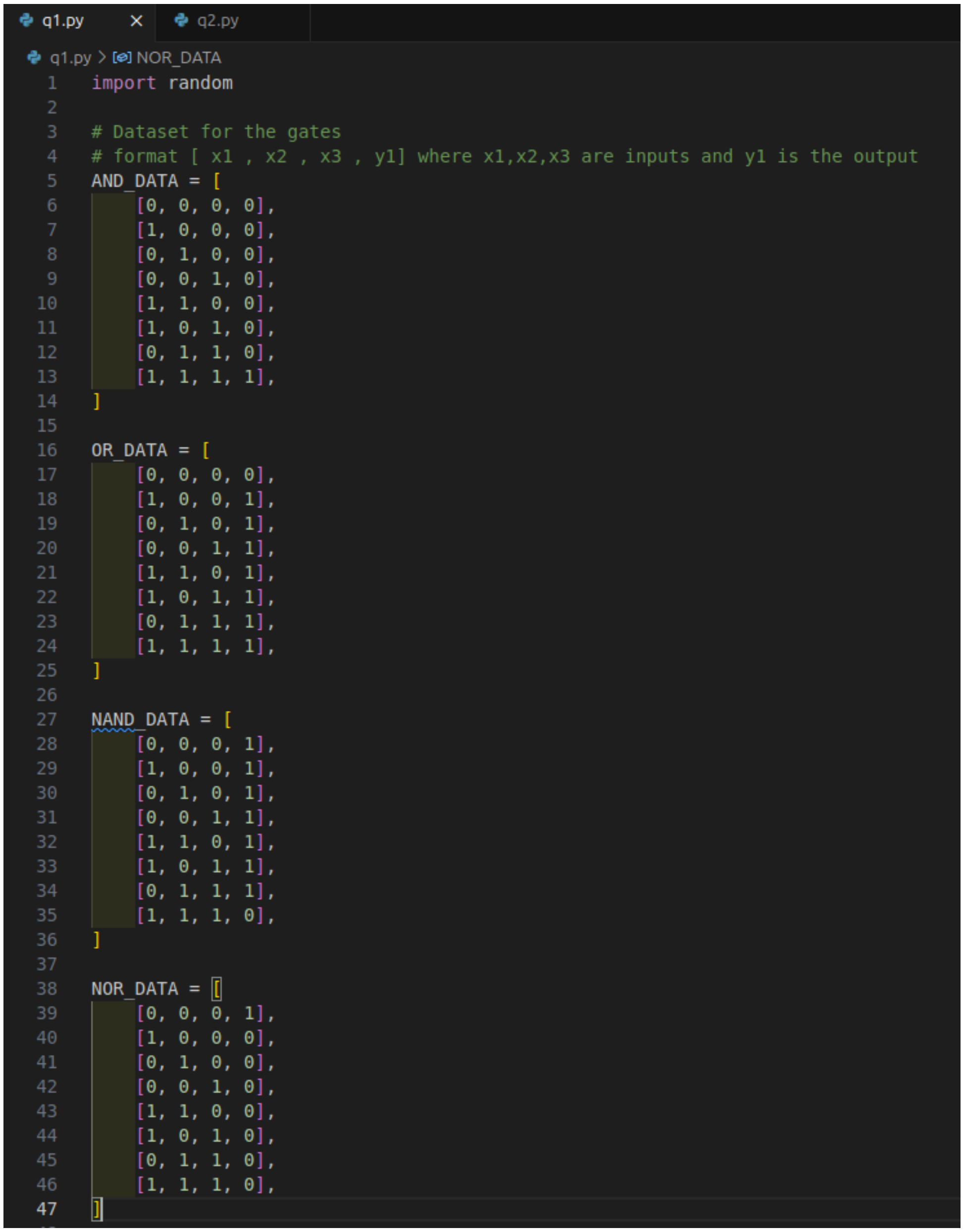
CSE 354 - LAB 5

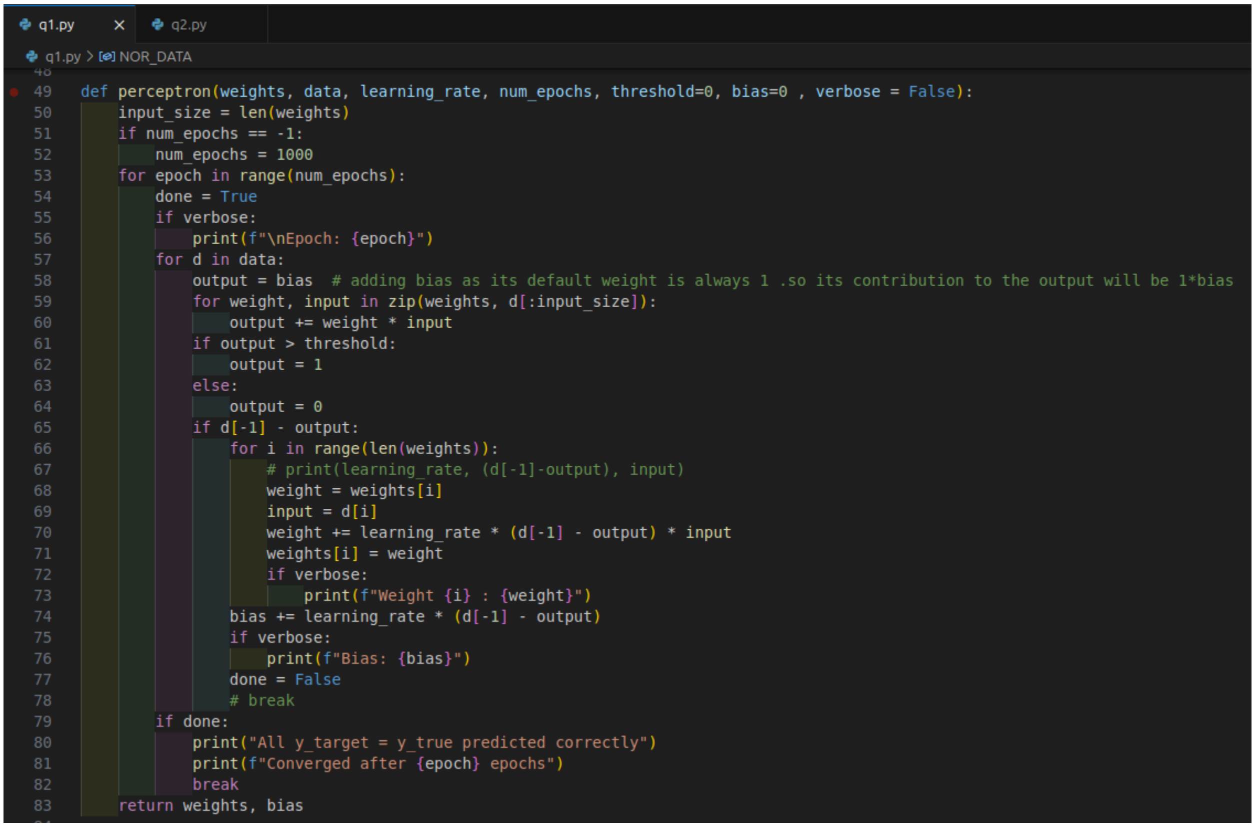
Arnav Jain - 220002018

Question 1

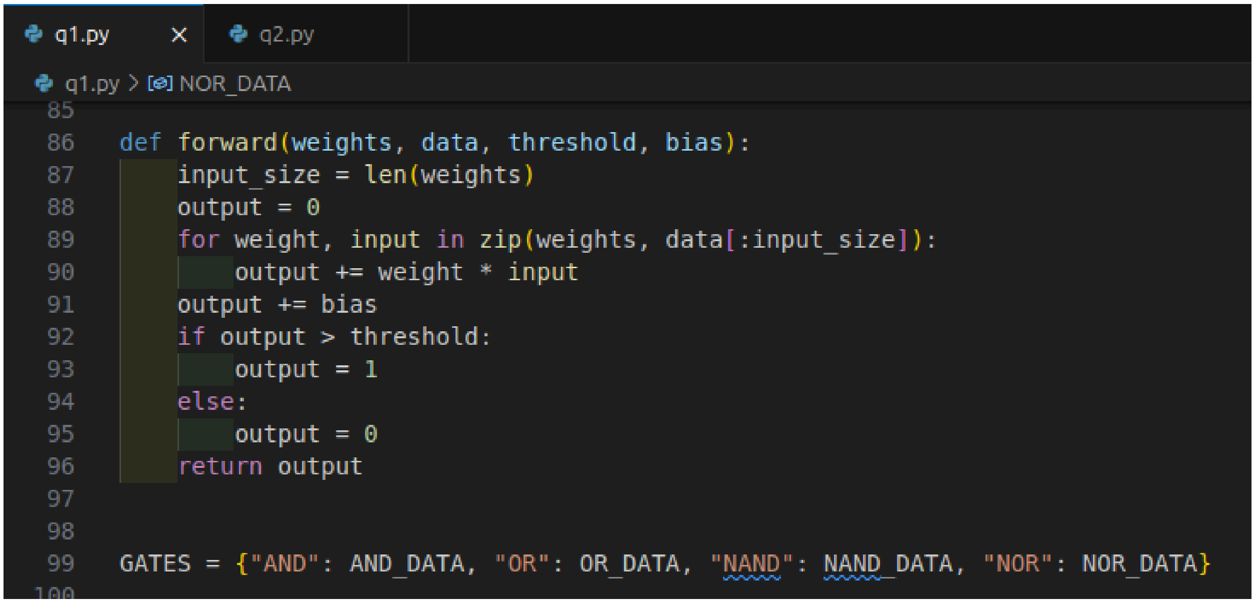
Code:



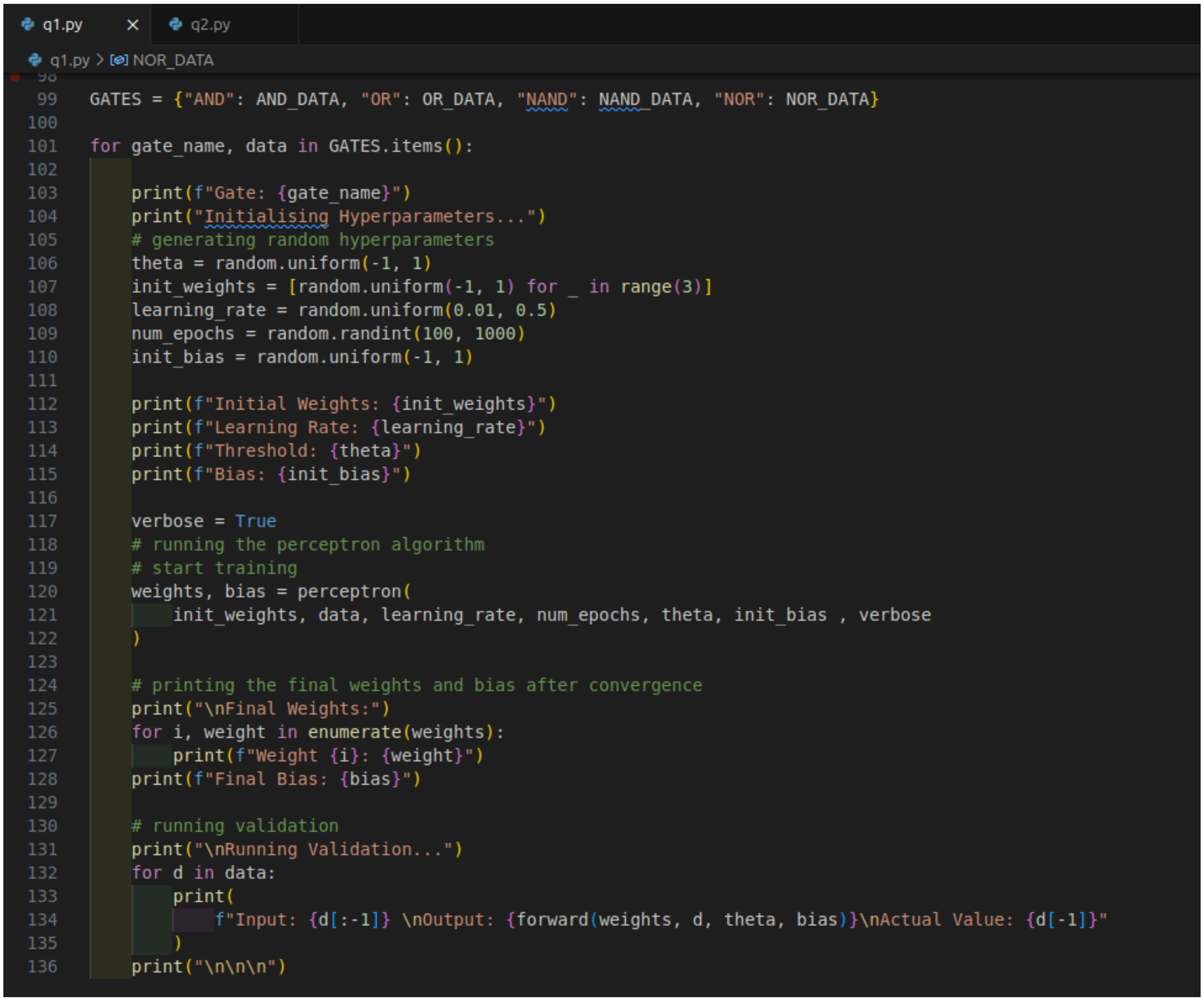
```python
import random

# Dataset for the gates
# format [ x1 , x2 , x3 , y1] where x1,x2,x3 are inputs and y1 is the output
AND_DATA = [
        [0, 0, 0, 0],
        [1, 0, 0, 0],
        [0, 1, 0, 0],
        [0, 0, 1, 0],
        [1, 1, 0, 0],
        [1, 0, 1, 0],
        [0, 1, 1, 0],
        [1, 1, 1, 1],
]

OR_DATA = [
        [0, 0, 0, 0],
        [1, 0, 0, 1],
        [0, 1, 0, 1],
        [0, 0, 1, 1],
        [1, 1, 0, 1],
        [1, 0, 1, 1],
        [0, 1, 1, 1],
        [1, 1, 1, 1],
]

NAND_DATA = [
        [0, 0, 0, 1],
        [1, 0, 0, 1],
        [0, 1, 0, 1],
        [0, 0, 1, 1],
        [1, 1, 0, 1],
        [1, 0, 1, 1],
        [0, 1, 1, 1],
        [1, 1, 1, 0],
]

NOR_DATA = [
        [0, 0, 0, 1],
        [1, 0, 0, 0],
        [0, 1, 0, 0],
        [0, 0, 1, 0],
        [1, 1, 0, 0],
        [1, 0, 1, 0],
        [0, 1, 1, 0],
        [1, 1, 1, 0],
]
```



```python
def perceptron(weights, data, learning_rate, num_epochs, threshold=0, bias=0 , verbose = False):
    input_size = len(weights)
    if num_epochs == -1:
        num_epochs = 1000
    for epoch in range(num_epochs):
        done = True
        if verbose:
            print(f"\nEpoch: {epoch}")
        for d in data:
            output = bias  # adding bias as its default weight is always 1 .so its contribution to the output will be 1*bias
            for weight, input in zip(weights, d[:input_size]):
                output += weight * input
            if output > threshold:
                output = 1
            else:
                output = 0
            if d[-1] - output:
                for i in range(len(weights)):
                    # print(learning_rate, (d[-1]-output), input)
                    weight = weights[i]
                    input = d[i]
                    weight += learning_rate * (d[-1] - output) * input
                    weights[i] = weight
                    if verbose:
                        print(f"Weight {i} : {weight}")
                bias += learning_rate * (d[-1] - output)
                if verbose:
                    print(f"Bias: {bias}")
                done = False
                # break
        if done:
            print("All y_target = y_true predicted correctly")
            print(f"Converged after {epoch} epochs")
            break
    return weights, bias
```

