MODUL IV

KOMPUTER GRAFIK 2D
ATRIBUTE GRAFIK DAN TRANSFORMASI 2D I & 2

D3 TEKNIK INFORMATIKA
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
POLITEKNIK NEGERI BANDUNG

MAHASISWA 000 | KOMPUTER GRAFIK | SEPTEMBER, 14 2023

CONTENTS

TIC TAC TOE GAME

https://playtictactoe.org/

Permainan Tic-Tac-Toe atau catur jawa atau XOX merupakan permainan classic yang digunakan untuk belajar pemrograman game. Selain Tic-Tac-Toe ada minesweeper, digger, snake, dan pong. Aturan main Tic-Tac-Toe sangat sederhana, 2 pemain berusaha menyelesaikan kondisi kemenangan yaitu ketika terdapat tiga tanda yang sama pada posisi vertikal, horizontal atau diagonal secara berurutan. Sebaliknya permainan akan draw jika 2 pemain tidak dapat menghasilkan kondisi tersebut.

Pemain 1 menulis X dan Pemain 2 menulis O pada papan 3x3

Pada praktikum sebelumnya kita telah membuat bentuk dasar lingkaran, garis dan kali dan akan memanfaatkan kode tersebut untuk membuat permainan ini.

Fitur-fitur Permainan Tic Tac Toe
1. Papan Grid 3x3 2. Pemain 1 bisa menuliskan X di Papan & Pemain 2 bisa menuliskan O di Papan 3. Kolom yang sudah dituliskan tidak boleh dituliskan kembali 4. Kondisi draw tidak ada X atau O yang sesuai dengan kondisi (Vertikal, Horizontal, Diagonal) 5. Kondisi menang ada X atau O yang sesuai dengan kondisi (Vertikal, Horizontal, Diagonal)

PYTHON 201: CLASS

Python 201.

1. Abstraksi
2.

KARYA 2D DENGAN OOP

Tictactoe

Asteroid

Stick Man

Kendaraan

ATTRIBUTE GRAFIK

Terdapat beberapa atribut Grafik yang dapat diimplemntasikan pada Komputer Grafik contohnya variasi dari bentuk dasar seperti Titik – titik, Titik – garis – titik, Garis – garis kosong garis garis, Dan variasi lainnya. Selain itu pengaturan dari tebal dan tipisnya bentuk 2D yang dihasilkan.

Secara implementasi untuk object 2D yang dibangun dari garis dapat diimplementasikan setelah Kumpulan points tercipta oleh algoritma line dda atau line bersenham, sedangkan pada lingkaran dan ellips pada algoritma Pembangunan lingkaran dan ellips atau dilakukan perubahan algoritma sehingga luaran dari algoritma lingkaran dan garis berupa array seperti pada garis.

TRANSFORMASI 2D

Transformasi 2D

TRANSLASI 2D

Translasi 2D

SCALING 2D

Scaling 2D

SHEAR 2D

Shear 2D

TASK PRAKTIKUM

TASK 1-3: REVIEW DAN EKSPLORASI

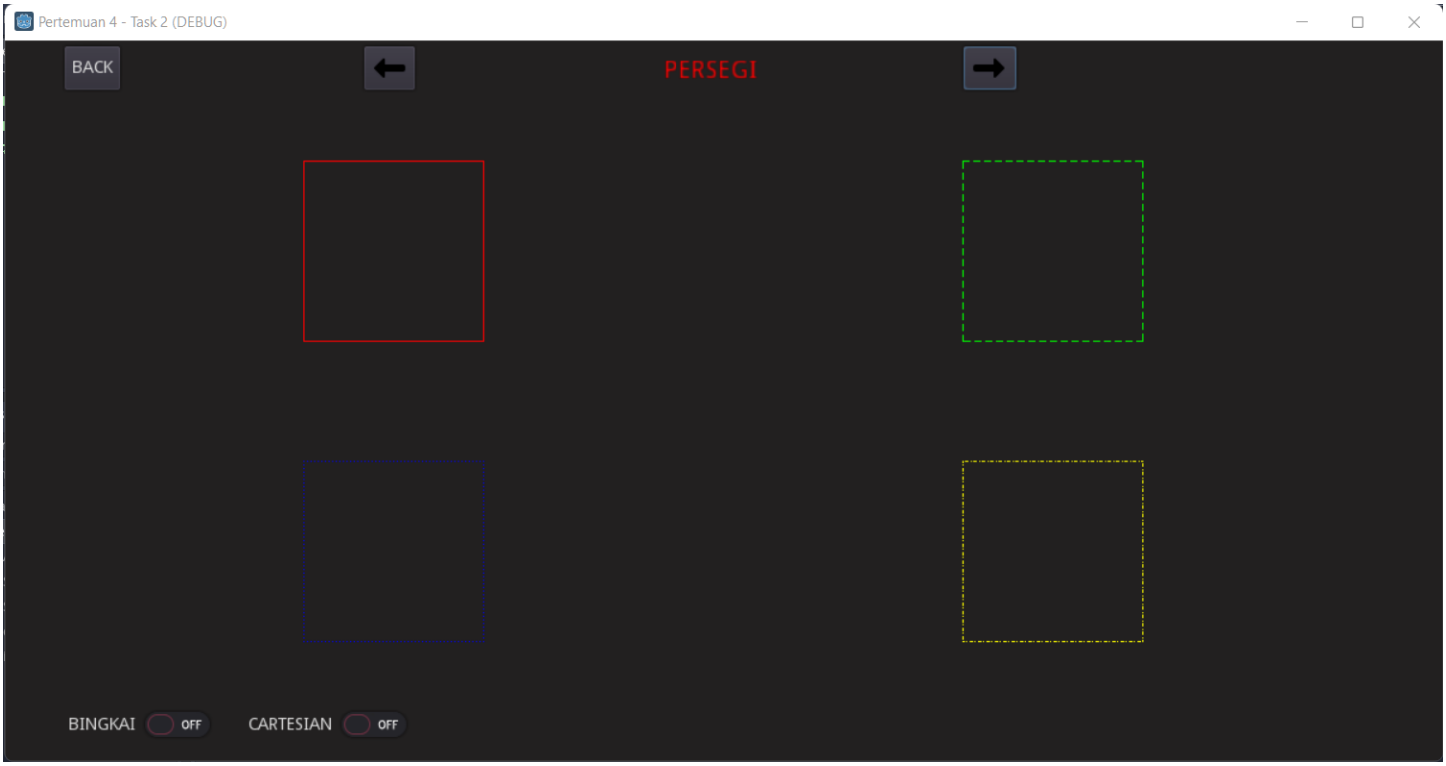
1. Download dan Buka File Pendukung Praktikum [KG2023_2X_001_D3_2022]_Modul4

