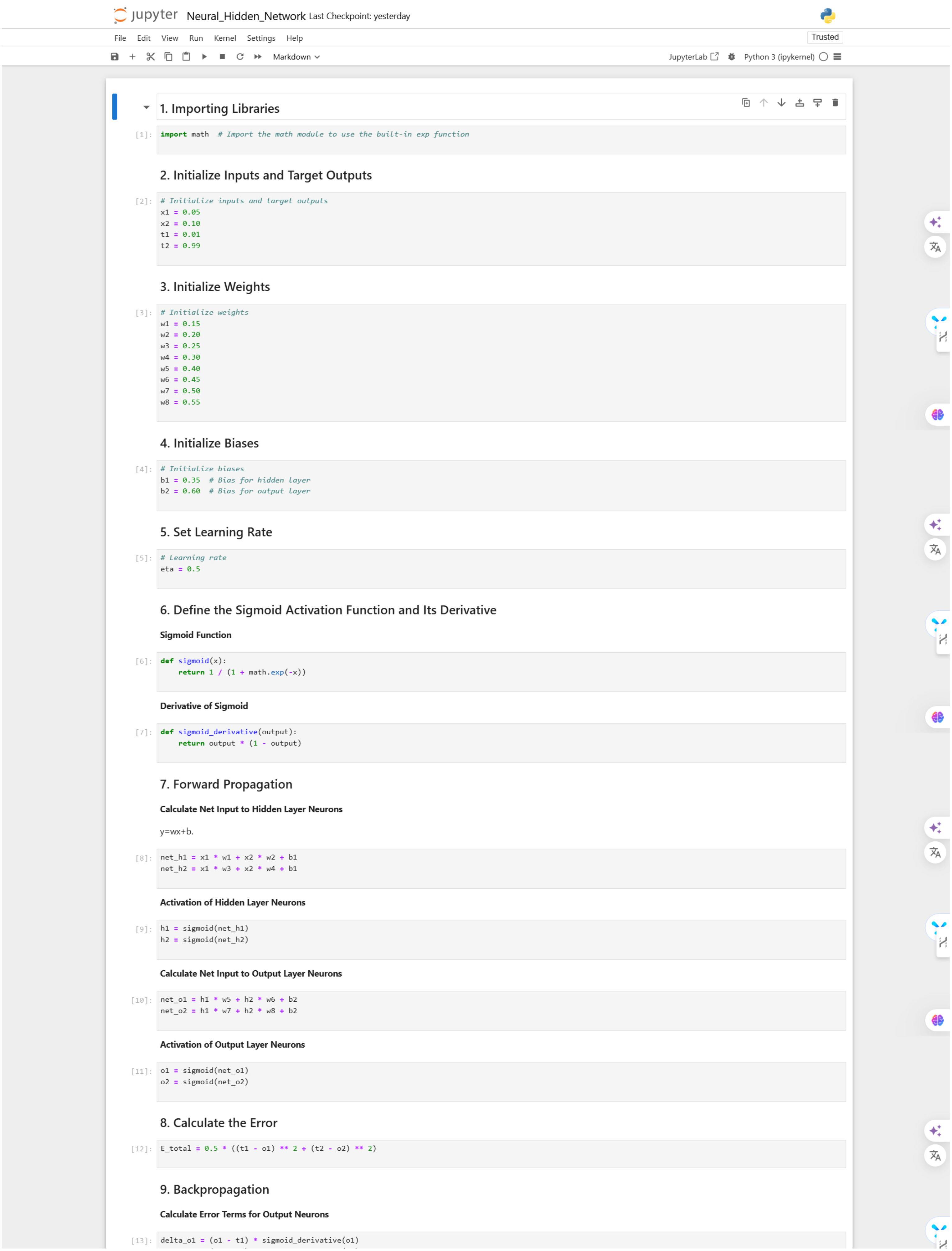


1. Importing Libraries

[1]:
```python
import math  # Import the math module to use the built-in exp function
```

2. Initialize Inputs and Target Outputs

[2]:
```python
# Initialize inputs and target outputs
x1 = 0.05
x2 = 0.10
t1 = 0.01
t2 = 0.99
```

3. Initialize Weights

[3]:
```python
# Initialize weights
w1 = 0.15
w2 = 0.20
w3 = 0.25
w4 = 0.30
w5 = 0.40
w6 = 0.45
w7 = 0.50
w8 = 0.55
```