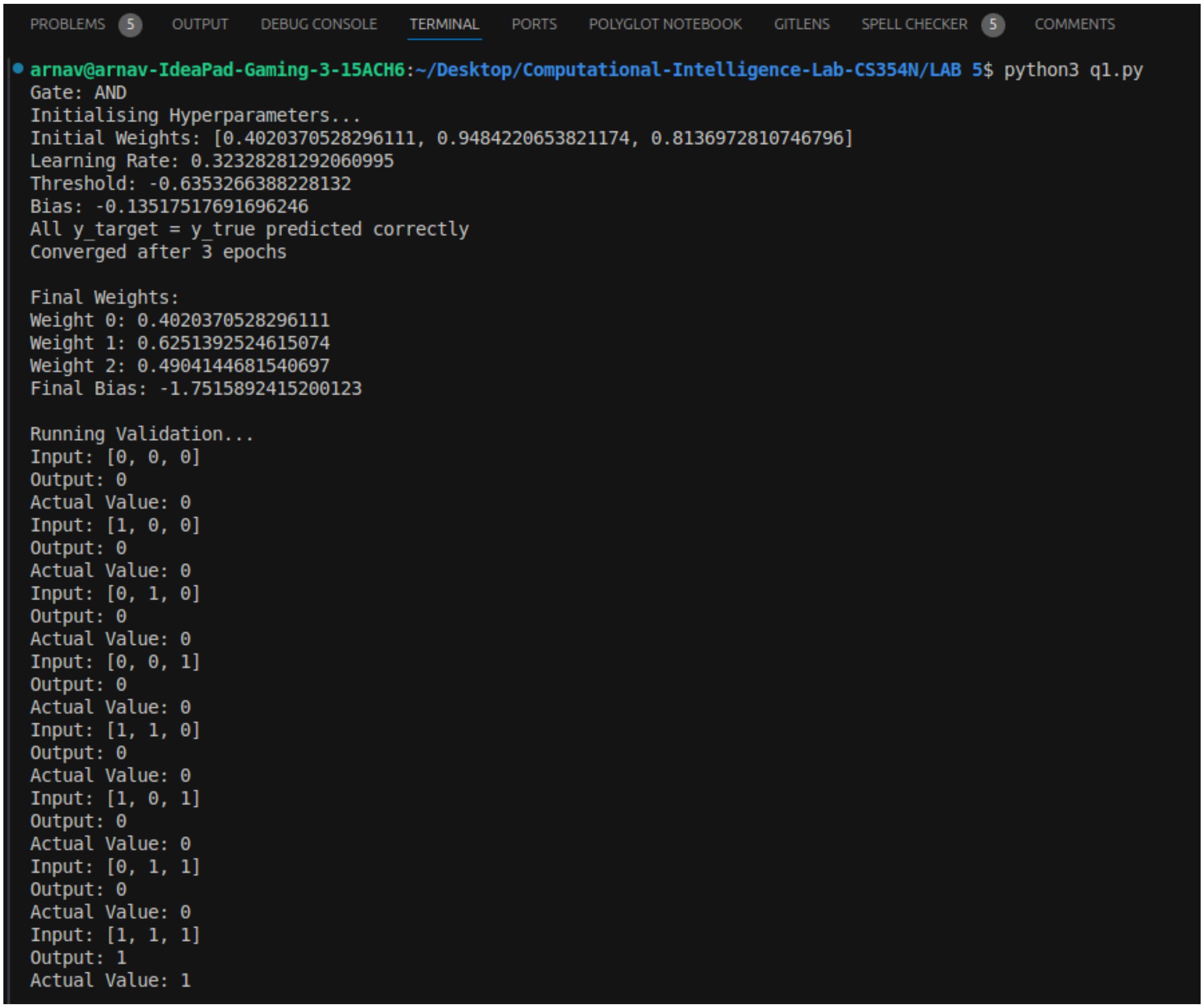


```python

def forward(weights, data, threshold, bias):
    input_size = len(weights)
    output = 0
    for weight, input in zip(weights, data[:input_size]):
        output += weight * input
    output += bias
    if output > threshold:
        output = 1
    else:
        output = 0
    return output


GATES = {"AND": AND_DATA, "OR": OR_DATA, "NAND": NAND_DATA, "NOR": NOR_DATA}
```

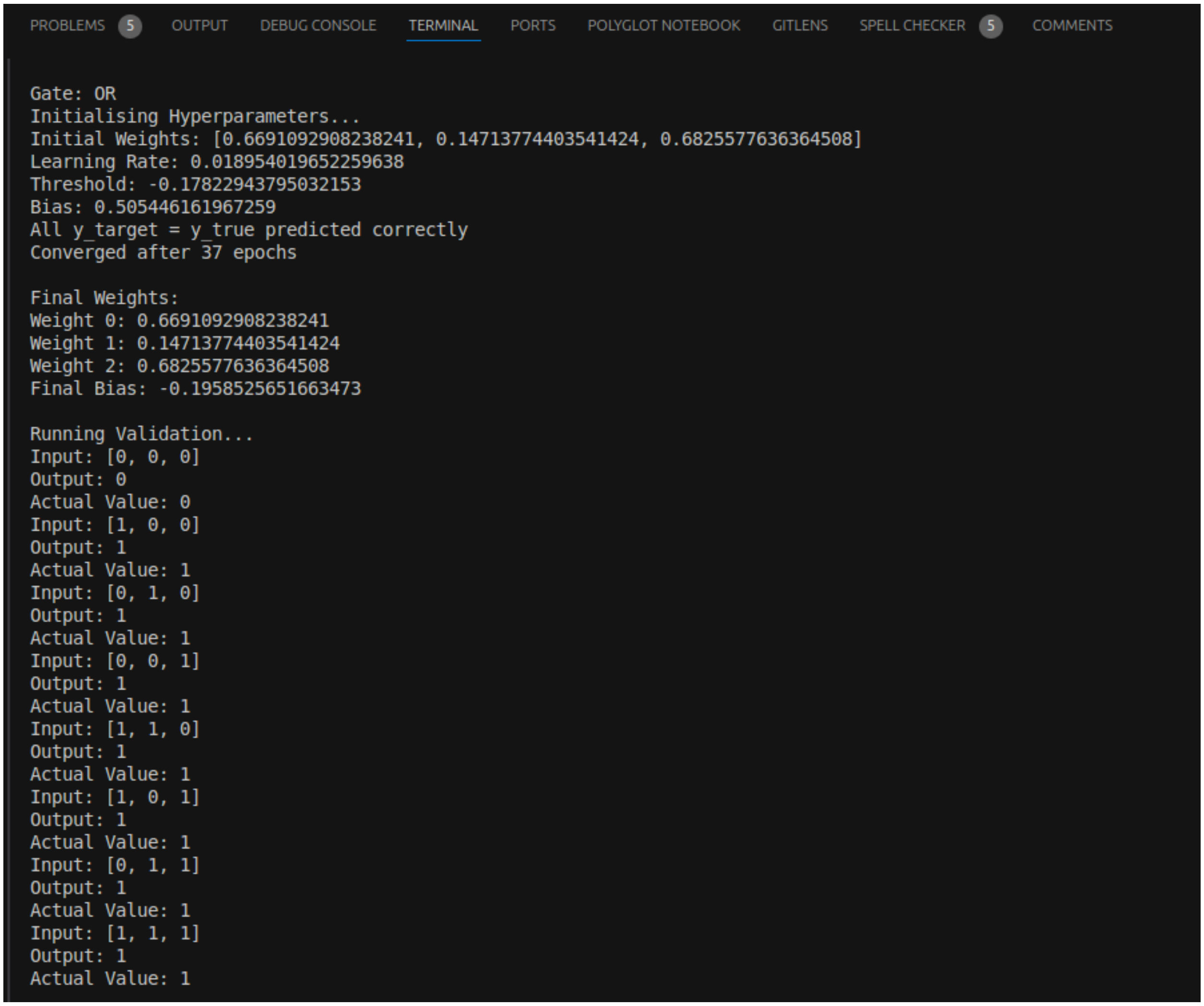


```python
GATES = {"AND": AND_DATA, "OR": OR_DATA, "NAND": NAND_DATA, "NOR": NOR_DATA}

for gate_name, data in GATES.items():

    print(f"Gate: {gate_name}")
    print("Initialising Hyperparameters...")
    # generating random hyperparameters
    theta = random.uniform(-1, 1)
    init_weights = [random.uniform(-1, 1) for _ in range(3)]
    learning_rate = random.uniform(0.01, 0.5)
    num_epochs = random.randint(100, 1000)
    init_bias = random.uniform(-1, 1)

    print(f"Initial Weights: {init_weights}")
    print(f"Learning Rate: {learning_rate}")
    print(f"Threshold: {theta}")
    print(f"Bias: {init_bias}")

    verbose = True
    # running the perceptron algorithm
    # start training
    weights, bias = perceptron(
        init_weights, data, learning_rate, num_epochs, theta, init_bias , verbose
    )

    # printing the final weights and bias after convergence
    print("\nFinal Weights:")
    for i, weight in enumerate(weights):
        print(f"Weight {i}: {weight}")
    print(f"Final Bias: {bias}")

    # running validation
    print("\nRunning Validation...")
    for d in data:
        print(
            f"Input: {d[:-1]} \nOutput: {forward(weights, d, theta, bias)}\nActual Value: {d[-1]}"
        )
    print("\n\n\n")
```