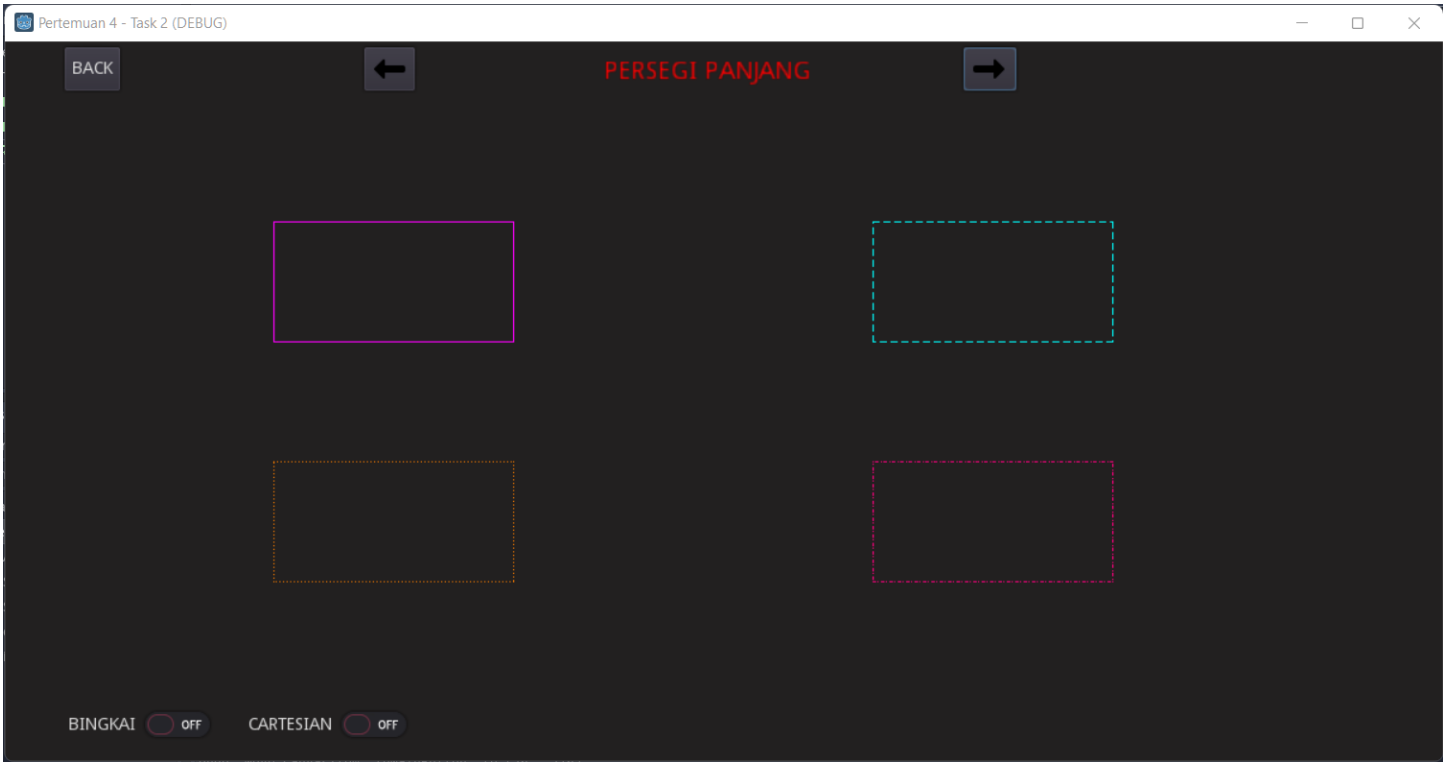
2. 001_Task01 adalah file tictactoe silakan coba dan analisa kode tersebut diskusikan jika ada yang tidak dipahami
3. 001_Task02 adalah file untuk memahami cara kerja object oriented di python
4. 001_Task03 adalah file untuk memahami cara kerja OOP terimplementasi pada tugas sebelumnya (Tictactoe, asteroid, stickman, dan kendaran).

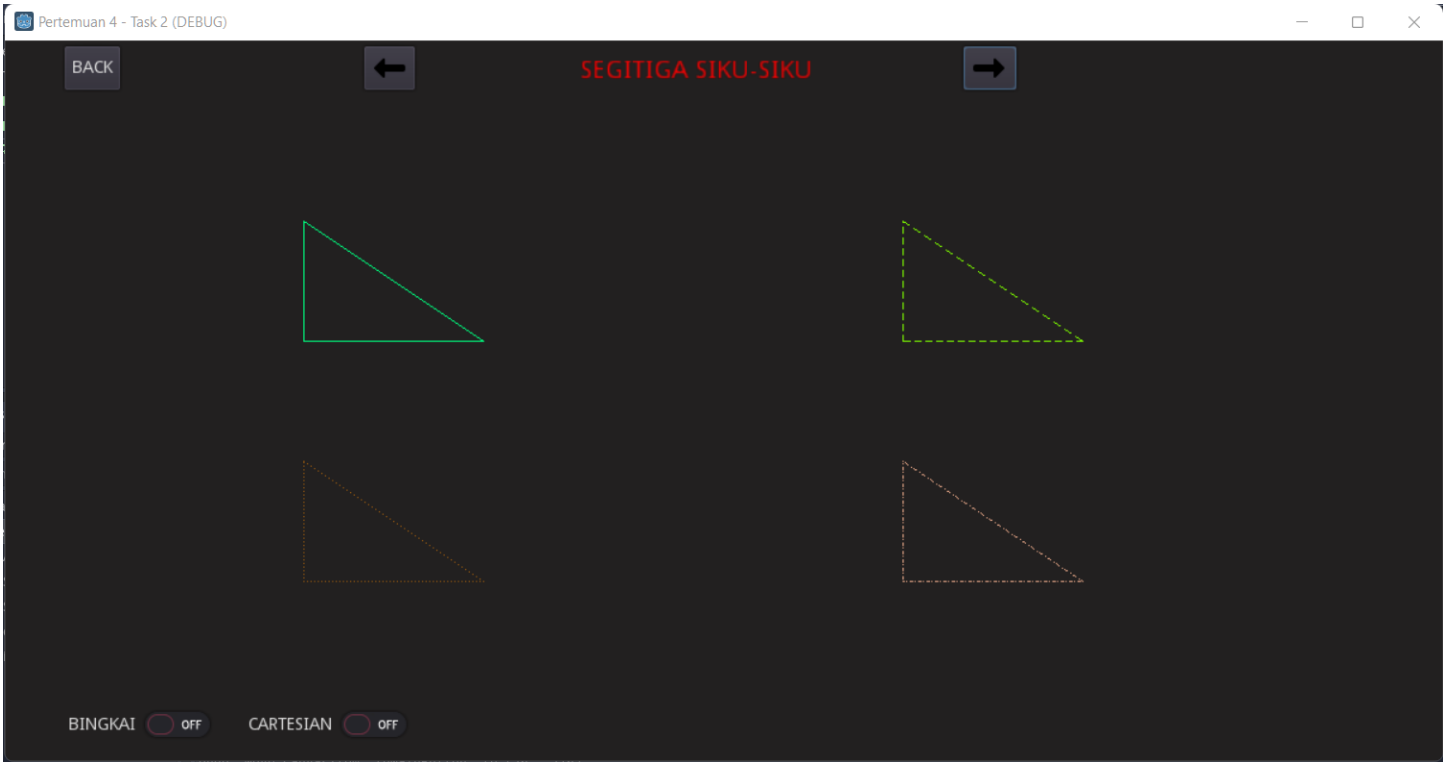
Lesson Learnt (Code, Print Screen Hasil Karya, dan Komentar)