4. Initialize Biases

[4]:
```python
# Initialize biases
b1 = 0.35  # Bias for hidden layer
b2 = 0.60  # Bias for output layer
```

5. Set Learning Rate

[5]:
```python
# Learning rate
eta = 0.5
```

6. Define the Sigmoid Activation Function and Its Derivative

Sigmoid Function

[6]:
```python
def sigmoid(x):
    return 1 / (1 + math.exp(-x))
```

Derivative of Sigmoid

[7]:
```python
def sigmoid_derivative(output):
    return output * (1 - output)
```

7. Forward Propagation

Calculate Net Input to Hidden Layer Neurons

y=wx+b.

[8]:
```python
net_h1 = x1 * w1 + x2 * w2 + b1
net_h2 = x1 * w3 + x2 * w4 + b1
```

Activation of Hidden Layer Neurons

[9]:
```python
h1 = sigmoid(net_h1)
h2 = sigmoid(net_h2)
```

Calculate Net Input to Output Layer Neurons

[10]:
```python
net_o1 = h1 * w5 + h2 * w6 + b2
net_o2 = h1 * w7 + h2 * w8 + b2
```

Activation of Output Layer Neurons

[11]:
```python
o1 = sigmoid(net_o1)
o2 = sigmoid(net_o2)
```

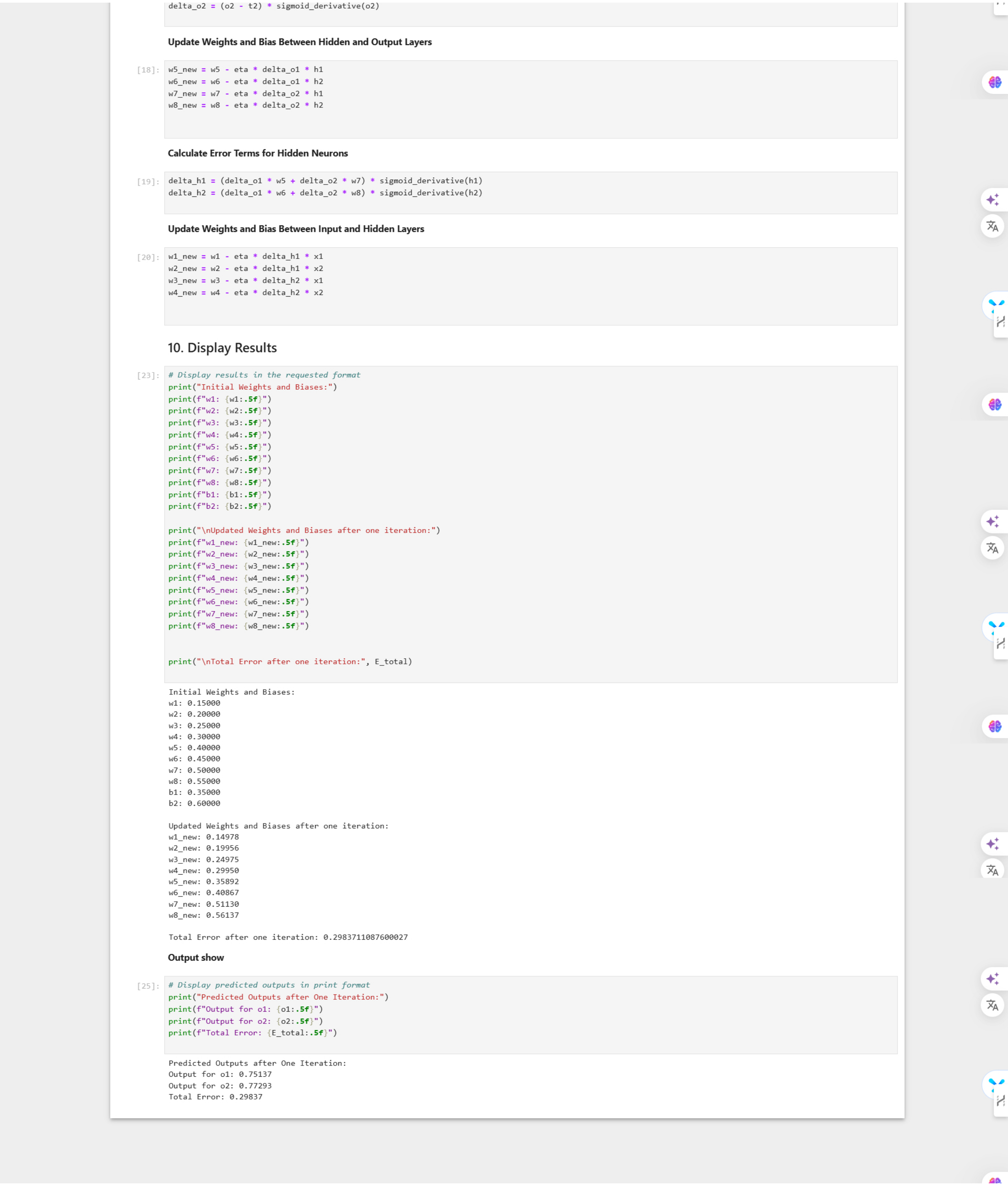
8. Calculate the Error

[12]:
```python
E_total = 0.5 * ((t1 - o1) ** 2 + (t2 - o2) ** 2)
```

9. Backpropagation

Calculate Error Terms for Output Neurons

[13]:
```python
delta_o1 = (o1 - t1) * sigmoid_derivative(o1)
```



```
delta_o2 = (o2 - t2) * sigmoid_derivative(o2)
```

Update Weights and Bias Between Hidden and Output Layers

[18]:
```
w5_new = w5 - eta * delta_o1 * h1
w6_new = w6 - eta * delta_o1 * h2
w7_new = w7 - eta * delta_o2 * h1
w8_new = w8 - eta * delta_o2 * h2
```

Calculate Error Terms for Hidden Neurons

[19]:
```
delta_h1 = (delta_o1 * w5 + delta_o2 * w7) * sigmoid_derivative(h1)
delta_h2 = (delta_o1 * w6 + delta_o2 * w8) * sigmoid_derivative(h2)
