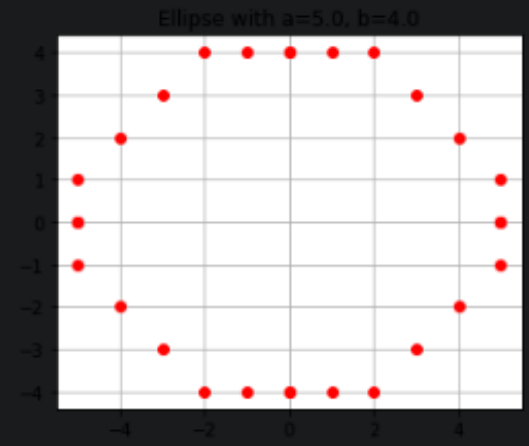
Q3. Program to draw an ellipse using midpoint algorithm.

CODE:

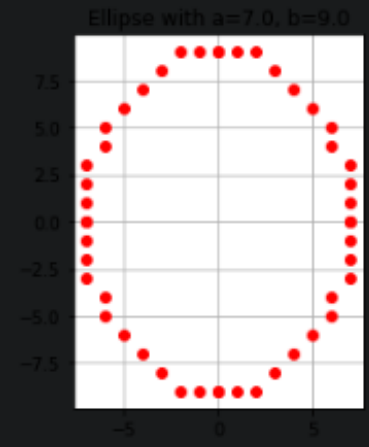
In [6]:
```
import matplotlib.pyplot as plt

def midpoint_ellipse(a, b):
    x, y = 0, b
    a_squared = a * a
    b_squared = b * b
    dx = 2 * b_squared * x
    dy = 2 * a_squared * y
    d1 = b_squared - (a_squared * b) + (0.25 * a_squared)
    while dx < dy:
        plt.plot(x, y, 'ro')
        plt.plot(-x, y, 'ro')
        plt.plot(x, -y, 'ro')
        plt.plot(-x, -y, 'ro')
        x += 1
        dx += 2 * b_squared
        if d1 < 0:
            d1 += b_squared * (2 * x + 1)
        else:
            y -= 1
            dy -= 2 * a_squared
            d1 += b_squared * (2 * x + 1) - a_squared * (2 * y - 1)
    d2 = b_squared * (x + 0.5) * (x + 0.5) + a_squared * (y - 1) * (y - 1) - a_squared * b_squared
    while y >= 0:
        plt.plot(x, y, 'ro')
        plt.plot(-x, y, 'ro')
        plt.plot(x, -y, 'ro')
        plt.plot(-x, -y, 'ro')
        y -= 1
        dy -= 2 * a_squared
        if d2 > 0:
            d2 += a_squared * (1 - 2 * y)
        else:
            x += 1
            dx += 2 * b_squared
            d2 += b_squared * (2 * x + 1) + a_squared * (1 - 2 * y)

def main():
    while True:
        print("Menu:")
        print("1. Draw Ellipse")
        print("2. Quit")
        choice = input("Enter your choice: ")
        if choice == '1':
            a = float(input("Enter the semi-major axis 'a': "))
            b = float(input("Enter the semi-minor axis 'b': "))
            plt.figure()
            plt.gca().set_aspect('equal', adjustable='box')
            midpoint_ellipse(a, b)
            plt.grid()
            plt.title(f"Ellipse with a={a}, b={b}")
            plt.show()
        elif choice == '2':
            print("Goodbye!")
            break
        else:
            print("Invalid choice. Please select a valid option.")

if __name__ == "__main__":
    main()
```

OUTPUT:

```
Menu:
1. Draw Ellipse
2. Quit
Enter your choice: 1
Enter the semi-major axis 'a': 5
Enter the semi-minor axis 'b': 4
```




```
Menu:
1. Draw Ellipse
2. Quit
Enter your choice: 1
Enter the semi-major axis 'a': 7
Enter the semi-minor axis 'b': 9
```




```
Menu:
1. Draw Ellipse
2. Quit
Enter your choice: 2
Goodbye!
```

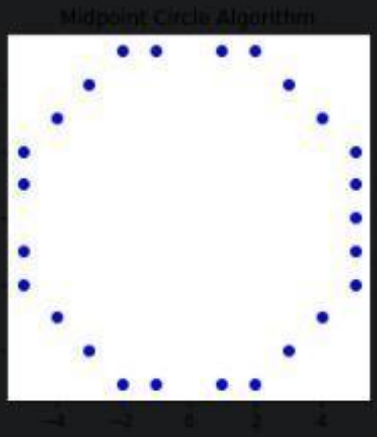
Q4. Program to draw a circle using midpoint algorithm.

