MODUL III

KOMPUTER GRAFIK 2D
BENTUK DASAR LANJUTAN DAN ATRIBUT GARIS

D3 TEKNIK INFORMATIKA
JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
POLITEKNIK NEGERI BANDUNG

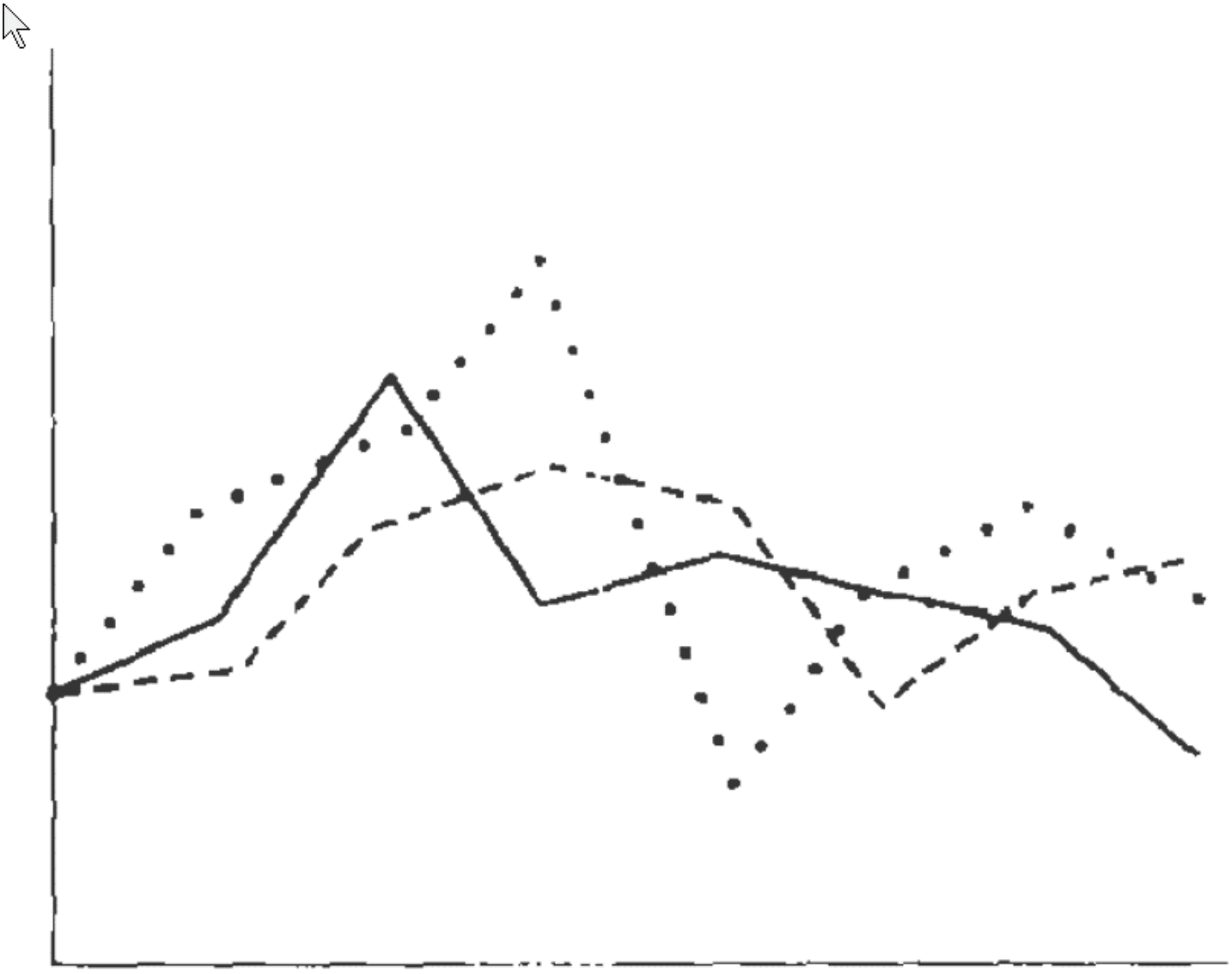
ALQAN NAZRA 068 | KOMPUTER GRAFIK | SEPTEMBER, 2 2024

CONTENTS

ATRIBUTE GARIS 1
TIC TAC TOE GAME 1
TASK PRAKTIKUM 2
PENGUMPULAN 17

ATRIBUTE GARIS

RAGAM ATTRIBUTE GARIS



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)

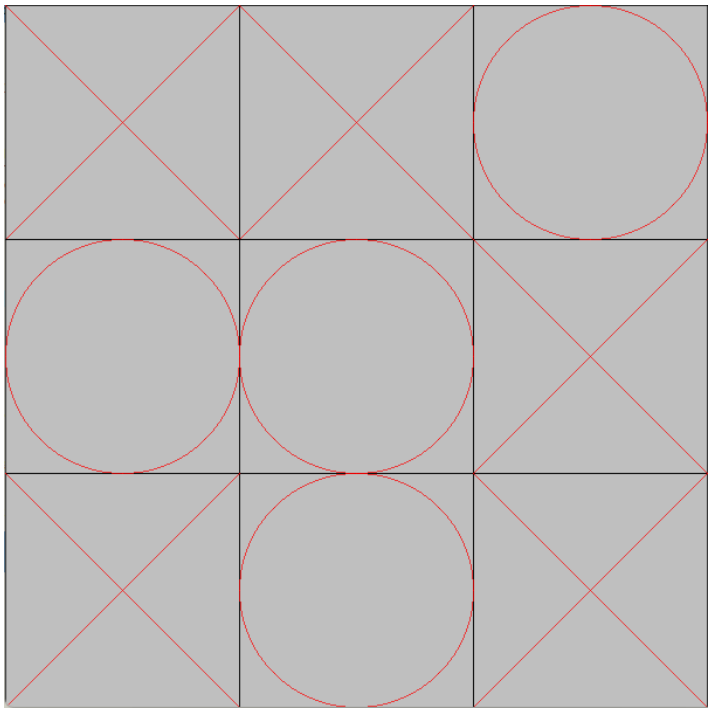
TASK PRAKTIKUM

TASK 1-2: PEMBELAJARAN OOP PYTHON DENGAN PROCESSING

1. Amati implementasi code untuk membuat tictactoe, Tictactoe, Asteroid, Stick Man, Kendaraan
2. Setelah diamati, modifikasi kode untuk memahami maksud dari OOP dan tuliskan temuan yang didapatkan.

Lesson Learnt (Hasil Karya, dan Komentar)

