

|  |
| --- |
| MODUL 1II |
| 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](#_Toc176154149)

[TIC TAC TOE GAME 1](#_Toc176154150)

[TASK PRAKTIKUM 2](#_Toc176154151)

[PENGUMPULAN 5](#_Toc176154152)

# ATRIBUTE GARIS

Ragam attribute garis

A line graph with dots

Description automatically generated

# TIC TAC TOE GAME

<https://playtictactoe.org/>

A group of kids playing a game

Description automatically generated

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) |
| A grid of circles and squares  Description automatically generated  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**   |  |  | | --- | --- | | A grid of circles and squares  Description automatically generated |  | |  |  |   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    A grid of circles and squares  Description automatically generatedTASK 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   |  |  | | --- | --- | |  | | |  |  |   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  A black background with red circles  Description automatically generatedMain 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  A white stick figure with a round object on it  Description automatically generated with medium confidence  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] |

Graphical user interface

Description automatically generated with low confidence

A picture containing graphical user interface

Description automatically generated

A picture containing text

Description automatically generated

Shape, polygon

Description automatically generated

|  |
| --- |
| 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]]        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:              x += xi              p += twodxdy          else:              p += twodx        return res | 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      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)      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   |  |  | | --- | --- | | A screenshot of a computer screen  Description automatically generated |  | |  |  |   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.