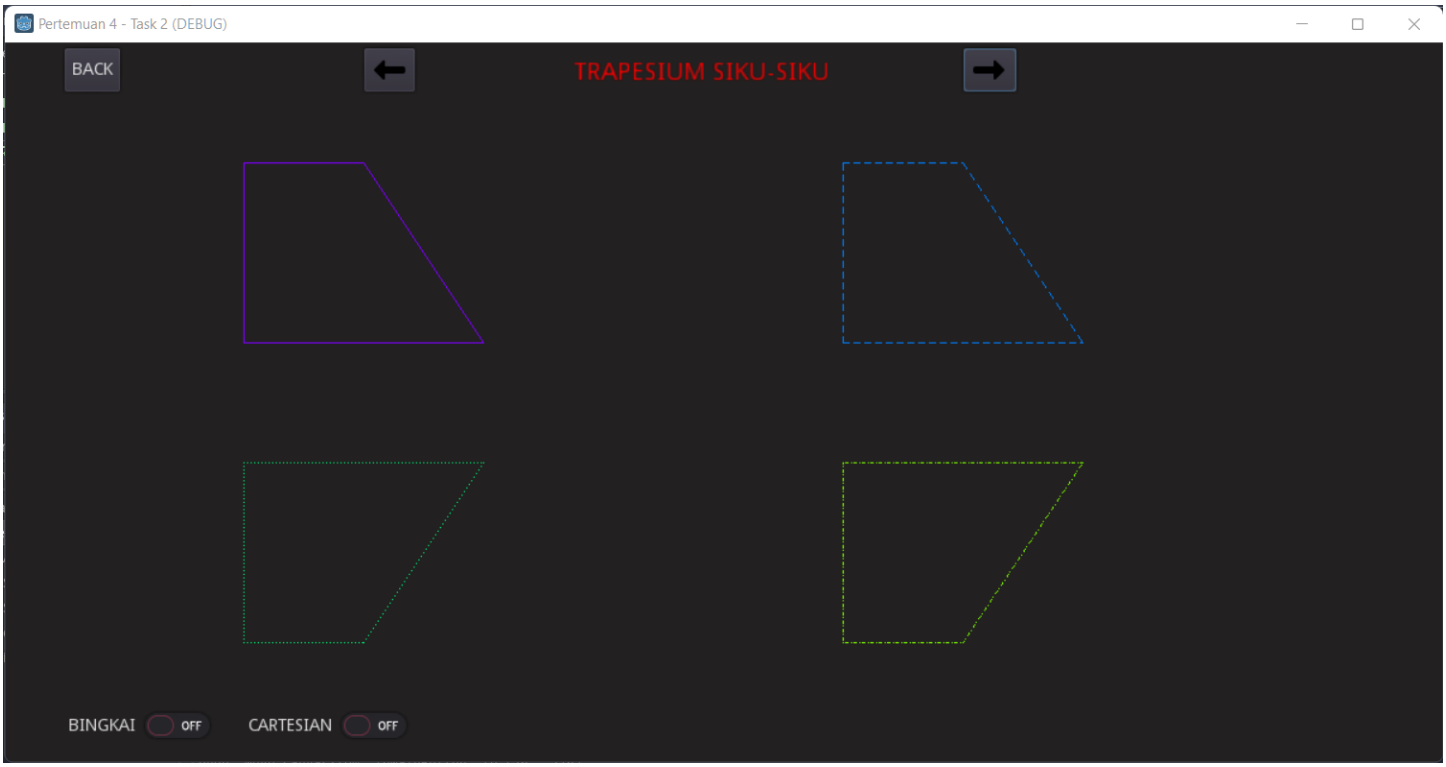
TASK 4: MEMBUAT FUNGSI BENTUK DASAR DENGAN ATTRIBUT GRAFIK

1. Copy Task 3
2. Buatlah Fungsi-fungsi Bentuk Dasar: Persegi, Persegi Panjang, Segitiga Siku-Siku, dan Trapesium Siku-Siku, Lingkaran dan Ellips dengan Attribut Grafik sbb:
   Titik – titik
   Titik – garis – titik
   Garis – garis kosong garis garis
   Dan variasi lainnya.
3. Konversi Karya 2D OOP dengan attribute grafik.


Pertemuan 4 - Task 2 (DEBUG)
BACK
PERSEGI
BINGKAI OFF
CARTESIAN OFF


Pertemuan 4 - Task 2 (DEBUG)
BACK
PERSEGI PANJANG
BINGKAI OFF
CARTESIAN OFF





Lesson Learnt (Code, Print Screen Hasil Karya, dan Komentar)

main.py

```
def setup():
    py5.size(800,600) #ubah size agar lebih besar

    py5.rect_mode(py5.CENTER)
    py5.background(191)
    primitif.basic.draw_margin(py5.width, py5.height, 25, c=[0, 0, 0, 255])
    primitif.basic.draw_kartesian(py5.width, py5.height, 25, c=[0, 0, 0, 255])
    c =[0,0,0,255]
```

Sebelumnya perlu untuk mengatur ukuran dari medianya, ukuran diubah agar hasil gambar lebih jelas, deklarasi juga c untuk warna. Digambarkan juga margin/garis tepi dan gari kartesian yang dibagi menjadi 4 kuadran.

basic.py

```
def draw_bentuk(pts, line_type, c):
    py5.stroke(c[0], c[1], c[2], c[3])
    fill_true = True
    j = 0

    if line_type == 0:
            py5.points(pts)
    elif line_type == 1:
        for x, y in pts:
            if j == 8:
                if fill_true:
                    fill_true = False
                else:
                    fill_true = True
                j = 0
            if fill_true:
                py5.point(x, y)
            j += 1
    elif line_type == 2:
        for x, y in pts:
            if x % 2 == 0 and y % 2 == 0:
                py5.point(x, y)
    elif line_type == 3:
        j = 10
        i = 1
        for x, y in pts:
            if j == 0:
                if fill_true:
                    fill_true = False
                    j = 7
                    if i == 1:
                        i = 2
                    else:
                        i = 1
                else:
                    fill_true = True
                    if i == 1:
                        j = 10
                    else:
                        j = 2
            if fill_true:
                py5.point(x, y)
            j -= 1
```

