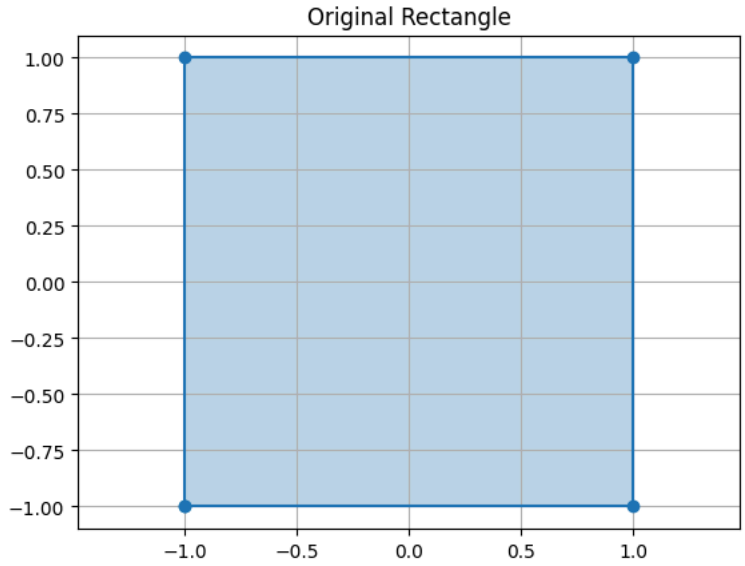
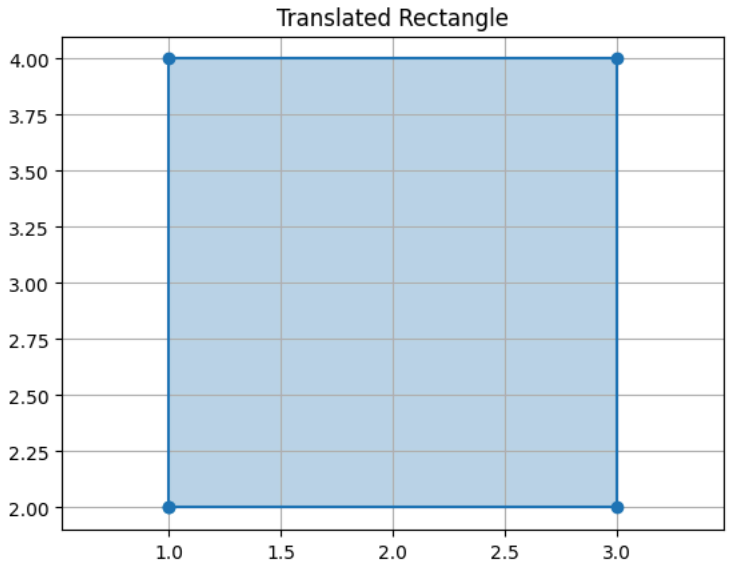
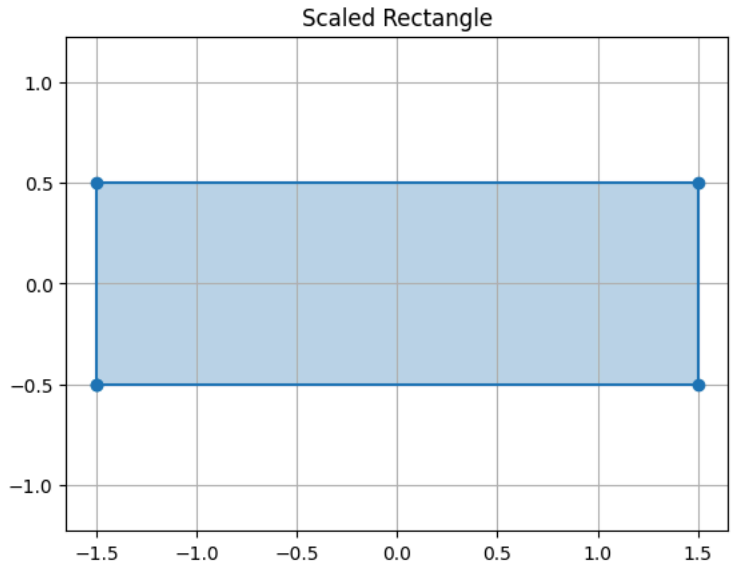
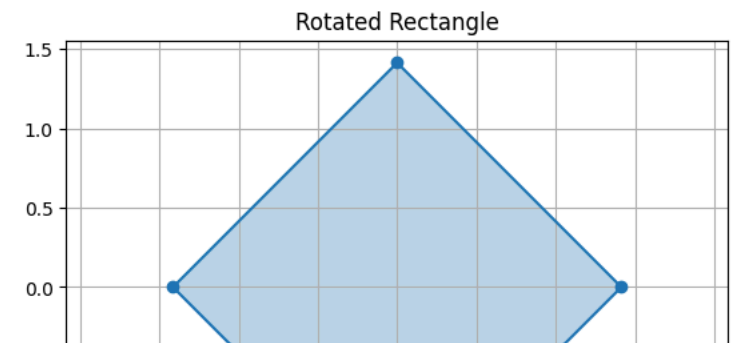
```
import numpy as np
import matplotlib.pyplot as plt

# Define 2D object: A rectangle (for demonstration purposes)
rectangle = np.array([[-1, -1], [1, -1], [1, 1], [-1, 1], [-1, -1]])

# Function for plotting the object
def plot_object(object_points, title='2D Object'):
    plt.plot(object_points[:, 0], object_points[:, 1], marker='o')
    plt.fill(object_points[:, 0], object_points[:, 1], alpha=0.3)
    plt.title(title)
    plt.axis('equal')
    plt.grid(True)
    plt.show()

# 1. Translation
def translate(object_points, tx, ty):
    translation_matrix = np.array([[1, 0, tx],
                                   [0, 1, ty],
                                   [0, 0, 1]])
    ones = np.ones((object_points.shape[0], 1))
    object_points_homogeneous = np.hstack((object_points, ones))
    translated_points = (translation_matrix @ object_points_homogeneous.T).T
    return translated_points[:, :2]

# 2. Scaling
def scale(object_points, sx, sy):
    scaling_matrix = np.array([[sx, 0, 0],
                               [0, sy, 0],
                               [0, 0, 1]])
    ones = np.ones((object_points.shape[0], 1))
    object_points_homogeneous = np.hstack((object_points, ones))
    scaled_points = (scaling_matrix @ object_points_homogeneous.T).T
    return scaled_points[:, :2]

# 3. Rotation
def rotate(object_points, angle_deg):
    angle_rad = np.deg2rad(angle_deg)
    rotation_matrix = np.array([[np.cos(angle_rad), -np.sin(angle_rad), 0],
                                [np.sin(angle_rad), np.cos(angle_rad), 0],
                                [0, 0, 1]])
    ones = np.ones((object_points.shape[0], 1))
    object_points_homogeneous = np.hstack((object_points, ones))
    rotated_points = (rotation_matrix @ object_points_homogeneous.T).T
    return rotated_points[:, :2]

# 4. Reflection (about X-axis)
def reflect_x(object_points):
    reflection_matrix = np.array([[-1, 0, 0],
                                  [0, 1, 0],
                                  [0, 0, 1]])
    ones = np.ones((object_points.shape[0], 1))
    object_points_homogeneous = np.hstack((object_points, ones))
    reflected_points = (reflection_matrix @ object_points_homogeneous.T).T
    return reflected_points[:, :2]

# 5. Shearing (X-shear)
def shear_x(object_points, shear_factor):
    shear_matrix = np.array([[1, shear_factor, 0],
                             [0, 1, 0],
                             [0, 0, 1]])
    ones = np.ones((object_points.shape[0], 1))
    object_points_homogeneous = np.hstack((object_points, ones))
    sheared_points = (shear_matrix @ object_points_homogeneous.T).T
    return sheared_points[:, :2]

# Plot the original object
plot_object(rectangle, title="Original Rectangle")

# 1. Apply Translation
translated_rectangle = translate(rectangle, 2, 3)
plot_object(translated_rectangle, title="Translated Rectangle")

# 2. Apply Scaling
scaled_rectangle = scale(rectangle, 1.5, 0.5)
plot_object(scaled_rectangle, title="Scaled Rectangle")

# 3. Apply Rotation
rotated_rectangle = rotate(rectangle, 45)
plot_object(rotated_rectangle, title="Rotated Rectangle")
```

