

## Modul 9

### PRAKTIKUM

#### 1. Praktikum 1

##### Code Program

```
# Import library yang diperlukan
import matplotlib.pyplot as plt
import numpy as np

# Fungsi untuk menggambar belah ketupat sebelum dan sesudah shear
def plot_belah_ketupat(original_points, sheared_points):
    # Gambar belah ketupat sebelum shear
    plt.figure(figsize=(6, 6))
    plt.subplot(1, 2, 1)
    plt.plot(*zip(*original_points, original_points[0]), marker='o')
    plt.title('Belah Ketupat Asli')
    plt.grid(True)

    # Gambar belah ketupat setelah shear
    plt.subplot(1, 2, 2)
    plt.plot(*zip(*sheared_points, sheared_points[0]), marker='o')
    plt.title('Belah Ketupat Setelah Shear')
    plt.grid(True)

    plt.show()

# Fungsi untuk melakukan shear pada objek belah ketupat
def shear_belah_ketupat(points, shear_x, shear_y):
    # Matriks transformasi shear
    shear_matrix = np.array([[1, shear_x],
                           [shear_y, 1]])

    # Mengalikan setiap titik dengan matriks shear
    sheared_points = []
    for point in points:
        new_point = np.dot(shear_matrix, point)
        sheared_points.append(new_point)

    return sheared_points

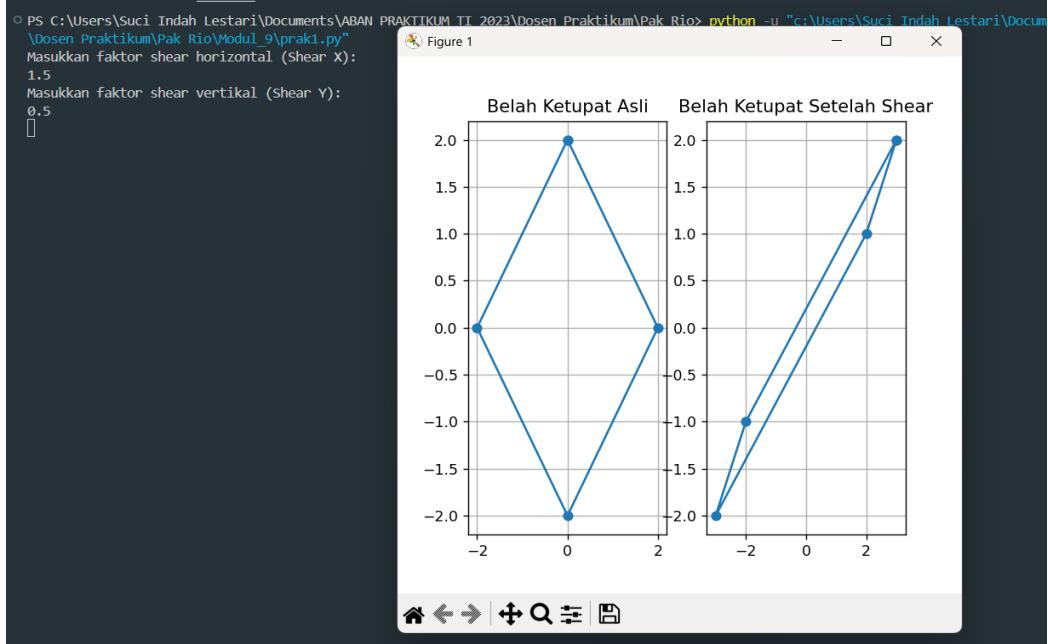
# Input dinamis dari user
print("Masukkan faktor shear horizontal (Shear X): ")
shear_x = float(input()) # Shear X (horizontal)
print("Masukkan faktor shear vertikal (Shear Y): ")
shear_y = float(input()) # Shear Y (vertikal)

# Koordinat asli belah ketupat
belah_ketupat_points = np.array([[0, 2], [2, 0], [0, -2], [-2, 0]])

# Lakukan transformasi shear
sheared_points = shear_belah_ketupat(belah_ketupat_points, shear_x, shear_y)

# Plot hasilnya
plot_belah_ketupat(belah_ketupat_points, sheared_points)
```

