LAPORAN PRAKTIKUM
GRAFIKA KOMPUTER
(Dosen : *Rio Priantama S.T., M.T.I.*)

Modul 7

Nama : Muhammad Rizal Nurfirdaus

NIM : 20230810088

Kelas : TINFC-2023-04

TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS KUNINGAN

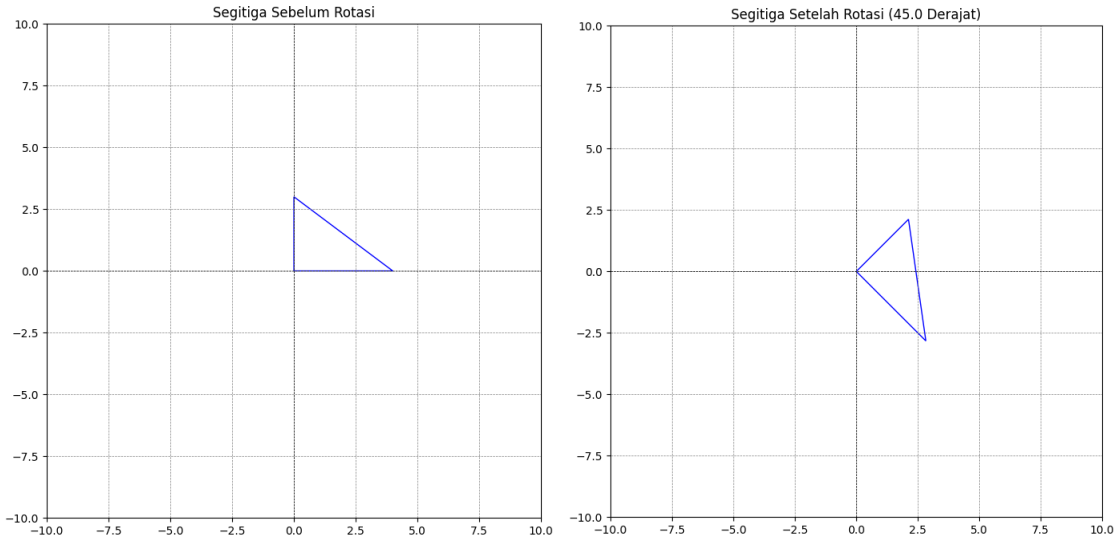
Praktikum

1. Praktikum 1 : Rotasi Objek Segitiga
   Source Code:

```
# Praktikum 1
# Import library yang diperlukan
import numpy as np
import matplotlib.pyplot as plt

# Fungsi untuk menggambar segitiga
def gambar_segitiga(titik):
    segitiga = plt.Polygon(titik, closed=True, fill=None,
edgecolor='b')
    plt.gca().add_patch(segitiga)


# Fungsi untuk melakukan rotasi
def rotasi(titik, sudut):
    radian = np.radians(sudut)

    # Matriks rotasi
    rotasi_matrix = np.array([
        [np.cos(radian), -np.sin(radian)],
        [np.sin(radian),  np.cos(radian)]
    ])

    # Melakukan rotasi pada setiap titik
    titik_rotated = np.dot(titik, rotasi_matrix)
    return titik_rotated


# Input titik segitiga dari pengguna
x1 = float(input("Masukkan koordinat x1: "))
y1 = float(input("Masukkan koordinat y1: "))
x2 = float(input("Masukkan koordinat x2: "))
y2 = float(input("Masukkan koordinat y2: "))
x3 = float(input("Masukkan koordinat x3: "))
y3 = float(input("Masukkan koordinat y3: "))

# Titik-titik segitiga
titik_asli = np.array([[x1, y1], [x2, y2], [x3, y3]])

# Input sudut rotasi
sudut_rotasi = float(input("Masukkan sudut rotasi (dalam derajat): "))


# ================================
#   GAMBAR SEBELUM ROTASI
# ================================
```

```
plt.figure(figsize=(8, 8))
plt.xlim(-10, 10)
plt.ylim(-10, 10)

plt.axhline(0, color='black', linewidth=0.5, linestyle='--')
plt.axvline(0, color='black', linewidth=0.5, linestyle='--')
plt.grid(color='gray', linestyle='--', linewidth=0.5)

gambar_segitiga(titik_asli)
plt.title('Segitiga Sebelum Rotasi')
plt.gca().set_aspect('equal', adjustable='box')
plt.show()


# ================================
#   PROSES ROTASI
# ================================
titik_rotated = rotasi(titik_asli, sudut_rotasi)


# ================================
#   GAMBAR SETELAH ROTASI
# ================================
plt.figure(figsize=(8, 8))
plt.xlim(-10, 10)
plt.ylim(-10, 10)

plt.axhline(0, color='black', linewidth=0.5, linestyle='--')
plt.axvline(0, color='black', linewidth=0.5, linestyle='--')
plt.grid(color='gray', linestyle='--', linewidth=0.5)

gambar_segitiga(titik_rotated)
plt.title(f'Segitiga Setelah Rotasi ({sudut_rotasi} Derajat)')
plt.gca().set_aspect('equal', adjustable='box')
plt.show()
```

