```python
import numpy as np
import sys

class TestNN:
    def __init__(self, input_size=1, hidden_nodes=10, output_size=1):
        np.random.seed(1)
        self.input_size = input_size
        self.hidden_nodes = hidden_nodes
        self.output_size = output_size

        self.w1 = np.random.randn(self.input_size, self.hidden_nodes) * 0.01
        self.b1 = np.zeros((1, self.hidden_nodes))
        self.w2 = np.random.randn(self.hidden_nodes, self.output_size) * 0.01
        self.b2 = np.zeros((1, self.output_size))

    def sig(self, z):
        return 1 / (1 + np.exp(-z))

    def sig_dev(self, z):
        return self.sig(z) * (1 - self.sig(z))

    def fw(self, X):
        self.z1 = np.dot(X, self.w1) + self.b1
        self.a1 = self.sig(self.z1)
        self.z2 = np.dot(self.a1, self.w2) + self.b2
        return self.z2

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

    def bw(self, X, y, y_pred):
        n_samples = X.shape[0]

        dL_dz2 = 2 * (y_pred - y) / n_samples
        dL_dw2 = np.dot(self.a1.T, dL_dz2)
        dL_db2 = np.sum(dL_dz2, axis=0, keepdims=True)

        dL_da1 = np.dot(dL_dz2, self.w2.T)
        dL_dz1 = dL_da1 * self.sig_dev(self.z1)
        dL_dw1 = np.dot(X.T, dL_dz1)
        dL_db1 = np.sum(dL_dz1, axis=0, keepdims=True)

        return dL_dw1, dL_db1, dL_dw2, dL_db2

    def update_parameters(self, grads, lrning_rate):
        dL_dw1, dL_db1, dL_dw2, dL_db2 = grads

        self.w1 -= lrning_rate * dL_dw1
        self.b1 -= lrning_rate * dL_db1
        self.w2 -= lrning_rate * dL_dw2
        self.b2 -= lrning_rate * dL_db2

    def trn(self, X, Y, max_iters=5000, lrning_rate=0.01, improvement_threshold=1e-6):
        prev_diff = float('inf')
        diffes = []
        for iter in range(max_iters):
            y_pred = self.fw(X)
            diff = self.compute_diff(y_pred, Y)
            diffes.append(diff)

            if (iter + 1) % 10000 == 0:
                print(f"Iter {iter + 1}/{max_iters}, Diff: {diff:.6f}")
                print(f"w1 mean: {np.mean(self.w1):.5f}, std: {np.std(self.w1):.5f}")
                print(f"b1 mean: {np.mean(self.b1):.5f}, std: {np.std(self.b1):.5f}")
                print(f"w2 mean: {np.mean(self.w2):.5f}, std: {np.std(self.w2):.5f}")
                print(f"b2 mean: {np.mean(self.b2):.5f}, std: {np.std(self.b2):.5f}")
                print(f"Grad Norm: {np.linalg.norm(self.w1) + np.linalg.norm(self.w2):.5f¬
}")

            grads = self.bw(X, Y, y_pred)
            self.update_parameters(grads, lrning_rate)

            if 0:
                if prev_diff - diff < improvement_threshold and iter > 500:
                    print(f"Stopping early at iter {iter + 1}, Diff: {diff:.6f}")
                    break

            prev_diff = diff

if __name__ == "__main__":
    def load_data(filename):
        X_list, Y_list = [], []
        with open(filename, 'r') as f:
            for line in f:
                if line.strip():
                    x, y = map(float, line.split())
                    X_list.append(x)
                    Y_list.append(y)

        X = np.array(X_list).reshape(-1, 1)
        Y = np.array(Y_list).reshape(-1, 1)

        X_min, X_max = X.min(), X.max()
        Y_min, Y_max = Y.min(), Y.max()
        X = (X - X_min) / (X_max - X_min)
        Y = (Y - Y_min) / (Y_max - Y_min)

        return X, Y, X_min, X_max, Y_min, Y_max

    if len(sys.argv) != 2:
        print("Usage: python script.py <data_file>")
        sys.exit(1)

    data_file = sys.argv[1]
    print(f"data_file = {data_file}")
    X, Y, X_min, X_max, Y_min, Y_max = load_data(data_file)
    print("First few samples (original and normalized):")
    for i in range(min(5, len(X))):
        x_original = X[i][0] * (X_max - X_min) + X_min
        y_original = Y[i][0] * (Y_max - Y_min) + Y_min
        print(f"x(normalized)={X[i][0]:.5f}, y(normalized)={Y[i][0]:.5f} | x(original)={x¬
_original:.5f}, y(original)={y_original:.5f}")

    nn = TestNN(input_size=1, hidden_nodes=10, output_size=1)
    nn.trn(X, Y, max_iters=300000, lrning_rate=0.01)

    print("First few predions (for verification):")
    for i in range(min(5, len(X))):
        y_pred = nn.fw(X[i].reshape(1, -1)) * (Y_max - Y_min) + Y_min
        x_original = X[i][0] * (X_max - X_min) + X_min
        print(f"x={x_original:.5f}, Preded y={y_pred[0][0]:.5f}")

    while True:
        user_input = input("Enter x (original scale) to pred y (or 'quit' to exit): ")
        if user_input.lower() == 'quit':
            break
        try:
            x_new = float(user_input)
            x_new_norm = (x_new - X_min) / (X_max - X_min)
            x_new_array = np.array([[x_new_norm]])
            y_pred_norm = nn.fw(x_new_array)
            y_pred = y_pred_norm * (Y_max - Y_min) + Y_min
            print(f"Preded y for x={x_new}: {y_pred[0][0]:.5f}")
        except ValueError:
            print("Invalid input. Please enter a numeric value.")
```