CODE:

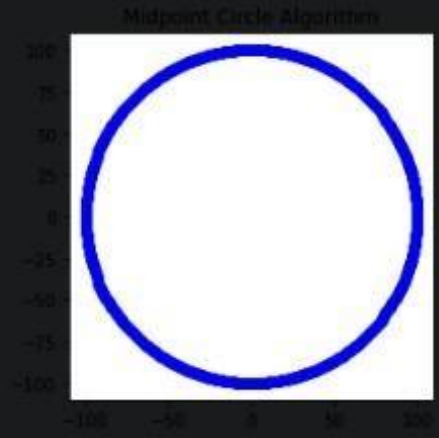
In [11]:
```
import matplotlib.pyplot as plt

def midpoint_circle(xc, yc, r):
    x = r
    y = 0
    p = 1 - r

    # Lists to store the points on the circle
    x_points = []
    y_points = []

    # Plot the initial point
    x_points.append(x)
    y_points.append(y)

    # Calculate points using the midpoint algorithm
    while x > y:
        y += 1
        if p <= 0:
            p = p + 2 * y + 1
        else:
            x -= 1
            p = p + 2 * y - 2 * x + 1
        if x < y:
            break
        x_points.extend([x, -x, x, -x, y, -y, y, -y])
        y_points.extend([y, y, -y, -y, x, x, -x, -x])

    return x_points, y_points

def plot_circle(xc, yc, r):
    x_points, y_points = midpoint_circle(xc, yc, r)

    # Plot the circle points using Matplotlib
    plt.plot(x_points, y_points, 'bo')
    plt.gca().set_aspect('equal', adjustable='box')
    plt.title('Midpoint Circle Algorithm')
    plt.show()

while True:
    print("Menu:")
    print("1. Draw a circle")
    print("2. Quit")
    choice = input("Enter your choice: ")

    if choice == "1":
        xc = int(input("Enter the x-coordinate of the center: "))
        yc = int(input("Enter the y-coordinate of the center: "))
        r = int(input("Enter the radius: "))
        plot_circle(xc, yc, r)
    elif choice == "2":
        break
    else:
        print("Invalid choice. Please select again.")
```

OUTPUT:

```
Menu:
1. Draw a circle
2. Quit
Enter your choice: 1
Enter the x-coordinate of the center: 0
Enter the y-coordinate of the center: 0
Enter the radius: 5
```




```
Menu:
1. Draw a circle
2. Quit
Enter your choice: 1
Enter the x-coordinate of the center: 0
Enter the y-coordinate of the center: 0
Enter the radius: 100
```




```
Menu:
1. Draw a circle
2. Quit
Enter your choice: 2
```

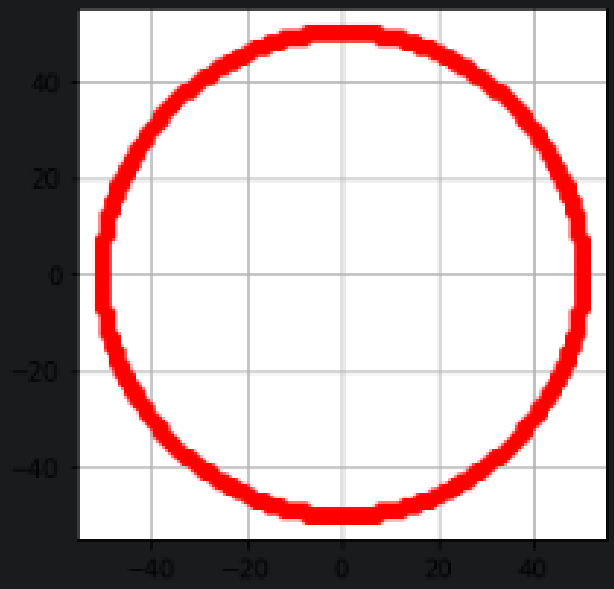
Q5. Program to draw a circle using Bresenham's algorithm.