## Hasil RUN



## 2. Praktikum 2

### Code Program

```
# Import library yang diperlukan
import matplotlib.pyplot as plt
import numpy as np

# Fungsi untuk menggambar segitiga sebelum dan sesudah shear
def plot_segitiga(original_points, sheared_points):
    # Gambar segitiga sebelum shear
    plt.figure(figsize=(6, 6))
    plt.subplot(1, 2, 1)
    # Tambahkan titik pertama di akhir untuk menutup segitiga
    original_closed = np.vstack([original_points, original_points[0]])
    plt.plot(original_closed[:, 0], original_closed[:, 1], marker='o')
    plt.title('Segitiga Asli')
    plt.grid(True)

    # Gambar segitiga setelah shear
    plt.subplot(1, 2, 2)
    # Tambahkan titik pertama di akhir untuk menutup segitiga
    sheared_closed = np.vstack([sheared_points, sheared_points[0]])
    plt.plot(sheared_closed[:, 0], sheared_closed[:, 1], marker='o')
    plt.title('Segitiga Setelah Shear')
    plt.grid(True)

    plt.show()

# Fungsi untuk melakukan shear pada objek segitiga
def shear_segitiga(points, shear_x, shear_y):
    # Matriks transformasi shear
    shear_matrix = np.array([[1, shear_x],
                           [shear_y, 1]])

    # Mengalikan setiap titik dengan matriks shear
    sheared_points = []
    for point in points:
        new_point = np.dot(shear_matrix, point)
        sheared_points.append(new_point)
```

```

    return sheared_points

# Input dinamis dari user
print("Masukkan faktor shear horizontal (Shear X): ")
shear_x = float(input()) # Shear X (horizontal)
print("Masukkan faktor shear vertikal (Shear Y): ")
shear_y = float(input()) # Shear Y (vertikal)

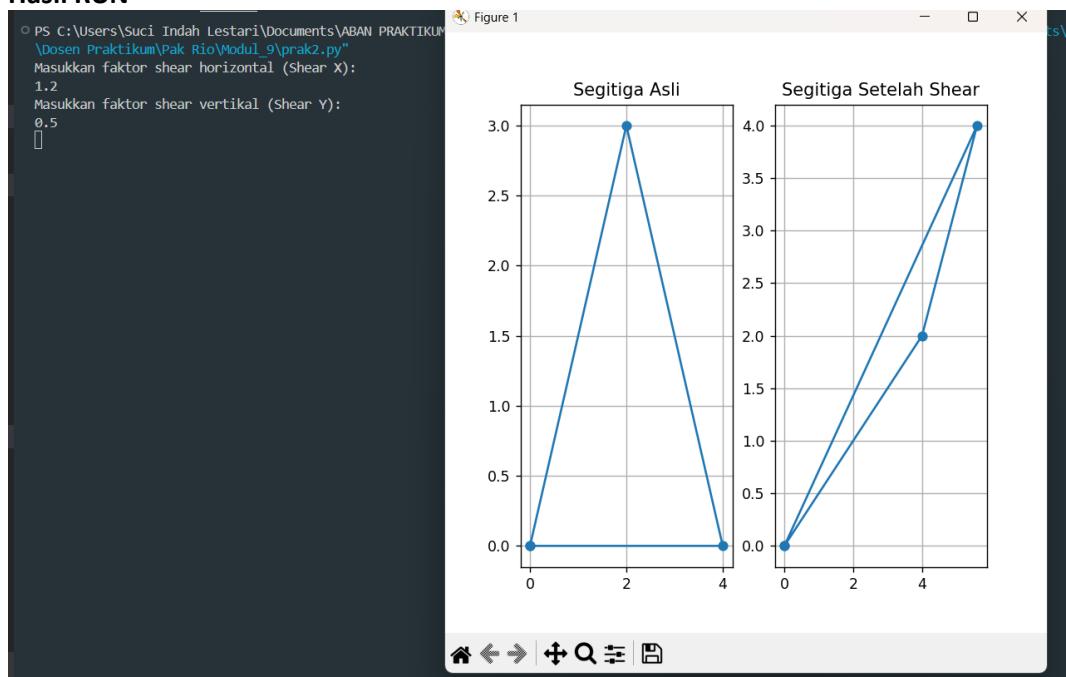
# Koordinat asli segitiga
segitiga_points = np.array([[0, 0], [4, 0], [2, 3]])

# Lakukan transformasi shear
sheared_points = shear_segitiga(segitiga_points, shear_x, shear_y)

# Plot hasilnya
plot_segitiga(segitiga_points, sheared_points)