Hasil Run:



Analisis: Rotasi yang ditunjukkan pada gambar memperlihatkan bagaimana setiap titik segitiga berubah posisi ketika dikenakan transformasi rotasi terhadap titik pusat koordinat (0,0). Bentuk segitiga tetap sama karena rotasi merupakan transformasi isometri, sehingga ukuran, panjang sisi, dan sudut internal tidak berubah. Yang mengalami perubahan hanyalah orientasi dan letak setiap titik relatif terhadap sumbu X dan Y. Hal ini terlihat dari titik yang berada tepat pada origin tetap berada pada posisinya, sedangkan dua titik lainnya berpindah sesuai perhitungan matriks rotasi.

Perbedaan visual antara gambar sebelum dan sesudah rotasi juga menunjukkan bahwa segitiga tampak “berputar menjauh” atau “mendekat” terhadap origin, bukan sekadar memutar orientasi di tempatnya. Hal ini terjadi karena rotasi dilakukan terhadap pusat koordinat, bukan terhadap pusat massa atau centroid segitiga. Jika rotasi dilakukan terhadap centroid, segitiga akan berputar pada posisinya tanpa mengalami perpindahan global.

Secara keseluruhan, hasil rotasi telah memenuhi sifat rotasi dua dimensi: bentuk objek tetap terjaga, orientasi berubah sesuai sudut yang diberikan, dan posisi objek bergantung pada titik pusat rotasi. Visualisasi menggunakan grid dan aspect ratio yang sama membuat perubahan orientasi tersebut terlihat jelas dan akurat tanpa distorsi skala.