CODE:

```
In [13]: import matplotlib.pyplot as plt

         def plot_circle_bresenham(center_x, center_y, radius):
             x = 0
             y = radius
             d = 3 - 2 * radius

             points = []

             while x <= y:
                 # Plot the eight-way symmetric points
                 points.append((x, y))
                 points.append((-x, y))
                 points.append((x, -y))
                 points.append((-x, -y))
                 points.append((y, x))
                 points.append((-y, x))
                 points.append((y, -x))
                 points.append((-y, -x))

                 if d < 0:
                     d += 4 * x + 6
                 else:
                     d += 4 * (x - y) + 10
                     y -= 1
                 x += 1

             # Translate the points to the given center
             points = [(x + center_x, y + center_y) for x, y in points]

             # Extract x and y coordinates for plotting
             x_values, y_values = zip(*points)

             # Create a Matplotlib figure and plot the circle
             plt.figure()
             plt.plot(x_values, y_values, 'ro')
             plt.gca().set_aspect('equal', adjustable='box')
             plt.grid()
             plt.show()

         while True:
             print("Menu:")
             print("1. Draw a circle using Bresenham's algorithm")
             print("2. Quit")
             choice = input("Enter your choice: ")

             if choice == '1':
                 center_x = int(input("Enter the x-coordinate of the center: "))
                 center_y = int(input("Enter the y-coordinate of the center: "))
                 radius = int(input("Enter the radius of the circle: "))
                 plot_circle_bresenham(center_x, center_y, radius)
             elif choice == '2':
                 break
             else:
                 print("Invalid choice. Please try again.")
```

OUTPUT:

```
Menu:
1. Draw a circle using Bresenham's algorithm
2. Quit
Enter your choice: 1
Enter the x-coordinate of the center: 0
Enter the y-coordinate of the center: 0
Enter the radius of the circle: 50
```




```
Menu:
1. Draw a circle using Bresenham's algorithm
2. Quit
Enter your choice: 2
```

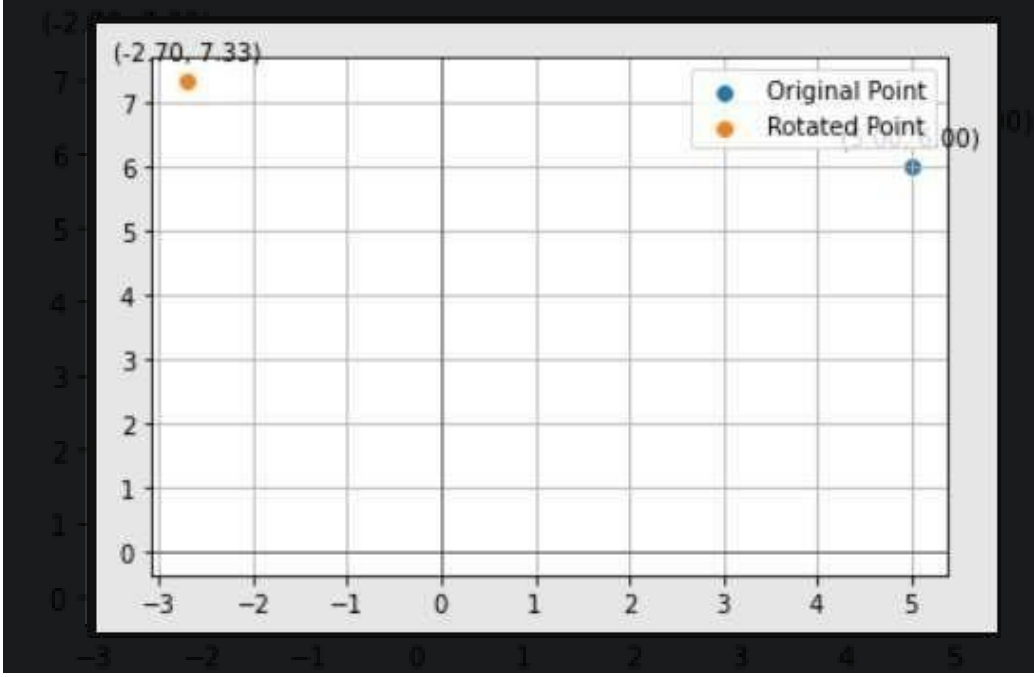
Q6. Program to rotate a given point about origin.

CODE:

In [14]:
```
import matplotlib.pyplot as plt
import math

def rotate_point(x, y, angle):
    # Convert the angle from degrees to radians
    angle_rad = math.radians(angle)

    # Perform the rotation
    x_rotated = x * math.cos(angle_rad) - y * math.sin(angle_rad)
    y_rotated = x * math.sin(angle_rad) + y * math.cos(angle_rad)

    return x_rotated, y_rotated

def plot_point(x, y, label):
    plt.scatter(x, y, label=label)
    plt.annotate(f'({x:.2f}, {y:.2f})', (x, y), textcoords="offset points", xytext=(0,10), ha='center')

def main():
    plt.figure()
    plt.grid(True)
    plt.axhline(0, color='black', lw=0.5)
    plt.axvline(0, color='black', lw=0.5)

    while True:
        print("Menu:")
        print("1. Rotate a point")
        print("2. Quit")
        choice = input("Enter your choice: ")

        if choice == '1':
            x = float(input("Enter the x-coordinate of the point: "))
            y = float(input("Enter the y-coordinate of the point: "))
            angle = float(input("Enter the rotation angle in degrees: "))

            x_rotated, y_rotated = rotate_point(x, y, angle)
            plot_point(x, y, "Original Point")
            plot_point(x_rotated, y_rotated, "Rotated Point")
            plt.legend()
            plt.show()

        elif choice == '2':
            break

        else:
            print("Invalid choice. Please enter a valid option.")

if __name__ == "__main__":
    main()
```