Pada method draw_bentuk, dibuat berbagai macam kondisi dengan parameter angka 0-3 yang dijadikan parameter. line_type akan menentukan jenis garis yang digunakan ketika method tsb dipanggil.
Jika line_type adalah 0, maka digambarkan variasi garis penuh, jika line_type adalah 1, maka digambarkan variasi garis putus putus, jika line_type adalah 2, maka digambarkan variasi dua titik dua titik terus berulang, dan jika line_type adalah 4, maka digambarkan yang lebih kompleks yaitu perulangan titik dan '~' yang akan ditampilkan

1. Persegi

   basic.py

```
def persegi(xa, ya, panjang):
    return np.concatenate(
        (
            primitif.line.line_bresenham(xa,ya,xa+panjang,ya)
            , primitif.line.line_bresenham(xa,ya+panjang,xa+panjang,ya+panjang)
            , primitif.line.line_bresenham(xa,ya,xa,ya+panjang)
            , primitif.line.line_bresenham(xa+panjang,ya, xa+panjang,ya+panjang)
            ),
        axis=0
    )
```

   Menggunakan method line_bresenham untuk membuat persegi, setiap sisinya dibuat satu persatu. Kode sudah didapatkan pada praktikum minggu ke-3, dan diperlukan penyesuaian.
   main.py. hal yang membedakan adalah menggunakan 'return np.concatenate()' di mana np sebagai library numpy, 'return np.concatenate()' berfungsi untuk mengembalikan hasil penggabungan/concat array atau matriks di dalam fungsi tsb agar hasil concat dapat digunakan di program lain atau disimpan di variabel. Hasilnya dapat dicetak sesuai keiiginan.

   main.py

```
#======persegi======#

    x, y = 50, 100
    sisi = 100
    tm =np.zeros(3) #inisiasi matriks
    #convert x dan y ke koordinat kartesian
    xo,yo = primitif.utility.convert_to_cartesian(x,y,py5.width,py5.height,25)
    xc,yc = (xo+x, yo-y)
```

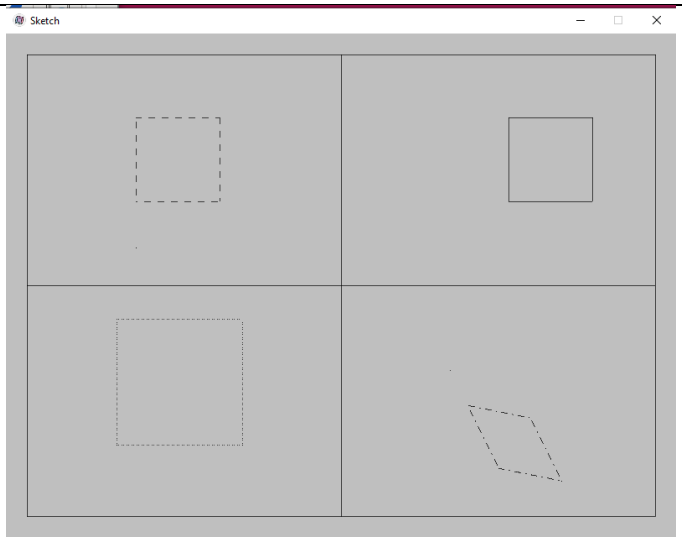
```
    #kuadran1
    persegi = primitif.basic.persegi(xc+100, yc, sisi)
    primitif.basic.draw_bentuk(persegi,0,c)

    #kuadran2
    persegi = primitif.basic.persegi(x+105,yc, sisi)
    primitif.basic.draw_bentuk(persegi,1,c)

    #kuadran3
    tm_kuadran3 = np.zeros(3)

    tm_kuadran3 = primitif.transformasiv2.rotate2D(180, 0, yc)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.reflectY2D(tm_kuadran3)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.scale2D(1.5,1.5, 0, 0, tm_kuadran3)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.translate2D(x-150, 2.2*yo, tm_kuadran3)
    print(tm_kuadran3)
    persegi = primitif.transformasiv2.transformPoints2D(persegi, tm_kuadran3)
    primitif.basic.draw_bentuk(persegi,2,c)

    #kuadran4
    tm_kuadran4 = np.zeros(3)

    tm_kuadran4 = primitif.transformasiv2.rotate2D(0, 0, yc)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.scale2D(0.5, 0.5, 0, 0, tm_kuadran4)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.translate2D(xc-200, yo, tm_kuadran4)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.shear2D(0.5, 0.2, tm_kuadran4)
    persegi = primitif.transformasiv2.transformPoints2D(persegi, tm_kuadran4)
    primitif.basic.draw_bentuk(persegi,3,c)
```

x dan y sebagai titik pusat persegi, inisiasi matriks untuk digunakan pada transformasi. Sebelumnya, x dan y koordinat kartesian dikonvert.

a) xo, yo = primitif.utility.convert_to_cartesian(x, y, py5.width, py5.height, 25): dua nilai koordinat x dan y, diambil karena dapat memungkinkan untuk mewakili suatu titik dalam sistem koordinat di layar. Nilai, py5.width dan py5.height, mewakili lebar dan tinggi. Nilai 25 untuk menggeser titik awal dalam sistem koordinat layar.
b) xo dan yo untuk menyimpan hasil konversi koordinat, fungsi primitif.utility.convert_to_cartesian untuk mengonversi koordinat layar jadi sistem kartesian.
c) xc, yc = (xo + x, yo - y): nilai xo ke x untuk menggeser titik horizontal dan mengurangkan nilai y dari yo untuk menggeser titik vertikal

pemanggilan persegi di setiap kuadran dilakukan di main dengan memperhatikan parameter yang ada di basic-nya. Nilai xc dan x dapat diubah untuk mendapatkan pemetaan yang diinginkan.

output



Di atas adalah outpunya, terbentuk 4 persegi, kuadran 1 menggunakan line_type 0, kuadran 1 menggunakan line_type 1, kuadran 1 menggunakan line_type 2, kuadran 1 menggunakan line_type 3. Untuk penjelasan transformasi akan dijelaskan kemudian.

2. Persegi Panjang

basic.py

```
99  def persegi_panjang(xa, ya, panjang, lebar):
100     return np.concatenate(
101         (
102             primitif.line.line_bresenham(xa,ya,xa+panjang,ya)
103             , primitif.line.line_bresenham(xa+panjang,ya,xa+panjang,ya+lebar)
104             , primitif.line.line_bresenham(xa+panjang,ya+lebar,xa,ya+lebar)
105             , primitif.line.line_bresenham(xa,ya+lebar,xa,ya)
106             ),
107         axis=0
108         )
```

Menggunakan method line_bresenham untuk menggmbar setiap sisi, penjelasan kurang lebih sama seperti pada bangun datar persegi. Parameter ditambahkan karena ukuran panjang dan lebar akan berbeda.

