In [3]:

```
import numpy as np

class NeuralNetwork:
    def __init__(self, input_layer_size, hidden_layer_size, ou
        self.input_layer_size = input_layer_size
        self.hidden_layer_size = hidden_layer_size
        self.output_layer_size = output_layer_size

        # Initialize weights with random values between -1 and
        self.W1 = np.random.uniform(low=-1, high=1, size=(self
        self.W2 = np.random.uniform(low=-1, high=1, size=(self

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

    def sigmoid_derivative(self, z):
        return z * (1 - z)

    def train(self, X, y, num_iterations, learning_rate):
        for i in range(num_iterations):
            # Forward propagation
            z1 = np.dot(X, self.W1)
            a1 = self.sigmoid(z1)
            z2 = np.dot(a1, self.W2)
            y_hat = self.sigmoid(z2)

            # Backward propagation
            delta3 = (y_hat - y) * self.sigmoid_derivative(y_h
            dW2 = np.dot(a1.T, delta3)
            delta2 = np.dot(delta3, self.W2.T) * self.sigmoid_
            dW1 = np.dot(X.T, delta2)

            # Update weights
            self.W1 -= learning_rate * dW1
            self.W2 -= learning_rate * dW2

    def predict(self, X):
        z1 = np.dot(X, self.W1)
        a1 = self.sigmoid(z1)
        z2 = np.dot(a1, self.W2)
        y_hat = self.sigmoid(z2)
        return y_hat

```

In [4]:

```
# Define XOR dataset with binary input and output
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([[0], [1], [1], [0]])

# Set learning rate and number of iterations
learning_rate = 0.1
num_iterations = 100000

# Train the neural network using backpropagation
nn = NeuralNetwork(2, 3, 1)
nn.train(X, y, num_iterations, learning_rate)

# Make predictions on new data and print output
new_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
for i in range(len(new_data)):
    output = nn.predict(new_data[i])
    print(f"Input: {new_data[i]}, Output: {output}")

```

```
Input: [0 0], Output: [0.00412306]
Input: [0 1], Output: [0.92656054]
Input: [1 0], Output: [0.92656642]
Input: [1 1], Output: [0.10162852]
```

In [ ]:

```

```