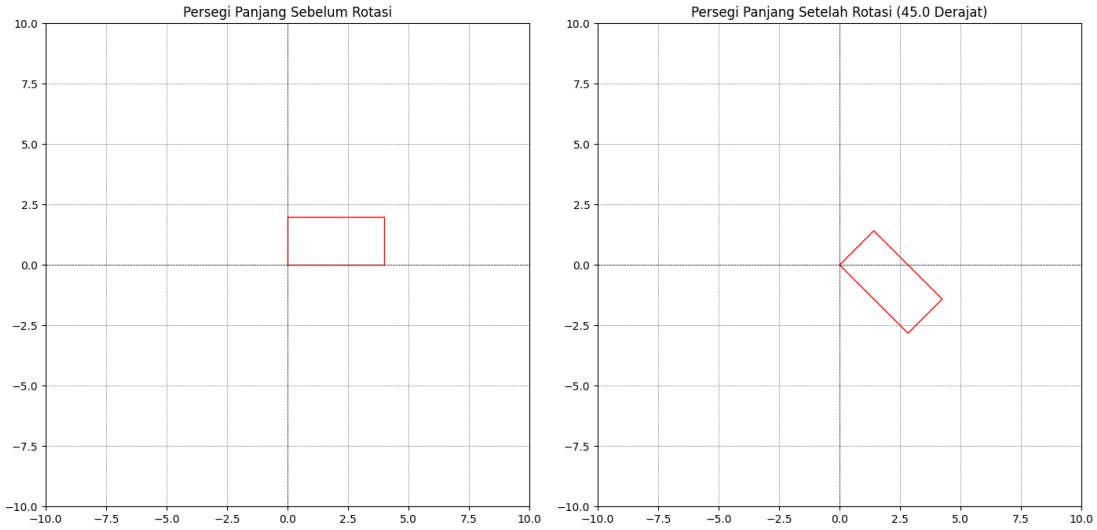
2. Praktikum 2 : Penskalaan Objek Persegi Panjang
   Source Code:

```
# Praktikum 2
# Import library yang diperlukan
import numpy as np
import matplotlib.pyplot as plt

# Fungsi untuk menggambar persegi panjang
def gambar_persegi_panjang(titik):
    # Menggambar persegi panjang berdasarkan titik yang diberikan
```

```
    persegi_panjang = plt.Polygon(titik, closed=True, fill=None,
edgecolor='r')
    plt.gca().add_patch(persegi_panjang)

# Fungsi untuk melakukan rotasi
def rotasi(titik, sudut):
    # Menghitung sudut dalam radian
    radian = np.radians(sudut)

    # Matriks rotasi
    rotasi_matrix = np.array([
        [np.cos(radian), -np.sin(radian)],
        [np.sin(radian), np.cos(radian)]
    ])

    # Melakukan rotasi pada setiap titik persegi panjang
    titik_rotated = np.dot(titik, rotasi_matrix)
    return titik_rotated

# Input titik persegi panjang dari pengguna
x1 = float(input("Masukkan koordinat x1: "))
y1 = float(input("Masukkan koordinat y1: "))
x2 = float(input("Masukkan koordinat x2: "))
y2 = float(input("Masukkan koordinat y2: "))
x3 = float(input("Masukkan koordinat x3: "))
y3 = float(input("Masukkan koordinat y3: "))
x4 = float(input("Masukkan koordinat x4: "))
y4 = float(input("Masukkan koordinat y4: "))

# Titik-titik persegi panjang
titik_asli = np.array([[x1, y1], [x2, y2], [x3, y3], [x4, y4]])

# Input sudut rotasi dari pengguna
sudut_rotasi = float(input("Masukkan sudut rotasi (dalam derajat): "))

# Menggambar persegi panjang sebelum rotasi
plt.figure(figsize=(8, 8))
plt.xlim(-10, 10)
plt.ylim(-10, 10)
plt.axhline(0, color='black', linewidth=0.5, ls='--')
plt.axvline(0, color='black', linewidth=0.5, ls='--')
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Gambar persegi panjang asli
gambar_persegi_panjang(titik_asli)
plt.title('Persegi Panjang Sebelum Rotasi')
plt.gca().set_aspect('equal', adjustable='box')
plt.show()
```

```python
# Melakukan rotasi
titik_rotated = rotasi(titik_asli, sudut_rotasi)

# Menggambar persegi panjang setelah rotasi
plt.figure(figsize=(8, 8))
plt.xlim(-10, 10)
plt.ylim(-10, 10)
plt.axhline(0, color='black', linewidth=0.5, ls='--')
plt.axvline(0, color='black', linewidth=0.5, ls='--')
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Gambar persegi panjang yang sudah dirotasi
gambar_persegi_panjang(titik_rotated)
plt.title(f'Persegi Panjang Setelah Rotasi ({sudut_rotasi} Derajat)')
plt.gca().set_aspect('equal', adjustable='box')
plt.show()
```

Hasil Run:



Analisis: Visualisasi sebelum dan sesudah rotasi menunjukkan bahwa persegi panjang mengalami perubahan orientasi ketika dikenakan transformasi rotasi terhadap titik pusat koordinat. Bentuk asli persegi panjang tetap dipertahankan karena rotasi merupakan transformasi isometri yang tidak mengubah panjang sisi maupun ukuran objek. Perubahan yang terlihat hanya pada sudut kemiringan dan posisi koordinat titik-titik sudut setelah diputar sebesar sudut rotasi yang ditentukan.

