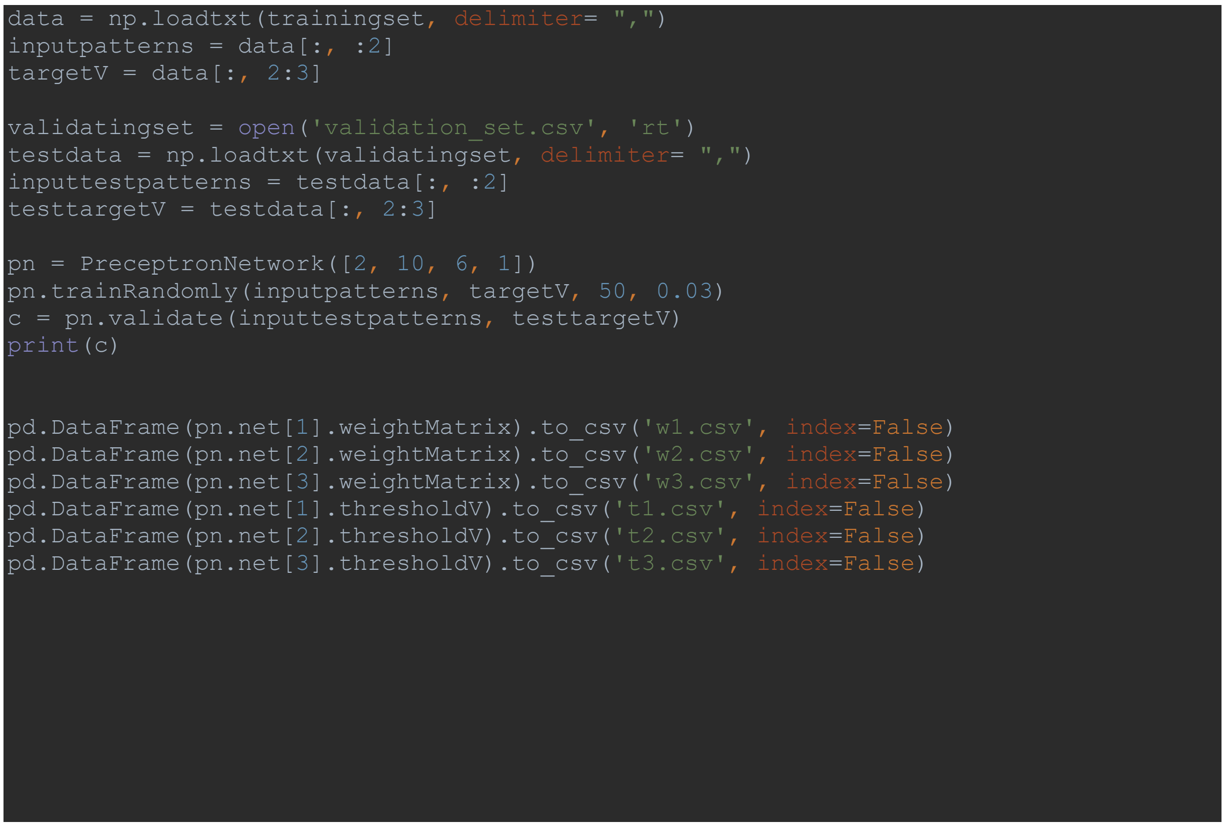
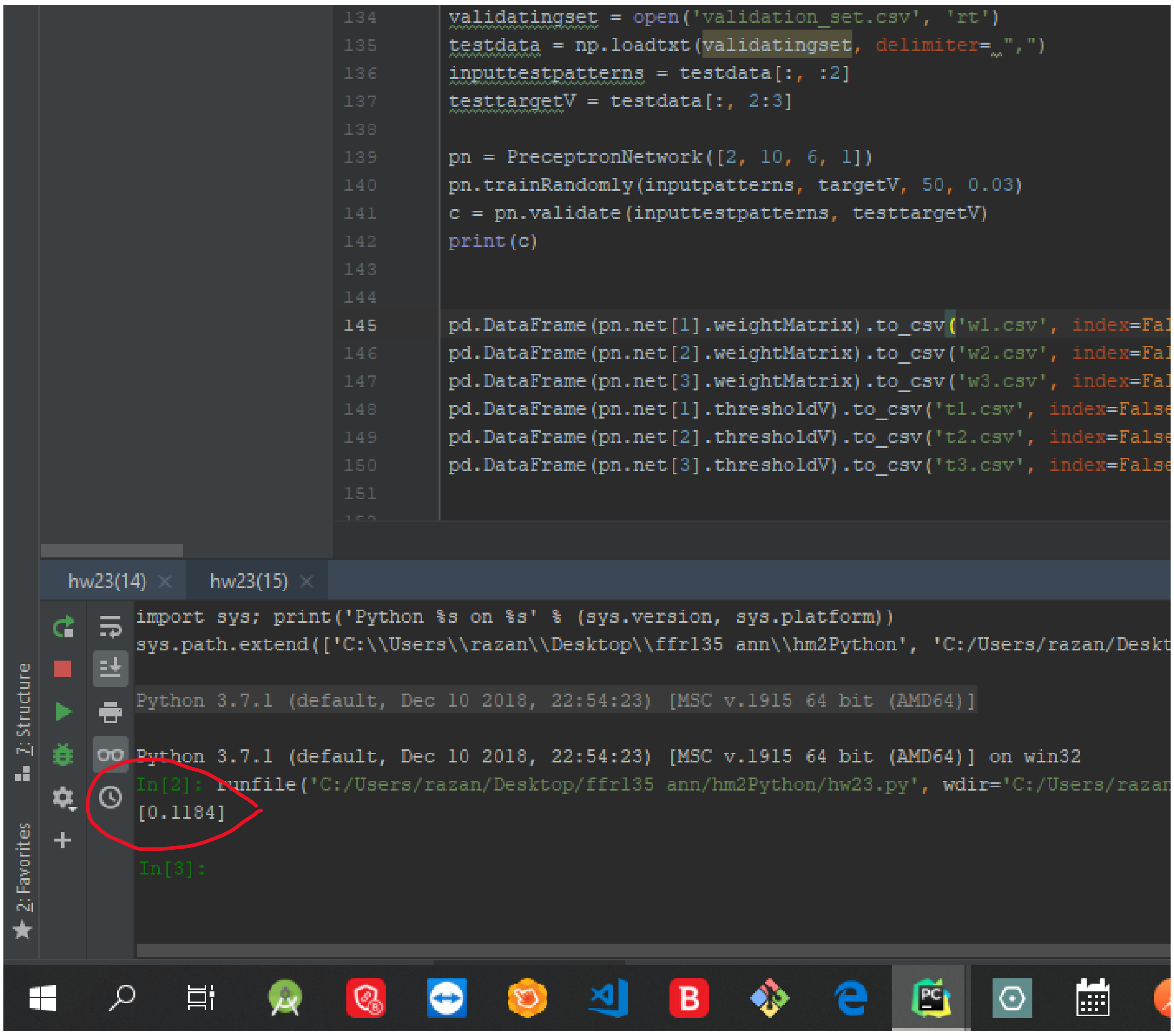
```
#Yazan Ghafir HW2-4 The code is in Python 2019-10-04 The result: 0.1184



import numpy as np
from random import randint
import pandas as pd


class Layer:
    # N is the number of the neurons in the layer
    # Nbefore is the number of the neurons in the layer before
    # we create Weight matrix, states vector,
    # thresholds vector, errors vector and local fields vector
    # and in each layer.
    def __init__(self, N, Nbefore):
        self.N = N
        self.Nbefore = Nbefore
        self.thresholdV = np.zeros(N)
        self.localFieldsV = np.zeros(N)
        self.statesV = np.zeros(N)
        self.errorsV = np.zeros(N)
        self.weightMatrix = np.random.normal(0, 0.8, (N, Nbefore))


class PreceptronNetwork:
    # numNLayer array of number of neurons in each layer
    def __init__(self, numNLayer):
        self.numNLayer = numNLayer
        self.net = [Layer(numNLayer[0], 0), Layer(numNLayer[1], numNLayer[0]),
Layer(numNLayer[2], numNLayer[1]),
                    Layer(numNLayer[len(numNLayer) - 1], numNLayer[len(numNLayer) -
2])]

    # the local fields is calculated by just matrix multiplication W * X for each
layer beginning with the first as an input
    def calcLocalFieldsAndStates(self):
        for i in range(1, len(self.numNLayer)):
            a = self.net[i].weightMatrix
            b = self.net[i-1].statesV
            tmp = np.dot(a, b)
            tmp = tmp - self.net[i].thresholdV
            np.copyto(self.net[i].localFieldsV, tmp)
            tmp = np.tanh(tmp)
            np.copyto(self.net[i].statesV, tmp)

    def calcErrors (self, target):
        tmp = target - self.net[len(self.numNLayer)-1].statesV