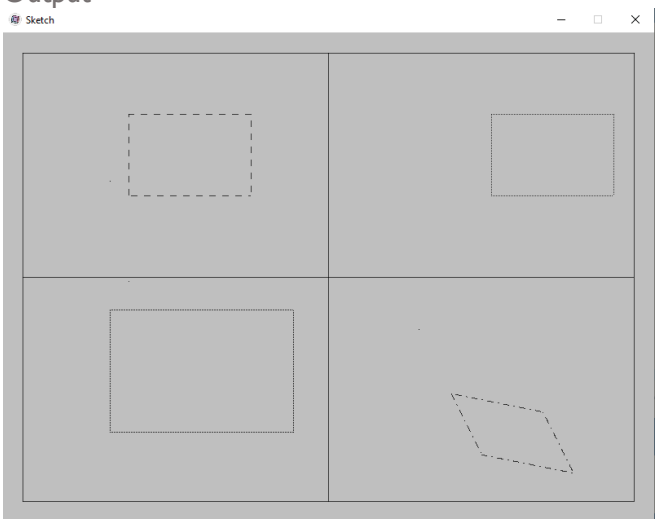
main.py

```
72  #======persegi panjang======#
73
74      x, y = 50, 100
75      panjang = 150
76      lebar = 100
77      tm =np.zeros(3) #inisiasi matriks
78
79      #convert x dan y ke koordinat kartesian
80      xo,yo = primitif.utility.convert_to_cartesian(x,y,py5.width,py5.height,25)
81      xc,yc = (xo+x, yo-y)
```

```
    #kuadran1
    persegi_panjang = primitif.basic.persegi_panjang(xc+100, yc, panjang, lebar)
    primitif.basic.draw_bentuk(persegi_panjang,2,c)

    #kuadran2
    persegi_panjang = primitif.basic.persegi_panjang(x+105,yc, panjang, lebar)
    primitif.basic.draw_bentuk(persegi_panjang,1,c)

    #kuadran3
    tm_kuadran3 = np.zeros(3)

    tm_kuadran3 = primitif.transformasiv2.rotate2D(180, 0, yc)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.reflectY2D(tm_kuadran3)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.scale2D(1.5,1.5, 0, 0, tm_kuadran3)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.translate2D(x-150, 2.2*yo, tm_kuadran3)
    print(tm_kuadran3)
    persegi_panjang = primitif.transformasiv2.transformPoints2D(persegi_panjang, tm_kuadran3)
    primitif.basic.draw_bentuk(persegi_panjang,0,c)

    #kuadran4
    tm_kuadran4 = np.zeros(3)

    tm_kuadran4 = primitif.transformasiv2.rotate2D(0, 0, yc)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.scale2D(0.5, 0.5, 0, 0, tm_kuadran4)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.translate2D(xc-200, yo, tm_kuadran4)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.shear2D(0.5, 0.2, tm_kuadran4)
    persegi_panjang = primitif.transformasiv2.transformPoints2D(persegi_panjang, tm_kuadran4)
    primitif.basic.draw_bentuk(persegi_panjang,3,c)
```

X, y adalah titik pusat persegi panjangnya. Convert juga koordinat kartesiannya. pemanggilan persegi di setiap kuadran dilakukan di main dengan memperhatikan parameter yang ada di basic-nya. Nilai xc dan x dapat diubah untuk mendapatkan pemetaan yang diinginkan.

Output



3. Segitiga siku-siku
   basic.py

```python
def segitiga_siku(xa, ya, alas, tinggi):
    return np.concatenate(
        (
            primitif.line.line_bresenham(xa,ya,xa+alas,ya)
            , primitif.line.line_bresenham(xa+alas,ya,xa,ya-tinggi)
            , primitif.line.line_bresenham(xa,ya-tinggi,xa,ya)
            ),
        axis=0
        )
```

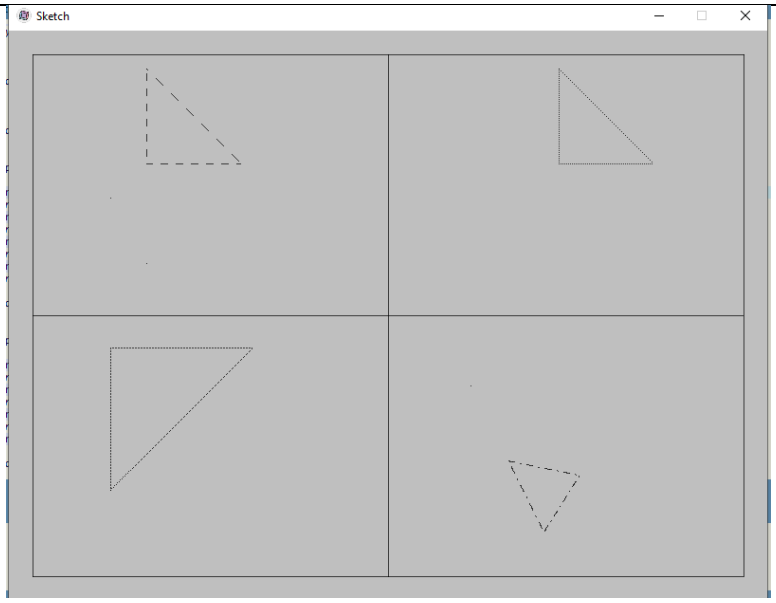
dibutuhkan tiga sisi untuk menggambar lingkaran

main.py

```python
#======segitiga siku siku======#

    x, y = 40, 80   # Ganti dengan koordinat yang diinginkan
    alas= 100
    tinggi=100

    #convert x dan y ke koordinat kartesian
    xo,yo = primitif.utility.convert_to_cartesian(x,y,py5.width,py5.height,25)
    xc,yc = (xo+x, yo-y)

    #kuadran1
    segitiga_siku = primitif.basic.segitiga_siku(xc+100, yc, alas, tinggi)
    primitif.basic.draw_bentuk(segitiga_siku,2,c)

    #kuadran2
    segitiga_siku = primitif.basic.segitiga_siku(x+105,yc, alas, tinggi)
    primitif.basic.draw_bentuk(segitiga_siku,1,c)

    #kuadran3
    tm_kuadran3 = np.zeros(3)

    tm_kuadran3 = primitif.transformasiv2.rotate2D(180, 0, yc)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.reflectY2D(tm_kuadran3)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.scale2D(1.5,1.5, 0, 0, tm_kuadran3)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.translate2D(x-150, 1.2*yo, tm_kuadran3)
    print(tm_kuadran3)
    segitiga_siku = primitif.transformasiv2.transformPoints2D(segitiga_siku, tm_kuadran3)
    primitif.basic.draw_bentuk(segitiga_siku,0,c)

    #kuadran4
    tm_kuadran4 = np.zeros(3)

    tm_kuadran4 = primitif.transformasiv2.rotate2D(0, 0, yc)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.scale2D(0.5, 0.5, 0, 0, tm_kuadran4)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.translate2D(xc-200, yo, tm_kuadran4)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.shear2D(0.5, 0.2, tm_kuadran4)
    segitiga_siku = primitif.transformasiv2.transformPoints2D(segitiga_siku, tm_kuadran4)
    primitif.basic.draw_bentuk(segitiga_siku,3,c)
```