OUTPUT:

```
Menu:
1. Rotate a point
2. Quit
Enter your choice: 1
Enter the x-coordinate of the point: 5
Enter the y-coordinate of the point: 6
Enter the rotation angle in degrees: 60
```



Q7. Program to rotate a given square by an angle (in radians).

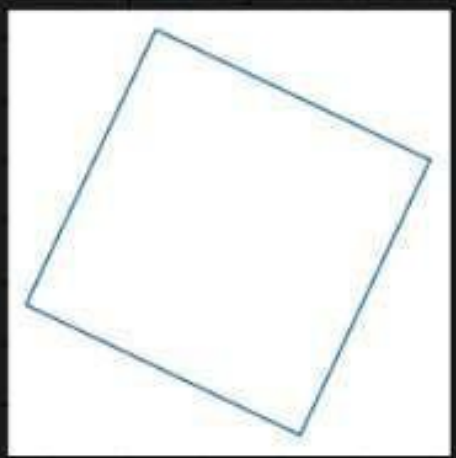
CODE:

```
In [18]: import matplotlib.pyplot as plt
         import numpy as np

         def draw_square():
             square = np.array([[0, 0], [1, 0], [1, 1], [0, 1], [0, 0]])
             plt.plot(square[:, 0], square[:, 1])
             plt.gca().set_aspect('equal', adjustable='box')
             plt.title("Original Square")
             plt.show()

         def rotate_square(angle):
             square = np.array([[0, 0], [1, 0], [1, 1], [0, 1], [0, 0]])
             rotation_matrix = np.array([[np.cos(angle), -np.sin(angle)],
                                         [np.sin(angle), np.cos(angle)]])
             rotated_square = np.dot(square, rotation_matrix)
             plt.plot(rotated_square[:, 0], rotated_square[:, 1])
             plt.gca().set_aspect('equal', adjustable='box')
             plt.title(f"Rotated Square (Angle: {angle} radians)")
             plt.show()

         while True:
             print("\nMenu:")
             print("1. Draw the original square")
             print("2. Rotate the square")
             print("3. Exit")

             choice = input("Enter your choice: ")

             if choice == '1':
                 draw_square()
             elif choice == '2':
                 angle = float(input("Enter the rotation angle in radians: "))
                 rotate_square(angle)
             elif choice == '3':
                 print("Exiting the program.")
                 break
             else:
                 print("Invalid choice. Please select a valid option.")
```

OUTPUT:

```
Menu:
1. Draw the original square
2. Rotate the square
3. Exit
Enter your choice: 1
```

```
Menu:
1. Draw the original square
2. Rotate the square
3. Exit
Enter your choice: 2
Enter the rotation angle in radians: 35
```



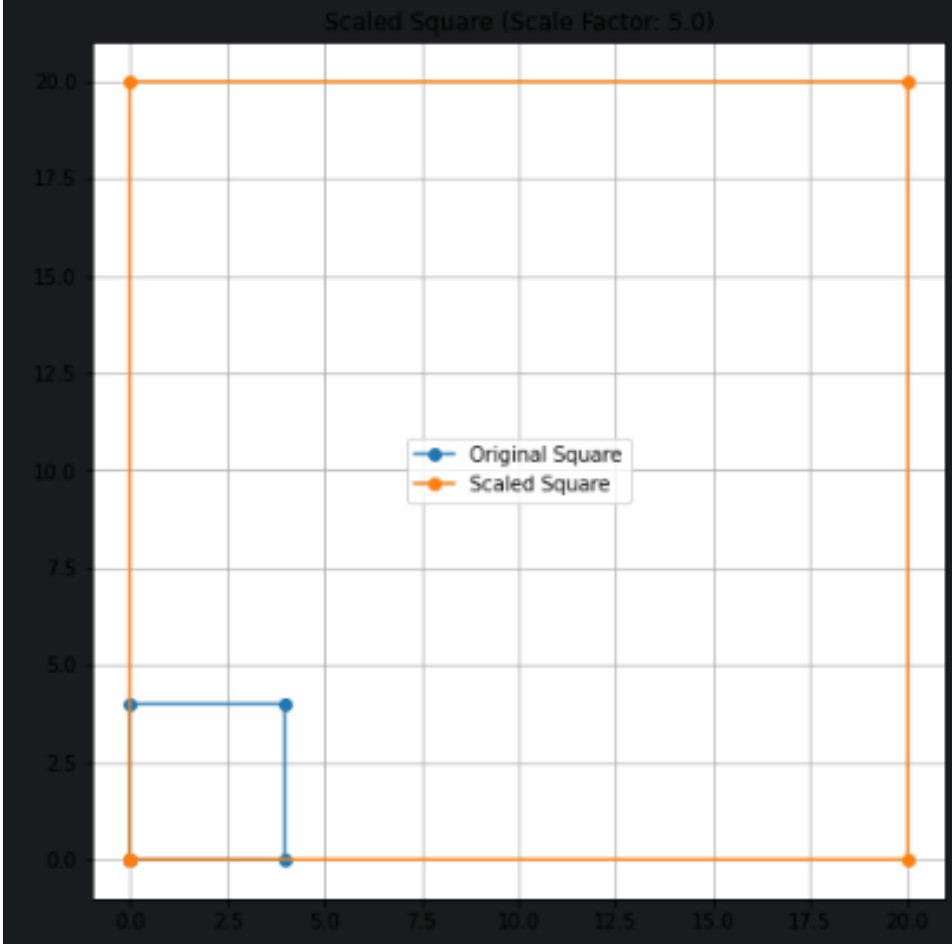
Q8. Program to scale a given square by given scaling factors.