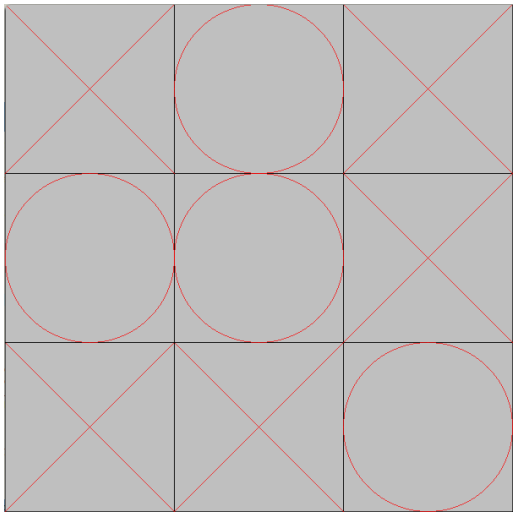
Task 1 Main 1

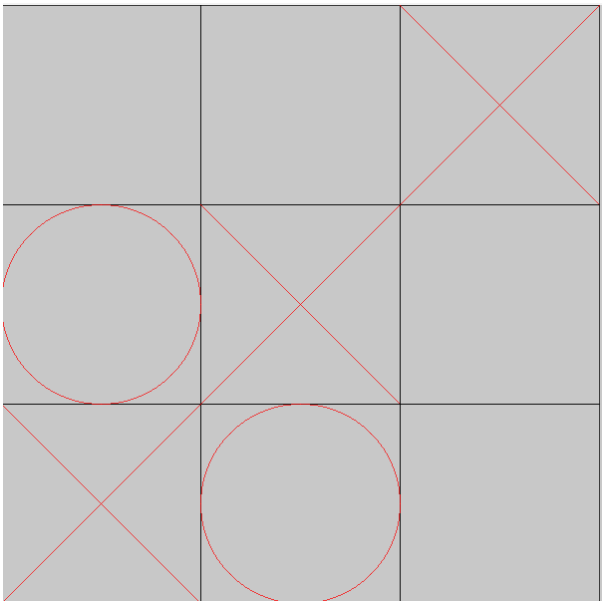


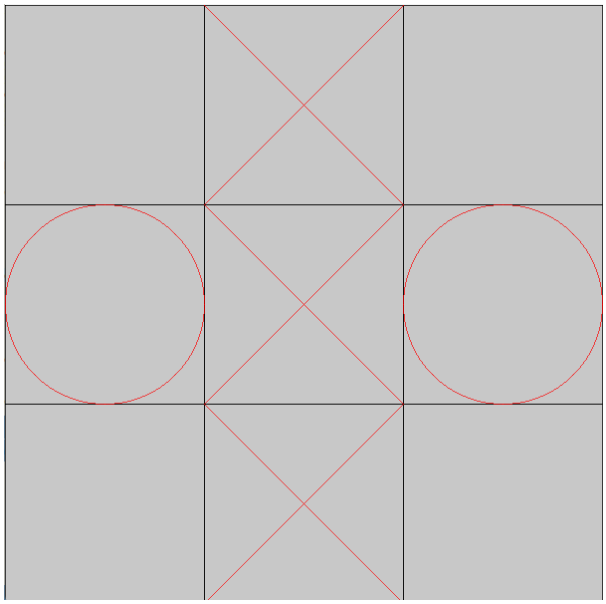
Pada program ini terdapat masalah dimana player satu dapat mengubah-ngubah bentuknya ini membuat game menjadi bug untuk menyelesaikan masalah ini saya mengubah kondisi saat mouse board kosong maka nilai turn akan nol begiitu X = 1 dan O = 2 dimana board ini memiliki nilai dimana mouse haru mengentahui bahwa bila board tidak nol makai a tidak bisa mengubah board itu dengan melakukan ini Variabel board == 0 maka mouse tidak dapat mengubah nilai dari board karena nilai variable board selalu terisi sehingga tidak dapat diubah oleh mouse.

Task 2

HASIL







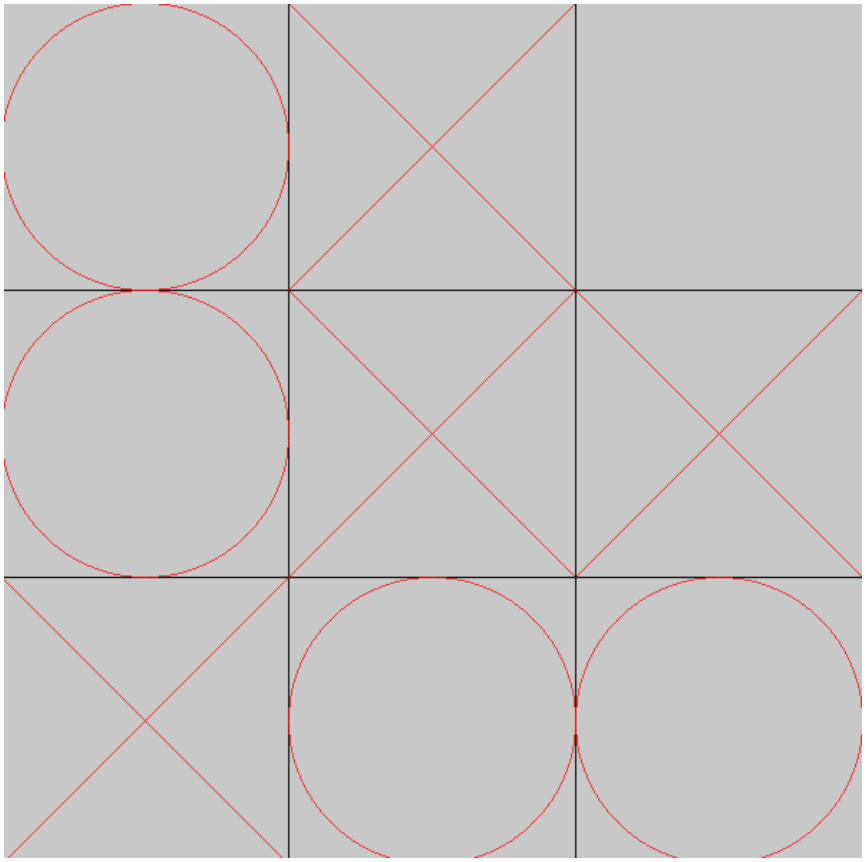
Pada program main2 ini memiliki masalah dimana progaram ini tidak dapat menentukan kemenangan bila kondisi kemenangan vertikal dan tegak lurus serta tidak ada kondisi draw, untuk menyelsaikan masalah ini maka kita menambahkan kondisi baru dengan menambang kondisi box terisi dengan dengan turn yang sama saat vertikal dan tegak lurus seperti ini

```
def find_winners():
    for y in range(3):
        if grid[y][0] == grid[y][1] == grid[y][2] != 0:
            return grid[y][0]
    for x in range(3):
        if grid[0][x] == grid[1][x] == grid[2][x] != 0:
            return grid[0][x]
    if grid[0][0] == grid[1][1] == grid[2][2] != 0:
        return grid[0][0]
    if grid[0][2] == grid[1][1] == grid[2][0] != 0:
        return grid[0][2]
    return None
```

Serta menambahkan logika baru untuk draw dimana ketika fungsi winner ini tidak bekerja atau tidak mengembalikan nilai dan ketika semua row terisi maka turn akan terisi nilai 3 dengan fungsi draw ini bertipe Boolean dimana akan berjalan Ketika fungsi ini mengembalikan nilai true. Seperti dibawah ini

```
def find_draw():

    if find_winners() is not None:
        return False

    for row in grid:
        if 0 in row:
            return False

    return True
```

TASK 3



Pada main3 ini terdapat masalah dimana AI yang sudah diassign tidak dapat mengisi kotak yang kosong seperti diatas sehingga game akan terhenti diposisi ini untuk menyelesaikan masalah ini maka kita harus membuat AI ini dapat mengetahui kondisi kotak terkini sebelum melakukan aksinya, dikarenakan ai akan aktif bila kita sebagai turn = 1 mulai menyerang maka ai juga akan menyerang maka serta player selalu melakukan input genap maka sementara AI juga memilki input genap membuatnya tidak bisa bergerak saat input ke 5. Dengan abstraksi diatas maka dibuatlah program seperti dibawah.

```
if ai == True:
    kosong = []
    for y in range(3):
        for x in range(3):
            if grid[y][x] == 0:
                kosong.append((x, y))

if kosong:
    index = random.randint(0, len(kosong) - 1)
    x, y = kosong[index]
    grid[y][x] = turn
    switch_turns()
```

Dengan program ini AI memiliki kemampuan untuk mengetahui kondisi dari masing-masing kotak sehingga dia tidak diam saat player sudah berada diindeks 8 dapat disusul oleh AI saat indeksnya 9 sehingga AI dapat bergerak 5 kali.

