The Nature of Code

by Daniel Shiffman
https://github.com/edycoleee/nature
https://editor.p5js.org/natureofcode/collections

Chapter 2. Forces

2.1 Forces and Newton's Laws of Motion

Penjelasan Materi Bab 2: Forces

1. Apa itu gaya (force)?

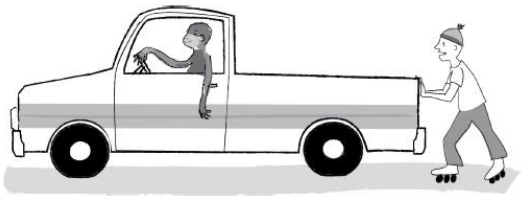
- Gaya itu adalah dorongan atau tarikan yang membuat benda bisa bergerak atau berubah kecepatan.
- Gaya punya arah dan besar, jadi dia seperti **vektor** (yang sudah kamu pelajari di bab 1).
- Contohnya, saat kamu mendorong meja, itu adalah gaya yang kamu berikan ke meja.

2. Hukum Newton 1: Benda diam tetap diam, benda bergerak tetap bergerak

- Kalau tidak ada gaya yang bekerja pada benda, benda yang diam akan tetap diam.
- Benda yang bergerak juga akan terus bergerak dengan kecepatan dan arah yang sama.
- Contohnya, bola yang kamu lempar di ruang hampa udara akan terus bergerak tanpa berhenti, karena tidak ada gaya yang melambatkannya.
- Tapi di dunia nyata, gaya seperti gesekan dan udara membuat benda akhirnya berhenti.

3. Hukum Newton 3: Gaya selalu ada pasangan

- Kalau kamu mendorong dinding, dinding juga memberi gaya balik ke tanganmu.
- Dua gaya ini besarnya sama tapi arahnya berlawanan.
- Tapi gaya-gaya ini bekerja pada benda yang berbeda, jadi mereka tidak saling meniadakan.
- Contohnya kamu mendorong mobil berat, kamu akan merasakan gaya balik di tanganmu, tapi mobilnya mungkin tidak bergerak.



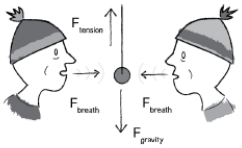


Figure 2.1: The pendulum doesn't move because all the forces cancel each other out (add up to a net force of zero).

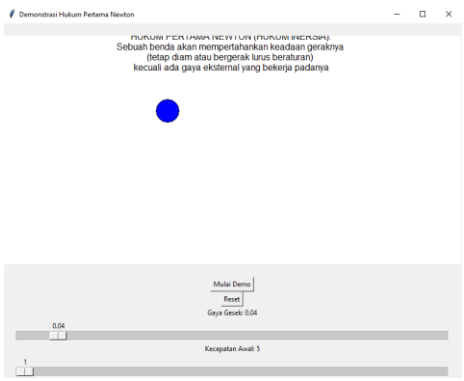
4. Menghubungkan gaya dan percepatan

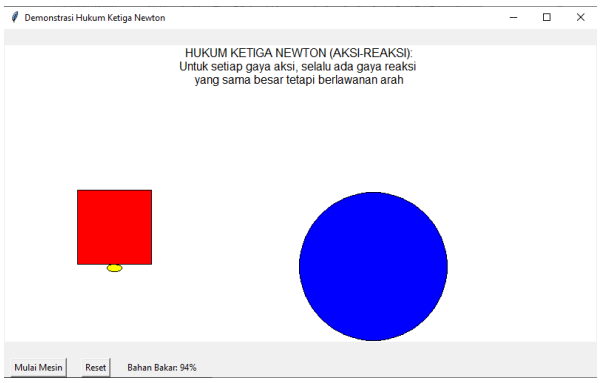
- Gaya bisa membuat benda berubah kecepatan, atau disebut percepatan.
- Semakin besar gaya yang bekerja, semakin besar percepatan benda itu (ini nanti di hukum Newton 2 yang akan dipelajari).
- Dalam simulasi, kita bisa hitung gaya dan gunakan untuk ubah kecepatan dan posisi benda.

Jika sebuah benda dalam keadaan equilibrium (keseimbangan gaya), maka:

- Semua gaya yang bekerja meniadakan satu sama lain → total (resultan) gaya = nol
- Maka tidak ada percepatan (a = 0) → berdasarkan hukum Newton ke-2: F = m × a
- Jadi, kecepatan (velocity) tidak berubah
- Akibatnya, gerakan benda akan tetap konstan, baik dalam arah maupun besarannya

```
velocity = PVector(2, 3)  # Kecepatan awal
if total_force.mag() == 0:
    # Tidak ada gaya yang bekerja → equilibrium
    # Maka velocity tetap
    position.add(velocity)  # Posisi berubah secara linear
```





Konsep Hukum Pertama Newton
Hukum Pertama Newton (Hukum Inersia) menyatakan:
"Sebuah benda yang diam akan tetap diam, dan benda yang bergerak akan tetap bergerak dengan kecepatan konstan pada garis lurus, kecuali jika ada gaya eksternal yang bekerja padanya.":

1. **Benda Diam**: Jika tidak ada gaya yang bekerja, benda akan tetap diam
2. **Benda Bergerak**: Jika tidak ada gaya yang bekerja, benda akan terus bergerak lurus dengan kecepatan sama
3. **Gaya Eksternal**: Untuk mengubah keadaan gerak benda, diperlukan gaya dari luar

Berikut program demonstrasi Hukum Pertama Newton dengan penjelasan setiap bagian:

COPY PASTE

```
#SCRIPT 1 - Demonstrasi Hukum Pertama Newton (Hukum Inersia) dengan Tkinter
# ----------------------------
# ⚙️ Demonstrasi Hukum Newton 1
# ----------------------------
import tkinter as tk
import random
from vector import Vector

class NewtonFirstLawDemo:
    def __init__(self, root):
        self.root = root
        self.root.title("Demonstrasi Hukum Pertama Newton")

        # Ukuran canvas
        self.width = 800
        self.height = 400
        self.canvas = tk.Canvas(root, width=self.width, height=self.height, bg='white')
        self.canvas.pack(pady=20)

        # Penjelasan teks
        explanation = """HUKUM PERTAMA NEWTON (HUKUM INERSIA):
Sebuah benda akan mempertahankan keadaan geraknya
(tetap diam atau bergerak lurus beraturan)
kecuali ada gaya eksternal yang bekerja padanya"""
        self.canvas.create_text(self.width//2, 30, text=explanation, font=('Arial',
12), justify='center')

        # Tombol kontrol
        self.start_btn = tk.Button(root, text="Mulai Demo", command=self.start_demo)
        self.start_btn.pack()
        self.reset_btn = tk.Button(root, text="Reset", command=self.reset)
        self.reset_btn.pack()

        # Slider gaya gesek
        self.friction_label = tk.Label(root, text="Gaya Gesek: 0")
```

```
        self.friction_label.pack()
        self.friction_slider = tk.Scale(root, from_=0, to=0.5, resolution=0.01,
orient='horizontal', command=self.set_friction)
        self.friction_slider.pack(fill='x', padx=20)

        # Slider kecepatan awal
        self.velocity_label = tk.Label(root, text="Kecepatan Awal: 5")
        self.velocity_label.pack()
        self.velocity_slider = tk.Scale(root, from_=1, to=20, orient='horizontal',
command=self.set_velocity)
        self.velocity_slider.pack(fill='x', padx=20)

        # Inisialisasi variabel
        self.ball = None
        self.ball_moving = False
        self.velocity = 5
        self.friction = 0

    def set_friction(self, value):
        self.friction = float(value)
        self.friction_label.config(text=f"Gaya Gesek: {self.friction:.2f}")

    def set_velocity(self, value):
        self.velocity = int(value)
        self.velocity_label.config(text=f"Kecepatan Awal: {self.velocity}")

    def start_demo(self):
        if not self.ball_moving:
            self.create_ball()
            self.ball_moving = True
            self.move_ball()

    def create_ball(self):
        if self.ball:
            self.canvas.delete(self.ball)
        start_y = random.randint(80, self.height - 50)

        # Posisi dan kecepatan pakai Vector
        self.pos = Vector(50, start_y)
        self.vel = Vector(self.velocity, 0)

        radius = 20
        self.radius = radius
        self.ball = self.canvas.create_oval(
            self.pos.x - radius, self.pos.y - radius,
            self.pos.x + radius, self.pos.y + radius,
            fill='blue'
        )

    def move_ball(self):
        if self.ball_moving:
            print(f"\n📍 Posisi: {self.pos}, Kecepatan: {self.vel}, Gesekan:
{self.friction}")

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

            # Update posisi bola di canvas
            self.canvas.coords(
                self.ball,
                self.pos.x - self.radius, self.pos.y - self.radius,
                self.pos.x + self.radius, self.pos.y + self.radius
```

```
            )

            # Terapkan gesekan jika ada
            if self.friction > 0:
                self.vel.mult(1 - self.friction)
                if self.vel.mag() < 0.01:
                    self.vel = Vector(0, 0)

            # Jika bola keluar dari sisi kanan, kembali ke kiri
            if self.pos.x > self.width + self.radius:
                self.pos.x = -self.radius
            elif self.pos.x < -self.radius:
                self.pos.x = self.width + self.radius

            # Lanjut animasi jika masih ada gerakan
            if self.vel.mag() > 0:
                self.root.after(50, self.move_ball)
            else:
                self.ball_moving = False

    def reset(self):
        self.ball_moving = False
        if self.ball:
            self.canvas.delete(self.ball)
            self.ball = None

# Jalankan program
if __name__ == "__main__":
    root = tk.Tk()
    app = NewtonFirstLawDemo(root)
    root.mainloop()
```

Hukum Aksi-Reaksi: roket mendorong gas (aksi ke bawah), dan gas mendorong roket (reaksi ke atas
Berikut program demonstrasi Hukum Ketiga Newton:

```
#COPY PASTE
#SCRIPT 2 - Demonstrasi Hukum Ketiga Newton (Aksi-Reaksi) dengan Tkinter
import tkinter as tk
from vector import Vector

class NewtonThirdLawDemo:
    def __init__(self, root):
        self.root = root
        self.root.title("Demonstrasi Hukum Ketiga Newton")

        # Inisialisasi canvas
        self.width = 800
        self.height = 400
        self.canvas = tk.Canvas(root, width=self.width, height=self.height, bg='white')
        self.canvas.pack(pady=20)

        # Tambahkan teks penjelasan
        explanation = """HUKUM KETIGA NEWTON (AKSI-REAKSI):
Untuk setiap gaya aksi, selalu ada gaya reaksi
yang sama besar tetapi berlawanan arah"""
        self.canvas.create_text(self.width//2, 30, text=explanation, font=('Arial',
12), justify='center')

        # Buat objek roket dan bumi
        self.rocket = self.canvas.create_rectangle(100, 200, 200, 300, fill='red',
tags='rocket')
```

```
        self.exhaust = self.canvas.create_oval(90, 300, 110, 310, fill='yellow',
state='hidden', tags='exhaust')
        self.earth = self.canvas.create_oval(400, 200, 600, 400, fill='blue',
tags='earth')

        # Posisi dan kecepatan roket menggunakan vektor
        self.rocket_pos = Vector(150, 250)
        self.rocket_vel = Vector(0, 0)
        self.earth_pos = Vector(500, 300)  # Posisi pusat bumi

        # Parameter gaya
        self.thrust = 0.5     # Gaya dorong dari roket
        self.gravity = 0.1    # Konstanta gravitasi simulasi
        self.fuel = 100       # Bahan bakar

        # Tombol kontrol
        self.start_btn = tk.Button(root, text="Mulai Mesin", command=self.start_engine)
        self.start_btn.pack(side='left', padx=10)
        self.reset_btn = tk.Button(root, text="Reset", command=self.reset)
        self.reset_btn.pack(side='left', padx=10)

        # Label bahan bakar
        self.fuel_label = tk.Label(root, text="Bahan Bakar: 100%")
        self.fuel_label.pack(side='left', padx=10)

        # Status simulasi
        self.simulating = False
        self.root.after(100, self.update)

    def start_engine(self):
        # Nyalakan mesin roket jika masih ada bahan bakar
        if self.fuel > 0:
            self.simulating = True
            self.canvas.itemconfig('exhaust', state='normal')

    def reset(self):
        # Reset posisi, kecepatan, bahan bakar
        self.simulating = False
        self.canvas.coords('rocket', 100, 200, 200, 300)
        self.canvas.itemconfig('exhaust', state='hidden')
        self.rocket_pos = Vector(150, 250)
        self.rocket_vel = Vector(0, 0)
        self.fuel = 100
        self.fuel_label.config(text="Bahan Bakar: 100%")

    def update(self):
        if self.simulating and self.fuel > 0:
            # AKSI: roket mendorong gas ke bawah → gaya ke atas
            force_up = Vector(0, -self.thrust)

            # Terapkan gaya reaksi ke roket (menambah kecepatan ke atas)
            self.rocket_vel.add(force_up)

            # Kurangi bahan bakar
            self.fuel -= 0.5
            self.fuel_label.config(text=f"Bahan Bakar: {max(0, int(self.fuel))}%")

            # Gaya gravitasi antara bumi dan roket (arah ke bumi)
            direction = Vector(self.earth_pos.x - self.rocket_pos.x,
                               self.earth_pos.y - self.rocket_pos.y)
            distance = max(50, direction.mag())  # Hindari jarak terlalu kecil
            direction.normalize()
```

```
            # Hitung besar gaya gravitasi dan ubah menjadi vektor
            gravity_strength = self.gravity * 1000 / (distance**2)
            gravity_force = direction.copy()
            gravity_force.mult(gravity_strength)

            # Terapkan gaya gravitasi ke roket
            self.rocket_vel.add(gravity_force)

            # Update posisi roket berdasarkan kecepatannya
            self.rocket_pos.add(self.rocket_vel)

            # Update posisi di canvas (menggunakan canvas.coords)
            x, y = self.rocket_pos.x, self.rocket_pos.y
            self.canvas.coords('rocket', x-50, y-50, x+50, y+50)
            self.canvas.coords('exhaust', x-10, y+50, x+10, y+60)

            # Jika bahan bakar habis, sembunyikan api knalpot
            if self.fuel <= 0:
                self.canvas.itemconfig('exhaust', state='hidden')

        # Jalankan fungsi ini terus-menerus setiap 50 ms
        self.root.after(50, self.update)

# Jalankan program
if __name__ == "__main__":
    root = tk.Tk()
    app = NewtonThirdLawDemo(root)
    root.mainloop()
```

Demonstrasi Konsep

1. **Peluncuran Roket**:
    - Mesin roket mendorong gas ke bawah (aksi)
    - Gas mendorong roket ke atas (reaksi)
    - Tanpa gaya reaksi, roket tidak bisa terbang
2. **Gravitasi Bumi**:
    - Bumi menarik roket ke pusatnya (aksi)
    - Roket menarik bumi dengan gaya sama (reaksi)
    - Karena massa bumi sangat besar, efek pada bumi tidak terlihat
3. **Kontrol Pengguna**:
    - Tombol "Mulai Mesin" untuk memulai aksi-reaksi
    - Bahan bakar terbatas menunjukkan gaya hanya ada saat ada interaksi
    - Reset untuk mengulang percobaan

Analogi Sederhana

1. **Berjalan**:
    - Kaki mendorong lantai ke belakang (aksi)
    - Lantai mendorong kaki ke depan (reaksi)
    - Hasilnya: Kita bergerak maju
2. **Mendorang Tembok**:
    - Tangan mendorong tembok (aksi)
    - Tembok mendorong tangan (reaksi)
    - Jika pakai sepatu roda, kita akan terdorong mundur

2.2 - Gaya dan Hukum Newton II

Penjelasan Materi Bab 2.2: Newton’s Second Law (Hukum Kedua Newton)

1. Apa isi Hukum Newton II?

- **Hukum Newton II bilang: Gaya = massa × percepatan**
- Ditulis: **F = m × a**
- Artinya:
    - Kalau kamu kasih gaya lebih besar, benda akan lebih cepat bergerak (percepatan lebih besar).
    - Kalau benda punya massa (berat) yang besar, percepatan jadi lebih kecil untuk gaya yang sama.
- Jadi percepatan benda itu berbanding lurus dengan gaya, dan berbanding terbalik dengan massanya.

2. Perbedaan Massa dan Berat

- **Massa** = jumlah materi dalam benda (misalnya 1 kg), tetap sama dimana saja.
- **Berat** = gaya gravitasi yang menarik benda ke bawah, berat berubah tergantung gravitasi (misal di Bumi atau di Bulan).

