

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 => ...)</code>
Filter list	<code>[d for d in list if ...]</code>	<code>`array.filter(obj =></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 terbesar
min()	Ambil data dengan nilai terkecil
sum()	Jumlahkan semua nilai
len()	Hitung jumlah item dalam list
sorted()	Urutkan 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 objek itu sendiri (instans dari class)

Fungsi	Artinya
<code>super()</code>	Menunjuk ke kelas induk (parent) , 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

# Defini dan Class-----

# Canvas-----
root = tk.Tk()
root.title("Belajar Canvas Tkinter")
canvas = tk.Canvas(root, width=800, height=800, bg="white")

# Main Program-----
#.....

# Loop -----
canvas.pack()
root.mainloop()
```

✓ Tabel Fungsi Dasar canvas Tkinter

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

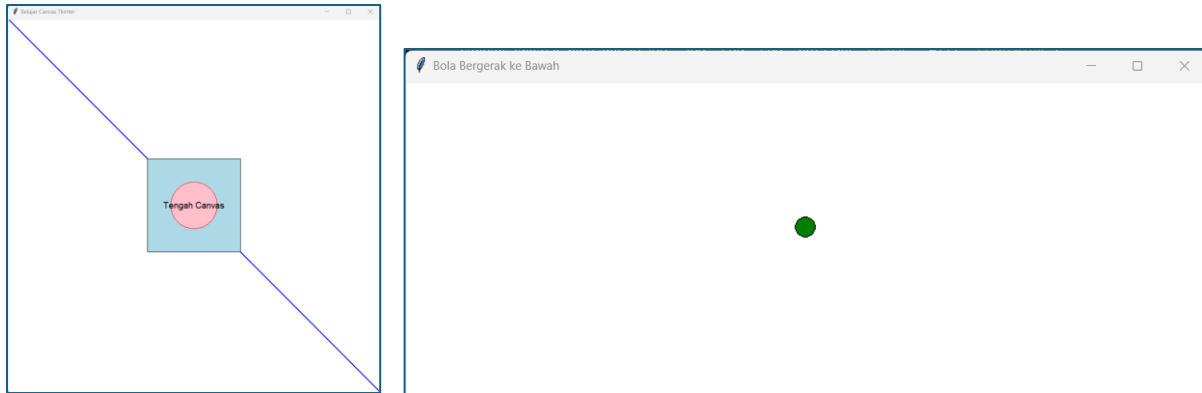
```
canvas.create_rectangle(300, 300, 500, 500, outline="black", fill="lightblue")
```

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")
```



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

# WINDOW UTAMA (ROOT)& Canvas-----
root = tk.Tk()
root.title("Belajar Canvas Tkinter")
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")

# Loop Program-----
canvas.pack()
root.mainloop()
```

4. Simulasi Gaya/Animasi (contoh sederhana)

Rumus dan Perhitungan

1. Variabel Utama

- posisi: Objek oval (bola) di canvas
- Koordinat awal: (390, 0, 410, 20) → bola diameter 20px di tengah atas layar
- Pergerakan: +5px vertikal tiap frame
- Interval waktu: 50ms (20 frame/detik)

2. Mekanisme Gerakan

canvas.move(objek, dx, dy)

Parameter:

- dx: 0 (tidak ada pergeseran horizontal)
- dy: 5 (pergeseran vertikal ke bawah)

Perhitungan Posisi:

Frame 0: (390, 0, 410, 20)

Frame 1: (390, 5, 410, 25)

Frame 2: (390, 10, 410, 30)

...

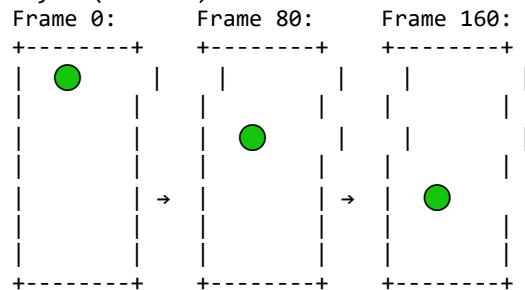
Frame N: (390, 5*N, 410, 20+5*N)

Contoh Perhitungan Frame-by-Frame

Frame	Waktu (ms)	Posisi Atas (y1)	Posisi Bawah (y2)	Kecepatan
0	0	0	20	5px/frame
1	50	5	25	↓
2	100	10	30	↓
3	150	15	35	↓
...	↓
160	8000	800	820	↓

Visualisasi Gerakan

Layar (800x800):



Karakteristik Gerakan

1. **Linear:** Bergerak lurus ke bawah tanpa percepatan
 2. **Konstan:** Kecepatan tetap 5px/frame (100px/detik)
 3. **Sederhana:** Tidak ada fisika kompleks (gravitasi, gesekan, dll)
-

Perhitungan Kecepatan

- **Kecepatan Pixel:**

5 px/frame × (1000ms/50ms) = 100 px/detik

- **Waktu sampai bawah:**

800px ÷ 100px/detik = 8 detik

Perbedaan dengan Script Vector

1. **Tanpa Konsep Fisika:**
 - o Tidak menggunakan vektor/kecepatan/percepatan
 - o Murni pergeseran pixel-based
2. **Implementasi Minimalis:**
 - o Hanya 1 objek yang digerakkan
 - o Tidak ada interaksi dengan mouse/tepi layar
3. **Performansi Ringan:**
 - o Cocok untuk animasi dasar
 - o Konsumsi resource sangat rendah

```
#2. BOLA BERGERAK KE BAWAH
import tkinter as tk

# Definisi, Fungsi, Class, dll -----
def gerak():
    canvas.move(posisi, 0, 5) # geser 5 piksel ke bawah
    canvas.after(50, gerak)   # ulangi tiap 50 milidetik

# Window Utama, Canvas-----
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")
gerak()

# Loop Program-----
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.

5. Berikut ini penjelasan sistem koordinat di Tkinter Canvas



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

Dalam Tkinter Canvas:

1. **Titik (0, 0)** ada di **pojok kiri atas**.
2. Arah **kanan** = tambah **X**
3. 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

```
#3. SISTEM KOORDINAT DI TKINTER CANVAS 1
import tkinter as tk

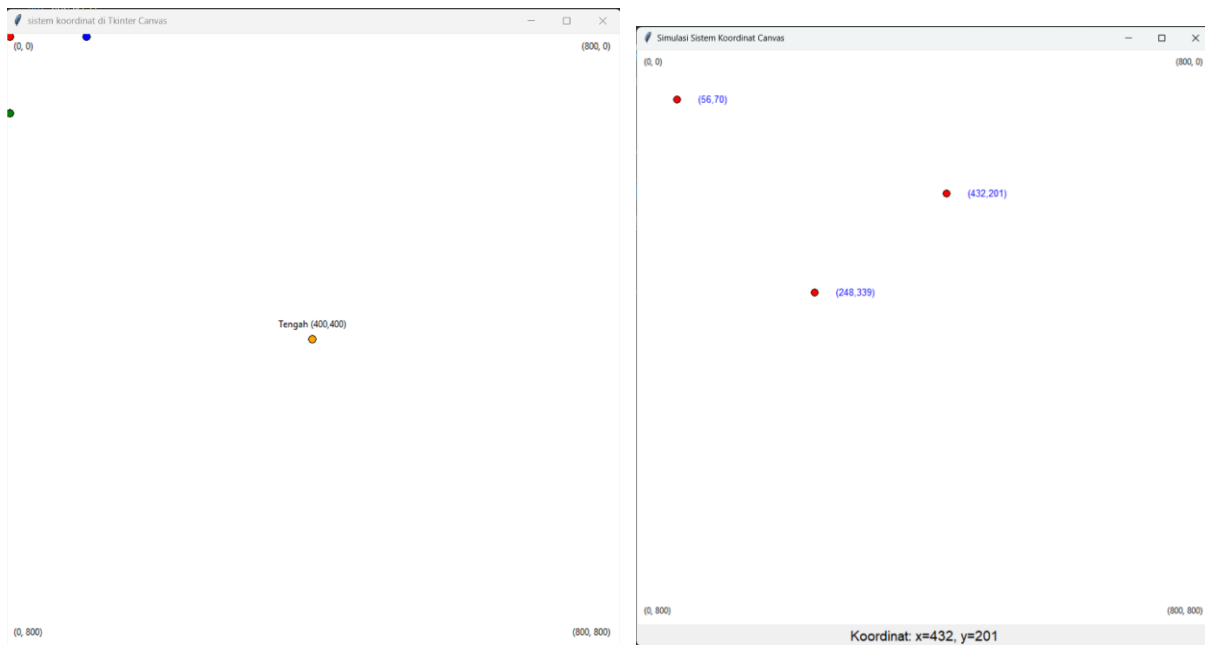
# Window Utama, Canvas-----
root = tk.Tk()
root.title("sistem koordinat di Tkinter Canvas")
canvas = tk.Canvas(root, width=800, height=800, bg="white")

# 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")

# Loop Program-----
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

