```
# Program 1
# From Internal 2
# Question 4
# Apply 3D homogenous transformation to scale an object with respect to to a
# pivot point. For the triangle A (3, 2, 2) B (6, 2, 2), C (6, 6, 2), rotate it
# anti-clockwise direction by 90degree about z axis keeping A (3, 2,2) fixed.
# Give the matrices for the original and rotated triangle.

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d import Poly3DCollection

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

    # Plot the triangle using Poly3DCollection
    triangle = Poly3DCollection([vertices], color='blue', edgecolors='r')
    ax.add_collection3d(triangle)

    # Set the limits of the plot
    ax.set_xlim([0, 10])
    ax.set_ylim([0, 10])
    ax.set_zlim([0, 10])

    # Set labels for axes
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')

    # Set title
    ax.set_title(title)

    # Display the plot
    plt.show()

def rotate_3d_z_axis(vertices, angle_degrees):
    angle_radians = np.radians(angle_degrees)
    rotation_matrix = np.array([[np.cos(angle_radians), -np.sin(angle_radians), -
3*np.cos(angle_radians) +2*np.sin(angle_radians)+3],
                                    [np.sin(angle_radians),
np.cos(angle_radians), -3*np.sin(angle_radians) -2*np.cos(angle_radians)+2],
                                    [0, 0, 1]])
    rotated_vertices = np.dot(vertices, rotation_matrix)
    return rotated_vertices
```

```
def main():
    vertices = np.array([[3, 2, 2],
                         [6, 2, 2],
                         [6, 6, 2],])
    while True:
        print("\nChoose a transformation:")
        print("1. Original")
        print("2. Rotate 90 degree anticlockwise z axis")
        print("3. Exit")

        choice = input("Enter your choice (1-3): ")

        if choice == '1':
            plot_3d_object(vertices, 'Original Triangle')
        elif choice == '2':
            angle = input("Enter rotation angle in degrees keeping A(3,2,2) fixed
: ")
            angle_degrees = float(angle)
            vertices = rotate_3d_z_axis(vertices,angle_degrees)
            print(vertices)
            plot_3d_object(vertices, 'Transformed Triangle by ')
        elif choice == '3':
            break
        else:
            print("Invalid choice. Please enter a valid option.")

    print("Exiting...")

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

Output

Choose a transformation:

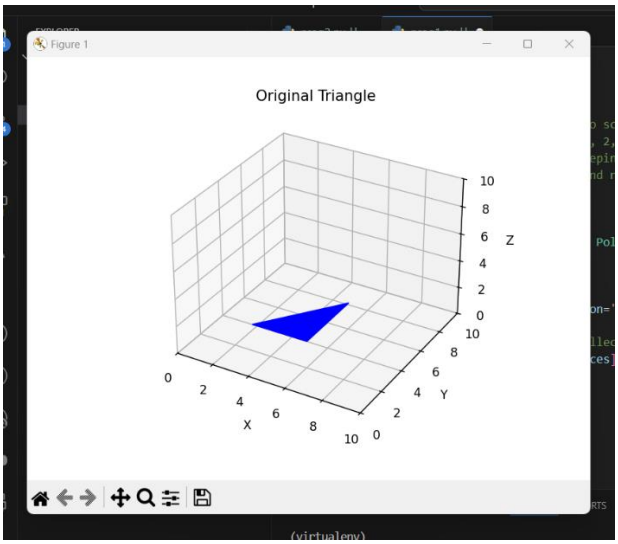
1. Original

2. Rotate 90 degree anticlockwise z axis

3. Exit

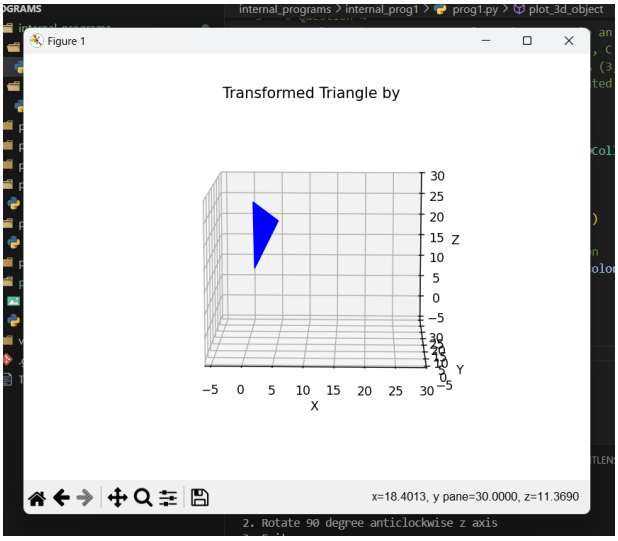
Enter your choice (1-3): 1

Original Triangle



Enter your choice (1-3): 2

Rotated Triangle by 90 degrees



Enter your choice (1-3): 3

Exiting...