CODE:

```
In [25]: import matplotlib.pyplot as plt
         import numpy as np

         def plot_square(side_length, scale_factor):
             # Create the original square
             original_square = np.array([[0, 0], [0, side_length], [side_length, side_length], [side_length, 0], [0, 0]])

             # Create the scaled square
             scaled_square = original_square * scale_factor

             # Create the plot
             plt.figure(figsize=(8, 8))
             plt.plot(original_square[:, 0], original_square[:, 1], label="Original Square", marker='o')
             plt.plot(scaled_square[:, 0], scaled_square[:, 1], label="Scaled Square", marker='o')
             plt.gca().set_aspect('equal', adjustable='box')
             plt.legend()
             plt.title(f"Scaled Square (Scale Factor: {scale_factor})")
             plt.grid()
             plt.show()

         def main():
             while True:
                 print("\nMenu:")
                 print("1. Scale Square")
                 print("2. Exit")
                 choice = input("Enter your choice: ")

                 if choice == '1':
                     side_length = float(input("Enter the side length of the square: "))
                     scale_factor = float(input("Enter the scale factor: "))
                     plot_square(side_length, scale_factor)
                 elif choice == '2':
                     print("Goodbye!")
                     break
                 else:
                     print("Invalid choice. Please select a valid option.")

         if __name__ == "__main__":
             main()
```

OUTPUT:

```
Menu:
1. Scale Square
2. Exit
Enter your choice: 1
Enter the side length of the square: 4
Enter the scale factor: 5
```




```
Menu:
1. Scale Square
2. Exit
Enter your choice: 2
Goodbye!
```

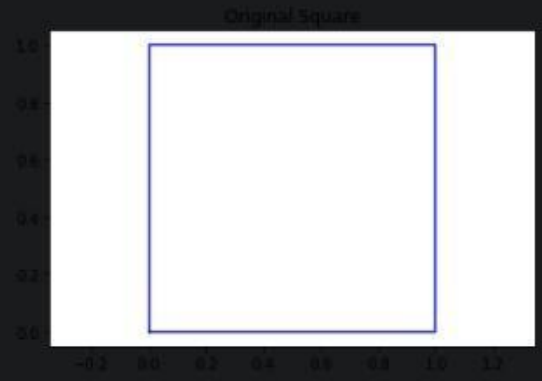
Q9. Program to translate a given square.

CODE:

