

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

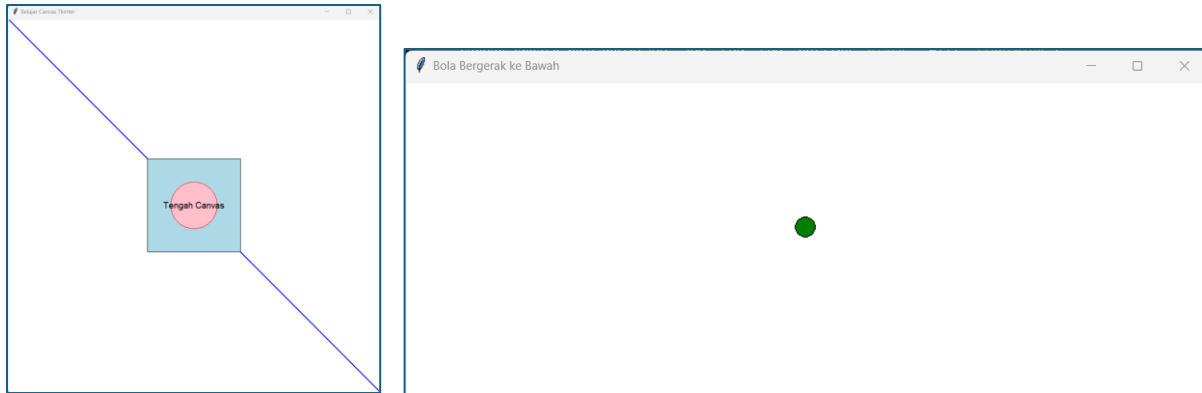
```
canvas.create_rectangle(300, 300, 500, 500, outline="black", fill="lightblue")
```

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)

...

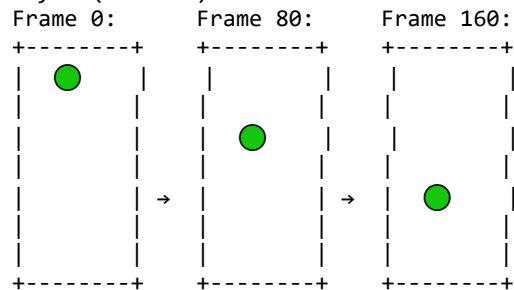
Frame N: (390, 5*N, 410, 20+5*N)

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

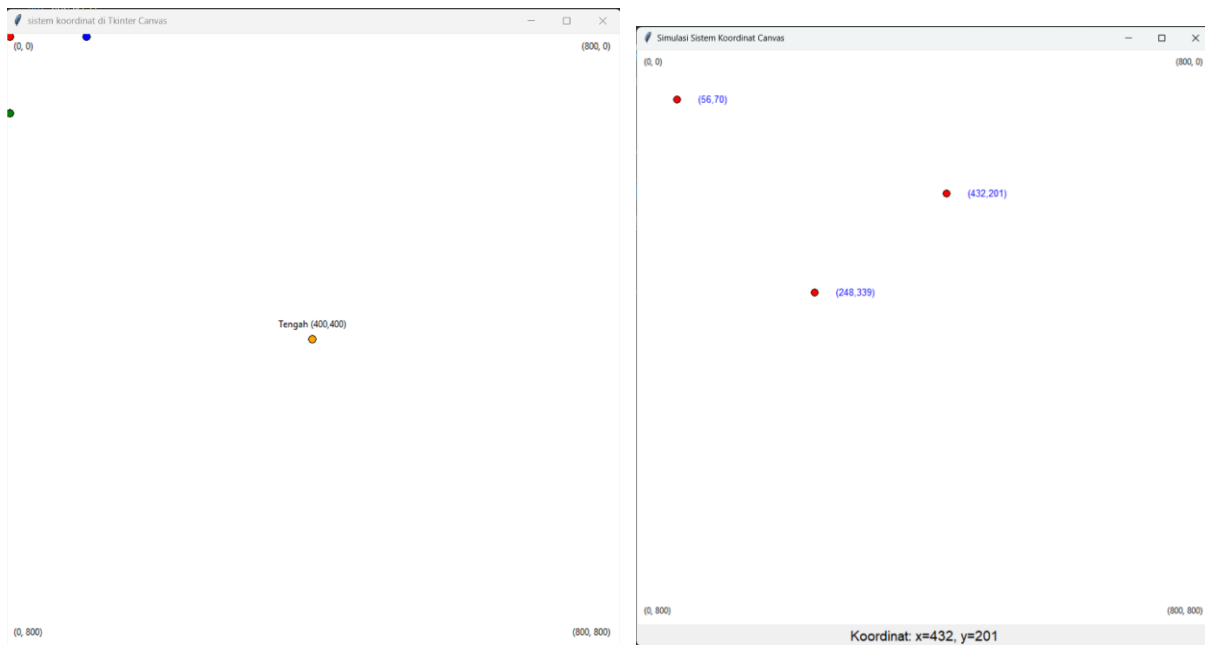
# Window Utama, Canvas-----
root = tk.Tk()
root.title("sistem koordinat di Tkinter Canvas")
canvas = tk.Canvas(root, width=800, height=800, bg="white")

# 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))

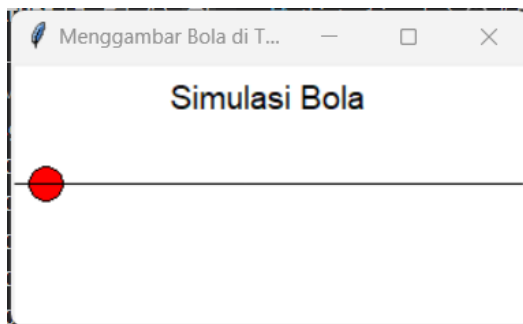
# Event binding
canvas.bind("<Button-1>", show_coordinates)