Fungsi dipanggil di main, xc dan x dapat diubah untuk menyesuaikan posisi pemetaan. Segitiga siku siku membutuhkan parameter alas dan tinggi

Output



Tipe tipe line juga dpat ditukar/diganti jika ada garis yang tidak berhasil dimunculkan dengan tipe line tertentu

4. Trapesium siku-siku

basic.py

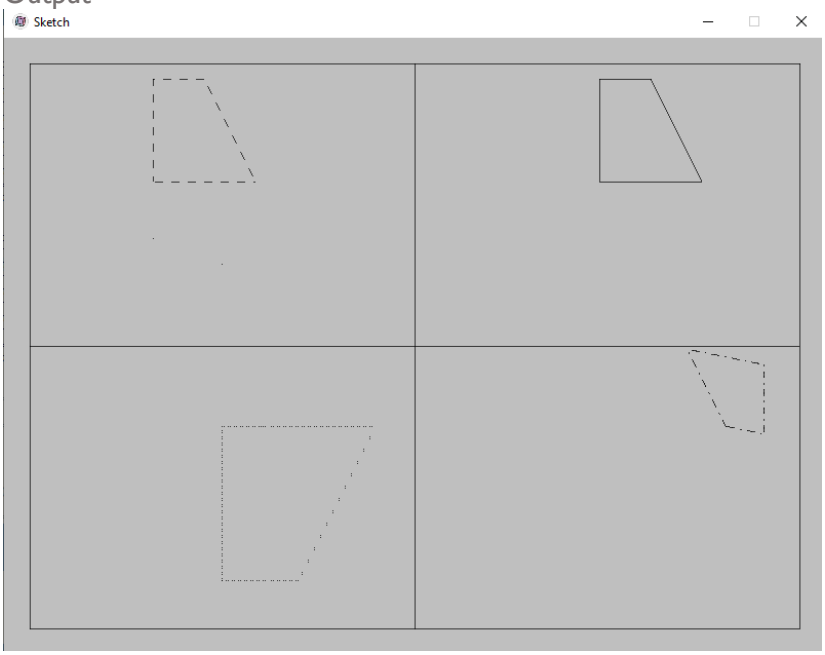
```
def trapesium_siku(xa, ya, aa, ab, tinggi):
    return np.concatenate(
        (
            primitif.line.line_bresenham(xa,ya,xa,ya-tinggi)
            , primitif.line.line_bresenham(xa,ya-tinggi,xa+aa,ya-tinggi)
            , primitif.line.line_bresenham(xa+aa,ya-tinggi,xa+ab,ya)
            , primitif.line.line_bresenham(xa+ab,ya,xa,ya)
            ),
        axis=0
        )
```

main.py

```
    x, y= 40, 80
    sisi1=50
    sisi2=100
    tinggi=100
    tm = np.zeros(3) #deklar matriks
     # Convert x dan y ke koordinat kartesian
    xo,yo = primitif.utility.convert_to_cartesian(x,y,py5.width,py5.height,25)
    xc,yc = (xo+x, yo-y)

    #kuadran1
    trapesium_siku=primitif.basic.trapesium_siku(xc+100, yc, sisi1, sisi2, tinggi)
    primitif.basic.draw_bentuk(trapesium_siku,0,c)

    #kuadran 2
    trapesium_siku=primitif.basic.trapesium_siku(x+105, yc, sisi1, sisi2, tinggi)
    primitif.basic.draw_bentuk(trapesium_siku,1,c)

    #kuadran 3a
    tm_kuadran3 = np.zeros(3)
    tm_kuadran3 = primitif.transformasiv2.rotate2D(180, 0, yc)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.reflectY2D(tm_kuadran3)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.scale2D(1.5,1.5, 0, 0, tm_kuadran3)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.translate2D(x-45, 1.4*yo, tm_kuadran3)
    print(tm_kuadran3)
    trapesium_siku = primitif.transformasiv2.transformPoints2D(trapesium_siku, tm_kuadran3)
    primitif.basic.draw_bentuk(trapesium_siku,2,c)

    # Kuadran 4
    tm_kuadran4 = np.zeros(3)
    tm_kuadran4 = primitif.transformasiv2.rotate2D(0, 0, yc)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.scale2D(0.5, 0.5, 0, 0, tm_kuadran4)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.translate2D(xc-15, 0, tm_kuadran4)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.shear2D(0.5, 0.2, tm_kuadran4)
    trapesium_siku = primitif.transformasiv2.transformPoints2D(trapesium_siku, tm_kuadran4)
    primitif.basic.draw_bentuk(trapesium_siku,3,c)
```

Pastsikan ukuran sisi1 dan ssisi2 berbeda agar tidak membentuk persegi/persegi panjang

Output



5. Lingkaran

basic.py

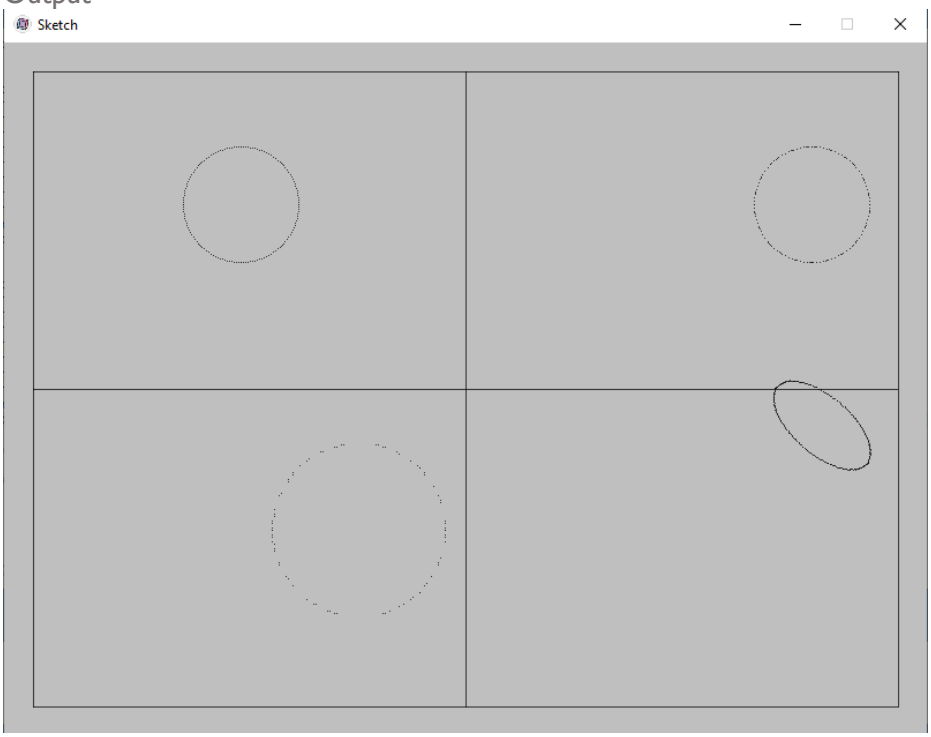
```
def circlePlotPoints(xc, yc, x, y):
    res = [
        [xc + x, yc + y],
        [xc - x, yc + y],
        [xc + x, yc - y],
        [xc - x, yc - y],
        [xc + y, yc + x],
        [xc - y, yc + x],
        [xc + y, yc - x],
        [xc - y, yc - x],
        ]
    return res

def lingkaran(xc, yc, radius):
    x = 0
    y = radius
    p = 1 - radius
    res = circlePlotPoints(xc, yc, x, y)
    while(x < y):
        x+=1
        if (p < 0):
            p+= 2*x + 1
        else:
            y-=1
            p+= 2*(x-y) + 1

        res = np.concatenate(
            (
                res
                , circlePlotPoints(xc, yc, x, y)
            ), axis=0)


    return res