TASK 2 Membuat Program Tic tac toe menjadi OOP

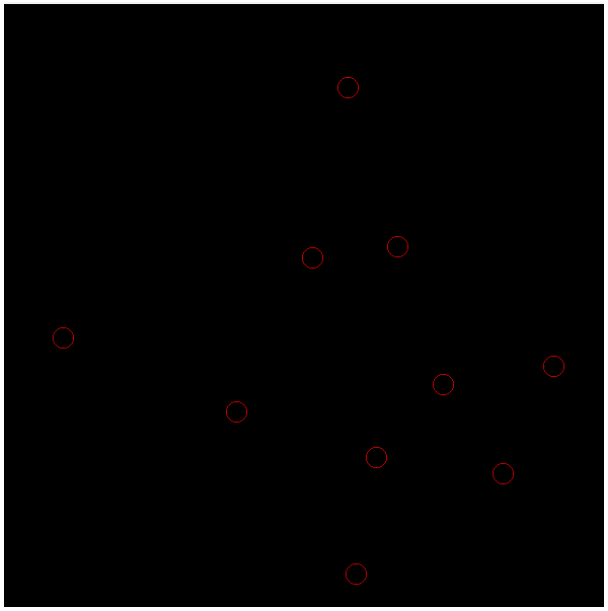
```
0 0 0
0 0 0
0 0 0

Pastikan mengisian memiliki range dari 0-2 bila lebih maka akan error

Kamu itu 1 dan AI adalah 2
Masukkan Koordinat Catur Jawa (x, y):
```

```
Masukkan Koordinat Catur Jawa (x, y): 0 0
1 0 0
0 0 0
0 0 0

^^^ Pergerakan AI ^^^
1 0 0
0 0 0
2 0 0
```

```
Player 1 Wins!
```

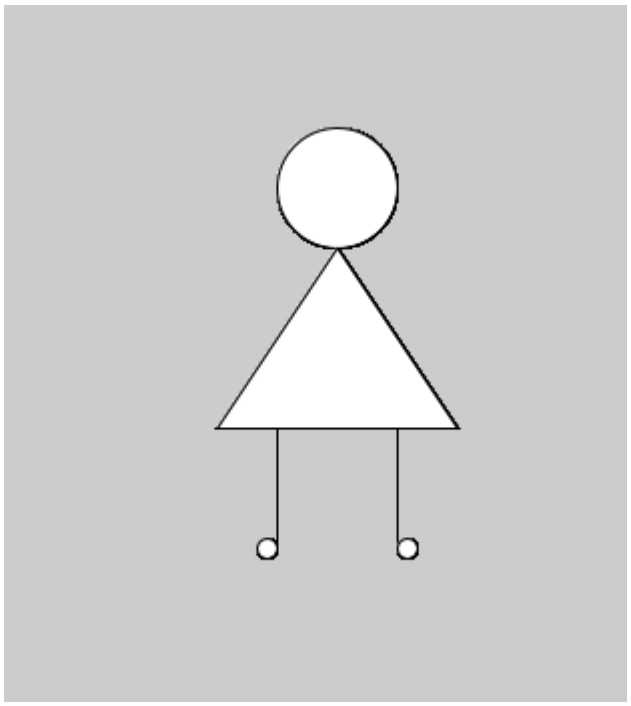
Pada Taks ini saya ditugas kan untuk mengubah program Tictactoe menjadi objeck dan class dimana proses ini dimulai dengen membuat Class dan objeck dimana pada phyton ini berbeda dengan JAVA dalam melakukan assign dimana pada phyton menggunakan __inti___ atau sebuah objek yang menampugn semua variable untuk method lainnya kalo bisa dibilan itu komposit seingga nilai komposit pada program say aitu catur ini dipakai hampir seluruh program yang terdapat pada class caturjawir.

Task 3
Main 1



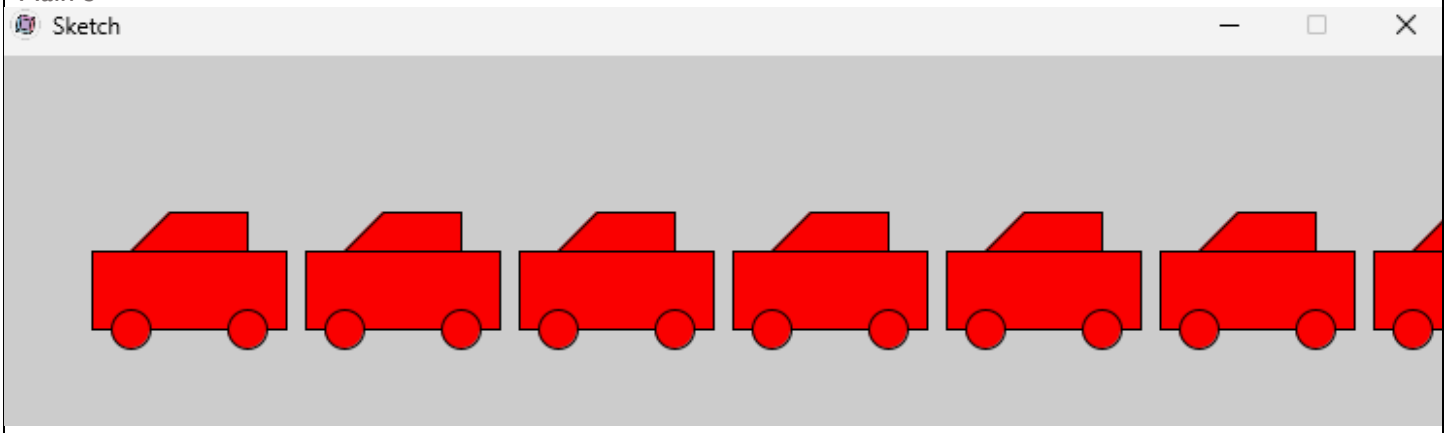
Pada program ini berkerja dengan OOP dimana pada Class Asteroid ini memiliki 2 method yang pertama yaitu method draw yang menggambarkan lingkaran dan method Move dimana 2 method ini memiliki 2 objeck yaitu x dan y 2 objeck ini memiliki fungsi dalam mebentuk dan menggerakan, untuk menggerakan lingkaran ini sebenarnya tidak bergerak namun melakukan print linkarang yang banyak dalam waktu yang cepat proses ini dilakukan dengan looping dimana looping ini menggunakan parameter yang tidak hingga membuatnya kooping terus menerus hingga program dihentikan, lingkaran ini memiliki kecepatan Gerakan kecapatan ini dipengaruhi oleh range random integer yang dimana random integer ini berfungsi untuk menetapkan lingkaran secara random nilai random intg ini akan selalu ditambah 1 sehingga akan bertambah terus menerus yang membuatnya bergerak bila ingin mempengaruhi kecepatannya ubah penambahan yang lebih besar maka lingkaran akan bergerak lebih depat.