Persegi panjang pada posisi awal berada pada orientasi horizontal. Setelah dilakukan rotasi sebesar sudut tertentu, objek tampak berputar menjauh dari orientasi awalnya dan menjadi miring terhadap sumbu X dan Y. Rotasi dilakukan terhadap titik pusat koordinat (0,0), sehingga objek juga mengalami perpindahan global, bukan hanya perubahan arah. Jika rotasi dilakukan terhadap pusat persegi panjang, maka bentuk akan berputar di tempat tanpa berpindah posisi keseluruhan.

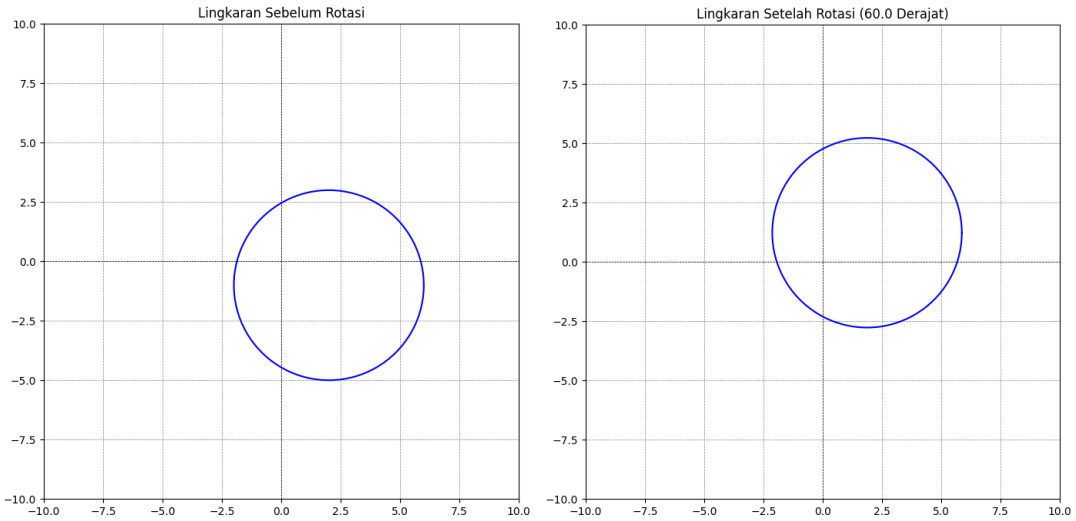
Secara keseluruhan, hasil yang diperoleh menggambarkan sifat dasar rotasi dua dimensi: objek tetap memiliki bentuk yang sama, tetapi orientasinya berubah sesuai matriks rotasi yang diterapkan. Grid dan aspek rasio yang setara membantu memperjelas perubahan orientasi persegi panjang setelah proses rotasi dilakukan. Jika sudut diperbesar atau diperkecil, perubahan orientasi akan semakin terlihat sesuai besar sudut rotasinya.