```

### Hasil RUN



### 3. Praktikum 3

#### Code Program

```

# Import library yang diperlukan
import numpy as np
import matplotlib.pyplot as plt

# Fungsi untuk menggambar lingkaran sebelum dan sesudah shear
def plot_lingkaran(original_points, sheared_points):
    # Gambar lingkaran sebelum shear
    plt.figure(figsize=(6, 6))
    plt.subplot(1, 2, 1)
    plt.plot(original_points[0], original_points[1], label='Lingkaran Asli')
    plt.title('Lingkaran Asli')
    plt.gca().set_aspect('equal', adjustable='box')
    plt.grid(True)

    # Gambar lingkaran setelah shear
    plt.subplot(1, 2, 2)
    plt.plot(sheared_points[0], sheared_points[1], label='Lingkaran Setelah Shear')

```

```

plt.title('Lingkaran Setelah Shear')
plt.gca().set_aspect('equal', adjustable='box')
plt.grid(True)

plt.show()

# Fungsi untuk melakukan shear pada lingkaran
def shear_lingkaran(points, shear_x, shear_y):
    # Matriks transformasi shear
    shear_matrix = np.array([[1, shear_x],
                            [shear_y, 1]])

    # Mengalikan setiap titik lingkaran dengan matriks shear
    sheared_points = np.dot(shear_matrix, points)

    return sheared_points

# Input dinamis dari user
print("Masukkan faktor shear horizontal (Shear X): ")
shear_x = float(input()) # Shear X (horizontal)
print("Masukkan faktor shear vertikal (Shear Y): ")
shear_y = float(input()) # Shear Y (vertikal)

# Menghasilkan koordinat lingkaran asli
theta = np.linspace(0, 2 * np.pi, 100) # Sudut lingkaran
radius = 5 # Jari-jari lingkaran
x = radius * np.cos(theta) # Koordinat X lingkaran
y = radius * np.sin(theta) # Koordinat Y lingkaran

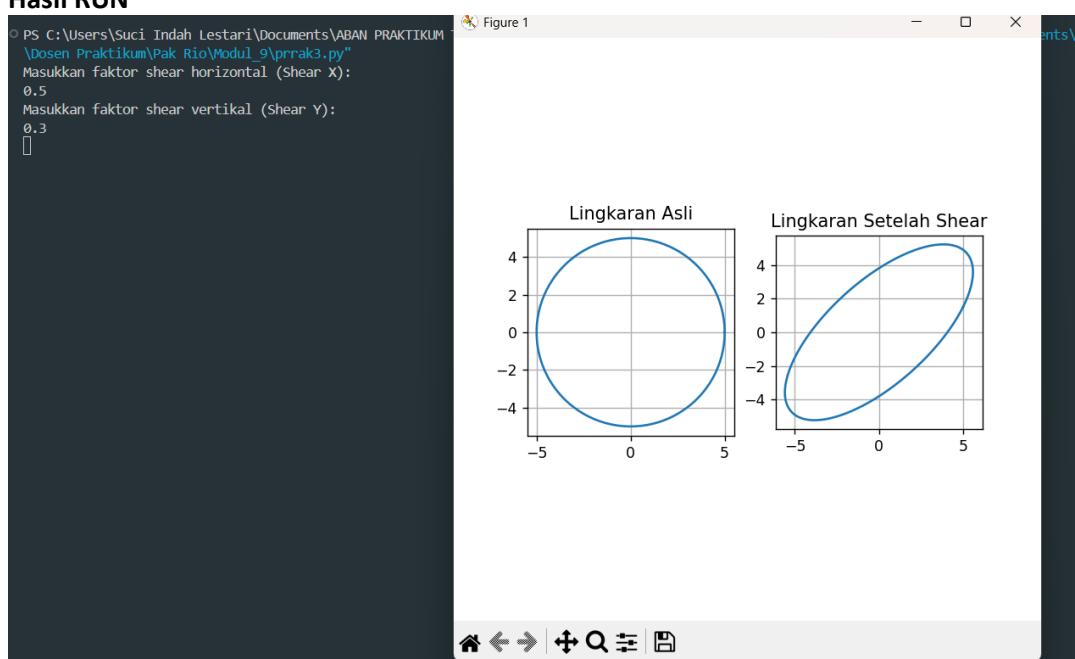
# Gabungkan koordinat X dan Y menjadi matriks
lingkaran_points = np.array([x, y])

# Lakukan transformasi shear
sheared_points = shear_lingkaran(lingkaran_points, shear_x, shear_y)

# Plot hasilnya
plot_lingkaran(lingkaran_points, sheared_points)