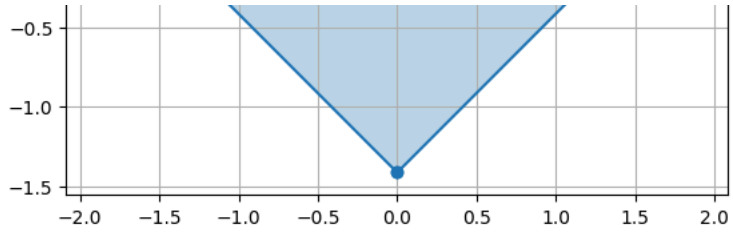
```
# 4. Apply Reflection
reflected_rectangle = reflect_x(rectangle)
plot_object(reflected_rectangle, title="Reflected Rectangle")

# 5. Apply Shearing
sheared_rectangle = shear_x(rectangle, 0.5)
plot_object(sheared_rectangle, title="Sheared Rectangle")

# Composite Transformation: Translation + Scaling
composite_rectangle = scale(translate(rectangle, 2, 3), 1.5, 0.5)
plot_object(composite_rectangle, title="Composite: Translation + Scaling")

# Composite Transformation: Scaling + Rotation
composite_rectangle_2 = rotate(scale(rectangle, 1.5, 0.5), 45)
plot_object(composite_rectangle_2, title="Composite: Scaling + Rotation")
```