3. Praktikum 3 : Pensklaan Objek Lingkaran
   Source Code:

```
# Praktikum 3
# Import library yang diperlukan
import numpy as np
import matplotlib.pyplot as plt

# Fungsi untuk menggambar lingkaran
def gambar_lingkaran(radius, center):
    # Membuat array sudut dari 0 hingga 2*pi
    theta = np.linspace(0, 2 * np.pi, 100)

    # Menghitung koordinat x dan y dari lingkaran
    x = radius * np.cos(theta) + center[0]
    y = radius * np.sin(theta) + center[1]

    # Menggambar lingkaran
    plt.plot(x, y, label='Lingkaran', color='b')

# Fungsi untuk melakukan rotasi (menggeser pusat lingkaran)
def rotasi(center_x, center_y, radius, sudut):
    # Menghitung sudut dalam radian
    radian = np.radians(sudut)

    # Titik awal di lingkaran sebelum rotasi (titik pusat, karena
lingkaran tidak berubah bentuk saat rotasi)
    x = center_x
    y = center_y

    # Menghitung koordinat pusat baru setelah rotasi
    # Menggunakan matriks rotasi: x' = x*cos(rad) - y*sin(rad); y' =
x*sin(rad) + y*cos(rad)
    x_rotated = x * np.cos(radian) - y * np.sin(radian)
    y_rotated = x * np.sin(radian) + y * np.cos(radian)

    return x_rotated, y_rotated

# Input pusat lingkaran dan radius dari pengguna
center_x = float(input("Masukkan koordinat pusat x: "))
center_y = float(input("Masukkan koordinat pusat y: "))
radius = float(input("Masukkan radius lingkaran: "))
```

```
center = (center_x, center_y)

# Input sudut rotasi dari pengguna
sudut_rotasi = float(input("Masukkan sudut rotasi (dalam derajat): "))

# Menggambar lingkaran sebelum rotasi
plt.figure(figsize=(8, 8))
plt.xlim(-10, 10)
plt.ylim(-10, 10)
plt.axhline(0, color='black', linewidth=0.5, ls='--')
plt.axvline(0, color='black', linewidth=0.5, ls='--')
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Gambar lingkaran asli
gambar_lingkaran(radius, center)
plt.title('Lingkaran Sebelum Rotasi')
plt.gca().set_aspect('equal', adjustable='box')
plt.show()

# Melakukan rotasi
x_rotated, y_rotated = rotasi(center_x, center_y, radius, sudut_rotasi)

# Menggambar lingkaran setelah rotasi
plt.figure(figsize=(8, 8))
plt.xlim(-10, 10)
plt.ylim(-10, 10)
plt.axhline(0, color='black', linewidth=0.5, ls='--')
plt.axvline(0, color='black', linewidth=0.5, ls='--')
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Gambar lingkaran yang sudah dirotasi
gambar_lingkaran(radius, (x_rotated, y_rotated))
plt.title(f'Lingkaran Setelah Rotasi ({sudut_rotasi} Derajat)')
plt.gca().set_aspect('equal', adjustable='box')
plt.show()
```

Hasil Run :



Analisis: Program tersebut menampilkan proses rotasi terhadap pusat sebuah lingkaran, bukan rotasi bentuk lingkarannya. Karena lingkaran bersifat simetris terhadap rotasi, yang berubah hanya posisi pusatnya, sementara ukuran dan bentuknya tetap sama.

Pertama, program menggambar lingkaran awal menggunakan pusat (x, y)dan radius yang diberikan. Setelah itu, pusat lingkaran dihitung ulang menggunakan matriks rotasi dengan sudut yang dimasukkan pengguna. Hasil perhitungan ini menghasilkan koordinat pusat baru (x', y'), sehingga lingkaran tampak bergeser ke posisi berbeda.

Secara visual, gambar pertama menunjukkan lingkaran pada posisi asal dengan pusat awal. Gambar kedua menunjukkan lingkaran setelah pusatnya diputar terhadap titik asal $(0,0)$. Perpindahan letak pusat ini mencerminkan efek rotasi, sementara bentuk lingkaran tetap identik.