# 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

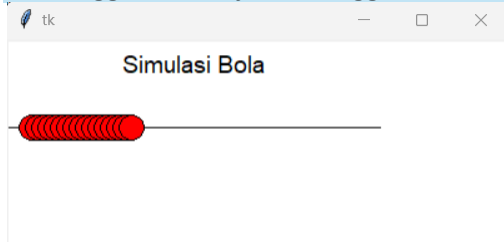
# 2. fungsi/class

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



```
# 6. 1 BOLA bergerak ke kanan >> Fungsi >> create_oval
# 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. Animasi bola bergerak ke kanan >> delete-recreate >> FUNGSI >> 1 BOLA

1. Alur Diagram Program

[Inisialisasi]



[Setup Canvas]

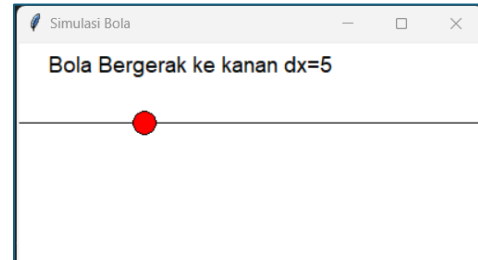
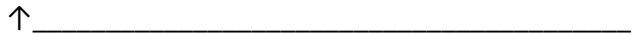


[Gambar Bola Awal]



[Loop Animasi] → [Hapus Bola Lama] → [Gambar Bola Baru] →

[Update Posisi]



2. Rumus yang Digunakan

- **Pergerakan Horizontal:** $x_{\text{new}} = x_{\text{old}} + dx$
 - dx = kecepatan horizontal (5 pixel/frame)
 - Tidak ada perubahan vertikal ($dy = 0$)
- **Posisi Bola:**
 - Koordinat oval: $(x, y, x+20, y+20)$
 - Diameter bola: 20 pixel (karena $x+20 - x = 20$)

3. Perhitungan Tiap Frame

Misal kita track 3 frame pertama:

Frame 0 (Inisialisasi):

- $x = 10$
- Bola digambar di: (10, 60, 30, 80)

Frame 1 (setelah 100ms):

1. Update posisi: $x = 10 + 5 = 15$
2. Hapus bola lama (ID bola)
3. Gambar bola baru di: (15, 60, 35, 80)

Frame 2 (setelah 200ms):

1. Update posisi: $x = 15 + 5 = 20$
2. Hapus bola lama
3. Gambar bola baru di: (20, 60, 40, 80)

Frame 3 (setelah 300ms):

1. Update posisi: $x = 20 + 5 = 25$
2. Hapus bola lama
3. Gambar bola baru di: (25, 60, 45, 80)

4. Mekanisme Animasi

- **FPS Implisit:** 100ms/frame = ~10 FPS
- **Redraw Strategy:**
 1. `canvas.delete(bola)` menghapus objek lama menggunakan ID referensi
 2. `create_oval()` membuat objek baru setiap frame
 3. `after(100, gerak)` menjadwalkan frame berikutnya

5. Batas Gerak

- Program tidak mengecek batas kanvas (WIDTH = 400), sehingga bola akan terus keluar dari layar
- Jika ingin memantul, perlu ditambahkan:

if $x \geq \text{WIDTH}$:

$dx = -dx$ # Membalik arah

6. Visualisasi Pergerakan

Frame	X-Posisi	Area Bola (x1,y1,x2,y2)
0	10	(10,60,30,80)
1	15	(15,60,35,80)
2	20	(20,60,40,80)
3	25	(25,60,45,80)
...

Program ini menunjukkan teknik animasi dasar di Tkinter dengan pendekatan **delete-recreate** setiap frame. Untuk optimasi, bisa menggunakan `canvas.move()` daripada menghapus dan membuat ulang objek.

6a. 1 BOLA bergerak ke kanan >> Fungsi >> `create_oval-delete`
import tkinter as tk

Definisi

Ukuran canvas

WIDTH,HEIGHT = 400,300

Posisi awal bola (x,y)

x,y = 10,60

Kecepatan bola (dx,dy)

dx,dy = 5,0

Fungsi gerak dengan redraw penuh

def gerak():

 global x, bola

 x += dx

 # Hapus bola lama

`canvas.delete(bola)`

 # Gambar bola baru di posisi x baru

`bola = canvas.create_oval(x, y, x + 20, y + 20, fill="red")`

 # Jadwalkan frame berikutnya

`canvas.after(100, gerak)`

Setup canvas

root = tk.Tk()

root.title("Simulasi Bola")

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

canvas.pack()

Gambar objek awal

bola = `canvas.create_oval(x, y, x + 20, y + 20, fill="red")`

black_line = `canvas.create_line(0, 70, WIDTH, 70, fill="black")`

info_text = `canvas.create_text(150, 20, text="Bola Bergerak ke kanan dx=5",`
font=("Arial", 14))

Mulai animasi

gerak()

root.mainloop()

9. Animasi bola bergerak ke kanan >>move() >> FUNGSI >> 1 BOLA

animasi bola menggunakan metode move():

1. Alur Diagram Program (Optimized Version)

[Inisialisasi Variabel]



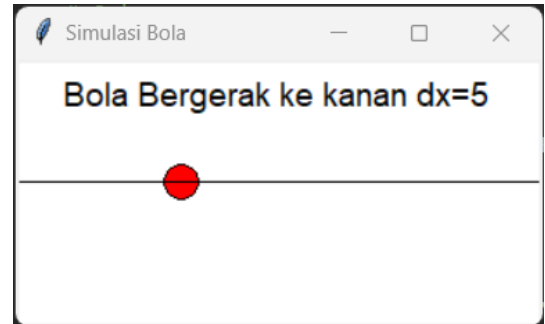
[Setup Canvas Tkinter]



[Gambar Objek Awal: Bola+Garis+Text]



[Loop Animasi] → [Pindahkan Bola] → [Update Posisi]



2. Rumus yang Digunakan

- **Fungsi move():**

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

- dx: perpindahan horizontal (5 pixel/frame)
- dy: perpindahan vertikal (0 pixel/frame)

- **Posisi Real-Time:**

$x_{\text{real}} = x_{\text{awal}} + (n_{\text{frame}} * dx)$

- n_{frame} : jumlah frame yang telah berlalu

3. Perhitungan Tiap Frame

Variabel Awal:

- Posisi awal bola: (10,60) sampai (30,80) (diameter 20px)
- Kecepatan: dx=5, dy=0
- Interval: 100ms/frame (~10 FPS)

Frame-by-Frame Movement:

Waktu (ms)	Frame	Perhitungan Posisi X	Koordinat Bola Baru (x1,y1,x2,y2)
0	0	$x = 10$	(10,60,30,80)
100	1	$x = 10 + 5 = 15$	(15,60,35,80)
200	2	$x = 15 + 5 = 20$	(20,60,40,80)
300	3	$x = 20 + 5 = 25$	(25,60,45,80)
...
$n*100$	n	$x = 10 + (n*5)$	(x,60,x+20,80)

4. Mekanisme Animasi

- **Optimized Movement:**

- Tidak ada delete() dan create_oval() setiap frame
- Menggunakan canvas.move() yang memodifikasi koordinat objek existing

- **Visual Tracking:**

Untuk debug posisi real-time:

def gerak():

`canvas.move(bola, dx, 0)`

`print(canvas.coords(bola))` # Output: [x1, y1, x2, y2]

```

        canvas.after(100, gerak)
5. Batas Gerak (Tambahan)
Jika ingin memantul di tepi kanan:
def gerak():
    global dx
    canvas.move(bola, dx, 0)

    # Cek batas kanan (WIDTH=300)
    if canvas.coords(bola)[2] >= WIDTH: #  $x_2 \geq \text{canvas width}$ 
        dx = -dx # Balik arah
    elif canvas.coords(bola)[0] <= 0: #  $x_1 \leq 0$ 
        dx = abs(dx) # Pastikan positif

```

canvas.after(100, gerak)
 6. Perbedaan dengan Versi Sebelumnya

Aspek	Versi delete-create_oval	Versi move()
Manipulasi Objek	Hapus dan buat baru	Modifikasi langsung
Performa	Lebih berat	Lebih ringan
Tracking Posisi	Manual (variabel x)	Otomatis (coords())
Memori	Bola baru setiap frame	1 objek tetap

Teknik move() lebih efisien karena:

1. Tidak ada operasi penghapusan
2. Koordinat diupdate langsung oleh Tkinter
3. Cocok untuk animasi dengan banyak objek

```

# 6b. 1 BOLA bergerak ke kanan >> Fungsi >> move

# 1. import
import tkinter as tk

# Definisi
# Ukuran canvas
WIDTH, HEIGHT = 400, 300

# Posisi awal bola (x,y)
x,y = 10,60

# Kecepatan bola (dx,dy)
dx,dy = 5,0

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

# 3. canvas

```

```

root = tk.Tk()
root.title("Simulasi Bola")
# 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(x, y, x + 20, y + 20, fill="red")
black_line = canvas.create_line(0, 70, WIDTH, 70, fill="black")
info_text = canvas.create_text(150, 20, text="Bola Bergerak ke kanan dx=5",
font=("Arial", 14))
gerak()

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

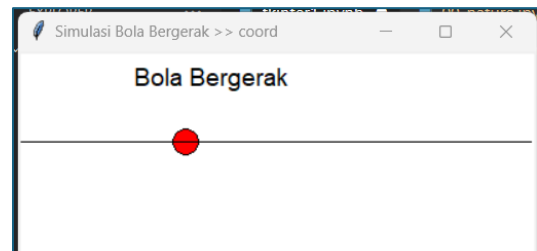
```

10. Animasi bola bergerak ke kanan >> coords()>> FUNGSI >> 1 BOLA

```

[INISIALISASI]
- [Variabel] → WIDTH, HEIGHT = 400, 300
  x, y = 10, 60
  dx, dy = 5, 0
- [Setup Canvas] → Buat window & canvas putih
- [Gambar Objek Awal] → Bola(10,60), Garis, Text
- [LOOP ANIMASI]
  - [Update Posisi] → x += dx
  - [Set Koordinat] → canvas.coords(bola, x, y, x+20, y+20)
  - [Delay 100ms] → canvas.after(100, gerak)
    ↑

```



Rumus Utama

1. Update Posisi Horizontal:

$x_{\text{new}} = x_{\text{old}} + dx$ # $dx = 5 \text{ pixel/frame}$

2. Koordinat Bola:

$\text{canvas.coords}(\text{bola}, x, y, x+20, y+20)$

- (x,y) = Pojok kiri-atas
- (x+20,y+20) = Pojok kanan-bawah (diameter 20px)

Perhitungan Frame-by-Frame

Parameter Awal:

- Posisi awal: (10,60) to (30,80)
- Kecepatan: 5px/frame
- Interval: 100ms/frame (~10 FPS)

Frame	Waktu (ms)	Perhitungan x	Koordinat Bola (x1,y1,x2,y2)
0	0	$x = 10$	(10, 60, 30, 80)
1	100	$10 + 5 = 15$	(15, 60, 35, 80)
2	200	$15 + 5 = 20$	(20, 60, 40, 80)
3	300	$20 + 5 = 25$	(25, 60, 45, 80)
...
n	$n \times 100$	$10 + (n \times 5)$	(x, 60, x+20, 80)

Analisis Metode coords()

1. Cara Kerja:

- Memodifikasi koordinat objek yang sudah ada
- Tidak perlu hapus/gambar ulang seperti pada metode delete-create_oval
- Tidak bergantung pada fungsi move()

2. Keunggulan:

- Presisi tinggi (kontrol manual koordinat)
- Cocok untuk animasi kompleks yang perlu kalkulasi posisi eksak
- Lebih efisien daripada delete-create tapi kurang efisien dibanding move()

3. Contoh Debugging:

```
def gerak():
    global x
    x += dx
    canvas.coords(bola, x, y, x+20, y+20)
    print(f"Posisi: {canvas.coords(bola)}") # Output: [x1, y1, x2, y2]
    canvas.after(100, gerak)
```

Lain

Fitur	delete-create_oval	move()	coords()
Kontrol Posisi	Manual (variabel x)	Otomatis	Manual (presisi)
Performa	Paling lambat	Tercepat	Menengah
Fleksibilitas	Tinggi	Rendah	Sangat Tinggi
Best Case	Animasi sederhana	Banyak objek	Animasi kompleks

Visualisasi Pergerakan

Frame 0: [(10,60)-(30,80)]

Frame 1: [(15,60)-(35,80)] → ← Bergerak 5px →

Frame 2: [(20,60)-(40,80)] → ← Bergerak 5px →

...

Metode ini ideal untuk kasus yang membutuhkan:

- Perubahan ukuran objek dinamis
- Rotasi atau transformasi kompleks
- Animasi dengan persamaan fisika (e.g., parabola, percepatan)

```
# 6c. 1 BOLA bergerak ke kanan >> Fungsi >> coords
# 1. import
import tkinter as tk

# Definisi
# Ukuran canvas
WIDTH, HEIGHT = 400, 300

# Posisi awal bola (x,y)
x,y = 10,60

# Kecepatan bola (dx,dy)
dx,dy = 5,0

# 3. fungsi gerak (tanpa .move)
def gerak():
    global x
    x += dx # Tambah posisi x
    canvas.coords(bola, x, y, x + 20, y + 20) # Update posisi bola
    canvas.after(100, gerak) # Ulangi setiap 100ms

# 4. canvas setup
root = tk.Tk()
root.title("Simulasi Bola Bergerak >> coord")

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

# 5. gambar objek
#gambar bola pertama kali
bola = canvas.create_oval(x, y, x + 20, y + 20, fill="red")
black_line = canvas.create_line(0, 70, WIDTH, 70, fill="black")
info_text = canvas.create_text(150, 20, text="Bola Bergerak", font=("Arial", 14))

# 6. mulai gerak (update posisi)
gerak()

# 7. loop utama
root.mainloop()
```

11. Menggambar object >> 2 BOLA

- Lihat contoh 1 bola kemudian modifikasi jadi 2 bola :

Judul : Menggambar 1 Bola Bola posisi (10,60) diameter 20, red Garis dari (0,70) sampai (400,70), hitam Info text (150,20) Simulasi Bola	root.title("Menggambar 1 Bola") bola = canvas.create_oval(10, 60, 30, 80, fill="red") garis = canvas.create_line(0, 70, 400, 70, fill="black") canvas.create_text(150, 20, text="Simulasi Bola", font=("Arial", 14))
---	--

5. Menggambar object >> 1 BOLA

1. import

```
import tkinter as tk
```

2. fungsi/class

3. canvas

```
root = tk.Tk()  
root.title("Menggambar 1 Bola")
```

Membuat Canvas

```
canvas = tk.Canvas(root, width=400, height=300, bg="white")  
canvas.pack()
```

4. main program

Menggambar canvas >> lihat sintaks

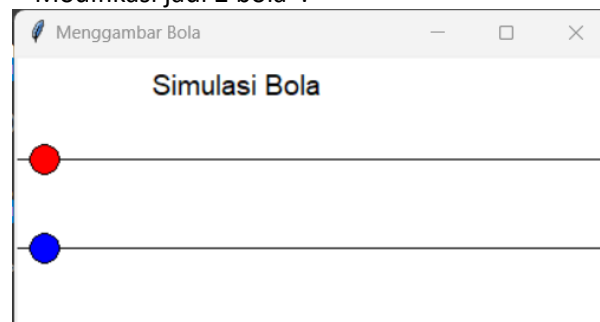
```
garis = canvas.create_line(0, 70, 400, 70, fill="black")  
bola = canvas.create_oval(10, 60, 30, 80, fill="red")  
info_text = canvas.create_text(150, 20, text="Simulasi Bola", font=("Arial", 14))
```

5. loop

Mengulangi frame / frame

```
root.mainloop()
```

- Modifikasi jadi 2 bola :



Judul : Menggambar 2 Bola Bola1 posisi (10,60) diameter 20, red Garis1 dari (0,70) sampai (400,70), hitam Bola2 posisi (10,120) diameter 20, blue	root.title("Menggambar 2 Bola") bola1 = canvas.create_oval(x1, y1, x1+20, y1+20, fill="red") garis1 = canvas.create_line(0, y1+10, WIDTH, y1+10, fill="black") bola2 = canvas.create_oval(x2, y2, x2+20, y2+20, fill="blue")
--	---

Garis2 dari (0,70) sampai (400,70), hitam Info text (150,20) Simulasi Bola WIDTH,HEIGHT = 400,300 # canvas	garis2 = canvas.create_line(0, y2+10, WIDTH, y2+10, fill="black") canvas.create_text(150, 20, text="Simulasi Bola", font=("Arial", 14))
--	---

7a. Menggambar object >> 2 BOLA

```
# 1. import
import tkinter as tk

# 2. fungsi/class
WIDTH,HEIGHT = 400,300 # Ukuran canvas
x1,y1 = 10,60 # bola1
x2,y2 = 10,120 # bola2

# 3. canvas
root = tk.Tk()
root.title("Menggambar Bola")

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

# 4. main program
# Menggambar canvas >> lihat sintaks
info_text = canvas.create_text(150, 20, text="Simulasi Bola", font=("Arial", 14))

garis1 = canvas.create_line(0, y1+10, WIDTH, y1+10, fill="black")
bola1 = canvas.create_oval(x1, y1, x1+20, y1+20, fill="red")

garis2 = canvas.create_line(0, y2+10, WIDTH, y2+10, fill="black")
bola2 = canvas.create_oval(x2, y2, x2+20, y2+20, fill="blue")

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

12. Menggambar object >> 2 BOLA dengan CLASS



Langkah Pembuatan Class Bola

1. Inisialisasi Class:

```
class Bola:
    def __init__(self, canvas, x, y, warna):
        self.canvas = canvas
        self.x = x
        self.y = y
        self.warna = warna
```

2. Gambar Komponen Visual:

o Garis Referensi:

```
self.canvas.create_line(0, y+10, WIDTH, y+10, fill="black")
```

o Bola (Oval):

```
self.shape = canvas.create_oval(x, y, x+20, y+20, fill=warna)
```

- **Teks Posisi:**
self.text_pos = canvas.create_text(x, y+25, text="", font=('Arial', 10))

3. Class Aplikasi Utama:

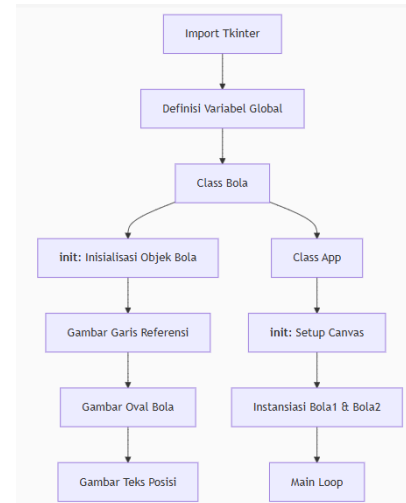
```
class App:
    def __init__(self, root):
        self.canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT,
bg="white")
        self.bola1 = Bola(self.canvas, x1, y1, COLOR1)
        self.bola2 = Bola(self.canvas, x2, y2, COLOR2)
```

📊 Perhitungan dan Mekanisme Class

- **Koordinat Bola:**
 - Setiap bola menyimpan properti x, y, dan shape secara independen.
 - Contoh: bola1 di (10,60) dan bola2 di (10,120).

