

## # 1. VARIABEL

# Pahami bahwa variabel menyimpan nilai yang bisa berubah, seperti kecepatan, posisi, warna.

```
x = 10 # posisi horizontal
speed = 5 # kecepatan ke kanan
```

```
print(x + speed) # hasilnya 15
```

# Assignment (Pengisian Nilai)

```
x += 2 # x = 17
```

```
x *= 3 # x = 36
```

```
print(x)
```

# contoh 1 keranjang a bulpen berisi sejumlah 6 bulpen

# contoh 1 keranjang b bulpen berisi sejumlah 4 buku

## #2. DICT, LIST, TUPLE

#LIST

#Contoh list (mutable)

#array (bisa diubah)

# keranjang buah berisi : berbagai jenis buah

```
buah = ["apel", "jeruk", "pisang"]
```

```
buah[0] = "mangga" # Bisa diubah
```

```
buah.append("anggur") # Bisa ditambah
```

```
print(buah) #['mangga', 'jeruk', 'pisang', 'anggur']
```

#keranjang yang boleh berisi macam2 barang: uang,buah,bolpen,dll

#TUPLE

#tuple di Python ≈ array dengan const di JavaScript

#array (tidak diubah)

# sebuah alamat dan kode plat kendaraan

```
lokasi = ("Jakarta", "B")
```

```
print(lokasi[0]) # Output: Jakarta
```

#keranjang yang boleh berisi macam2 barang: uang,buah,bolpen,dll tapi isinya tidak boleh diubah2

# DICT di Python = object di JavaScript

# kumpulan data user : nama, usia

```
user = {
    "nama": "Andi",
    "usia": 25
}
```

```

print(user["nama"]) # Output: Andi
user["alamat"] = "Semarang"
print(user) #{'nama': 'Andi', 'usia': 25, 'alamat': 'Semarang'}
user["nama"] = "Budi" # Ubah nama
print(user) #{'nama': 'Budi', 'usia': 25, 'alamat': 'Semarang'}
#keranjang yang berisi barang berpola (key : value) : buah :10, pulpen
: 5 dll

```

Tujuan	Python	JavaScript setara
Tambah item	<code>list.append(dict)</code>	<code>array.push(object)</code>
Ubah isi dict	<code>list[i]["key"] = new_value</code>	<code>array[i].key = new_value</code>
Looping	<code>for item in list:</code>	<code>for (const item of array)</code>
Akses item	<code>list[i]["key"]</code>	<code>array[i].key</code>
Hapus item	<code>del list[i]</code>	<code>array.splice(i, 1)</code>
Panjang list	<code>len(list)</code>	<code>array.length</code>
Cek kunci	<code>"key" in list[i]</code>	<code>"key" in array[i]</code>
Cari item	<code>next(d for d in list if ...)</code>	<code>array.find(obj =&gt; ...)</code>
Filter list	<code>[d for d in list if ...]</code>	<code>`array.filter(obj =&gt;</code>

### # 3. FUNCTION (FUNGSI)

# Pahami bahwa fungsi adalah blok perintah yang bisa dipanggil berulang kali, untuk mengelompokkan logika tertentu.

```

def fungsi1():
    print("Halo, ini fungsi!")

```

```

fungsi1() # Halo, ini fungsi!

```

```

def fungsi2(nama):
    print("Halo, ini fungsi!")

```

```

fungsi2("silmi") # Halo, ini fungsi!

```

#Urutan tetap penting: parameter dengan default harus di belakang.

```

def fungsi3(nama="default nama"):
    print("Halo", nama)

```

```

fungsi3() # Output: Halo default nama
fungsi3("Silmi") # Output: Halo Silmi

```

#\*args → untuk jumlah argumen tak terbatas (seperti array)  
 #\*angka akan berisi tuple (3, 4, 5)

```

def total(*angka):
    print("Semua angka:", angka)
    print("Jumlah:", sum(angka))

total(3, 4, 5) # Semua angka: (3, 4, 5), Jumlah: 12

***kwargs: Argumen Kata Kunci Tak Terbatas
def biodata(**data):
    for k, v in data.items():
        print(f"{k}: {v}")

biodata(nama="Silmi", alamat="Semarang", usia=13)

#Fungsi yang Mengembalikan Nilai (return)
def tambah(a, b):
    return a + b

hasil = tambah(3, 4)
print("Hasilnya:", hasil) # 7

```

Fitur	Penjelasan
*args	Menangkap banyak argumen biasa → dikemas sebagai tuple
**kwargs	Menangkap banyak argumen kunci-nilai → jadi dict
return	Mengembalikan nilai dari fungsi
Method class	Fungsi dalam class, selalu menerima self sebagai argumen pertama

#### # 4. GLOBAL VARIABLE

# Pahami bahwa variabel bisa diakses di mana-mana kalau dideklarasikan sebagai global.  
 # Biasanya dipakai saat ingin mengubah/mengambil nilai dari luar fungsi.

```

nama = "silmi" # variabel global

def sapa():
    global alamat
    alamat = "semarang"
    print("Halo: ", nama, "Alamat : ", alamat)

sapa() # Halo, ini fungsi!

```

```
print("Alamat : ", alamat)
```

## 5. Apa Itu Ternary Operator di Python?

Ternary operator adalah **cara singkat** untuk menulis pernyataan if-else dalam **satu baris**.

### **Format Umum:**

```
nilai_jika_true if kondisi else nilai_jika_false
```


### **Contoh 1: Menentukan Nilai Terbesar**

```
a = 5
```

```
b = 10
```

```
maks = a if a > b else b
```

```
print("Nilai terbesar adalah:", maks)
```

 **Output:**

```
Nilai terbesar adalah: 10
```

### **Versi biasa:**

```
if a > b:
```

```
    maks = a
```

```
else:
```


```
    maks = b
```

### **Contoh 2: Cek Bilangan Genap atau Ganjil**

```
angka = 7
```

```
jenis = "Genap" if angka % 2 == 0 else "Ganjil"
```

```
print(f"{angka} adalah bilangan {jenis}")
```

 **Output:**

```
7 adalah bilangan Ganjil
```

### **Versi biasa:**

```
if angka % 2 == 0:
```

```
    jenis = "Genap"
```

```
else:
```


```
    jenis = "Ganjil"
```

### **Contoh 3: Cek Umur**

```
umur = 15
```

```
status = "Dewasa" if umur >= 18 else "Anak-anak"
```

```
print("Status:", status)
```

 Output:  
Status: Anak-anak

### Kapan Dipakai?

Gunakan ternary operator jika:

- Hanya ada **dua kemungkinan (if dan else)**.
- Ingin menulis kode **lebih ringkas** dan mudah dibaca.

Jika logikanya lebih kompleks (pakai elif, atau lebih dari satu aksi), sebaiknya tetap pakai if-else biasa.

### # 6. FUNGSI Khusus : max() min() sum() len() sorted()

# max() Ambil data dengan nilai terbesar

```
angka = [5, 2, 9, 1, 2, 3]
```

```
terbesar = max(angka)
```

```
print(terbesar) # 9
```

# sum() Jumlahkan semua nilai

```
print(sum(angka)) #
```

# sorted() Urutkan data

```
print(sorted(angka)) #
```

# filter() → Menyaring Data Sesuai Kondisi

```
genap = list(filter(lambda x: x % 2 == 0, angka))
```

```
print(genap) #
```

# map() → Mengubah Setiap Elemen

```
dikali2 = list(map(lambda x: x * 2, angka))
```

```
print(dikali2) #
```

Fungsi	Kegunaan
max()	Ambil data dengan nilai <b>terbesar</b>
min()	Ambil data dengan nilai <b>terkecil</b>
sum()	<b>Jumlahkan</b> semua nilai
len()	Hitung <b>jumlah item</b> dalam list
sorted()	<b>Urutkan</b> data dari kecil ke besar (default)

### Operasi Aritmatika (Matematika)

Operator	Arti	Contoh	Hasil
+	Penjumlahan	5 + 2	7

Operator	Arti	Contoh	Hasil
-	Pengurangan	5 - 2	3
*	Perkalian	5 * 2	10
/	Pembagian	5 / 2	2.5
//	Pembagian bulat	5 // 2	2
%	Sisa bagi (mod)	5 % 2	1
**	Pangkat	2 ** 3	8

### Assignment (Pengisian Nilai)

Bentuk	Sama dengan
x += 1	x = x + 1
x -= 1	x = x - 1
x *= 2	x = x * 2
x /= 2	x = x / 2
x //= 2	x = x // 2

### Operasi Logika (Boolean)

Operator	Arti	Contoh	Hasil
and	True jika keduanya True	True and False	False
or	True jika salah satu True	True or False	True
not	Membalik nilai Boolean	not True	False

### Operator Perbandingan

Operator	Arti	Contoh	Hasil
==	Sama dengan	3 == 3	True
!=	Tidak sama	3 != 4	True
>	Lebih dari	5 > 2	True
<	Kurang dari	5 < 2	False
>=	Lebih atau sama	5 >= 5	True
<=	Kurang atau sama	4 <= 5	True


## # 7. List of Dict di Python : Array Object

```
siswa = [
    {"nama": "Silmi", "alamat": "Semarang"},
    {"nama": "Edy", "alamat": "Jakarta"}
]
```

#  1. Akses Data

```
print(siswa[0]["nama"])    # Silmi
print(siswa[1]["alamat"])  # Jakarta
```

```
siswa.append({"nama": "gita", "alamat": "Bandung"})
```

#  3. Ubah Data dalam Dict

```
siswa[1]["alamat"] = "Surabaya"
```

# Loop Semua Data

```
for s in siswa:
    print(f"{s['nama']} tinggal di {s['alamat']}")
```

#Silmi tinggal di Semarang

#Edy tinggal di Jakarta

#gita tinggal di Bandung

```
print(siswa) #[{'nama': 'Silmi', 'alamat': 'Semarang'}, {'nama': 'Edy', 'alamat': 'Jakarta'}, {'nama': 'gita', 'alamat': 'Bandung'}]
```

```
siswa.append({"bebas": "tidak beraturan"})
```

```
print(siswa) #[{'nama': 'Silmi', 'alamat': 'Semarang'}, {'nama': 'Edy', 'alamat': 'Surabaya'}, {'nama': 'gita', 'alamat': 'Bandung'}, {'bebas': 'tidak beraturan'}]
```

for s in siswa:



```
    print(f"{s['nama']} tinggal di {s['alamat']}") #error karena array
object isinya tidak beraturan >> dibutuhkan class supaya bentuk array
object siswa selalu beraturan
```

#  List of dict fleksibel tapi  tidak menjamin struktur data yang tetap.

#Contoh: kita bisa tambah {"bebas": "tidak beraturan"} yang menyebabkan error saat looping.

# analogi sebuah ruangan[] yang berisi banyak keranjang{}, dengan keranjang isinya berpola (key: value)

## # 8. OOP (Object-Oriented Programming) di Python

#  List of dict fleksibel tapi  tidak menjamin struktur data yang tetap.

#Contoh: kita bisa tambah {"bebas": "tidak beraturan"} yang menyebabkan error saat looping.

# Versi OOP (Class)

```
# Class Siswa sebagai template
class Siswa:
    def __init__(self, nama, alamat):
        self.nama = nama
        self.alamat = alamat

    def tampilkan(self):
        print(f"{self.nama} tinggal di {self.alamat}")

# List of Object (bukan dict lagi)
daftar_siswa = [
    Siswa("Silmi", "Semarang"),
    Siswa("Edy", "Jakarta"),
    Siswa("Gita", "Bandung")
]

# Tambah siswa baru
daftar_siswa.append(Siswa("Lina", "Medan"))

# Ubah data Edy (indeks 1)
daftar_siswa[1].alamat = "Surabaya"

# Tampilkan semua siswa
for s in daftar_siswa:
    s.tampilkan()
```

### Kenapa Class Lebih Baik?

Fitur	List of Dict	List of Object (Class)
Struktur konsisten	✗ Bisa tidak beraturan	✓ Terjamin oleh <code>__init__()</code>
Bisa punya method	✗ Tidak	✓ Bisa ( <code>tampilkan()</code> , dll)
Validasi / aturan atribut	✗ Tidak bisa	✓ Bisa diatur di dalam class
Reusability (OOP)	✗ Tidak fleksibel	✓ Bisa inheritance (pewarisan)
Autocomplete di editor	✗ Tidak ada	✓ Umumnya tersedia di IDE

### Perbedaan self dan super()

Fungsi	Artinya
self	Menunjuk ke <b>objek itu sendiri</b> (instans dari class)



<b>Fungsi</b>	<b>Artinya</b>
<code>super()</code>	Menunjuk ke <b>kelas induk (parent)</b> , berguna saat kita extend kelas lain
Gunanya	Akses atribut/method milik object
Kapan pakai	Di semua method instance

---

## #9. Fungsi Khusus untuk List of Object

```
class Siswa:
    def __init__(self, nama, nilai):
        self.nama = nama
        self.nilai = nilai

siswa = [
    Siswa("Silmi", 88),
    Siswa("Edy", 95),
    Siswa("Gita", 80),
]

# max() Ambil data dengan nilai terbesar
terbaik = max(siswa, key=lambda s: s.nilai)
print(terbaik.nama) # Edy

# min() Ambil data dengan nilai terkecil
terendah = min(siswa, key=lambda s: s.nilai)
print(terendah.nama) # Gita

# sum() Jumlahkan semua nilai
total_nilai = sum(s.nilai for s in siswa)
print(total_nilai) # 263

# len() Hitung jumlah item dalam list
print(len(siswa)) # 3

# sorted() Urutkan data
urut = sorted(siswa, key=lambda s: s.nilai, reverse=True)
for s in urut:
    print(s.nama, s.nilai)

# filter() → Menyaring Data Sesuai Kondisi
# Ambil yang nilainya lulus (>=75)
lulus = list(filter(lambda s: s.nilai >= 75, siswa))
```

```

for s in lulus:
    print(f"{s.nama} lulus dengan nilai {s.nilai}")

# map() → Mengubah Setiap Elemen
# Ubah jadi list of string
hasil = list(map(lambda s: f"{s.nama}: {s.nilai}", siswa))

print(hasil)
# ['Silmi: 88', 'Edy: 95', 'Gita: 70', 'Rani: 60']

# Kombinasi filter() + map()
# Cetak nama siswa yang lulus
hasil = list(map(lambda s: s.nama, filter(lambda s: s.nilai >= 75,
siswa)))
print(hasil) # ['Silmi', 'Edy']

