Lab: Robot Energy Race

Objective: Learn how energy, efficiency, and small variations in performance affect the outcome of a robot competition.

Part 1: Deterministic Robot

```
class Robot:
    def __init__(self, name, energy, efficiency):
        self.name = name
        self.energy = energy
        self.efficiency = efficiency
        self.distance = 0

    def move(self, speed):
        cost = speed**2 / self.efficiency
        if self.energy >= cost:
            self.energy -= cost
            self.distance += speed
```

Exercise 1

1.1. Create a robot **"**Manel**"** with energy = 100 and efficiency = 10.

1.2. Move at 30 m/s and calculate remaining energy.

1.3. Create a method to verify the energy and distance

```
Hint:

robot1 = Robot("Manel", 100, 10)
robot1.move(30)
print(robot1)
```

Part 2: Robot Competition

```
class RobotCompetition:
    def __init__(self, *robots):
        self.robots = robots

    def start_race(self, steps=10):
        for step in range(steps):
            for r in self.robots:
                r.move(10)  # constant speed

        # Determine the winner without lambda
        winner = self.robots[0]
        for r in self.robots[1:]:
            if r.distance > winner.distance:
                winner = r

        print(f"Winner: {winner.name} ({winner.distance:.1f}m)")
```

Exercise 2

2.1. Create three robots: Tó (energy=100, efficiency=8), Xico (energy=100, efficiency=10), and Manel (energy=100, efficiency=10).

2.2. Run a **10-step race**

2.3. Which robot won?

2.4. How does efficiency affect distance?

Hint:

```
r1 = Robot("Toino", 100, 8)

race = RobotCompetition(r1, r2)

race.start_race(steps=10)
```

Part 3: Robot with Randomness

```
import random
class RandomRobot(Robot):
    def move(self, speed):
        # Add a small adjustment to speed: ±10%
        adjustment = random.uniform(0.9, 1.1)
        adjusted_speed = speed * adjustment

        # Optional: also adjust efficiency slightly
        adjusted_efficiency = self.efficiency * random.uniform(0.95, 1.05)

        cost = adjusted_speed**2 / adjusted_efficiency
        if self.energy >= cost:
            self.energy -= cost
            self.distance += adjusted_speed

        return adjusted_speed
```

Exercise 3

3.1. Replace the deterministic robots with `RandomRobot` instances

3.2. How does randomness influence the winner?

3.3. Compare the total distance of deterministic vs random robots.

3.4. What strategy seems safer: high efficiency or occasional bursts of speed?