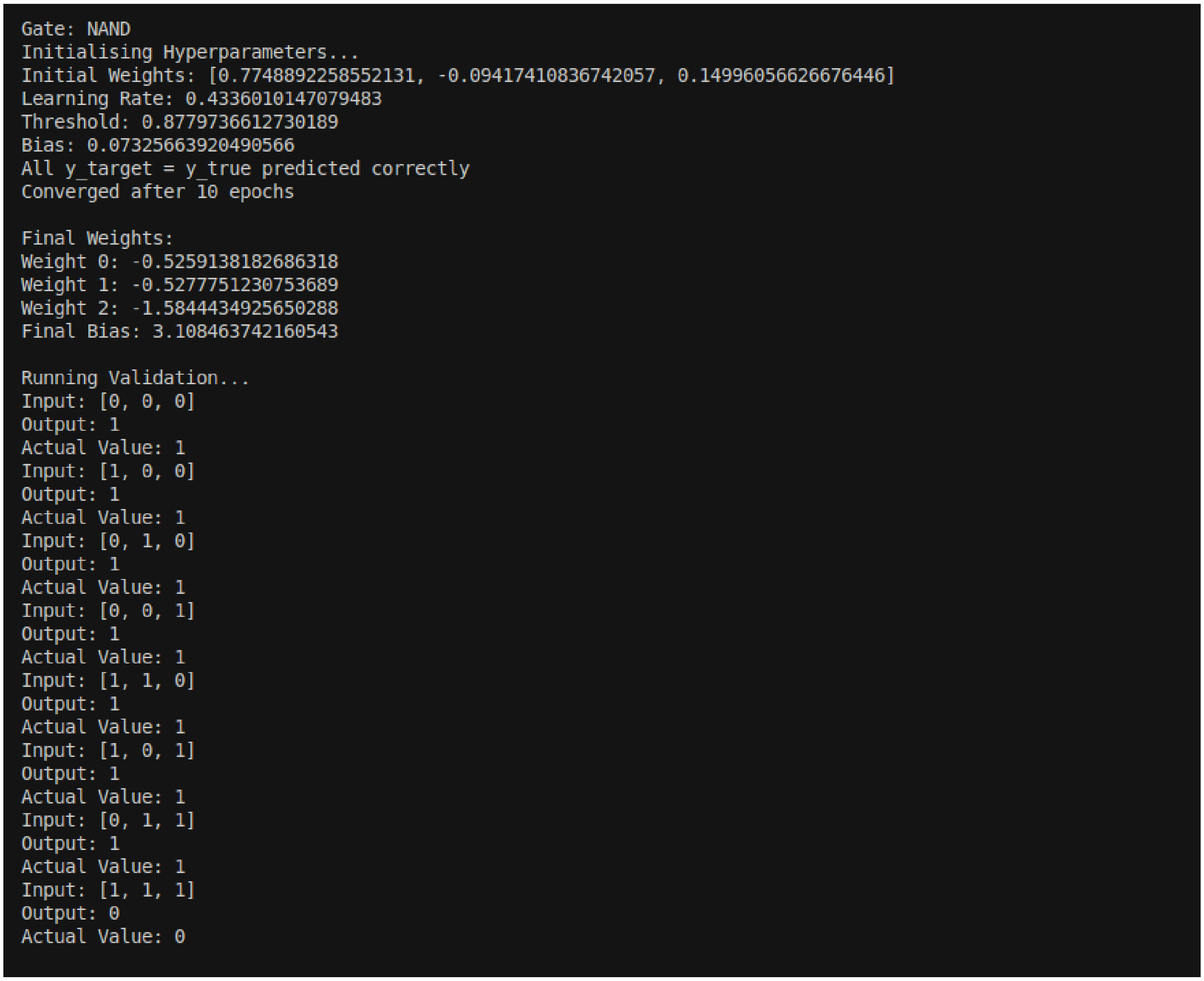
Result:



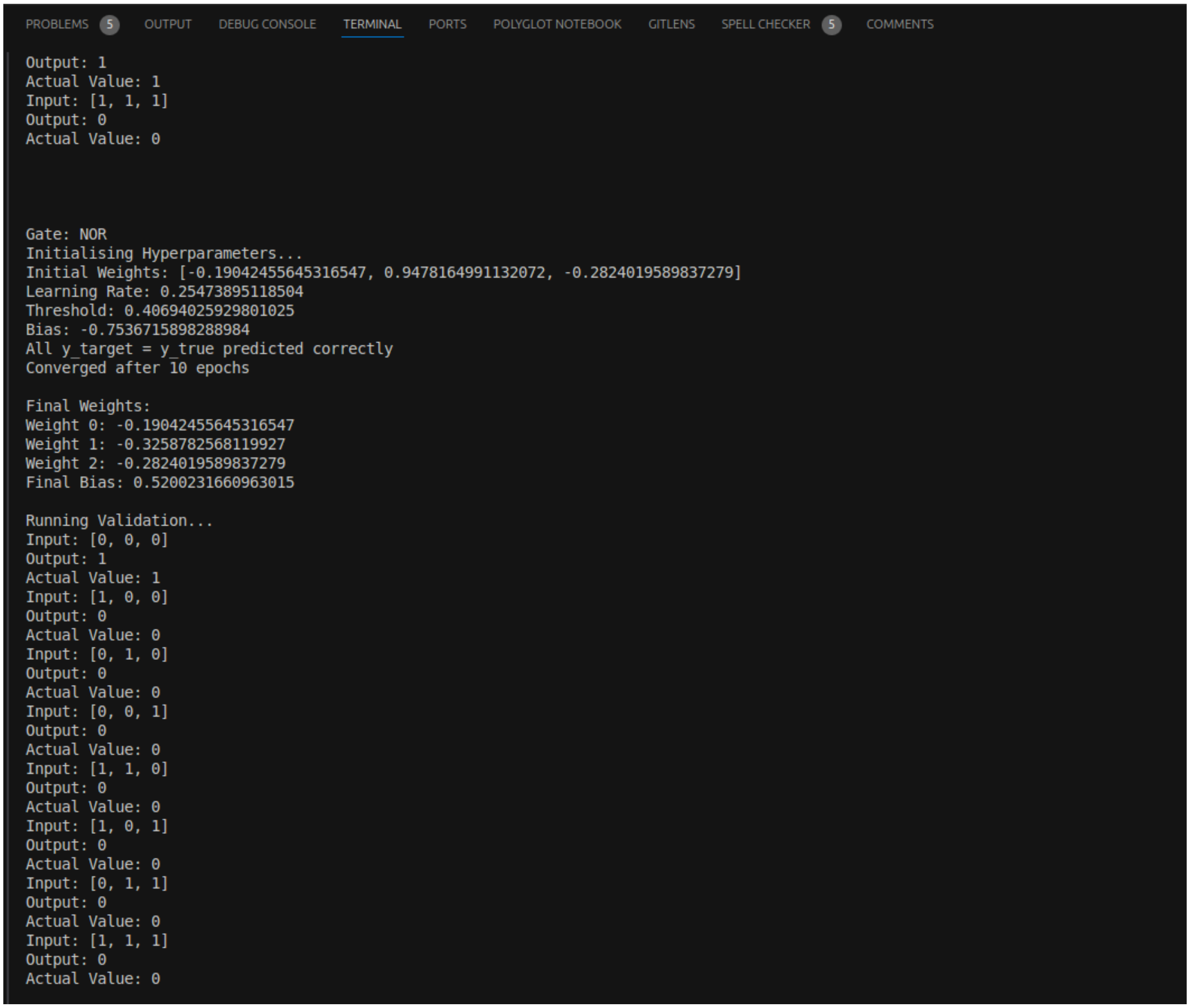
```
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Computational-Intelligence-Lab-CS354N/LAB 5$ python3 q1.py
Gate: AND
Initialising Hyperparameters...
Initial Weights: [0.4020370528296111, 0.9484220653821174, 0.8136972810746796]
Learning Rate: 0.32328281292060995
Threshold: -0.6353266388228132
Bias: -0.13517517691696246
All y_target = y_true predicted correctly
Converged after 3 epochs

Final Weights:
Weight 0: 0.4020370528296111
Weight 1: 0.6251392524615074
Weight 2: 0.4904144681540697
Final Bias: -1.7515892415200123

Running Validation...
Input: [0, 0, 0]
Output: 0
Actual Value: 0
Input: [1, 0, 0]
Output: 0
Actual Value: 0
Input: [0, 1, 0]
Output: 0
Actual Value: 0
Input: [0, 0, 1]
Output: 0
Actual Value: 0
Input: [1, 1, 0]
Output: 0
Actual Value: 0
Input: [1, 0, 1]
Output: 0
Actual Value: 0
Input: [0, 1, 1]
Output: 0
Actual Value: 0
Input: [1, 1, 1]
Output: 1
Actual Value: 1
```



```
Gate: OR
Initialising Hyperparameters...
Initial Weights: [0.6691092908238241, 0.14713774403541424, 0.6825577636364508]
Learning Rate: 0.018954019652259638
Threshold: -0.17822943795032153
Bias: 0.505446161967259
All y_target = y_true predicted correctly
Converged after 37 epochs

Final Weights:
Weight 0: 0.6691092908238241
Weight 1: 0.14713774403541424
Weight 2: 0.6825577636364508
Final Bias: -0.1958525651663473

Running Validation...
Input: [0, 0, 0]
Output: 0
Actual Value: 0
Input: [1, 0, 0]
Output: 1
Actual Value: 1
Input: [0, 1, 0]
Output: 1
Actual Value: 1
Input: [0, 0, 1]
Output: 1
Actual Value: 1
Input: [1, 1, 0]
Output: 1
Actual Value: 1
Input: [1, 0, 1]
Output: 1
Actual Value: 1
Input: [0, 1, 1]
Output: 1
Actual Value: 1
Input: [1, 1, 1]
Output: 1
Actual Value: 1
```



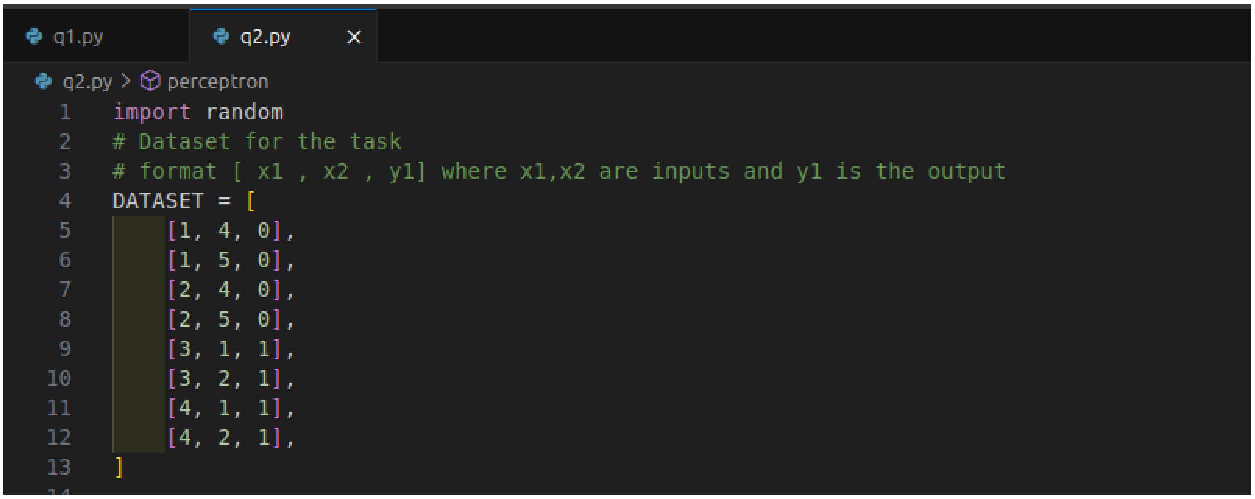
```
Gate: NAND
Initialising Hyperparameters...
Initial Weights: [0.7748892258552131, -0.09417410836742057, 0.14996056626676446]
Learning Rate: 0.4336010147079483
Threshold: 0.8779736612730189
Bias: 0.07325663920490566
All y_target = y_true predicted correctly
Converged after 10 epochs

Final Weights:
Weight 0: -0.5259138182686318
Weight 1: -0.5277751230753689
Weight 2: -1.5844434925650288
Final Bias: 3.108463742160543

Running Validation...
Input: [0, 0, 0]
Output: 1
Actual Value: 1
Input: [1, 0, 0]
Output: 1
Actual Value: 1
Input: [0, 1, 0]
Output: 1
Actual Value: 1
Input: [0, 0, 1]
Output: 1
Actual Value: 1
Input: [1, 1, 0]
Output: 1
Actual Value: 1
Input: [1, 0, 1]
Output: 1
Actual Value: 1
Input: [0, 1, 1]
Output: 1
Actual Value: 1
Input: [1, 1, 1]
Output: 0
Actual Value: 0
```