```

#### # 9. Fungsi dalam Class = Method

```

# lulus() dan tampilkan() disebut method
# Semua method harus punya self sebagai parameter pertama
class Siswa:
    def __init__(self, nama, nilai):
        self.nama = nama
        self.nilai = nilai

    def lulus(self):
        return self.nilai >= 75

    def tampilkan(self):
        print(f"{self.nama} - Nilai: {self.nilai}")

# Pakai class
s1 = Siswa("Silmi", 80)
s1.tampilkan()           # Silmi - Nilai: 80
print(s1.lulus())        # True

```

#### # 10. \_\_init\_\_() dengan Nilai Default

```

#Urutan tetap penting: parameter dengan default harus di belakang.
class Siswa:
    def __init__(self, nama="Anonim", alamat="Tidak diketahui"):
        self.nama = nama
        self.alamat = alamat

    def tampilkan(self):

```

```

        print(f"{self.nama} tinggal di {self.alamat}")

s1 = Siswa("Silmi", "Semarang")
s2 = Siswa("Edy")                # alamat default
s3 = Siswa()                    # nama dan alamat default

s1.tampilkan() # Silmi tinggal di Semarang
s2.tampilkan() # Edy tinggal di Tidak diketahui
s3.tampilkan() # Anonim tinggal di Tidak diketahui

```

#### # 11. Gabungan: Method dengan \*args, return, dll

```

class Kalkulator:
    def jumlahkan(self, *angka):
        return sum(angka)

k = Kalkulator()
print(k.jumlahkan(1, 2, 3, 4)) # Output: 10

```

#### # 12. \_\_str\_\_() untuk cetak objek dengan lebih rapi

```

# Kelas Bola
class Siswa:
    #self menunjuk ke objek itu sendiri
    def __init__(self, nama, alamat):
        self.nama = nama
        self.alamat = alamat
        self.kelas = 8

    def identitas(self):
        print(self.nama, self.alamat, self.kelas)

    #__str__() untuk cetak objek dengan lebih rapi
    def __str__(self):
        return f>Nama: {self.nama}, Alamat: {self.alamat}, Kelas:
{self.kelas}"

```

```

siswa1 = Siswa("silmi", "semarang")
siswa1.identitas() #silmi semarang 8

```

```

siswa2 = Siswa("edy", "pati")
siswa2.identitas() #edy pati 8

```

```

print(siswa1) # Nama: silmi, Alamat: semarang, Kelas: 8

```

#### #2. Mewarisi Kelas (Pewarisan / Inheritance)

```

class SiswaOlimpiade(Siswa): # mewarisi dari Siswa
    def __init__(self, nama, alamat, lomba):

```

```

        #super()    menunjuk ke kelas induk (parent), berguna saat
kita extend kelas lain
        super().__init__(nama, alamat) # panggil konstruktor dari
kelas Siswa
        self.lomba = lomba

    def identitas(self):
        # menambahkan info lomba ke identitas
        print(self.nama, self.alamat, self.kelas, "Lomba:",
self.lomba)

siswa3 = SiswaOlimpiade("dina", "solo", "matematika")
siswa3.identitas() #dina solo 8 Lomba: matematika

```

### # 13. Latihan Class , if else, max

```

class Siswa:
    def __init__(self, nama, alamat, nilai):
        self.nama = nama
        self.alamat = alamat
        self.nilai = nilai

    def predikat(self):
        if self.nilai >= 90:
            return "Sangat Baik"
        elif self.nilai >= 75:
            return "Baik"
        else:
            return "Perlu Bimbingan"

daftar_siswa = [
    Siswa("Rina", "Surabaya", 92),
    Siswa("Budi", "Semarang", 85),
    Siswa("Wati", "Solo", 70),
]

terbaik = max(daftar_siswa, key=lambda s: s.nilai)
print(terbaik.nama, terbaik.predikat())

```

## TKINTER

Dalam **Tkinter**, Canvas adalah widget yang digunakan untuk menggambar bentuk-bentuk grafis seperti garis, lingkaran, persegi panjang, teks, gambar, dan animasi. Canvas sering dipakai untuk simulasi visual dan permainan.

### 1. Pengertian Canvas

Canvas adalah bidang gambar kosong tempat kita bisa menggambar elemen-elemen grafis. Kita bisa menggambar:

- Garis (`create_line`)
- Persegi panjang (`create_rectangle`)
- Lingkaran/oval (`create_oval`)
- Teks (`create_text`)
- Gambar (`create_image`)
- Poligon (`create_polygon`)

Contoh dasar pembuatan canvas:

```
import tkinter as tk

root = tk.Tk()
root.title("Belajar Canvas Tkinter")

canvas = tk.Canvas(root, width=800, height=800, bg="white")
#Main Program
#.....

canvas.pack()
root.mainloop()
```

### ✓ Tabel Fungsi Dasar canvas Tkinter

Fungsi	Kegunaan	Contoh Sintaks
<code>create_oval</code>	Gambar lingkaran atau bola	<code>canvas.create_oval(x1, y1, x2, y2, fill="red")</code>
<code>create_rectangle</code>	Gambar persegi atau persegi panjang	<code>canvas.create_rectangle(x1, y1, x2, y2, fill="blue")</code>
<code>create_line</code>	Gambar garis lurus	<code>canvas.create_line(x1, y1, x2, y2, fill="black")</code>
<code>move</code>	Menggerakkan objek ke arah tertentu	<code>canvas.move(objek_id, dx, dy)</code>
<code>coords</code>	Mengubah posisi/ukuran objek	<code>canvas.coords(objek_id, x1, y1, x2, y2)</code>
<code>delete</code>	Menghapus objek dari canvas	<code>canvas.delete(objek_id)</code>

Fungsi	Kegunaan	Contoh Sintaks
itemconfig	Ubah warna, teks, dll dari objek	canvas.itemconfig(objek_id, fill="green")
create_text	Menampilkan teks di canvas	canvas.create_text(x, y, text="Halo!", font=("Arial", 12))

### 📌 Penjelasan Tambahan

Istilah	Penjelasan
x1, y1, x2, y2	Titik kiri-atas dan kanan-bawah dari area gambar (kotak pembungkus bentuk)
fill="warna"	Warna isi dari bentuk (misalnya "red", "blue", "green")
dx, dy	Perpindahan objek di sumbu x dan y (misal dx=10 artinya geser ke kanan 10 piksel)
objek_id	ID unik yang diberikan saat objek dibuat, digunakan untuk mengubah objek

## 2. Sistem Koordinat di Canvas Tkinter

Canvas di Tkinter menggunakan **sistem koordinat kartesian kiri-atas**, yang berarti:

- Titik (0, 0) berada di **pojok kiri atas** canvas.
- Arah sumbu-X → ke **kanan**
- Arah sumbu-Y → ke **bawah**

(0,0) -----> X (800)

|  
|  
|  
v

Y (800)

### Penjelasan Koordinat

- x = 0 → paling kiri, x = 800 → paling kanan
- y = 0 → paling atas, y = 800 → paling bawah

Contoh:

# Gambar titik di tengah canvas

```
canvas.create_oval(395, 395, 405, 405, fill="red") # Titik tengah (400,400)
```

## 3. Contoh Menggambar Berbagai Objek

# Garis dari kiri atas ke kanan bawah

```
canvas.create_line(0, 0, 800, 800, fill="blue", width=2)
```

# Persegi panjang di tengah

```
canvas.create_rectangle(300, 300, 500, 500, outline="black", fill="lightblue")
```

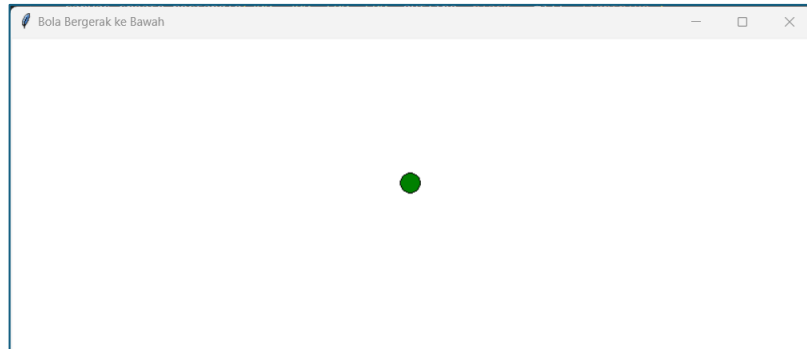
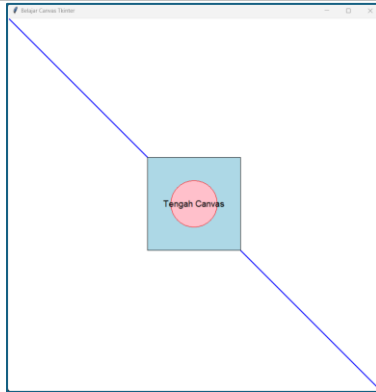
# Lingkaran (oval) di tengah

```

canvas.create_oval(350, 350, 450, 450, outline="red", fill="pink")

# Teks di tengah
canvas.create_text(400, 400, text="Tengah Canvas", font=("Arial", 14), fill="black")

```



#### 4. Tips Penggunaan

- Gunakan koordinat relatif dari ukuran canvas untuk objek simetris:
- `tengah_x = 800 // 2`
- `tengah_y = 800 // 2`
- `canvas.create_text(tengah_x, tengah_y, text="Tengah!")`
- Untuk animasi, kamu bisa memindahkan objek dengan `canvas.move()` berdasarkan sumbu X/Y.

#### #1. BELAJAR CANVAS TKINTER

```
import tkinter as tk
```

```
root = tk.Tk()
root.title("Belajar Canvas Tkinter")
```

```
# 1. Pengertian Canvas
```

```
canvas = tk.Canvas(root, width=800, height=800, bg="white")
```

```
#Main Program
```

```
#.....
```

```
# 2. Gambar titik di tengah canvas
```

```
canvas.create_oval(395, 395, 405, 405, fill="red") # Titik tengah (400,400)
```

```
# 3. Garis dari kiri atas ke kanan bawah
```

```
canvas.create_line(0, 0, 800, 800, fill="blue", width=2)
```

```
# Persegi panjang di tengah
```

```
canvas.create_rectangle(300, 300, 500, 500, outline="black",
fill="lightblue")
```

```
# Lingkaran (oval) di tengah
```

```
canvas.create_oval(350, 350, 450, 450, outline="red", fill="pink")
```

```
# Teks di tengah
```

```
canvas.create_text(400, 400, text="Tengah Canvas", font=("Arial", 14),  
fill="black")
```

```
canvas.pack()  
root.mainloop()
```

---

## 5. Simulasi Gaya/Animasi (contoh sederhana)

### #2. BOLA BERGERAK KE BAWAH

```
import tkinter as tk
```

```
root = tk.Tk()  
root.title("Bola Bergerak ke Bawah")  
canvas = tk.Canvas(root, width=800, height=800, bg="white")
```

```
#Main Program
```

```
#.....
```

```
posisi = canvas.create_oval(390, 0, 410, 20, fill="green")
```

```
def gerak():  
    canvas.move(posisi, 0, 5) # geser 5 piksel ke bawah  
    canvas.after(50, gerak)   # ulangi tiap 50 milidetik
```

```
gerak()
```

```
canvas.pack()  
root.mainloop()
```

---

## Kesimpulan

- Canvas adalah tempat menggambar objek grafis.
- Sistem koordinat: (0, 0) di kiri atas, dan (800, 800) di kanan bawah (jika ukuran canvas 800x800).
- X ke kanan, Y ke bawah.
- Objek digambar berdasarkan koordinat (x1, y1, x2, y2) tergantung jenis bentuknya.

Berikut ini penjelasan **sistem koordinat di Tkinter Canvas**



### 1. Titik Awal (0, 0) di Pojok Kiri Atas

Dalam Tkinter Canvas:

- Titik (0, 0) ada di pojok kiri atas.
- Arah **kanan** = tambah X
- Arah **bawah** = tambah Y



### Contoh 1: Gambar Titik di (0, 0)

```
canvas.create_oval(0, 0, 10, 10, fill="red")
```

Ini membuat lingkaran kecil (titik) di pojok kiri atas.





### 2. Bergerak ke Kanan = Tambah X

```
canvas.create_oval(100, 0, 110, 10, fill="blue")
```



Titik ini berada **100 piksel** ke kanan dari kiri. Artinya  $x = 100$ ,  $y = 0$ .

---



### 3. Bergerak ke Bawah = Tambah Y

```
canvas.create_oval(0, 100, 10, 110, fill="green")
```



Titik ini berada **100 piksel** ke bawah dari atas. Artinya  $x = 0$ ,  $y = 100$ .

---



### 4. Titik Tengah Canvas (400, 400)

Misalnya ukuran canvas 800x800, titik tengahnya adalah:

```
canvas.create_oval(395, 395, 405, 405, fill="orange")
```

```
canvas.create_text(400, 380, text="Tengah (400,400)", fill="black")
```

---



### 5. Buat Bentuk di Sudut-sudut Canvas

```
# Kiri Atas (0, 0)
```

```
canvas.create_text(10, 10, text="(0, 0)", anchor="nw")
```

```
# Kanan Atas (800, 0)
```

```
canvas.create_text(790, 10, text="(800, 0)", anchor="ne")
```

```
# Kiri Bawah (0, 800)
```

```
canvas.create_text(10, 790, text="(0, 800)", anchor="sw")
```

```
# Kanan Bawah (800, 800)
```

```
canvas.create_text(790, 790, text="(800, 800)", anchor="se")
```

---



### 6. Buat Kotak di Tengah Canvas

```
# Buat kotak dari (350, 350) ke (450, 450)
```

```
canvas.create_rectangle(350, 350, 450, 450, outline="black",  
fill="lightblue")
```

```
canvas.create_text(400, 340, text="Kotak di Tengah", fill="black")
```

---



### Penjelasan Mudah

Bayangkan kamu menggambar di kertas. Tapi bedanya:

- Ujung kiri atas adalah titik (0, 0).
  - Makin ke kanan, X makin besar.
  - Makin ke bawah, Y makin besar.
  - Titik tengah kertas = (400, 400) kalau ukuran 800x800.
-

Berikut ini adalah **simulasi sederhana Tkinter** untuk membantu memahami **sistem koordinat Canvas**. Saat kamu **klik di area canvas**, titik akan digambar, dan **koordinat X dan Y** ditampilkan di layar.