```

### Hasil RUN



## TUGAS

1. Buatlah kode program sederhana untuk menampilkan shear untuk objek persegi Panjang berikut :

### Code Program

```
# Import library yang diperlukan
import matplotlib.pyplot as plt
import numpy as np

# Fungsi untuk menggambar persegi panjang sebelum dan sesudah shear
def plot_persegi_panjang(original_points, sheared_points):
    plt.figure(figsize=(6, 8))

    # Gambar persegi panjang asli
    plt.subplot(2, 1, 1)
    original_closed = np.vstack([original_points, original_points[0]])
    plt.plot(original_closed[:, 0], original_closed[:, 1], marker='o')
    plt.title('Persegi Panjang Asli')
    plt.grid(True)

    # Gambar persegi panjang setelah shear
    plt.subplot(2, 1, 2)
    sheared_closed = np.vstack([sheared_points, sheared_points[0]])
    plt.plot(sheared_closed[:, 0], sheared_closed[:, 1], marker='o')
    plt.title('Persegi Panjang Setelah Shear')
    plt.grid(True)

    plt.tight_layout()
    plt.show()

# Fungsi untuk melakukan shear pada persegi panjang
def shear_persegi_panjang(points, shear_x, shear_y):
    shear_matrix = np.array([[1, shear_x],
                           [shear_y, 1]])
    sheared_points = []
    for point in points:
        new_point = np.dot(shear_matrix, point)
        sheared_points.append(new_point)
    return np.array(sheared_points)

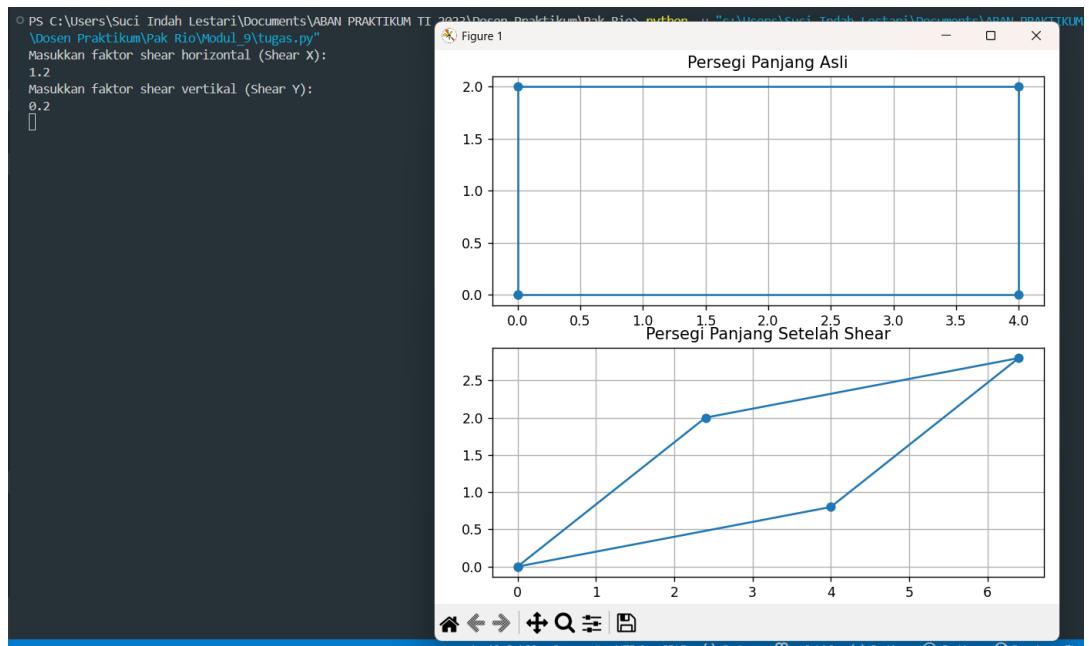
# Input faktor shear dari pengguna
print("Masukkan faktor shear horizontal (Shear X): ")
shear_x = float(input())
print("Masukkan faktor shear vertikal (Shear Y): ")
shear_y = float(input())

# Koordinat persegi panjang (misalnya 4 titik sudut)
persegi_panjang_points = np.array([[0, 0], [4, 0], [4, 2], [0, 2]])

# Lakukan shear
sheared_points = shear_persegi_panjang(persegi_panjang_points, shear_x,
                                         shear_y)

# Tampilkan hasil
plot_persegi_panjang(persegi_panjang_points, sheared_points)
```