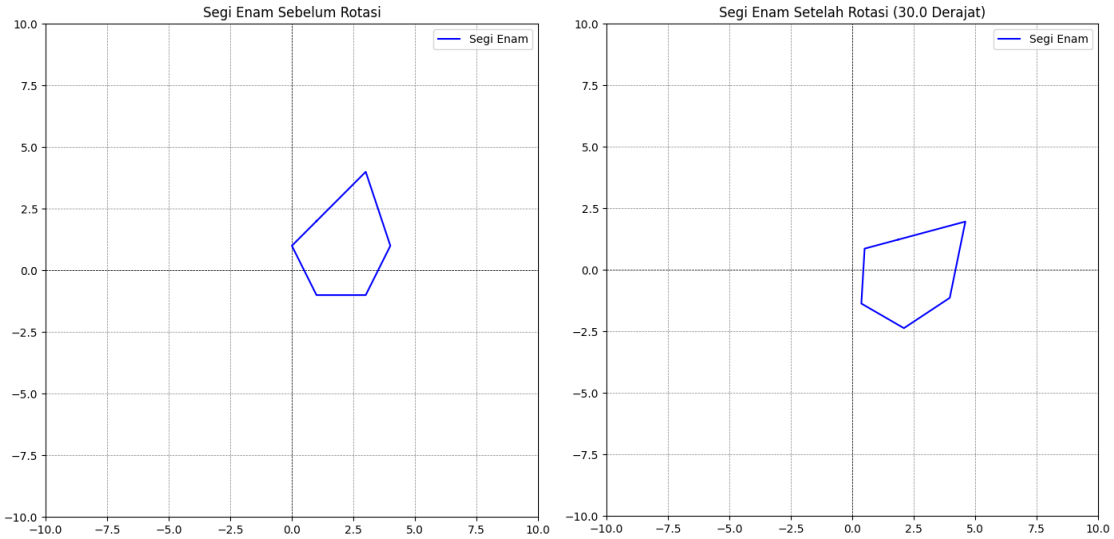
4. Praktikum 4 : Penskalaan Objek Lingkaran dan Persegi Panjang

```
# Praktikum 4
# Import library yang diperlukan
import numpy as np
import matplotlib.pyplot as plt

# Fungsi untuk menggambar segi enam
def gambar_segi_enam(titik):
    # Menggambar segi enam dengan mengulangi titik pertama di akhir
    titik_close = np.append(titik, [titik[0]], axis=0)
    plt.plot(titik_close[:, 0], titik_close[:, 1], label='Segi Enam',
color='b')
    # plt.scatter(titik[:, 0], titik[:, 1]) # Menambahkan titik
(opsional)

# Fungsi untuk melakukan rotasi
def rotasi(titik, sudut):
    # Menghitung sudut dalam radian
    radian = np.radians(sudut)
```

```
    # Matriks rotasi
    rotasi_matrix = np.array([
        [np.cos(radian), -np.sin(radian)],
        [np.sin(radian), np.cos(radian)]
    ])

    # Menghitung titik yang sudah dirotasi
    titik_rotated = np.dot(titik, rotasi_matrix)
    return titik_rotated

# Input titik-titik segi enam dari pengguna
print("Masukkan koordinat titik-titik segi enam (x, y):")
titik = []
for i in range(6):
    x = float(input(f"Koordinat x titik {i+1}: "))
    y = float(input(f"Koordinat y titik {i+1}: "))
    titik.append([x, y])

# Konversi list menjadi array numpy
titik_asli = np.array(titik)

# Input sudut rotasi dari pengguna
sudut_rotasi = float(input("Masukkan sudut rotasi (dalam derajat): "))

# Menggambar segi enam sebelum rotasi
plt.figure(figsize=(8, 8))
plt.xlim(-10, 10)
plt.ylim(-10, 10)
plt.axhline(0, color='black', linewidth=0.5, ls='--')
plt.axvline(0, color='black', linewidth=0.5, ls='--')
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Gambar segi enam asli
gambar_segi_enam(titik_asli)
plt.title('Segi Enam Sebelum Rotasi')
plt.gca().set_aspect('equal', adjustable='box')
plt.legend()
plt.show()

# Melakukan rotasi
titik_rotated = rotasi(titik_asli, sudut_rotasi)

# Menggambar segi enam setelah rotasi
plt.figure(figsize=(8, 8))
plt.xlim(-10, 10)
plt.ylim(-10, 10)
plt.axhline(0, color='black', linewidth=0.5, ls='--')
```

```
plt.axvline(0, color='black', linewidth=0.5, ls='--')
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Gambar segi enam yang sudah dirotasi
gambar_segi_enam(titik_rotated)
plt.title(f'Segi Enam Setelah Rotasi ({sudut_rotasi} Derajat)')
plt.gca().set_aspect('equal', adjustable='box')
plt.legend()
plt.show()
```

Hasil Run :



Analisis : Program pada Praktikum 4 ini menunjukkan bagaimana sebuah **segi enam beraturan maupun tidak beraturan** dapat diputar (dirotasi) terhadap titik pusat koordinat (0,0)menggunakan **matriks rotasi 2D**. Setiap titik penyusun segi enam diproses satu per satu sehingga seluruh bentuk berubah orientasi, tetapi **tetap mempertahankan ukuran dan bentuk geometrisnya**.