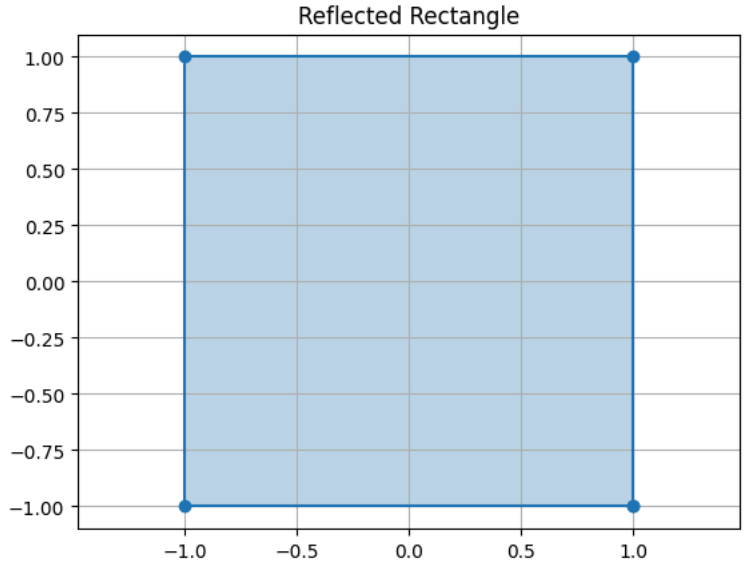

Original Rectangle
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.0
−0.5
0.0
0.5
1.0

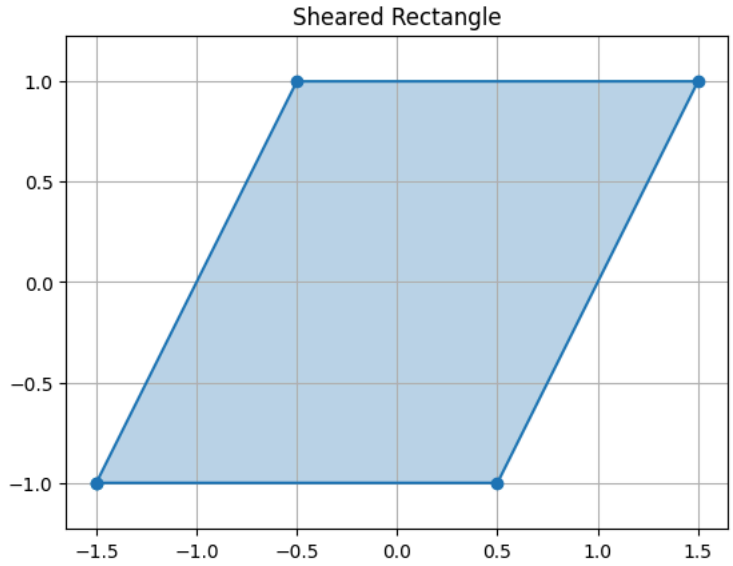

Translated Rectangle
4.00
3.75
3.50
3.25
3.00
2.75
2.50
2.25
2.00
1.0
1.5
2.0
2.5
3.0