```
import tkinter as tk

# Window Utama, Canvas-----
root = tk.Tk()
root.title("Simulasi Sistem Koordinat Canvas")
# Buat Canvas 800x800
canvas = tk.Canvas(root, width=800, height=800, bg="white")
canvas.pack()

# Main Program-----
# 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))

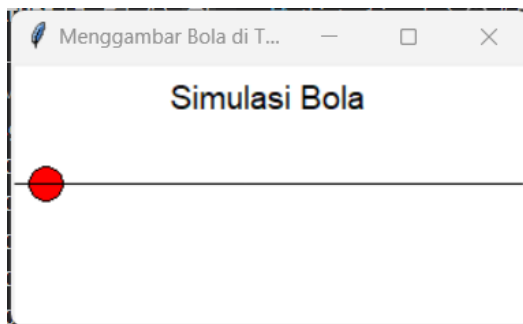
# Event binding
canvas.bind("<Button-1>", show_coordinates)

# 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")

# Loop Program-----
root.mainloop()
```

6. Menggambar object >> 1 BOLA



```
# 5. Menggambar object >> 1 BOLA

# 1. import
import tkinter as tk

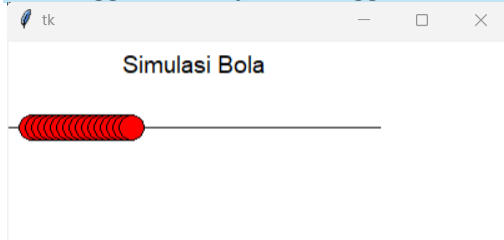
# 2. fungsi/class

# Window Utama, 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()
```

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



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

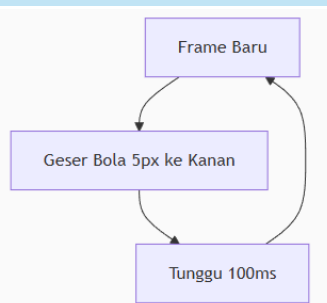
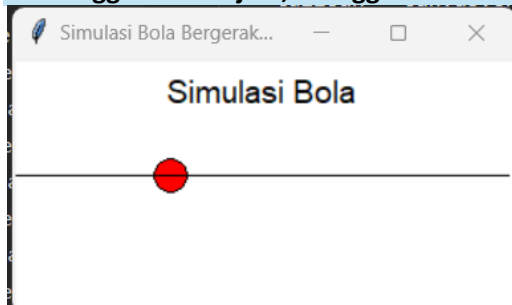
# Definisi, Fungsi, Class, dll -----
# Ukuran canvas
WIDTH = 400
HEIGHT = 400
# Posisi awal
x = 10
y = 60
# fungsi/class
def gerak():
    global x, y
    canvas.create_oval(x, y, x+20, y+20, fill="red")
    x += 5
    canvas.after(100, gerak)

# Window Utama, 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()

# Loop Program-----
# Mengulangi frame / frame
root.mainloop()
```

8. Menggambar object, Menggerakkan object >> FUNGSI >> 1 BOLA



Rumus dan Perhitungan

1. Variabel Utama

- bola: Objek oval (diameter 20px) di posisi awal (10,60)-(30,80)
- garis: Garis horizontal sebagai referensi (y=70)
- Pergerakan: +5px horizontal tiap frame
- Interval waktu: 100ms (10 frame/detik)

2. Mekanisme Gerakan

canvas.move(objek, dx, dy)

Parameter:

- dx: 5 (pergeseran horizontal ke kanan)
- dy: 0 (tidak ada pergeseran vertikal)

Perhitungan Posisi:

Frame 0: (10,60,30,80)

Frame 1: (15,60,35,80)

Frame 2: (20,60,40,80)

...

Frame N: (10+5*N,60,30+5*N,80)

Contoh Perhitungan Frame-by-Frame

Frame	Waktu (ms)	Posisi Kiri (x1)	Posisi Kanan (x2)	Kecepatan
0	0	10	30	5px/frame
1	100	15	35	→
2	200	20	40	→
...	→
58	5800	300	320	→

Visualisasi Gerakan

Layar (300x150):

Frame 0: Frame 20: Frame 58:

```
+-----+ +-----+ +-----+
|   ●   | |   ●   | |   ●   |
|_____| |_____| |_____|
|Simulasi Bola| |Simulasi Bola| |Simulasi Bola|
+-----+ +-----+ +-----+
```

Karakteristik Gerakan

1. **Linear:** Bergerak lurus ke kanan tanpa percepatan
 2. **Konstan:** Kecepatan tetap 5px/frame (50px/detik)
 3. **Sederhana:** Tidak ada fisika kompleks
 4. **Presisi:** Selalu sejajar dengan garis referensi (y=60-80)
-

Perhitungan Kecepatan & Durasi

- **Kecepatan Pixel:**

5 px/frame × (1000ms/100ms) = 50 px/detik

- **Waktu sampai kanan:**

(300-30)px ÷ 50px/detik = 5.4 detik

Komponen Tambahan

1. **Garis Referensi:**

- Posisi tetap di y=70
- Membantu verifikasi bola tidak bergerak vertikal

2. **Text Statis:**

- Posisi tetap di (150,20)
- Tidak terpengaruh animasi

Perbedaan dengan Script Sebelumnya

1. **Gerakan Horizontal** (bukan vertikal)
2. **Ada Komponen Tambahan** (garis dan text)
3. **Interval Waktu Lebih Lama** (100ms vs 50ms)
4. **Ukuran Canvas Lebih Kecil** (300x150 vs 800x800)

