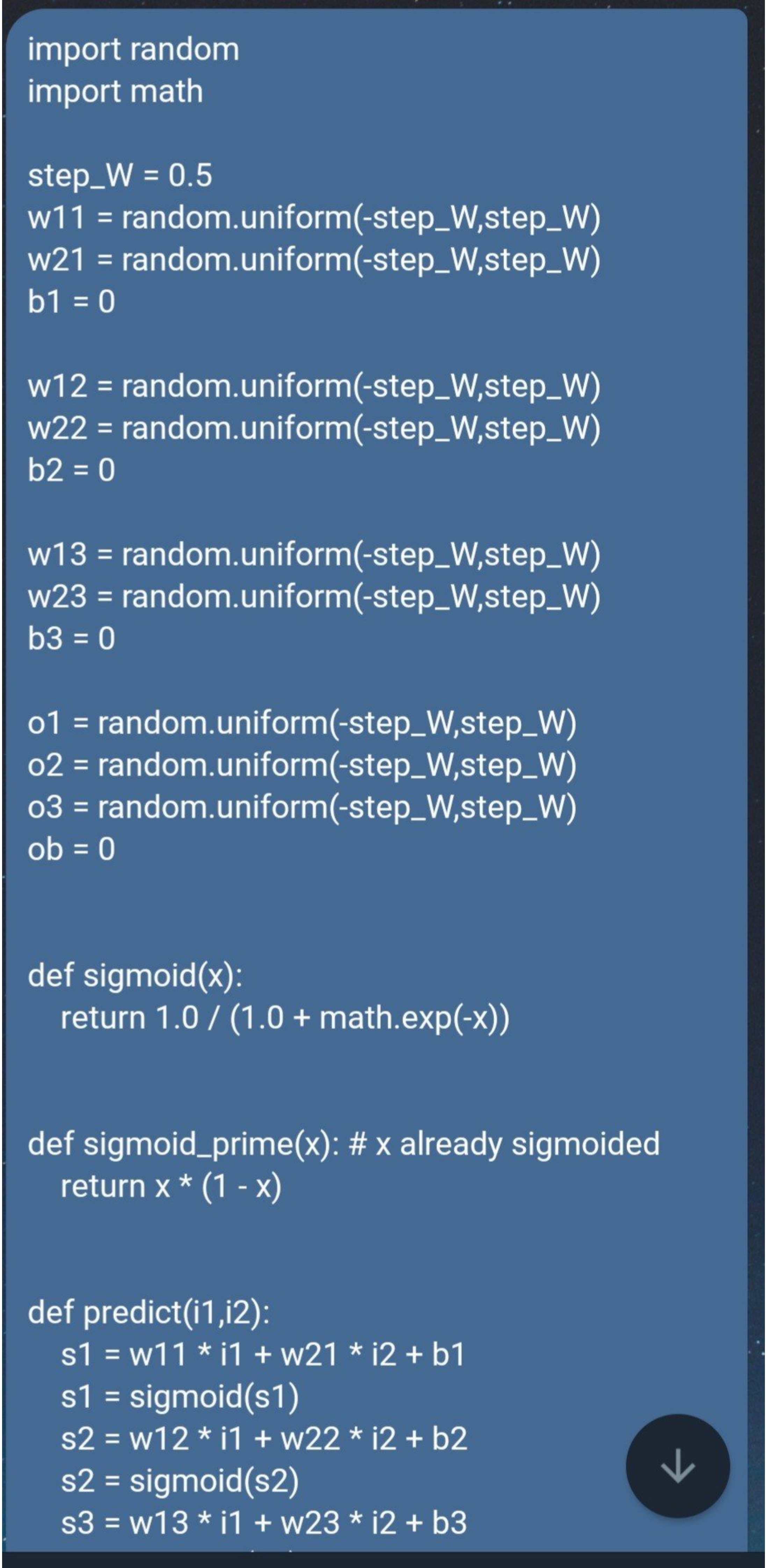
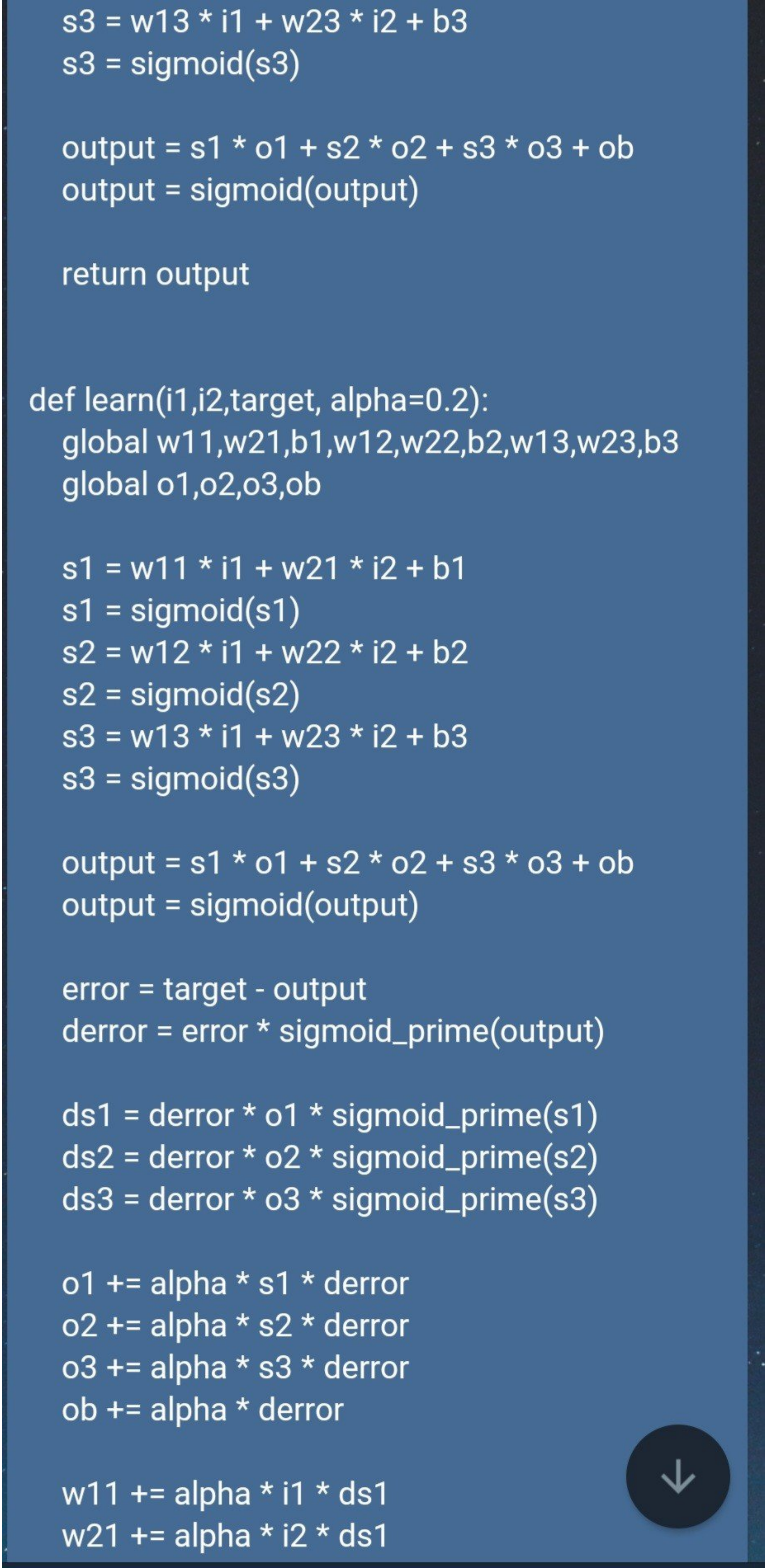
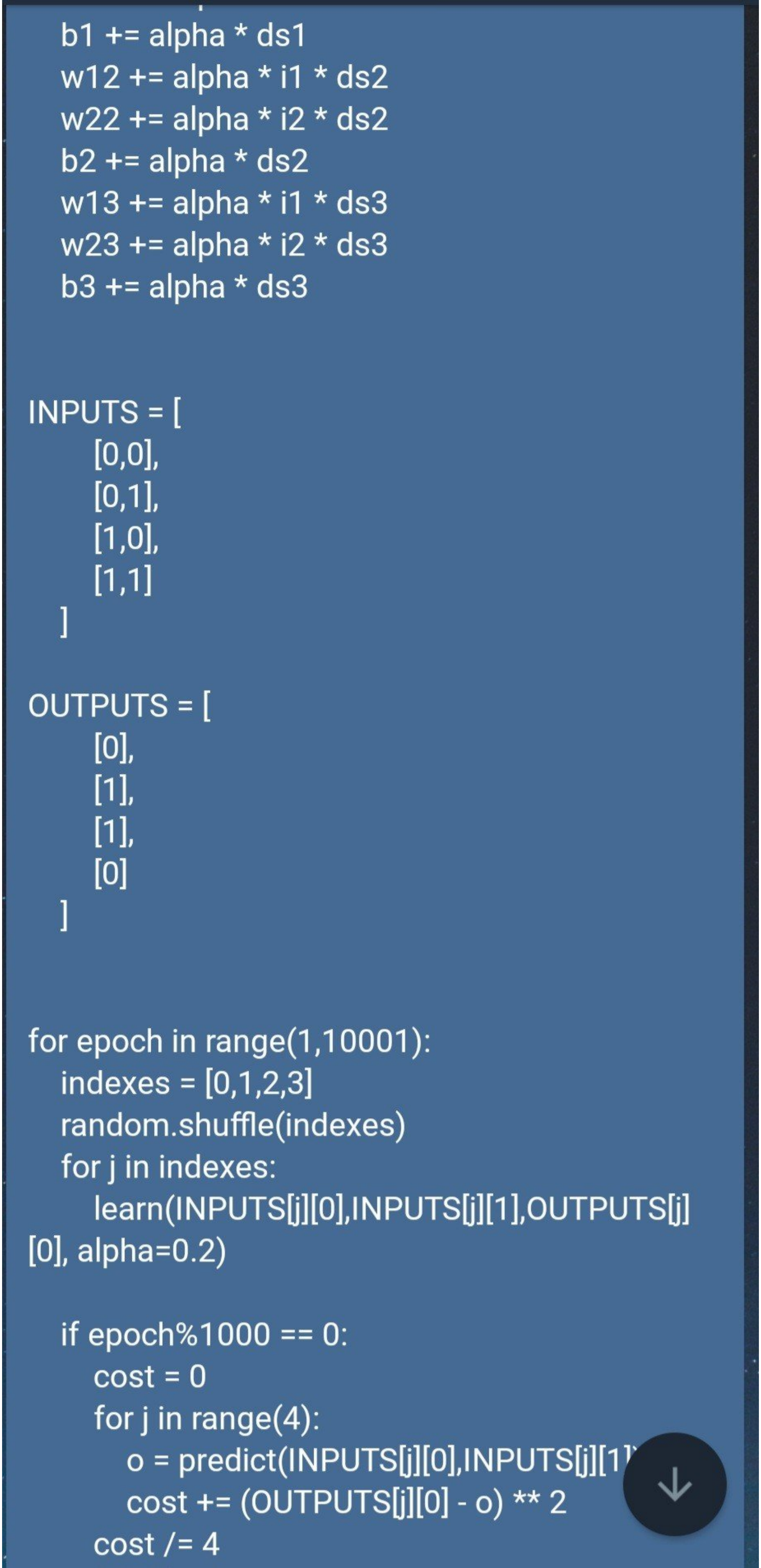


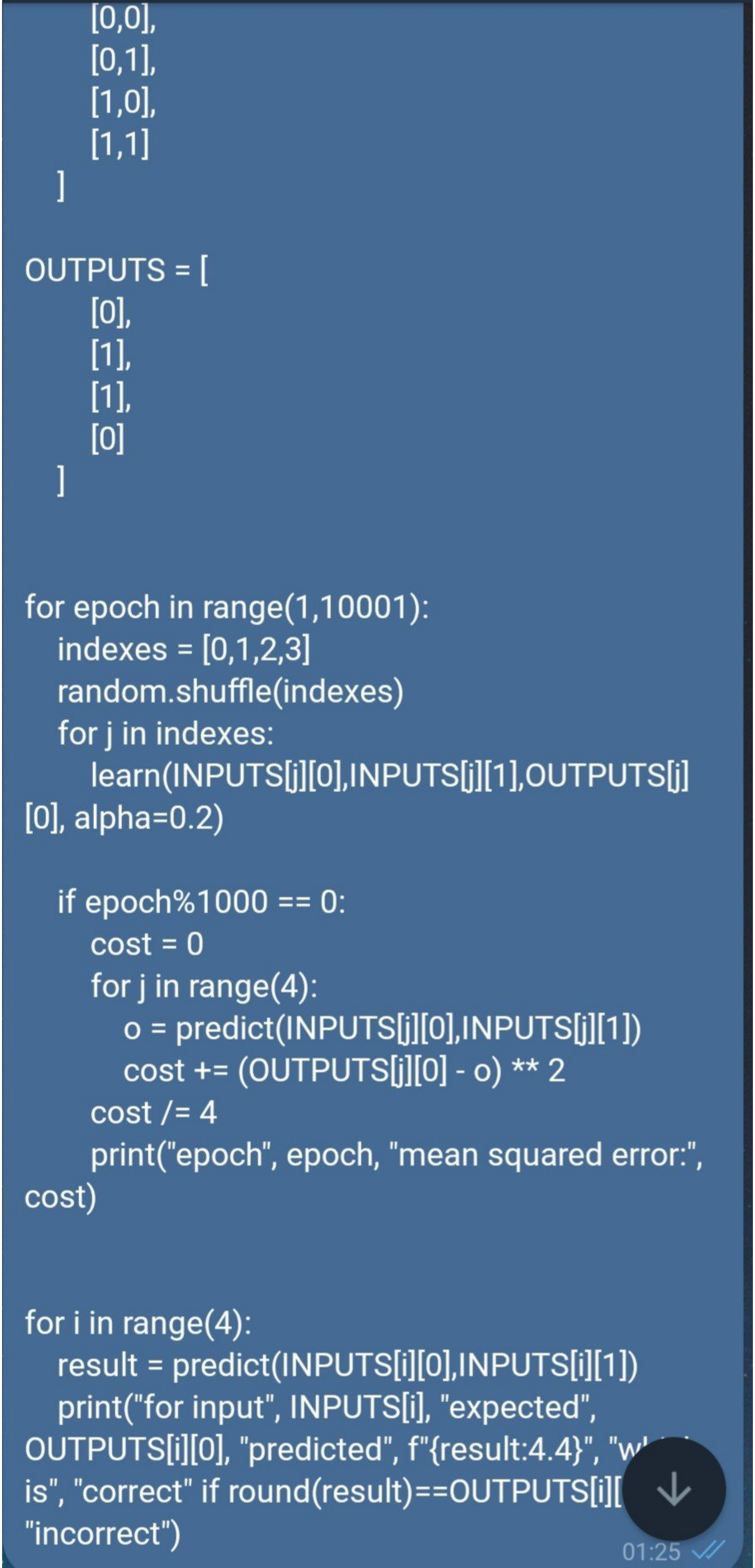
```
import random
import math

step_W = 0.5
w11 = random.uniform(-step_W,step_W)
w21 = random.uniform(-step_W,step_W)
b1 = 0

w12 = random.uniform(-step_W,step_W)
w22 = random.uniform(-step_W,step_W)
b2 = 0

w13 = random.uniform(-step_W,step_W)
w23 = random.uniform(-step_W,step_W)
b3 = 0

o1 = random.uniform(-step_W,step_W)
o2 = random.uniform(-step_W,step_W)
o3 = random.uniform(-step_W,step_W)
ob = 0


def sigmoid(x):
    return 1.0 / (1.0 + math.exp(-x))