✅ Kelebihan Menggunakan Class vs Fungsi Biasa

Aspek	Class-Based	Fungsi Biasa
Organisasi Kode	Modular (terenkapsulasi)	Linear (rentan spaghetti code)
Reusabilitas	Bisa dipakai ulang untuk banyak bola	Harus menulis ulang logika
Manajemen State	Properti disimpan dalam objek	Variabel global kompleks
Ekstensibilitas	Mudah ditambah metode/fitur baru	Sulit di-maintain
Contoh Use Case	Animasi 10+ bola dengan interaksi	Animasi sederhana 1-2 objek



7b. Menggambar object >> 2 BOLA >> CLASS

1. Import modul
import tkinter as tk

Definisi
Ukuran canvas
WIDTH, HEIGHT = 400, 300

Posisi awal bola (x,y)
x1,y1,COLOR1 = 10,60,"red" # bola1
x2,y2,COLOR2 = 10,120,"blue" # bola2

2. Kelas Bola
class Bola:
 def __init__(self, canvas, x, y, warna):


```

self.canvas = canvas
self.warna = warna

# Buat garis referensi
self.canvas.create_line(0, y+10, WIDTH, y+10, fill="black")
# Buat bola
self.shape = canvas.create_oval(x, y, x+20, y+20, fill=warna)
# Buat teks posisi di bawah bola
self.text_pos = canvas.create_text(x, y + 25, anchor='nw', text="", font=('Arial', 10), fill='black')
# Buat teks status sekali saja
self.status_text = canvas.create_text(150, 40, text="Bola Diam", font=("Arial", 14))

```

3. Kelas Aplikasi Utama

class App:

```

def __init__(self, root):
    self.root = root
    self.root.title("Simulasi Bola")
    self.canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
    self.canvas.pack()

```

Buat bola

```

self.bola1 = Bola(self.canvas, x1, y1, COLOR1)
self.bola2 = Bola(self.canvas, x2, y2, COLOR2)

```

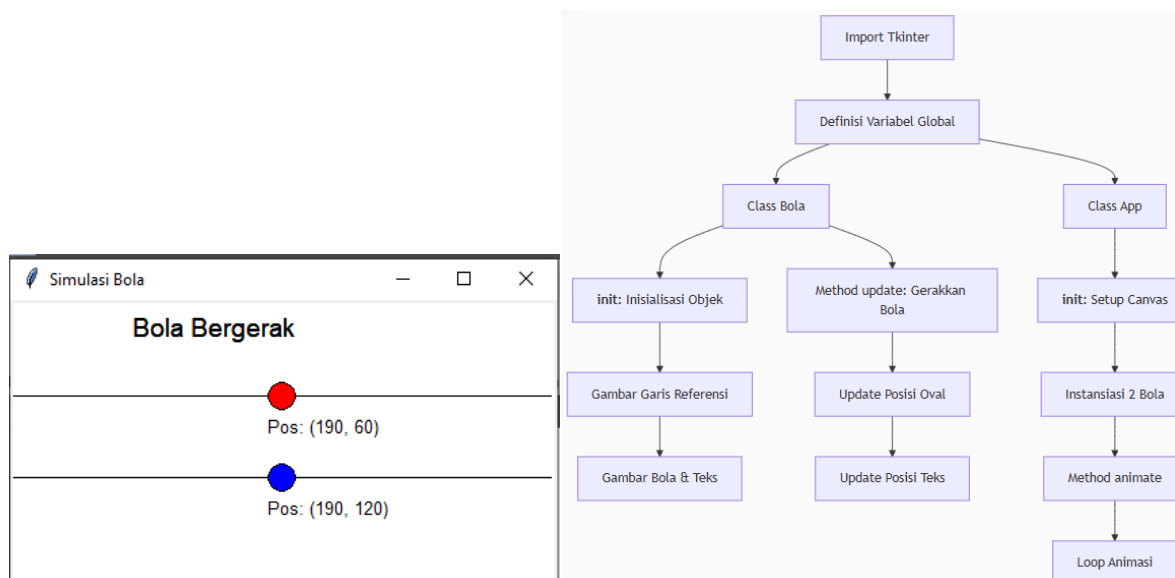
4. Main Program

```

if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()

```

12. Animasi object >> menggerakkan 2 BOLA dengan CLASS



Rumus Utama

1. Pergerakan Horizontal:

$x1_baru = x1_lama + dx$
 $x2_baru = x2_lama + dx$

- $dx = 5$ (kecepatan konstan)

2. Koordinat Bola:

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

- $(x1, y1)$ = Pojok kiri-atas
- $(x2, y2)$ = Pojok kanan-bawah (diameter 20px)

3. Posisi Teks:

`canvas.coords(text_pos, x1, y2+5)`

Perhitungan Tiap Frame (2 Bola)

Parameter Awal:

- Bola Merah: (10,60) to (30,80)
- Bola Biru: (10,120) to (30,140)
- Kecepatan: 5px/frame
- Interval: 100ms/frame (~10 FPS)

Frame 0 (Inisialisasi):

Bola	Posisi Oval	Posisi Teks
Merah	(10,60,30,80)	(10,85)
Biru	(10,120,30,140)	(10,145)

Frame 1 (100ms):

Bola	Perhitungan	Posisi Baru	Teks Posisi
Merah	$10+5=15$, $30+5=35$	(15,60,35,80)	"Pos: (15,60)"
Biru	$10+5=15$, $30+5=35$	(15,120,35,140)	"Pos: (15,120)"

Frame 2 (200ms):

Bola	Perhitungan	Posisi Baru
Merah	$15+5=20$, $35+5=40$	(20,60,40,80)
Biru	$15+5=20$, $35+5=40$	(20,120,40,140)

Frame n ($n \times 100$ ms):

Bola	Rumus Umum
Merah	$x = 10 + n \times 5$
Biru	$x = 10 + n \times 5$

Mekanisme Class

1. Inisialisasi Objek:

```
class Bola:
```

```

def __init__(self, canvas, x, y, warna):
    self.shape = canvas.create_oval(...)
    self.text_pos = canvas.create_text(...)
2. Update Posisi:
def update(self):
    pos = self.canvas.coords(self.shape) # Ambil posisi saat ini
    x1, y1, x2, y2 = pos
    x1 += self.dx
    x2 += self.dx
    self.canvas.coords(self.shape, x1, y1, x2, y2) # Update oval
    self.canvas.coords(self.text_pos, x1, y2+5) # Update teks
3. Loop Animasi:
def animate(self):
    self.bola1.update()
    self.bola2.update()
    self.canvas.after(100, self.animate) # Rekursif

```

⚡ Keunggulan Implementasi Ini

1. **Dual Animation:**
 - Kedua bola bergerak independen dengan mekanisme sama
 - Mudah ditambah bola ke-3, ke-4, dst.
2. **Real-Time Tracking:**
 - Teks posisi update otomatis setiap frame
 - Debugging mudah dengan print(pos)

Frame 0:

Merah [(10,60)-(30,80)] Biru [(10,120)-(30,140)]

Frame 1:

Merah [(15,60)-(35,80)] Biru [(15,120)-(35,140)]

Frame 2:

Merah [(20,60)-(40,80)] Biru [(20,120)-(40,140)]

...

Dengan pendekatan class, kode menjadi lebih terstruktur dan siap untuk pengembangan lebih kompleks!



```

# 8a. Simulasi Gerak ke kanan >> 2 BOLA >> CLASS >> canvas.coords
# 1. Import modul
import tkinter as tk

# Definisi
WIDTH, HEIGHT = 400, 300
x1, y1, COLOR1 = 10, 60, "red" # bola1
x2, y2, COLOR2 = 10, 120, "blue" # bola2
# Kecepatan bola (dx,dy)
dx,dy = 5,0

# 2. Kelas Bola
class Bola:
    def __init__(self, canvas, x, y, warna):
        self.canvas = canvas
        self.x = x
        self.y = y
        self.warna = warna

```

```

        self.dx = dx # kecepatan ke kanan

        # Buat garis referensi
        self.canvas.create_line(0, y + 10, WIDTH, y + 10, fill="black")

        # Buat bola dan teks
        self.shape = canvas.create_oval(x, y, x + 20, y + 20, fill=warna)
        self.text_pos = canvas.create_text(x, y + 25, anchor='nw', text="",
font=('Arial', 10), fill='black')
        self.status_text = canvas.create_text(150, 20, text="Bola Bergerak",
font=("Arial", 14))

    def update(self):
        # Ambil posisi bola
        pos = self.canvas.coords(self.shape) # [x1, y1, x2, y2]
        x1, y1, x2, y2 = pos

        # Tambah posisi horizontal
        x1 += self.dx
        x2 += self.dx

        # Update posisi bola
        self.canvas.coords(self.shape, x1, y1, x2, y2)

        # Update posisi teks
        self.canvas.coords(self.text_pos, x1, y2 + 5)

        # Update isi teks posisi
        self.canvas.itemconfig(self.text_pos, text=f"Pos: ({int(x1)}, {int(y1)})")

# 3. Kelas Aplikasi Utama
class App:
    def __init__(self, root):
        self.root = root
        self.root.title("Simulasi Bola")
        self.canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
        self.canvas.pack()

        # Buat bola
        self.bola1 = Bola(self.canvas, x1, y1, COLOR1)
        self.bola2 = Bola(self.canvas, x2, y2, COLOR2)

        # Jalankan animasi
        self.animate()

    def animate(self):
        self.bola1.update()
        self.bola2.update()
        self.canvas.after(100, self.animate)

# 4. Main Program
if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()

```

13. vektor dan bagaimana cara melakukan operasi matematika dasar pada vektor.

◆ Apa itu Vektor?