Potensi Error

1. **Bola Keluar Canvas:**

- Pada frame 58 (x2=320) melebihi lebar canvas (300)
- Solusi: Tambahkan pengecekan tepi

if canvas.coords(bola)[2] > 300: *# jika x2 > 300*

canvas.coords(bola, 10,60,30,80) *# reset posisi*

2. **Akurasi Waktu:**

- after() tidak menjamin timing tepat 100ms
- Bergantung pada beban sistem

```
# DENGAN FINGSI 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 deengan move")
# Membuat Canvas
canvas = tk.Canvas(root, width=300, height=150, bg="white")
canvas.pack()

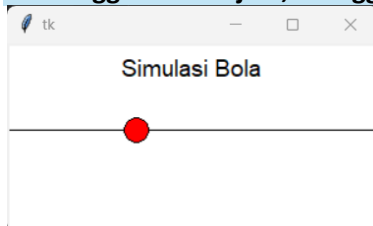
# 4. main program
# Menggambar canvas >> lihat sintaks
bola = canvas.create_oval(10, 60, 30, 80, fill="red")
garis = canvas.create_line(0, 70, 300, 70, fill="black")
tulisan = canvas.create_text(150, 20, text="Simulasi Bola", font=("Arial", 14))
```


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

9. 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()

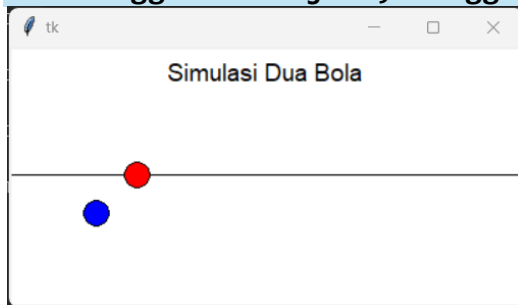
# 4. main program
# Tambahan garis dan teks
garis = canvas.create_line(0, 70, 300, 70, fill="black")
tulisan = canvas.create_text(150, 20, text="Simulasi Bola", font=("Arial", 14))
```

```
# 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

10. 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))

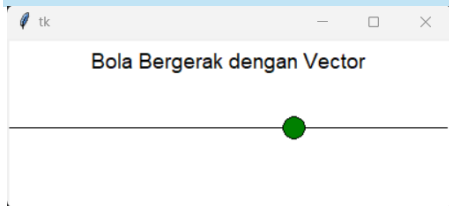
# Buat dua bola dengan kecepatan berbeda
bola_merah = Bola(canvas, 10, 90, "red", kecepatan=5)
bola_biru = Bola(canvas, 10, 120, "blue", kecepatan=3)

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

# 5. loop
root.mainloop()

```

11. 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
# Tambahan 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 ingingerakkan bola ke kanan.

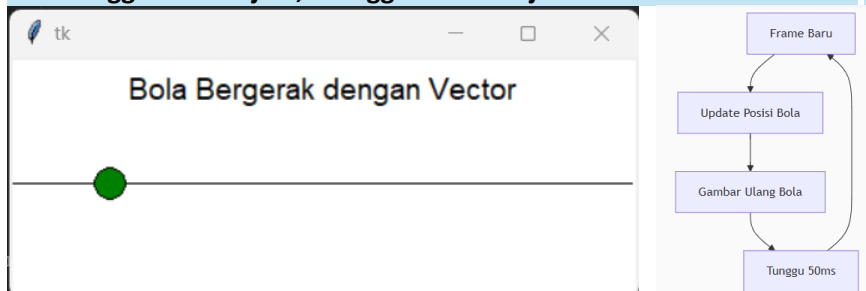
🆚 Perbandingan Langsung

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	<input checked="" type="checkbox"/> Script 9
Belajar konsep vektor dalam simulasi	<input checked="" type="checkbox"/> Script 10
Simulasi gerakan 2D yang kompleks (fisika, gaya, tumbukan)	<input checked="" type="checkbox"/> Script 10
Pengembangan simulasi edukasi sains	<input checked="" type="checkbox"/> Script 10

12. Menggambar object, Menggerakkan object >> CLASS >> 2 BOLA >> VECTOR >> COORD



Rumus dan Perhitungan

1. Variabel Utama

- position: Vektor posisi bola (x,y)
- velocity: Vektor kecepatan (2,0) → konstan ke kanan
- radius: 10 (diameter bola 20px)
- Interval waktu: 50ms (20 frame/detik)

2. Mekanisme Gerakan

position.x += velocity.x

position.y += velocity.y

Perhitungan Posisi:

Frame 0: (10,70)-(30,90)

Frame 1: (12,70)-(32,90)

Frame 2: (14,70)-(34,90)

...

Frame N: (10+2*N,70)-(30+2*N,90)

Contoh Perhitungan Frame-by-Frame

Frame	Waktu (ms)	Posisi (x1,y1)	Posisi (x2,y2)	Kecepatan
0	0	(10,70)	(30,90)	(2,0)
1	50	(12,70)	(32,90)	→
2	100	(14,70)	(34,90)	→
...	→
185	9250	(380,70)	(400,90)	→

Visualisasi Gerakan

Frame 0: Frame 50: Frame 100:

```

+-----+ +-----+ +-----+
|   ●   | |   ●   | |   ●   |
|_____| |_____| |_____|
|Teks Judul| |Teks Judul| |Teks Judul|
+-----+ +-----+ +-----+
```

Karakteristik Gerakan

1. **Gerakan Linear:** Lurus ke kanan dengan kecepatan konstan
 2. **Presisi Matematis:** Menggunakan operasi vektor
 3. **Frame Rate Konsisten:** 20 FPS (50ms per frame)
 4. **Dasar Fisika:** Memisahkan konsep posisi dan kecepatan
-

Perhitungan Kecepatan & Durasi

- **Kecepatan Pixel:**

2 px/frame × 20 frame/detik = 40 px/detik

- **Waktu sampai kanan:**

(400-30)px ÷ 40px/detik ≈ 9.25 detik

Implementasi Class Vector

```
class Vector:
```

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

```
    def add(self, other): # Operasi penjumlahan vektor
        self.x += other.x
        self.y += other.y
```

Contoh Penggunaan:

```
pos = Vector(10,70)
```

```
vel = Vector(2,0)
```

```
pos.add(vel) # pos menjadi (12,70)
```

Perbedaan dengan Script Move() Sederhana

1. **Struktur Berorientasi Objek:**
 - o Memisahkan logika vektor dan bola
 - o Lebih mudah dikembangkan
2. **Dasar Fisika:**
 - o Memisahkan posisi dan kecepatan
 - o Potensi untuk ditambah percepatan
3. **Presisi Koordinat:**
 - o Menggunakan float untuk koordinat
 - o Bisa diubah jadi integer untuk koordinat pixel

```
# 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 (canvas.move)	Script 11 (canvas.coords)
Metode penggerak objek	Relatif, menggeser posisi saat ini dengan delta (velocity)	Absolut, atur posisi objek langsung di canvas
Update posisi internal	Setelah move(), posisi internal ditambah velocity	Sebelum update posisi gambar, posisi internal ditambah velocity
Potensi akumulasi error	Bisa terjadi error posisi jika sering move(), karena posisi internal bisa tidak sinkron dengan posisi canvas	Lebih presisi karena posisi canvas di-set ulang persis sesuai posisi internal
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	coords()
Ingin geser bola secara halus	move() ✓
Buat animasi jatuh, loncat, gerak dinamis	move() ✓
Ingin objek langsung loncat ke posisi baru	coords()

💡 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

13. 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})"

```

Bola di Tengah Layar

 main.py (Program Tkinter)

```

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

# Definisi, Fungsi, Class, dll -----
# Ukuran canvas
WIDTH = 600
HEIGHT = 600
BALL_DIAMETER = 10

# Window Utama, Canvas-----
root = tk.Tk()
root.title("Bola di Tengah Layar")

```

```

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

# Main Program-----
# 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")

# Loop Program-----
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.

14. 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

# Definisi, Fungsi, Class, dll -----
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)

# Window Utama, Canvas-----
root = tk.Tk()
root.title("Bola di Tengah - Class Ball")
canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

