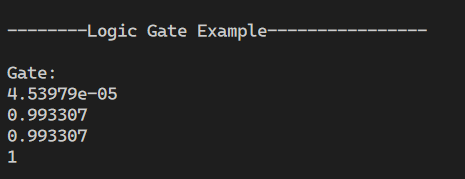
This code defines a simple implementation of a Multi-Layer Perceptron (MLP) neural network and includes a Backpropagation (BP) algorithm for training.

Here's a description of the key components and functionalities:

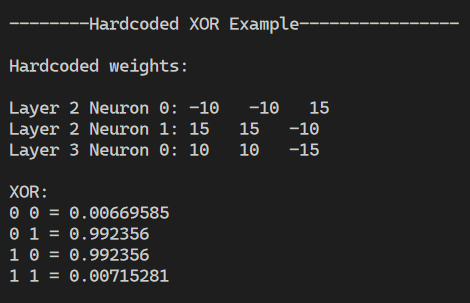
1. **Perceptron Class**: This class represents a single perceptron node in the neural network. It has methods for initialization, running (calculating output), setting weights, and applying the sigmoid activation function.
2. **MultiLayerPerceptron Class**: This class represents the entire neural network. It is initialized with parameters such as the number of layers, bias, and learning rate (eta). It has methods for setting weights, running input through the network, displaying weights, and implementing backpropagation.
3. **Initialization**: Perceptrons are initialized with random values. For each perceptron, a set of random weights is generated using the frand() function.
4. **Running**: Input is run through a neural network, with bias added to the input, and the weighted sum of inputs for each perceptron is calculated. This sum is then passed through a sigmoid activation function to obtain the output.
5. **Training with Backpropagation (BP)**: The BP algorithm is implemented in the bp() method. It consists of several steps:
   * Feed an input through the neural network and calculate the Mean Squared Error (MSE).
   * Calculate error terms for the network outputs and for the hidden units using the chain rule-derived formula.
   * Calculate weight deltas and update weights based on these error terms and the learning rate.

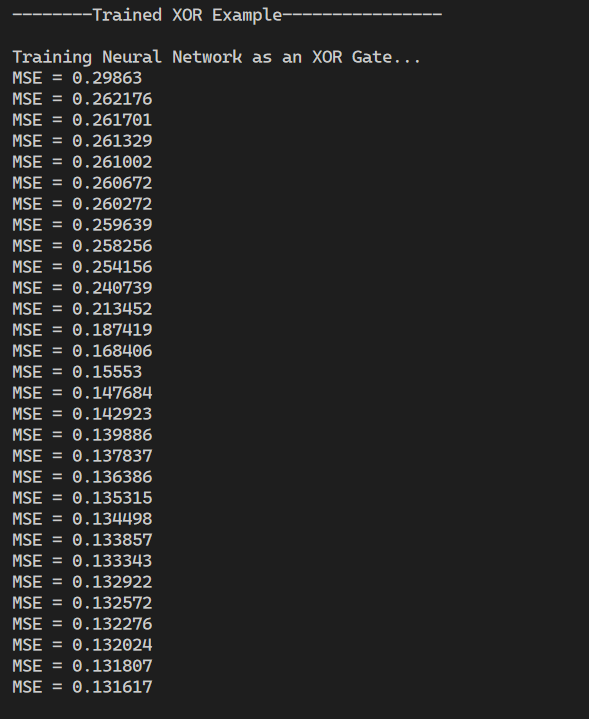
This implementation provides a solid foundation for building and training a multi-layer perceptron neural network for supervised learning purposes.

Results:

Implementing a Logic OR GATE.   


Implementing an XOR Logical GATE





* This section indicates the training process of the neural network to act as an XOR gate.
* The Mean Squared Error (MSE) decreases over successive iterations, indicating the network is learning.
* Trained weights for each neuron in each layer are shown. These weights are compared to the hardcoded weights.
* Finally, the outputs for each combination of inputs (0 0, 0 1, 1 0, 1 1) after training are displayed.

This output demonstrates the process of training a neural network to perform the XOR operation, comparing the results with both hardcoded weights and trained weights.