```
# Program 2
# From Internal 2
# Question 7
# Apply appropriate algorithm to draw a 3D unit cube at origin, rotate it by 45
# degrees on z-axis. Translate the original polygon (without rotation) by 10
# Units on x axis and scale by a factor of (2,2, 3). Give the matrices
# for the original and transformed 3d cube.

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d import Poly3DCollection

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

    # Plot the 3D object
    for i in range(len(vertices)):
        ax.scatter(vertices[i, 0], vertices[i, 1], vertices[i, 2], color='b')
        ax.text(vertices[i, 0], vertices[i, 1], vertices[i, 2], f'P{i}',
ha='right')

    # Connect vertices to form edges of the object
    edges = [[0, 1, 2, 3, 0],
             [4, 5, 6, 7, 4],
             [0, 4], [1, 5], [2, 6], [3, 7]]
    for edge in edges:
        ax.plot(vertices[edge, 0], vertices[edge, 1], vertices[edge, 2], 'b-')

    plt.show()

# Translated
def translate_3d_object(vertices):
    translated_vertices = vertices + np.array([10, 0, 0])
    print(translated_vertices)
    return translated_vertices

# Rotated by 45
def rotate_3d_z_axis(vertices,angle_degrees):
```

```
    angle_radians = np.radians(angle_degrees)
    rotation_matrix = np.array([[np.cos(angle_radians), -np.sin(angle_radians),
0],
                                    [np.sin(angle_radians),
np.cos(angle_radians),0],
                                    [0, 0, 1]])
    rotated_vertices = np.dot(vertices, rotation_matrix)
    return rotated_vertices

# Scaled by (2,2,3)
def scale_3d_object(vertices):
    scaled_vertices = vertices * np.array([2, 2, 3])
    return scaled_vertices
def main():
    vertices = np.array([[0, 0, 0],
    [1, 0, 0],
    [1, 1, 0],
    [0, 1, 0],
    [0, 0, 1],
    [1, 0, 1],
    [1, 1, 1],
    [0, 1, 1]])
    while True:
        print("\nChoose a transformation:")
        print("1. Original")
        print("2. Rotate 45 degree anticlockwise z axis")
        print("3. Translate and Scale")
        print("4. Exit")

        choice = input("Enter your choice (1-4): ")

        if choice == '1':
            plot_3d_object(vertices, 'Original Cube')
        elif choice == '2':
            angle = "45"
            angle_degrees = float(angle)
            vertices = rotate_3d_z_axis(vertices,angle_degrees)

            plot_3d_object(vertices, 'Transformed Triangle by '+angle +"
degrees")
        elif choice == '3':
            vertices = translate_3d_object(vertices)
            plot_3d_object(vertices, 'Translated Cube')

            vertices = scale_3d_object(vertices)
```

```
            plot_3d_object(vertices, 'Scaled Cube')

        elif choice == '4':
            break
        else:
            print("Invalid choice. Please enter a valid option.")

    print("Exiting...")

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

Output

Choose a transformation:

1. Original

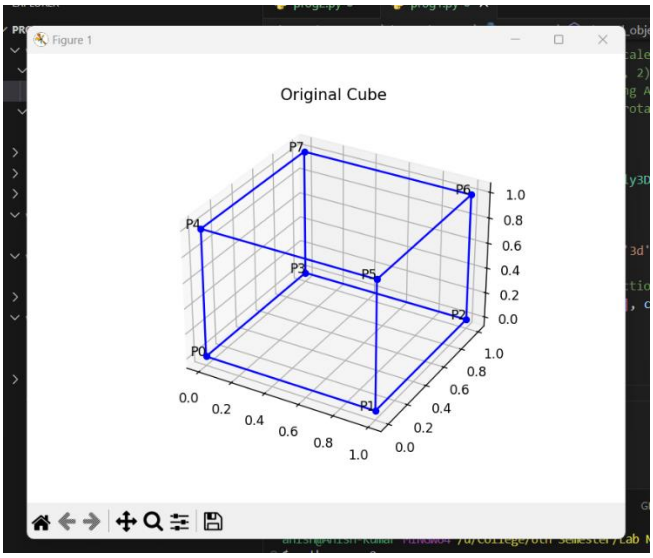
2. Rotate 45 degree anticlockwise z axis