Sebelum rotasi dilakukan, program menerima enam titik dari pengguna dan menggambarkannya sebagai sebuah poligon tertutup. Bentuk awal ini ditampilkan pada grafik pertama. Setelah menerima sudut rotasi, program kemudian menerapkan **transformasi rotasi** melalui perkalian matriks, yaitu:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Transformasi ini menghasilkan posisi baru untuk setiap titik, sehingga seluruh segi enam tampak berputar mengelilingi titik pusat. Pada grafik kedua terlihat bahwa orientasi segi enam berubah sesuai sudut yang dimasukkan, namun jarak antar titik tetap sama. Dengan demikian, visualisasi sebelum dan sesudah rotasi memperlihatkan bahwa program berhasil menerapkan transformasi rotasi secara matematis dan grafis dengan benar.

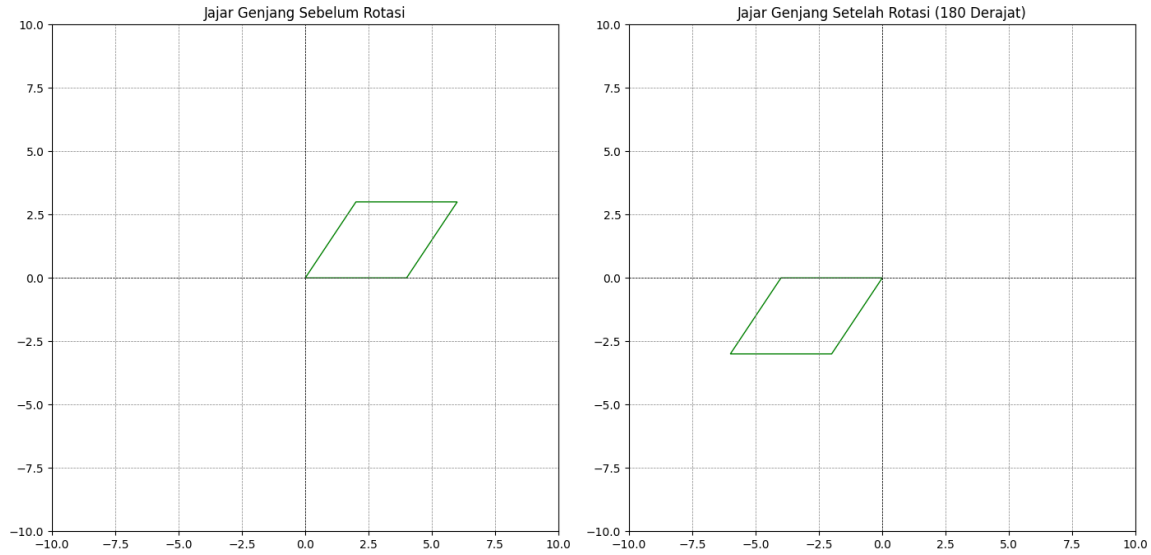
LATIHAN/TUGAS

1. Buatlah kode program sederhana untuk menggambar objek jajar genjang yang dirotasi sebanyak 180 derajat.
   Source Code:

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

# Fungsi untuk menggambar jajar genjang
def gambar_jajar_genjang(titik):
    # Jajar genjang adalah poligon dengan 4 titik
    jajar_genjang = plt.Polygon(titik, closed=True, fill=None,
edgecolor='g')
    plt.gca().add_patch(jajar_genjang)

# Fungsi untuk melakukan rotasi
def rotasi(titik, sudut):
    # Menghitung sudut dalam radian
    radian = np.radians(sudut)

    # Matriks rotasi 2D
    # R(theta) = [[cos(theta), -sin(theta)], [sin(theta), cos(theta)]]
    rotasi_matrix = np.array([
        [np.cos(radian), -np.sin(radian)],
        [np.sin(radian), np.cos(radian)]
    ])

    # Melakukan rotasi pada setiap titik jajar genjang
    titik_rotated = np.dot(titik, rotasi_matrix)
    return titik_rotated

# Input titik jajar genjang (4 titik)
print("Masukkan koordinat 4 titik jajar genjang (x, y):")
x1 = float(input("Koordinat x1: "))
y1 = float(input("Koordinat y1: "))
x2 = float(input("Koordinat x2: "))
y2 = float(input("Koordinat y2: "))
x3 = float(input("Koordinat x3: "))
y3 = float(input("Koordinat y3: "))
x4 = float(input("Koordinat x4: "))
y4 = float(input("Koordinat y4: "))