3. Kenapa penting untuk pemrograman simulasi?

- Dengan hukum ini, kita bisa membuat simulasi yang lebih realistis.
- Kita bisa membuat fungsi applyForce yang akan menambahkan gaya ke suatu benda.
- Fungsi itu akan menghitung percepatan baru dengan rumus:
- percepatan = gaya / massa
- Dengan percepatan, kita bisa update kecepatan dan posisi benda.

```
#SCRIPT 3 - Demonstrasi Hukum Kedua Newton (F = m * a) dengan Simulasi
from vector import Vector

# === CLASS MOVER UNTUK BENDA ===
class Mover:
    def __init__(self, mass, x, y):
        self.mass = mass
        self.location = Vector(x, y)
        self.velocity = Vector(0, 0)
        self.acceleration = Vector(0, 0)

    def applyForce(self, force):
        # F = m * a → a = F / m
        f = force.copy()
        f.div(self.mass)
        self.acceleration.add(f)

    def update(self):
        self.velocity.add(self.acceleration)
        self.location.add(self.velocity)
        self.acceleration.mult(0)  # reset after update

# === INISIALISASI ===
mass = 1  # bola biru
mover = Mover(mass=mass, x=100, y=200)
wind = Vector(0.1, 0)  # gaya angin

# === SIMULASI 10 LANGKAH ===
print("Langkah | Gaya (F) | Percepatan (a) | Kecepatan (v) | Posisi (x)")
print("--------|----------|----------------|----------------|----------------")

for step in range(11):
    # Tampilkan data sebelum update
    print(f"{step:>6} | "
          f"({wind.x:.2f}) | "
          f"({wind.x / mover.mass:.2f})        | "
          f"({mover.velocity.x:.2f})        | "
          f"({mover.location.x:.2f})")
```

```
    # Terapkan gaya dan update posisi
    mover.applyForce(wind)
    mover.update()
```

Langkah	Gaya (F)	Percepatan (a)	Kecepatan (v)	Posisi (x)
0	(0.10)	(0.10)	(0.00)	(100.00)
1	(0.10)	(0.10)	(0.10)	(100.10)
2	(0.10)	(0.10)	(0.20)	(100.30)
3	(0.10)	(0.10)	(0.30)	(100.60)
4	(0.10)	(0.10)	(0.40)	(101.00) .....

Kesimpulannya:

- Gaya **konstan** sebesar 0.1 (N) diterapkan pada benda dengan massa 1 kg.
- Percepatan yang dihasilkan adalah $a = \frac{F}{m} = \frac{0.1}{1} = 0.1$ (m/s²)tetap konstan setiap langkah.
- Karena percepatan konstan, kecepatan bertambah sebesar 0.1 setiap langkah (bertambah terus menerus).
- Akibatnya, posisi benda terus bertambah semakin jauh setiap langkah karena kecepatan bertambah terus.
- Jadi, gaya konstan menyebabkan percepatan konstan yang membuat kecepatan bertambah secara linear, sehingga benda bergerak semakin cepat tiap waktu.

Ini adalah ilustrasi langsung dari Hukum Kedua Newton (F = m * a) dan gerak dengan percepatan konstan.

Contoh Simulasi Python Tkinter
Konsep:

- Kita punya objek bergerak (lingkaran)
- Kita bisa berikan gaya (seperti angin atau gravitasi) ke objek itu
- Objek akan bergerak sesuai gaya dan massanya

"Kalau dua benda didorong dengan gaya yang sama, benda yang **lebih ringan** akan bergerak **lebih cepat**. Ini karena percepatannya lebih besar."
Ini adalah **Hukum Newton II**:

Ini adalah **Hukum Newton II**:

$$\text{Percepatan (a)} = \frac{\text{Gaya (F)}}{\text{Massa (m)}}$$

Misal:

- Gaya $F = 0.1$
- Massa $m = 1$ → maka percepatan $a = 0.1/1 = 0.1$
- Massa $m = 4$ → maka percepatan $a = 0.1/4 = 0.025$

Jadi benda berat **bergerak lebih lambat** walau gayanya sama.

💻 **Kode + Komentar Lengkap + Perhitungan ditampilkan**

```
#SCRIPT 4 - Demonstrasi Hukum Kedua Newton (F = m * a) dengan Tkinter
import tkinter as tk
from vector import Vector

# === CLASS UNTUK BENDA BERGERAK (MOVER) ===
class Mover:
    def __init__(self, mass, x, y, canvas, color):
        self.mass = mass
        self.location = Vector(x, y)
        self.velocity = Vector(0, 0)
        self.acceleration = Vector(0, 0)
        self.radius = mass * 10
        self.canvas = canvas
        self.color = color
```

```
        # Gambar objek bola sekali saja, lalu simpan ID-nya
        r = self.radius
        self.id = canvas.create_oval(x - r, y - r, x + r, y + r, fill=color)

    def applyForce(self, force):
        # F = m * a → a = F / m
        f = force.copy()
        f.div(self.mass)
        self.acceleration.add(f)

    def update(self):
        self.velocity.add(self.acceleration)
        self.location.add(self.velocity)
        self.acceleration.mult(0)

        # Update posisi di canvas dengan canvas.coords
        x, y = self.location.x, self.location.y
        r = self.radius
        self.canvas.coords(self.id, x - r, y - r, x + r, y + r)

# === CLASS SIMULASI UTAMA ===
class Simulation:
    def __init__(self, root):
        self.root = root
        self.width = 600
        self.height = 400
        self.canvas = tk.Canvas(root, width=self.width, height=self.height, bg="white")
        self.canvas.pack()

        # Gaya angin (konstan ke kanan)
        self.wind = Vector(0.1, 0)

        # Inisialisasi dua mover dengan massa berbeda
        self.mover1 = Mover(mass=1, x=100, y=150, canvas=self.canvas, color="blue")
        self.mover2 = Mover(mass=4, x=100, y=250, canvas=self.canvas, color="red")

        # Label untuk info teks perhitungan
        self.info_text = tk.Label(root, font=("Courier", 12), justify="left",
bg="white", anchor="w")
        self.info_text.pack(fill="both")

        self.update()

    def update(self):
        # Terapkan gaya ke kedua mover
        self.mover1.applyForce(self.wind)
        self.mover2.applyForce(self.wind)

        # Update posisi mereka
        self.mover1.update()
        self.mover2.update()

        # Hitung percepatan masing-masing untuk ditampilkan
        acc1 = self.wind.copy()
        acc1.div(self.mover1.mass)
        acc2 = self.wind.copy()
        acc2.div(self.mover2.mass)

        # Update teks info
        info = (
            f"Gaya angin (F): ({self.wind.x:.2f}, {self.wind.y:.2f})\n"
```

```
            f"Mover Biru (massa={self.mover1.mass}): a = F/m = {acc1.x:.2f}\n"
            f"Mover Merah (massa={self.mover2.mass}): a = F/m = {acc2.x:.2f}\n\n"
            f"Kecepatan Biru: v = ({self.mover1.velocity.x:.2f},
{self.mover1.velocity.y:.2f})\n"
            f"Kecepatan Merah: v = ({self.mover2.velocity.x:.2f},
{self.mover2.velocity.y:.2f})"
        )
        self.info_text.config(text=info)

        # Lanjutkan animasi
        self.root.after(20, self.update)

# === JALANKAN PROGRAM ===
if __name__ == "__main__":
    root = tk.Tk()
    root.title("Simulasi Hukum Newton II (F = m * a)")
    app = Simulation(root)
    root.mainloop()
```

📊 **Hasil yang Ditampilkan**
Setiap frame, akan melihat:
Gaya angin (F): (0.10, 0.00)
Mover Biru (massa=1): a = F/m = 0.10
Mover Merah (massa=4): a = F/m = 0.03

Kecepatan Biru: v = (1.20, 0.00)
Kecepatan Merah: v = (0.30, 0.00)

📌 **Kesimpulan**

- Bola **biru** ringan → lebih cepat.
- Bola **merah** berat → lebih lambat.
- Gaya sama, tapi reaksi beda karena massa beda.
- Inilah **Hukum Newton II**: a=F/ma = F / m

Penjelasan contoh:

- Kita buat dua objek Mover, satu dengan massa kecil (1), satu massa besar (4).
- Kita beri gaya angin ke kanan yang sama untuk keduanya.
- Karena massa berbeda, percepatan yang didapat berbeda:
    - Massa kecil → percepatan besar → bergerak cepat
    - Massa besar → percepatan kecil → bergerak lambat
- Visualisasi lingkaran biru (ringan) bergerak lebih cepat dari lingkaran merah (berat).
- Ini contoh nyata hukum Newton II: **percepatan = gaya / massa**

2.3 Force Accumulation

Penjelasan Bab 2.3: Force Accumulation (Penjumlahan Gaya)
1. Apa itu Force Accumulation?

- Bayangkan kita punya banyak gaya yang bekerja pada sebuah benda sekaligus, misalnya **angin** dan **gravitasi**.
- Kalau kita pakai cara sederhana sebelumnya, di mana kita *langsung ganti* percepatan dengan gaya baru setiap kali applyForce() dipanggil, maka gaya sebelumnya akan hilang.
- Jadi kalau kita panggil applyForce(wind) lalu applyForce(gravity), percepatan akhirnya cuma yang terakhir saja (gravity), dan angin jadi tidak berpengaruh.

2. Bagaimana solusinya?

- Kita harus **menambahkan semua gaya** yang bekerja ke dalam satu variabel percepatan.

- Jadi percepatan = (jumlah semua gaya) / massa
- Ini disebut **Force Accumulation**, yaitu mengumpulkan semua gaya agar efeknya digabung.

3. Apa yang perlu diperhatikan?

- Karena percepatan hasil penjumlahan gaya itu hanya berlaku untuk satu frame (satu langkah waktu),
- Maka sebelum hitung gaya baru di frame berikutnya, kita harus **reset percepatan ke nol** dulu.
- Kalau tidak, gaya-gaya akan terus bertambah dan percepatan akan jadi sangat besar, tidak realistis.

4. Ringkas dalam kode

- applyForce(force) → *tambah* gaya ke percepatan, jangan ganti langsung.
- update() → hitung kecepatan dan posisi berdasarkan percepatan, lalu reset percepatan jadi nol.

Berikut ini adalah **simulasi sederhana** bola yang **mengakumulasi 2 gaya saja**:

- Gaya **gravitasi** ke bawah.
- Gaya **angin** ke kanan (aktif saat mouse ditekan).

Simulasi ini sangat cocok karena:

- Sederhana,
- Menjelaskan konsep **penjumlahan gaya (∑F)** dan
- Hubungannya dengan **percepatan dan gerakan**.

```
# SCRIPT 5 - Dua Gaya pada Benda (Simulasi Hukum Kedua Newton)
from vector import Vector

# === CLASS MOVER (BENDA) ===
class Mover:
    def __init__(self, mass, x, y):
        self.mass = mass
        self.position = Vector(x, y)
        self.velocity = Vector(0, 0)
        self.acceleration = Vector(0, 0)
        self.total_force = Vector(0, 0)

    def applyForce(self, force):
        # Gaya dijumlahkan dulu, belum diubah jadi percepatan
        self.total_force.add(force)

    def update(self):
        # 1. Hitung percepatan dari total gaya
        acc = self.total_force.copy()
        acc.div(self.mass)
        self.acceleration = acc

        # 2. Tambah percepatan ke kecepatan
        self.velocity.add(self.acceleration)

        # 3. Tambah kecepatan ke posisi
        self.position.add(self.velocity)

        # 4. Reset gaya total setelah update
        self.total_force = Vector(0, 0)

# === SIMULASI 10 LANGKAH ===
mover = Mover(mass=2, x=0, y=0)

# Gaya tetap
gravity = Vector(0, 0.1)  # ke bawah
wind = Vector(0.1, 0)     # ke kanan

print("Langkah | Posisi (x,y)    | Kecepatan (x,y) | Percepatan (x,y)")
```

```
print("-" * 60)

for step in range(1, 11):
    # Tambahkan gaya-gaya ke benda
    mover.applyForce(gravity)
    mover.applyForce(wind)

    # Update posisi benda
    mover.update()

    # Cetak hasil
    print(f"{step:>6} | {mover.position} | {mover.velocity} | {mover.acceleration}")
```

Langkah | Posisi (x,y) | Kecepatan (x,y) | Percepatan (x,y)
---
1 | (0.05, 0.05) | (0.05, 0.05) | (0.05, 0.05)
2 | (0.15, 0.15) | (0.10, 0.10) | (0.05, 0.05)
3 | (0.30, 0.30) | (0.15, 0.15) | (0.05, 0.05)
4 | (0.50, 0.50) | (0.20, 0.20) | (0.05, 0.05)
5 | (0.75, 0.75) | (0.25, 0.25) | (0.05, 0.05)……

Catatan Penjelasan:

- **applyForce()** menampung semua gaya dalam total_force.
- **update()** baru menghitung percepatan dari total gaya lalu memutakhirkan kecepatan dan posisi.
- Gaya gravitasi dan angin konstan menyebabkan percepatan konstan → kecepatan bertambah → posisi makin jauh.

Exercise 2. implementasi simulasi balon helium naik, memantul di atas jendela, dan ada gaya angin yang berubah-ubah.

Penjelasan Detail Latihan 2.1

Tujuan:

- Simulasikan **balon helium** yang bergerak **ke atas** (seperti mengapung).
- Balon harus **memantul saat menyentuh atas jendela** supaya tidak hilang.
- Tambahkan **gaya angin** yang berubah-ubah secara halus dari waktu ke waktu, supaya balon bergerak sedikit ke kiri-kanan.
- Gaya angin bisa dibuat mirip-mirip **Perlin noise** — tapi untuk anak SMP kita buat dengan fungsi sederhana, seperti nilai sinus yang berubah seiring waktu.

Tahapan Detail

1. Buat objek balon

- Balon punya properti posisi, kecepatan, percepatan, dan massa.
- Massa bisa kecil karena balon ringan.
- Posisi awal di bawah (bawah jendela).

2. Tambahkan gaya-gaya:

- **Gaya angkat helium**: gaya ke atas, nilai tetap kecil tapi positif.
- **Gaya angin**: gaya ke samping, nilainya berubah setiap frame.
- Gaya lain seperti gravitasi kita abaikan karena balon naik.

3. Update gerak balon:

- Percepatan = jumlah gaya / massa
- Kecepatan ditambah percepatan
- Posisi ditambah kecepatan

4. Cek tabrakan dengan atas jendela:

- Jika balon menyentuh atau melewati batas atas (y <= radius), maka kecepatan y dibalik (dipantulkan)
- Agar balon tidak "nempel" di atas, tempatkan posisi balon tepat di batas atas setelah pantulan.

5. Buat gaya angin berubah

- Gunakan fungsi sin yang nilai sudutnya bertambah setiap frame agar bergoyang halus ke kiri dan kanan.

Contoh Kode Python Tkinter untuk Simulasi Balon Helium

```
#SCRIPT 6 - Simulasi Balon Helium dengan Fisika dan Koordinat
import tkinter as tk
import math
from vector import Vector

class Balloon:
    def __init__(self, mass, x, y):
        self.mass = mass
        self.location = Vector(x, y)
        self.velocity = Vector(0, 0)
        self.acceleration = Vector(0, 0)
        self.radius = mass * 15
        self.forces = []

    def applyForce(self, force):
        self.forces.append(force.copy())
        f = force.copy()
        f.div(self.mass)
        self.acceleration.add(f)

    def update(self):
        self.velocity.add(self.acceleration)
        self.location.add(self.velocity)
        self.acceleration.mult(0)
        self.forces.clear()

    def checkEdges(self, height):
        if self.location.y - self.radius < 0:
            self.location.y = self.radius
            self.velocity.y *= -0.7

class Simulation:
    def __init__(self, root):
        self.root = root
        self.width = 400
        self.height = 600
        self.canvas = tk.Canvas(root, width=self.width, height=self.height,
bg="skyblue")
        self.canvas.pack()

        self.balloon = Balloon(mass=1, x=self.width//2, y=self.height-50)
        self.balloon_id = None
        self.coord_text_id = None
        self.frame_count = 0
        self.info_text = None

        self.setup_info_panel()
        self.create_balloon()
        self.update()

    def setup_info_panel(self):
        info_frame = tk.Frame(self.root)
        info_frame.pack(fill=tk.X, padx=10, pady=5)

        tk.Label(info_frame, text="Variabel Fisika:", font=('Arial', 10,
'bold')).pack(anchor='w')
```

```
        self.info_text = tk.Text(info_frame, height=10, width=50, font=('Courier', 9))
        self.info_text.pack()

        rumus = (
            "Rumus:\n"
            "1. Hukum II Newton: F = m × a\n"
            "2. Percepatan: a = F_total / m\n"
            "3. Kecepatan: v = v + a × Δt\n"
            "4. Posisi: pos = pos + v × Δt\n"
        )
        self.info_text.insert(tk.END, rumus)
        self.info_text.config(state=tk.DISABLED)

    def create_balloon(self):
        """Membuat objek balon pertama kali"""
        x, y, r = self.balloon.location.x, self.balloon.location.y, self.balloon.radius
        self.balloon_id = self.canvas.create_oval(
            x - r, y - r, x + r, y + r,
            fill="pink", outline="red", width=2
        )
        self.coord_text_id = self.canvas.create_text(
            x + r + 10, y, anchor="w", font=('Courier', 10),
            text=f"x={x:.1f}, y={y:.1f}"
        )

    def update_physics_info(self):
        self.info_text.config(state=tk.NORMAL)
        self.info_text.delete(1.0, tk.END)

        helium_force = Vector(0, -0.15)
        wind_strength = math.sin(self.frame_count * 0.05) * 0.1
        wind_force = Vector(wind_strength, 0)

        total_force = Vector(helium_force.x + wind_force.x,
                             helium_force.y + wind_force.y)
        acceleration = Vector(total_force.x / self.balloon.mass,
                              total_force.y / self.balloon.mass)

        self.info_text.insert(tk.END, "=== PERHITUNGAN FRAME ===\n\n")
        self.info_text.insert(tk.END, f"1. Gaya Helium: {helium_force}\n")
        self.info_text.insert(tk.END, f"2. Gaya Angin: {wind_force}\n")
        self.info_text.insert(tk.END, f"   (wind_strength =
sin({self.frame_count}×0.05)×0.1 = {wind_strength:.3f})\n")
        self.info_text.insert(tk.END, f"\n3. Total Gaya (F): {total_force}\n")
        self.info_text.insert(tk.END, f"4. Percepatan (a = F/m): {acceleration}\n")
        self.info_text.insert(tk.END, f"\n5. Kecepatan (v): {self.balloon.velocity}\n")
        self.info_text.insert(tk.END, f"6. Posisi: {self.balloon.location}
(x={self.balloon.location.x:.1f}, y={self.balloon.location.y:.1f})\n")

        if self.balloon.location.y - self.balloon.radius < 0:
            self.info_text.insert(tk.END, "\n7. BALON MENABRAK ATAS!\n")
            self.info_text.insert(tk.END, f"   Kecepatan setelah pantul:
{self.balloon.velocity}\n")

        self.info_text.config(state=tk.DISABLED)

    def update(self):
        # 1. Hitung gaya
        helium_force = Vector(0, -0.15)
        wind_strength = math.sin(self.frame_count * 0.05) * 0.1
        wind_force = Vector(wind_strength, 0)
        # 2. Terapkan gaya
```

```
        self.balloon.applyForce(helium_force)
        self.balloon.applyForce(wind_force)
        # 3. Update posisi dan kecepatan
        self.balloon.update()
        self.balloon.checkEdges(self.height)
        # 4. Update posisi objek di canvas
        x, y, r = self.balloon.location.x, self.balloon.location.y, self.balloon.radius
        self.canvas.coords(self.balloon_id, x - r, y - r, x + r, y + r)
        self.canvas.coords(self.coord_text_id, x + r + 10, y)
        self.canvas.itemconfig(self.coord_text_id, text=f"x={x:.1f}, y={y:.1f}")
        # 5. Update info panel
        self.update_physics_info()
        # 6. Loop
        self.frame_count += 1
        self.root.after(50, self.update)

if __name__ == "__main__":
    root = tk.Tk()
    root.title("Simulasi Balon Helium dengan Fisika & Koordinat (coords)")
    sim = Simulation(root)
    root.mainloop()
```