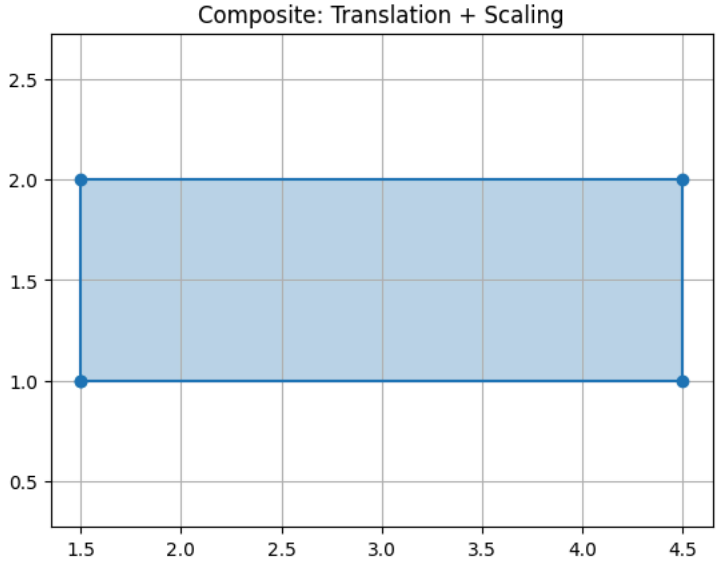

Scaled Rectangle
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Rotated Rectangle
1.5
1.0
0.5
0.0

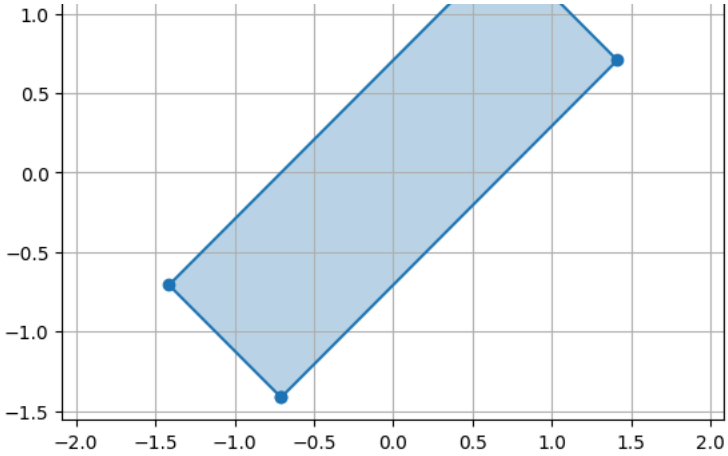

−0.5
−1.0
−1.5
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0


Reflected Rectangle
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.0 −0.5 0.0 0.5 1.0


Sheared Rectangle
1.0
0.5
0.0
−0.5
−1.0
−1.5 −1.0 −0.5 0.0 0.5 1.0 1.5


Composite: Translation + Scaling
2.5
2.0
1.5
1.0
0.5
1.5 2.0 2.5 3.0 3.5 4.0 4.5


Composite: Scaling + Rotation
1.5


1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the image
image = cv2.imread('/content/Logo.jpg')
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Translation Function
def translate_image(image, tx, ty):
    height, width = image.shape[:2]
    translation_matrix = np.float32([[1, 0, tx], [0, 1, ty]])
    translated_image = cv2.warpAffine(image, translation_matrix, (width, height))
    return translated_image

# Translate the image
translated_image = translate_image(image_rgb, 50, 50)

# Display the translated image
plt.imshow(translated_image)
plt.title("Translated Image")
plt.axis("off")
plt.show()
```


```
# Reflection Function
def reflect_image(image, mode="horizontal"):
    if mode == "horizontal":
        reflected_image = cv2.flip(image, 1)  # Flip horizontally
    elif mode == "vertical":
        reflected_image = cv2.flip(image, 0)  # Flip vertically
    else:
        reflected_image = cv2.flip(image, -1)  # Flip both axes
    return reflected_image

# Reflect the image horizontally
reflected_image = reflect_image(image_rgb, mode="horizontal")

# Display the reflected image
plt.imshow(reflected_image)
plt.title("Reflected Image")
plt.axis("off")
plt.show()
```