### Tujuan Simulasi

- Menunjukkan letak titik berdasarkan koordinat.
- Menjelaskan bahwa (0, 0) adalah pojok kiri atas.
- Menunjukkan arah sumbu X dan Y.



### Kode Lengkap: Klik & Tampilkan Koordinat >> copy paste

```
#3. SISTEM KOORDINAT DI TKINTER CANVAS 1
root = tk.Tk()
root.title("sistem koordinat di Tkinter Canvas")
canvas = tk.Canvas(root, width=800, height=800, bg="white")

#Main Program
#1. Titik Awal (0, 0) di Pojok Kiri Atas
canvas.create_oval(0, 0, 10, 10, fill="red")

#🕒 2. Bergerak ke Kanan = Tambah X
canvas.create_oval(100, 0, 110, 10, fill="blue")

#🕒 3. Bergerak ke Bawah = Tambah Y
canvas.create_oval(0, 100, 10, 110, fill="green")

#📊 4. Titik Tengah Canvas (400, 400)
canvas.create_oval(395, 395, 405, 405, fill="orange")
canvas.create_text(400, 380, text="Tengah (400,400)", fill="black")

#📐 5. Buat Bentuk di Sudut-sudut Canvas
# Kiri Atas (0, 0)
canvas.create_text(10, 10, text="(0, 0)", anchor="nw")

# Kanan Atas (800, 0)
canvas.create_text(790, 10, text="(800, 0)", anchor="ne")

# Kiri Bawah (0, 800)
canvas.create_text(10, 790, text="(0, 800)", anchor="sw")

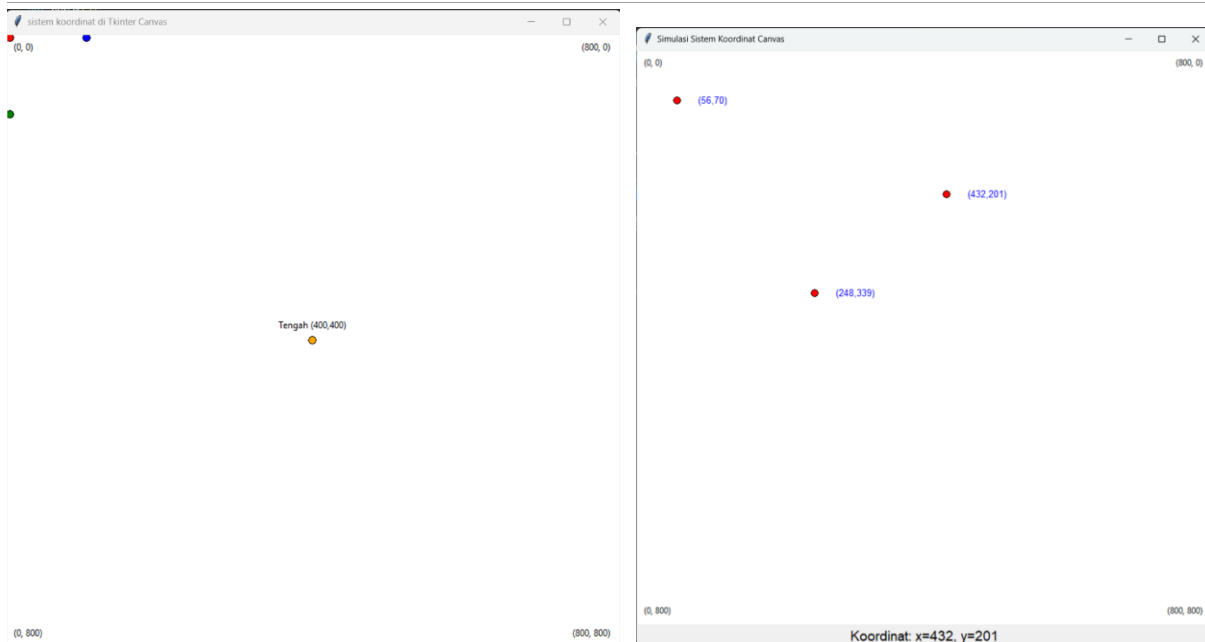
# Kanan Bawah (800, 800)
canvas.create_text(790, 790, text="(800, 800)", anchor="se")

#📦 6. Buat Kotak di Tengah Canvas
# Buat kotak dari (350, 350) ke (450, 450)
#canvas.create_rectangle(350, 350, 450, 450, outline="black",
fill="lightblue")
#canvas.create_text(400, 340, text="Kotak di Tengah", fill="black")
```

```
canvas.pack()  
root.mainloop()
```

### 📌 Penjelasan

- Klik di mana saja di canvas.
- Titik merah akan muncul di tempat kamu klik.
- Di bawah canvas akan muncul tulisan: Koordinat: x=..., y=...
- Di dekat titik juga muncul angka koordinatnya.
- Kamu bisa lihat bahwa semakin ke kanan, angka X bertambah.
- Semakin ke bawah, angka Y bertambah.



#### #4. SIMULASI KOORDINAT CANVAS 2 COPY-PASTE

```
import tkinter as tk
```

```
# Buat jendela utama
```

```
root = tk.Tk()
```

```
root.title("Simulasi Sistem Koordinat Canvas")
```

```
# Buat Canvas 800x800
```

```
canvas = tk.Canvas(root, width=800, height=800, bg="white")
```

```
canvas.pack()
```

```
# Label untuk menampilkan koordinat
```

```
label = tk.Label(root, text="Klik di canvas untuk melihat koordinat",  
font=("Arial", 14))
```

```
label.pack()
```

```
# Fungsi saat mouse diklik
```

```

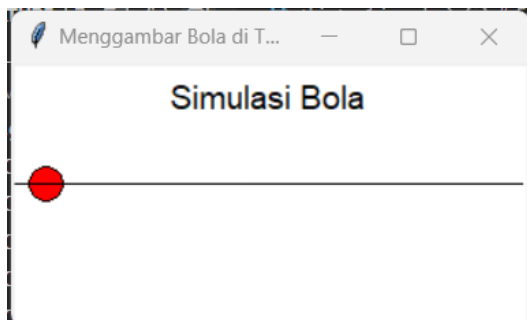
def show_coordinates(event):
    x, y = event.x, event.y
    # Gambar titik kecil di lokasi klik
    canvas.create_oval(x-5, y-5, x+5, y+5, fill="red")
    # Tampilkan koordinat di label
    label.config(text=f"Koordinat: x={x}, y={y}")
    # Tampilkan teks koordinat di canvas dekat titik
    canvas.create_text(x+30, y, text=f"({x},{y})", anchor="w", fill="blue",
font=("Arial", 10))

# Event binding
canvas.bind("<Button-1>", show_coordinates)

# Tambahkan garis sumbu
canvas.create_line(0, 0, 800, 0, fill="gray") # Garis horizontal atas
canvas.create_line(0, 0, 0, 800, fill="gray") # Garis vertikal kiri
# Kiri Atas (0, 0)
canvas.create_text(10, 10, text="(0, 0)", anchor="nw", fill="black")
# Kanan Atas (800, 0)
canvas.create_text(790, 10, text="(800, 0)", anchor="ne")
# Kiri Bawah (0, 800)
canvas.create_text(10, 790, text="(0, 800)", anchor="sw")
# Kanan Bawah (800, 800)
canvas.create_text(790, 790, text="(800, 800)", anchor="se")
# Jalankan aplikasi
root.mainloop()

```

## # 5. Menggambar object >> 1 BOLA



## # 5. Menggambar object >> 1 BOLA

```

# 1. import
import tkinter as tk

# 2. fungsi/class

# 3. canvas

```

```

root = tk.Tk()
root.title("Menggambar Bola di Tkinter")

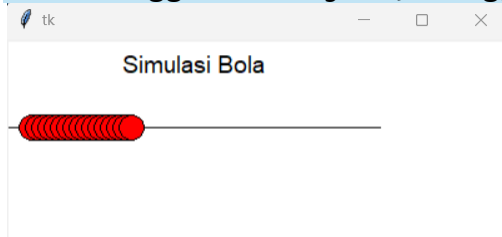
# Membuat Canvas
canvas = tk.Canvas(root, width=300, height=150, bg="white")
canvas.pack()

# 4. main program
# Menggambar canvas >> lihat sintaks
bola = canvas.create_oval(10, 60, 30, 80, fill="red")
garis = canvas.create_line(0, 70, 300, 70, fill="black")
tulisan = canvas.create_text(150, 20, text="Simulasi Bola",
font=("Arial", 14))

# 5. loop
# Mengulangi frame / frame
root.mainloop()

```

# 6. Menggambar object, Menggerakkan object >> FUNGSI >> 1 BOLA



```

# salah karena setiap frame membuat object bola baru
# 1. import
import tkinter as tk

#2. definisi
# Ukuran canvas
WIDTH = 400
HEIGHT = 400

# Posisi awal
x = 10
y = 60

# 2. fungsi/class
def gerak():
    global x, y
    canvas.create_oval(x, y, x+20, y+20, fill="red")
    x += 5
    canvas.after(100, gerak)

```

```

# 3. canvas
root = tk.Tk()
#root.title("Menggambar dan Menggerakkan Bola x += 5")
# Membuat Canvas
canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

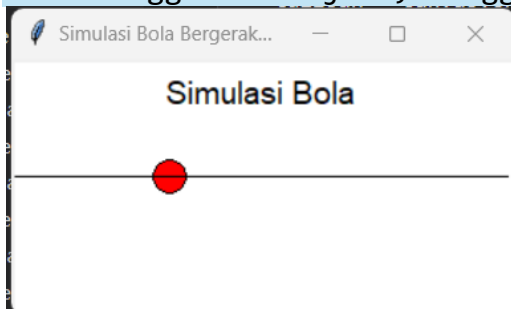
# 4. main program
# Menggambar canvas >> lihat sintaks
garis = canvas.create_line(0, 70, 300, 70, fill="black")
tulisan = canvas.create_text(150, 20, text="Simulasi Bola",
font=("Arial", 14))

gerak()

# 5. loop
# Mengulangi frame / frame
root.mainloop()

```

# 7. Menggambar object, Menggerakkan object >> FUNGSI >> 1 BOLA



# DENGAN FUNGSI BOLA, DAN MOVE

```

# 1. import
import tkinter as tk

# 2. fungsi/class
def gerak():
    canvas.move(bola, 5, 0)
    canvas.after(100, gerak)

# 3. canvas
root = tk.Tk()
root.title("Simulasi Bola Bergerak dengan move")
# Membuat Canvas
canvas = tk.Canvas(root, width=300, height=150, bg="white")
canvas.pack()

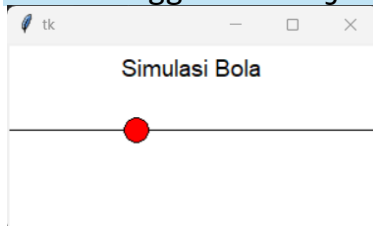
```

```
# 4. main program
# Menggambar canvas >> lihat sintaks
bola = canvas.create_oval(10, 60, 30, 80, fill="red")
garis = canvas.create_line(0, 70, 300, 70, fill="black")
tulisan = canvas.create_text(150, 20, text="Simulasi Bola",
font=("Arial", 14))
gerak()

# 5. loop
# Mengulangi frame / frame
root.mainloop()
```

Aspek	Script 6	Script 7
Cara menggambar bola	create_oval di setiap frame	Satu kali create_oval, lalu move
Jumlah objek di canvas	Semakin banyak (1 setiap frame)	Tetap satu objek
Efisiensi memori	Boros (objek menumpuk)	Hemat (satu objek digerakkan)
Visual animasi	Seperti banyak jejak bola	Bola benar-benar bergerak
Cocok untuk simulasi nyata	✗ Tidak cocok	✓ Cocok

# 8. Menggambar object, Menggerakkan object >> CLASS >> 1 BOLA



# DENGAN CLASS BOLA, DAN MOVE

```
# 1. import
import tkinter as tk

# 2. fungsi/class
# Kelas Bola
class Bola:
    def __init__(self, canvas, x, y, warna):
        self.canvas = canvas
        self.shape = canvas.create_oval(x, y, x+20, y+20, fill=warna)
```

```

        self.kecepatan = 5

    def gerak(self):
        # Gerakkan bola ke kanan
        self.canvas.move(self.shape, self.kecepatan, 0)
        # Panggil lagi fungsi gerak setelah 100ms
        self.canvas.after(100, self.gerak)

# 3. canvas
# Buat jendela utama dan canvas
root = tk.Tk()
canvas = tk.Canvas(root, width=300, height=150, bg="white")
canvas.pack()

# 4. main program
# Tambahkan garis dan teks
garis = canvas.create_line(0, 70, 300, 70, fill="black")
tulisan = canvas.create_text(150, 20, text="Simulasi Bola",
font=("Arial", 14))

# Buat satu bola dan mulai gerak
bola1 = Bola(canvas, 10, 60, "red")
bola1.gerak()

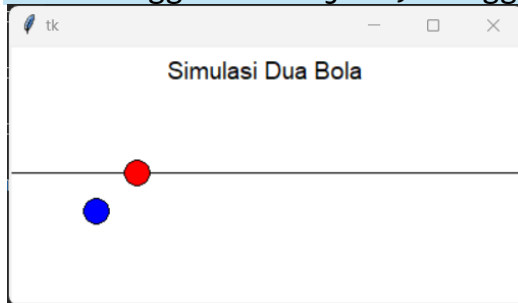
# 5. loop
root.mainloop()

```

Aspek	Script 7 (Fungsi)	Script 8 (Class)
Struktur kode	Sederhana, fungsi global	Modular, berbasis objek (OOP)
Skalabilitas (banyak bola)	Sulit	Mudah tinggal buat objek baru
Reusability (kode dapat dipakai ulang)	Rendah	Tinggi
Kejelasan tanggung jawab objek	Tidak terpisah	Jelas: setiap objek mengurus dirinya
Cocok untuk pembelajaran awal	✅ Ya	✅ Ya, jika sudah paham class
Cocok untuk simulasi kompleks	❌ Kurang cocok	✅ Sangat cocok