def sigmoid_prime(x): # x already sigmoided
    return x * (1 - x)


def predict(i1,i2):
    s1 = w11 * i1 + w21 * i2 + b1
    s1 = sigmoid(s1)
    s2 = w12 * i1 + w22 * i2 + b2
    s2 = sigmoid(s2)
    s3 = w13 * i1 + w23 * i2 + b3
```



```
    s3 = w13 * i1 + w23 * i2 + b3
    s3 = sigmoid(s3)

    output = s1 * o1 + s2 * o2 + s3 * o3 + ob
    output = sigmoid(output)

    return output


def learn(i1,i2,target, alpha=0.2):
    global w11,w21,b1,w12,w22,b2,w13,w23,b3
    global o1,o2,o3,ob

    s1 = w11 * i1 + w21 * i2 + b1
    s1 = sigmoid(s1)
    s2 = w12 * i1 + w22 * i2 + b2
    s2 = sigmoid(s2)
    s3 = w13 * i1 + w23 * i2 + b3
    s3 = sigmoid(s3)

    output = s1 * o1 + s2 * o2 + s3 * o3 + ob
    output = sigmoid(output)

    error = target - output
    derror = error * sigmoid_prime(output)

    ds1 = derror * o1 * sigmoid_prime(s1)
    ds2 = derror * o2 * sigmoid_prime(s2)
    ds3 = derror * o3 * sigmoid_prime(s3)

    o1 += alpha * s1 * derror
    o2 += alpha * s2 * derror
    o3 += alpha * s3 * derror
    ob += alpha * derror

    w11 += alpha * i1 * ds1
    w21 += alpha * i2 * ds1
```



```
    b1 += alpha * ds1
    w12 += alpha * i1 * ds2
    w22 += alpha * i2 * ds2
    b2 += alpha * ds2
    w13 += alpha * i1 * ds3
    w23 += alpha * i2 * ds3
    b3 += alpha * ds3


