

## # 1. VARIABEL

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

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

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

# Assignment (Pengisian Nilai)

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

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

```
print(x)
```

# contoh 1 keranjang a bulpen berisi sejumlah 6 bulpen

# contoh 1 keranjang b bulpen berisi sejumlah 4 buku

## #2. DICT, LIST, TUPLE

#LIST

#Contoh list (mutable)

#array (bisa diubah)

# keranjang buah berisi : berbagai jenis buah

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

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

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

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

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

#TUPLE

#tuple di Python ≈ array dengan const di JavaScript

#array (tidak diubah)

# sebuah alamat dan kode plat kendaraan

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

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

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

# DICT di Python = object di JavaScript

# kumpulan data user : nama, usia

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

```

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

```

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

### # 3. FUNCTION (FUNGSI)

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

```

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

```

```

fungsi1() # Halo, ini fungsi!

```

```

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

```

```

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

```

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

```

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

```

```

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

```

#\*args → untuk jumlah argumen tak terbatas (seperti array)

#\*angka akan berisi tuple (3, 4, 5)

```

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

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

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

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

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

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

```

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

#### # 4. GLOBAL VARIABLE

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

```

nama = "silmi" # variabel global

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

sapa() # Halo, ini fungsi!

```

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

## 5. Apa Itu Ternary Operator di Python?

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

---

### **Format Umum:**

nilai\_jika\_true if kondisi else nilai\_jika\_false

---


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

```
a = 5
```

```
b = 10
```

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

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

 **Output:**

Nilai terbesar adalah: 10

### **Versi biasa:**

```
if a > b:
```

```
    maks = a
```

```
else:
```

```
    maks = b
```


---

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

```
angka = 7
```

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

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

 **Output:**

7 adalah bilangan Ganjil

### **Versi biasa:**

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

```
    jenis = "Genap"
```

```
else:
```

```
    jenis = "Ganjil"
```


---

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

```
umur = 15
```

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

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

 Output:  
Status: Anak-anak

### Kapan Dipakai?

Gunakan ternary operator jika:

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

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

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

# max() Ambil data dengan nilai terbesar

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

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

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

# sum() Jumlahkan semua nilai

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

# sorted() Urutkan data

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

# filter() → Menyaring Data Sesuai Kondisi

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

```
print(genap) #
```

# map() → Mengubah Setiap Elemen

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

```
print(dikali2) #
```

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

### Operasi Aritmatika (Matematika)

Operator	Arti	Contoh	Hasil
+	Penjumlahan	5 + 2	7

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

### Assignment (Pengisian Nilai)

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

### Operasi Logika (Boolean)

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

### Operator Perbandingan

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


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

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

#  1. Akses Data

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

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

#  3. Ubah Data dalam Dict

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

# Loop Semua Data

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

#Silmi tinggal di Semarang

#Edy tinggal di Jakarta

#gita tinggal di Bandung

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

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

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

for s in siswa:



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

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

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

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

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

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

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

# Versi OOP (Class)

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

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

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

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

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

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

---

### Kenapa Class Lebih Baik?

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

---

### Perbedaan self dan super()

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



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

---

## #9. Fungsi Khusus untuk List of Object

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

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

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

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

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

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

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

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

```

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

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

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

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

```

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

```

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

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

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

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

```

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

```

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

    def tampilkan(self):

```

```

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

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

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

```

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

```

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

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

```

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

```

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

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

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

```

```

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

```

```

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

```

```

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

```

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

```

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

```

```

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

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

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

```

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

```

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

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

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

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

```

### TKINTER

Dalam **Tkinter**, Canvas adalah widget yang digunakan untuk menggambar bentuk-bentuk grafis seperti garis, lingkaran, persegi panjang, teks,

gambar, dan animasi. Canvas sering dipakai untuk simulasi visual dan permainan.