## # 9. Menggambar object, Menggerakkan object >> CLASS >> 2 BOLA



```
# 1. import
import tkinter as tk

# 2. fungsi/class
# Kelas Bola
class Bola:
    def __init__(self, canvas, x, y, warna, kecepatan):
        self.canvas = canvas
        self.shape = canvas.create_oval(x, y, x+20, y+20, fill=warna)
        self.kecepatan = kecepatan

    def gerak(self):
        # Gerakkan bola ke kanan
        self.canvas.move(self.shape, self.kecepatan, 0)
        # Panggil fungsi ini terus menerus
        self.canvas.after(100, self.gerak)

# 3. canvas
# Buat jendela utama dan canvas
root = tk.Tk()
canvas = tk.Canvas(root, width=400, height=200, bg="white")
canvas.pack()

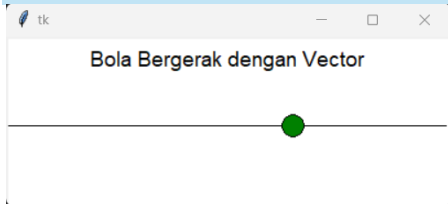
# 4. main program
# Tambahkan garis dan teks
garis = canvas.create_line(0, 100, 400, 100, fill="black")
tulisan = canvas.create_text(200, 20, text="Simulasi Dua Bola",
font=("Arial", 14))

# Buat dua bola dengan kecepatan berbeda
bola_merah = Bola(canvas, 10, 90, "red", kecepatan=5)
bola_biru = Bola(canvas, 10, 120, "blue", kecepatan=3)

# Jalankan keduanya
bola_merah.gerak()
bola_biru.gerak()
```

```
# 5. loop
root.mainloop()
```

```
# 10. Menggambar object, Menggerakkan object >> CLASS >> 2 BOLA >>
VECTOR MOVE
```



```
# 1. import
import tkinter as tk

# 2. fungsi/class
# Buat class Vector2D sendiri
class Vector:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    # Tambah vektor
    def add(self, other):
        self.x += other.x
        self.y += other.y

# Kelas Bola pakai posisi & kecepatan vektor
class Bola:
    def __init__(self, canvas, x, y, warna):
        self.canvas = canvas
        self.position = Vector(x, y)          # Posisi awal
        self.velocity = Vector(2, 0)          # Kecepatan ke kanan
        self.radius = 10                      # Ukuran bola
        self.shape = canvas.create_oval(
            x, y, x + self.radius*2, y + self.radius*2, fill=warna
        )

    def update(self):
        # Geser objek relatif dengan kecepatan
        self.canvas.move(self.shape, self.velocity.x, self.velocity.y)

        # Update juga posisi internal untuk keperluan logika lain
        self.position.add(self.velocity)

        # Jadwalkan update selanjutnya
        self.canvas.after(50, self.update)
```

```
# 3. canvas
# Inisialisasi Tkinter dan Canvas
root = tk.Tk()
canvas = tk.Canvas(root, width=400, height=150, bg="white")
canvas.pack()

# 4. main program
# Tambahkan garis dan teks
canvas.create_line(0, 80, 400, 80, fill="black")
canvas.create_text(200, 20, text="Bola Bergerak dengan Vector",
font=("Arial", 14))

# Buat bola dan jalankan
bola = Bola(canvas, 10, 70, "green")
bola.update()

# 5. loop
root.mainloop()
```

### Script 9: Class Bola dengan Parameter Kecepatan (Numerik)

#### ► Cara Kerja:

- Setiap objek Bola memiliki:
  - Posisi awal (x, y)
  - Warna
  - Kecepatan (dalam bentuk **angka**: kecepatan=5 artinya  $x += 5$ )
- Fungsi gerak() hanya memindahkan bola ke kanan sejauh kecepatan setiap 100 milidetik.
- Terdapat **dua bola** yang bergerak **dengan kecepatan berbeda**.

#### Kelebihan:

- Sederhana dan mudah dimengerti.
- Menggunakan class OOP agar dapat menambah objek bola dengan mudah.
- Bisa digunakan sebagai dasar untuk banyak bola.

#### Kekurangan:

- Gerak hanya satu arah: kanan (hanya x, tidak ada y).
- Tidak bisa dengan mudah diperluas ke arah diagonal atau logika fisika seperti percepatan atau tumbukan.
- **Tidak pakai struktur vektor**, jadi sulit jika ingin menerapkan arah gerak kompleks.

### Script 10: Class Bola dengan Vector untuk Posisi & Kecepatan

#### ► Cara Kerja:

- Dibuat class Vector(x, y) untuk menyimpan posisi dan kecepatan.

- Setiap objek Bola memiliki:
  - position: posisi dalam bentuk vektor 2D.
  - velocity: kecepatan dalam bentuk vektor 2D.
- Fungsi update() memindahkan bola berdasarkan komponen velocity.x dan velocity.y.

#### 💡 Kelebihan:

- ☒ Lebih fleksibel dan realistis:
  - Bisa gerak diagonal, atas-bawah, kiri-kanan.
  - Bisa tambahkan gaya, percepatan, tumbukan, gravitasi, drag, dll.
- ☒ Terstruktur untuk simulasi fisika nyata (mirip p5.js dan *Nature of Code*).
- ☒ Lebih cocok untuk pengembangan game/simulasi edukatif.

#### ⚠️ Kekurangan:

- Lebih kompleks: perlu memahami konsep **vektor** dan manipulasi objek.
- Untuk pemula, mungkin terlihat “berat” jika hanya ingin gerakan bola ke kanan.

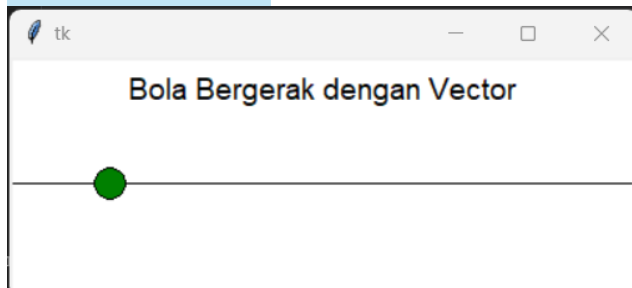
#### VS Perbandingan Langsung

Aspek	Script 9 (Tanpa Vektor)	Script 10 (Dengan Vektor)
Representasi posisi	x dan y biasa	Vector2D (position = Vector(x, y))
Representasi kecepatan	Angka tunggal (mis. 5)	Vector2D (velocity = Vector(x, y))
Arah gerak	Satu arah: kanan	Bebas (bisa kiri, atas, bawah, diagonal)
Jumlah bola	2 bola (merah dan biru)	1 bola (hijau), tapi bisa ditambah mudah
Skala waktu (after)	100ms	50ms
Potensi untuk pengembangan	Terbatas (hanya gerak x)	Sangat luas (simulasi fisika, collision, dll)
Cocok untuk pemula	<input checked="" type="checkbox"/> Ya	⚠️ Butuh sedikit pemahaman vektor
Cocok untuk simulasi fisika	<input checked="" type="checkbox"/> Kurang	<input checked="" type="checkbox"/> Sangat cocok

#### 🎓 Kesimpulan Akhir

Tujuan Belajar / Aplikasi Kamu	Gunakan Script...
Belajar dasar animasi 2 objek bergerak	✓ Script 9
Belajar konsep vektor dalam simulasi	✓ Script 10
Simulasi gerakan 2D yang kompleks (fisika, gaya, tumbukan)	✓ Script 10
Pengembangan simulasi edukasi sains	✓ Script 10

# 11. Menggambar object, Menggerakkan object >> CLASS >> 2 BOLA >> VECTOR >> COORD



```
# 1. import
import tkinter as tk

# 2. fungsi/class
# Buat class Vector2D sendiri
class Vector:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    # Tambah vektor
    def add(self, other):
        self.x += other.x
        self.y += other.y

# Kelas Bola pakai posisi & kecepatan vektor
class Bola:
    def __init__(self, canvas, x, y, warna):
        self.canvas = canvas
        self.position = Vector(x, y)          # Posisi awal
        self.velocity = Vector(2, 0)         # Kecepatan ke kanan
        self.radius = 10                     # Ukuran bola
        self.shape = canvas.create_oval(
            x, y, x + self.radius*2, y + self.radius*2, fill=warna
        )
```

```

def update(self):
    # Tambahkan kecepatan ke posisi
    self.position.add(self.velocity)
    # Update posisi gambar bola di canvas
    self.canvas.coords(
        self.shape,
        self.position.x,
        self.position.y,
        self.position.x + self.radius*2,
        self.position.y + self.radius*2
    )
    # Panggil ulang update setiap 50ms
    self.canvas.after(50, self.update)

# 3. canvas
# Inisialisasi Tkinter dan Canvas
root = tk.Tk()
canvas = tk.Canvas(root, width=400, height=150, bg="white")
canvas.pack()

# 4. main program
# Tambahkan garis dan teks
canvas.create_line(0, 80, 400, 80, fill="black")
canvas.create_text(200, 20, text="Bola Bergerak dengan Vector",
font=("Arial", 14))

# Buat bola dan jalankan
bola = Bola(canvas, 10, 70, "green")
bola.update()

# 5. loop
root.mainloop()

```

---

#### **Script 10: Update Posisi dengan canvas.move()**

```

def update(self):
    # Geser objek relatif dengan kecepatan
    self.canvas.move(self.shape, self.velocity.x, self.velocity.y)

    # Update juga posisi internal untuk keperluan logika lain
    self.position.add(self.velocity)

    # Jadwalkan update selanjutnya
    self.canvas.after(50, self.update)

```

**Cara Kerja:**

- Menggerakkan bola dengan fungsi `canvas.move()` yang menggeser objek secara relatif (relative move).
- Setelah itu, update posisi internal `self.position` dengan menambahkan velocity.
- Jadi, posisi bola di canvas digeser *berdasarkan kecepatan*.

### ● Script 11: Update Posisi dengan `canvas.coords()`

```
def update(self):
    # Tambahkan kecepatan ke posisi
    self.position.add(self.velocity)

    # Update posisi gambar bola di canvas secara absolut dengan coords
    self.canvas.coords(
        self.shape,
        self.position.x,
        self.position.y,
        self.position.x + self.radius*2,
        self.position.y + self.radius*2
    )

    # Panggil ulang update setiap 50ms
    self.canvas.after(50, self.update)
```

#### Cara Kerja:

- Pertama-tama update posisi internal `self.position` dengan kecepatan.
- Lalu set posisi **absolut** bola di canvas menggunakan `canvas.coords()`.
- Fungsi `coords()` menetapkan ulang posisi oval ke koordinat baru (x1, y1, x2, y2).
- Ini *mengatur posisi langsung*, bukan menggeser relatif.

### ⚖ Perbandingan Utama

Aspek	Script 10 ( <code>canvas.move</code> )	Script 11 ( <code>canvas.coords</code> )
Metode penggerak objek	Relatif, menggeser posisi saat ini dengan delta (velocity)	Absolut, atur posisi objek langsung di canvas
Update posisi internal	Setelah <code>move()</code> , posisi internal ditambah velocity	Sebelum update posisi gambar, posisi internal ditambah velocity
Potensi akumulasi error	Bisa terjadi error posisi jika sering <code>move()</code> , karena posisi internal bisa tidak sinkron dengan posisi canvas	Lebih presisi karena posisi canvas di-set ulang persis sesuai posisi internal

Aspek	Script 10 (canvas.move)	Script 11 (canvas.coords)
Kompatibilitas logika	Mudah untuk animasi sederhana, tapi kurang cocok jika ingin manipulasi posisi kompleks	Lebih fleksibel dan akurat untuk simulasi posisi kompleks dan koreksi posisi
Kejelasan kode	Lebih sederhana dan intuitif jika hanya ingin menggerakkan objek	Perlu perhitungan ulang posisi dan koordinat setiap frame
Kinerja	Biasanya sedikit lebih cepat karena hanya menggeser	Sedikit lebih berat karena menggambar ulang posisi

### 🌟 Kapan Pakai Mana?

Tujuan / Kebutuhan	Pilih Script
Animasi sederhana, hanya geser objek	Script 10 (move)
Simulasi fisika, kalkulasi posisi kompleks	Script 11 (coords)
Membutuhkan presisi posisi absolut	Script 11 (coords)
Ingin update posisi yang mudah dan cepat	Script 10 (move)

### 💡 Ringkasan

- `canvas.move()` menggeser posisi objek secara relatif ke posisi sekarang. Cocok jika kamu hanya ingin "memindahkan" objek tanpa perlu tahu posisi tepatnya di canvas.
- `canvas.coords()` menentukan posisi absolut objek di canvas dengan meng-set bounding box-nya. Cocok untuk simulasi fisika di mana posisi sebenarnya harus dipantau dan dikontrol secara akurat.

### ✅ Cara Menggerakkan Gambar di Tkinter: `coords()` vs `move()`

Cara	Apa yang Terjadi	Cocok untuk
<code>create_oval(...)</code>	Membuat objek baru setiap kali	Jejak, lintasan, "path"
<code>move()</code> atau <code>coords()</code>	Memindahkan objek yang sudah ada	Bola bergerak tanpa jejak

### 🟢 Inti Masalah

Gimana caranya memindahkan gambar (seperti bola) yang sudah ada di canvas?

Ada 2 cara utama di Tkinter:

#### 1. `canvas.coords(item, x1, y1, x2, y2)`

- Artinya: ubah posisi objek ke posisi tertentu.



- Cocok jika kamu tahu koordinat pasti.
- Harus hitung ulang posisi kiri atas dan kanan bawah.

```
canvas.coords(objek_id, x1, y1, x2, y2)
```

2. **canvas.move(item, dx, dy)**

- Artinya: **geser posisi objek relatif terhadap posisi sekarang.**
- Lebih mudah, karena cukup tahu arah dan jarak.
- Cocok untuk animasi atau gerak halus.

```
canvas.move(objek_id, dx, dy)
```

---

## Analogi Sehari-Hari

### Gaya Gerak Analogi

`coords()` Teleportasi (loncat ke X,Y)

`move()` Mengemudi (geser perlahan)

---

### Contoh Kode

Dengan `coords()`:

# Memindahkan bola ke posisi x, y tertentu

```
canvas.coords(  
    bola_id,  
    x,  
    y,  
    x + radius*2,  
    y + radius*2  
)
```



Dengan `move()`:

# Menggeser bola berdasarkan kecepatan

```
canvas.move(bola_id, dx, dy)
```

---

## Kapan Gunakan yang Mana?

Situasi	Gunakan
Tahu koordinat target yang pasti	<code>coords()</code>
Ingin geser bola secara halus	<code>move()</code> 
Buat animasi jatuh, loncat, gerak dinamis	<code>move()</code> 
Ingin objek langsung loncat ke posisi baru	<code>coords()</code>

---

### Tips Sempel

`move()` itu **relatif** dan cocok buat animasi.