Bayangkan kamu ingin menunjukkan arah dan jarak ke suatu tempat dari rumahmu. Misalnya, kamu bilang:

"Aku pergi 3 langkah ke kanan dan 4 langkah ke atas."

Itu adalah **vektor**! Dalam komputer, kita bisa menuliskannya sebagai:

$a = \text{Vector}(3, 4)$

Artinya: vektor **a** punya arah **kanan 3** dan **atas 4**.

🔧 Fungsi-fungsi Dasar

1. Penjumlahan Vektor: `add()`

`a.add(b)`

Artinya kita menambahkan vektor **b** ke **a**.

Misalnya:

- $a = (3, 4)$
- $b = (1, 2)$

Maka:

- $3 + 1 = 4$
- $4 + 2 = 6$

Hasil:

$a = (4, 6)$

➡ Sekarang vektor **a** berubah jadi lebih panjang dan berubah arah sedikit.

2. Pengurangan Vektor: `sub()`

`a.sub(b)`

Artinya kita mengurangi vektor **b** dari **a**.

- $a = (4, 6)$
- $b = (1, 2)$

Maka:

- $4 - 1 = 3$
- $6 - 2 = 4$

Hasil:

$a = (3, 4)$

➡ Ini sama seperti kamu mundur kembali sejauh vektor **b**.

3. Perkalian Skalar: `mult(c)`

`a.mult(3)`

Kita mengalikan **a** dengan angka **3**. Artinya:

- $3 \times 3 = 9$
- $4 \times 3 = 12$

Hasil:

$a = (9, 12)$

➡ Ini membuat vektor jadi **3 kali lebih panjang**. Bayangkan kamu berjalan 3 kali lebih jauh dengan arah yang sama.

4. Pembagian Skalar: div(c)

a.div(3)

Kita membagi vektor **a** dengan angka **3**:

- $9 \div 3 = 3$
- $12 \div 3 = 4$

Hasil:

a = (3, 4)

➡ Ini membuat vektor **jadi lebih pendek**, tapi arah tetap sama.

🔙 Kesimpulan

Vektor bisa **ditambah, dikurang, diperbesar, dan dikecilkan**. Dalam kode, kita pakai fungsi:

- add() untuk penjumlahan,
- sub() untuk pengurangan,
- mult() untuk perkalian,
- div() untuk pembagian.

Semua perubahan akan **mengubah isi vektor tersebut**.

```
from vector import Vector

# 1. add(), sub(), mult(), div() – Operasi Dasar Vektor
a = Vector(3, 4)
b = Vector(1, 2)
c = 3

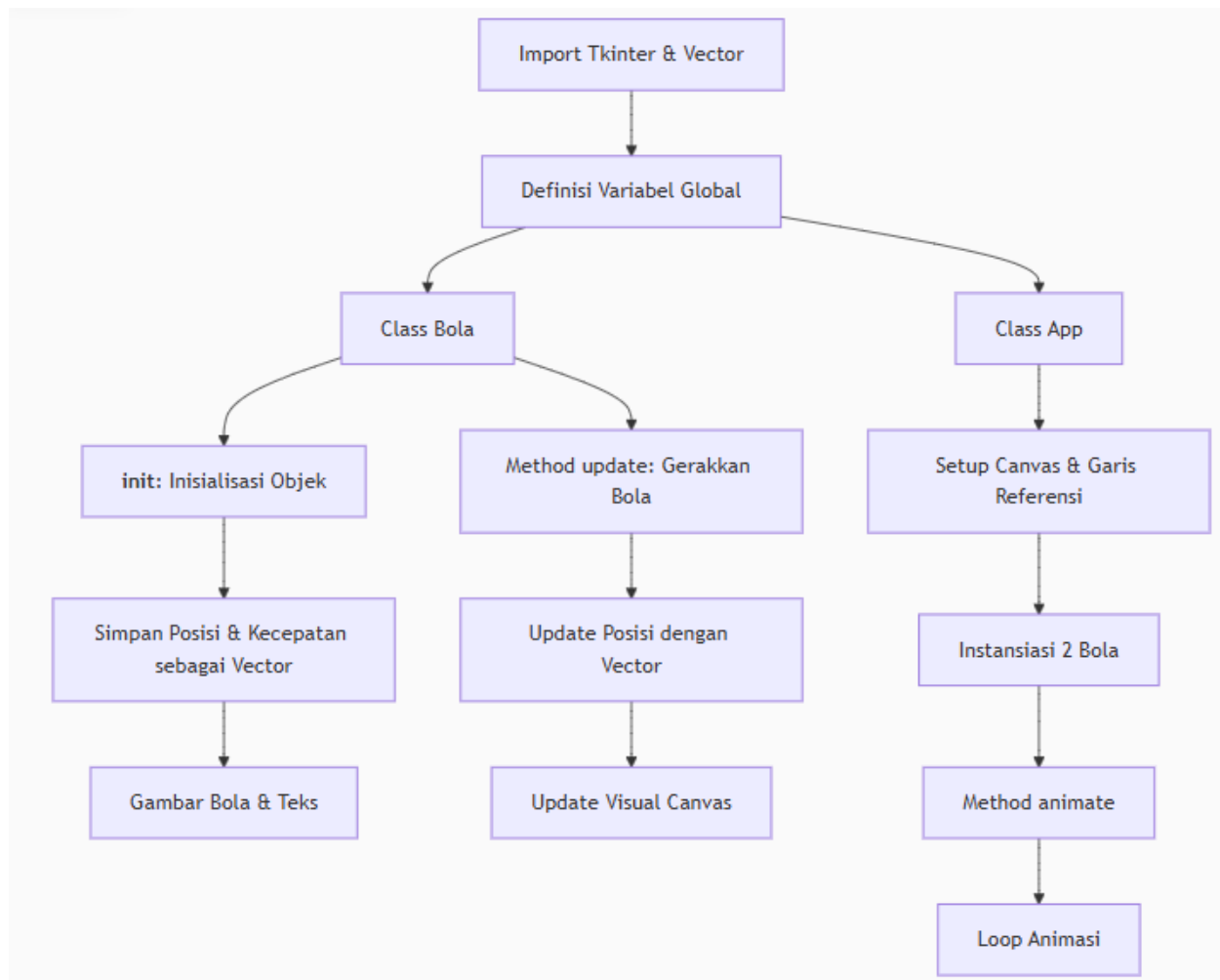
# Penjumlahan vektor a dan b hasilnya disimpan di a
print("# Operasi Dasar")
a.add(b) # a = (3+1, 4+2) → (4, 6)
print("(3, 4) + (1, 2) : ", a) #(4.00, 6.00)

# Pengurangan vektor a dan b hasilnya disimpan di a
a.sub(b) # a = (4-1, 6-2) → (3, 4)
print("(4, 6) - (1, 2) : ", a) #(3.00, 4.00)

# Perkalian vektor a dengan skalar c hasilnya disimpan di a
a.mult(c) # a = (3*3, 4*3) → (9, 12)
print("(3, 4) * 3 : ", a) #(9.00, 12.00)

# Pembagian vektor a dengan skalar c hasilnya disimpan di a
a.div(c) # a = (9/3, 12/3) → (3, 4)
print("(9, 12) / 3 : ", a) #(3.00, 4.00)
```

14. Animasi 2 bola menggunakan class dengan pendekatan **Vector**:



Rumus Utama (Vector-Based)

1. **Operasi Vector:**

`pos.add(vel) # pos = pos + vel`

- Setara dengan:
`pos.x += vel.x`
`pos.y += vel.y`

2. **Posisi Bola:**

`canvas.coords(shape, pos.x, pos.y, pos.x+20, pos.y+20)`
 ○ Diameter tetap 20px

3. **Kecepatan Awal:**

`vel = Vector(dx, dy) # (5, 0)`

📊 Perhitungan Tiap Frame

Parameter Awal:

- Bola Merah: `pos = Vector(10,60)`, `vel = Vector(5,0)`
- Bola Biru: `pos = Vector(10,120)`, `vel = Vector(5,0)`
- Interval: 100ms/frame (~10 FPS)

Frame 0 (Inisialisasi):

Bola	Posisi (Vector)	Visual Oval	Teks Posisi
Merah	(10,60)	(10,60,30,80)	"Pos: (10,60)"

Biru	(10,120)	(10,120,30,140)	"Pos: (10,120)"
------	----------	-----------------	-----------------

Frame 1 (100ms):

Bola	Operasi Vector	Posisi Baru	Visual Oval	Teks Posisi
Merah	(10,60)+(5,0)	(15,60)	(15,60,35,80)	"Pos: (15,60)"
Biru	(10,120)+(5,0)	(15,120)	(15,120,35,140)	"Pos: (15,120)"

Frame 2 (200ms):