INPUTS = [
        [0,0],
        [0,1],
        [1,0],
        [1,1]
    ]

OUTPUTS = [
        [0],
        [1],
        [1],
        [0]
    ]


for epoch in range(1,10001):
    indexes = [0,1,2,3]
    random.shuffle(indexes)
    for j in indexes:
        learn(INPUTS[j][0],INPUTS[j][1],OUTPUTS[j][0], alpha=0.2)

    if epoch%1000 == 0:
        cost = 0
        for j in range(4):
            o = predict(INPUTS[j][0],INPUTS[j][1
            cost += (OUTPUTS[j][0] - o) ** 2
        cost /= 4
```



```
        [0,0],
        [0,1],
        [1,0],
        [1,1]
    ]

OUTPUTS = [
        [0],
        [1],
        [1],
        [0]
    ]


for epoch in range(1,10001):
    indexes = [0,1,2,3]
    random.shuffle(indexes)
    for j in indexes:
        learn(INPUTS[j][0],INPUTS[j][1],OUTPUTS[j]
[0], alpha=0.2)

    if epoch%1000 == 0:
        cost = 0
        for j in range(4):
            o = predict(INPUTS[j][0],INPUTS[j][1])
            cost += (OUTPUTS[j][0] - o) ** 2
        cost /= 4
        print("epoch", epoch, "mean squared error:",
cost)


for i in range(4):
    result = predict(INPUTS[i][0],INPUTS[i][1])
    print("for input", INPUTS[i], "expected",
OUTPUTS[i][0], "predicted", f"{result:4.4}", "w
is", "correct" if round(result)==OUTPUTS[i][
"incorrect")
```

01:25