Main 2



Pada program ini merupakan implementasi dari pembuatan object dengan memanfaatkan ukuran width dan height menjadi satu kesatuan dimana hal ini dapat dicapai dengan memiliki titik x dan y yang sama satu sama lain dengan dicapai dari bagian ujung dari titik

Main 3



Pada program ini merupakan implemetasi looping object dimana prose looping ini melakuakn looping dengan parameter lebar layar itu sendiri looping ini range didalamnya dikarenkan parameter penghentinnya menggunakan range dimana range ini merupakan parameter yang diatur awal start dan berhentinya dengan penggunaan seperti ini
Range (start,stop,step):
start: nilai awal deret angka (default adalah 0)
stop: nilai akhir deret angka (tidak termasuk dalam deret)
step: langkah antara angka dalam deret (default adalah 1)
dengan looping mobil yang diouput dapat diatur barisnya mulai dari mana dan akan berhenti pada posisi dimana

TASK 4: MEMBUAT FUNGSI BENTUK DASAR

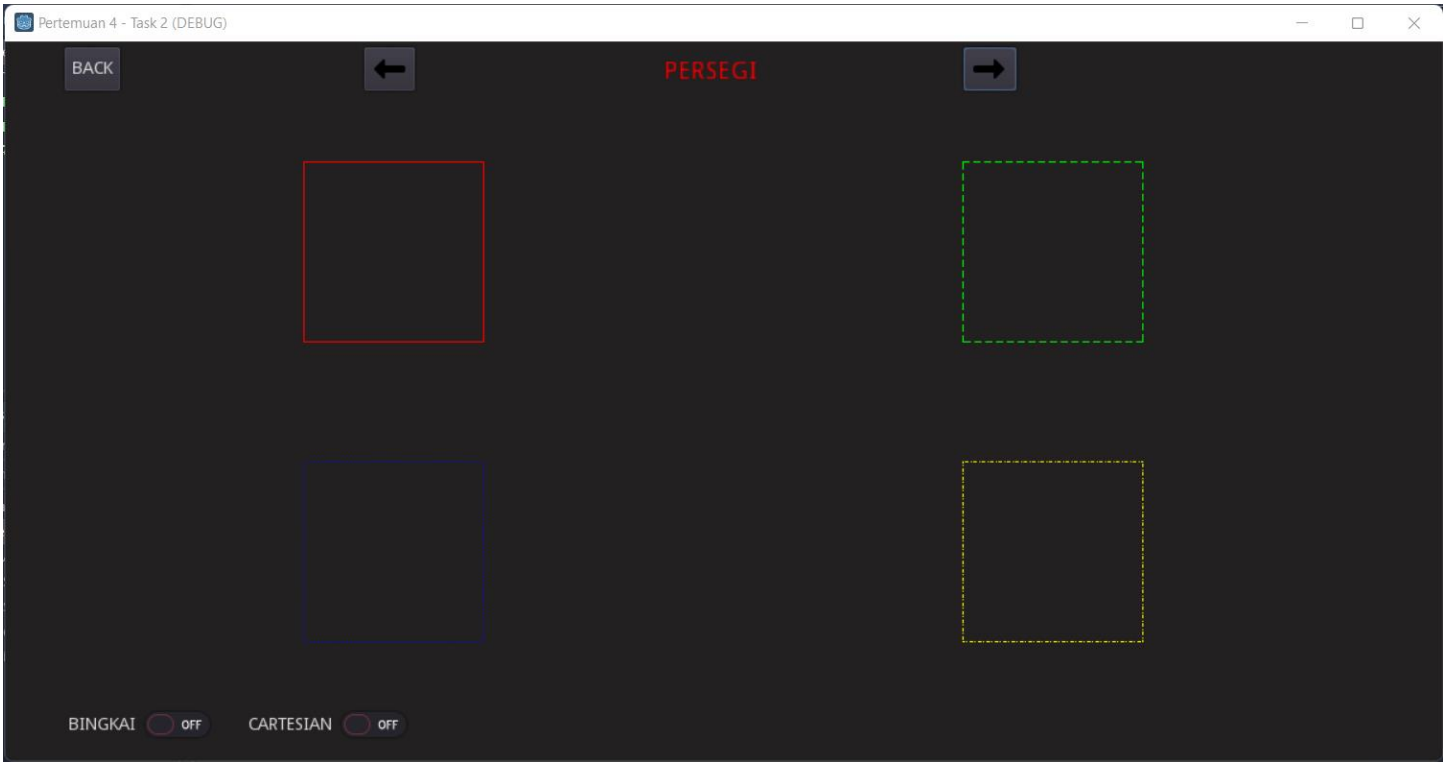
1. Buatlah Class Bentuk Dasar menggunakan algoritma generalisasi line bersenham sbb: Persegi, Persegi Panjang, Segitiga Siku-Siku, dan Trapesium Siku-Siku

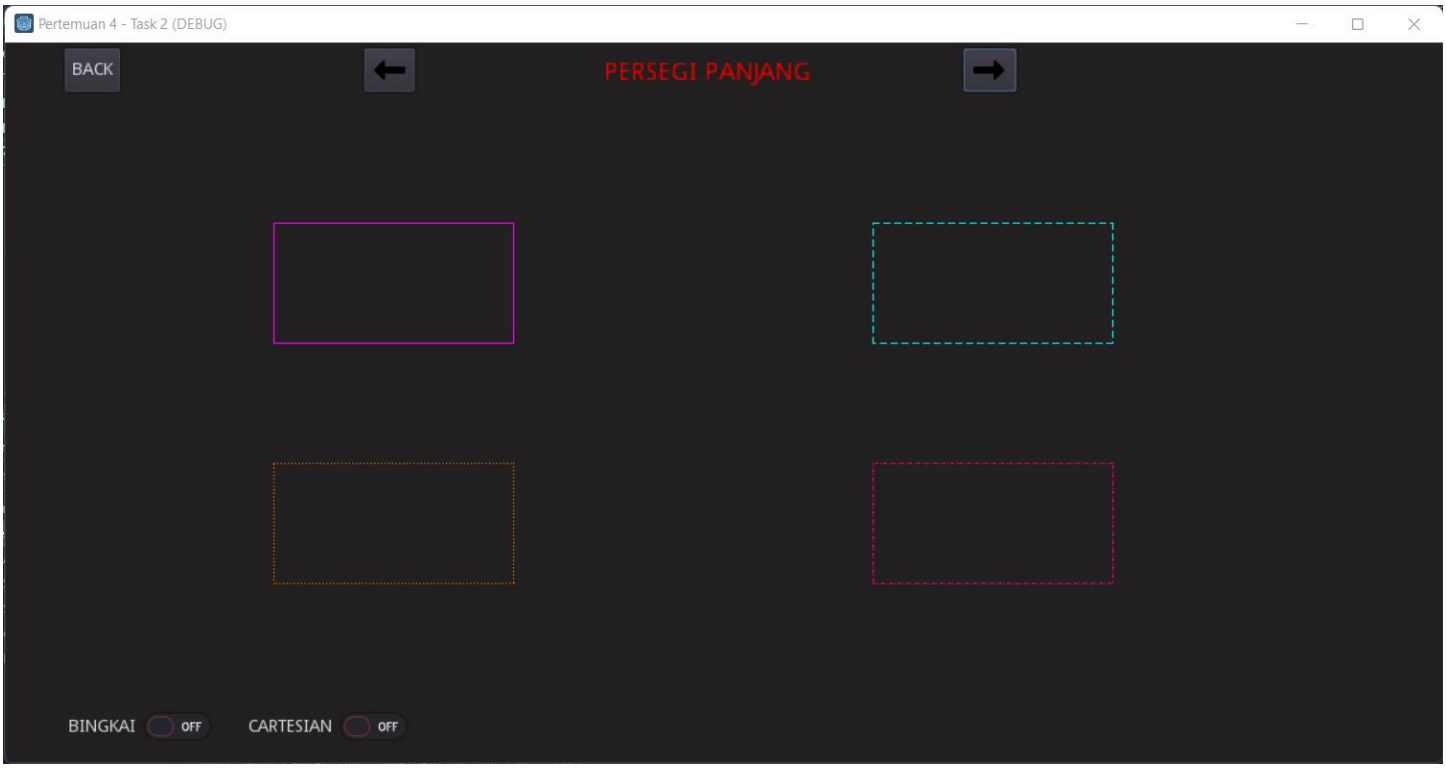
2. Buatlah Ragam Attribute Garis untuk setiap Bentuk Dasar. Hints (jadikan parameter bukan hardcoding, manipulasi dilakukan setelah koordinat garis terbentuk atau manipulasi hasil array of koordinat)
3. Posisikan Bentuk Dasar menjadi 4 Quadran.