`coords()` itu **absolut** dan cocok buat pengaturan posisi presisi.

---

## Kesimpulan Akhir

Perbandingan	move()	coords()
Jenis gerak	Relatif (geser) ✓	Absolut (lompat ke posisi)
Mudah dipakai	Ya, cukup tahu dx, dy ✓	Tidak, harus hitung posisi
Cocok animasi	Ya ✓	Kurang cocok

## 12. Membuat **bola** (lingkaran) berdiameter 10 piksel

Bagus! Kita mulai dari **tahap 1** untuk belajar Python Tkinter dengan sistem vektor (`vector.py`) dan `canvas.coords`.

### Tujuan:

- Membuat **bola** (lingkaran) berdiameter 10 piksel.
- Ditempatkan tepat **di tengah layar** (600x600 piksel).
- Gunakan sistem `Vector` agar mudah untuk mengatur posisi nanti.

### Struktur File

Untuk tahap ini, kita pakai dua file:

1. `vector.py` – berisi class `Vector`
2. `main.py` – program utama Tkinter

### `vector.py` (Class `Vector 2D`)

```
import math
```

```
class Vector:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def copy(self):
        return Vector(self.x, self.y)

    def add(self, other):
        self.x += other.x
        self.y += other.y

    def sub(self, other):
        self.x -= other.x
        self.y -= other.y

    def mult(self, scalar):
        self.x *= scalar
        self.y *= scalar

    def div(self, scalar):
        if scalar != 0:
            self.x /= scalar
            self.y /= scalar
```

```

def mag(self):
    return math.sqrt(self.x ** 2 + self.y ** 2)

def normalize(self):
    m = self.mag()
    if m != 0:
        self.div(m)

def __str__(self):
    return f"Vector({self.x}, {self.y})"

```



### main.py (Program Tkinter)

```

# SCRIPT 12 : Bola di Tengah Layar
import tkinter as tk
from vector import Vector

# Ukuran canvas
WIDTH = 600
HEIGHT = 600
BALL_DIAMETER = 10

# Inisialisasi jendela utama
root = tk.Tk()
root.title("Bola di Tengah Layar")

canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

# Posisi tengah layar (pakai vector)
center = Vector(WIDTH / 2, HEIGHT / 2)

# Hitung koordinat lingkaran dari posisi tengah
r = BALL_DIAMETER / 2
x0 = center.x - r
y0 = center.y - r
x1 = center.x + r
y1 = center.y + r

# Gambar bola
ball = canvas.create_oval(x0, y0, x1, y1, fill="blue")

# Jalankan aplikasi

```

```
root.mainloop()
```

## 💡 Penjelasan

- Kita buat `Vector` sebagai class posisi.
- Titik tengah canvas adalah `(300, 300)` karena ukuran `600×600`.
- Untuk menggambar lingkaran, digunakan `create_oval(x0, y0, x1, y1):`
  - `(x0, y0)` adalah pojok kiri atas bounding box lingkaran.
  - `(x1, y1)` adalah pojok kanan bawah.

## 13. class `Ball` agar lebih modular dan objektif.

### 🎯 Tujuan Tahap Ini:

Membuat class `Ball` dengan:

- **`x, y`** (titik tengah awal)
- **`diameter`**
- **`warna`**
- Menggambar bola di `Canvas`
- Gunakan `Vector` untuk posisi

### 🔧 Struktur File

- `vector.py` (tetap sama)
- `main.py` (pakai class `Ball`)

### 📄 `main.py` (pakai class `Ball`)

```
# SCRIPT 13 : Bola di Tengah Layar - Class Ball
import tkinter as tk
from vector import Vector

WIDTH = 600
HEIGHT = 600

class Ball:
    def __init__(self, canvas, x, y, diameter, color):
        self.canvas = canvas
        self.position = Vector(x, y)          # posisi tengah bola
        self.diameter = diameter
        self.color = color
        self.id = self.draw()

    def draw(self):
        # Hitung koordinat bounding box berdasarkan posisi tengah
        r = self.diameter / 2
        x0 = self.position.x - r
        y0 = self.position.y - r
        x1 = self.position.x + r
        y1 = self.position.y + r
        return self.canvas.create_oval(x0, y0, x1, y1, fill=self.color)

# Inisialisasi jendela Tkinter
```

```

root = tk.Tk()
root.title("Bola di Tengah - Class Ball")

canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

# Buat bola di tengah layar
center_x = WIDTH / 2
center_y = HEIGHT / 2
ball = Ball(canvas, x=center_x, y=center_y, diameter=10, color="blue")

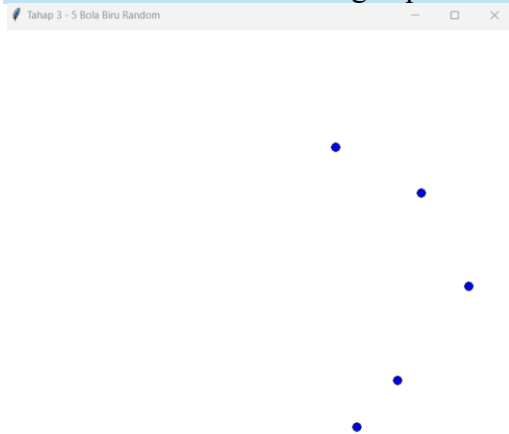
root.mainloop()

```

### Penjelasan Class Ball

- `__init__`: menyimpan canvas, posisi, diameter, warna
- `draw()`: menggambar bola di canvas berdasarkan posisi tengah (Vector)
- `self.id`: menyimpan ID objek canvas, berguna nanti untuk move, coords, dll

**Membuat 5 bola biru dengan posisi acak (x, y) di dalam canvas berukuran 600×600.**



### Tujuan:

- Tambahkan **5 bola biru**
- Posisi (x, y) ditentukan secara acak
- Tetap gunakan class Ball
- Diameter tetap 10 piksel

### Update main.py

```

import tkinter as tk
import random
from vector import Vector

WIDTH = 600
HEIGHT = 600

class Ball:
    def __init__(self, canvas, x, y, diameter, color):

```

```

        self.canvas = canvas
        self.position = Vector(x, y)
        self.diameter = diameter
        self.color = color
        self.id = self.draw()

    def draw(self):
        r = self.diameter / 2
        x0 = self.position.x - r
        y0 = self.position.y - r
        x1 = self.position.x + r
        y1 = self.position.y + r
        return self.canvas.create_oval(x0, y0, x1, y1, fill=self.color)

# Inisialisasi Tkinter
root = tk.Tk()
root.title("Tahap 3 - 5 Bola Biru Random")

canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

# Buat 5 bola biru dengan posisi acak
balls = []
for _ in range(5):
    x = random.randint(10, WIDTH - 10) # agar tidak keluar batas
    y = random.randint(10, HEIGHT - 10)
    ball = Ball(canvas, x=x, y=y, diameter=10, color="blue")
    balls.append(ball)

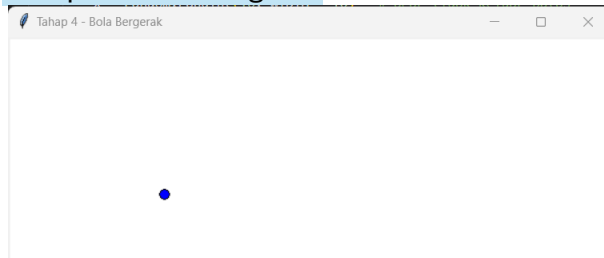
# Jalankan Tkinter loop
root.mainloop()

```

### 🔍 Penjelasan:

- `random.randint(10, WIDTH - 10)` memastikan bola tidak keluar batas canvas (karena radius = 5).
- Setiap bola disimpan dalam list `balls` (berguna nanti untuk gerakan, deteksi tabrakan, dll).

### Tahap 4 – bola bergerak



### 🎯 Tujuan:

- Tambahkan **1 bola biru di posisi (100, 100)**
- Bola **bergerak ke bawah dan ke kanan** dengan kecepatan tetap
- Gunakan `Vector` untuk kecepatan (`velocity`)

- Gunakan `canvas.coords` untuk memindahkan bola di layar

#### main.py (Tahap 4 – bola bergerak)

```
import tkinter as tk
from vector import Vector

WIDTH = 600
HEIGHT = 600

class Ball:
    def __init__(self, canvas, x, y, diameter, color):
        self.canvas = canvas
        self.position = Vector(x, y)
        self.velocity = Vector(2, 2) # kecepatan ke bawah dan kanan
        self.diameter = diameter
        self.color = color
        self.id = self.draw()

    def draw(self):
        r = self.diameter / 2
        x0 = self.position.x - r
        y0 = self.position.y - r
        x1 = self.position.x + r
        y1 = self.position.y + r
        return self.canvas.create_oval(x0, y0, x1, y1, fill=self.color)

    def update(self):
        # Tambahkan velocity ke posisi
        self.position.add(self.velocity)

        # Update posisi objek di canvas
        r = self.diameter / 2
        x0 = self.position.x - r
        y0 = self.position.y - r
        x1 = self.position.x + r
        y1 = self.position.y + r
        self.canvas.coords(self.id, x0, y0, x1, y1)

# Fungsi animasi
def animate():
    ball.update()
    root.after(20, animate) # panggil ulang setiap 20 ms

# Inisialisasi Tkinter
root = tk.Tk()
root.title("Tahap 4 - Bola Bergerak")

canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

# Buat bola biru di (100, 100)
ball = Ball(canvas, x=100, y=100, diameter=10, color="blue")

# Mulai animasi
animate()
```

```
root.mainloop()
```

### Penjelasan:

- `velocity = Vector(2, 2)` artinya bola bergerak +2px kanan dan +2px bawah setiap frame
- `self.position.add(self.velocity)` menambahkan kecepatan ke posisi
- `canvas.coords()` mengupdate posisi gambar bola

### sebuah segitiga

### Tujuan:

- Tambahkan **sebuah segitiga** (bisa mewakili pesawat/objek lain)
- Bergerak ke **atas** dengan kecepatan tetap
- Jika **melewati batas atas**, maka **muncul kembali dari bawah**
- Gunakan koordinat dengan `canvas.create_polygon()` dan `canvas.coords()` untuk pergerakan

### main.py (Tambahan untuk segitiga bergerak ke atas)

```
import tkinter as tk
from vector import Vector

WIDTH = 600
HEIGHT = 600

class Triangle:
    def __init__(self, canvas, x, y, size, color):
        self.canvas = canvas
        self.position = Vector(x, y)
        self.velocity = Vector(0, -2) # bergerak ke atas
        self.size = size
        self.color = color
        self.id = self.draw()

    def draw(self):
        # Buat segitiga menghadap ke atas
        s = self.size
        x = self.position.x
        y = self.position.y
        points = [x, y - s, x - s, y + s, x + s, y + s] # titik puncak dan
        dua bawah
        return self.canvas.create_polygon(points, fill=self.color)

    def update(self):
        # Tambahkan kecepatan ke posisi
        self.position.add(self.velocity)

        # Jika melewati batas atas, pindah ke bawah
        if self.position.y < -self.size:
            self.position.y = HEIGHT + self.size
```



```

        # Hitung ulang titik-titik segitiga
        s = self.size
        x = self.position.x
        y = self.position.y
        points = [x, y - s, x - s, y + s, x + s, y + s]
        self.canvas.coords(self.id, *points)

# Fungsi animasi
def animate():
    triangle.update()
    root.after(20, animate)

# Inisialisasi Tkinter
root = tk.Tk()
root.title("Tahap 5 - Segitiga Bergerak Looping ke Atas")

canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

# Buat segitiga di tengah bawah layar
triangle = Triangle(canvas, x=WIDTH//2, y=HEIGHT-50, size=20, color="green")

# Mulai animasi
animate()

root.mainloop()

```

---

### Penjelasan:

- Segitiga menghadap ke atas dengan 3 titik: puncak atas, sisi kiri bawah, sisi kanan bawah
- Bergerak ke atas (`velocity = Vector(0, -2)`)
- Bila `y < -size`, dianggap sudah lewat atas, lalu dipindah ke bawah dengan `y = HEIGHT + size`

---

Keren! Sekarang kita masuk ke **Tahap 6**, di mana kita akan membuat simulasi jet kecil:

---

### Tujuan Tahap 6:

- Tambahkan **segitiga jet** (seperti sebelumnya)
- Tambahkan **dua kotak kecil di belakangnya** (sebagai ekor jet)
- Jet **bergerak ke atas kanan** dengan **kecepatan awal kecil**
- Tambahkan **percepatan tetap**
- **Batasi kecepatan maksimum = 15**

---

### Hal-hal yang ditambahkan:

- `velocity`: vektor kecepatan awal kecil, misal `(1, -1)`
  - `acceleration`: vektor percepatan tetap, misal `(0.1, -0.1)`
  - Jika `velocity.magnitude() > 15`, maka percepatan tidak ditambahkan lagi
  - `draw()` membuat segitiga dan 2 kotak kecil sebagai ekor jet
-