Proses Setiap Frame:

1. **Gaya Helium** (Vector(0, -0.15)):
   - Selalu mendorong balon **ke atas** (y negatif).
   - Contoh: Jika massa balon = 1 kg → percepatan = 0.15 m/s² ke atas.
2. **Gaya Angin** (math.sin(...) * 0.1):
   - Berubah pelan antara -0.1 sampai 0.1.
   - Membuat balon bergerak kiri-kanan secara alami.
3. **Update Posisi**:
   - Kecepatan diupdate: v = v + a.
   - Posisi diupdate: pos = pos + v.
4. **Pantulan di Atas**:
   - Jika balon mencapai y = 0 (tepi atas), kecepatan y dibalik dan dikurangi 30%.
   -

3. Rumus Utama yang Digunakan

1. **Hukum II Newton**:
   F_total = m × a
   a = F_total / m
2. **Perubahan Kecepatan**:
   v_baru = v_lama + a × Δt
   (Δt = interval waktu per frame)
3. **Perubahan Posisi**:
   pos_baru = pos_lama + v × Δt
4. **Gaya Angin**:
   wind_strength = sin(time × 0.05) × 0.1

4. Contoh Perhitungan Lengkap

Misal pada frame ke-5:

1. **Gaya Angin**:
   sin(5 × 0.05) × 0.1 = sin(0.25) × 0.1 ≈ 0.247 × 0.1 ≈ 0.025
2. **Total Gaya**:
   F_total = (0.025, -0.15) + (0, -0.15) = (0.025, -0.30)
3. **Percepatan (massa=1 kg)**:
   a = F_total / m = (0.025/1, -0.30/1) = (0.025, -0.30)
4. **Kecepatan Baru (misal v_lama=(0.4, -0.5))**:
   v_baru = (0.4 + 0.025, -0.5 + -0.30) = (0.425, -0.80)

5. **Posisi Baru (misal pos_lama=(200, 550))**:
   pos_baru = (200 + 0.425, 550 + -0.80) ≈ (200.425, 549.20)

Konsep Fisika dalam Simulasi

1. **Gaya Helium**:
   - Konstan ke atas (Vector(0, -0.15))
   - Menyebabkan percepatan vertikal
2. **Gaya Angin**:
   - Berubah pelan menggunakan sin(time)
   - Menyebabkan gerakan osilasi kiri-kanan
3. **Tumbukan**:
   - Saat balon mencapai y=0 (atas layar)
   - Kecepatan-y dibalik & dikurangi 30% (v_y *= -0.7)
4. **Massa**:
   - Massa balon mempengaruhi percepatan (a = F/m)
   - Massa besar = lebih sulit bergerak

2.4 Dealing with Mass

Penjelasan Materi 2.4 — Menggunakan Massa di Simulasi Gaya

1. Apa itu massa (mass)?

- Massa adalah ukuran berapa banyak "materi" yang ada dalam sebuah objek.
- Massa itu hanya angka (scalar), bukan vektor.
- Massa berbeda dengan ukuran. Misalnya bola kecil bisa lebih berat (massa besar) dari bola besar kalau bahan pembuatnya lebih padat.
- Dalam simulasi kita, massa bisa dibuat angka saja, misalnya 10.

2. Kenapa perlu massa?

- Newton bilang: **Gaya (force) = Massa × Percepatan**
- Jadi **Percepatan = Gaya / Massa**
- Ini berarti objek yang massanya besar akan lebih sulit bergerak (percepatan kecil kalau gaya sama).
- Massa mempengaruhi seberapa besar percepatan yang didapat jika dikenai gaya.

3. Bagaimana memasukkan massa di kode?

- Di kelas Mover kita tambahkan variabel mass yang menyimpan angka massa.
- Saat menerima gaya dari luar (applyForce), gaya tersebut harus dibagi dulu dengan massa sebelum ditambahkan ke percepatan.
- Ini mensimulasikan efek massa pada gerak objek.

4. Masalah penting saat membagi gaya dengan massa:

- Jika gaya (force) langsung dibagi massa tanpa membuat salinan (copy), maka gaya asli akan berubah.
- Akibatnya, kalau ada beberapa objek menerima gaya yang sama, gaya itu sudah berubah (berkurang) setelah dibagi massa objek pertama.
- Solusinya: buat **salinan gaya** sebelum membaginya dengan massa agar gaya asli tetap utuh untuk objek lain.

Contoh Implementasi Python Tkinter

```
# SCRIPT 7 - Simulasi Massa dan Gaya dengan Tkinter
import tkinter as tk
from vector import Vector
# ----------------------------
# Class Mover untuk benda bergerak
# ----------------------------
class Mover:
    def __init__(self, canvas, mass, x, y, color):
```

```
        self.canvas = canvas
        self.mass = mass  # Massa benda (kg)
        self.location = Vector(x, y)  # Posisi awal
        self.velocity = Vector(0, 0)  # Kecepatan awal
        self.acceleration = Vector(0, 0)  # Percepatan awal
        self.radius = mass * 3  # Ukuran bola proporsional dengan massa
        self.color = color

        # Gambar bola pertama kali di posisi awal
        x1, y1 = x - self.radius, y - self.radius
        x2, y2 = x + self.radius, y + self.radius
        self.body = canvas.create_oval(x1, y1, x2, y2, fill=color)

    def apply_force(self, force):
        """Menerapkan gaya ke benda (a = F/m)"""
        f = force.copy()         # Salin gaya
        f.div(self.mass)         # Hitung percepatan dari gaya: a = F/m
        self.acceleration.add(f) # Tambahkan percepatan ke benda

    def update(self):
        """Perbarui kecepatan dan posisi"""
        self.velocity.add(self.acceleration)  # v = v + a
        self.location.add(self.velocity)      # pos = pos + v
        self.acceleration = Vector(0, 0)      # Reset percepatan

        # Perbarui posisi bola menggunakan canvas.coords
        x, y, r = self.location.x, self.location.y, self.radius
        self.canvas.coords(self.body, x - r, y - r, x + r, y + r)

    def get_info(self):
        """Mengembalikan info perhitungan untuk ditampilkan"""
        return (
            f"Mass: {self.mass} kg\n"
            f"Gaya: (1.00, 0.00) N\n"
            f"Percepatan (a = F/m): {Vector(1,0).copy().div(self.mass) or
Vector(1,0)}\n"
            f"Kecepatan: {self.velocity}\n"
            f"Posisi: {self.location}\n"
        )

# ----------------------------
# Class Simulation
# ----------------------------
class Simulation:
    def __init__(self, root):
        self.root = root
        self.width = 500
        self.height = 400

        # Canvas utama untuk menggambar bola
        self.canvas = tk.Canvas(root, width=self.width, height=self.height, bg="white")
        self.canvas.pack()

        # Label teks di bawah canvas untuk perhitungan fisika
        self.info_label = tk.Label(root, text="", font=("Courier", 10), justify="left",
anchor="w")
        self.info_label.pack(padx=10, anchor="w")

        # Inisialisasi dua bola dengan massa berbeda
        self.mover1 = Mover(self.canvas, mass=5, x=100, y=200, color="blue")
        self.mover2 = Mover(self.canvas, mass=15, x=100, y=100, color="red")
```

```
        # Gaya konstan ke kanan (angin)
        self.wind = Vector(1, 0)

        # Jalankan simulasi
        self.update()

    def update(self):
        # Terapkan gaya angin ke masing-masing bola
        self.mover1.apply_force(self.wind)
        self.mover2.apply_force(self.wind)

        # Perbarui posisi dan kecepatan masing-masing bola
        self.mover1.update()
        self.mover2.update()

        # Buat teks gabungan untuk kedua benda
        info_text = "--- Bola Biru (massa kecil) ---\n"
        info_text += self.mover1.get_info()
        info_text += "\n--- Bola Merah (massa besar) ---\n"
        info_text += self.mover2.get_info()

        # Tampilkan teks di bawah canvas
        self.info_label.config(text=info_text)

        # Lanjutkan animasi tiap 30 ms
        self.root.after(30, self.update)

# ----------------------------
# Jalankan program utama
# ----------------------------
if __name__ == "__main__":
    root = tk.Tk()
    root.title("Simulasi Massa dan Gaya (Newton II) dengan Tkinter")
    sim = Simulation(root)
    root.mainloop()
```

Penjelasan Kode:

- Kita buat dua objek Mover dengan massa berbeda: 5 dan 15.
- Gaya angin ke kanan sama-sama diberikan ke kedua objek.
- Karena massa berbeda, percepatan yang didapat berbeda:
    - Massa kecil → percepatan besar → bergerak cepat.
    - Massa besar → percepatan kecil → bergerak lebih lambat.
- Jadi kamu bisa lihat si biru (massa kecil) bergerak lebih cepat ke kanan daripada si merah.

Rumus dan Perhitungan

Hukum II Newton:

$F = m \times a \rightarrow a = F / m$

Contoh Perhitungan Frame Pertama:

1. **Objek Biru (5kg)**:
    - Gaya: (1, 0)
    - Percepatan: (1/5, 0/5) = (0.20, 0.00)
    - Kecepatan: (0 + 0.20, 0 + 0) = (0.20, 0.00)
    - Posisi: (100 + 0.20, 200 + 0) = (100.20, 200.00)
2. **Objek Merah (15kg)**:
    - Gaya: (1, 0)
    - Percepatan: (1/15, 0/15) ≈ (0.07, 0.00)

- Kecepatan: (0 + 0.07, 0 + 0) ≈ (0.07, 0.00)
- Posisi: (100 + 0.07, 300 + 0) ≈ (100.07, 300.00)

4. Hasil yang Diamati
   - **Objek dengan massa lebih kecil** (biru):
     - Percepatan lebih besar (0.20 m/s² vs 0.07 m/s²)
     - Bergerak lebih cepat ke kanan
     - Menempuh jarak lebih jauh dalam waktu sama
   - **Objek dengan massa lebih besar** (merah):
     - Percepatan lebih kecil
     - Bergerak lebih lambat
     - Butuh waktu lebih lama untuk menempuh jarak sama
5. Konsep Fisika yang Ditunjukkan
   1. **Hubungan massa-percepatan**:
      - Massa berbanding terbalik dengan percepatan untuk gaya yang sama
      - a = F/m → massa besar = percepatan kecil
   2. **Gerak Lurus Berubah Beraturan**:
      - Kecepatan bertambah secara konstan oleh percepatan
      - Posisi berubah sesuai integral kecepatan
   3. **Visualisasi Inersia**:
      - Benda bermassa besar lebih "malas" bergerak (inersia besar)
      - Benda bermassa kecil lebih mudah dipercepat

Static Method dan applyForce() dengan Static Method div()

Penjelasan Materi: Static Method dan applyForce() dengan Static Method div()

1. Apa itu static method?

- Static method adalah fungsi yang milik kelas, bukan milik objek.
- Jadi, kita bisa panggil method ini tanpa harus buat objek dulu.
- Di materi Processing, PVector.div() bisa dipakai sebagai static method untuk membagi vector dengan scalar dan menghasilkan vector baru.

2. Kenapa pakai static method div()?

- Sebelumnya kita buat salinan vector force dulu dengan .get(), baru dibagi dengan massa.
- Dengan static method div(), kita bisa langsung buat vector baru hasil pembagian dari vector lama, tanpa merubah vector asal.
- Ini membantu menjaga force asli tetap utuh, jadi bisa dipakai lagi untuk objek lain.

3. Bentuk umum static method div():

```
PVector f = PVector.div(force, mass);
```

- Ini artinya: buat vector baru f = force dibagi mass.

4. Jadi fungsi applyForce() versi static method jadi:

```
void applyForce(PVector force) {
  PVector f = PVector.div(force, mass);
  acceleration.add(f);
}
```

Apa yang Simulasi Ini Lakukan?

Program ini menunjukkan dua bola (biru dan merah) yang didorong angin:

- Bola biru (massa 5kg) bergerak lebih cepat
- Bola merah (massa 15kg) bergerak lebih lambat

Ini membuktikan **Hukum Newton II**: *"Percepatan benda berbanding lurus dengan gaya dan berbanding terbalik dengan massanya"* (F = m × a)

Perbedaan dengan Konsep Sebelumnya

Konsep Sebelumnya	Konsep Sekarang (Static Method)
Method div() ada di dalam class Vector dan mengubah vektor asli	Method div() adalah static, tidak mengubah vektor asli tetapi membuat vektor baru
Cara pakai: force.div(mass)	Cara pakai: Vector.div(force, mass)
Lebih sederhana	Lebih aman karena tidak mengubah data asli

Contoh Implementasi di Python Tkinter (Konsep Static Method div())

```
# SCRIPT 8 - Simulasi Hukum Kedua Newton (F = m * a) dengan Static Method
import tkinter as tk
from vector import Vector

# ----------------------------
# Class Mover untuk benda yang bergerak
# ----------------------------
class Mover:
    def __init__(self, canvas, mass, x, y, color):
        self.canvas = canvas
        self.mass = mass  # Massa benda (kg)
        self.location = Vector(x, y)  # Posisi awal
        self.velocity = Vector(0, 0)  # Kecepatan awal
        self.acceleration = Vector(0, 0)  # Percepatan awal
        self.radius = mass * 3  # Ukuran bola tergantung massa
        self.color = color

        # Gambar awal bola di canvas dan simpan ID-nya
        r = self.radius
        self.body = canvas.create_oval(x - r, y - r, x + r, y + r, fill=color)

    def apply_force(self, force):
        """Menerapkan gaya ke benda menggunakan static method div"""
        f = Vector.dived(force, self.mass)  # Hitung percepatan: a = F/m
        self.acceleration.add(f)  # Tambahkan percepatan

    def update(self):
        """Update posisi dan kecepatan benda"""
        self.velocity.add(self.acceleration)  # v = v + a
        self.location.add(self.velocity)      # pos = pos + v
        self.acceleration = Vector(0, 0)      # Reset percepatan

        # Perbarui posisi bola dengan canvas.coords (tidak menggambar ulang)
        x, y, r = self.location.x, self.location.y, self.radius
        self.canvas.coords(self.body, x - r, y - r, x + r, y + r)

    def get_info(self):
        """Mengembalikan info perhitungan untuk ditampilkan"""
        return (
            f"Mass: {self.mass} kg\n"
            f"Gaya: (1.00, 0.00) N\n"
            f"Percepatan (a = F/m): {Vector(1,0).copy().div(self.mass) or
Vector(1,0)}\n"
            f"Kecepatan: {self.velocity}\n"
            f"Posisi: {self.location}\n"
        )

# ----------------------------
# Class Simulation untuk menjalankan animasi
# ----------------------------
```

```
class Simulation:
    def __init__(self, root):
        self.root = root
        self.width = 400
        self.height = 400

        # Canvas utama untuk menggambar bola
        self.canvas = tk.Canvas(root, width=self.width, height=self.height, bg="white")
        self.canvas.pack()

        # Label teks di bawah canvas untuk perhitungan fisika
        self.info_label = tk.Label(root, text="", font=("Courier", 10), justify="left",
anchor="w")
        self.info_label.pack(padx=10, anchor="w")

        # Buat dua bola dengan massa berbeda
        self.mover1 = Mover(self.canvas, mass=5, x=100, y=200, color="blue")   # Bola
biru, massa kecil
        self.mover2 = Mover(self.canvas, mass=15, x=100, y=300, color="red")   # Bola
merah, massa besar

        # Gaya angin ke kanan sebesar 1 Newton
        self.wind = Vector(1, 0)

        # Mulai animasi
        self.update()

    def update(self):
        # Terapkan gaya ke kedua bola
        self.mover1.apply_force(self.wind)
        self.mover2.apply_force(self.wind)

        # Perbarui posisi kedua bola
        self.mover1.update()
        self.mover2.update()

        # Buat teks gabungan untuk kedua benda
        info_text = "--- Bola Biru (massa kecil) ---\n"
        info_text += self.mover1.get_info()
        info_text += "\n--- Bola Merah (massa besar) ---\n"
        info_text += self.mover2.get_info()

        # Tampilkan teks di bawah canvas
        self.info_label.config(text=info_text)

        # Jadwalkan frame berikutnya setelah 30 ms
        self.root.after(30, self.update)

# ----------------------------
# Jalankan program utama
# ----------------------------
if __name__ == "__main__":
    root = tk.Tk()
    root.title("Simulasi Hukum Newton II (F = m * a) dengan Static Method")
    sim = Simulation(root)
    root.mainloop()
```

3. Alur Program dan Perhitungan Fisika
Langkah-langkah Simulasi

1. **Buat Dua Bola**:
    - Bola biru (5kg) di kiri
    - Bola merah (15kg) di kanan
2. **Terapkan Gaya Angin** (1 Newton ke kanan):

```
wind = Vector(1, 0)  # Gaya 1 Newton ke kanan
```

3. **Hitung Percepatan** (a = F/m):
    - Bola biru: a = 1N / 5kg = 0.2 m/s²
    - Bola merah: a = 1N / 15kg ≈ 0.067 m/s²
4. **Update Kecepatan & Posisi**:

```
velocity.x += acceleration.x  # Tambahkan percepatan ke kecepatan
location.x += velocity.x      # Gerakkan benda
```

5. **Hasil**:
    - Bola biru bergerak **3× lebih cepat** daripada bola merah
    - Sesuai rumus: a = F/m → massa lebih besar = percepatan lebih kecil

---

4. Contoh Tampilan Output
Setelah 5 detik:

- Bola biru (5kg):
    - Posisi x: 160 pixel
    - Kecepatan: 1.0 pixel/frame
- Bola merah (15kg):
    - Posisi x: 120 pixel
    - Kecepatan: 0.33 pixel/frame

Kenapa Beda?