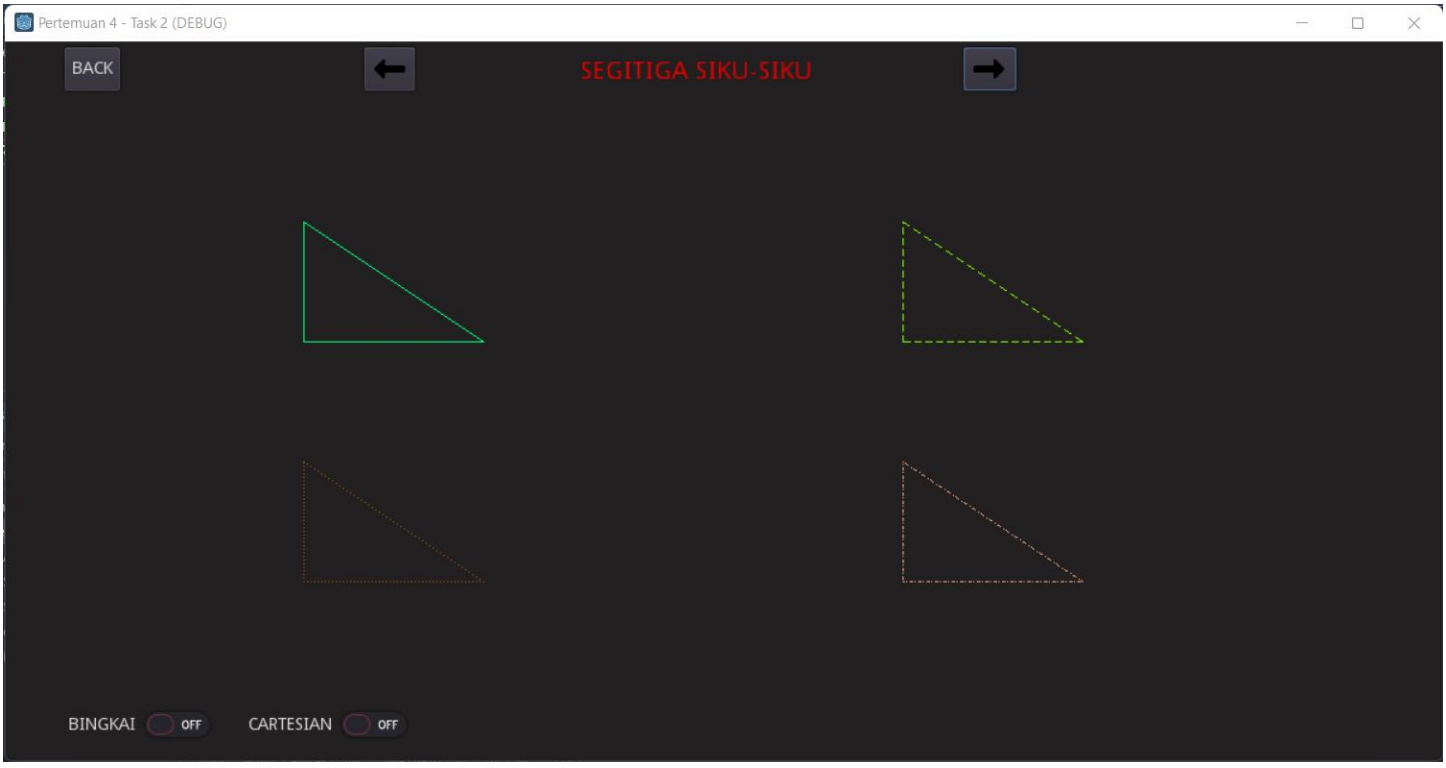
Untuk mengubah posisi dapat menggunakan fungsi convert to cartesian berikut

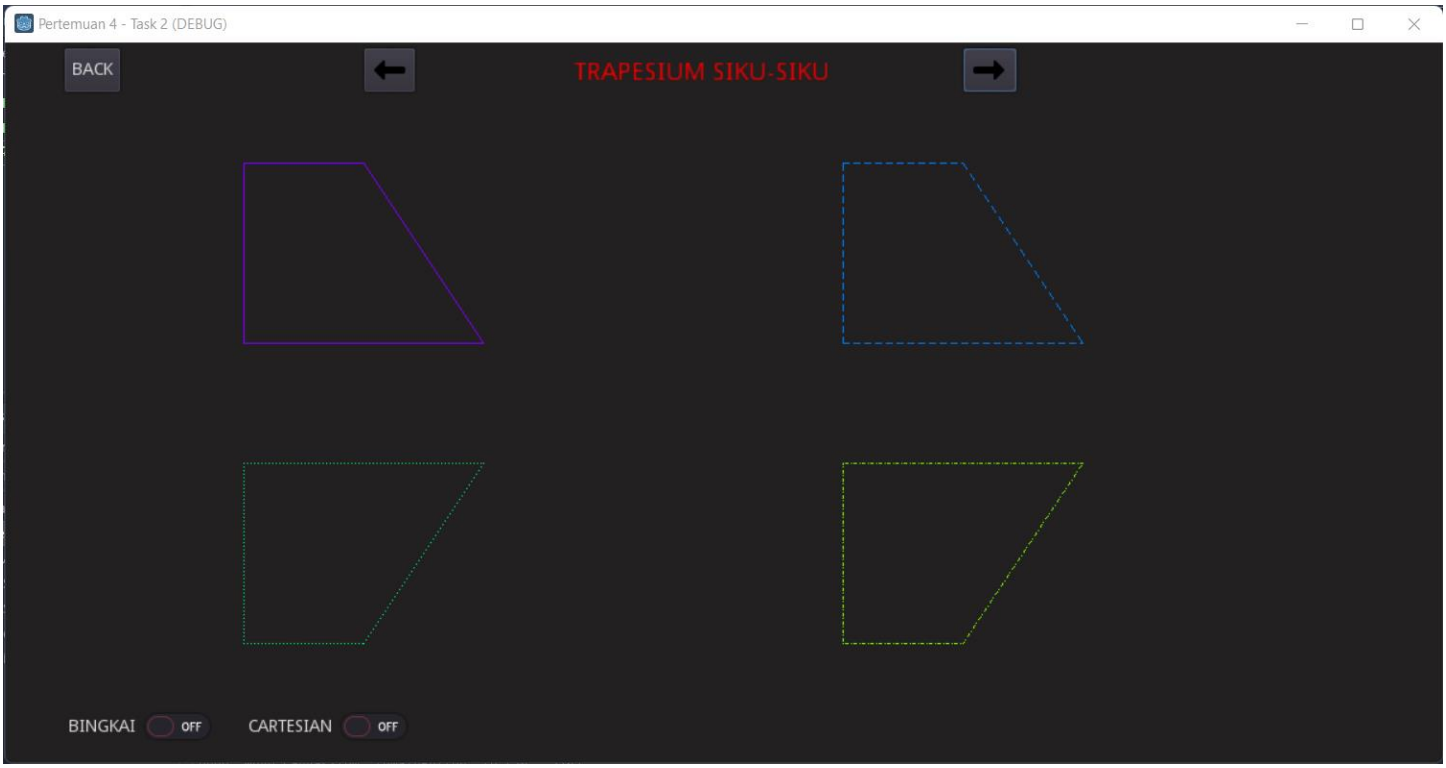
Utility.py

```
import math

def convert_to_pixel(xa, ya, xb, yb, width, height, margin):
    return [margin+xa, height-margin-ya, margin+xb, height-margin-yb]

def convert_to_cartesian(xa, ya, xb, yb, width, height, margin):
    axis = math.ceil(width/2)
    ordinat = math.ceil(height/2)
    return [axis+xa, ordinat-ya, axis+xb, ordinat-yb]
```