3. Translate and Scale

4. Exit

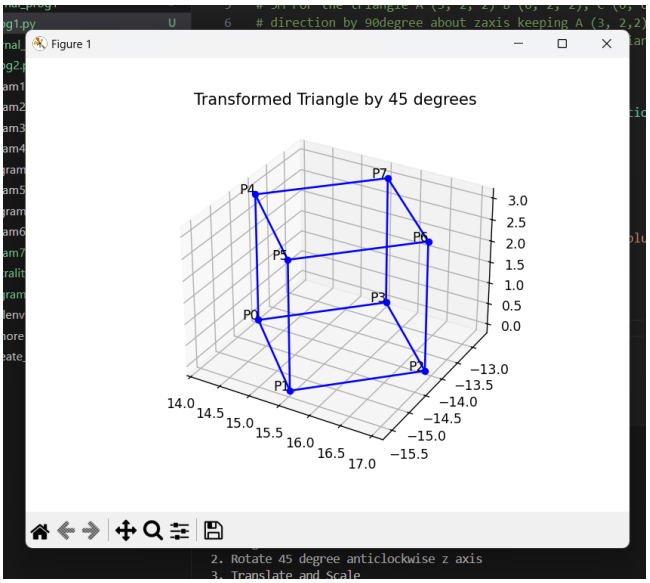
Enter your choice (1-4): 1

Original Cube



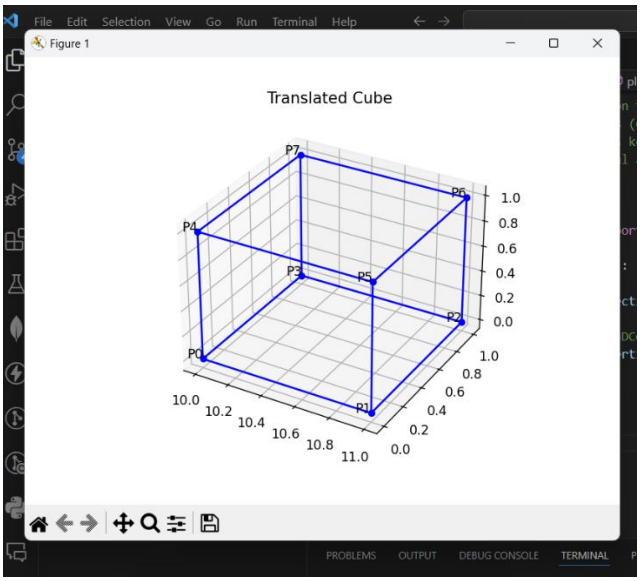
Enter your choice (1-4): 2

Rotated Cube

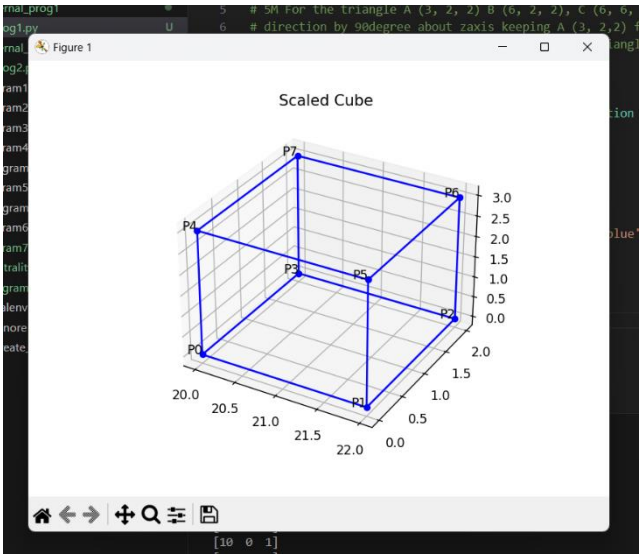


Enter your choice (1-4): 3

Translated Cube



Scaled Cube



Enter your choice (1-4): 4

Exiting...