# Main Program-----
# 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")

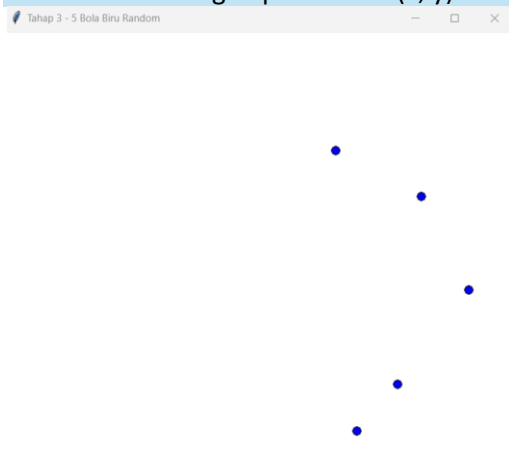
# Loop Program-----
root.mainloop()

```

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

15. 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

# Definisi, Fungsi, Class, dll -----
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)

# Window Utama, Canvas-----
root = tk.Tk()
root.title("Tahap 3 - 5 Bola Biru Random")
canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

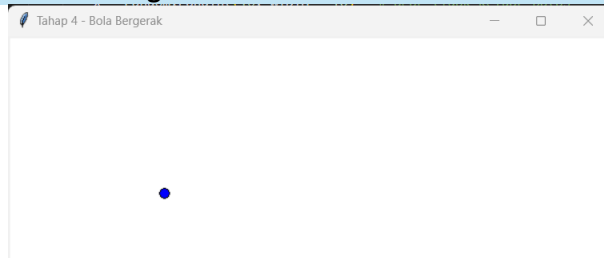
# Main Program-----
# Buat 5 bola biru dengan posisi acak
balls = []
for _ in range(5):
    x = random.randint(10, WIDTH - 10) # agar tidak keluar batas
    y = random.randint(10, HEIGHT - 10)
    ball = Ball(canvas, x=x, y=y, diameter=10, color="blue")
    balls.append(ball)

# Loop Program-----
root.mainloop()
```

Penjelasan:

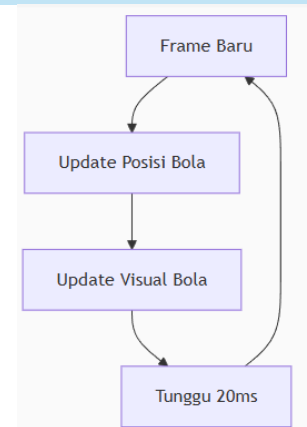
- `random.randint(10, WIDTH - 10)` memastikan bola tidak keluar batas canvas (karena radius = 5).
- Setiap bola disimpan dalam list `balls` (berguna nanti untuk gerakan, deteksi tabrakan, dll).

16. 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



Rumus dan Perhitungan

1. Variabel Utama

- position: Vektor posisi bola (x,y)
- velocity: Vektor kecepatan (2,2) → diagonal kanan-bawah
- diameter: 10 (radius 5px)
- Interval waktu: 20ms (50 frame/detik)

2. Mekanisme Gerakan

python

Copy

Download

```
position.x += velocity.x
```

```
position.y += velocity.y
```

Perhitungan Posisi:

text

Copy

Download

Frame 0: (100,100)-(110,110)

Frame 1: (102,102)-(112,112)

Frame 2: (104,104)-(114,114)

...

Frame N: (100+2*N,100+2*N)-(110+2*N,110+2*N)

Contoh Perhitungan Frame-by-Frame

Frame	Waktu (ms)	Posisi (x,y)	Kecepatan	Keterangan
0	0	(100,100)	(2,2)	Posisi awal
1	20	(102,102)	(2,2)	Bergerak kanan-bawah
2	40	(104,104)	(2,2)	

Frame	Waktu (ms)	Posisi (x,y)	Kecepatan	Keterangan
...	
250	5000	(600,600)	(2,2)	Keluar canvas (600x600)

Visualisasi Gerakan

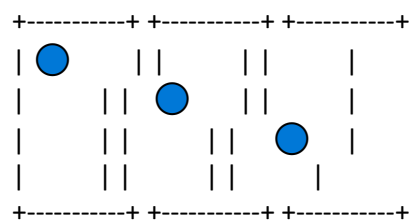
text

Copy

Download

Layar (600x600):

Frame 0: Frame 50: Frame 100:



Karakteristik Gerakan

1. **Gerakan Diagonal:** 45° kanan-bawah (karena $v_x=v_y=2$)
2. **Kecepatan Konstan:** Tidak ada percepatan
3. **Frame Rate Tinggi:** 50 FPS (20ms per frame)
4. **Presisi Vektor:** Koordinat menggunakan nilai float (meski di canvas di-round ke integer)

Perhitungan Kecepatan & Durasi

- **Kecepatan Pixel:**

2 px/frame × 50 frame/detik = 100 px/detik (diagonal)

Komponen horizontal/vertikal: $100/\sqrt{2} \approx 70.71$ px/detik

- **Waktu sampai tepi:**

$\text{Min}(600-100, 600-100)/2 = 250$ frame (5 detik)

Implementasi Class Ball

def update(self):

 self.position.add(self.velocity) # Operasi vektor

 r = self.diameter / 2

 self.canvas.coords(

 self.id,

 self.position.x - r, # x0

 self.position.y - r, # y0

 self.position.x + r, # x1

 self.position.y + r # y1

)

Perbedaan dengan Script Sebelumnya

1. **Gerakan Diagonal** (bukan horizontal/vertikal saja)

2. **Frame Rate Lebih Tinggi** (20ms vs 50ms)
3. **Ukuran Canvas Lebih Besar** (600x600 vs 400x150)
4. **Implementasi Vector External** (import dari modul terpisah)

 main.py (Tahap 4 – bola bergerak)

```
import tkinter as tk
from vector import Vector

# Definisi, Fungsi, Class, dll -----
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

# Window Utama, Canvas-----
root = tk.Tk()
root.title("Tahap 4 - Bola Bergerak")
canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

# Main Program-----
```