```

Lingkaran memerlukan circle points untuk memetakan 8 titik titik agar bisa membentuk lingkaran

main.py

```
    x, y= 100, 80
    radius=50

    tm = np.zeros(3) #deklar matriks

    # Convert x dan y ke koordinat kartesian
    xo,yo = primitif.utility.convert_to_cartesian(x,y,py5.width,py5.height,25)
    xc,yc = (xo+x, yo-y)
#kuadran1
lingkaran=primitif.basic.lingkaran(xc+100, yc, radius)
primitif.basic.draw_bentuk(lingkaran,3,c)

#kuadran 2
lingkaran=primitif.basic.lingkaran(x+105, yc, radius)
primitif.basic.draw_bentuk(lingkaran,1,c)

#kuadran 3
tm_kuadran3 = np.zeros(3)
tm_kuadran3 = primitif.transformasiv2.rotate2D(180, 0, yc)
print(tm_kuadran3)
tm_kuadran3 = primitif.transformasiv2.reflectY2D(tm_kuadran3)
print(tm_kuadran3)
tm_kuadran3 = primitif.transformasiv2.scale2D(1.5,1.5, 0, 0, tm_kuadran3)
print(tm_kuadran3)
tm_kuadran3 = primitif.transformasiv2.translate2D(x-100, 1.6*yo, tm_kuadran3)
print(tm_kuadran3)
lingkaran = primitif.transformasiv2.transformPoints2D(lingkaran, tm_kuadran3)
primitif.basic.draw_bentuk(lingkaran,2,c)

# Kuadran 4
tm_kuadran4 = np.zeros(3)
tm_kuadran4 = primitif.transformasiv2.rotate2D(0, 0, yc)
print(tm_kuadran4)
tm_kuadran4 = primitif.transformasiv2.scale2D(0.5, 0.5, 0, 0, tm_kuadran4)
print(tm_kuadran4)
tm_kuadran4 = primitif.transformasiv2.translate2D(xc-150, 0, tm_kuadran4)
print(tm_kuadran4)
tm_kuadran4 = primitif.transformasiv2.shear2D(0.5, 0.2, tm_kuadran4)
lingkaran = primitif.transformasiv2.transformPoints2D(lingkaran, tm_kuadran4)
primitif.basic.draw_bentuk(lingkaran,0,c)
```

Output



6. Ellips

basic.py

```
def ellips(xc, yc, Rx, Ry):
    Rx2 = Rx * Rx
    Ry2 = Ry * Ry
    twoRx2 = 2 * Rx2
    twoRy2 = 2 * Ry2
    p = 0
    x = 0
    y = Ry
    px = 0
    py = twoRx2 * y

    res = ellipsePlotPoints(xc, yc, x, y)

    p = round(Ry2 - (Rx2 * Ry) + (0.25 * Rx2))
    while px < py:
        x += 1
        px += twoRy2
        if p < 0:
            p += Ry2 + px
        else:
            y -= 1
            py -= twoRx2
            p += Ry2 + px - py
        res = np.concatenate(
            (
                res
                , ellipsePlotPoints(xc, yc, x, y)
            ), axis=0)

    p = round(Ry2 * (x + 0.5) * (x + 0.5) + Rx2 * (y - 1) * (y - 1) - Rx2 * Ry2)
    while y > 0:
        y -= 1
        py -= twoRx2
        if p > 0:
            p += Rx2 - py
        else:
```

```
      p = round(Ry2 * (x + 0.5) * (x + 0.5) + Rx2 * (y - 1) * (y - 1) - Rx2 * Ry2)
      while y > 0:
          y -= 1
          py -= twoRx2
          if p > 0:
              p += Rx2 - py
          else:
              x += 1
              px += twoRy2
              p += Rx2 - py + px
          res = np.concatenate(
              (
                  res
                  , ellipsePlotPoints(xc, yc, x, y)
              ), axis=0)

      return res
```

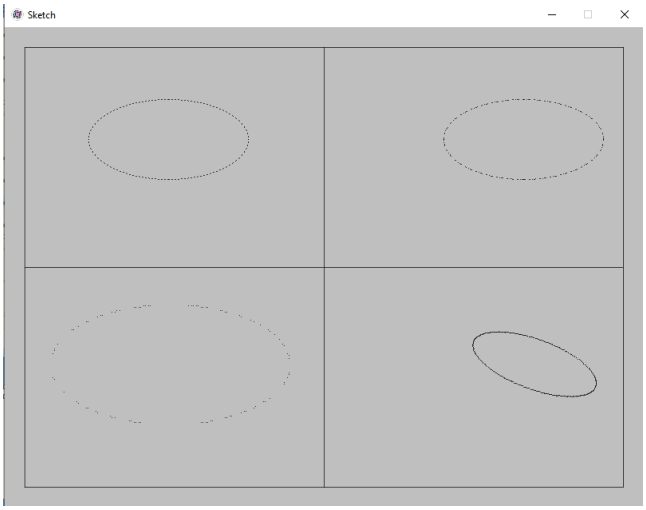
Perlu 4 titik untuk membentuk ellips

main.py

```
    x, y= 100, 80
    radiusx=100
    radiusy=50

    tm = np.zeros(3) #deklar matriks

    # Convert x dan y ke koordinat kartesian
    xo,yo = primitif.utility.convert_to_cartesian(x,y,py5.width,py5.height,25)
    xc,yc = (xo+x, yo-y)
```

```
    #kuadran1
    ellips=primitif.basic.ellips(xc+50, yc, radiusx, radiusy)
    primitif.basic.draw_bentuk(ellips,3,c)

    #kuadran 2
    ellips=primitif.basic.ellips(x+105, yc, radiusx, radiusy)
    primitif.basic.draw_bentuk(ellips,1,c)

    #kuadran 3
    tm_kuadran3 = np.zeros(3)
    tm_kuadran3 = primitif.transformasiv2.rotate2D(180, 0, yc)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.reflectY2D(tm_kuadran3)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.scale2D(1.5,1.5, 0, 0, tm_kuadran3)
    print(tm_kuadran3)
    tm_kuadran3 = primitif.transformasiv2.translate2D(x-200, 1.6*yo, tm_kuadran3)
    print(tm_kuadran3)
    ellips = primitif.transformasiv2.transformPoints2D(ellips, tm_kuadran3)
    primitif.basic.draw_bentuk(ellips,2,c)

    # Kuadran 4
    tm_kuadran4 = np.zeros(3)
    tm_kuadran4 = primitif.transformasiv2.rotate2D(0, 0, yc)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.scale2D(0.5, 0.5, 0, 0, tm_kuadran4)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.translate2D(xc-200, 0.5*yo, tm_kuadran4)
    print(tm_kuadran4)
    tm_kuadran4 = primitif.transformasiv2.shear2D(0.5, 0.2, tm_kuadran4)
    ellips = primitif.transformasiv2.transformPoints2D(ellips, tm_kuadran4)
    primitif.basic.draw_bentuk(ellips,0,c)
```