Bola	Operasi Vector	Posisi Baru	Visual Oval
Merah	(15,60)+(5,0)	(20,60)	(20,60,40,80)
Biru	(15,120)+(5,0)	(20,120)	(20,120,40,140)

Frame n (n×100ms):

Bola	Rumus Umum
Merah	pos = (10 + n×5, 60)
Biru	pos = (10 + n×5, 120)



Mekanisme Class dengan Vector

- Inisialisasi Vector:**

```
self.pos = Vector(x, y)      # Posisi awal
self.vel = Vector(dx, dy)    # Kecepatan
```
- Update dengan Vector:**

```
def update(self):
    self.pos.add(self.vel)    # Operasi vector
    self.canvas.coords(self.shape,
                        self.pos.x, self.pos.y,
                        self.pos.x+20, self.pos.y+20)
```
- Visual Tracking:**

```
# Update teks posisi
self.canvas.itemconfig(self.text_pos,
                        text=f"Pos: ({int(self.pos.x)}, {int(self.pos.y)})")
```



Keunggulan Vector-Based Approach

- Fleksibilitas Gerak:**
 - Mudah modifikasi kecepatan:

```
self.vel = Vector(3, 2) # Gerak diagonal
```
- Fisika Kompleks:**
 - Tambah percepatan:

```
self.acc = Vector(0, 0.1) # Gravitasi
def update(self):
```



```

self.vel.add(self.acc)
self.pos.add(self.vel)

```

3. Operasi Matematis:

- Rotasi, normalisasi, dll. bisa diimplementasikan di class Vector

Visualisasi Pergerakan

Frame 0:

Merah [(10,60)-(30,80)] Biru [(10,120)-(30,140)]


Frame 1:

Merah [(15,60)-(35,80)] Biru [(15,120)-(35,140)]

Frame 2:

Merah [(20,60)-(40,80)] Biru [(20,120)-(40,140)]

...

Implementasi dengan **Vector** memungkinkan pengembangan simulasi fisika yang lebih realistis dengan clean code! 

```

# 9. Simulasi Gerak ke kanan >> 2 BOLA >> CLASS >> canvas.coords >> Vector
import tkinter as tk
from vector import Vector # pastikan vector.py ada di folder yang sama

# Definisi
WIDTH, HEIGHT = 400, 300
x1, y1, COLOR1 = 10, 60, "red" # bola1
x2, y2, COLOR2 = 10, 120, "blue" # bola2
# Kecepatan bola (dx,dy)
dx,dy = 5,0

# Kelas Bola
class Bola:
    def __init__(self, canvas, x, y, warna):
        self.canvas = canvas
        self.pos = Vector(x, y)
        self.vel = Vector(dx, dy) # kecepatan ke kanan
        self.warna = warna

        # Gambar awal
        self.shape = self.canvas.create_oval(
            self.pos.x, self.pos.y, self.pos.x + 20, self.pos.y + 20, fill=self.warna
        )
        self.text_pos = self.canvas.create_text(
            self.pos.x, self.pos.y + 25, anchor='nw', text="", font=('Arial', 10),
            fill='black'
        )

    def update(self):
        # Tambah kecepatan ke posisi
        self.pos.add(self.vel)

        # Update bola
        self.canvas.coords(self.shape,
                           self.pos.x, self.pos.y,
                           self.pos.x + 20, self.pos.y + 20)

```

```

        # Update teks posisi
        self.canvas.coords(self.text_pos, self.pos.x, self.pos.y + 25)
        self.canvas.itemconfig(self.text_pos,
                                text=f"Pos: ({int(self.pos.x)}, {int(self.pos.y)})")

# Kelas Aplikasi
class App:
    def __init__(self, root):
        self.root = root
        self.root.title("Simulasi Bola dengan Vector")
        self.canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
        self.canvas.pack()

        # Gambar garis referensi
        self.canvas.create_line(0, y1 + 10, WIDTH, y1 + 10, fill="black")
        self.canvas.create_line(0, y2 + 10, WIDTH, y2 + 10, fill="black")

        # Buat bola
        self.bola1 = Bola(self.canvas, x1, y1, COLOR1)
        self.bola2 = Bola(self.canvas, x2, y2, COLOR2)

        # Jalankan animasi
        self.animate()

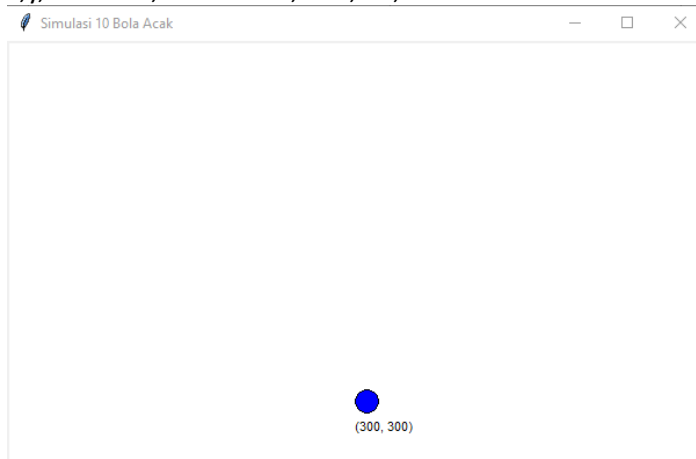
    def animate(self):
        self.bola1.update()
        self.bola2.update()
        self.canvas.after(100, self.animate)

# Program utama
if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()

```

15. Menggambar object >> 1 BOLA

x,y,diameter,color = 300, 300, 20, "blue"



```

# 10a. Menggambar object >> 1 BOLA >> random + teks posisi

import tkinter as tk

# Ukuran canvas
WIDTH, HEIGHT = 600, 600
# bola
x,y,diameter,color = 300, 300, 20, "blue"

# Kelas Bola
class Bola:
    def __init__(self, canvas, x, y, diameter, color):
        self.canvas = canvas
        self.color = color
        self.diameter = diameter
        self.x = x
        self.y = y

        # Gambar bola
        self.shape = canvas.create_oval(
            x, y, x + diameter, y + diameter, fill=color, outline="black"
        )

        # Tampilkan posisi di bawah bola
        self.text = canvas.create_text(
            x, y + diameter + 5, anchor='nw',
            text=f"({x}, {y})", font=("Arial", 8), fill="black"
        )

# Kelas Aplikasi Utama
class App:
    def __init__(self, root):
        self.root = root
        self.root.title("Simulasi 10 Bola Acak")
        self.canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
        self.canvas.pack()
        self.balls = Bola(self.canvas, x, y, diameter, color)

# Main Program
if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()

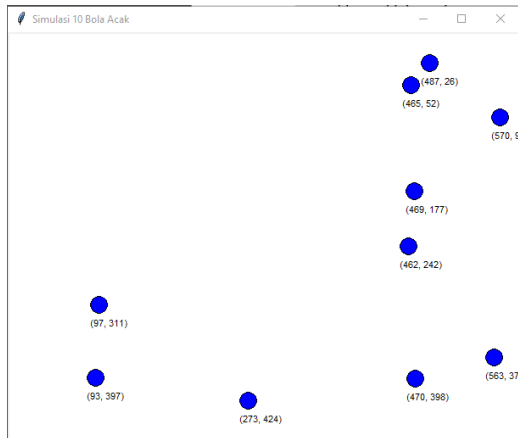
```

16. Menggambar object >> 10 BOLA >> random + teks posisi

```

diameter = 20
x = random.randint(0, WIDTH - diameter)
y = random.randint(0, HEIGHT - diameter - 20) # beri ruang untuk teks
color = "blue"

```



```
# 10b. Menggambar object >> 10 BOLA >> random + teks posisi
import tkinter as tk
import random

# Ukuran canvas
WIDTH, HEIGHT = 600, 600

# Kelas Bola
class Bola:
    def __init__(self, canvas, x, y, diameter, color):
        self.canvas = canvas
        self.color = color
        self.diameter = diameter
        self.x = x
        self.y = y

        # Gambar bola
        self.shape = canvas.create_oval(
            x, y, x + diameter, y + diameter, fill=color, outline="black"
        )

        # Tampilkan posisi di bawah bola
        self.text = canvas.create_text(
            x, y + diameter + 5, anchor='nw',
            text=f"({x}, {y})", font=("Arial", 8), fill="black"
        )

# Kelas Aplikasi Utama
class App:
    def __init__(self, root):
        self.root = root
        self.root.title("Simulasi 10 Bola Acak")
        self.canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
        self.canvas.pack()

        self.balls = []
        for _ in range(10):
            diameter = 20
            x = random.randint(0, WIDTH - diameter)
```

```

        y = random.randint(0, HEIGHT - diameter - 20) # beri ruang untuk teks
        color = "blue"
        bola = Bola(self.canvas, x, y, diameter, color)
        self.balls.append(bola)

# Main Program
if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()