## Kode Lengkap main.py

```
import tkinter as tk
from vector import Vector

WIDTH = 600
HEIGHT = 600
MAX_SPEED = 15

class Jet:
    def __init__(self, canvas, x, y, size, color):
        self.canvas = canvas
        self.position = Vector(x, y)
        self.velocity = Vector(1, -1)          # kecepatan awal
        self.acceleration = Vector(0.1, -0.1)  # percepatan tetap
        self.size = size
        self.color = color

        # Buat objek canvas: segitiga + 2 kotak di belakang
        self.body_id = self.draw_body()
        self.jet1_id = self.draw_jet(offset_x=-size/2.5)
        self.jet2_id = self.draw_jet(offset_x=size/2.5)

    def draw_body(self):
        s = self.size
        x = self.position.x
        y = self.position.y
        points = [x, y - s, x - s, y + s, x + s, y + s]
        return self.canvas.create_polygon(points, fill=self.color)

    def draw_jet(self, offset_x):
        # Kotak kecil di belakang jet
        s = self.size / 5
        x = self.position.x + offset_x
        y = self.position.y + self.size + s
        return self.canvas.create_rectangle(x - s, y - s, x + s, y + s,
        fill="orange")

    def update(self):
        # Tambahkan percepatan jika kecepatan belum melewati batas
        if self.velocity.magnitude() < MAX_SPEED:
            self.velocity.add(self.acceleration)

        # Update posisi
        self.position.add(self.velocity)

        # Reset jika keluar layar kanan/bawah
        if self.position.x > WIDTH + self.size:
            self.position.x = -self.size
        if self.position.y < -self.size:
            self.position.y = HEIGHT + self.size

        # Update segitiga (badan)
        s = self.size
        x = self.position.x
        y = self.position.y
        points = [x, y - s, x - s, y + s, x + s, y + s]
```

```

self.canvas.coords(self.body_id, *points)

# Update ekor jet 1 (kiri)
offset_x = -s / 2.5
jx1 = self.position.x + offset_x
jy1 = self.position.y + s + s/5
jet_size = s / 5
self.canvas.coords(self.jet1_id,
    jx1 - jet_size, jy1 - jet_size,
    jx1 + jet_size, jy1 + jet_size
)

# Update ekor jet 2 (kanan)
offset_x = s / 2.5
jx2 = self.position.x + offset_x
jy2 = self.position.y + s + s/5
self.canvas.coords(self.jet2_id,
    jx2 - jet_size, jy2 - jet_size,
    jx2 + jet_size, jy2 + jet_size
)

# Fungsi animasi
def animate():
    jet.update()
    root.after(20, animate)

# Inisialisasi Tkinter
root = tk.Tk()
root.title("Tahap 6 - Jet dengan Ekornya dan Percepatan")

canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

# Buat jet
jet = Jet(canvas, x=100, y=HEIGHT - 50, size=20, color="green")

# Jalankan animasi
animate()

root.mainloop()

```

---

### Penjelasan:

- Jet punya **velocity** dan **acceleration**
  - Setiap frame, **velocity** ditambah percepatan (selama belum mencapai 15)
  - Jet terdiri dari 3 objek canvas: 1 segitiga, 2 kotak kecil di belakangnya
  - Saat melewati kanan/atas layar, dia muncul lagi dari kiri/bawah
- 

Mantap, kita masuk ke **Tahap 7**:

---

### Tujuan Tahap 7:

- Buat **pesawat jet diam di tempat** (tidak bergerak maju)
- Bisa **berbelok kanan/kiri** menggunakan **tombol panah**

- Arah belokan memengaruhi **rotasi segitiga** (body) dan dua kotak ekor

---

### Konsep Baru:

- Kita perkenalkan `self.angle` (dalam **radian** atau derajat)
- Gunakan **trigonometri** (`sin` dan `cos`) untuk menghitung posisi titik segitiga yang berputar sesuai arah
- Kendali belok:  $\leftarrow$  mengurangi sudut,  $\rightarrow$  menambah sudut

---

### Kode `main.py` Tahap 7

```
import tkinter as tk
import math
from vector import Vector

WIDTH = 600
HEIGHT = 600

class Jet:
    def __init__(self, canvas, x, y, size, color):
        self.canvas = canvas
        self.position = Vector(x, y)
        self.size = size
        self.color = color
        self.angle = 0 # dalam radian

        # Buat objek canvas
        self.body_id = self.canvas.create_polygon(self.get_body_points(),
fill=self.color)
        self.jet1_id = self.canvas.create_rectangle(0, 0, 0, 0,
fill="orange")
        self.jet2_id = self.canvas.create_rectangle(0, 0, 0, 0,
fill="orange")

    def get_body_points(self):
        # Hitung 3 titik segitiga berdasarkan sudut rotasi
        s = self.size
        a = self.angle
        cx, cy = self.position.x, self.position.y

        # Titik puncak segitiga (ke depan)
        front = Vector(math.cos(a), math.sin(a)).mult(s)
        # Titik kiri dan kanan bawah segitiga
        left = Vector(math.cos(a + 2.5), math.sin(a + 2.5)).mult(s)
        right = Vector(math.cos(a - 2.5), math.sin(a - 2.5)).mult(s)

        return [
            cx + front.x, cy + front.y,
            cx + left.x, cy + left.y,
            cx + right.x, cy + right.y
        ]

    def get_jet_positions(self):
        # Dua ekor di belakang jet
        s = self.size / 3
        a = self.angle + math.pi # arah belakang
```

```

        offset = Vector(math.cos(a), math.sin(a)).mult(self.size * 0.6)

        # Posisi jet kiri/kanan (rotasi dikit)
        left_offset = Vector(math.cos(a + 0.3), math.sin(a +
0.3)).mult(self.size * 0.4)
        right_offset = Vector(math.cos(a - 0.3), math.sin(a -
0.3)).mult(self.size * 0.4)

        # Buat posisi rectangle (x0,y0,x1,y1)
        def rect(center):
            return (
                center.x - s + self.position.x, center.y - s +
self.position.y,
                center.x + s + self.position.x, center.y + s +
self.position.y,
            )

        return rect(left_offset), rect(right_offset)

    def update(self):
        # Update body
        self.canvas.coords(self.body_id, *self.get_body_points())

        # Update jet belakang
        jet1, jet2 = self.get_jet_positions()
        self.canvas.coords(self.jet1_id, *jet1)
        self.canvas.coords(self.jet2_id, *jet2)

    def turn_left(self, event=None):
        self.angle -= 0.1 # radian
        self.update()

    def turn_right(self, event=None):
        self.angle += 0.1
        self.update()

# Animasi diam
def animate():
    jet.update()
    root.after(20, animate)

# Setup Tkinter
root = tk.Tk()
root.title("Tahap 7 - Jet Berbelok dengan Panah")

canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

# Buat jet di tengah layar
jet = Jet(canvas, WIDTH//2, HEIGHT//2, size=30, color="green")

# Bind tombol panah
root.bind("<Left>", jet.turn_left)
root.bind("<Right>", jet.turn_right)

# Mulai animasi
animate()

```

```
root.mainloop()
```

---

### 🌀 Penjelasan Singkat:

- Sudut (`self.angle`) menentukan arah pesawat
- Titik segitiga dihitung dengan `math.cos` dan `math.sin` untuk rotasi
- Dua kotak kecil di belakang dihitung pakai sudut belakang + sedikit offset
- Jet **tidak bergerak**, hanya berputar sesuai tombol

---

Bagus! Di **Tahap 8**, kita akan:

---

### 🌀 Tujuan:

- Jet **diam**, bisa **berbelok kiri/kanan** (sudah ada)
- Sekarang, ketika **tombol panah atas ditekan**, jet akan **bergerak maju** sesuai arah hadapnya (berdasarkan sudut `angle`)

---

### ✓ Fitur Baru:

- Tambahkan **kecepatan (`velocity`)** sebagai `Vector`
- Saat tombol ↑ **ditekan**, jet akan bergerak ke depan dengan kecepatan tetap (misal: 2 piksel per frame)

---

### ✓ Kode `main.py` Tahap 8

Pastikan `vector.py` sudah ada dari tahap sebelumnya.

```
import tkinter as tk
import math
from vector import Vector

WIDTH = 600
HEIGHT = 600

class Jet:
    def __init__(self, canvas, x, y, size, color):
        self.canvas = canvas
        self.position = Vector(x, y)
        self.size = size
        self.color = color
        self.angle = 0 # dalam radian

        # Buat objek canvas
        self.body_id = self.canvas.create_polygon(self.get_body_points(),
fill=self.color)
        self.jet1_id = self.canvas.create_rectangle(0, 0, 0, 0,
fill="orange")
        self.jet2_id = self.canvas.create_rectangle(0, 0, 0, 0,
fill="orange")

    def get_body_points(self):
        # Hitung 3 titik segitiga berdasarkan sudut rotasi
        s = self.size
        a = self.angle
        cx, cy = self.position.x, self.position.y
```

```

# Titik puncak segitiga (ke depan)
front = Vector(math.cos(a), math.sin(a)).mult(s)
# Titik kiri dan kanan bawah segitiga
left = Vector(math.cos(a + 2.5), math.sin(a + 2.5)).mult(s)
right = Vector(math.cos(a - 2.5), math.sin(a - 2.5)).mult(s)

return [
    cx + front.x, cy + front.y,
    cx + left.x, cy + left.y,
    cx + right.x, cy + right.y
]

def get_jet_positions(self):
    # Dua ekor di belakang jet
    s = self.size / 3
    a = self.angle + math.pi # arah belakang
    offset = Vector(math.cos(a), math.sin(a)).mult(self.size * 0.6)

    # Posisi jet kiri/kanan (rotasi dikit)
    left_offset = Vector(math.cos(a + 0.3), math.sin(a +
0.3)).mult(self.size * 0.4)
    right_offset = Vector(math.cos(a - 0.3), math.sin(a -
0.3)).mult(self.size * 0.4)

    # Buat posisi rectangle (x0,y0,x1,y1)
    def rect(center):
        return (
            center.x - s + self.position.x, center.y - s +
self.position.y,
            center.x + s + self.position.x, center.y + s +
self.position.y,
        )

    return rect(left_offset), rect(right_offset)

def update(self):
    # Update body
    self.canvas.coords(self.body_id, *self.get_body_points())

    # Update jet belakang
    jet1, jet2 = self.get_jet_positions()
    self.canvas.coords(self.jet1_id, *jet1)
    self.canvas.coords(self.jet2_id, *jet2)

def turn_left(self, event=None):
    self.angle -= 0.1 # radian
    self.update()

def turn_right(self, event=None):
    self.angle += 0.1
    self.update()

# Animasi diam
def animate():
    jet.update()
    root.after(20, animate)

```

```
# Setup Tkinter
root = tk.Tk()
root.title("Tahap 7 - Jet Berbelok dengan Panah")

canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

# Buat jet di tengah layar
jet = Jet(canvas, WIDTH//2, HEIGHT//2, size=30, color="green")

# Bind tombol panah
root.bind("<Left>", jet.turn_left)
root.bind("<Right>", jet.turn_right)

# Mulai animasi
animate()
root.mainloop()
```

---

### Penjelasan:

- Saat ↑ ditekan → arah hadap dikonversi jadi Vector → ditambahkan ke velocity
- Jet terus bergerak karena velocity ditambahkan ke position
- Jet bisa keluar layar dan muncul kembali (pakai modulus % WIDTH, % HEIGHT)

---

Bagus! Di **Tahap 9**, kita akan menambahkan:

---

### Tujuan:

Membuat **kompas** di UI yang:

- Menunjukkan arah hadap jet
- Berputar mengikuti sudut rotasi (`angle`) jet
- Tetap **di pojok layar**, tidak ikut bergerak bersama jet

---

### Desain Kompas:

- Kompas = **lingkaran kecil** + **jarum penunjuk arah** (segitiga atau garis)
- Letak tetap di pojok kiri atas (misalnya (60, 60) dengan radius 40)
- Jarum menunjuk ke arah `angle` jet

---

```
import tkinter as tk
import math
from vector import Vector
```

```
WIDTH = 600
HEIGHT = 600
```

```
class Jet:
```



```

def __init__(self, canvas, x, y, size=30, color="green"):
    self.canvas = canvas
    self.pos = Vector(x, y)
    self.vel = Vector(0, 0)
    self.acc = Vector(0, 0)
    self.size = size
    self.color = color
    self.angle = 0 # Radian

    # Hanya buat segitiga
    self.body_id = self.canvas.create_polygon(self.get_body_points(),
fill=self.color)

def get_body_points(self):
    # Hitung 3 titik segitiga berdasarkan posisi dan arah
    a = self.angle
    s = self.size
    cx, cy = self.pos.x, self.pos.y

    front = Vector.fromAngle(a).multed(s)
    left = Vector.fromAngle(a + 2.5).multed(s * 0.6)
    right = Vector.fromAngle(a - 2.5).multed(s * 0.6)

    return [
        cx + front.x, cy + front.y,
        cx + left.x, cy + left.y,
        cx + right.x, cy + right.y
    ]

def update(self):
    # Terapkan percepatan ke kecepatan
    self.vel.add(self.acc)
    self.acc.mult(0) # Reset percepatan

    # Tambahkan efek gesekan agar tidak terus meluncur
    self.vel.mult(0.98) # Semakin dekat ke 1, semakin licin

```

```

# Update posisi
self.pos.add(self.vel)

# Perbarui posisi segitiga
self.canvas.coords(self.body_id, *self.get_body_points())

self.draw_ui()

def turn_left(self, event=None):
    self.angle -= 0.1

def turn_right(self, event=None):
    self.angle += 0.1

def boost(self, event=None):
    force = Vector.fromAngle(self.angle).multed(0.2)
    self.acc.add(force)
def draw_ui(self):
    # Hapus UI sebelumnya jika ada
    if hasattr(self, 'ui_ids'):
        for item in self.ui_ids:
            self.canvas.delete(item)

    self.ui_ids = []

    # -- Teks koordinat dan kecepatan --
    pos_text = f"Pos: ({self.pos.x:.1f}, {self.pos.y:.1f})"
    speed_text = f"Speed: {self.vel.mag():.2f}"
    angle_text = f"Angle: {math.degrees(self.angle)%360:.1f}°"

    self.ui_ids.append(
        self.canvas.create_text(80, 20, text=pos_text, anchor="w", fill="white",
font=("Consolas", 10))
    )
    self.ui_ids.append(
        self.canvas.create_text(80, 35, text=speed_text, anchor="w", fill="white",
font=("Consolas", 10))
    )

```

```

    )
    self.ui_ids.append(
        self.canvas.create_text(80, 50, text=angle_text, anchor="w", fill="white",
font=("Consolas", 10))
    )

    # -- Kompas: Segitiga arah tetap di pojok kanan atas --
    compass_size = 20
    cx, cy = 700, 40 # Posisi tetap di pojok
    a = self.angle

    # Titik segitiga kompas
    front = Vector.fromAngle(a).setted_mag(compass_size)
    left = Vector.fromAngle(a + 2.5).setted_mag(compass_size * 0.6)
    right = Vector.fromAngle(a - 2.5).setted_mag(compass_size * 0.6)

    points = [
        cx + front.x, cy + front.y,
        cx + left.x, cy + left.y,
        cx + right.x, cy + right.y,
    ]

    self.ui_ids.append(
        self.canvas.create_polygon(points, fill="red", outline="white")
    )

    self.ui_ids.append(
        self.canvas.create_text(cx, cy + 30, text="Kompas", fill="white",
font=("Arial", 9))
    )

    # Fungsi animasi terus-menerus
    def animate():
        jet.update()
        root.after(20, animate)

    # Setup Tkinter