Output



TASK 5 MEMBUAT FUNGSI-FUNGSI TRANSFORMASI 2D,

1. Copy Task 5, buatlah fungsi-fungsi transformasi 2D dengan operasi matrix yang dibantu dengan numpy dengan menggunakan homogeneous coordinate dengan titik (0,0) adalah origin monitor.
2. Buatlah Translasi, Scaling, Shear, dan Rotasi

Lesson Learnt (Code, Print Screen Hasil Karya, dan Komentar)

1. Translasi

```python
import math
import numpy as np

def translate2D(tx, ty, tm = np.zeros(3)):
    # Matriks transformasi translasi
    translation_matrix = np.array([[1, 0, tx],
                                   [0, 1, ty],
                                   [0, 0, 1]])

    # Mengalikan matriks transformasi dengan matriks transformasi sebelumnya
    tm = np.dot(translation_matrix, tm)

    return tm
```

2. Scaling

```python
def scale2D(sx, sy, refx, refy, tm=np.zeros(3)):
    # Matriks transformasi scaling
    scaling_matrix = np.array([[sx, 0, (1-sx) * refx],
                               [0, sy, (1-sy) * refy],
                               [0, 0, 1]])
    tm = np.dot(tm, scaling_matrix)
    return tm
```

3. Shear

```python
def shear2D(shx, shy, tm = np.zeros(3)):
    shear_matrix = np.array([[1, shx, 0],
                             [shy, 1, 0],
                             [0, 0, 1]])

    return np.dot(shear_matrix, tm)
```

4. Rotasi

```python
def rotate2D(a, refx, refy):
    # Menghitung sudut dalam radian
    angle_rad = math.radians(a)
    rotation_matrix = np.array([[math.cos(angle_rad), -math.sin(angle_rad), refx * (1-math.cos(angle_rad)) + refy * math.sin(angle_rad)],
                                [math.sin(angle_rad), math.cos(angle_rad), refy * (1-math.cos(angle_rad)) - refx * math.sin(angle_rad)],
                                [0, 0, 1]])
    tm = rotation_matrix
    return tm
```

Dapat dilihat di oenjelasan task 4 bahwa kuadran 3 dan 4 saya lakukan beberapa transformasi. Pemanggilannya di main cukup memperhatikan parameter dari method mthod transformasi. Saya menginisiasi setiap matriks di kuadran 3 dan 4 agar tidak saling terkait, jadi dibuat matriks baru.

Transformasi yang sdah diubah ke python

```python
import math

# Definisikan matriks
def matrix3x3SetIdentity(m):
    for i in range(3):
        for j in range(3):
            m[i][j] = 1 if i == j else 0

# Kalikan dua matriks 3x3
def matrix3x3PreMultiply(a, b):
    tmp = [[0 for _ in range(3)] for _ in range(3)]
    for r in range(3):
        for c in range(3):
            tmp[r][c] = (
                a[r][0] * b[0][c] +
                a[r][1] * b[1][c] +
                a[r][2] * b[2][c]
            )

    for r in range(3):
        for c in range(3):
            b[r][c] = tmp[r][c]

# Inisialisasi matriks transformasi
theMatrix = [[0 for _ in range(3)] for _ in range(3)]
matrix3x3SetIdentity(theMatrix)

# Fungsi translasi 2D
def translate2D(tx, ty):
    global theMatrix
    m = [[0 for _ in range(3)] for _ in range(3)]
    matrix3x3SetIdentity(m)
    m[0][2] = tx
    m[1][2] = ty
    matrix3x3PreMultiply(m, theMatrix)

# Fungsi penskalaan 2D
def scale2D(sx, sy, refpt):
    global theMatrix
    m = [[0 for _ in range(3)] for _ in range(3)]
    matrix3x3SetIdentity(m)
    m[0][0] = sx
    m[0][2] = (1 - sx) * refpt[0]
    m[1][1] = sy
    m[1][2] = (1 - sy) * refpt[1]
    matrix3x3PreMultiply(m, theMatrix)
```

```python
# Fungsi rotasi 2D
def rotate2D(a, refpt):
    global theMatrix
    m = [[0 for _ in range(3)] for _ in range(3)]
    matrix3x3SetIdentity(m)
    a = math.radians(a)
    m[0][0] = math.cos(a)
    m[0][1] = -math.sin(a)
    m[0][2] = refpt[0] * (1 - math.cos(a)) + refpt[1] * math.sin(a)
    m[1][0] = math.sin(a)
    m[1][1] = math.cos(a)
    m[1][2] = refpt[1] * (1 - math.cos(a)) - refpt[0] * math.sin(a)
    matrix3x3PreMultiply(m, theMatrix)

# Fungsi transformasi titik-titik 2D
def transformPoints2D(pts):
    global theMatrix
    transformed_pts = []
    for pt in pts:
        x = theMatrix[0][0] * pt[0] + theMatrix[0][1] * pt[1] + theMatrix[0][2]
        y = theMatrix[1][0] * pt[0] + theMatrix[1][1] * pt[1] + theMatrix[1][2]
        transformed_pts.append((x, y))
    return transformed_pts

# Contoh penggunaan
if __name__ == "__main__":
    # Definisikan titik-titik awal
    points = [(1, 1), (2, 2), (3, 3)]

    # Terapkan transformasi translasi
    translate2D(1, 2)
    transformed_points = transformPoints2D(points)
    print("Hasil transformasi translasi:", transformed_points)

    # Reset matriks transformasi ke identitas
    matrix3x3SetIdentity(theMatrix)

    # Terapkan transformasi penskalaan
    scale2D(2, 0.5, (1, 1))
    transformed_points = transformPoints2D(points)
    print("Hasil transformasi penskalaan:", transformed_points)

    # Reset matriks transformasi ke identitas
    matrix3x3SetIdentity(theMatrix)

    # Terapkan transformasi rotasi
    rotate2D(45, (1, 1))
    transformed_points = transformPoints2D(points)
    print("Hasil transformasi rotasi:", transformed_points)
```

saya belum sempat mengubah menjadi OOP

PENGUMPULAN

Ikuti Format yang diberikan di Google Classroom.