## 1. Pengertian Canvas

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

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

Contoh dasar pembuatan canvas:

```
import tkinter as tk

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

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

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

### ✓ Tabel Fungsi Dasar canvas Tkinter

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

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

### Penjelasan Tambahan

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

## 2. Sistem Koordinat di Canvas Tkinter

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

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

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

|  
|  
|  
v

Y (800)

### Penjelasan Koordinat

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

Contoh:

# Gambar titik di tengah canvas

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

## 3. Contoh Menggambar Berbagai Objek

# Garis dari kiri atas ke kanan bawah

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

# Persegi panjang di tengah

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

# Lingkaran (oval) di tengah
canvas.create_oval(350, 350, 450, 450, outline="red", fill="pink")

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

---

#### 4. Tips Penggunaan

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

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

```
import tkinter as tk

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

#Main Program
#.....
posisi = canvas.create_oval(390, 0, 410, 20, fill="green")

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

gerak()

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

---

#### Kesimpulan

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

Berikut ini penjelasan sistem koordinat di Tkinter Canvas

---


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

Dalam Tkinter Canvas:

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

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


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

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

---

### 2. Bergerak ke Kanan = Tambah X


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

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

---

### 3. Bergerak ke Bawah = Tambah Y

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

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

---

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

Misalnya ukuran canvas 800x800, titik tengahnya adalah:

```
canvas.create_oval(395, 395, 405, 405, fill="orange")
canvas.create_text(400, 380, text="Tengah (400,400)", fill="black")
```

---

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

# Kiri Atas (0, 0)

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

# Kanan Atas (800, 0)

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

# Kiri Bawah (0, 800)

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

# Kanan Bawah (800, 800)

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

---

### 6. Buat Kotak di Tengah Canvas

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

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



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

---

### **Penjelasan Mudah**

Bayangkan kamu menggambar di kertas. Tapi bedanya:

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

---

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

---

### **Tujuan Simulasi**

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

---

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

```
import tkinter as tk

# Buat jendela utama
root = tk.Tk()
root.title("Simulasi Sistem Koordinat Canvas")

# Buat Canvas 800x800
canvas = tk.Canvas(root, width=800, height=800, bg="white")
canvas.pack()

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

# Fungsi saat mouse diklik
def show_coordinates(event):
    x, y = event.x, event.y
    # Gambar titik kecil di lokasi klik
    canvas.create_oval(x-5, y-5, x+5, y+5, fill="red")
    # Tampilkan koordinat di label
    label.config(text=f"Koordinat: x={x}, y={y}")
    # Tampilkan teks koordinat di canvas dekat titik
```

```

        canvas.create_text(x+30, y, text=f"({x},{y})", anchor="w",
        fill="blue", font=("Arial", 10))

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

# Tambahkan garis sumbu
canvas.create_line(0, 0, 800, 0, fill="gray")    # Garis horizontal
atas
canvas.create_line(0, 0, 0, 800, fill="gray")    # Garis vertikal kiri
canvas.create_text(10, 10, text="(0, 0)", anchor="nw", fill="black")

# Jalankan aplikasi
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.

## # 5. Menggambar object >> 1 BOLA

```

# 1. import
import tkinter as tk

# 2. fungsi/class

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

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

# 4. main program
# Menggambar canvas >> lihat sintaks
bola = canvas.create_oval(10, 60, 30, 80, fill="red")
garis = canvas.create_line(0, 70, 300, 70, fill="black")

```

```
tulisan = canvas.create_text(150, 20, text="Simulasi Bola",  
font=("Arial", 14))
```

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

---

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

```
# salah karena setiap frame membuat object bola baru
```

```
# 1. import  
import tkinter as tk
```

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

```
# Posisi awal  
x = 10  
y = 60
```

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

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

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

```
gerak()
```

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

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

```
# 8. Menggambar object, Menggerakkan object >> CLASS >> 1 BOLA
# DENGAN CLASS BOLA, DAN MOVE
```

```
# 1. import
import tkinter as tk

# 2. fungsi/class
# Kelas Bola
class Bola:
    def __init__(self, canvas, x, y, warna):
        self.canvas = canvas
        self.shape = canvas.create_oval(x, y, x+20, y+20, fill=warna)
        self.kecepatan = 5

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

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

# 4. main program
# 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

Aspek	Script 7 (Fungsi)	Script 8 (Class)
Reusability (kode dapat dipakai ulang)	Rendah	Tinggi
Kejelasan tanggung jawab objek	Tidak terpisah	Jelas: setiap objek mengurus dirinya
Cocok untuk pembelajaran awal	✓ Ya	✓ Ya, jika sudah paham class
Cocok untuk simulasi kompleks	✗ Kurang cocok	✓ Sangat cocok

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

```
# 1. import
import tkinter as tk

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

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

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

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

# Buat dua bola dengan kecepatan berbeda
bola_merah = Bola(canvas, 10, 90, "red", kecepatan=5)
```

```
bola_biru = Bola(canvas, 10, 120, "blue", kecepatan=3)
```

```
# Jalankan keduanya
```

```
bola_merah.gerak()
```

```
bola_biru.gerak()
```

```
# 5. loop
```

```
root.mainloop()
```