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


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



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

CODE

```
import py5
import primitif.line
import primitif.basic
import primitif.utility
import config

class ShapeDrawer:
    def __init__(self, margin, color):
        self.margin = margin
        self.color = color

    def draw_margin(self, width, height):
        py5.stroke(*self.color)
        self.draw_bentuk(primitif.line.line_dda(self.margin, self.margin, width - self.margin, self.margin))
        self.draw_bentuk(primitif.line.line_dda(self.margin, height - self.margin, width - self.margin, height - self.margin))
        self.draw_bentuk(primitif.line.line_dda(self.margin, self.margin, self.margin, height - self.margin))
        self.draw_bentuk(primitif.line.line_dda(width - self.margin, self.margin, width - self.margin, height - self.margin))

    def draw_grid(self, width, height):
        xa = self.margin
        ya = 2 * self.margin
        xb = width - xa
        yb = height - ya
        y_range = height / self.margin

        py5.stroke(*self.color)
        for _ in range(1, int(y_range)):
            self.draw_bentuk(primitif.line.line_dda(xa, ya, xb, ya))
            ya += self.margin

        xa = 2 * self.margin
        ya = self.margin
        xb = width - xa
        yb = height - ya
        x_range = width / self.margin
        for _ in range(1, int(x_range)):
            self.draw_bentuk(primitif.line.line_dda(xa, ya, xa, yb))
            xa += self.margin

    def draw_kartesian(self, width, height):
        py5.stroke(*self.color)
```

```
        self.draw_bentuk(primitif.line.line_dda(width / 2, self.margin, width / 2, height - self.margin))
        self.draw_bentuk(primitif.line.line_dda(self.margin, height / 2, width - self.margin, height / 2))

    def draw_bentuk(self, pts):
        for pt in pts:
            py5.point(pt[0], pt[1])

def setup():
    py5.size(800, 600)
    py5.rect_mode(py5.CENTER)

def draw():
    width = 800
    height = 600
    py5.background(191)
    color = [0, 0, 0, 255]
    margin = 25
    drawer = ShapeDrawer(margin, color)
    drawer.draw_margin(width, height)
    drawer.draw_kartesian(width, height)

    half_width = width // 2
    half_height = height // 2

    if config.anim <= 3 * config.times:
        primitif.basic.persegi(6 * half_width // 4, half_height // 2, 100, c=[255, 0, 0, 255]) # 1
        primitif.basic.persegi_2(half_width // 2, half_height // 2, 100, c=[255, 0, 0, 255]) # 2
        primitif.basic.persegi_3(half_width // 2, 3 * half_height // 2, 100, c=[255, 0, 0, 255]) # 3
        primitif.basic.persegi_4(6 * half_width // 4, 3 * half_height // 2, 100, c=[255, 0, 0, 255]) # 4

    elif config.anim <= 6 * config.times:
        primitif.basic.persegi_panjang(6 * half_width // 4, half_height // 2, 200, 100, c=[255, 0, 0, 255]) # 1
        primitif.basic.persegi_panjang_2(half_width // 2, half_height // 2, 200, 100, c=[255, 0, 0, 255]) # 2
        primitif.basic.persegi_panjang_3(half_width // 2, 3 * half_height // 2, 200, 100, c=[255, 0, 0, 255]) # 3
        primitif.basic.persegi_panjang_4(6 * half_width // 4, 3 * half_height // 2, 200, 100, c=[255, 0, 0, 255]) # 4

    elif config.anim <= 9 * config.times:
        primitif.basic.segitiga_siku(6 * half_width // 4, half_height // 2, 50, 100) # Kuadran 1
        primitif.basic.segitiga_siku_2(half_width // 2, half_height // 2, 50, 100) # Kuadran 2
        primitif.basic.segitiga_siku_3(half_width // 2, 3 * half_height // 2, 50, 100) # Kuadran 3
        primitif.basic.segitiga_siku_4(6 * half_width // 4, 3 * half_height // 2, 50, 100) # Kuadran 4

    elif config.anim <= 12 * config.times:
        primitif.basic.trapesium_siku(6 * half_width // 4, half_height // 2, 100, 200, 100) # 1
        primitif.basic.trapesium_siku_2(half_width // 2, half_height // 2, 100, 200, 100) # 2
        primitif.basic.trapesium_siku_3(half_width // 2, 3 * half_height // 2, 100, 200, 100) # 3
        primitif.basic.trapesium_siku_4(6 * half_width // 4, 3 * half_height // 2, 100, 200, 100) # 4


    if config.anim > 13 * config.times:
        config.anim = 0

    config.anim += 1

py5.run_sketch()
```

```
import numpy as np

def round(x):
    return int(x + 0.5)

def line_dda(xa, ya, xb, yb):
    dx = abs(xb - xa)
    dy = abs(yb - ya)
    length = max(dx, dy)

    dx = (xb - xa) / length
    dy = (yb - ya) / length

    x = xa
    y = ya

    res = [[xa, ya]]
```

```
import primitif.line
import py5
import numpy as np

def draw_margin(width, height, margin, c=[0,0,0,255], line_type=0):
    py5.stroke(c[0], c[1], c[2], c[3])
    draw_bentuk(primitif.line.line_dda(margin, margin, width - margin, margin),
line_type)
    draw_bentuk(primitif.line.line_dda(margin, height - margin, width - margin,
height - margin), line_type)
    draw_bentuk(primitif.line.line_dda(margin, margin, margin, height - margin),
line_type)
    draw_bentuk(primitif.line.line_dda(width - margin, margin, width - margin,
height - margin), line_type)

def draw_grid(width, height, margin, c=[0,0,0,255], line_type=0):
    xa = margin
```

```
    for i in range(length + 1):
        res.append([round(x), round(y)])
        x = x + dx
        y = y + dy

    return np.array(res)

def line_bresenham(xa, ya, xb, yb):
    if abs(yb - ya) < abs(xb - xa):
        if xa > xb:
            return np.array(line_low(xb, yb, xa, ya))
        else:
            return np.array(line_low(xa, ya, xb, yb))
    else:
        if ya > yb:
            return np.array(line_high(xb, yb, xa, ya))
        else:
            return np.array(line_high(xa, ya, xb, yb))

def line_low(xa, ya, xb, yb, dash_length=0):
    res = [[xa, ya]]
    dx = xb - xa
    dy = yb - ya
    yi = 1
    if dy < 0:
        yi = -1
        dy = -dy
    p = 2 * dy - dx
    twody = 2 * dy
    twodydx = 2 * (dy - dx)
    y = ya

    draw = True
    count = 0

    for x in range(xa, xb):
        if draw:
            res.append([x, y])
        count += 1
        if count == dash_length:
            draw = not draw
            count = 0
        if p > 0:
            y += yi
            p += twodydx
        else:
            p += twody

    return res

def line_high(xa, ya, xb, yb, dash_length=0):
    res = [[xa, ya]]

    dx = xb - xa
    dy = yb - ya
    xi = 1
    if dx < 0:
        xi = -1
        dx = -dx
    p = 2 * dx - dy
    twodx = 2 * dx
    twodxdy = 2 * (dx - dy)
    x = xa

    draw = True
    count = 0

    for y in range(ya, yb):
        if draw:
            res.append([x, y])
        count += 1
        if count == dash_length:
            draw = not draw
            count = 0
        if p > 0:
```

```
    ya = 2 * margin
    xb = width - xa
    yb = height - ya
    y_range = height / margin

    py5.stroke(c[0], c[1], c[2], c[3])
    for count in range(1, int(y_range)):
        draw_bentuk(primitif.line.line_dda(xa, ya, xb, ya), line_type)
        ya = ya + margin

    xa = 2 * margin
    ya = margin
    xb = width - xa
    yb = height - ya
    x_range = width / margin
    for count in range(1, int(x_range)):
        draw_bentuk(primitif.line.line_dda(xa, ya, xa, yb), line_type)
        xa = xa + margin

def draw_kartesian(width, height, margin, c=[0,0,0,255], line_type=0):
    py5.stroke(c[0], c[1], c[2], c[3])
    draw_bentuk(primitif.line.line_dda(width / 2, margin, width / 2, height -
margin), line_type)
    draw_bentuk(primitif.line.line_dda(margin, height / 2, width - margin,
height / 2), line_type)

def persegi(xa, ya, panjang, c=[0,0,0,255], line_type=0):
    x = xa - (panjang // 2)
    y = ya - (panjang // 2)

    py5.stroke(c[0], c[1], c[2], c[3])
    draw_bentuk(primitif.line.line_bresenham(x, y, x + panjang, y), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y, x + panjang, y +
panjang), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y + panjang, x, y +
panjang), line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + panjang, x, y), line_type)

def persegi_2(xa, ya, panjang, c=[0,0,0,255], line_type=1):
    x = xa - (panjang // 2)
    y = ya - (panjang // 2)

    py5.stroke(c[0], c[1], c[2], c[3])
    draw_bentuk(primitif.line.line_bresenham(x, y, x + panjang, y), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y, x + panjang, y +
panjang), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y + panjang, x, y +
panjang), line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + panjang, x, y), line_type)

def persegi_3(xa, ya, panjang, c=[0,0,0,255], line_type=2):
    x = xa - (panjang // 2)
    y = ya - (panjang // 2)

    py5.stroke(c[0], c[1], c[2], c[3])
    draw_bentuk(primitif.line.line_bresenham(x, y, x + panjang, y), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y, x + panjang, y +
panjang), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y + panjang, x, y +
panjang), line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + panjang, x, y), line_type)

def persegi_4(xa, ya, panjang, c=[0,0,0,255], line_type=3):
    x = xa - (panjang // 2)
    y = ya - (panjang // 2)

    py5.stroke(c[0], c[1], c[2], c[3])
    draw_bentuk(primitif.line.line_bresenham(x, y, x + panjang, y), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y, x + panjang, y +
panjang), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y + panjang, x, y +
panjang), line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + panjang, x, y), line_type)

def persegi_panjang(xa, ya, panjang, lebar, c=[0,0,0,255], line_type=0):
    x = xa - (panjang // 2)
```