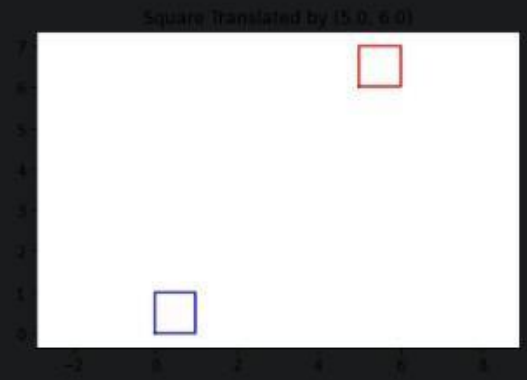
In [30]:
```
import matplotlib.pyplot as plt
import numpy as np

# Function to draw a square
def draw_square():
    square = np.array([[0, 0], [1, 0], [1, 1], [0, 1], [0, 0]])
    plt.plot(square[:, 0], square[:, 1], 'b')

# Function to translate the square
def translate_square(tx, ty):
    square = np.array([[0, 0], [1, 0], [1, 1], [0, 1], [0, 0]])
    translated_square = square + [tx, ty]
    return translated_square

# Main menu
while True:
    print("\nMenu:")
    print("1. Draw square")
    print("2. Translate square")
    print("3. Exit")

    choice = input("Enter your choice (1/2/3): ")

    if choice == '1':
        plt.figure()
        draw_square()
        plt.axis('equal')
        plt.title('Original Square')
        plt.show()

    elif choice == '2':
        try:
            tx = float(input("Enter translation in the x-direction: "))
            ty = float(input("Enter translation in the y-direction: "))
            translated_square = translate_square(tx, ty)
            plt.figure()
            draw_square()
            plt.plot(translated_square[:, 0], translated_square[:, 1], 'r')
            plt.axis('equal')
            plt.title(f'Square Translated by ({tx}, {ty})')
            plt.show()
        except ValueError:
            print("Invalid input. Please enter valid numbers for translation.")

    elif choice == '3':
        break

    else:
        print("Invalid choice. Please enter a valid option.")
```

OUTPUT:

```
Menu:
1. Draw square
2. Translate square
3. Exit
Enter your choice (1/2/3): 1
```




```
Menu:
1. Draw square
2. Translate square
3. Exit
Enter your choice (1/2/3): 2
Enter translation in the x-direction: 5
Enter translation in the y-direction: 6
```




```
Menu:
1. Draw square
2. Translate square
3. Exit
Enter your choice (1/2/3): 3
```

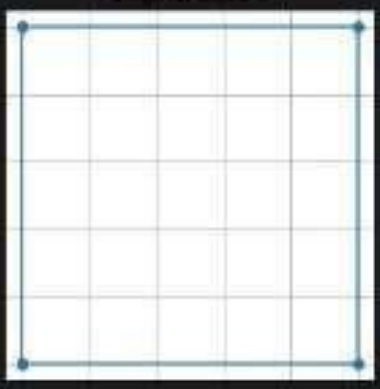
Q10. Program to reflect a given square about x-axis.

CODE:

```
In [36]: import matplotlib.pyplot as plt

         def draw_square(x, y):
             # Define the vertices of the square
             square = [(x, y), (x + 1, y), (x + 1, y + 1), (x, y + 1), (x, y)]

             # Extract x and y coordinates for plotting
             x_coords, y_coords = zip(*square)

             # Plot the square
             plt.plot(x_coords, y_coords, marker='o')

         def reflect_about_x_axis(square):
             # Reflect the square about the x-axis
             reflected_square = [(x, -y) for x, y in square]

             return reflected_square

         def main():
             while True:
                 print("Menu:")
                 print("1. Draw the original square")
                 print("2. Reflect the square about the x-axis")
                 print("3. Exit")

                 choice = input("Enter your choice (1/2/3): ")

                 if choice == '1':
                     # Draw the original square
                     plt.figure()
                     draw_square(0, 0)
                     plt.gca().set_aspect('equal', adjustable='box')
                     plt.xlabel("X-axis")
                     plt.ylabel("Y-axis")
                     plt.title("Original Square")
                     plt.grid()
                     plt.show()

                 elif choice == '2':
                     # Reflect the square about the x-axis and draw both squares
                     original_square = [(0, 0), (1, 0), (1, 1), (0, 1), (0, 0)]
                     reflected_square = reflect_about_x_axis(original_square)

                     plt.figure()
                     draw_square(0, 0)
                     draw_square(0, -1)  # Draw the reflected square below the original
                     plt.gca().set_aspect('equal', adjustable='box')
                     plt.xlabel("X-axis")
                     plt.ylabel("Y-axis")
                     plt.title("Original and Reflected Square")
                     plt.grid()
                     plt.show()

                 elif choice == '3':
                     print("Exiting program.")
                     break

                 else:
                     print("Invalid choice. Please enter 1, 2, or 3.")

         if __name__ == "__main__":
             main()
```

OUTPUT:

```
Menu:
1. Draw the original square
2. Reflect the square about the x-axis
3. Exit
Enter your choice (1/2/3): 1
```



```
Menu:
1. Draw the original square
2. Reflect the square about the x-axis
3. Exit
Enter your choice (1/2/3): 2
```