```

17. Simulasi 1 Bola Bergerak

```

x, y, diameter, color = 100, 100, 10, "blue"
vel = Vector(2, 2)

```



```

import tkinter as tk
from vector import Vector

# Ukuran canvas
WIDTH, HEIGHT = 300, 300

# Bola
x, y, diameter, color = 100, 100, 10, "blue"

# Kelas Bola
class Bola:
    def __init__(self, canvas, x, y, diameter, color):
        self.canvas = canvas
        self.diameter = diameter
        self.pos = Vector(x, y)
        self.vel = Vector(2, 2)

        # Gambar bola
        self.shape = canvas.create_oval(
            self.pos.x, self.pos.y,
            self.pos.x + diameter, self.pos.y + diameter,
            fill=color, outline="black"
        )

        # Teks posisi
        self.text = canvas.create_text(
            self.pos.x, self.pos.y + diameter + 5,

```

```

        anchor='nw', text="", font=("Arial", 8), fill="black"
    )

def update(self):
    # Update posisi
    self.pos.add(self.vel)

    # Perbarui posisi bola
    self.canvas.coords(
        self.shape,
        self.pos.x, self.pos.y,
        self.pos.x + self.diameter, self.pos.y + self.diameter
    )

    # Perbarui teks posisi
    self.canvas.coords(self.text, self.pos.x, self.pos.y + self.diameter + 5)
    self.canvas.itemconfig(self.text, text=f"({int(self.pos.x)},
{int(self.pos.y)})")

# Kelas Aplikasi
class App:
    def __init__(self, root):
        self.root = root
        self.root.title("Simulasi Bola Bergerak")
        self.canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
        self.canvas.pack()

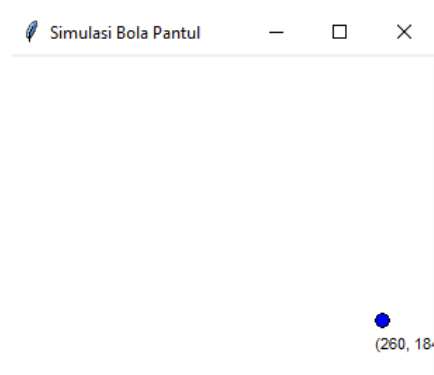
        self.bola = Bola(self.canvas, x, y, diameter, color)
        self.animate()

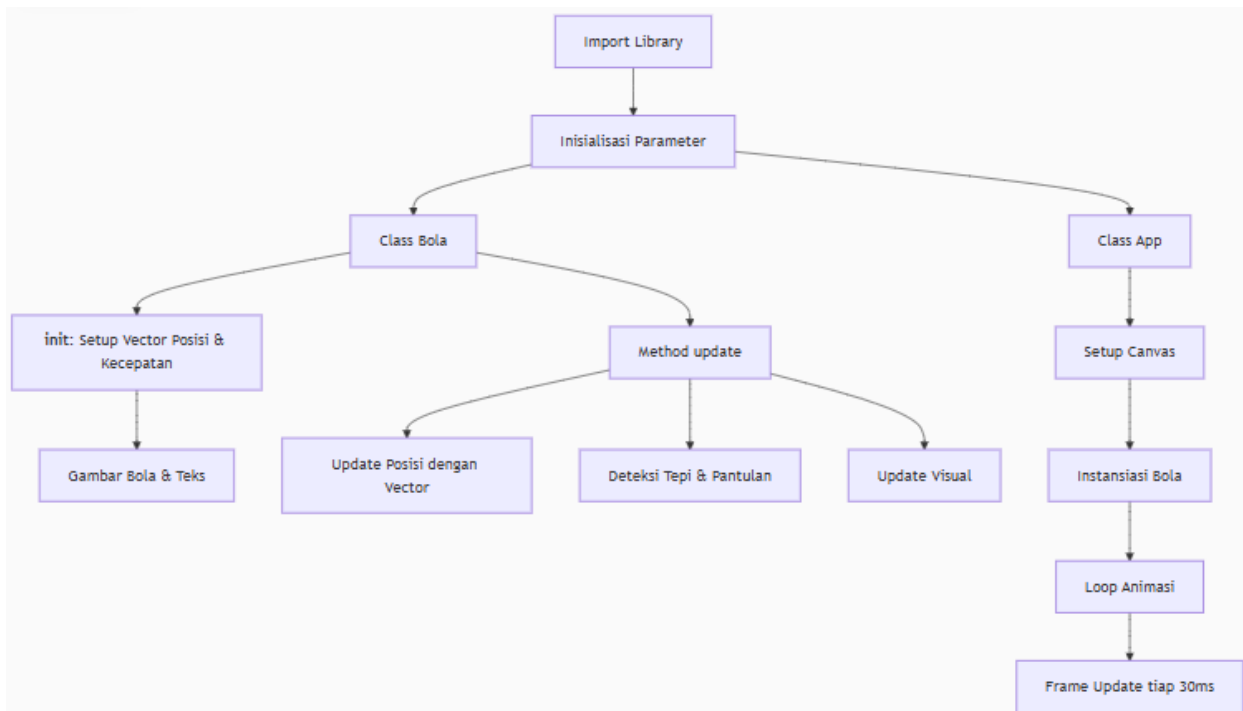
    def animate(self):
        self.bola.update()
        self.canvas.after(30, self.animate)

# Main Program
if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()

```

17. Simulasi 1 Bola Bergerak Memantul jika terkena dinding (x,y)





1. **Gerakan Bola:**
`pos_new = pos_old + vel # Operasi vector addition`
2. **Pantulan Tepi:**
 - Horizontal:
`if x ≤ 0 or x + diameter ≥ WIDTH:
 vel.x *= -1 # Membalik arah horizontal`
 - Vertikal:
`if y ≤ 0 or y + diameter ≥ HEIGHT:
 vel.y *= -1 # Membalik arah vertikal`
3. **Posisi Visual:**
`canvas.coords(shape, x, y, x+diameter, y+diameter)`

 Perhitungan Frame-by-Frame

Parameter Awal:

- Posisi: (100,100)
- Kecepatan: (5,2) pixel/frame
- Diameter: 10 pixel
- Interval: 30ms/frame (~33 FPS)

Frame 0 (Inisialisasi):

- Posisi: (100,100)
- Visual: (100,100,110,110)
- Teks: "(100,100)"

Frame 1 (30ms):


- Operasi: (100,100) + (5,2) = (105,102)
- Posisi Baru: (105,102)
- Visual: (105,102,115,112)
- Teks: "(105,102)"

Frame 2 (60ms):

- Operasi: (105,102) + (5,2) = (110,104)
- Posisi Baru: (110,104)

Frame n ($n \times 30\text{ms}$):

- Posisi Umum: $(100 + n \times 5, 100 + n \times 2)$

 Mekanisme Pantulan

Contoh Kasus Pantul:

1. **Pantul Kanan** (Frame 40):
 - Hitung: $100 + 40 \times 5 = 300$ (WIDTH=300)
 - Kondisi: $300 + 10 \geq 300 \rightarrow \text{True}$
 - Aksi: $\text{vel.x} = -5$
 - Hasil: Gerak ke kiri
2. **Pantul Bawah** (Frame 100):
 - Hitung: $100 + 100 \times 2 = 300$ (HEIGHT=300)
 - Kondisi: $300 + 10 \geq 300 \rightarrow \text{True}$
 - Aksi: $\text{vel.y} = -2$
 - Hasil: Gerak ke atas

 Visualisasi Gerakan

Frame 0: (100,100) \rightarrow

Frame 1: (105,102) \nearrow

Frame 2: (110,104) \nearrow

...


Frame 40: (300,180) \leftarrow (pantul kanan)

Frame 41: (295,182) \nwarrow


...