## Hasil RUN



## Modul 10

### Praktikum

#### 1. Praktikum 1

##### Code Program

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Fungsi untuk menerapkan translasi pada objek
def translate_cube(x, y, z, tx, ty, tz):
    # Mengubah koordinat
    translation_matrix = np.array([tx, ty, tz])
    translated_x = x + translation_matrix[0]
    translated_y = y + translation_matrix[1]
    translated_z = z + translation_matrix[2]
    return translated_x, translated_y, translated_z

# Input translasi dari pengguna
tx = float(input("Masukkan nilai translasi pada sumbu x: "))
ty = float(input("Masukkan nilai translasi pada sumbu y: "))
tz = float(input("Masukkan nilai translasi pada sumbu z: "))

# Membuat objek kubus
def create_cube(length):
    x = np.array([0, 0, 0, 0, length, length, length, length])
    y = np.array([0, 0, length, length, 0, 0, length, length])
    z = np.array([0, length, 0, length, 0, length, 0, length])
    return x, y, z

length = 3 # panjang sisi kubus
x, y, z = create_cube(length)

# translasi pada objek
translated_x, translated_y, translated_z = translate_cube(x, y, z, tx, ty, tz)

# translasi X
# translasi Y
# translasi Z

# Fungsi untuk menggambar kubus dengan garis
def draw_cube_wireframe(ax, x, y, z, color, label):
    # Definisi edges kubus (menghubungkan titik-titik)
    edges = [
        [0, 1], [1, 3], [3, 2], [2, 0], # bottom face
        [4, 5], [5, 7], [7, 6], [6, 4], # top face
        [0, 4], [1, 5], [2, 6], [3, 7] # vertical edges
    ]

    for edge in edges:
        points = np.array([edge[0], edge[1]])
        ax.plot3D([x[points[0]], x[points[1]]],
                  [y[points[0]], y[points[1]]],
                  [z[points[0]], z[points[1]]],
                  color=color, linewidth=2)
```