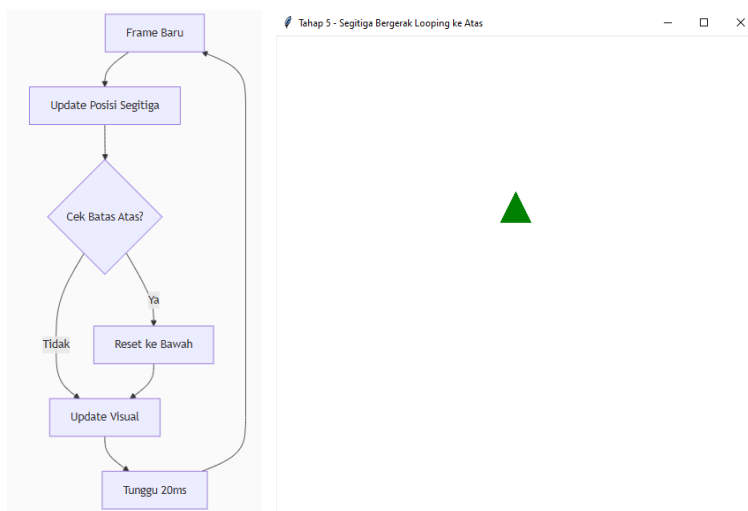
```
# Buat bola biru di (100, 100)
ball = Ball(canvas, x=100, y=100, diameter=10, color="blue")
# Mulai animasi
animate()

# Loop Program-----
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

17. sebuah segitiga



Rumus dan Perhitungan

1. Variabel Utama

- `position`: Vektor posisi segitiga (x,y)
- `velocity`: Vektor kecepatan (0,-2) → bergerak ke atas
- `size`: 20 (panjang sisi segitiga)
- Interval waktu: 20ms (50 frame/detik)

2. Koordinat Segitiga

Titik-titik segitiga sama kaki menghadap atas:

Titik 1: (x, y-size) → Puncak

Titik 2: (x-size, y+size) → Kiri bawah

Titik 3: (x+size, y+size) → Kanan bawah

3. Mekanisme Gerakan

`position.y += velocity.y` # `velocity.y = -2`

Perhitungan Posisi Y:

Frame 0: $y = \text{HEIGHT} - 50 = 550$

Frame 1: $y = 550 + (-2) = 548$

Frame 2: $y = 548 + (-2) = 546$

...

Frame N: $y = 550 - 2 * N$

Contoh Perhitungan Frame-by-Frame

Frame	Waktu (ms)	Posisi Y	Keterangan
0	0	550	Posisi awal
1	20	548	Bergerak naik 2px
2	40	546	
...	
275	5500	0	Segitiga menyentuh tepi atas
276	5520	-2	Trigger reset posisi
277	5540	598	Muncul kembali di bawah

Mekanisme Looping

if position.y < -size: *# Jika sepenuhnya keluar layar*
 position.y = HEIGHT + size *# Reset ke bawah*

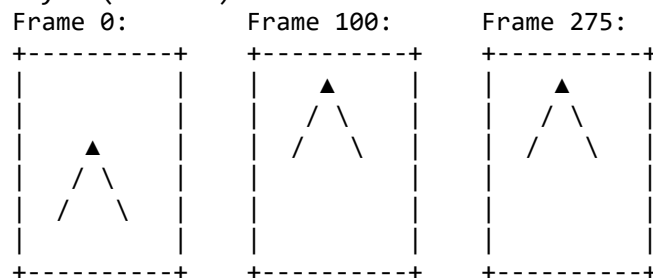
Ilustrasi:

Frame N: y = -21 (size=20)

Frame N+1: y = 600+20 = 620

Visualisasi Gerakan

Layar (600x600):



Karakteristik Gerakan

1. **Gerakan Vertikal:** Lurus ke atas dengan kecepatan konstan
 2. **Efek Infinite Scroll:** Looping saat mencapai tepi atas
 3. **Bentuk Segitiga:** Selalu menghadap ke atas
 4. **Frame Rate Tinggi:** 50 FPS (20ms per frame)
-

Perhitungan Kecepatan & Durasi

- **Kecepatan Vertikal:**

2 px/frame × 50 frame/detik = 100 px/detik ke atas

- **Waktu 1 Loop:**

$(600 + 2 \times 20) \text{px} / 100 \text{px/detik} = 6.4 \text{ detik}$

Implementasi Class Triangle

```
def update(self):
    self.position.add(self.velocity) # Gerakan vertikal

    # Cek batas atas
    if self.position.y < -self.size:
        self.position.y = HEIGHT + self.size # Looping

    # Update koordinat visual
    s = self.size
    points = [
        self.position.x, self.position.y - s, # Puncak
        self.position.x - s, self.position.y + s, # Kiri bawah
        self.position.x + s, self.position.y + s # Kanan bawah
    ]
    self.canvas.coords(self.id, *points)
```

Perbedaan dengan Script Bola

1. **Bentuk Objek:** Segitiga vs Lingkaran
 2. **Arah Gerakan:** Vertikal vs Diagonal
 3. **Mekanisme Looping:** Reset posisi vs keluar layar
 4. **Perhitungan Koordinat:** Polygon vs Oval
-

```
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()

```

Perbedaan antara **Jet berbasis Vector** dengan **Polygon biasa** :

1. Cara Menggambar

Jet Vector	Polygon Biasa
Menggunakan rumus matematika (sin/cos) untuk hitung titik	Langsung tentukan koordinat titik-titiknya
Titik otomatis menyesuaikan sudut dan ukuran	Titik statis (tidak bisa diubah mudah)
Contoh: $[x+\cos(\text{angle})*\text{size}, y+\sin(\text{angle})*\text{size}]$	Contoh: [100,70, 90,100, 110,100]

Analoginya:

- Vector = Menggambar dengan rumus (seperti pakai matematika)
- Polygon biasa = Menggambar titik-titik langsung (seperti menghubungkan dots)

2. Kelenturan Gerakan

Jet Vector	Polygon Biasa
Bisa dirotasi dengan mudah (cukup ubah angle)	Harus hitung ulang semua titik manual
Ukuran bisa diubah tanpa rumit	Harus ubah semua koordinat satu per satu

Contoh:

- Jika pesawat mau hadap kanan:

- Vector: cukup ubah angle = 0
- Polygon biasa: harus hitung ulang 3 titik secara manual

3. Contoh Kode

Jet Vector (Pintar):

```
def get_body_points(self):
    a = self.angle # Sudut pesawat
    front = Vector.fromAngle(a) * self.size # Otomatis hitung titik
    return [front.x, front.y, ...] # Dihitung pakai rumus
```

Polygon Biasa (Manual):

```
# Titik harus dihitung sendiri
canvas.create_polygon([100,70, 90,100, 110,100], fill="green")
```

4. Kelebihan Jet Vector

1. **Gampang Diputar:**
Cukup ubah nilai angle → bentuk otomatis menyesuaikan
2. **Gampang Diperbesar:**
Ubah size → semua titik menyesuaikan proporsi
3. **Siap untuk Gerakan:**
Tinggal tambah kecepatan → langsung bisa terbang!

5. Kapan Pakai Polygon Biasa?

- Untuk bentuk sederhana yang **tidak perlu diubah-ubah**
- Contoh:

```
# Segitiga statis
canvas.create_polygon([100,50, 50,100, 150,100], fill="blue")
```

Analog Mainan

- **Jet Vector** = Lego Technic (bisa disusun ulang dengan mudah)
- **Polygon Biasa** = Lego biasa (susunannya tetap)

Tips :

1. Pakai **Polygon biasa** jika cuma butuh gambar diam
2. Pakai **Vector** jika mau buat objek yang bisa bergerak/berputar

Semakin paham? 😊 Kalau mau, kita bisa coba modifikasi scriptnya bersama!

18. Pesawat (Jet) Diam

Alur Program (Cerita Singkat)

1. Kita buat layar game hitam ukuran 800x600
2. Kita gambar pesawat jet hijau di posisi (100,100) yang menghadap ke atas
3. Setiap 20 milidetik, program memperbarui tampilan
4. Pesawat tetap diam di tempat, hanya menampilkan info posisinya

Bagian-bagian Pesawat (Jet)

1. **Badan Pesawat:**
 - Bentuk segitiga dengan 3 titik:
 - **Titik Atas** (hidung pesawat)
 - **Titik Kiri Bawah** (sayap kiri)

- **Titik Kanan Bawah** (sayap kanan)
 - Ukuran: 30 pixel (tinggi dari hidung ke ekor)
- 2. **Informasi Layar:**
 - Posisi (X,Y) pesawat
 - Kecepatan (karena diam, selalu 0)

Rumus Sederhana

1. Posisi Titik-titik Pesawat:

- Titik Atas:

$$X = \text{pos.x} + (\text{size} \times \cos(\text{sudut}))$$

$$Y = \text{pos.y} + (\text{size} \times \sin(\text{sudut}))$$

- Titik Sayap:

$$\text{Lebar sayap} = \text{size} \times 0.6$$

$$\text{Sudut sayap} = \text{sudut utama} \pm 2.5 \text{ radian}$$

2. Sudut Pesawat:

- $-\pi/2$ radian = -90° (menghadap tepat ke atas)

Contoh Perhitungan

Misal pesawat di posisi (100,100):

1. Titik Atas:

$$X = 100 + (30 \times \cos(-90^\circ)) = 100 + 0 = 100$$

$$Y = 100 + (30 \times \sin(-90^\circ)) = 100 - 30 = 70$$

2. Titik Kiri Bawah:

$$X = 100 + (18 \times \cos(-87.5^\circ)) \approx 100 + 0.8 \approx 100.8$$

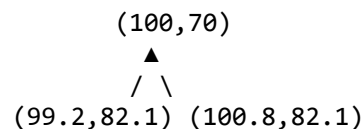
$$Y = 100 + (18 \times \sin(-87.5^\circ)) \approx 100 - 17.9 \approx 82.1$$

3. Titik Kanan Bawah:

$$X = 100 + (18 \times \cos(-92.5^\circ)) \approx 100 - 0.8 \approx 99.2$$

$$Y \approx 100 - 17.9 \approx 82.1$$

Visualisasi Pesawat



Info yang Ditampilkan

1. Posisi: (100.0, 100.0)
2. Kecepatan: 0.00 / 15 (diam/maksimum 15)

Analog Sederhana

Bayangkan seperti:

1. Kita menggambar segitiga di kertas grafik
2. Menuliskan koordinatnya di pojok kertas
3. Segitiga ini tidak bergerak, hanya diam di tempat

Untuk Dicoba

1. Ubah warna pesawat (ganti "green" jadi "red")
2. Ubah posisi awal (ganti angka 100,100)

3. Ubah ukuran pesawat (ganti angka 30)

Dengan script ini, kalian bisa belajar:

- Koordinat 2D (X,Y)
- Bentuk geometri sederhana
- Dasar-dasar pembuatan game

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

# Definisi, Fungsi, Class, dll -----
WIDTH = 800
HEIGHT = 600

class Jet:
    #definisikan semua variable yang akan digunakan
    def __init__(self, canvas, x, y, size=30, color="green"):
        self.canvas = canvas #canvas
        self.pos = Vector(x, y) #posisi awal
        self.size = size # ukuran jet
        self.color = color # warna jet
        self.angle = -math.pi / 2 # 📌 Menghadap ke atas -90° (menghadap tepat ke
atas)
        #0 → ke kanan ,  $\pi/2$  → ke bawah,  $\pi$  → ke kiri,  $-\pi/2$  → ke atas
        # membuat segitiga dengan create polygon
        self.body_id = self.canvas.create_polygon(self.get_body_points(),
fill=self.color)
        # membuat id masing2 object >> canvas coords
        self.ui_ids = []
        # jalankan draw_ui
        self.draw_ui()

    # mendapatkan 3 titik segitiga dengan rumus matematika >> front, left, right
    def get_body_points(self):
        a = self.angle
        s = self.size
        cx, cy = self.pos.x, self.pos.y
        #Titik Atas:  $X = pos.x + (size \times \cos(sudut))$ ,  $Y = pos.y + (size \times \sin(sudut))$ 
        front = Vector.fromAngle(a).multed(s)
        # Titik Sayap: Lebar sayap =  $size \times 0.6$ , Sudut sayap = sudut utama  $\pm 2.5$ 
radian
        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):
        # Jet tetap diam
        self.canvas.coords(self.body_id, *self.get_body_points())
        self.draw_ui()
```

```

def draw_ui(self):
    for item in self.ui_ids:
        self.canvas.delete(item)
    self.ui_ids.clear()

    pos_text = f"Pos: ({self.pos.x:.1f}, {self.pos.y:.1f})"
    speed_text = "Speed: 0.00 / 15"

    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)))

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

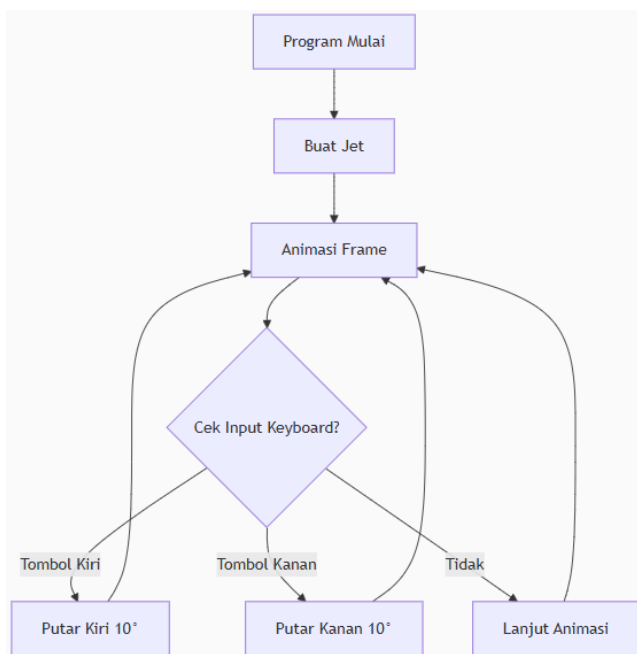
# Window Utama, Canvas-----
root = tk.Tk()
root.title("Jet Diam Menghadap Atas")
canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="black")
canvas.pack()

# Main Program-----
jet = Jet(canvas, 100, 100)
animate()

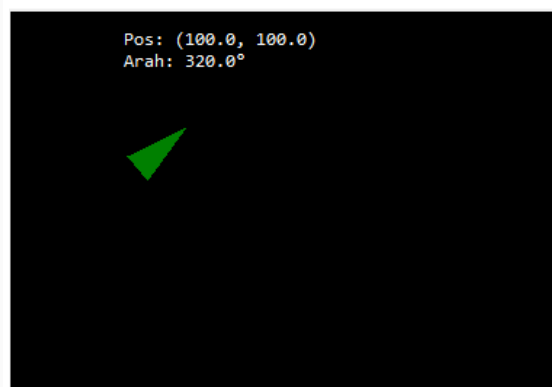
# Loop Program-----
root.mainloop()

```

19. Pesawat (Jet) Berbelok dengan panah



Jet Bisa Diputar Kiri-Kanan



Cara Kerja Pesawat Jet

1. Bentuk Pesawat

- **3 Titik Utama:**
 1. **Hidung** (depan pesawat)
 2. **Sayap Kiri**
 3. **Sayap Kanan**

Rumus Titik:

Hidung = (pos.x + cos(sudut)*ukuran, pos.y + sin(sudut)*ukuran)

Sayap = (pos.x + cos(sudut±2.5)*ukuran*0.6, pos.y + sin(sudut±2.5)*ukuran*0.6)

2. Rotasi Pesawat

- **Tombol Kiri:** Kurangi 10° dari sudut
- **Tombol Kanan:** Tambah 10° ke sudut

Contoh:

- Awal: -90° (menghadap atas)
- Tekan 1x kanan: -90° + 10° = -80°
- Tekan 2x kiri: -80° - 20° = -100°

Perhitungan Setiap Frame

Frame 0 (Awal):

- Posisi: (100,100)
- Sudut: -90° (atas)
- Titik:
 - Hidung: (100 + cos(-90°)*30, 100 + sin(-90°)*30) ≈ (100,70)
 - Sayap Kiri: (100 + cos(-87.5°)*18, 100 + sin(-87.5°)*18) ≈ (100.8,82.1)
 - Sayap Kanan: (100 + cos(-92.5°)*18, 100 + sin(-92.5°)*18) ≈ (99.2,82.1)

