```
#Importing the numpy library
import numpy as np

#Function sigmoide
def sigmoid(x):
    return 1/(1+np.exp(-x))

#Derivative of the sigmoid function
def sigmoid_prime(x):
    return sigmoid(x) * (1 - sigmoid(x))

#Defaul values
learnrate = 0.5
x = np.array([1, 2, 3, 4])
y = np.array(0.5)
b = 0.5

#  Initial weights
## Calculate one gradient descent step for each weight
w = np.array([0.5, -0.5, 0.3, 0.1])

# TODO: Calculate the node's linear combination of inputs and weights
h = np.dot(x, w)

# TODO: Calculate output of neural network
nn_output = sigmoid(h)

# TODO: Calculate error of neural network
error = y - nn_output

# TODO: Calculate the error term
error_term = error * sigmoid_prime(h)

# TODO: Calculate change in weights
del_w = learnrate * error_term * x

def second(w):
    #Run many times as needed
    h = np.dot(x, w) + b
    nn_output = sigmoid(h)
    error = y - nn_output
    error_term = error * sigmoid_prime(h)
    del_w = learnrate * error_term * x

    #Return all values
    return h, nn_output, error, error_term, del_w

#Run many times as needed
for i in range(2):
    print('w', i+1, '\n')

    print('Output da rede: ')
    print(nn_output)

    print('\nO erro do output:')
    print(error)

    print('\nO termo do erro:')
    print(error_term)

    print('\nAtualizar o passo:')
    print(del_w)

    w = w + del_w #Updates weights
    h, nn_output, error, error_term, del_w = second(w)

    print('-----------------------------------------\n')
```