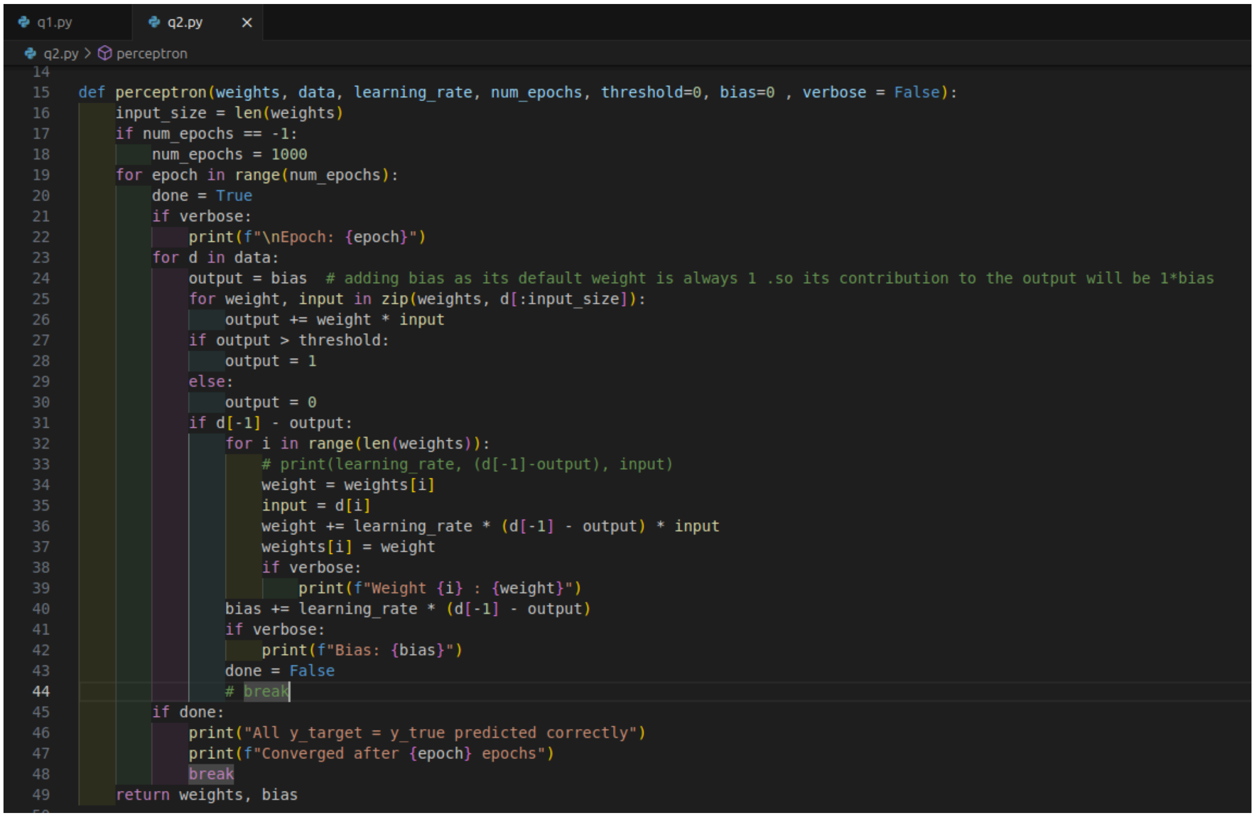
```
Output: 1
Actual Value: 1
Input: [1, 1, 1]
Output: 0
Actual Value: 0




Gate: NOR
Initialising Hyperparameters...
Initial Weights: [-0.19042455645316547, 0.9478164991132072, -0.2824019589837279]
Learning Rate: 0.25473895118504
Threshold: 0.40694025929801025
Bias: -0.7536715898288984
All y_target = y_true predicted correctly
Converged after 10 epochs

Final Weights:
Weight 0: -0.19042455645316547
Weight 1: -0.3258782568119927
Weight 2: -0.2824019589837279
Final Bias: 0.5200231660963015

Running Validation...
Input: [0, 0, 0]
Output: 1
Actual Value: 1
Input: [1, 0, 0]
Output: 0
Actual Value: 0
Input: [0, 1, 0]
Output: 0
Actual Value: 0
Input: [0, 0, 1]
Output: 0
Actual Value: 0
Input: [1, 1, 0]
Output: 0
Actual Value: 0
Input: [1, 0, 1]
Output: 0
Actual Value: 0
Input: [0, 1, 1]
Output: 0
Actual Value: 0
Input: [1, 1, 1]
Output: 0
Actual Value: 0
```

Question 2

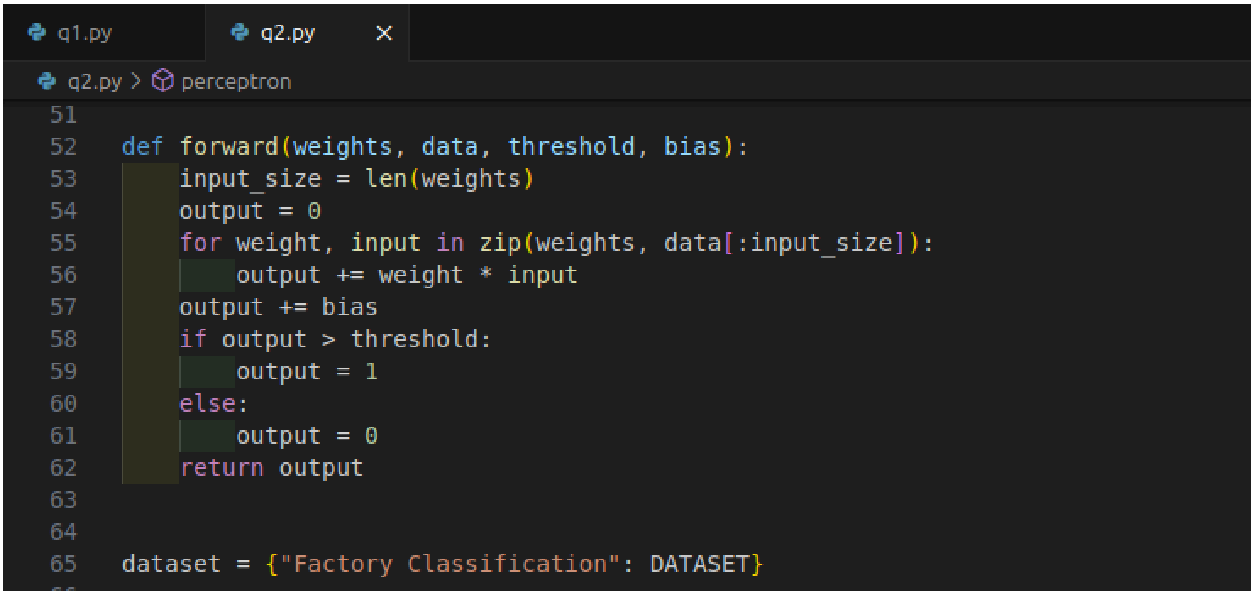
Code:



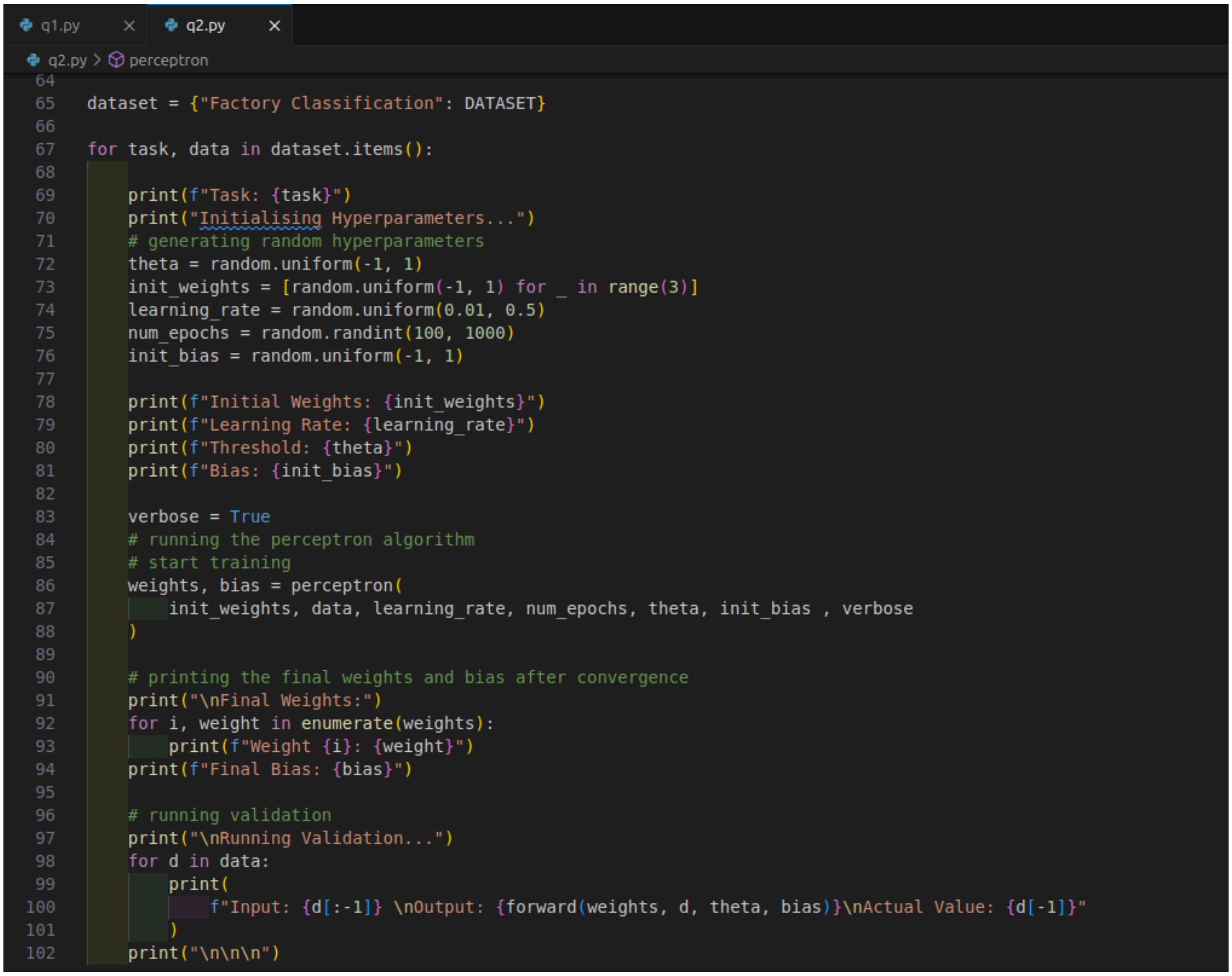
```python
import random
# Dataset for the task
# format [ x1 , x2 , y1] where x1,x2 are inputs and y1 is the output
DATASET = [
        [1, 4, 0],
        [1, 5, 0],
        [2, 4, 0],
        [2, 5, 0],
        [3, 1, 1],
        [3, 2, 1],
        [4, 1, 1],
        [4, 2, 1],
]
```



```python

def perceptron(weights, data, learning_rate, num_epochs, threshold=0, bias=0 , verbose = False):
    input_size = len(weights)
    if num_epochs == -1:
        num_epochs = 1000
    for epoch in range(num_epochs):
        done = True
        if verbose:
            print(f"\nEpoch: {epoch}")
        for d in data:
            output = bias  # adding bias as its default weight is always 1 .so its contribution to the output will be 1*bias
            for weight, input in zip(weights, d[:input_size]):
                output += weight * input
            if output > threshold:
                output = 1
            else:
                output = 0
            if d[-1] - output:
                for i in range(len(weights)):
                    # print(learning_rate, (d[-1]-output), input)
                    weight = weights[i]
                    input = d[i]
                    weight += learning_rate * (d[-1] - output) * input
                    weights[i] = weight
                    if verbose:
                        print(f"Weight {i} : {weight}")
                bias += learning_rate * (d[-1] - output)
                if verbose:
                    print(f"Bias: {bias}")
                done = False
                # break
        if done:
            print("All y_target = y_true predicted correctly")
            print(f"Converged after {epoch} epochs")
            break
    return weights, bias
```

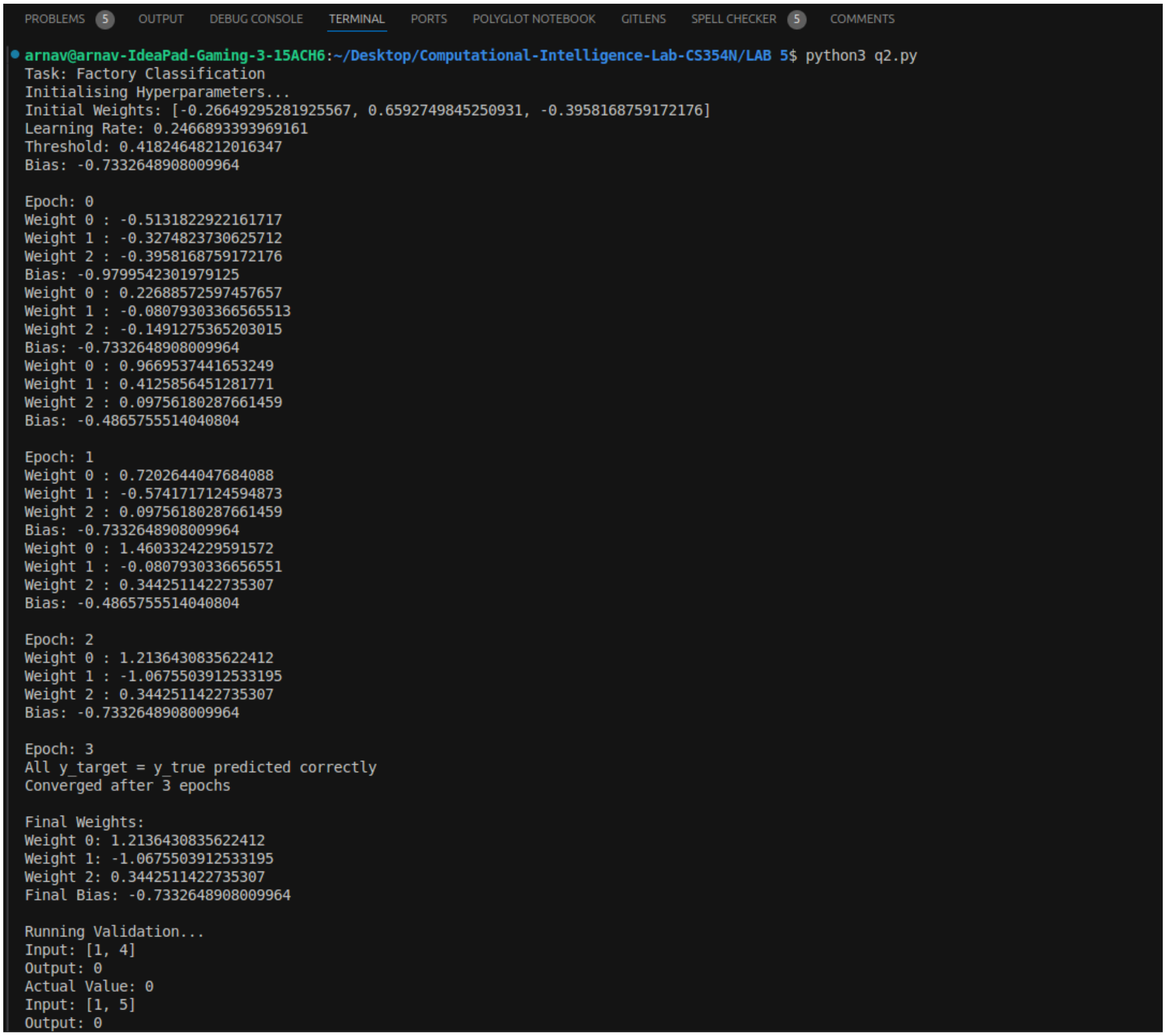


```python

def forward(weights, data, threshold, bias):
    input_size = len(weights)
    output = 0
    for weight, input in zip(weights, data[:input_size]):
        output += weight * input
    output += bias
    if output > threshold:
        output = 1
    else:
        output = 0
    return output


dataset = {"Factory Classification": DATASET}
```