```

# Plot titik-titik untuk visualisasi
ax.scatter(x, y, z, color=color, s=50, alpha=0.6, label=label)

# Visualisasi objek
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Penggambaran objek asli
draw_cube_wireframe(ax, x, y, z, 'cyan', 'Asli')

# Penggambaran objek yang sudah ditranslasi
draw_cube_wireframe(ax, translated_x, translated_y, translated_z, 'red',
'Translasi')

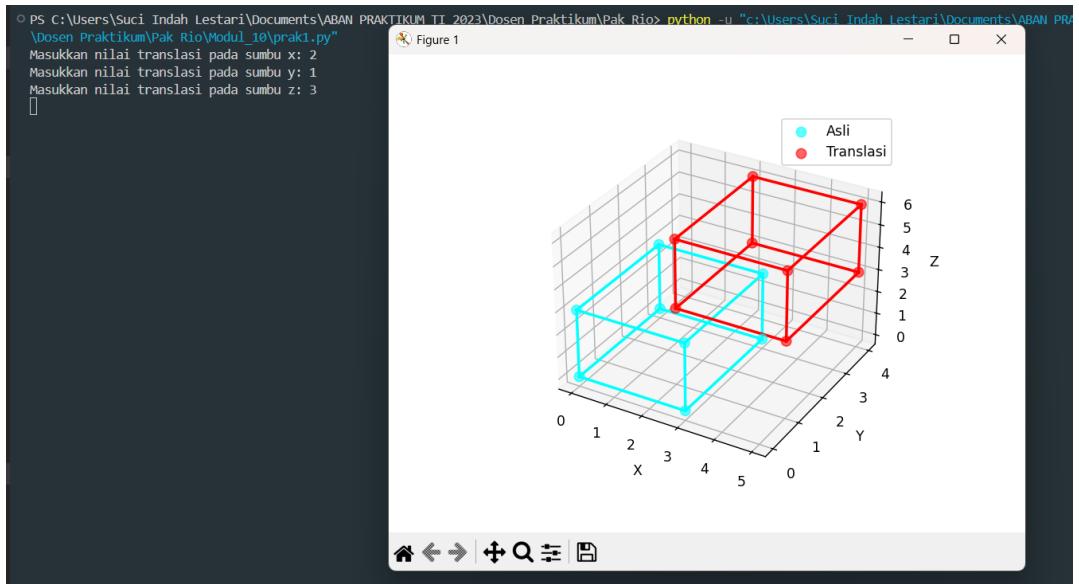
# Menambahkan label dan legenda
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.legend()

# Menghindari error dengan membuat handles untuk legend
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, labels)

# Tampilkan grafik
plt.show()

```

### Hasil RUN



## 2. Praktikum 2

### Code Program

```

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

# Fungsi untuk membuat objek limas
def create_pyramid(base_length, height):

```

```

# Titik-titik sudut dasar limas
base_x = np.array([0, base_length, base_length, 0])
base_y = np.array([0, 0, base_length, base_length])
base_z = np.zeros(4)

# Titik puncak limas
apex = np.array([[base_length / 2, base_length / 2, height]])

# Menggabungkan semua titik
x = np.concatenate((base_x, apex[:, 0]))
y = np.concatenate((base_y, apex[:, 1]))
z = np.concatenate((base_z, apex[:, 2]))

return x, y, z, apex

# Fungsi untuk menerapkan penskalaan pada objek
def scale_pyramid(x, y, z, scale_x, scale_y, scale_z):
    # Matriks penskalaan
    scale_matrix = np.array([scale_x, scale_y, scale_z])

    # Mengubah koordinat
    scaled_x = x * scale_matrix[0]
    scaled_y = y * scale_matrix[1]
    scaled_z = z * scale_matrix[2]

    return scaled_x, scaled_y, scaled_z

# Parameter geometri objek
base_length = 4
height = 4

# Pengambilan input faktor penskalaan
scale_x = float(input("Masukkan faktor penskalaan pada sumbu X: "))
scale_y = float(input("Masukkan faktor penskalaan pada sumbu Y: "))
scale_z = float(input("Masukkan faktor penskalaan pada sumbu Z: "))