- Gaya sama (1N), tapi massa berbeda:

```
a_biru = 1N / 5kg = 0.2 m/s²
a_merah = 1N / 15kg ≈ 0.067 m/s²
```

- Percepatan biru 3× lebih besar → lebih cepat!

---

5. Keuntungan Static Method div()

1. **Tidak Mengubah Data Asli**:

```
# Sebelum (ubah vektor asli):
force.div(mass)  # force berubah

# Sekarang (buat vektor baru):
new_vector = Vector.div(force, mass)  # force tetap
```

2. **Lebih Jelas**:
    - Langsung terlihat bahwa kita membuat vektor baru
    - Tidak ada efek samping yang tak terduga
3. **Aman dari Pembagian Nol**:
    - Ada pengecekan if scalar != 0

---

6. Analogi Sederhana
Bayangkan dua mobil mainan:

- Mobil biru (5kg) dan mobil merah (15kg)
- Kita dorong dengan gaya sama (1N)
- **Hasil**:
    - Mobil biru meluncur cepat (massa kecil)
    - Mobil merah bergerak lambat (massa besar)

Static Method seperti **teman pembantu** yang:

1. Ambil mobil asli
2. Hitung kecepatan baru

3. Berikan mobil baru yang sudah dihitung
4. Mobil asli tetap utuh tidak berubah

Kesimpulan sederhana:

- Kita buat apply_force() yang gak merubah gaya asli, tapi membuat gaya baru yang sudah dibagi massa.
- Kita pakai static method supaya lebih rapi dan aman.
- Hasilnya, dua objek dengan massa berbeda bergerak berbeda meskipun kena gaya yang sama.

2.5 - Creating Forces

🧠 Konsep Fisika yang Sederhana

Apa itu Gaya (Force)?

Gaya adalah sesuatu yang bisa membuat benda bergerak, berhenti, atau berubah arah.

Contoh gaya sehari-hari:

- Angin mendorong layang-layang (gaya angin)
- Bola jatuh karena gravitasi (gaya gravitasi)
- Dorongan tangan (gaya manual)

Dalam fisika, gaya adalah **vektor**—artinya punya arah dan besar.

🎮 Tujuan Kita

- Menambahkan gaya ke objek
- Objek merespons gaya sesuai massanya
- Banyak objek, masing-masing punya massa dan lokasi awal berbeda
- Gaya bisa: angin, gravitasi, atau gaya buatan

💡 Inti Kode dalam Python (versi sederhana dengan Tkinter)

Kita akan menggunakan:

- Vector: untuk menyimpan posisi, kecepatan, percepatan
- Mover: benda yang bisa bergerak
- apply_force: fungsi untuk menambahkan gaya (mengikuti hukum Newton F = m × a)

🎓 Konsep yang Dipelajari

1. **Gaya (Force)**: sesuatu yang mendorong atau menarik.
2. **Percepatan (Acceleration)**: bagaimana kecepatan berubah.
3. **Kecepatan (Velocity)**: seberapa cepat dan arah bola bergerak.
4. **Posisi (Position)**: di mana bola berada.
5. **Hukum Newton ke-2**: $\mathbf{F} = m \cdot \mathbf{a} \Rightarrow \mathbf{a} = \frac{\mathbf{F}}{m}$

💡 Penjelasan Langkah Demi Langkah

Tahap	Penjelasan	Contoh
1. Gaya diberikan	Gravitasi selalu bekerja. Angin hanya saat mouse ditekan.	Vector(0, 0.3) dan Vector(0.1, 0)
2. Gaya dibagi massa → percepatan	a=Fma = \frac{F}{m}a=mF	Jika Fy=0.3F_y = 0.3Fy=0.3 dan m=2m = 2m=2 → ay=0.15a_y = 0.15ay=0.15
3. Update kecepatan	v=v+av = v + av=v+a	Kecepatan terus bertambah tiap frame
4. Update posisi	p=p+vp = p + vp=p+v	Bola turun perlahan lebih cepat
5. Reset percepatan	Supaya gaya baru bisa diterapkan di frame selanjutnya	self.acceleration.mult(0)

🧪 Contoh Simulasi dengan Tkinter

```
# SCRIPT 9 - Simulasi gaya dengan variabel angin dan massa
import tkinter as tk
from vector import Vector

# === CLASS MOVER UNTUK BENDA BERGERAK ===
class Mover:
    def __init__(self, mass, x, y):
        self.mass = mass                              # Massa benda (kg)
        self.position = Vector(x, y)                  # Posisi awal (x, y)
        self.velocity = Vector(0, 0)                  # Kecepatan awal (diam)
        self.acceleration = Vector(0, 0)              # Percepatan awal nol
        self.radius = mass * 10                       # Ukuran bola sebanding dengan
massa

    def apply_force(self, force):
        """Menerapkan gaya ke benda: a = F / m"""
        a = Vector.div(force, self.mass)              # Hitung percepatan dari gaya
        self.acceleration.add(a)                      # Tambahkan ke total percepatan

    def update(self):
        """Update posisi berdasarkan percepatan dan kecepatan"""
        self.velocity.add(self.acceleration)          # v = v + a
        self.position.add(self.velocity)              # s = s + v
        self.acceleration = Vector(0, 0)              # Reset percepatan

    def check_edges(self, height):
        """Pantulkan jika menyentuh dasar canvas"""
        if self.position.y > height - self.radius:
            self.position.y = height - self.radius
            self.velocity.y *= -1                     # Pantulan: arah kecepatan
vertikal dibalik

# === CLASS UTAMA UNTUK SIMULASI ===
class Simulation:
    def __init__(self, root):
        self.root = root
        self.width = 600
        self.height = 400

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

        # Membuat objek bola
        self.mover = Mover(mass=2, x=100, y=50)

        # Gaya tetap
        self.gravity = Vector(0, 0.3)      # Gravitasi ke bawah
        self.wind = Vector(0.1, 0)         # Angin ke kanan

        self.mouse_pressed = False         # Status mouse ditekan

        # Event mouse
        self.root.bind("<ButtonPress-1>", self.on_mouse_down)
        self.root.bind("<ButtonRelease-1>", self.on_mouse_up)

        # Gambar bola pertama kali menggunakan create_oval
        r = self.mover.radius
        x, y = self.mover.position.x, self.mover.position.y
        self.ball = self.canvas.create_oval(x - r, y - r, x + r, y + r, fill="skyblue")

        # Label informasi
```

```
        self.info_label = tk.Label(root, font=("Courier", 10), justify="left",
bg="white", anchor="w")
        self.info_label.pack(fill="both")

        # Mulai update loop
        self.update()

    def on_mouse_down(self, event):
        """Deteksi ketika mouse diklik"""
        self.mouse_pressed = True

    def on_mouse_up(self, event):
        """Deteksi ketika mouse dilepas"""
        self.mouse_pressed = False

    def update(self):
        # === 1. Hitung gaya total ===
        total_force = self.gravity.copy()         # Awalnya hanya gravitasi
        if self.mouse_pressed:
            total_force.add(self.wind)            # Tambahkan angin jika mouse ditekan

        # === 2. Terapkan gaya ke benda ===
        self.mover.apply_force(total_force)

        # === 3. Update posisi, kecepatan, dan cek pantulan ===
        self.mover.update()
        self.mover.check_edges(self.height)

        # === 4. Update posisi bola di canvas menggunakan canvas.coords ===
        r = self.mover.radius
        x = self.mover.position.x
        y = self.mover.position.y
        self.canvas.coords(self.ball, x - r, y - r, x + r, y + r)  # Menggeser bola ke
posisi baru

        # === 5. Hitung percepatan (a = F / m) untuk ditampilkan ===
        fx = total_force.x
        fy = total_force.y
        mass = self.mover.mass
        ax = fx / mass
        ay = fy / mass

        # === 6. Tampilkan informasi fisika di label ===
        info = f"""
Rumus: a = F / m

Gaya total:
  Fx = {fx:.2f}, Fy = {fy:.2f}
Percepatan:
  ax = Fx/m = {fx:.2f}/{mass} = {ax:.2f}
  ay = Fy/m = {fy:.2f}/{mass} = {ay:.2f}
Kecepatan:
  vx = {self.mover.velocity.x:.2f}
  vy = {self.mover.velocity.y:.2f}
Posisi:
  x = {x:.2f}
  y = {y:.2f}

Klik kiri untuk menambah gaya angin ke kanan.
"""
        self.info_label.config(text=info)
```

```
        # === 7. Ulangi update setelah 30ms ===
        self.root.after(30, self.update)

# === JALANKAN PROGRAM ===
if __name__ == "__main__":
    root = tk.Tk()
    root.title("Simulasi Gaya dan Percepatan (F = m × a)")
    Simulation(root)
    root.mainloop()
```

📘 **Penjelasan**

- Tiap bulatan punya “massa” → makin berat, makin lambat ia bergerak kalau kena gaya.
- Gaya “angin” mendorong ke kanan → semua bulatan bergerak ke kanan pelan-pelan.
- Gaya “gravitasi” menarik ke bawah → makin berat, makin kuat ditarik ke bawah.
- Setiap frame (30 milidetik), kita update posisi semua bulatan.

🧠 **Kesimpulan**

- **Gaya** membuat bola **berubah arah atau kecepatannya**.
- Kita bisa menambahkan banyak gaya → hasilnya digabung.
- **Percepatan = Gaya ÷ Massa**
- Semakin besar massanya, semakin kecil efek dari gaya itu.
- **Simulasi ini mengikuti hukum Newton**: F = m × a

Bagus! Sekarang kita akan melanjutkan **penjelasan Example 2.2: "Forces acting on many objects"** dari buku *The Nature of Code*, dan **mengubahnya ke Python Tkinter**

🎯 **Tujuan Utama**

- Banyak objek (Mover) memiliki massa berbeda.
- Semua objek diberi gaya yang **sama**: angin dan gravitasi.
- Gaya tersebut diterapkan ke **masing-masing objek**, dan responsnya **berbeda** karena massanya berbeda.
- Objek yang ringan akan **bergerak lebih cepat**, karena gaya dibagi dengan massa (hukum Newton kedua).

📘 **Penjelasan Konsep Sederhana**

Hukum Newton II:
"Semakin berat benda, semakin lambat ia akan bergerak jika diberi gaya yang sama."
Misalnya:

- Dorong bola pingpong dan bola besi dengan kekuatan yang sama.
- Bola pingpong akan bergerak jauh lebih cepat → karena massanya kecil.

📦 **Struktur Program Python Tkinter**

```
# SCRIPT 10 - Simulasi gaya dengan variabel angin dan massa utk 4 bola
import tkinter as tk
from vector import Vector

# === CLASS MOVER UNTUK OBJEK BOLA BERGERAK ===
class Mover:
    def __init__(self, mass, x, y, color="skyblue"):
        self.mass = mass                            # Massa bola
        self.position = Vector(x, y)                # Posisi awal bola
        self.velocity = Vector(0, 0)                # Kecepatan awal (vx = 0, vy = 0)
        self.acceleration = Vector(0, 0)            # Percepatan awal
        self.radius = mass * 10                     # Radius bola bergantung pada massa
```

```
        self.color = color                          # Warna bola
        self.id = None                              # ID bola di canvas (diset nanti)

    def apply_force(self, force):
        # Hitung percepatan: a = F / m dan tambahkan ke total percepatan
        a = Vector.div(force, self.mass)
        self.acceleration.add(a)

    def update(self):
        # Update kecepatan dan posisi, lalu reset percepatan
        self.velocity.add(self.acceleration)
        self.position.add(self.velocity)
        self.acceleration = Vector(0, 0)

    def check_edges(self, height):
        # Jika bola jatuh melewati bawah canvas, pantulkan ke atas
        if self.position.y > height - self.radius:
            self.position.y = height - self.radius
            self.velocity.y *= -1

# === CLASS SIMULASI UTAMA ===
class Simulation:
    def __init__(self, root):
        self.root = root
        self.width = 800
        self.height = 500
        self.canvas = tk.Canvas(root, width=self.width, height=self.height, bg="white")
        self.canvas.pack()

        # Buat 4 bola dengan massa berbeda
        self.movers = [
            Mover(1, 100, 50, "red"),
            Mover(2, 250, 50, "green"),
            Mover(3, 400, 50, "blue"),
            Mover(4, 550, 50, "orange"),
        ]

        # Gaya konstan
        self.gravity = Vector(0, 0.3)  # Gaya gravitasi ke bawah
        self.wind = Vector(0.1, 0)     # Gaya angin ke kanan
        self.mouse_pressed = False     # Status mouse ditekan

        # Buat bola-bola di canvas dan simpan ID-nya
        for mover in self.movers:
            r = mover.radius
            x, y = mover.position.x, mover.position.y
            mover.id = self.canvas.create_oval(x - r, y - r, x + r, y + r,
fill=mover.color)

        # Event mouse untuk angin
        self.root.bind("<ButtonPress-1>", self.on_mouse_down)
        self.root.bind("<ButtonRelease-1>", self.on_mouse_up)

        self.update()  # Jalankan animasi

    def on_mouse_down(self, event):
        self.mouse_pressed = True  # Ketika mouse ditekan, angin aktif

    def on_mouse_up(self, event):
        self.mouse_pressed = False  # Saat mouse dilepas, angin berhenti

    def update(self):
```

```
        self.canvas.delete("text")  # Hapus teks sebelumnya (bukan bola)

        text_y = 20  # Posisi awal teks keterangan

        for mover in self.movers:
            # === 1. Hitung gaya total ===
            total_force = self.gravity.copy()
            if self.mouse_pressed:
                total_force.add(self.wind)

            # === 2. Terapkan gaya, update posisi dan periksa tepi ===
            mover.apply_force(total_force)
            mover.update()
            mover.check_edges(self.height)

            # === 3. Update posisi bola menggunakan canvas.coords ===
            x, y, r = mover.position.x, mover.position.y, mover.radius
            self.canvas.coords(mover.id, x - r, y - r, x + r, y + r)

            # === 4. Hitung dan tampilkan informasi gaya, a, v, posisi ===
            fx, fy = total_force.x, total_force.y
            mass = mover.mass
            ax, ay = fx / mass, fy / mass

            info = (
                f"Bola mass={mass} | "
                f"F: ({fx:.2f}, {fy:.2f}) | "
                f"a: ({ax:.2f}, {ay:.2f}) | "
                f"v: ({mover.velocity.x:.2f}, {mover.velocity.y:.2f}) | "
                f"pos: ({x:.2f}, {y:.2f})"
            )
            # Gambar teks info di bawah canvas
            self.canvas.create_text(400, text_y, text=info, font=("Courier", 10),
anchor="n", tags="text")
            text_y += 18

        # === 5. Teks petunjuk pengguna ===
        self.canvas.create_text(
            400, self.height - 20,
            text="Klik kiri mouse untuk menambah gaya angin ke kanan (wind)",
            font=("Arial", 10), fill="black", tags="text"
        )

        # === 6. Looping animasi setiap 30 ms ===
        self.root.after(30, self.update)

# === JALANKAN PROGRAM ===
if __name__ == "__main__":
    root = tk.Tk()
    root.title("Simulasi Gaya dan Gerakan - 4 Bola (canvas.coords)")
    Simulation(root)
    root.mainloop()
```

🎯 **Tujuan Exercise 2.3**

Alih-alih membiarkan objek **mental saat menabrak tepi jendela**, kita ingin:

1. Membuat **gaya tak terlihat** (invisible force) yang mendorong objek **kembali ke tengah layar**.
2. Gaya ini akan **semakin kuat** jika objek **semakin dekat ke tepi**.

