```python
import numpy as np

# Activation function (Sigmoid) and its derivative
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):
    return x * (1 - x)

# Input dataset (AND gate example)
X = np.array([[1, 1]])
y = np.array([[1]])

# Set random seed for reproducibility
np.random.seed(1)

# Initialize weights
input_to_hidden_weights = np.random.rand(2, 2)   # 2 inputs → 2 hidden neurons
hidden_to_output_weights = np.random.rand(2, 1)  # 2 hidden neurons → 1 output

# Training loop
for epoch in range(10):  # simple 10 epochs
    # ----------- Forward Pass -----------
    hidden_input = np.dot(X, input_to_hidden_weights)
    hidden_output = sigmoid(hidden_input)

    final_input = np.dot(hidden_output, hidden_to_output_weights)
    predicted_output = sigmoid(final_input)

    # ----------- Backward Pass -----------
    error = y - predicted_output
    print(f"Epoch {epoch+1}, Error: {error[0][0]:.4f}")

    # Calculate gradient for output layer
```

```
    d_output = error * sigmoid_derivative(predicted_output)

    # Calculate gradient for hidden layer
    error_hidden = d_output.dot(hidden_to_output_weights.T)
    d_hidden = error_hidden * sigmoid_derivative(hidden_output)

    # ----------- Update Weights -----------
    hidden_to_output_weights += hidden_output.T.dot(d_output)
    input_to_hidden_weights += X.T.dot(d_hidden)

# Final Output
print("Predicted Output after training:", predicted_output)
```

```
Epoch 1, Error: 0.4610
Epoch 2, Error: 0.4352
Epoch 3, Error: 0.4113
Epoch 4, Error: 0.3890
Epoch 5, Error: 0.3685
Epoch 6, Error: 0.3495
Epoch 7, Error: 0.3321
Epoch 8, Error: 0.3160
Epoch 9, Error: 0.3012
Epoch 10, Error: 0.2876
Predicted Output after training: [[0.71237374]]
```

Double-click (or enter) to edit