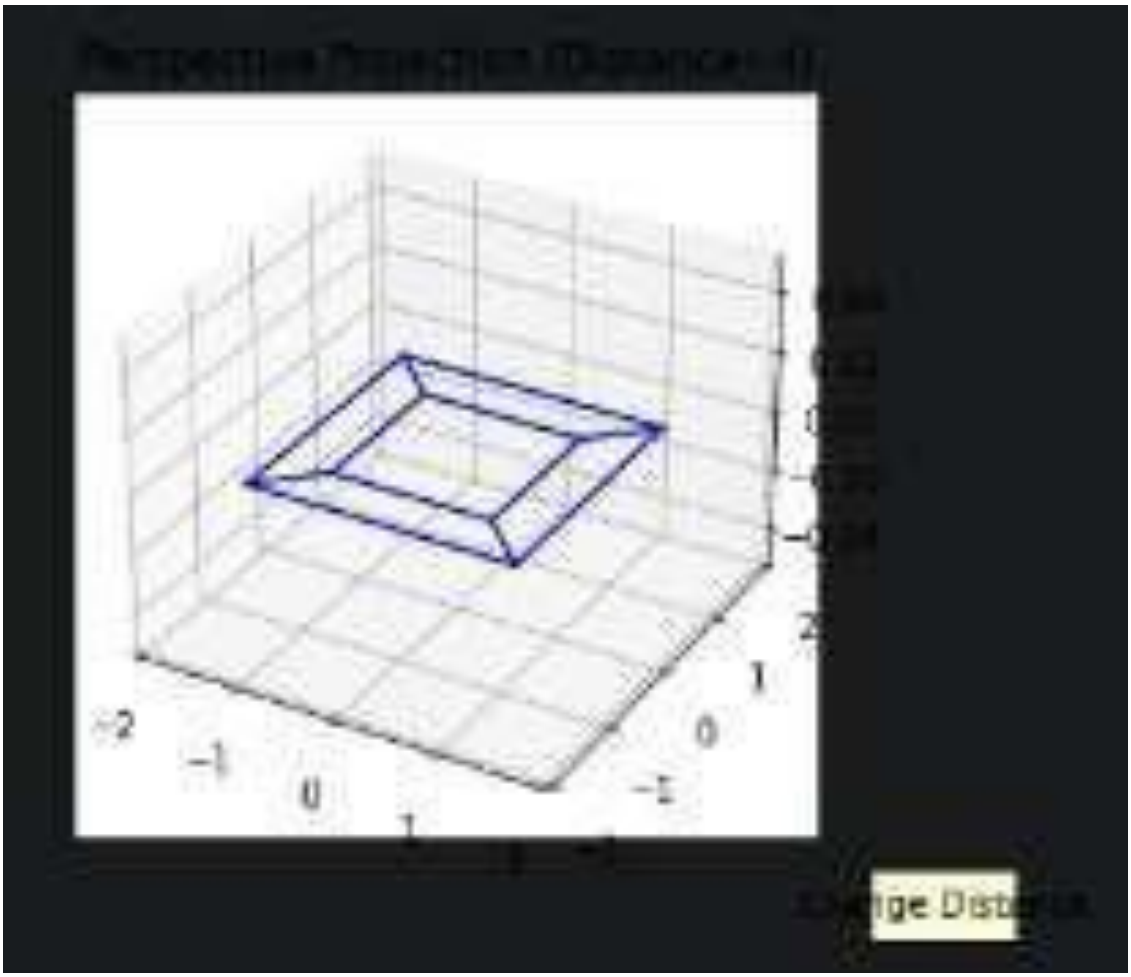
```
Menu:
1. Draw the original square
2. Reflect the square about the x-axis
3. Exit
Enter your choice (1/2/3): 3
Exiting program.
```

Q11. Program to show a 3D object in 2D using perspective transformation.

CODE:

```
In [42]: import numpy as np
         import matplotlib.pyplot as plt
         from mpl_toolkits.mplot3d import Axes3D
         from matplotlib.widgets import Button

         def perspective_projection(points, distance):
             return points / (1 + points[:, 2] / distance)[:, np.newaxis]

         def plot_3d_object():
             fig = plt.figure()
             ax = fig.add_subplot(111, projection='3d')

             # Define the 3D object (vertices and edges)
             vertices = np.array([[1, 1, 1],
                                  [1, -1, 1],
                                  [-1, -1, 1],
                                  [-1, 1, 1],
                                  [1, 1, -1],
                                  [1, -1, -1],
                                  [-1, -1, -1],
                                  [-1, 1, -1]])

             edges = [(0, 1), (1, 2), (2, 3), (3, 0),
                      (4, 5), (5, 6), (6, 7), (7, 4),
                      (0, 4), (1, 5), (2, 6), (3, 7)]

             # Define the distance for perspective projection
             distance = 4

             def draw_2d_projection(distance):
                 ax.cla()  # Clear the previous plot
                 ax.set_xlim(-2, 2)
                 ax.set_ylim(-2, 2)

                 # Perform perspective projection
                 projected_vertices = perspective_projection(vertices, distance)

                 # Draw the 2D projection
                 for edge in edges:
                     ax.plot(projected_vertices[edge, 0], projected_vertices[edge, 1], 'b-')

                 ax.set_title(f'Perspective Projection (Distance={distance})')
                 plt.draw()

             draw_2d_projection(distance)  # Initial plot

             # Create a button for changing the perspective distance
             ax_button = plt.axes([0.8, 0.02, 0.1, 0.075])
             button = Button(ax_button, 'Change Distance', color='lightgoldenrodyellow')

             def update_distance(event):
                 new_distance = 2  # Set your desired new distance
                 draw_2d_projection(new_distance)

             button.on_clicked(update_distance)

             plt.show()

         if __name__ == "__main__":
             plot_3d_object()
```