# Membuat objek limas
x, y, z, apex = create_pyramid(base_length, height)

# Melakukan penskalaan
scaled_x, scaled_y, scaled_z = scale_pyramid(x, y, z, scale_x, scale_y, scale_z)

# Visualisasi objek
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Objek asli
ax.add_collection3d(
    Poly3DCollection(
        [list(zip(x[:4], y[:4], z[:4]))],
        color='gray',
        alpha=0.5,
        label='Asli'
    )
)

# Objek setelah skala
ax.add_collection3d(

```

```

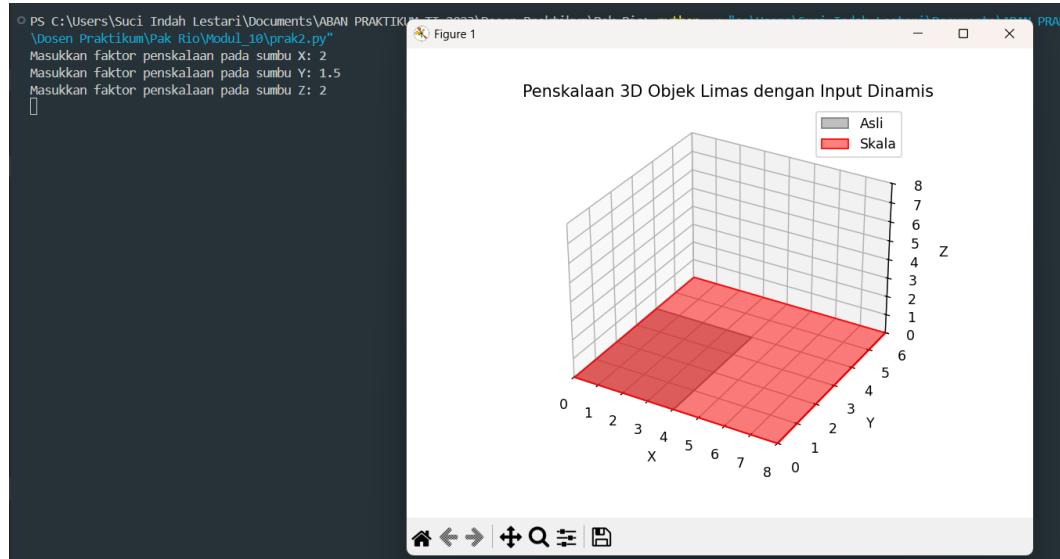
Poly3DCollection(
    [list(zip(scaled_x[:4], scaled_y[:4], scaled_z[:4]))],
    color='red',
    alpha=0.5,
    label='Skala'
)
)

# Label sumbu dan legenda
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Penskalaan 3D Objek Limas dengan Input Dinamis')
ax.legend()

#menentukan batas sumbu
ax.set_xlim([0, base_length * scale_x])
ax.set_ylim([0, base_length * scale_y])
ax.set_zlim([0, height * scale_z])
plt.show()

```

### Hasil RUN



## Tugas

1. Buatlah refleksi dari praktikum modul terkait translasi dan penskalaan objek 3D.

Jawab :

- Saat menerapkan translasi, saya melihat bagaimana kubus berpindah ke lokasi baru sesuai input pengguna. Hal ini membantu memahami bahwa translasi sangat berguna untuk mengatur posisi objek dalam ruang 3D.
- Saat menerapkan penskalaan, saya menyadari bahwa penskalaan uniform menjaga proporsi objek, sedangkan penskalaan non-uniform dapat mendistorsi bentuk. Misalnya, limas yang diperbesar hanya pada sumbu X akan tampak melebar ke samping.
- Visualisasi dengan matplotlib 3D sangat membantu karena saya bisa langsung melihat perbedaan antara objek asli dan hasil transformasi.