# Titik-titik jajar genjang
titik_asli = np.array([[x1, y1], [x2, y2], [x3, y3], [x4, y4]])

# Sudut rotasi yang diminta dalam tugas
```

```
sudut_rotasi = 180

# --- Visualisasi Sebelum Rotasi ---
plt.figure(figsize=(8, 8))
plt.xlim(-10, 10)
plt.ylim(-10, 10)
plt.axhline(0, color='black', linewidth=0.5, ls='--')
plt.axvline(0, color='black', linewidth=0.5, ls='--')
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Gambar jajar genjang asli
gambar_jajar_genjang(titik_asli)
plt.title('Jajar Genjang Sebelum Rotasi')
plt.gca().set_aspect('equal', adjustable='box')
plt.show()

# --- Melakukan Rotasi ---
titik_rotated = rotasi(titik_asli, sudut_rotasi)

# --- Visualisasi Setelah Rotasi ---
plt.figure(figsize=(8, 8))
plt.xlim(-10, 10)
plt.ylim(-10, 10)
plt.axhline(0, color='black', linewidth=0.5, ls='--')
plt.axvline(0, color='black', linewidth=0.5, ls='--')
plt.grid(color='gray', linestyle='--', linewidth=0.5)

# Gambar jajar genjang yang sudah dirotasi
gambar_jajar_genjang(titik_rotated)
plt.title(f'Jajar Genjang Setelah Rotasi ({sudut_rotasi} Derajat)')
plt.gca().set_aspect('equal', adjustable='box')
plt.show()
```

Hasil Run :



Analisis: Program ini membaca empat koordinat yang membentuk jajar genjang, menampilkan poligon tersebut, lalu mengaplikasikan rotasi 2D sebesar 180° menggunakan matriks rotasi standar dan menampilkan hasilnya. Inti matematisnya adalah matriks rotasi R(θ) = $\begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}$; untuk setiap titik (x, y)operasi $[x\ y] \cdot R$memberikan koordinat baru setelah rotasi. Karena rotasi dilakukan terhadap titik asal (origin), rotasi 180° secara khusus menghasilkan $(x, y) \mapsto (-x, -y)$— jadi seluruh jajar genjang tercermin ke kuadran berlawanan relatif terhadap origin.

Beberapa catatan implementasi penting:

1. Penggunaan np.dot(titik, rotasi_matrix) benar jika titik berupa array dengan bentuk (4,2) sehingga setiap baris dianggap satu titik (x,y). Hasilnya juga berformat (4,2).
2. Visualisasi sudah rapi (axis equal, grid, garis sumbu) — tetapi rotasi terhadap origin mungkin bukan yang diinginkan jika pengguna ingin memutar objek di tempatnya; untuk rotasi terhadap pusat poligon harus dikurangi centroid dulu, rotasi, lalu ditambahkan kembali centroid.
3. Input manual via input() bagus untuk tugas interaktif, tapi rentan kesalahan; untuk reproducibility lebih baik sediakan default contoh atau baca dari file/daftar; juga bisa menambahkan validasi jumlah titik dan tipe numerik.

Pengamatan pada gambar: gambar pertama menunjukkan jajar genjang di area positif (mis. di sebelah kanan/atas origin), dan gambar kedua setelah rotasi 180° menempatkan poligon pada posisi yang simetris terhadap origin (bergerak ke kuadran berlawanan), ukuran dan bentuknya identik karena rotasi adalah transformasi isometri (tidak merubah panjang atau sudut). Jika tujuan visual adalah membandingkan sebelum/dengan sesudah, pertimbangkan menampilkan keduanya pada satu plot (dengan warna/label berbeda) sehingga pergeseran posisi lebih mudah diamati.

Saran perbaikan singkat: tambahkan fungsi rotasi_tentang_titik(titik, sudut, pusat) yang otomatis menghitung centroid bila pusat=None; tambahkan penanganan error untuk input; dan berikan opsi menampilkan koordinat numerik sebelum/ sesudah rotasi. Secara

algoritmis program ini O(n) terhadap jumlah titik (di sini n=4), numeriknya stabil untuk sudut umum, dan solusi yang dipakai sudah sesuai untuk kebutuhan praktikum rotasi 2D.