OUTPUT:



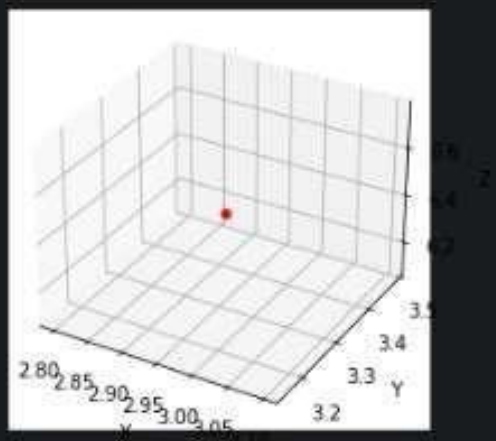
Q12. Program to rotate a point about an arbitrary axis in 3 dimensions.

CODE:

In [59]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Function to rotate a point in 3D
def rotate_point(point, axis, angle_degrees):
    angle_radians = np.radians(angle_degrees)
    u = axis / np.linalg.norm(axis)
    cos_theta = np.cos(angle_radians)
    sin_theta = np.sin(angle_radians)
    rotation_matrix = np.array([
        [cos_theta + u[0]**2 * (1 - cos_theta), u[0] * u[1] * (1 - cos_theta) - u[2] * sin_theta, u[0]
         * u[2] * (1 - cos_theta) + u[1] * sin_theta],
        [u[1] * u[0] * (1 - cos_theta) + u[2] * sin_theta, cos_theta + u[1]**2
         * (1 - cos_theta), u[1] * u[2] * (1 - cos_theta) - u[0] * sin_theta],
        [u[2] * u[0] * (1 - cos_theta) - u[1] * sin_theta, u[2] * u[1]
         * (1 - cos_theta) + u[0] * sin_theta, cos_theta + u[2]**2 * (1 - cos_theta)]
    ])
    rotated_point = np.dot(rotation_matrix, point)
    return rotated_point

# Function to display the 3D plot
def plot_point(point):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(point[0], point[1], point[2], c='r', marker='o')
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')
    plt.show()

# Main menu loop
while True:
    print("Menu:")
    print("1. Rotate a point")
    print("2. Quit")
    choice = input("Enter your choice: ")

    if choice == '1':
        x = float(input("Enter the X coordinate of the point: "))
        y = float(input("Enter the Y coordinate of the point: "))
        z = float(input("Enter the Z coordinate of the point: "))
        point = np.array([x, y, z])

        ax = float(input("Enter the X coordinate of the rotation axis: "))
        ay = float(input("Enter the Y coordinate of the rotation axis: "))
        az = float(input("Enter the Z coordinate of the rotation axis: "))
        axis = np.array([ax, ay, az])

        angle_degrees = float(input("Enter the rotation angle in degrees: "))

        rotated_point = rotate_point(point, axis, angle_degrees)
        print("Rotated point:", rotated_point)
        plot_point(rotated_point)

    elif choice == '2':
        print("Exiting the program.")
        break

    else:
        print("Invalid choice. Please try again.")
```

OUTPUT:

```
Menu:
1. Rotate a point
2. Quit
Enter your choice: 1
Enter the X coordinate of the point: 3
Enter the Y coordinate of the point: 4
Enter the Z coordinate of the point: 6
Enter the X coordinate of the rotation axis: 5
Enter the Y coordinate of the rotation axis: 4
Enter the Z coordinate of the rotation axis: 7
Enter the rotation angle in degrees: 35
Rotated point: [2.949054   3.31921694 6.42540889]
```




```
Menu:
1. Rotate a point
2. Quit
Enter your choice: 2
Exiting the program.
```