```

Update Weights and Bias Between Input and Hidden Layers

[20]:
```
w1_new = w1 - eta * delta_h1 * x1
w2_new = w2 - eta * delta_h1 * x2
w3_new = w3 - eta * delta_h2 * x1
w4_new = w4 - eta * delta_h2 * x2
```

10. Display Results

[23]:
```
# Display results in the requested format
print("Initial Weights and Biases:")
print(f"w1: {w1:.5f}")
print(f"w2: {w2:.5f}")
print(f"w3: {w3:.5f}")
print(f"w4: {w4:.5f}")
print(f"w5: {w5:.5f}")
print(f"w6: {w6:.5f}")
print(f"w7: {w7:.5f}")
print(f"w8: {w8:.5f}")
print(f"b1: {b1:.5f}")
print(f"b2: {b2:.5f}")

print("\nUpdated Weights and Biases after one iteration:")
print(f"w1_new: {w1_new:.5f}")
print(f"w2_new: {w2_new:.5f}")
print(f"w3_new: {w3_new:.5f}")
print(f"w4_new: {w4_new:.5f}")
print(f"w5_new: {w5_new:.5f}")
print(f"w6_new: {w6_new:.5f}")
print(f"w7_new: {w7_new:.5f}")
print(f"w8_new: {w8_new:.5f}")


print("\nTotal Error after one iteration:", E_total)
```

```
Initial Weights and Biases:
w1: 0.15000
w2: 0.20000
w3: 0.25000
w4: 0.30000
w5: 0.40000
w6: 0.45000
w7: 0.50000
w8: 0.55000
b1: 0.35000
b2: 0.60000

Updated Weights and Biases after one iteration:
w1_new: 0.14978
w2_new: 0.19956
w3_new: 0.24975
w4_new: 0.29950
w5_new: 0.35892
w6_new: 0.40867
w7_new: 0.51130
w8_new: 0.56137

Total Error after one iteration: 0.2983711087600027
```

Output show

[25]:
```
# Display predicted outputs in print format
print("Predicted Outputs after One Iteration:")
print(f"Output for o1: {o1:.5f}")
print(f"Output for o2: {o2:.5f}")
print(f"Total Error: {E_total:.5f}")
```

```
Predicted Outputs after One Iteration:
Output for o1: 0.75137
Output for o2: 0.77293
Total Error: 0.29837
```