import matplotlib.pyplot as plt

import numpy as np

import random

import math

# Speed input (same for both)

s = float(input("Enter speed (e.g., 20): "))

# Random initial positions between 1 and 1000

xf, yf = random.uniform(1, 1000), random.uniform(1, 1000)

xb, yb = random.uniform(1, 1000), random.uniform(1, 1000)

# Time and storage for plotting

time = 0

fighter\_path\_x = [xf]

fighter\_path\_y = [yf]

bomber\_path\_x = [xb]

bomber\_path\_y = [yb]

def distance(x1, y1, x2, y2):

    return math.hypot(x2 - x1, y2 - y1)

print("Initial Positions:")

print(f"Fighter: ({xf:.2f}, {yf:.2f})")

print(f"Bomber: ({xb:.2f}, {yb:.2f})\n")

# Simulation loop

while True:

    d = distance(xf, yf, xb, yb)

    print(f"time={time}   xf={xf:.2f}  yf={yf:.2f}  xb={xb:.2f}  yb={yb:.2f}  distance={d:.2f}")

    if 100 < d < 900:

        print(f"\nThe bomber was destroyed at {time} second(s).")

        break

    elif d <= 100 or d >= 900:

        if d >= 900:

            print(f"\nThe bomber escaped from sight at {time} second(s).")

        else:

            print(f"\nThe bomber was too close and avoided destruction at {time} second(s).")

        break

    # Bomber moves randomly

    xb += random.uniform(-50, 50)

    yb += random.uniform(-50, 50)

    xb = max(1, min(1000, xb))  # Keep inside bounds

    yb = max(1, min(1000, yb))

    # Fighter pursues: move towards bomber

    dx = xb - xf

    dy = yb - yf

    theta = math.atan2(dy, dx)

    xf += s \* math.cos(theta)

    yf += s \* math.sin(theta)

    # Store paths

    fighter\_path\_x.append(xf)

    fighter\_path\_y.append(yf)

    bomber\_path\_x.append(xb)

    bomber\_path\_y.append(yb)

    time += 1

# Plotting paths

plt.figure(figsize=(10, 6))

plt.plot(fighter\_path\_x, fighter\_path\_y, label="Fighter Path", color="blue", marker='o')

plt.plot(bomber\_path\_x, bomber\_path\_y, label="Bomber Path", color="red", linestyle='--', marker='x')

plt.xlabel("X Position")

plt.ylabel("Y Position")

plt.title("Pure Pursuit Simulation: Fighter vs Bomber")

plt.legend()

plt.grid(True)

plt.show()