Frame 100: (200,300) \nwarrow (pantul bawah)

Frame 101: (195,298) \swarrow

 Keunggulan Implementasi

1. **Fleksibilitas Fisika:**
 - Mudah ditambah gravitasi:
`self.acc = Vector(0, 0.5)`
`def update(self):`
 `self.vel.add(self.acc)`
 `self.pos.add(self.vel)`
2. **Presisi Matematis:**
 - Operasi vector memastikan konsistensi perhitungan
3. **Optimasi Visual:**
 - Frame rate tinggi (30ms) untuk animasi halus

Dengan pendekatan ini, simulasi fisika menjadi lebih realistis dan mudah dikembangkan! 

```
import tkinter as tk
from vector import Vector # Gunakan vector.py

# Ukuran canvas
WIDTH, HEIGHT = 300, 300

# Bola awal
x, y, diameter, color = 100, 100, 10, "blue"

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



```

        self.canvas = canvas
        self.diameter = diameter
        self.pos = Vector(x, y)
        self.vel = Vector(5, 2)

        # Gambar bola
        self.shape = canvas.create_oval(
            self.pos.x, self.pos.y,
            self.pos.x + diameter, self.pos.y + diameter,
            fill=color, outline="black"
        )

        # Teks posisi
        self.text = canvas.create_text(
            self.pos.x, self.pos.y + diameter + 5,
            anchor='nw', text="", font=("Arial", 8), fill="black"
        )

    def update(self):
        # Tambahkan kecepatan ke posisi
        self.pos.add(self.vel)

        # Pantul jika menyentuh tepi
        if self.pos.x <= 0 or self.pos.x + self.diameter >= WIDTH:
            self.vel.x *= -1
        if self.pos.y <= 0 or self.pos.y + self.diameter >= HEIGHT:
            self.vel.y *= -1

        # Update posisi bola
        self.canvas.coords(
            self.shape,
            self.pos.x, self.pos.y,
            self.pos.x + self.diameter, self.pos.y + self.diameter
        )

        # Update teks posisi
        self.canvas.coords(self.text, self.pos.x, self.pos.y + self.diameter + 5)
        self.canvas.itemconfig(self.text, text=f"({int(self.pos.x)},
{int(self.pos.y)})")

class App:
    def __init__(self, root):
        self.root = root
        self.root.title("Simulasi Bola Pantul")
        self.canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
        self.canvas.pack()

        self.bola = Bola(self.canvas, x, y, diameter, color)
        self.animate()

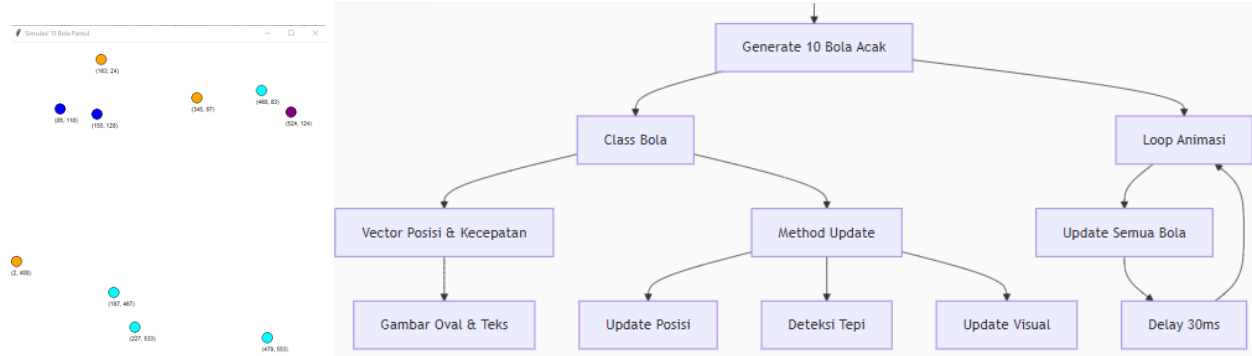
    def animate(self):
        self.bola.update()
        self.canvas.after(30, self.animate)

# Main Program

```

```
if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()
```

17. Simulasi 10 Bola Bergerak Memantul jika terkena dinding (x,y)



Rumus Fisika

- Gerakan Bola:**

$$\text{posisi_baru} = \text{posisi} + \text{kecepatan} \quad \# \text{Operasi penjumlahan vector}$$
- Pantulan Tepi:**
 if $x \leq 0$ or $x + \text{diameter} \geq \text{WIDTH}$:
 $\text{kecepatan.x} \times -1$ # Pantul horizontal
 if $y \leq 0$ or $y + \text{diameter} \geq \text{HEIGHT}$:
 $\text{kecepatan.y} \times -1$ # Pantul vertikal
- Kecepatan Acak:**

$$\text{vel} = \text{Vector}(\text{random.choice}([-3,-2,-1,1,2,3]), \text{random.choice}([-3,-2,-1,1,2,3]))$$

Perhitungan Frame-by-Frame

Parameter:

- Diameter bola: 20 pixel
- Kecepatan: [-3,-2,-1,1,2,3] pixel/frame
- Interval: 30ms/frame (~33 FPS)

Contoh 3 Bola Pertama:

Bola	Posisi Awal	Kecepatan Awal	Gerakan Awal
1	(120, 80)	(2, -1)	↘
2	(300, 400)	(-3, 2)	↖
3	(50, 500)	(1, -3)	↙

Frame 0:

- Bola 1: (120,80)-(140,100)
- Bola 2: (300,400)-(320,420)
- Bola 3: (50,500)-(70,520)

Frame 1 (30ms):

- Bola 1: $(120+2, 80-1) = (122,79)-(142,99)$
- Bola 2: $(300-3, 400+2) = (297,402)-(317,422)$
- Bola 3: $(50+1, 500-3) = (51,497)-(71,517)$

Frame n:

- Bola i: $(x_n, y_n) = (x_o + nvel_x, y_o + nvel_y)$



Mekanisme Pantulan

Contoh Kasus:

1. Bola 1:

- Frame 20: $y = 80 - 20*1 = 60$ (normal)
- Frame 40: $x = 120 + 40*2 = 200$ (normal)
- Frame 100: $y = 80 - 100*1 = -20 \rightarrow$ pantul ($y \leq 0$)
 - $vel.y = -1 \rightarrow 1$

2. Bola 2:

- Frame 50: $x = 300 - 50*3 = 150$ (normal)
- Frame 100: $x = 300 - 100*3 = 0 \rightarrow$ pantul ($x \leq 0$)
 - $vel.x = -3 \rightarrow 3$



Visualisasi Tabrakan

Frame 0:

- Bola1: $(120,80) \rightarrow$
- Bola2: $(300,400) \leftarrow$
- Bola3: $(50,500) \swarrow$

Frame 30:

- Bola1: $(180,50) \searrow$ (setelah beberapa pantul)
- Bola2: $(150,460) \nwarrow$
- Bola3: $(80,410) \swarrow$



Fitur Khusus

1. Warna Acak:

`color = random.choice(["blue","red","green","orange","purple","cyan"])`

2. Posisi Acak:

`x = random.randint(0, WIDTH-20)`

`y = random.randint(0, HEIGHT-20)`

3. Kecepatan Non-Zero:

`random.choice([-3,-2,-1,1,2,3])` # Tidak ada 0

Simulasi ini menunjukkan bagaimana OOP dan Vector mempermudah pengelolaan banyak objek dengan fisika kompleks! 🚀

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

# Ukuran canvas
WIDTH, HEIGHT = 600, 600

# Jumlah bola
JUMLAH_BOLA = 10
```

```

class Bola:
    def __init__(self, canvas, x, y, diameter, color):
        self.canvas = canvas
        self.diameter = diameter
        self.pos = Vector(x, y)
        # Kecepatan acak antara -3 dan 3 (hindari 0)
        self.vel = Vector(random.choice([-3, -2, -1, 1, 2, 3]),
                           random.choice([-3, -2, -1, 1, 2, 3]))

        # Gambar bola
        self.shape = canvas.create_oval(
            x, y, x + diameter, y + diameter, fill=color, outline="black"
        )

        # Teks posisi di bawah bola
        self.text = canvas.create_text(
            x, y + diameter + 5, anchor='nw',
            text="", font=("Arial", 8), fill="black"
        )

    def update(self):
        # Tambah posisi
        self.pos.add(self.vel)

        # Cek tabrakan tepi dan pantulkan
        if self.pos.x <= 0 or self.pos.x + self.diameter >= WIDTH:
            self.vel.x *= -1
        if self.pos.y <= 0 or self.pos.y + self.diameter >= HEIGHT:
            self.vel.y *= -1

        # Perbarui posisi bola
        self.canvas.coords(
            self.shape,
            self.pos.x, self.pos.y,
            self.pos.x + self.diameter, self.pos.y + self.diameter
        )

        # Perbarui teks posisi
        self.canvas.coords(self.text, self.pos.x, self.pos.y + self.diameter + 5)
        self.canvas.itemconfig(self.text, text=f"({int(self.pos.x)},
{int(self.pos.y)})")

class App:
    def __init__(self, root):
        self.root = root
        self.root.title("Simulasi 10 Bola Pantul")
        self.canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
        self.canvas.pack()

        self.bolas = []
        for _ in range(JUMLAH_BOLA):
            x = random.randint(0, WIDTH - 20)
            y = random.randint(0, HEIGHT - 20)
            color = random.choice(["blue", "red", "green", "orange", "purple",
"cyan"])

```

```
        bola = Bola(self.canvas, x, y, diameter=20, color=color)
        self.bolas.append(bola)

    self.animate()

    def animate(self):
        for bola in self.bolas:
            bola.update()
        self.canvas.after(30, self.animate)

# Main Program
if __name__ == "__main__":
    root = tk.Tk()
    app = App(root)
    root.mainloop()
```