```

```

root = tk.Tk()
root.title("Tahap 8 - Jet Bergerak Maju")

canvas = tk.Canvas(root, width=800, height=600, bg="black")
canvas.pack()

# Jet awal
jet = Jet(canvas, WIDTH//2, HEIGHT//2, size=30, color="green")

# Kontrol keyboard
root.bind("<Left>", jet.turn_left)
root.bind("<Right>", jet.turn_right)
root.bind("<Up>", jet.boost)

# Jalankan animasi
animate()
root.mainloop()

```



#### Hasil:

- Kompas bundar akan muncul di kiri atas layar
- Jarumnya selalu menunjuk ke arah jet bergerak (arah `angle`)
- Efek rotasi visual terasa seperti HUD pesawat



#### Opsi Tambahan (Jika Mau):

- Tambahkan label arah: N, E, S, W
- Ganti pointer dengan segitiga kecil (pakai `create_polygon`)

---

Bagus! Di **Tahap 10**, kita akan menambahkan fitur **menembak peluru** 🚀



#### Tujuan:

- Jet bisa **menembak peluru** saat menekan tombol `spasi`
- Peluru berupa **garis pendek** yang:
  - Bergerak ke arah hadap jet
  - **Hilang otomatis** setelah 10 langkah (frame)

---

```

import tkinter as tk
import math
from vector import Vector

```

```

WIDTH = 600

```

```
HEIGHT = 600
```

```
class Jet:
    def __init__(self, canvas, x, y, size=30, color="green"):
        self.canvas = canvas
        self.pos = Vector(x, y)
        self.vel = Vector(0, 0)
        self.acc = Vector(0, 0)
        self.size = size
        self.color = color
        self.angle = 0 # Radian
        self.bullets = [] # List peluru aktif

        # Hanya buat segitiga
        self.body_id = self.canvas.create_polygon(self.get_body_points(),
fill=self.color)

    def get_body_points(self):
        # Hitung 3 titik segitiga berdasarkan posisi dan arah
        a = self.angle
        s = self.size
        cx, cy = self.pos.x, self.pos.y

        front = Vector.fromAngle(a).multed(s)
        left = Vector.fromAngle(a + 2.5).multed(s * 0.6)
        right = Vector.fromAngle(a - 2.5).multed(s * 0.6)

        return [
            cx + front.x, cy + front.y,
            cx + left.x, cy + left.y,
            cx + right.x, cy + right.y
        ]

    def shoot(self):
        bullet = Bullet(self.canvas, self.pos, self.angle)
        self.bullets.append(bullet)

    def update(self):
        # Terapkan percepatan ke kecepatan
        self.vel.add(self.acc)
        self.acc.mult(0) # Reset percepatan

        # Tambahkan efek gesekan agar tidak terus meluncur
        self.vel.mult(0.98) # Semakin dekat ke 1, semakin licin

        # Update posisi
        self.pos.add(self.vel)

        # Perbarui posisi segitiga
        self.canvas.coords(self.body_id, *self.get_body_points())
```

```

        self.draw_ui()

    def turn_left(self, event=None):
        self.angle -= 0.1

    def turn_right(self, event=None):
        self.angle += 0.1

    def boost(self, event=None):
        force = Vector.fromAngle(self.angle).multed(0.2)
        self.acc.add(force)
    def draw_ui(self):
        # Hapus UI sebelumnya jika ada
        if hasattr(self, 'ui_ids'):
            for item in self.ui_ids:
                self.canvas.delete(item)

        self.ui_ids = []

        # -- Teks koordinat dan kecepatan --
        pos_text = f"Pos: ({self.pos.x:.1f}, {self.pos.y:.1f})"
        speed_text = f"Speed: {self.vel.mag():.2f}"
        angle_text = f"Angle: {math.degrees(self.angle)%360:.1f}°"

        self.ui_ids.append(
            self.canvas.create_text(80, 20, text=pos_text, anchor="w",
            fill="white", font=("Consolas", 10))
        )
        self.ui_ids.append(
            self.canvas.create_text(80, 35, text=speed_text, anchor="w",
            fill="white", font=("Consolas", 10))
        )
        self.ui_ids.append(
            self.canvas.create_text(80, 50, text=angle_text, anchor="w",
            fill="white", font=("Consolas", 10))
        )

        # -- Kompas: Segitiga arah tetap di pojok kanan atas --
        compass_size = 20
        cx, cy = 700, 40 # Posisi tetap di pojok
        a = self.angle

        # Titik segitiga kompas
        front = Vector.fromAngle(a).setted_mag(compass_size)
        left = Vector.fromAngle(a + 2.5).setted_mag(compass_size * 0.6)
        right = Vector.fromAngle(a - 2.5).setted_mag(compass_size * 0.6)

        points = [
            cx + front.x, cy + front.y,

```

```

        cx + left.x, cy + left.y,
        cx + right.x, cy + right.y,
    ]

    self.ui_ids.append(
        self.canvas.create_polygon(points, fill="red", outline="white")
    )

    self.ui_ids.append(
        self.canvas.create_text(cx, cy + 30, text="Kompas", fill="white",
font=("Arial", 9))
    )

    for bullet in self.bullets[:]:
        bullet.update()
        bullet.draw()
        if bullet.is_dead():
            bullet.destroy()
            self.bullets.remove(bullet)

class Bullet:
    def __init__(self, canvas, pos, angle):
        self.canvas = canvas
        self.pos = pos.copy()
        self.vel = Vector.fromAngle(angle).setted_mag(10)
        self.lifespan = 30 # Hilang setelah 10 frame
        self.line = None

    def update(self):
        self.pos.add(self.vel)
        self.lifespan -= 1

    def draw(self):
        # Hitung ujung garis pendek sebagai peluru
        end = self.pos.added(self.vel.normalized().multed(10))
        if self.line:
            self.canvas.coords(self.line, self.pos.x, self.pos.y, end.x,
end.y)
        else:
            self.line = self.canvas.create_line(
                self.pos.x, self.pos.y, end.x, end.y,
                fill="yellow", width=2
            )

    def is_dead(self):
        return self.lifespan <= 0

    def destroy(self):
        if self.line:
            self.canvas.delete(self.line)

```

```

# Fungsi animasi terus-menerus
def animate():
    jet.update()
    root.after(20, animate)

# Setup Tkinter
root = tk.Tk()
root.title("Tahap 8 - Jet Bergerak Maju")

canvas = tk.Canvas(root, width=800, height=600, bg="black")
canvas.pack()

# Jet awal
jet = Jet(canvas, WIDTH//2, HEIGHT//2, size=30, color="green")

# Kontrol keyboard
root.bind("<Left>", jet.turn_left)
root.bind("<Right>", jet.turn_right)
root.bind("<Up>", jet.boost)
root.bind("<space>", lambda e: jet.shoot())

# Jalankan animasi
animate()
root.mainloop()

```



#### Testing:

- Jalankan
- Tekan `spasi` → muncul garis kecil meluncur ke depan arah jet
- Garis menghilang setelah 10 langkah



#### Optional Upgrade:

- Ganti garis dengan lingkaran kecil (`create_oval`)
- Tambahkan efek suara atau animasi
- Deteksi tabrakan (untuk musuh di masa depan)

---

Mantap! Di **Tahap 11**, kita akan menambahkan **10 musuh/target** di layar



#### Tujuan:

- Membuat 10 objek musuh (target) berupa **bola**
  - Diletakkan secara **acak** di layar (pakai `random`)
  - Diam saja (nanti bisa bergerak atau bereaksi di tahap lanjut)
-



### ✓ Langkah Implementasi:

1. Buat class `Target`
2. Inisialisasi 10 target secara acak
3. Gambar mereka sebagai bola
4. Simpan dalam list `targets`

---

```
import tkinter as tk
import math
from vector import Vector
import random
```

```
WIDTH = 600
HEIGHT = 600
```

```
class Jet:
```

```
    def __init__(self, canvas, x, y, size=30, color="green"):
```

```
        self.canvas = canvas
```

```
        self.pos = Vector(x, y)
```

```
        self.vel = Vector(0, 0)
```

```
        self.acc = Vector(0, 0)
```

```
        self.size = size
```

```
        self.color = color
```

```
        self.angle = 0 # Radian
```

```
        self.bullets = [] # List peluru aktif
```

```
        # Hanya buat segitiga
```

```
        self.body_id = self.canvas.create_polygon(self.get_body_points(),
fill=self.color)
```

```
    def get_body_points(self):
```

```
        # Hitung 3 titik segitiga berdasarkan posisi dan arah
```

```
        a = self.angle
```

```
        s = self.size
```

```
        cx, cy = self.pos.x, self.pos.y
```

```
        front = Vector.fromAngle(a).multed(s)
```

```
        left = Vector.fromAngle(a + 2.5).multed(s * 0.6)
```

```
        right = Vector.fromAngle(a - 2.5).multed(s * 0.6)
```

```

    return [
        cx + front.x, cy + front.y,
        cx + left.x, cy + left.y,
        cx + right.x, cy + right.y
    ]

def shoot(self):
    bullet = Bullet(self.canvas, self.pos, self.angle)
    self.bullets.append(bullet)

def update(self):
    # Terapkan percepatan ke kecepatan
    self.vel.add(self.acc)
    self.acc.mult(0) # Reset percepatan

    # Tambahkan efek gesekan agar tidak terus meluncur
    self.vel.mult(0.98) # Semakin dekat ke 1, semakin licin

    # Update posisi
    self.pos.add(self.vel)

    # Perbarui posisi segitiga
    self.canvas.coords(self.body_id, *self.get_body_points())

    self.draw_ui()

    # Update musuh (sementara hanya gambar ulang)
    for enemy in enemies:
        enemy.draw()

def turn_left(self, event=None):
    self.angle -= 0.1

def turn_right(self, event=None):
    self.angle += 0.1

```

```

def boost(self, event=None):
    force = Vector.fromAngle(self.angle).multed(0.2)
    self.acc.add(force)
def draw_ui(self):
    # Hapus UI sebelumnya jika ada
    if hasattr(self, 'ui_ids'):
        for item in self.ui_ids:
            self.canvas.delete(item)

    self.ui_ids = []

    # -- Teks koordinat dan kecepatan --
    pos_text = f"Pos: ({self.pos.x:.1f}, {self.pos.y:.1f})"
    speed_text = f"Speed: {self.vel.mag():.2f}"
    angle_text = f"Angle: {math.degrees(self.angle)%360:.1f}°"

    self.ui_ids.append(
        self.canvas.create_text(80, 20, text=pos_text, anchor="w", fill="white",
font=("Consolas", 10))
    )
    self.ui_ids.append(
        self.canvas.create_text(80, 35, text=speed_text, anchor="w", fill="white",
font=("Consolas", 10))
    )
    self.ui_ids.append(
        self.canvas.create_text(80, 50, text=angle_text, anchor="w", fill="white",
font=("Consolas", 10))
    )

    # -- Kompas: Segitiga arah tetap di pojok kanan atas --
    compass_size = 20
    cx, cy = 700, 40 # Posisi tetap di pojok
    a = self.angle

    # Titik segitiga kompas
    front = Vector.fromAngle(a).setted_mag(compass_size)

```

```

left = Vector.fromAngle(a + 2.5).setted_mag(compass_size * 0.6)
right = Vector.fromAngle(a - 2.5).setted_mag(compass_size * 0.6)

points = [
    cx + front.x, cy + front.y,
    cx + left.x, cy + left.y,
    cx + right.x, cy + right.y,
]

self.ui_ids.append(
    self.canvas.create_polygon(points, fill="red", outline="white")
)

self.ui_ids.append(
    self.canvas.create_text(cx, cy + 30, text="Kompas", fill="white",
font=("Arial", 9))
)

for bullet in self.bullets[:]:
    bullet.update()
    bullet.draw()
    if bullet.is_dead():
        bullet.destroy()
    self.bullets.remove(bullet)

class Bullet:
    def __init__(self, canvas, pos, angle):
        self.canvas = canvas
        self.pos = pos.copy()
        self.vel = Vector.fromAngle(angle).setted_mag(10)
        self.lifespan = 30 # Hilang setelah 10 frame
        self.line = None

    def update(self):
        self.pos.add(self.vel)
        self.lifespan -= 1

```

```

def draw(self):
    # Hitung ujung garis pendek sebagai peluru
    end = self.pos.added(self.vel.normalized().multed(10))
    if self.line:
        self.canvas.coords(self.line, self.pos.x, self.pos.y, end.x, end.y)
    else:
        self.line = self.canvas.create_line(
            self.pos.x, self.pos.y, end.x, end.y,
            fill="yellow", width=2
        )

def is_dead(self):
    return self.lifespan <= 0

def destroy(self):
    if self.line:
        self.canvas.delete(self.line)

class Enemy:
    def __init__(self, canvas, x, y, radius=10, color="red"):
        self.canvas = canvas
        self.pos = Vector(x, y)
        self.radius = radius
        self.color = color
        self.id = self.canvas.create_oval(
            x - radius, y - radius, x + radius, y + radius,
            fill=self.color
        )

    def draw(self):
        # Perbarui posisi (diam untuk sekarang)
        x, y = self.pos.x, self.pos.y
        r = self.radius
        self.canvas.coords(self.id, x - r, y - r, x + r, y + r)

# Fungsi animasi terus-menerus
def animate():

```

```

jet.update()
root.after(20, animate)

# Setup Tkinter
root = tk.Tk()
root.title("Tahap 8 - Jet Bergerak Maju")

canvas = tk.Canvas(root, width=800, height=600, bg="black")
canvas.pack()

# Jet awal
jet = Jet(canvas, WIDTH//2, HEIGHT//2, size=30, color="green")
# Buat 10 musuh secara acak
enemies = []
for _ in range(10):
    x = random.randint(50, WIDTH - 50)
    y = random.randint(50, HEIGHT - 50)
    enemies.append(Enemy(canvas, x, y))

# Kontrol keyboard
root.bind("<Left>", jet.turn_left)
root.bind("<Right>", jet.turn_right)
root.bind("<Up>", jet.boost)
root.bind("<space>", lambda e: jet.shoot())

# Jalankan animasi
animate()
root.mainloop()