🧠 **Penjelasan Konsep**

Bayangkan kamu bermain trampolin:

- Saat kamu jauh dari pinggir, tidak ada yang terjadi.
- Tapi saat kamu hampir jatuh ke luar, ada tangan tak terlihat (gaya) yang **mendorongmu kembali ke tengah**.
- **Semakin dekat ke tepi, semakin besar dorongannya!**

📘 Rumus Gaya "Push Back"

Kita akan buat gaya ini bekerja seperti "pegas imajiner":

- Jika jarak ke tepi < batas tertentu (misal 50px), kita hitung gaya:
- force = (batas - jarak_ke_tepi) * kekuatan

Gaya ini akan:

- Menekan ke **kanan** jika dekat **kiri**.
- Menekan ke **kiri** jika dekat **kanan**.
- Menekan ke **bawah** jika dekat **atas**.
- Menekan ke **atas** jika dekat **bawah**.

📦 Update Program Python Tkinter

```
# SCRIPT 11 - Simulasi dengan invisible edge push (revisi dengan canvas.coords)
import tkinter as tk
from vector import Vector
# =====================
# Kelas Bola (objek fisik)
class Mover:
    def __init__(self, mass, x, y, color="skyblue"):
        self.mass = mass
        self.position = Vector(x, y)             # Posisi awal
        self.velocity = Vector(0, 0)             # Kecepatan awal
        self.acceleration = Vector(0, 0)         # Percepatan awal
        self.radius = mass * 10                  # Radius berdasarkan massa
        self.color = color                       # Warna bola

    def apply_force(self, force):
        # Menghitung percepatan: a = F / m
        a = Vector.dived(force, self.mass)
        self.acceleration.add(a)

    def apply_edge_push(self, width, height, margin=50, strength=0.05):
        # Menambahkan gaya dorong jika bola terlalu dekat tepi
        if self.position.x < margin:
            push = Vector((margin - self.position.x) * strength, 0)
            self.apply_force(push)
        if self.position.x > width - margin:
            push = Vector((width - margin - self.position.x) * strength, 0)
            self.apply_force(push)
        if self.position.y < margin:
            push = Vector(0, (margin - self.position.y) * strength)
            self.apply_force(push)
        if self.position.y > height - margin:
            push = Vector(0, (height - margin - self.position.y) * strength)
            self.apply_force(push)

    def update(self):
        # Update kecepatan dan posisi berdasarkan percepatan
        self.velocity.add(self.acceleration)
        self.position.add(self.velocity)
        self.acceleration = Vector(0, 0)  # Reset percepatan setiap frame

# =====================
# Kelas Simulasi
class Simulation:
```

```
    def __init__(self, root):
        self.width = 600
        self.height = 400
        self.canvas = tk.Canvas(root, width=self.width, height=self.height, bg="white")
        self.canvas.pack()

        # Membuat bola
        self.mover = Mover(2, 100, 300, color="orange")

        # Gaya gravitasi dan angin
        self.gravity = Vector(0, 0.2)
        self.wind = Vector(0.05, 0)

        # ID untuk objek canvas
        self.ball_id = None
        self.text_id = None
        self.instruction_id = None

        self.draw()  # Mulai loop animasi

    def draw(self):
        # Reset gaya total setiap frame
        total_force = Vector()

        # Terapkan gaya gravitasi
        self.mover.apply_force(self.gravity)
        total_force.add(self.gravity)

        # Terapkan gaya angin
        self.mover.apply_force(self.wind)
        total_force.add(self.wind)

        # Terapkan dorongan jika dekat tepi
        self.mover.apply_edge_push(self.width, self.height)

        # Update posisi dan kecepatan
        self.mover.update()

        # Ambil posisi dan radius bola
        x, y = self.mover.position.x, self.mover.position.y
        r = self.mover.radius

        # Gambar bola atau update posisi menggunakan canvas.coords
        if self.ball_id is None:
            self.ball_id = self.canvas.create_oval(x - r, y - r, x + r, y + r,
fill=self.mover.color)
        else:
            self.canvas.coords(self.ball_id, x - r, y - r, x + r, y + r)  # Update
posisi bola

        # Hitung data fisika
        fx, fy = total_force.x, total_force.y
        ax = fx / self.mover.mass
        ay = fy / self.mover.mass

        info_text = (
            f"Rumus: a = F / m\n"
            f"Ftotal = ({fx:.2f}, {fy:.2f})\n"
            f"Percepatan = ({ax:.2f}, {ay:.2f})\n"
            f"Kecepatan = ({self.mover.velocity.x:.2f}, {self.mover.velocity.y:.2f})\n"
            f"Posisi = ({x:.2f}, {y:.2f})"
        )
```

```
        # Gambar atau update teks info fisika
        if self.text_id is None:
            self.text_id = self.canvas.create_text(10, 10, anchor="nw",
font=("Courier", 12), text=info_text)
        else:
            self.canvas.itemconfig(self.text_id, text=info_text)

        # Gambar atau update instruksi (gunakan coords untuk jaga posisinya tetap)
        if self.instruction_id is None:
            self.instruction_id = self.canvas.create_text(
                self.width / 2, self.height - 20,
                text="Dorongan tepi akan muncul jika bola terlalu dekat tepi",
                font=("Arial", 10), fill="gray"
            )
        else:
            self.canvas.coords(self.instruction_id, self.width / 2, self.height - 20)

        # Loop animasi setiap 30 ms
        self.canvas.after(30, self.draw)

# ======================
# Jalankan program
if __name__ == "__main__":
    root = tk.Tk()
    root.title("Simulasi Gaya Sederhana - 1 Bola dengan Edge Push")
    Simulation(root)
    root.mainloop()
```

Konsep	Penjelasan
Gaya	Mendorong bola untuk bergerak
Percepatan	Dihitung dari gaya dibagi massa
Kecepatan	Bertambah setiap frame dari percepatan
Posisi	Diperbarui dari kecepatan
Push-Back	Jika bola dekat tepi, seolah ada **pegas yang mendorong balik**
Rumus	gaya = (batas - jarak) * kekuatan

2.6: Gravity on Earth and Modeling a Force

🎯 Tujuan Materi

Memperbaiki simulasi agar **lebih realistis secara fisika**:

- Dalam contoh sebelumnya, objek kecil **jatuh lebih cepat** (karena massa kecil → percepatan besar).
- Tapi di dunia nyata, semua benda (besar/kecil) **jatuh dengan kecepatan yang sama** jika tidak ada hambatan udara.

🧠 Penjelasan Sederhana

Bayangkan kamu menjatuhkan bola tenis dan bola basket dari lantai dua. Kira-kira, **jatuhnya barengan gak?**
✅ Iya, karena **gaya gravitasi bekerja lebih besar ke benda besar**, tapi **dibagi massa juga**, jadi hasil akhirnya **sama-sama cepatnya**.

📘 Rumus Gaya Gravitasi yang Realistis

Gaya gravitasi:

📘 Rumus Gaya Gravitasi yang Realistis

Gaya gravitasi:

$$\text{force} = mass \times g$$

> Artinya: benda besar ditarik lebih kuat, tapi saat menghitung percepatan:

$$\text{acceleration} = \frac{\text{force}}{\text{mass}} = \frac{mass \times g}{mass} = g$$

Jadi massanya **batal sendiri** → percepatan semua benda **sama**.

🎯 Tujuan Simulasi

Menjelaskan bahwa:

- Gaya gravitasi mempengaruhi benda tergantung massanya.
- Semakin besar massa, semakin besar pula gaya gravitasinya.
- Tapi... percepatan (a = F / m) tetap sama! Jadi, dua benda jatuh dengan kecepatan yang sama!

📦 Tahapan Simulasi

1. **Class Vector**: Untuk menyimpan dan mengolah vektor (posisi, kecepatan, percepatan, gaya).
2. **Class Mover**: Untuk setiap benda, memiliki massa, posisi, kecepatan, dan metode untuk menerapkan gaya.
3. **Rumus Gaya**:
   - Gaya gravitasi: F = m × g
   - Percepatan: a = F / m
   - Update kecepatan: v += a
   - Update posisi: pos += v
4. **Tampilan**: Lingkaran jatuh dari atas dengan massa berbeda, tetapi kecepatan jatuhnya sama.

✅ Contoh Implementasi Python Tkinter

```
# SCRIPT 12 - Simulasi Gravitasi dengan Tkinter
import tkinter as tk
from vector import Vector

# ========================
# Class Mover: mewakili bola atau benda yang jatuh
# ========================
class Mover:
    def __init__(self, mass, x, y):
        self.mass = mass
        self.radius = mass * 5  # Ukuran bola berdasarkan massa
        self.location = Vector(x, y)
        self.velocity = Vector(0, 0)
        self.acceleration = Vector(0, 0)

        # Simpan ID untuk menggambar objek canvas
        self.oval_id = None      # ID untuk bola
        self.label_id = None     # ID untuk teks massa di tengah bola
        self.info_id = None      # ID untuk info fisika di bawah canvas

    # Fungsi untuk menambahkan gaya pada benda (F/m = a)
    def apply_force(self, force):
        f = Vector.dived(force, self.mass)
        self.acceleration.add(f)

    # Perbarui kecepatan dan posisi benda
    def update(self):
```

```
        self.velocity.add(self.acceleration)
        self.location.add(self.velocity)
        self.acceleration = Vector(0, 0)  # Reset percepatan setelah update

    # Gambar atau update posisi bola dan label massa
    def display(self, canvas):
        x, y = self.location.x, self.location.y
        r = self.radius

        # Gambar atau update posisi bola
        if self.oval_id is None:
            self.oval_id = canvas.create_oval(x - r, y - r, x + r, y + r,
fill="skyblue")
        else:
            canvas.coords(self.oval_id, x - r, y - r, x + r, y + r)  # Perbarui posisi
bola

        # Gambar atau update teks massa
        if self.label_id is None:
            self.label_id = canvas.create_text(x, y, text=f"{self.mass}kg",
font=("Arial", 10), fill="black")
        else:
            canvas.coords(self.label_id, x, y)  # Perbarui posisi teks

    # Gambar atau update info fisika di bawah layar
    def display_info(self, canvas, offset_y):
        fg = self.mass * 0.1  # Gaya gravitasi (g = 0.1)
        a = fg / self.mass    # Percepatan = F/m
        v = self.velocity.y   # Kecepatan vertikal

        info_text = f"Massa = {self.mass} kg, Fg = {fg:.2f} N, a = {a:.2f} m/s², v =
{v:.2f} m/s"

        if self.info_id is None:
            self.info_id = canvas.create_text(
                100, offset_y,
                anchor="w", font=("Arial", 10),
                fill="black", text=info_text
            )
        else:
            canvas.itemconfig(self.info_id, text=info_text)  # Update teks
            canvas.coords(self.info_id, 100, offset_y)       # Update posisi teks

# ========================
# Class GravitySimulation: untuk menjalankan simulasi
# ========================
class GravitySimulation:
    def __init__(self, root):
        self.root = root
        self.WIDTH = 600
        self.HEIGHT = 400

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

        # Buat 2 bola dengan massa berbeda
        self.movers = [
            Mover(mass=10, x=150, y=50),
            Mover(mass=2, x=350, y=50)
        ]

        self.gravity = Vector(0, 0.1)  # Gaya gravitasi ke bawah
```

```
        self.update()  # Mulai simulasi

    def update(self):
        for i, mover in enumerate(self.movers):
            # Hitung gaya gravitasi dan terapkan ke masing-masing benda
            gravity_force = Vector.multed(self.gravity, mover.mass)
            mover.apply_force(gravity_force)

            mover.update()  # Update posisi berdasarkan gaya
            mover.display(self.canvas)  # Gambar atau update bola
            mover.display_info(self.canvas, offset_y=300 + i * 20)  # Info di bawah

        self.root.after(30, self.update)  # Loop update setiap 30 ms

# ========================
# Jalankan program
# ========================
if __name__ == "__main__":
    root = tk.Tk()
    root.title("Simulasi Gravitasi Berdasarkan Massa")
    app = GravitySimulation(root)
    root.mainloop()
```

🧪 **Contoh Output (di layar):**
Massa = 10 kg, Fg = 1.00 N, a = 0.10 m/s², v = 0.30
Massa = 2 kg, Fg = 0.20 N, a = 0.10 m/s², v = 0.30

📘 **Penjelasan Sederhana:**

- Gaya gravitasi = massa × gravitasi
- Tapi percepatan jatuh = gaya ÷ massa, hasilnya tetap sama!
- Maka dua bola jatuh **dengan kecepatan yang sama**, meskipun beratnya beda.

📹 **Apa yang Terjadi Sekarang?**

- Semua bola jatuh **dengan kecepatan sama**, besar kecil massanya.
- Tapi... angin tetap konstan → bola kecil lebih terdorong ke kanan.

🧠 **Penjelasan**

- **Gaya gravitasi** = ditarik ke bawah.
- Semakin berat → **gaya tarik lebih kuat**.
- Tapi juga lebih berat → **susah dipercepat**.
- Jadi semua benda jatuh **sama cepatnya**.

2.7 Friction

Mantap, kita lanjut ke materi berikutnya: **Memahami Rumus Gaya Gesekan** dan menerapkannya

---

🎯 **Tujuan Materi**
Mengenalkan:

- Cara **membaca rumus fisika**
- Mengenali **gaya vektor** (arah dan besar)
- Menerapkan **gaya gesekan (friction)** yang bekerja **melawan arah gerak**

📘 **Rumus Gaya Gesekan**
Rumus umum gaya gesekan:

Rumus umum gaya gesekan:

$$\vec{F}_{friction} = -\mu \cdot N \cdot \hat{v}$$

Penjelasan simbol:

- $\vec{F}_{friction}$: gaya gesekan (vektor)
- μ: koefisien gesekan (angka seberapa kasar permukaan)
- N: gaya normal (biasanya = massa × gravitasi, di permukaan datar)
- $\hat{v}$: vektor arah kecepatan (tapi sudah *dinyormalkan*, alias panjangnya = 1)
- Tanda – menunjukkan bahwa gaya ini **melawan arah gerak**.

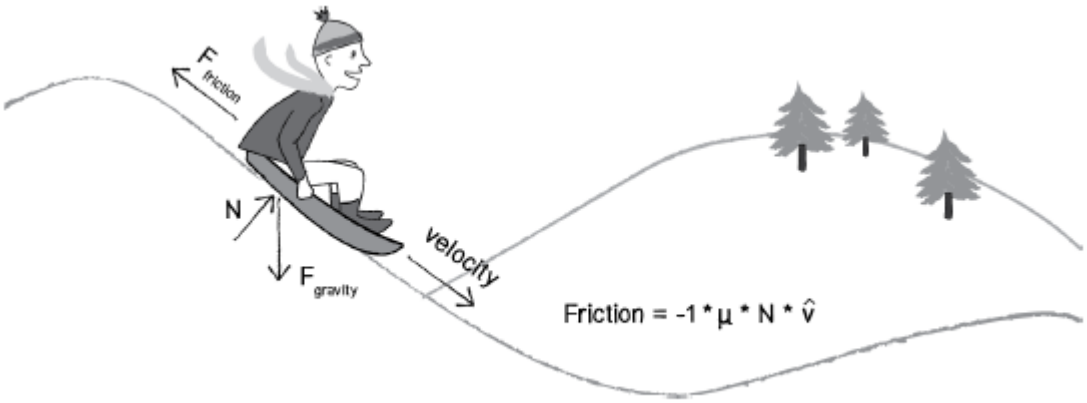


Figure 2.3

🧠 **Penjelasan**

Bayangkan kamu dorong kotak di lantai kasar. Walau kamu sudah dorong, kotaknya lama-lama berhenti. Itu karena **gesekan menahan gerak kotak**. Gesekan selalu **berlawanan arah** dengan arah kamu dorong.

🎯 **Tujuan Pembelajaran**

- Memahami **gaya gesekan sebagai gaya yang memperlambat gerak**
- Memahami konsep **dissipative force** (gaya yang mengurangi energi)
- Menghitung **arah dan besar gaya gesekan**
- Menerapkannya di simulasi Python Tkinter sederhana

📘 **Penjelasan Konsep Fisika**

Gaya Gesekan adalah gaya yang terjadi ketika dua permukaan saling bersentuhan dan **bergerak relatif** satu sama lain.

- Kalau kamu **mendorong kotak di lantai**, maka kotak itu akan melambat dan berhenti karena lantai **menahan gerak** kotak.
- Gaya gesekan **selalu melawan arah gerak**.
- Energi gerak (kinetik) berubah menjadi panas → disebut **dissipative force**.