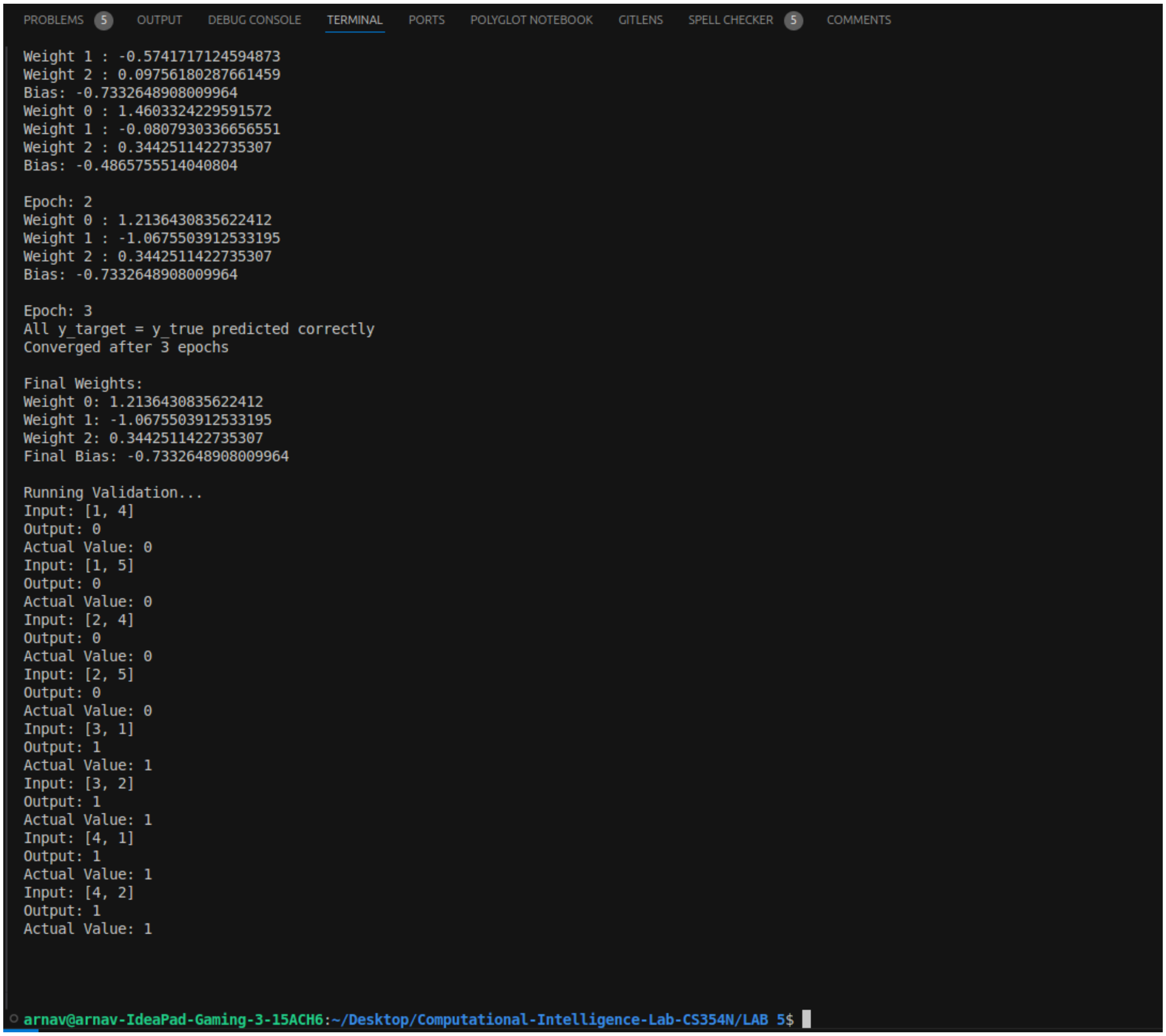


```python

dataset = {"Factory Classification": DATASET}

for task, data in dataset.items():

    print(f"Task: {task}")
    print("Initialising Hyperparameters...")
    # generating random hyperparameters
    theta = random.uniform(-1, 1)
    init_weights = [random.uniform(-1, 1) for _ in range(3)]
    learning_rate = random.uniform(0.01, 0.5)
    num_epochs = random.randint(100, 1000)
    init_bias = random.uniform(-1, 1)

    print(f"Initial Weights: {init_weights}")
    print(f"Learning Rate: {learning_rate}")
    print(f"Threshold: {theta}")
    print(f"Bias: {init_bias}")

    verbose = True
    # running the perceptron algorithm
    # start training
    weights, bias = perceptron(
        init_weights, data, learning_rate, num_epochs, theta, init_bias , verbose
    )

    # printing the final weights and bias after convergence
    print("\nFinal Weights:")
    for i, weight in enumerate(weights):
        print(f"Weight {i}: {weight}")
    print(f"Final Bias: {bias}")

    # running validation
    print("\nRunning Validation...")
    for d in data:
        print(
            f"Input: {d[:-1]} \nOutput: {forward(weights, d, theta, bias)}\nActual Value: {d[-1]}"
        )
    print("\n\n\n")
```

Result:



```
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Computational-Intelligence-Lab-CS354N/LAB 5$ python3 q2.py
Task: Factory Classification
Initialising Hyperparameters...
Initial Weights: [-0.26649295281925567, 0.6592749845250931, -0.3958168759172176]
Learning Rate: 0.2466893393969161
Threshold: 0.41824648212016347
Bias: -0.7332648908009964

Epoch: 0
Weight 0 : -0.5131822922161717
Weight 1 : -0.3274823730625712
Weight 2 : -0.3958168759172176
Bias: -0.9799542301979125
Weight 0 : 0.22688572597457657
Weight 1 : -0.08079303366565513
Weight 2 : -0.1491275365203015
Bias: -0.7332648908009964
Weight 0 : 0.9669537441653249
Weight 1 : 0.4125856451281771
Weight 2 : 0.09756180287661459
Bias: -0.4865755514040804

Epoch: 1
Weight 0 : 0.7202644047684088
Weight 1 : -0.5741717124594873
Weight 2 : 0.09756180287661459
Bias: -0.7332648908009964
Weight 0 : 1.4603324229591572
Weight 1 : -0.0807930336656551
Weight 2 : 0.3442511422735307
Bias: -0.4865755514040804

Epoch: 2
Weight 0 : 1.2136430835622412
Weight 1 : -1.0675503912533195
Weight 2 : 0.3442511422735307
Bias: -0.7332648908009964

Epoch: 3
All y_target = y_true predicted correctly
Converged after 3 epochs

Final Weights:
Weight 0: 1.2136430835622412
Weight 1: -1.0675503912533195
Weight 2: 0.3442511422735307
Final Bias: -0.7332648908009964

Running Validation...
Input: [1, 4]
Output: 0
Actual Value: 0
Input: [1, 5]
Output: 0
```



```
Weight 1 : -0.5741717124594873
Weight 2 : 0.09756180287661459
Bias: -0.7332648908009964
Weight 0 : 1.4603324229591572
Weight 1 : -0.0807930336656551
Weight 2 : 0.3442511422735307
Bias: -0.4865755514040804

Epoch: 2
Weight 0 : 1.2136430835622412
Weight 1 : -1.0675503912533195
Weight 2 : 0.3442511422735307
Bias: -0.7332648908009964

Epoch: 3
All y_target = y_true predicted correctly
Converged after 3 epochs

Final Weights:
Weight 0: 1.2136430835622412
Weight 1: -1.0675503912533195
Weight 2: 0.3442511422735307
Final Bias: -0.7332648908009964

Running Validation...
Input: [1, 4]
Output: 0
Actual Value: 0
Input: [1, 5]
Output: 0
Actual Value: 0
Input: [2, 4]
Output: 0
Actual Value: 0
Input: [2, 5]
Output: 0
Actual Value: 0
Input: [3, 1]
Output: 1
Actual Value: 1
Input: [3, 2]
Output: 1
Actual Value: 1
Input: [4, 1]
Output: 1
Actual Value: 1
Input: [4, 2]
Output: 1
Actual Value: 1



arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Computational-Intelligence-Lab-CS354N/LAB 5$
```

For code refer here:

https://github.com/arnavjain2710/Computational-Intelligence-Lab-CS354N/tree/main/LAB%205