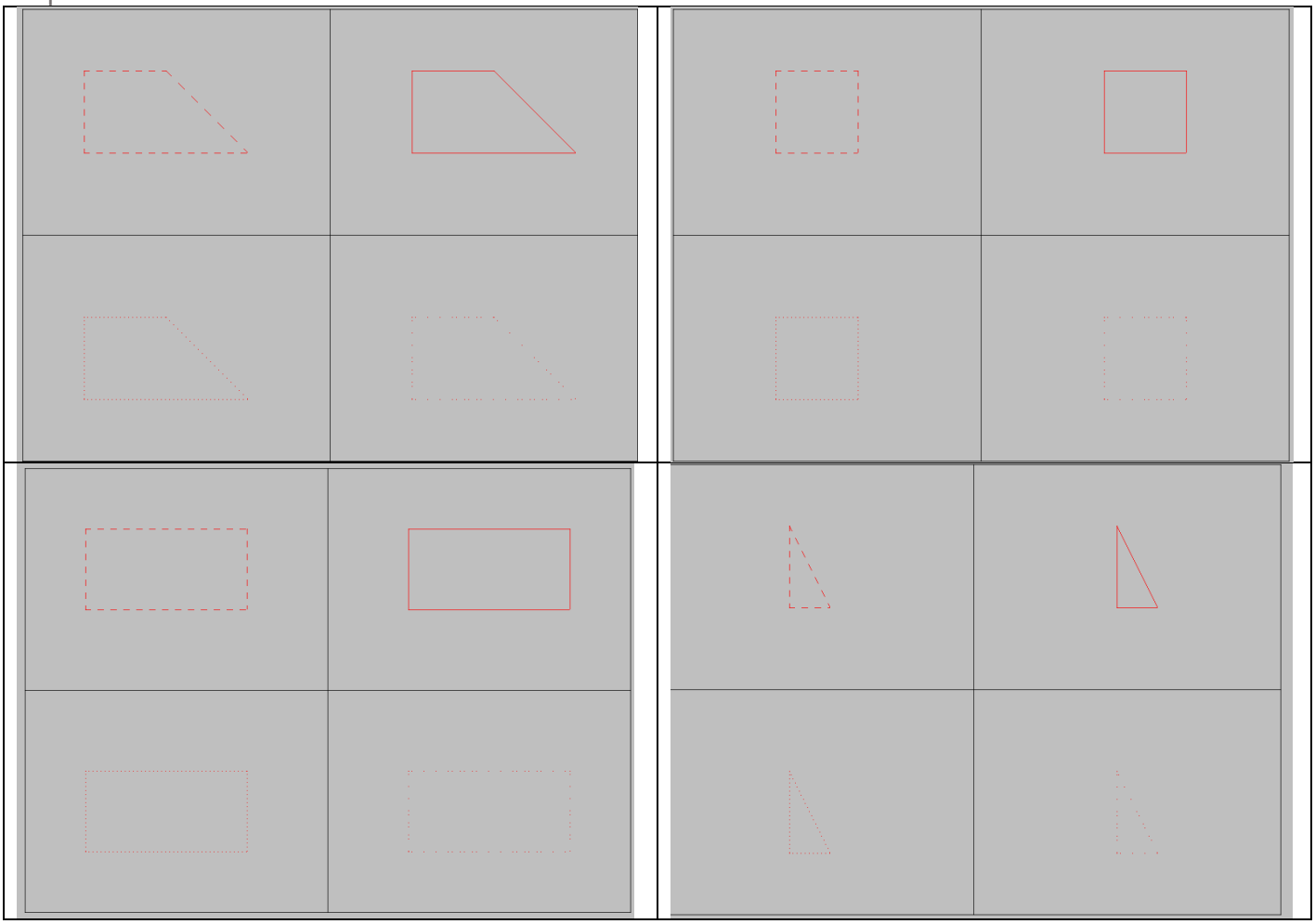
```
            x += xi
            p += twodxdy
        else:
            p += twodx

    return res
```

```
    y = ya - (lebar // 2)

    py5.stroke(c[0], c[1], c[2], c[3])
    draw_bentuk(primitif.line.line_bresenham(x, y, x + panjang, y), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y, x + panjang, y +
lebar), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y + lebar, x, y +
lebar), line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + lebar, x, y), line_type)

def persegi_panjang_2(xa, ya, panjang, lebar, c=[0,0,0,255], line_type=1):
    x = xa - (panjang // 2)
    y = ya - (lebar // 2)

    py5.stroke(c[0], c[1], c[2], c[3])
    draw_bentuk(primitif.line.line_bresenham(x, y, x + panjang, y), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y, x + panjang, y +
lebar), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y + lebar, x, y +
lebar), line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + lebar, x, y), line_type)

def persegi_panjang_3(xa, ya, panjang, lebar, c=[0,0,0,255], line_type=2):
    x = xa - (panjang // 2)
    y = ya - (lebar // 2)

    py5.stroke(c[0], c[1], c[2], c[3])
    draw_bentuk(primitif.line.line_bresenham(x, y, x + panjang, y), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y, x + panjang, y +
lebar), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y + lebar, x, y +
lebar), line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + lebar, x, y), line_type)

def persegi_panjang_4(xa, ya, panjang, lebar, c=[0,0,0,255], line_type=3):
    x = xa - (panjang // 2)
    y = ya - (lebar // 2)

    py5.stroke(c[0], c[1], c[2], c[3])
    draw_bentuk(primitif.line.line_bresenham(x, y, x + panjang, y), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y, x + panjang, y +
lebar), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + panjang, y + lebar, x, y +
lebar), line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + lebar, x, y), line_type)

def segitiga_siku(xa, ya, alas, tinggi, c=[255,0,0,255], line_type=0):
    py5.stroke(c[0], c[1], c[2], c[3])
    x = xa - alas // 2
    y = ya - tinggi // 2

    draw_bentuk(primitif.line.line_bresenham(x, y, x + alas, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x + alas, y + tinggi, x, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + tinggi, x, y), line_type)

def segitiga_siku_2(xa, ya, alas, tinggi, c=[255,0,0,255], line_type=1):
    py5.stroke(c[0], c[1], c[2], c[3])
    x = xa - alas // 2
    y = ya - tinggi // 2

    draw_bentuk(primitif.line.line_bresenham(x, y, x + alas, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x + alas, y + tinggi, x, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + tinggi, x, y), line_type)

def segitiga_siku_3(xa, ya, alas, tinggi, c=[255,0,0,255], line_type=2):
    py5.stroke(c[0], c[1], c[2], c[3])
    x = xa - alas // 2
    y = ya - tinggi // 2

    draw_bentuk(primitif.line.line_bresenham(x, y, x + alas, y + tinggi),
line_type)
```

```
    draw_bentuk(primitif.line.line_bresenham(x + alas, y + tinggi, x, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + tinggi, x, y), line_type)

def segitiga_siku_4(xa, ya, alas, tinggi, c=[255,0,0,255], line_type=3):
    py5.stroke(c[0], c[1], c[2], c[3])
    x = xa - alas // 2
    y = ya - tinggi // 2

    draw_bentuk(primitif.line.line_bresenham(x, y, x + alas, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x + alas, y + tinggi, x, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + tinggi, x, y), line_type)

def trapesium_siku(xa, ya, aa, ab, tinggi, c=[255,0,0,255], line_type=0):
    py5.stroke(c[0], c[1], c[2], c[3])
    x = xa - ab // 2
    y = ya - tinggi // 2

    draw_bentuk(primitif.line.line_bresenham(x, y, x + aa, y), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + aa, y, x + ab, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x + ab, y + tinggi, x, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + tinggi, x, y), line_type)

def trapesium_siku_2(xa, ya, aa, ab, tinggi, c=[255,0,0,255], line_type=1):
    py5.stroke(c[0], c[1], c[2], c[3])
    x = xa - ab // 2
    y = ya - tinggi // 2

    draw_bentuk(primitif.line.line_bresenham(x, y, x + aa, y), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + aa, y, x + ab, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x + ab, y + tinggi, x, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + tinggi, x, y), line_type)

def trapesium_siku_3(xa, ya, aa, ab, tinggi, c=[255,0,0,255], line_type=2):
    py5.stroke(c[0], c[1], c[2], c[3])
    x = xa - ab // 2
    y = ya - tinggi // 2

    draw_bentuk(primitif.line.line_bresenham(x, y, x + aa, y), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + aa, y, x + ab, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x + ab, y + tinggi, x, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + tinggi, x, y), line_type)

def trapesium_siku_4(xa, ya, aa, ab, tinggi, c=[255,0,0,255], line_type=3):
    py5.stroke(c[0], c[1], c[2], c[3])
    x = xa - ab // 2
    y = ya - tinggi // 2

    draw_bentuk(primitif.line.line_bresenham(x, y, x + aa, y), line_type)
    draw_bentuk(primitif.line.line_bresenham(x + aa, y, x + ab, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x + ab, y + tinggi, x, y + tinggi),
line_type)
    draw_bentuk(primitif.line.line_bresenham(x, y + tinggi, x, y), line_type)

def draw_bentuk(pts, line_type):
    if line_type == 0:
        # Full line
        for x, y in pts:
            py5.point(x, y)

    elif line_type == 1:
        # Dashed line
        dash_length = 8
        for i, (x, y) in enumerate(pts):
            if (i // dash_length) % 2 == 0:  # Draw every dash_length segment
                py5.point(x, y)
```

```
    elif line_type == 2:
        # Dotted line
        dot_interval = 5
        for i, (x, y) in enumerate(pts):
            if i % dot_interval == 0:  # Draw every dot_interval point
                py5.point(x, y)

    elif line_type == 3:
        # Dashed and dotted line
        dash_length = 8
        dot_interval = 5
        for i, (x, y) in enumerate(pts):
            if (i // dash_length) % 2 == 0:  # Draw every dash_length segment
                if i % dot_interval == 0:  # Draw every dot_interval point within
the segment
                    py5.point(x, y)
```