🧪 **Rumus Friction**

🧪 **Rumus Friction**

$$\vec{F}_{friction} = -\mu \cdot N \cdot \hat{v}$$

- μ: Koefisien gesekan (semakin besar, semakin kasar)
- N: Gaya normal (biasanya dianggap 1 di simulasi sederhana)
- $\hat{v}$: Arah gerak (vektor kecepatan yang dinormalisasi)

🧩 **Langkah-langkah Menerapkan Friction**

Kita terapkan langkah-langkah ini **secara sederhana dan visual** dalam Python Tkinter, seperti sebelumnya:

✅ **1. Buat vektor friction dari kecepatan**

friction = mover.velocity.copy()

✅ **2. Normalisasi (jadikan vektor arah saja)**

friction.normalize()

✅ **3. Balik arahnya (karena melawan gerak)**

friction.multiply(-1)

✅ **4. Kalikan dengan mu * normal**

```
mu = 0.01        # kecil = licin, besar = kasar
normal = 1       # asumsi permukaan datar
friction_magnitude = mu * normal
friction.multiply(friction_magnitude)
```

✅ **5. Terapkan gaya tersebut ke objek**

mover.apply_force(friction)

📘 **Konsep Fisika: Gaya Gesek (Friction)**

Saat benda menyentuh permukaan (tanah), akan muncul gaya gesek yang menghambat geraknya.

Rumus Friction:

$$\vec{f}_{gesek} = -c \cdot \hat{v}$$

- c: koefisien gesekan (semakin besar, semakin berat geraknya)
- $\hat{v}$: vektor arah kecepatan, tetapi dibalik arahnya

```
# SCRIPT 13 - Simulasi Gaya Gesek dengan Tkinter
import tkinter as tk
from vector import Vector

# ======== Class Mover: Benda yang Diberi Gaya ========
class Mover:
    def __init__(self, canvas, x, y, mass):
        self.canvas = canvas               # Simpan referensi canvas
        self.mass = mass                   # Massa benda
        self.radius = mass * 8             # Radius tergantung massa
        self.position = Vector(x, y)       # Posisi awal
        self.velocity = Vector(0, 0)       # Kecepatan awal
        self.acceleration = Vector(0, 0)   # Percepatan awal

        # Gambar bola sekali, dan simpan id objeknya
        self.id = self.canvas.create_oval(
            x - self.radius, y - self.radius,
            x + self.radius, y + self.radius,
            fill="skyblue", outline="black", width=2
        )

    def apply_force(self, force):
        # Rumus Newton: a = F / m
        f = Vector.dived(force, self.mass)
        self.acceleration.add(f)

    def update(self):
        # Tambah percepatan ke kecepatan, lalu ke posisi
        self.velocity.add(self.acceleration)
        self.position.add(self.velocity)
        self.acceleration = Vector(0, 0)  # Reset percepatan tiap frame

        # Perbarui posisi gambar bola di canvas
        x, y, r = self.position.x, self.position.y, self.radius
        self.canvas.coords(self.id, x - r, y - r, x + r, y + r)
```

```
    def contact_edge(self, height):
        # Deteksi jika menyentuh bawah
        return self.position.y + self.radius >= height

    def bounce_edges(self, width, height):
        bounce = -0.9
        if self.position.x > width - self.radius:
            self.position.x = width - self.radius
            self.velocity.x *= bounce
        elif self.position.x < self.radius:
            self.position.x = self.radius
            self.velocity.x *= bounce

        if self.position.y > height - self.radius:
            self.position.y = height - self.radius
            self.velocity.y *= bounce

# ======== Setup Simulasi Tkinter ========
WIDTH, HEIGHT = 640, 400
root = tk.Tk()
root.title("Simulasi Gaya Gesek - The Nature of Code")
canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

# Label teks untuk info fisika
text = tk.Label(root, text="", font=("Courier", 10), justify="left", anchor="nw")
text.pack()

# Buat objek Mover (benda) dengan massa 5
mover = Mover(canvas, WIDTH // 2, 30, mass=5)

# Flag untuk mendeteksi mouse ditekan
mouse_pressed = [False]

# ======== Fungsi Simulasi Frame-by-Frame ========
def draw():
    # Tidak perlu hapus bola dari canvas karena pakai canvas.coords

    # 1. Gaya gravitasi konstan ke bawah
    gravity = Vector(0, 1)
    mover.apply_force(gravity)
    result = f"Gaya:\n  Gravitasi : {gravity}"

    # 2. Gaya angin ke kanan jika mouse ditekan
    if mouse_pressed[0]:
        wind = Vector(0.2, 0)
        mover.apply_force(wind)
        result += f"\n  Angin     : {wind}"

    # 3. Gaya gesek saat menyentuh lantai >> hilangkan jika tidak ingin ada friksi
    if mover.contact_edge(HEIGHT):
        c = 0.5
        friction = mover.velocity.copy()  # Ambil arah kecepatan
        friction.mult(-1)                 # Balik arah
        friction.normalize()              # Jadikan unit vector
        friction.mult(c)                  # Kalikan dengan koefisien
        mover.apply_force(friction)
        result += f"\n  Gesekan   : {friction}"

    # 4. Update posisi, cek pantulan, dan tampilkan
    mover.update()
```

```
    mover.bounce_edges(WIDTH, HEIGHT)

    # 5. Tampilkan info numerik
    result += f"\n\nPercepatan : {mover.acceleration}"
    result += f"\nKecepatan  : {mover.velocity}"
    result += f"\nPosisi     : {mover.position}"
    text.config(text=result)

    # Jalankan kembali setelah 30 ms
    root.after(30, draw)

# ======== Fungsi Interaksi Mouse ========
def on_press(event): mouse_pressed[0] = True
def on_release(event): mouse_pressed[0] = False
canvas.bind("<ButtonPress-1>", on_press)
canvas.bind("<ButtonRelease-1>", on_release)

# ======== Jalankan Simulasi ========
draw()
root.mainloop()
```

Bagian	Penjelasan
Vector	Kelas vektor 2D dengan operasi seperti penjumlahan, pembagian, normalisasi.
Mover	Bola yang dapat diberi gaya (gravitasi, gesekan, angin).
apply_force()	Menambahkan gaya ke percepatan.
update()	Mengubah kecepatan dan posisi.
contact_edge()	Mengecek apakah bola menyentuh tanah.
bounce_edges()	Bola memantul saat menyentuh tepi.
draw()	Fungsi utama yang menjalankan simulasi.

Visualisasi yang Dirasakan

- Benda akan **terus melambat** meski masih ada angin atau gravitasi.
- Benda bisa **berhenti** sebelum menyentuh sisi layar.
- Jika koefisien gesekan **mu ditambah**, benda **semakin cepat berhenti**.

Penjelasan Ringan

Bayangkan kamu meluncur di jalan yang licin (misal: es), kamu meluncur jauh.
Tapi kalau jalannya kasar (kayak aspal), kamu cepat berhenti.
Nah, di simulasi ini kita **atur kasar/halusnya permukaan dengan nilai μ**.

Eksperimen Seru

- **Ubah nilai mu** menjadi:
  - 0.001: sangat licin (seperti es)
  - 0.05: agak kasar
  - 0.2: sangat kasar (seperti karpet atau aspal)
- **Matikan gaya angin**, biarkan hanya gravitasi dan friction
- **Ubah massa** objek, lihat apakah benda berat lebih lambat melambat

---

2.8 Air and Fluid Resistance

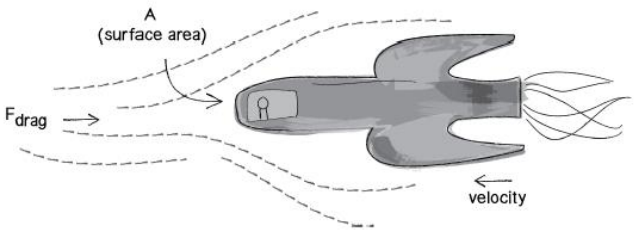
🌊 Konsep Dasar: Fluid Resistance (Gaya Hambat Fluida)

Ketika benda bergerak di udara atau air, benda tersebut mengalami hambatan. Hambatan ini disebut **fluid resistance** atau **drag force**. Sama seperti gesekan, drag membuat benda melambat.

Rumus Sederhana Drag Force:

$$\mathbf{Fd} = -c \cdot v^2 \cdot \hat{v}$$

- **c**: koefisien drag (semakin besar nilainya, hambatan makin kuat)
- **v**: kecepatan (magnitude dari velocity vector)
- $\hat{v}$: arah kecepatan (unit vector)
- Tanda **minus (-)** berarti drag force selalu melawan arah gerak.



🧪 1. Gaya Gravitasi (Gravity)

📘 Rumus:

$$\vec{F}_g = m \cdot \vec{g}$$

- m: massa benda (kg)
- $\vec{g}$: percepatan gravitasi (m/s²), dalam simulasi ini: $\vec{g} = (0, 0.1)$
- Arah: ke bawah (sumbu Y positif)

💻 Implementasi di kode:

```python
gravity = Vector(0, 0.1 * mover.mass)
mover.apply_force(gravity)
```

🧱 3. Hukum Newton II: Percepatan akibat gaya

📘 Rumus:

$$\vec{a} = \frac{\vec{F}_{\text{total}}}{m}$$

- $\vec{F}_{\text{total}}$: jumlah gaya (gravitasi + drag)
- m: massa benda
- $\vec{a}$: percepatan (m/s²)

💻 Implementasi di kode:

```python
f.div(self.mass)  # a = F / m
self.acceleration.add(f)
```

🧿 2. Gaya Hambat Fluida (Fluid Resistance / Drag)

📘 Rumus:

$$\vec{F}_d = -\frac{1}{2} \cdot c \cdot v^2 \cdot \hat{v}$$

(disederhanakan jadi):

$$\vec{F}_d = -c \cdot v^2 \cdot \hat{v}$$

- c: koefisien hambatan (drag coefficient), contoh: 0.1
- v: kecepatan benda (besar/magnitude)
- $\hat{v}$: arah unit dari kecepatan, yaitu v dibagi panjangnya → $\hat{v} = \vec{v}/|\vec{v}|$
- Tanda minus (–) menunjukkan arah berlawanan dari kecepatan

💻 Implementasi di kode:

```python
speed = mover.velocity.mag()
drag_magnitude = self.c * speed ** 2
drag = mover.velocity.copy()
drag.mult(-1)
drag.set_mag(drag_magnitude)
```

🏃 4. Update Kecepatan dan Posisi

📘 Rumus:

$$\vec{v} = \vec{v} + \vec{a} \cdot \Delta t$$

$$\vec{p} = \vec{p} + \vec{v} \cdot \Delta t$$

- Δt: waktu per frame (disederhanakan = 1 unit di simulasi)
- $\vec{v}$: kecepatan
- $\vec{p}$: posisi

💻 Implementasi di kode:

```python
self.velocity.add(self.acceleration)
self.position.add(self.velocity)
self.acceleration = Vector(0, 0)  # Reset setiap frame
```

🔁 5. Pemantulan saat menyentuh lantai

📘 Rumus:

$$v_y = -v_y \cdot d$$

- d: koefisien redaman (misal 0.9)
- v_y: kecepatan sumbu-y dibalik dan diredam

💻 Implementasi di kode:

```python
if self.position.y > height - self.radius:
    self.velocity.y *= -0.9
    self.position.y = height - self.radius
```

📊 Contoh Perhitungan Manual (1 Langkah)

Misal:

- Massa $m = 2$
- Kecepatan awal $\vec{v} = (0, 3)$
- Koefisien drag $c = 0.1$

1. Gaya gravitasi:

$$\vec{F}_g = 2 \cdot 0.1 = (0, 0.2)$$

2. Gaya drag:

$$v = |\vec{v}| = 3, \quad \vec{F}_d = -0.1 \cdot 3^2 = -0.9, \quad \vec{F}_d = (0, -0.9)$$

3. Total gaya:

$$\vec{F}_{\text{total}} = \vec{F}_g + \vec{F}_d = (0, 0.2 + -0.9) = (0, -0.7)$$

4. Percepatan:

$$\vec{a} = \frac{\vec{F}_{\text{total}}}{m} = (0, -0.7/2) = (0, -0.35)$$

5. Update kecepatan:

$$\vec{v}_{\text{baru}} = \vec{v} + \vec{a} = (0, 3) + (0, -0.35) = (0, 2.65)$$

6. Update posisi:

$$\vec{p}_{\text{baru}} = \vec{p} + \vec{v}_{\text{baru}}$$

```
# SCRIPT 13 - Simulasi Bola jatuh di Cairan
import tkinter as tk
import random
from vector import Vector
# ------------ Kelas Cairan (Liquid) ------------
class Liquid:
    def __init__(self, x, y, w, h, c):
        self.x = x
        self.y = y
        self.w = w
        self.h = h
        self.c = c  # Koefisien drag (resistance)

    # Cek apakah mover berada di dalam cairan
    def contains(self, mover):
        pos = mover.position
        return self.x < pos.x < self.x + self.w and self.y < pos.y < self.y + self.h

    # Hitung gaya hambat fluida (drag)
    def calculate_drag(self, mover):
        speed = mover.velocity.mag()
        drag_magnitude = self.c * speed ** 2

        # Arah drag berlawanan dengan arah kecepatan
        drag = mover.velocity.copy()
        drag.mult(-1)
        drag.set_mag(drag_magnitude)
        return drag

    # Gambar cairan di canvas
    def show(self, canvas):
        canvas.create_rectangle(self.x, self.y, self.x + self.w, self.y + self.h,
fill="#ccf")

# ------------ Kelas Benda Bergerak (Mover) ------------
class Mover:
```

```
    def __init__(self, x, y, mass):
        self.mass = mass
        self.radius = mass * 8
        self.position = Vector(x, y)
        self.velocity = Vector(0, 0)
        self.acceleration = Vector(0, 0)

    # Terapkan gaya ke benda (F = m * a → a = F / m)
    def apply_force(self, force):
        f = force.copy()
        f.div(self.mass)
        self.acceleration.add(f)

    # Update posisi berdasarkan kecepatan dan percepatan
    def update(self):
        self.velocity.add(self.acceleration)
        self.position.add(self.velocity)
        self.acceleration = Vector(0, 0)  # Reset percepatan setelah digunakan

    # Gambar benda di canvas
    def show(self, canvas):
        x = self.position.x
        y = self.position.y
        r = self.radius
        canvas.create_oval(x - r, y - r, x + r, y + r, fill="gray", outline="black")

    # Cek apakah menyentuh lantai, jika ya, pantul
    def check_edges(self, height):
        if self.position.y > height - self.radius:
            self.velocity.y *= -0.9
            self.position.y = height - self.radius

# ------------ Program Simulasi ------------
class Simulation:
    def __init__(self, root):
        self.root = root
        self.width = 640
        self.height = 360
        self.canvas = tk.Canvas(root, width=self.width, height=self.height)
        self.canvas.pack()
        self.movers = []
        self.liquid = Liquid(0, self.height // 2, self.width, self.height // 2, 0.1)
        self.reset()

        self.root.bind("<Button-1>", self.mouse_pressed)
        self.draw()

    #Buat ulang objek mover secara acak
    def reset(self):
        self.movers = []
        for i in range(9):
            x = 40 + i * 60
            m = random.uniform(0.5, 3)
            self.movers.append(Mover(x, 0, m))

    # # Exercise 2.5 – Variasi Ketinggian Jatuh
    # def reset(self):
    #     self.movers = []
    #     # Tiga bola dengan ketinggian berbeda
    #     heights = [50, 100, 150]  # Tiga ketinggian berbeda
    #     for y in heights:
    #         x_pos = random.randint(100, 700)
```

```
    #         m = random.uniform(1, 2.5)  # Massa acak sedang
    #         self.movers.append(Mover(x_pos, y, m))

    # Jika mouse diklik, ulang simulasi
    def mouse_pressed(self, event):
        self.reset()

    # Fungsi utama untuk menggambar setiap frame
    def draw(self):
        self.canvas.delete("all")  # Hapus semua gambar lama
        self.liquid.show(self.canvas)  # Gambar cairan

        for mover in self.movers:
            # Jika mover masuk cairan, terapkan gaya drag
            if self.liquid.contains(mover):
                drag = self.liquid.calculate_drag(mover)
                mover.apply_force(drag)

            # Gaya gravitasi ke bawah (F = m * g)
            gravity = Vector(0, 0.1 * mover.mass)
            mover.apply_force(gravity)

            mover.update()
            mover.check_edges(self.height)
            mover.show(self.canvas)

        # Jalankan ulang fungsi draw tiap 16 ms (sekitar 60 fps)
        self.root.after(16, self.draw)

# ------------ Jalankan Program ------------
root = tk.Tk()
root.title("Simulasi Fluid Resistance (Sederhana)")
app = Simulation(root)
root.mainloop()
```

📌 **Kesimpulan Konsep**

- Benda jatuh karena **gravitasi**.
- Di dalam air, kecepatannya **melambat** karena **drag** (gaya hambat).
- Hukum Newton II digunakan untuk menghitung **percepatan** dari total gaya.
- Posisi diupdate tiap waktu berdasarkan **kecepatan**.

Kita akan lanjutkan simulasi sebelumnya (fluid resistance) dengan menerapkan tiga exercise berikut:

✅ **Exercise 2.5 – Variasi Ketinggian Jatuh**

Tujuan: Menjatuhkan bola dari berbagai ketinggian untuk mengamati efek drag force saat menyentuh air.