Frame Setelah Rotasi Kanan 1x:

- Sudut: -80°
- Titik:
 - Hidung: (100 + cos(-80°)*30, 100 + sin(-80°)*30) ≈ (105.1,70.5)
 - Sayap: (dihitung dengan rumus yang sama)

Informasi yang Ditampilkan

1. **Posisi (X,Y):**
Pos: (100.0, 100.0)
2. **Arah Pesawat:**
Arah: 270.0° (karena -90° = 270° dalam lingkaran penuh)

Cara Menggunakan Program

1. **Tombol Panah Kiri:** Putar pesawat ke kiri
2. **Tombol Panah Kanan:** Putar pesawat ke kanan
3. **Pesawat tetap di tempat,** hanya berputar

Contoh Visual Rotasi

Awal (270°): Putar Kanan 1x (280°): Putar Kiri 2x (260°):



Tips Belajar

1. `math.radians(10)` = Ubah 10° ke bentuk radian
2. `math.degrees(angle)` = Ubah radian ke derajat
3. Coba ubah angka 10 di `rotate_left/right` untuk membuat putaran lebih cepat/lambat!

Coba Sendiri!

Modifikasi kode untuk:

1. Tambahkan tombol atas/bawah untuk maju/mundur
2. Ubah warna pesawat saat berputar
3. Buat pesawat berputar otomatis

Semakin sering dicoba, semakin paham! 😊🚀

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

# Definisi, Fungsi, Class, dll -----
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.size = size
        self.color = color
        self.angle = -math.pi / 4 # Menghadap ke kiri atas
        self.body_id = self.canvas.create_polygon(self.get_body_points(),
fill=self.color)
        self.ui_ids = []
        self.draw_ui()

    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 update(self):
        # Tidak ada pergerakan sama sekali
        self.canvas.coords(self.body_id, *self.get_body_points())
        self.draw_ui()

    def draw_ui(self):
        for item in self.ui_ids:
            self.canvas.delete(item)
```

```

self.ui_ids.clear()

pos_text = f"Pos: ({self.pos.x:.1f}, {self.pos.y:.1f})"
speed_text = "Speed: 0.00 / 15"

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)))

# Fungsi animasi (tetap dijalankan, tapi jet tidak bergerak)
def animate():
    jet.update()
    root.after(20, animate)

# Window Utama, Canvas-----
root = tk.Tk()
root.title("Jet Diam di Koordinat (200, 200)")

canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="black")
canvas.pack()
# Main Program-----
# Jet muncul diam di (100, 100)
jet = Jet(canvas, 200, 200)

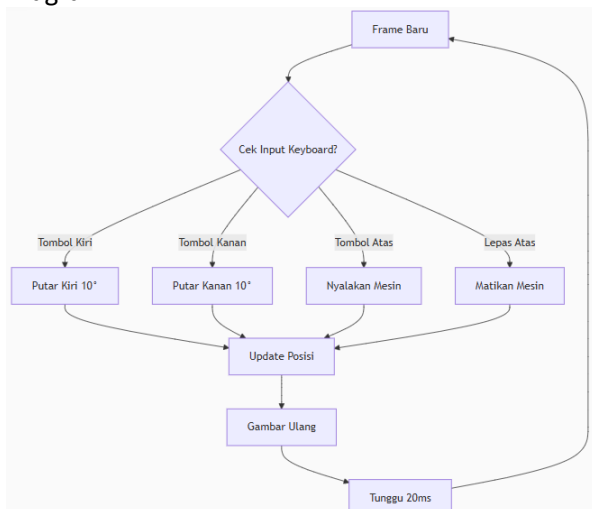
animate()
# Loop Program-----
root.mainloop()

```

17. Jet yang bisa digerakkan dan di tambah gaya maju

Diagram Alur Program



Diagram



Cara Kerja Jet

1. Kontrol Pesawat

-  Kiri: Putar kiri 10°

-  **Kanan:** Putar kanan 10°
-  **Atas:** Nyalakan mesin (dorong maju)
- **(Lepas Atas):** Matikan mesin

2. Fisika Sederhana

- **Mesin menyala** → Tambah kecepatan ke arah hadapan pesawat
- **Kecepatan maks** = 15 pixel/frame
- **Keluar layar** → Muncul di sisi seberang

Rumus Penting

1. Arah Hadap:

`Vector.fromAngle(angle)` # Menghitung arah dari sudut (dalam radian)

Contoh: Sudut -90° (atas) → Arah (0, -1)

2. Dorongan (Thrust):

`thrust = Vector.fromAngle(angle) * 0.2` # Dorongan kecil ke depan

3. Kecepatan:

`kecepatan_baru = kecepatan_lama + dorongan`

if `kecepatan_baru > MAX_SPEED`:

`kecepatan_baru = MAX_SPEED` # Batasi kecepatan

Contoh Perhitungan

Frame 0 (Awal):

- Posisi: (100, 100)
- Sudut: -90° (atas)
- Kecepatan: (0, 0)

Frame 1 (Tekan):

1. Hitung dorongan:
 - Arah: (0, -1) * 0.2 = (0, -0.2)
2. Update kecepatan:
 - (0,0) + (0,-0.2) = (0, -0.2)
3. Update posisi:
 - (100,100) + (0,-0.2) = (100, 99.8)

Frame 2 (Tekan lalu):

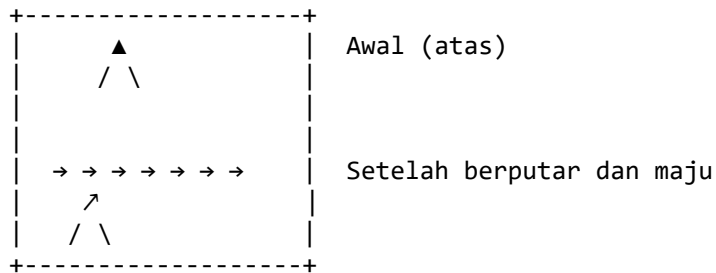
1. Putar 10° kanan → sudut baru -80°
 2. Hitung dorongan baru:
 - Arah: $(\cos(-80^\circ), \sin(-80^\circ)) \approx (0.17, -0.98)$
 - Dorongan: $(0.17, -0.98) * 0.2 \approx (0.03, -0.2)$
 3. Update kecepatan dan posisi...
-

Informasi Layar

1. **Posisi:** (x, y)
Contoh: Pos: (105.3, 98.7)
 2. **Kecepatan:** 0.00 / 15
(kecepatan saat ini / maksimum)
 3. **Arah:** 270.0°
(0°=kanan, 90°=bawah, 180°=kiri, 270°=atas)
-

Visualisasi Gerakan

Layar 800x600:





Tips Belajar

1. Ubah Nilai Ini:

`MAX_SPEED = 20` # Biar lebih cepat

`self.angle -= math.radians(15)` # Putar lebih tajam

2. Coba Sendiri:

- Tekan  +  bersamaan untuk belok sambil maju
- Lihat bagaimana kecepatan bertambah pelan

Fakta Keren:

- Pesawatmu menggunakan **matematika vektor** seperti di game roket sungguhan!
- Semua hitungan pakai **cos/sin** untuk gerakan halus.