```



#### Testing:

- Jalankan
- Lihat ada 10 bola oranye tersebar secara acak di layar
- Belum bereaksi terhadap peluru (itu di tahap 12 😊)



#### Opsional:

- Bisa tambahkan animasi bergerak lambat
- Atau label angka di tengah bola

---

Keren, sekarang kita lanjut ke **Tahap 12**:

✨ **Deteksi tabrakan peluru dan target, dan hapus target jika kena peluru.**

---

### 🎯 Tujuan:

- Setiap peluru dicek apakah **menabrak** target
  - Jika ya:
    - Target dihapus dari layar
    - Target dihapus dari list `targets`
    - (Bonus: peluru juga bisa dihapus langsung)
- 

### ✅ Strategi:

- Gunakan **jarak Euclidean** antara peluru dan target
  - Jika jarak lebih kecil dari radius target → tabrakan
  - Loop:
    - Untuk setiap peluru, cek ke semua target
- 

### ✅ Modifikasi `animate()`:

🔧 *Tambahkan fungsi untuk cek tabrakan:*

```
def check_collisions():
    global bullets, targets
    remaining_targets = []
    for target in targets:
        hit = False
        for bullet in bullets:
            dx = bullet.position.x - target.position.x
            dy = bullet.position.y - target.position.y
            distance = math.hypot(dx, dy)
            if distance < target.r:
                hit = True
                bullet.life = 0 # peluru langsung mati
                break
        if not hit:
            remaining_targets.append(target)
        else:
            target.destroy()
    targets[:] = remaining_targets
```

---

🔧 *Panggil `check_collisions()` dalam `animate()`:*

```
def animate():
    jet.update()
    compass.update(jet.angle)

    # Update peluru
    alive_bullets = []
    for b in bullets:
        if b.update():
            alive_bullets.append(b)
```

```
        else:
            b.destroy()
    bullets[:] = alive_bullets

    # Cek tabrakan peluru dan target
    check_collisions()

    root.after(20, animate)
```

---

#### ✓ Hasil:

- Jika peluru menyentuh bola target → bola target **menghilang**
- Peluru juga **langsung hilang**

---

#### 🔧 Tips Pengujian:

- Tekan ↑ untuk maju
- Tekan →/← untuk arahkan jet
- Tekan spasi untuk tembak peluru
- Arahkan ke target dan tembak 🎯

---

#### ✓ Opsional:

- Tambahkan skor tiap kali musuh kena
- Tambahkan animasi ledakan

---

Bagus sekali! Di **Tahap 13**, kita akan:

---

#### 🎯 Tujuan:

1. **Menjelaskan ulang konsep peluru dan target**
2. **Buat simulasi sederhana 2 bola:**
  - 1 bola diam (target)
  - 1 bola bergerak (peluru)
  - Peluru diarahkan agar **menabrak target**
3. **Menjelaskan perhitungan arah dan kecepatan**
4. **Menampilkan info di layar agar mudah dipahami anak SMP**

---

#### 🧠 Penjelasan Konsep Peluru & Target

- **Peluru** adalah benda kecil (garis atau bola) yang bergerak lurus dengan kecepatan tetap.
- **Target** adalah bola diam di layar.
- Kita deteksi **tabrakan** dengan rumus:  
$$\text{jarak} = (x_1 - x_2)^2 + (y_1 - y_2)^2$$
  
Jika  $\text{jarak} < \text{radius}$ , berarti peluru **menabrak** target.

---

#### ⚙️ Simulasi 2 Bola (Peluru & Target)

##### ✖️ Fitur:

- Bola 1 (peluru): biru, bergerak menuju target



- Bola 2 (target): merah, diam
- Menampilkan posisi dan jarak di layar
- Menampilkan tulisan “TABRAKAN!” saat terjadi tabrakan

## ✓ Kode Lengkap (sederhana dan edukatif)

```
import tkinter as tk
import math

class Vector:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def copy(self):
        return Vector(self.x, self.y)

    def add(self, other):
        self.x += other.x
        self.y += other.y

    def sub(self, other):
        return Vector(self.x - other.x, self.y - other.y)

    def mag(self):
        return math.sqrt(self.x ** 2 + self.y ** 2)

    def normalize(self):
        m = self.mag()
        if m != 0:
            self.x /= m
            self.y /= m

    def mult(self, n):
        self.x *= n
        self.y *= n

class Ball:
    def __init__(self, canvas, x, y, color):
        self.canvas = canvas
        self.position = Vector(x, y)
        self.r = 15
        self.color = color
        self.id = canvas.create_oval(
            x - self.r, y - self.r, x + self.r, y + self.r, fill=color
        )

    def move(self, velocity):
        self.position.add(velocity)
        x, y = self.position.x, self.position.y
        self.canvas.coords(self.id, x - self.r, y - self.r, x + self.r, y +
self.r)

def update():
    global message_id
```

```

# gerakkan peluru
bullet.move(velocity)

# hitung jarak ke target
dx = bullet.position.x - target.position.x
dy = bullet.position.y - target.position.y
distance = math.hypot(dx, dy)

# update tulisan info
canvas.itemconfig(info_text, text=f"Jarak = {distance:.2f}")

# cek tabrakan
if distance < bullet.r + target.r:
    canvas.itemconfig(info_text, text="TABRAKAN!")

root.after(50, update)

# Inisialisasi
root = tk.Tk()
root.title("Simulasi Peluru dan Target")
canvas = tk.Canvas(root, width=600, height=400, bg="white")
canvas.pack()

# Target diam di tengah kanan
target = Ball(canvas, 500, 200, "red")

# Peluru mulai dari kiri
bullet = Ball(canvas, 100, 100, "blue")

# Hitung vektor arah dari peluru ke target
direction = target.position.sub(bullet.position)
direction.normalize()
direction.mult(5) # kecepatan tetap
velocity = direction

# Info tulisan
info_text = canvas.create_text(300, 20, text="", font=("Arial", 14))

update()
root.mainloop()

```

---

### Penjelasan:

1. Bola biru = peluru
2. Bola merah = target diam
3. Peluru **bergerak lurus ke arah target**
4. Jarak dihitung terus-menerus
5. Kalau jaraknya kecil → muncul tulisan **TABRAKAN!**

---

### Tambahan Pengetahuan:

- `direction = target - bullet` → mencari arah
  - `normalize()` → menjadikan arah 1 satuan
  - `mult(5)` → membuat peluru jalan 5 piksel per update
-

Keren! Di **Tahap 14**, kita akan menambahkan:

### 🎯 Tujuan:

1. **Skor**: setiap kali peluru mengenai target, skor bertambah.
2. **UI "Game Selesai"**: jika semua target habis, muncul tulisan "Game Selesai".
3. **Efek visual**: saat jet menembak (spasi ditekan), **segitiga jet jadi merah** sebentar sebagai animasi tembakan.

### ✅ Yang akan kita ubah:

- Tambahkan `score` dan tampilkan di atas layar.
- Tambahkan pengecekan jika `len(targets) == 0`.
- Tambahkan efek `jet.set_color('red')` saat tembak dan reset ke putih setelah beberapa milidetik.

```
import tkinter as tk
import math
from vector import Vector
import random

WIDTH = 800
HEIGHT = 600

class Jet:
    def __init__(self, canvas, x, y, size=30, color="green"):
        self.canvas = canvas
        self.pos = Vector(x, y)
        self.vel = Vector(0, 0)
        self.acc = Vector(0, 0)
        self.size = size
        self.color = color
        self.angle = 0
        self.bullets = []
        self.score = 0
        self.shoot_timer = 0
        self.ui_ids = []

        self.body_id = self.canvas.create_polygon(self.get_body_points(),
        fill=self.color)

    def get_body_points(self):
        a = self.angle
        s = self.size
        cx, cy = self.pos.x, self.pos.y

        front = Vector.fromAngle(a).multed(s)
        left = Vector.fromAngle(a + 2.5).multed(s * 0.6)
```

```

right = Vector.fromAngle(a - 2.5).multed(s * 0.6)

return [
    cx + front.x, cy + front.y,
    cx + left.x, cy + left.y,
    cx + right.x, cy + right.y
]

def shoot(self):
    bullet = Bullet(self.canvas, self.pos, self.angle)
    self.bullets.append(bullet)
    self.shoot_timer = 5

def update(self):
    self.vel.add(self.acc)
    self.acc.mult(0)
    self.vel.mult(0.98)
    self.pos.add(self.vel)

    self.pos.x %= WIDTH
    self.pos.y %= HEIGHT

    self.canvas.coords(self.body_id, *self.get_body_points())
    self.draw_ui()

    for bullet in self.bullets[:]:
        bullet.update()
        bullet.draw()
        if bullet.is_dead():
            bullet.destroy()
            self.bullets.remove(bullet)

    for bullet in self.bullets[:]:
        for enemy in enemies[:]:
            if enemy.hit_by(bullet):
                enemy.trigger_hit()
                self.canvas.delete(enemy.id)
                enemies.remove(enemy)

                self.canvas.delete(bullet.line)
                self.bullets.remove(bullet)

                self.score += 1
                break

    if self.shoot_timer > 0:
        self.shoot_timer -= 1
        self.canvas.itemconfig(self.body_id, fill="red")
    else:
        self.canvas.itemconfig(self.body_id, fill=self.color)

```

```

def turn_left(self, event=None):
    self.angle -= 0.1

def turn_right(self, event=None):
    self.angle += 0.1

def boost(self, event=None):
    force = Vector.fromAngle(self.angle).multed(0.2)
    self.acc.add(force)

def draw_ui(self):
    for item in self.ui_ids:
        self.canvas.delete(item)
    self.ui_ids = []

    pos_text = f"Pos: ({self.pos.x:.1f}, {self.pos.y:.1f})"
    speed_text = f"Speed: {self.vel.mag():.2f}"
    angle_text = f"Angle: {math.degrees(self.angle)%360:.1f}°"
    score_text = f"Skor: {self.score}"

    self.ui_ids.append(self.canvas.create_text(80, 20, text=pos_text,
        anchor="w", fill="white", font=("Consolas", 10)))
    self.ui_ids.append(self.canvas.create_text(80, 35, text=speed_text,
        anchor="w", fill="white", font=("Consolas", 10)))
    self.ui_ids.append(self.canvas.create_text(80, 50, text=angle_text,
        anchor="w", fill="white", font=("Consolas", 10)))
    self.ui_ids.append(self.canvas.create_text(80, 70, text=score_text,
        anchor="w", fill="white", font=("Consolas", 10)))

    # Kompas
    compass_size = 20
    cx, cy = 700, 40
    a = self.angle
    front = Vector.fromAngle(a).setted_mag(compass_size)
    left = Vector.fromAngle(a + 2.5).setted_mag(compass_size * 0.6)
    right = Vector.fromAngle(a - 2.5).setted_mag(compass_size * 0.6)

    points = [
        cx + front.x, cy + front.y,
        cx + left.x, cy + left.y,
        cx + right.x, cy + right.y,
    ]

    self.ui_ids.append(self.canvas.create_polygon(points, fill="red",
        outline="white"))
    self.ui_ids.append(self.canvas.create_text(cx, cy + 30,
        text="Kompas", fill="white", font=("Arial", 9)))

    if not enemies:

```

```

        self.ui_ids.append(
            self.canvas.create_text(400, 300, text="Game Selesai",
fill="yellow", font=("Arial", 24, "bold"))
        )

class Bullet:
    def __init__(self, canvas, pos, angle):
        self.canvas = canvas
        self.pos = pos.copy()
        self.vel = Vector.fromAngle(angle).setted_mag(10)
        self.lifespan = 30
        self.line = None

    def update(self):
        self.pos.add(self.vel)
        self.lifespan -= 1
        self.pos.x %= WIDTH
        self.pos.y %= HEIGHT

    def draw(self):
        end = self.pos.added(self.vel.normalized().multed(10))
        if self.line:
            self.canvas.coords(self.line, self.pos.x, self.pos.y, end.x,
end.y)
        else:
            self.line = self.canvas.create_line(
                self.pos.x, self.pos.y, end.x, end.y,
                fill="yellow", width=2
            )

    def is_dead(self):
        return self.lifespan <= 0

    def destroy(self):
        if self.line:
            self.canvas.delete(self.line)

class Enemy:
    def __init__(self, canvas, x, y, radius=10, color="red"):
        self.canvas = canvas
        self.pos = Vector(x, y)
        self.radius = radius
        self.color = color
        self.id = self.canvas.create_oval(
            x - radius, y - radius, x + radius, y + radius,
            fill=self.color
        )
        self.hit_timer = 0

    def draw(self):

```

```

        if self.hit_timer > 0:
            self.hit_timer -= 1
            if self.hit_timer % 2 == 0:
                self.canvas.itemconfig(self.id, fill="white")
            else:
                self.canvas.itemconfig(self.id, fill="red")
        else:
            self.canvas.itemconfig(self.id, fill=self.color)

        x, y = self.pos.x, self.pos.y
        r = self.radius
        self.canvas.coords(self.id, x - r, y - r, x + r, y + r)

    def hit_by(self, bullet):
        dx = bullet.pos.x - self.pos.x
        dy = bullet.pos.y - self.pos.y
        distance = math.sqrt(dx*dx + dy*dy)
        return distance < self.radius

    def trigger_hit(self):
        self.hit_timer = 6

def animate():
    jet.update()
    for enemy in enemies:
        enemy.draw()
    root.after(20, animate)

# Setup Tkinter
root = tk.Tk()
root.title("Jet Tembak Musuh")

canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="black")
canvas.pack()

jet = Jet(canvas, WIDTH//2, HEIGHT//2, size=30, color="green")

# Buat musuh acak
enemies = []
for _ in range(10):
    x = random.randint(50, WIDTH - 50)
    y = random.randint(50, HEIGHT - 50)
    enemies.append(Enemy(canvas, x, y))

# Kontrol
root.bind("<Left>", jet.turn_left)
root.bind("<Right>", jet.turn_right)
root.bind("<Up>", jet.boost)
root.bind("<space>", lambda e: jet.shoot())

```

```
animate()  
root.mainloop()
```