Perubahan:

- Tambahkan berbagai bola (Mover) di posisi y yang berbeda-beda saat inisialisasi.
- Simulasi menunjukkan: semakin tinggi bola dijatuhkan, semakin besar kecepatannya saat menyentuh air → drag force juga makin besar.

```
# Membuat 5 bola dijatuhkan dari berbagai ketinggian
        self.root.bind("<Button-1>", self.mouse_pressed)
        self.draw()

    # #Buat ulang objek mover secara acak
    # def reset(self):
```

```
    #     self.movers = []
    #     for i in range(9):
    #         x = 40 + i * 60
    #         m = random.uniform(0.5, 3)
    #         self.movers.append(Mover(x, 0, m))
```

```
    # Exercise 2.5 – Variasi Ketinggian Jatuh
    def reset(self):
        self.movers = []
        # Tiga bola dengan ketinggian berbeda
        heights = [50, 100, 150]  # Tiga ketinggian berbeda
        for y in heights:
            x_pos = random.randint(100, 700)
            m = random.uniform(1, 2.5)  # Massa acak sedang
            self.movers.append(Mover(x_pos, y, m))

    # Jika mouse diklik, ulang simulasi
    def mouse_pressed(self, event):
        self.reset()
```

✅ **Exercise 2.6 – Drag Berdasarkan Sisi Persegi**

Tujuan: Menghitung drag force berdasarkan luas permukaan benda yang bersentuhan dengan air.

Perubahan:

- Buat class Box sebagai turunan dari Mover dengan parameter ukuran sisi.
- Dalam metode drag, gunakan area = side_length (karena kotak menyentuh dengan sisi bawahnya).

```
# SCRIPT 15 - Simulasi Gaya Drag pada Kotak di Cairan
import tkinter as tk
from vector import Vector

# ======== Box Class ========
class Box:
    def __init__(self, x, y, w, h, mass):
        self.width = w
        self.height = h
        self.mass = mass
        self.position = Vector(x, y)
        self.velocity = Vector(0, 0)
        self.acceleration = Vector(0, 0)
    def apply_force(self, force):
        f = Vector.dived(force, self.mass)
        self.acceleration.add(f)
    def update(self):
        self.velocity.add(self.acceleration)
        self.position.add(self.velocity)
        self.acceleration = Vector(0, 0)
    def show(self, canvas):
        canvas.coords(self.id,
            self.position.x - self.width / 2,
            self.position.y - self.height / 2,
            self.position.x + self.width / 2,
            self.position.y + self.height / 2)
    def set_id(self, canvas, color="orange"):
        self.id = canvas.create_rectangle(
            self.position.x - self.width / 2,
            self.position.y - self.height / 2,
            self.position.x + self.width / 2,
            self.position.y + self.height / 2,
            fill=color, outline="black", width=2
        )
```

```
# ======== Setup Tkinter ========
WIDTH, HEIGHT = 640, 480
root = tk.Tk()
root.title("Box Drag Force with Surface Area")
canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

fluid_top = HEIGHT // 2
canvas.create_rectangle(0, fluid_top, WIDTH, HEIGHT, fill="lightblue", outline="")

# Dua kotak dengan area bawah berbeda
boxes = [
    Box(200, 100, w=40, h=60, mass=2),
    Box(400, 100, w=100, h=60, mass=2),
]

for box in boxes:
    box.set_id(canvas)

# ======== Loop Simulasi ========
def draw():
    for box in boxes:
        gravity = Vector(0, 0.5)
        box.apply_force(gravity)

        # Cek apakah kotak masuk ke area air
        if box.position.y + box.height/2 >= fluid_top:
            c = 0.0005  # koefisien drag lebih kecil agar tidak terlalu kuat
            speed = box.velocity.mag()
            if speed > 0:
                drag_dir = box.velocity.copy()
                drag_dir.normalize()
                drag_dir.mult(-1)  # arah drag berlawanan

                A = box.width  # permukaan menyentuh air
                drag_magnitude = c * A * speed * speed
                drag = drag_dir
                drag.mult(drag_magnitude)

                box.apply_force(drag)

        box.update()
        box.show(canvas)

    root.after(30, draw)

draw()
root.mainloop()
```

✅ **Exercise 2.7 – Lift-Induced Drag (Gaya Angkat)**

Tujuan: Menambahkan gaya lift (tegak lurus arah gerak) untuk meniru efek sayap pesawat.

Konsep:

- Gaya lift searah dengan vektor tegak lurus kecepatan (velocity.perpendicular()).
- Gaya lift **hanya bekerja saat kecepatan cukup tinggi** dan benda memiliki “sudut”.

🧪 Gaya-gaya yang Bekerja:

1. **Gaya Gravitasi**

$$\vec{F}_{gravity} = m \cdot \vec{g}$$

Di kode: `gravity = Vector(0, 0.3)`
(arah ke bawah dengan percepatan tetap)

2. **Gaya Hambat Udara (Drag Force)**

$$\vec{F}_{drag} = -c_d \cdot v^2 \cdot \hat{v}$$

- c_d: koefisien drag
- v: kecepatan (magnitudo dari `velocity`)
- $\hat{v}$: arah kecepatan (unit vector)
- `drag_dir.mult(-1)`: arah drag berlawanan arah kecepatan

3. **Gaya Angkat (Lift Force)**

$$\vec{F}_{lift} = -c_l \cdot v^2 \cdot \hat{v}_\perp$$

- c_l: koefisien lift
- $\hat{v}_\perp$: arah tegak lurus terhadap kecepatan
- `velocity.rotate90()` → vektor tegak lurus

📌 Contoh Perhitungan *(misalnya pada suatu frame)*:

Misalkan:

- Kecepatan awal $\vec{v} = (4, 0) \Rightarrow v = 4$
- Koefisien drag: $c_d = 0.01$
- Koefisien lift: $c_l = 0.015$ atau 0.05 jika tombol ↑ ditekan

---

🔙 Drag Force:

$$F_{drag} = -c_d \cdot v^2 = -0.01 \cdot 4^2 = -0.01 \cdot 16 = -0.16$$

Arah: berlawanan dengan vektor kecepatan → $(4, 0)$ → unit vector = $(1, 0)$

$$\vec{F}_{drag} = -0.16 \cdot (1, 0) = (-0.16, 0)$$

🪽 Lift Force (tanpa ↑ ditekan):

$$F_{lift} = -c_l \cdot v^2 = -0.015 \cdot 16 = -0.24$$

Arah: tegak lurus kecepatan → kecepatan (4, 0) ⟶ rotasi 90° ⟶ (0, 4)
Normalize → (0, 1)

$$\vec{F}_{lift} = -0.24 \cdot (0, 1) = (0, -0.24)$$

Jika tombol ↑ ditekan:

$$F_{lift} = -0.05 \cdot 16 = -0.8 \Rightarrow \vec{F}_{lift} = (0, -0.8)$$

🔁 Total Gaya yang Diterapkan:

$$\vec{F}_{total} = \vec{F}_{gravity} + \vec{F}_{drag} + \vec{F}_{lift}$$

Dengan contoh tombol ↑ ditekan:

- Gravitasi: $(0, 0.3)$
- Drag: $(-0.16, 0)$
- Lift: $(0, -0.8)$

$$\vec{F}_{total} = (-0.16, -0.5)$$

🔁 Percepatan dan Update:

$$\vec{a} = \frac{\vec{F}_{total}}{m} = \vec{F}_{total}$$

(langsung dipakai karena massa = 1)

Update posisi:

```python
velocity += acceleration
position += velocity
```

```
# SCRIPT 16 - Simulasi Gaya Lift dan Drag pada Sayap Pesawat
import tkinter as tk
from vector import Vector

# ======= Sayap Pesawat =======
class Wing:
    def __init__(self, x, y):
        self.position = Vector(x, y)
        self.velocity = Vector(5, 0)
        self.acceleration = Vector(7, 0)
        self.width = 60
        self.height = 20
        self.mass = 1
    def apply_force(self, force):
        f = Vector(force.x / self.mass, force.y / self.mass)
        self.acceleration.add(f)
    def update(self):
        self.velocity.add(self.acceleration)
        #self.velocity.limit(30)
        self.position.add(self.velocity)
```

```
        self.acceleration = Vector(0, 0)
    def show(self, canvas):
        canvas.coords(self.id,
            self.position.x - self.width/2,
            self.position.y - self.height/2,
            self.position.x + self.width/2,
            self.position.y + self.height/2)
    def create(self, canvas):
        self.id = canvas.create_rectangle(
            self.position.x - self.width/2,
            self.position.y - self.height/2,
            self.position.x + self.width/2,
            self.position.y + self.height/2,
            fill="red"
        )

# ======= Setup Tkinter =======
WIDTH, HEIGHT = 800, 600
root = tk.Tk()
root.title("Simulasi Gaya Lift dan Drag")
canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

# Buat sayap
wing = Wing(100, HEIGHT/2)
wing.create(canvas)

# Tombol keyboard
key_pressed = {"Up": False}
def key_down(event):
    if event.keysym == "Up":
        key_pressed["Up"] = True
def key_up(event):
    if event.keysym == "Up":
        key_pressed["Up"] = False
root.bind("<KeyPress>", key_down)
root.bind("<KeyRelease>", key_up)

# ======= Loop utama =======
def draw():
    canvas.delete("info")

    # Gaya gravitasi
    gravity = Vector(0, 0.1)
    wing.apply_force(gravity)

    # Gaya drag (melawan gerak)
    drag = Vector(-wing.velocity.x * 0.02, -wing.velocity.y * 0.02)
    wing.apply_force(drag)

    # Gaya lift saat tombol ditekan
    if key_pressed["Up"]:
        lift = Vector(0, -0.6)
        wing.apply_force(lift)
    else:
        lift = Vector(0, 0)

    # Update gerakan & gambar
    wing.update()
    wing.show(canvas)

    # Info di layar
```

```
    canvas.create_text(10, 10, anchor="nw", text=f"Kecepatan: ({wing.velocity.x:.2f}, {wing.velocity.y:.2f})", tag="info")
    canvas.create_text(10, 30, anchor="nw", text=f"Lift Aktif: {'Ya' if key_pressed['Up'] else 'Tidak'}", tag="info")

    root.after(30, draw)

draw()
root.mainloop()
```

🧠 **Penjelasan Singkat**

- **Gravitasi** konstan: menarik ke bawah.
- **Lift** aktif saat tombol "Up" ditekan: mendorong ke atas.
- **Drag** selalu aktif: memperlambat kecepatan.
- **Velocity.limit(15)** menjaga kecepatan tetap masuk akal (mencegah Overflow).

2.9 Gravitational Attraction

💡 **Penjelasan Singkat**

- Setiap benda dengan massa saling tarik-menarik karena gaya gravitasi.
- Gaya tarik ini dihitung menggunakan rumus fisika:

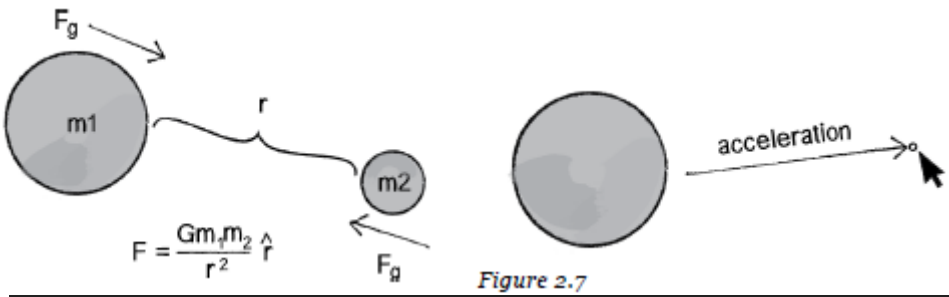


Figure 2.7

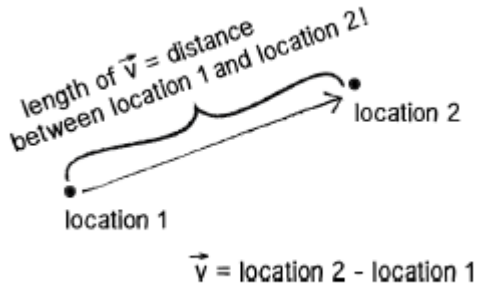
- Gaya tarik ini dihitung menggunakan rumus fisika:

$$F = G \times \frac{m1 \times m2}{r^2}$$

  - `G` : Konstanta gravitasi (di simulasi bisa kita buat `1` agar lebih sederhana).
  - `m1` , `m2` : Massa kedua objek.
  - `r` : Jarak antara dua objek.

G: Konstanta gravitasi (di simulasi bisa kita buat 1 agar lebih sederhana).

- m1, m2: Massa kedua objek.
- r: Jarak antara dua objek.



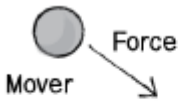


Figure 2.8

Struktur Program

- Mover: Benda yang bisa bergerak karena gaya.
- Attractor: Benda yang diam tapi memberikan gaya tarik.
- apply_force: Fungsi untuk menambahkan gaya ke Mover.

Kode Lengkap Python Tkinter

```
# SCRIPT 17 - Simulasi Gaya Tarik antara Dua Benda
import tkinter as tk
from vector import Vector

# === CLASS MOVER (BENDA YANG DITARIK) ===
class Mover:
    def __init__(self, x, y, mass, canvas):
        self.location = Vector(x, y)
        self.velocity = Vector(0, 0)
        self.acceleration = Vector(0, 0)
        self.mass = mass
        self.canvas = canvas
        self.radius = mass * 2

        self.id = canvas.create_oval(x - self.radius, y - self.radius,
                                     x + self.radius, y + self.radius,
                                     fill='skyblue')

    def apply_force(self, force):
        # F = m * a → a = F / m
        a = force.dived(self.mass)  # vektor percepatan
        self.acceleration = self.acceleration.added(a)

    def update(self):
        self.velocity = self.velocity.added(self.acceleration)
        self.location = self.location.added(self.velocity)
        self.acceleration = Vector(0, 0)

    def display(self):
        x, y, r = self.location.x, self.location.y, self.radius
        self.canvas.coords(self.id, x - r, y - r, x + r, y + r)

# === CLASS PENARIK (PUSAT GAYA) ===
class Attractor:
    def __init__(self, x, y, mass, canvas):
        self.location = Vector(x, y)
        self.mass = mass
        self.canvas = canvas
        self.radius = mass * 2
        self.G = 1  # konstanta gravitasi
```

```
        self.id = canvas.create_oval(x - self.radius, y - self.radius,
                                     x + self.radius, y + self.radius,
                                     fill='orange')

    def attract(self, m):
        # Gaya arah = posisi attractor - posisi mover
        force = self.location.subbed(m.location)
        distance = force.mag()
        distance = max(5, min(distance, 25))  # batasi jarak

        # Normalisasi arah dan hitung kekuatan gaya
        direction = force.normalized()
        strength = (self.G * self.mass * m.mass) / (distance ** 2)
        force_vector = direction.multed(strength)
        return force_vector

    def display(self):
        # Tidak perlu update posisi
        pass

# === SETUP TKINTER ===
WIDTH, HEIGHT = 600, 400
root = tk.Tk()
root.title("Simulasi Gaya Tarik Gravitasi")

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

# Label teks informasi
info_text = canvas.create_text(10, 10, anchor="nw", text="", font=("Courier", 10),
fill="black")

# Objek utama
attractor = Attractor(WIDTH / 2, HEIGHT / 2, 20, canvas)
mover = Mover(100, 100, 4, canvas)

# === LOOP UTAMA ===
def draw():
    # Hitung gaya tarik
    force = attractor.attract(mover)
    mover.apply_force(force)
    mover.update()
    mover.display()

    # Update teks info
    info = (
        f"Gaya (F):        {force}\n"
        f"Percepatan (a):  {mover.acceleration}\n"
        f"Kecepatan (v):   {mover.velocity}\n"
        f"Posisi (s):      {mover.location}"
    )
    canvas.itemconfig(info_text, text=info)

    root.after(33, draw)

draw()
root.mainloop()
```

🎯 **Penjelasan Visual**

- **Lingkaran oranye**: Attractor (objek pusat gravitasi).

- **Lingkaran biru**: Mover (objek yang ditarik).
- Semakin dekat Mover ke Attractor, semakin besar gaya tariknya.
- Kita membatasi jarak minimum dan maksimum agar simulasi tetap stabil.