Output



Pada task ini saya memalukan modifikasi pada line.py dimana pada fiile ini saya menambangkan pengkondisian pada line_low dan Line_high modifikasi ini dikarenakan pada 2 fungsi ini merupakan parameter agar line_bersemham ini dapat diekseskusi dengna memanipulasi code ini maka ditambah logika untuk dapat melakukan gari kosong,kosongnya

hal ini dapat dilakuakan dengan melakukan pengkondisian pada parameter dash yang ingin dimasukkan, disini untuk mematikan linennya menggunakan Dash_line dimana ini bekerja dengan seperti ini

```
for y in range(ya, yb):
        if draw:
            res.append([x, y])
        count += 1
        if count == dash_length:
            draw = not draw
            count = 0
```

Pada say nilai loop ini memenuhi dash_length maka ia tidak akan menghasilkan garis untuk dibentu atau incremenet tidak ditulis logic ini diimplemetasikan pada low_line dan High_line.

Pada Basic saya menambahkan fungsi lagi dimaan fungsi ini berkerja untuk dapat menentukan sebuah panjang dari length yang akan dilaksanakan pada line dimana pada ini saya membuat 4 pengkondisian dimana 0 ini tidak ada flow yang dimanssukan bila 1 maka dash_line akan diiisi 5 dan akan membentuk garis putus-putus, bila 2 maka akan membuat garis titik-titik dan bila 3 maka ia akan akan membaut garis putus-putus dan titik-titik seperti dibawah ini

```
def draw_bentuk(pts, line_type):
    if line_type == 0:
        for x, y in pts:
            py5.point(x, y)

    elif line_type == 1:
        dash_length = 8
        for i, (x, y) in enumerate(pts):
            if (i // dash_length) % 2 == 0:
                py5.point(x, y)

    elif line_type == 2:
        # Dotted line
        dot_interval = 5
        for i, (x, y) in enumerate(pts):
            if i % dot_interval == 0:
                py5.point(x, y)

    elif line_type == 3:
        # Dashed and dotted line
        dash_length = 8
        dot_interval = 5
        for i, (x, y) in enumerate(pts):
            if (i // dash_length) % 2 == 0:
                if i % dot_interval == 0:
                    py5.point(x, y)
```

Untuk file main ini saya membaut OOP didalamnya kelas yang dibentuk ada 1 yaitu kelas drawer dimana kelas ini berfungsi untuk dapat menghasilkan garis kartesius dalam class ini saya menghadapi kendala dimana saat saya membuat class untuk setiap shape untuk dianimasikan saya berfikir tidak perlu untuk memasukkan class karena pada bagian kelas karena pada draw animasi hanya memanggil fungsi pada besic bila dibuat kelas maka ia akan kurang efektif

PENGUMPULAN

Ikuti Format yang diberikan di Google Classroom.