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

```
# 1. import
```

```
import tkinter as tk
```

```
# 2. fungsi/class
```

```
# Buat class Vector2D sendiri
```

```
class Vector:
```

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

```
        self.x = x
```

```
        self.y = y
```

```
# Tambah vektor
```

```
def add(self, other):
```

```
    self.x += other.x
```

```
    self.y += other.y
```

```
# Kelas Bola pakai posisi & kecepatan vektor
```

```
class Bola:
```

```
    def __init__(self, canvas, x, y, warna):
```

```
        self.canvas = canvas
```

```
        self.position = Vector(x, y)          # Posisi awal
```

```
        self.velocity = Vector(2, 0)          # Kecepatan ke kanan
```

```
        self.radius = 10                      # Ukuran bola
```

```
        self.shape = canvas.create_oval(
```

```
            x, y, x + self.radius*2, y + self.radius*2, fill=warna
```

```
        )
```

```
    def update(self):
```

```
        # Geser objek relatif dengan kecepatan
```

```
        self.canvas.move(self.shape, self.velocity.x, self.velocity.y)
```

```
        # Update juga posisi internal untuk keperluan logika lain
```

```
        self.position.add(self.velocity)
```

```
        # Jadwalkan update selanjutnya
```

```
        self.canvas.after(50, self.update)
```

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

# 4. main program
# 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 ingin gerakan bola ke kanan.

#### VS Perbandingan Langsung

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

#### 🎓 Kesimpulan Akhir

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

Kalau kamu ingin melanjutkan dengan simulasi **gravitasi**, **percepatan**, **tumbukan**, atau **interaksi antar bola**, maka **struktur dengan Vector (Script 10)** adalah fondasi terbaik.

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

```
# 1. import
import tkinter as tk
```

```
# 2. fungsi/class
# Buat class Vector2D sendiri
class Vector:
```

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

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

```
# Kelas Bola pakai posisi & kecepatan vektor
class Bola:
```

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

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

```

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

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

# 4. main program
# 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 10: Update Posisi dengan canvas.move()**

```

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

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

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

```

#### **Cara Kerja:**

- Menggerakkan bola dengan fungsi **canvas.move()** yang menggeser objek secara relatif (relative move).

- Setelah itu, update posisi internal `self.position` dengan menambahkan `velocity`.
- Jadi, posisi bola di canvas digeser *berdasarkan kecepatan*.

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

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

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

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

#### Cara Kerja:

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

### ⚖ Perbandingan Utama

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

Aspek	Script 10 (canvas.move)	Script 11 (canvas.coords)
	jika ingin manipulasi posisi kompleks	posisi kompleks dan koreksi posisi
Kejelasan kode	Lebih sederhana dan intuitif jika hanya ingin menggerakkan objek	Perlu perhitungan ulang posisi dan koordinat setiap frame
Kinerja	Biasanya sedikit lebih cepat karena hanya menggeser	Sedikit lebih berat karena menggambar ulang posisi

### 🌟 Kapan Pakai Mana?

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

### 💡 Ringkasan

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

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

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

### 🟢 Inti Masalah

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

Ada 2 cara utama di Tkinter:

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

- Artinya: ubah posisi objek ke posisi tertentu.
- Cocok jika kamu tahu koordinat pasti.
- Harus hitung ulang posisi kiri atas dan kanan bawah.

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

## 2. `canvas.move(item, dx, dy)`

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

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

---

## Analogi Sehari-Hari

### Gaya Gerak Analogi

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

`move()` Mengemudi (geser perlahan)

---

### Contoh Kode

#### Dengan `coords()`:

# Memindahkan bola ke posisi x, y tertentu

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



#### Dengan `move()`:

# Menggeser bola berdasarkan kecepatan

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

---

## Kapan Gunakan yang Mana?

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

---



## Tips Sempel


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

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

---

## Kesimpulan Akhir

Perbandingan	<code>move()</code>	<code>coords()</code>
Jenis gerak	Relatif (geser) 	Absolut (lompat ke posisi)
Mudah dipakai	Ya, cukup tahu dx, dy 	Tidak, harus hitung posisi

Perbandingan	move()	coords()
Cocok animasi	Ya 	Kurang cocok

---