Berikut adalah **penjelasan lengkap dan implementasi Python Tkinter** untuk:

1. **Example 2.6**: Simulasi satu *Mover* dan satu *Attractor*.
2. **Example 2.7**: Simulasi banyak *Movers* yang tertarik oleh satu *Attractor*.

```
# SCRIPT 18 - Simulasi Gravitasi Multi Mover
import tkinter as tk
from vector import Vector

# ======== CLASS MOVER (BENDA BERGERAK) ========
class Mover:
    def __init__(self, x, y, mass, canvas):
        self.location = Vector(x, y)
        self.velocity = Vector(0, 0)
        self.acceleration = Vector(0, 0)
        self.mass = mass
        self.r = self.mass * 2  # Ukuran lingkaran
        self.canvas = canvas
        self.oval = canvas.create_oval(x - self.r, y - self.r,
                                       x + self.r, y + self.r,
                                       fill='skyblue')
        self.text = canvas.create_text(0, 0, anchor="nw", font=("Arial", 10),
fill="black")

    def apply_force(self, force):
        # F = m * a => a = F / m
        f = force.dived(self.mass)  # tidak ubah force asli
        self.acceleration = self.acceleration.added(f)  # akumulasi percepatan

    def update(self):
        self.velocity = self.velocity.added(self.acceleration)
        self.location = self.location.added(self.velocity)
        self.acceleration = Vector(0, 0)

    def display(self):
        x, y = self.location.x, self.location.y
        r = self.r
        self.canvas.coords(self.oval, x - r, y - r, x + r, y + r)

        # Tampilkan informasi perhitungan di atas benda
        info =
f"Pos:({x:.1f},{y:.1f})\nVel:({self.velocity.x:.1f},{self.velocity.y:.1f})"
        self.canvas.coords(self.text, x + r + 5, y - r)
        self.canvas.itemconfig(self.text, text=info)

# ======== CLASS ATTRACTOR (SUMBER GAYA TARIK) ========
class Attractor:
    def __init__(self, x, y, mass, canvas):
        self.location = Vector(x, y)
        self.mass = mass
        self.G = 1
        self.r = self.mass * 2
        self.canvas = canvas
        self.oval = canvas.create_oval(x - self.r, y - self.r,
```

```
                                    x + self.r, y + self.r,
                                    fill='orange')

    def attract(self, m):
        # Arah gaya: dari benda ke pusat
        force = self.location.subbed(m.location)
        distance = force.mag()
        distance = max(5, min(distance, 25))  # Clamp jarak
        direction = force.normalized()  # arah vektor
        strength = (self.G * self.mass * m.mass) / (distance ** 2)
        force = direction.multed(strength)
        return force

    def display(self):
        x, y = self.location.x, self.location.y
        r = self.r
        self.canvas.coords(self.oval, x - r, y - r, x + r, y + r)

# ======== SETUP TKINTER ========
WIDTH, HEIGHT = 600, 400
root = tk.Tk()
root.title("Simulasi Gravitasi - Multi Mover")

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

# Buat attractor dan daftar mover
attractor = Attractor(WIDTH / 2, HEIGHT / 2, 20, canvas)

movers = [
    Mover(100, 100, 4, canvas),
    Mover(200, 50, 6, canvas),
    Mover(150, 250, 3, canvas)
]

# Loop utama animasi
def draw():
    for mover in movers:
        force = attractor.attract(mover)
        mover.apply_force(force)
        mover.update()
        mover.display()

    attractor.display()
    root.after(33, draw)

draw()
root.mainloop()
```

🧠 **Exercise 2.9 – Desain Gaya Kreatif Sendiri**

💡 **Buat aturan gaya sendiri**:

- Gaya **semakin kuat jika jauh** (kebalikan gravitasi).
- Gaya **menarik jika jauh, menolak jika dekat**.

Contoh Gaya Unik: Menarik Jauh, Menolak Dekat

```
def custom_force(self, m):
    force = self.location.sub(m.location)
    distance = force.mag()
```

```
    force = force.normalize()

    # Jika terlalu dekat, tolak
    if distance < 50:
        strength = - (self.G * self.mass * m.mass) / (distance ** 2)
    else:
        strength = (self.G * self.mass * m.mass) / (distance ** 2)

    return force.mult(strength)
```

Ganti attract() di atas dengan custom_force() untuk efek eksperimen kreatif.

🎨 Ide Visualisasi Tambahan

Eksperimen	Ide Tampilan
Trail warna-warni	Gunakan warna berbeda untuk jejak
Pola simetris	Letakkan Attractors dalam pola geometris
Interaksi mouse	Tambah Attractor saat klik

🎯 Tujuan Exercise 2.9

Mendesain **gaya buatan sendiri**, bukan hanya gaya gravitasi biasa:

- Contoh 1: **Gaya semakin kuat jika jauh**.
- Contoh 2: **Menarik objek jauh, menolak objek dekat**.
- Tujuannya adalah mengembangkan **aturan sendiri** untuk interaksi antar objek.

✅ Implementasi Python Tkinter

- Kita akan buat sistem dengan banyak **Mover** dan satu **Attractor buatan**.
- Gaya akan mengikuti aturan: **menarik jika jauh, menolak jika dekat**.

🧪 Kode Lengkap dengan Komentar

```
# SCRIPT 19 - Simulasi Gaya Tarik Buatan Sendiri
import tkinter as tk
import random
from vector import Vector

# ---- Objek yang bergerak (Mover) ----
class Mover:
    def __init__(self, mass, x, y, canvas):
        self.mass = mass
        self.location = Vector(x, y)
        self.velocity = Vector(0, 0)
        self.acceleration = Vector(0, 0)
        self.trail = []

        self.r = self.mass * 3
        self.canvas = canvas
        self.oval = canvas.create_oval(x - self.r, y - self.r,
                                       x + self.r, y + self.r,
                                       fill="blue", outline="")

    def apply_force(self, force):
        f = force.multed(1 / self.mass)
        self.acceleration.add(f)

    def update(self):
        self.velocity.add(self.acceleration)
        self.location.add(self.velocity)
        self.acceleration = Vector(0, 0)
```

```
        self.trail.append((self.location.x, self.location.y))
        if len(self.trail) > 40:
            self.trail.pop(0)

    def display(self):
        # Update posisi oval dengan .coords()
        x, y, r = self.location.x, self.location.y, self.r
        self.canvas.coords(self.oval, x - r, y - r, x + r, y + r)

        # Gambar jejak (garis-garis)
        # for i in range(len(self.trail) - 1):
        #     x1, y1 = self.trail[i]
        #     x2, y2 = self.trail[i + 1]
        #     self.canvas.create_line(x1, y1, x2, y2, fill="skyblue", width=1)

# ---- Attractor dengan Gaya Buatan ----
class Attractor:
    def __init__(self, x, y, canvas):
        self.mass = 20
        self.location = Vector(x, y)
        self.G = 0.6
        self.r = self.mass
        self.canvas = canvas
        self.oval = canvas.create_oval(x - self.r, y - self.r,
                                       x + self.r, y + self.r,
                                       fill="orange", outline="")

    def custom_force(self, m):
        force = self.location.subbed(m.location)
        distance = force.mag()
        distance = max(5.0, min(distance, 100.0))  # Clamp jarak
        force = force.normalized()

        # Gaya: tarik jika jauh, tolak jika dekat
        if distance < 60:
            strength = - (self.G * self.mass * m.mass) / (distance ** 2)
        else:
            strength = (self.G * self.mass * m.mass) / (distance ** 2)

        return force.multed(strength)

    def display(self):
        x, y, r = self.location.x, self.location.y, self.r
        self.canvas.coords(self.oval, x - r, y - r, x + r, y + r)

# ---- Setup Tkinter ----
WIDTH, HEIGHT = 800, 600
root = tk.Tk()
root.title("Exercise 2.9 - Gaya Buatan Sendiri")
canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="black")
canvas.pack()

# Attractor di tengah
attractor = Attractor(WIDTH // 2, HEIGHT // 2, canvas)

# Banyak mover
movers = [Mover(random.uniform(1, 3),
                random.randint(0, WIDTH),
                random.randint(0, HEIGHT),
                canvas)
          for _ in range(20)]
```

```
# ---- Loop animasi ----
def draw():
    canvas.delete("trail")  # Hapus jejak saja (bukan oval)
    attractor.display()
    for mover in movers:
        force = attractor.custom_force(mover)
        mover.apply_force(force)
        mover.update()
        mover.display()
    root.after(33, draw)

draw()
root.mainloop()
```

🧠 **Pertanyaan Eksploratif**

Pertanyaan	**Untuk Melatih Logika**
Apa yang terjadi jika G dinaikkan?	Gaya jadi lebih kuat
Bagaimana jika distance < 60 diganti distance < 100?	Area tolak jadi lebih besar
Bagaimana jika - dihilangkan?	Semua gaya jadi tarik
Bisakah kita tambahkan gaya rotasi atau angin?	Bisa! Bisa dikombinasikan

---

Mutual Attraction / Gravitational Attraction

🧠 **1. Penjelasan Materi**

Apa yang sedang disimulasikan?

- Dua benda (A dan B) saling **menarik** satu sama lain, seperti **gravitasi** antara planet.
- Gaya tarik tergantung pada:
  - **Jarak** antara dua benda.
  - **Massa** masing-masing benda.
  - Rumus gaya gravitasi: $F = \frac{G \cdot m_1 \cdot m_2}{r^2}$

Di mana:

- FF: gaya tarik
- GG: konstanta gravitasi (di sini = 1)
- m1,m2m_1, m_2: massa benda A dan B
- rr: jarak antar benda

📜 **2. Tahapan dan Struktur Script**

Struktur:

- Class Vector: Operasi matematika 2D (tambah, kurang, magnitude, normalisasi).
- Class Body: Representasi benda (massa, posisi, kecepatan, akselerasi).
- Fungsi attract(): Hitung gaya tarik dan diterapkan ke benda lain.
- Fungsi apply_force(): Mengubah percepatan dari gaya.
- Fungsi update(): Update posisi berdasarkan kecepatan dan percepatan.
- Fungsi display(): Gambar benda di kanvas dengan posisi baru.
- Main loop animate(): Jalankan simulasi terus-menerus (33 ms = 30 FPS).

📘 **4. Rumus dan Perhitungan Matematis**

- **Gaya Gravitasi**: $F = \frac{G \cdot m_1 \cdot m_2}{r^2}$
- **Arah Gaya**:
  - Ambil vektor arah dari posisi A ke B

    - Normalisasi (buat panjang = 1)
    - Kalikan dengan besar gaya FF
- **Percepatan** (Newton 2): $a = \frac{F}{m}$
- **Update Kecepatan dan Posisi**: $v = v + a$ $p = p + v$

```
# SCRIPT 20 - Simulasi Gaya Tarik Gravitasi Sederhana
import tkinter as tk
import math
import time
from vector import Vector

# ------------------------
# Class Body (benda dengan massa, posisi, kecepatan, dll)
# ------------------------
class Body:
    def __init__(self, canvas, x, y, mass, color):
        self.canvas = canvas
        self.mass = mass
        self.position = Vector(x, y)
        self.velocity = Vector(0, 0)
        self.acceleration = Vector(0, 0)
        self.radius = math.sqrt(mass) * 2
        self.color = color
        self.shape = canvas.create_oval(x - self.radius, y - self.radius,
                                        x + self.radius, y + self.radius,
                                        fill=color)

    # Terapkan gaya ke benda (F = ma)
    def apply_force(self, force):
        a = force.dived(self.mass)
        self.acceleration.add(a)

    # Tarik benda lain dengan gaya gravitasi
    def attract(self, other):
        force = self.position.subbed(other.position)
        distance = max(5, min(force.mag(), 25))  # Hindari terlalu dekat atau jauh
        G = 1
        strength = (G * self.mass * other.mass) / (distance * distance)
        direction = force.normalized()
        attraction = direction.multed(strength)
        other.apply_force(attraction)
        return distance, strength  # Untuk ditampilkan di layar

    # Update posisi benda berdasarkan kecepatan dan percepatan
    def update(self):
        self.velocity.add(self.acceleration)
        self.position.add(self.velocity)
        self.acceleration = Vector(0, 0)  # Reset percepatan setelah update

    # Tampilkan benda di canvas
    def display(self):
        x = self.position.x
        y = self.position.y
        self.canvas.coords(self.shape,
                           x - self.radius, y - self.radius,
                           x + self.radius, y + self.radius)

# ------------------------
# Inisialisasi Tkinter
```

```
# ------------------------
WIDTH = 640
HEIGHT = 480
root = tk.Tk()
root.title("Simulasi Mutual Gravitational Attraction")
canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg='white')
canvas.pack()

# Label untuk menampilkan info
info = tk.Label(root, text="", font=("Arial", 12))
info.pack()

# ------------------------
# Buat dua benda
# ------------------------
bodyA = Body(canvas, 200, 240, 20, 'red')
bodyB = Body(canvas, 440, 240, 20, 'blue')

# Kecepatan awal (agar mereka bergerak melingkar)
bodyA.velocity = Vector(0, -1.5)
bodyB.velocity = Vector(0, 1.5)

# ------------------------
# Fungsi animasi utama
# ------------------------
def animate():
    # Bersihkan layar dulu
    canvas.delete("vector")

    # Gaya tarik antara kedua benda
    d1, f1 = bodyA.attract(bodyB)
    d2, f2 = bodyB.attract(bodyA)

    # Update posisi dan gambar ulang
    bodyA.update()
    bodyB.update()
    bodyA.display()
    bodyB.display()

    # Tampilkan informasi gaya dan jarak
    info.config(text=f"Jarak: {d1:.2f} px | Gaya tarik: {f1:.4f} N\n"
                     f"Kecepatan A: ({bodyA.velocity.x:.2f},
{bodyA.velocity.y:.2f})\n"
                     f"Kecepatan B: ({bodyB.velocity.x:.2f}, {bodyB.velocity.y:.2f})")

    # Jalankan ulang animasi tiap 33 ms (30 FPS)
    root.after(33, animate)

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

simulasi *Mutual Attraction* dari *The Nature of Code* oleh Daniel Shiffman

---

🔢 1. Penjelasan Rumus dan Perhitungan Matematika

Simulasi ini memakai hukum gravitasi universal Newton: $F = G \cdot \frac{m_1 \cdot m_2}{r^2}$

Keterangan:

- **F** = gaya tarik antar dua benda
- **G** = konstanta gravitasi (misal: 0.4 untuk skala simulasi)
- **m1, m2** = massa benda pertama dan kedua
- **r** = jarak antara dua benda

Gaya ini punya arah ke arah benda lain dan besar gaya tergantung **massa** dan **jaraknya**.

📦 2. Tahapan dalam Script Python Tkinter

1. **Class Vector**
   Representasi 2D (x, y) — untuk posisi, kecepatan, percepatan, dan gaya.
2. **Class Mover**
   Objek benda: punya massa, posisi (pos), kecepatan (vel), dan percepatan (acc).
3. **Metode attract()**
   Menghitung gaya tarik dari satu Mover ke Mover lain dengan rumus gravitasi Newton.
4. **Metode apply_force()**
   Gaya dibagi massa → percepatan ditambahkan: $a = \frac{F}{m}$
5. **Metode update()**
   Perbarui kecepatan dan posisi: $v = v + a, \quad p = p + v$
6. **Metode show()**
   Gambar posisi benda di Canvas.

💬 3. Komentar # di Setiap Fungsi dan Hasil Diperlihatkan di Layar

Semua fungsi sudah diberi komentar di kode sebelumnya. Di bawah layar, ditampilkan:

- Posisi benda (pos)
- Kecepatan (V)
- Jarak ke matahari (dSun)

Contoh di layar:
Mover 1: Pos=(-123.4,87.2) V=2.3 dSun=150.2

```
# SCRIPT 21 - Mutual Attraction Simulation
import tkinter as tk
from vector import Vector
import random
import math

# ==== Mover Class ====
class Mover:
    def __init__(self, canvas, x, y, vx, vy, mass):
        self.canvas = canvas
        self.mass = mass
        self.position = Vector(x, y)
        self.velocity = Vector(vx, vy)
        self.acceleration = Vector(0, 0)
        self.radius = mass / 2
        self.body = canvas.create_oval(
            x - self.radius, y - self.radius,
            x + self.radius, y + self.radius,
            fill="skyblue"
        )

    def apply_force(self, force):
        f = force.dived(self.mass)  # F = m * a -> a = F / m
        self.acceleration.add(f)
```

```
    def attract(self, other):
        G = 0.4  # konstanta gravitasi
        force = self.position.subbed(other.position)
        distance = force.mag()
        distance = max(5.0, min(distance, 25.0))  # batas bawah dan atas jarak
        strength = G * self.mass * other.mass / (distance * distance)
        force = force.normalized().multed(strength)
        other.apply_force(force)

    def update(self):
        self.velocity.add(self.acceleration)
        self.position.add(self.velocity)
        self.acceleration = Vector(0, 0)

    def display(self):
        x = self.position.x
        y = self.position.y
        r = self.radius
        self.canvas.coords(self.body, x - r, y - r, x + r, y + r)

# ==== Setup Tkinter ====
WIDTH, HEIGHT = 800, 600
root = tk.Tk()
root.title("Mutual Attraction Simulation")
canvas = tk.Canvas(root, width=WIDTH, height=HEIGHT, bg="black")
canvas.pack()

# ==== Buat Matahari (Sun) di tengah ====
sun = Mover(canvas, WIDTH // 2, HEIGHT // 2, 0, 0, 100)
sun.canvas.itemconfig(sun.body, fill="orange")

# ==== Buat banyak mover mengorbit sun ====
movers = []
for _ in range(30):
    angle = random.uniform(0, 2 * math.pi)
    distance = random.uniform(100, 200)
    x = WIDTH // 2 + math.cos(angle) * distance
    y = HEIGHT // 2 + math.sin(angle) * distance
    vx = -math.sin(angle) * random.uniform(1, 2)
    vy = math.cos(angle) * random.uniform(1, 2)
    mass = random.uniform(5, 10)
    movers.append(Mover(canvas, x, y, vx, vy, mass))

# ==== Loop animasi ====
def animate():
    for mover in movers:
        sun.attract(mover)  # sun menarik semua mover
        for other in movers:
            if mover != other:
                mover.attract(other)  # saling tarik antar mover
        mover.update()
        mover.display()
    root.after(33, animate)

animate()
root.mainloop()
```