```python
# gradient_descent_linear_regression.py
# CS5710 Machine Learning — Home Assignment 1 — Q7
# Author: Ashmitha Kumbham
# University: University of Central Missouri
# Course: CS5710 Machine Learning (Fall 2025)

import numpy as np
import matplotlib.pyplot as plt

rng = np.random.default_rng(42)

def generate_data(n=200, x_low=0.0, x_high=5.0, true_intercept=3.0, true_slope=4.0, noise_std=1.0):
    x = rng.uniform(x_low, x_high, size=n)
    eps = rng.normal(0.0, noise_std, size=n)
    y = true_intercept + true_slope * x + eps
    return x.reshape(-1, 1), y

def add_bias(X):
    # Add a column of ones for the intercept term
    return np.hstack([np.ones((X.shape[0], 1)), X])

def normal_equation(X, y):
    # Closed-form solution: theta = (X^T X)^(-1) X^T y
    XtX = X.T @ X
    Xty = X.T @ y
    theta = np.linalg.inv(XtX) @ Xty
    return theta  # [intercept, slope]

def mse(y_pred, y_true):
    return np.mean((y_pred - y_true) ** 2)

def gradient_descent(X, y, lr=0.05, iters=1000):
    m, n = X.shape
    theta = np.zeros(n)  # [theta0, theta1]
    history = []
    for t in range(iters):
        y_pred = X @ theta
        residual = y_pred - y
        grad = (2.0 / m) * (X.T @ residual)  # gradient of MSE
        theta -= lr * grad
        history.append(mse(y_pred, y))
    return theta, np.array(history)

def main():
    # Generate synthetic data
    X_raw, y = generate_data(n=200, x_low=0.0, x_high=5.0, true_intercept=3.0, true_slope=4.0, noise_std=1.0)
    X = add_bias(X_raw)  # shape (m, 2) with first column ones

    # Closed-form (Normal Equation)
    theta_ne = normal_equation(X, y)
    intercept_ne, slope_ne = theta_ne[0], theta_ne[1]
    print(f"[Normal Equation] intercept={intercept_ne:.4f}, slope={slope_ne:.4f}")

    # Gradient Descent
    theta_gd, loss_hist = gradient_descent(X, y, lr=0.05, iters=1000)
    intercept_gd, slope_gd = theta_gd[0], theta_gd[1]
    print(f"[Gradient Descent] intercept={intercept_gd:.4f}, slope={slope_gd:.4f}")
    print(f"Final MSE (GD): {loss_hist[-1]:.4f}")

    # Predictions for plotting lines
    x_line = np.linspace(X_raw.min(), X_raw.max(), 200).reshape(-1, 1)
    X_line = add_bias(x_line)
    y_line_ne = X_line @ theta_ne
    y_line_gd = X_line @ theta_gd

    # Figure 1: raw data + both fitted lines
    plt.figure()
    plt.scatter(X_raw, y, s=15, label="Raw data")
    plt.plot(x_line, y_line_ne, label="Closed-form fit")
    plt.plot(x_line, y_line_gd, linestyle="--", label="GD fit")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.title("Linear Regression: Data and Fitted Lines")
    plt.legend()
    plt.tight_layout()
    plt.savefig("fig_data_and_fits.png", dpi=180)
```

```
    # Figure 2: loss curve for Gradient Descent
    plt.figure()
    plt.plot(np.arange(len(loss_hist)), loss_hist)
    plt.xlabel("Iteration")
    plt.ylabel("MSE")
    plt.title("Gradient Descent: Loss vs Iterations")
    plt.tight_layout()
    plt.savefig("fig_loss_curve.png", dpi=180)

if __name__ == "__main__":
    main()
```

```
[Normal Equation] intercept=2.6908, slope=4.1318
[Gradient Descent] intercept=2.6908, slope=4.1318
Final MSE (GD): 0.9958
```