import matplotlib.pyplot as plt

import numpy as np

# Function to calculate a point on the cubic Bezier curve

def bezier\_point(t, P0, P1, P2, P3):

    x = (1 - t)\*\*3 \* P0[0] + 3 \* (1 - t)\*\*2 \* t \* P1[0] + 3 \* (1 - t) \* t\*\*2 \* P2[0] + t\*\*3 \* P3[0]

    y = (1 - t)\*\*3 \* P0[1] + 3 \* (1 - t)\*\*2 \* t \* P1[1] + 3 \* (1 - t) \* t\*\*2 \* P2[1] + t\*\*3 \* P3[1]

    return (x, y)

# Define control points (can be changed or input manually)

P0 = (0, 0)

P1 = (1, 2)

P2 = (3, 3)

P3 = (4, 0)

# Number of points on the curve

num\_points = 100

# Generate curve points

curve\_points = []

print("Bezier Curve Points:")

for i in range(num\_points + 1):

    t = i / num\_points

    point = bezier\_point(t, P0, P1, P2, P3)

    curve\_points.append(point)

    print(f"t = {t:.2f} -> x = {point[0]:.4f}, y = {point[1]:.4f}")

# Split into X and Y for plotting

x\_vals, y\_vals = zip(\*curve\_points)

# Plotting the curve and control points

plt.figure(figsize=(8, 6))

plt.plot(x\_vals, y\_vals, label="Cubic Bézier Curve", color='blue')

plt.plot(\*zip(P0, P1, P2, P3), 'ro--', label="Control Points")

plt.title("Cubic Bézier Curve Simulation")

plt.xlabel("X")

plt.ylabel("Y")

plt.legend()

plt.grid(True)

plt.axis("equal")

plt.show()