```
import tkinter as tk
import math
#from vector import Vector # pastikan ada vector.py

import math

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

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

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

    def limit(self, max_val):
        m = self.mag()
        if m > max_val:
            return self.setted_mag(max_val)
        return self

    def setted_mag(self, new_mag):
        m = self.mag()
        if m == 0:
            return Vector(0, 0)
        return Vector(self.x * new_mag / m, self.y * new_mag / m)
```

```

def multed(self, n):
    return Vector(self.x * n, self.y * n)

    @staticmethod
    def fromAngle(angle):
        return Vector(math.cos(angle), math.sin(angle))

WIDTH = 800
HEIGHT = 600
MAX_SPEED = 15

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 = -math.pi / 2 # Menghadap ke atas
        self.body_id = self.canvas.create_polygon(self.get_body_points(),
fill=self.color)
        self.ui_ids = []
        self.thrusting = False
        self.draw_ui()

    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 update(self):
        # Tambahkan dorongan (thrust) hanya saat panah atas ditekan
        if self.thrusting:
            thrust = Vector.fromAngle(self.angle).setted_mag(0.2)
            self.acc = thrust
        else:
            self.acc = Vector(0, 0)

        # Update velocity dan batas maksimal
        self.vel.add(self.acc)
        self.vel = self.vel.limit(MAX_SPEED)

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

```



```

    # Wrap layar jika keluar
    if self.pos.y > HEIGHT + self.size:
        self.pos.y = -self.size
    elif self.pos.y < -self.size:
        self.pos.y = HEIGHT + self.size

    if self.pos.x > WIDTH + self.size:
        self.pos.x = -self.size
    elif self.pos.x < -self.size:
        self.pos.x = WIDTH + self.size

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

def draw_ui(self):
    for item in self.ui_ids:
        self.canvas.delete(item)
    self.ui_ids.clear()

    pos_text = f"Pos: ({self.pos.x:.1f}, {self.pos.y:.1f})"
    speed_text = f"Speed: {self.vel.mag():.2f} / {MAX_SPEED}"
    angle_deg = math.degrees(self.angle) % 360
    angle_text = f"Arah: {angle_deg:.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)))

    def rotate_left(self):
        self.angle -= math.radians(10)

    def rotate_right(self):
        self.angle += math.radians(10)

    def set_thrust(self, state: bool):
        self.thrusting = state

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

# Keyboard handler
def on_key_press(event):
    if event.keysym == 'Left':
        jet.rotate_left()
    elif event.keysym == 'Right':
        jet.rotate_right()
    elif event.keysym == 'Up':
        jet.set_thrust(True)

```

```

def on_key_release(event):
    if event.keysym == 'Up':
        jet.set_thrust(False)

# Setup Tkinter
root = tk.Tk()
root.title("Jet: Rotasi dan Dorong")

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

jet = Jet(canvas, 100, 100)

# Bind semua tombol
root.bind("<KeyPress>", on_key_press)
root.bind("<KeyRelease>", on_key_release)

animate()
root.mainloop()

```

17. Game >> copy paste

```

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()

```

19. vector.py

```

# vector.py
import math

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

    # ----- Penambahan -----
    def add(self, other):
        # Menambahkan nilai vektor lain ke vektor ini (ubah self)
        self.x += other.x
        self.y += other.y
        return self

```

```

def added(self, other):
    # Menghasilkan vektor baru hasil penjumlahan tanpa mengubah self
    return Vector(self.x + other.x, self.y + other.y)

# ----- Pengurangan -----
def sub(self, other):
    # Mengurangi nilai vektor lain dari vektor ini (ubah self)
    self.x -= other.x
    self.y -= other.y
    return self

def subbed(self, other):
    # Menghasilkan vektor baru hasil pengurangan tanpa mengubah self
    return Vector(self.x - other.x, self.y - other.y)

# ----- Perkalian dengan skalar -----
def mult(self, scalar):
    # Mengalikan nilai vektor dengan skalar (ubah self)
    self.x *= scalar
    self.y *= scalar
    return self

def multed(self, scalar):
    # Menghasilkan vektor baru hasil perkalian dengan skalar
    return Vector(self.x * scalar, self.y * scalar)

# ----- Pembagian dengan skalar -----
def div(self, scalar):
    # Membagi nilai vektor dengan skalar (ubah self)
    if scalar != 0:
        self.x /= scalar
        self.y /= scalar
        return self

def dived(self, scalar):
    # Menghasilkan vektor baru hasil pembagian dengan skalar
    if scalar != 0:
        return Vector(self.x / scalar, self.y / scalar)
    return Vector(0, 0)

# ----- Magnitudo -----
def mag(self):
    # Menghitung panjang (magnitudo) vektor
    return math.sqrt(self.x**2 + self.y**2)

def magnitude(self):
    return self.mag() # alias

# ----- Normalisasi -----
def normalize(self):
    # Mengubah self menjadi unit vector (arah tetap, panjang = 1)
    m = self.mag()
    if m != 0:
        self.div(m)

```

```

        else:
            self.x, self.y = 0, 0

def normalized(self):
    # Menghasilkan vektor unit baru tanpa mengubah self
    m = self.mag()
    if m != 0:
        return Vector(self.x / m, self.y / m)
    return Vector(0, 0)

# ----- Limit magnitude -----
def limit(self, max_val):
    # Batasi panjang maksimum vektor (ubah self)
    if self.mag() > max_val:
        self.normalize()
        self.mult(max_val)

def limited(self, max_val):
    # Hasilkan vektor baru dengan panjang maksimum tertentu
    v = self.copy()
    if v.mag() > max_val:
        return v.normalized().mult(max_val)
    return v

# ----- Set magnitude -----
def set_mag(self, new_mag):
    # Mengubah panjang vektor (ubah self)
    self.normalize()
    self.mult(new_mag)

# def setted_mag(self, new_mag):
#     # Menghasilkan vektor baru dengan panjang tertentu
#     return self.normalized().mult(new_mag)

def setted_mag(self, value):
    m = self.mag()
    if m != 0:
        return self.copy().div(m).mult(value)
    return Vector()

# ----- Heading (sudut arah vektor) -----
def heading(self):
    # Kembalikan sudut dalam radian terhadap sumbu x
    return math.atan2(self.y, self.x) if (self.x != 0 or self.y != 0) else 0

def heading_deg(self):
    # Sudut dalam derajat
    return math.degrees(self.heading())

# ----- Copy / clone -----
def copy(self):
    # Menghasilkan salinan dari vektor ini
    return Vector(self.x, self.y)

# ----- Set nilai -----

```



```

def set(self, x, y):
    # Menetapkan nilai x dan y
    self.x = x
    self.y = y

# ----- Rotasi 90 derajat (hanya horizontal mirror) -----
def rotate90x(self):
    # Menghasilkan vektor baru dengan x dibalik
    return Vector(self.x, -self.y)

def rotate90y(self):
    # Menghasilkan vektor baru dengan x dibalik
    return Vector(-self.x, self.y)

@staticmethod
def fromAngle(angle):
    return Vector(math.cos(angle), math.sin(angle))

@staticmethod
def from_angle(angle, length=1):
    return Vector(math.sin(angle) * length, math.cos(angle) * length)

@staticmethod
def sub_vectors(v1, v2):
    return Vector(v1.x - v2.x, v1.y - v2.y)

# ----- Ambil posisi -----
def get_position(self):
    # Mengembalikan posisi vektor sebagai tuple
    return (self.x, self.y)

# ----- Representasi -----
def __repr__(self):
    return f"Vector({self.x:.2f}, {self.y:.2f})"

def __str__(self):
    return f"({self.x:.2f}, {self.y:.2f})"

```