        tmp = self.gP(self.net[len(self.numNLayer)-1].localFieldsV) * tmp
        np.copyto(self.net[len(self.numNLayer)-1].errorsV, tmp)

        for i in range(len(self.numNLayer)-2, 0, -1):
            for j in range(0, len(self.net[i].errorsV)):
                a2 = self.net[i+1].weightMatrix[:, [j]]
                b2 = np.transpose(self.net[i+1].errorsV)
                tmp2 = np.dot(b2, a2)
                st = self.gP(self.net[i].localFieldsV[j])
                self.net[i].errorsV[j] = st * tmp2


    def updateWeights (self, learningRate):
        for i in range(len(self.numNLayer)-1, 0, -1):
            errorL = self.net[i].errorsV
            xtransLm1 = np.transpose(self.net[i-1].statesV)
            tmp = np.outer(errorL, xtransLm1)
            tmp2 = learningRate * tmp
            tmp3 = learningRate * errorL
            self.net[i].weightMatrix += tmp2
            self.net[i].thresholdV -= tmp3


    # feed a pattern
    def feed(self, p):
        np.copyto(self.net[0].statesV, p)

    #train the network
    def train(self,inputV, targetV, trainingRatio, learningRate):
        for j in range(0, trainingRatio):
            for i in range(0, len(targetV)):
                self.feedforward(inputV[i])
                self.backpropagate(targetV[i], learningRate)

    # train the network
    def trainRandomly(self, inputV, targetV, trainingRatio, learningRate):
        for j in range(0, trainingRatio):
            copyInputV = np.zeros((len(inputV), 2))
            copytargetV = np.zeros((len(targetV), 1))
            np.copyto(copyInputV, inputV)
            np.copyto(copytargetV, targetV)
            copyInputV = copyInputV.tolist()
            copytargetV = copytargetV.tolist()
            for i in range(0, len(targetV)):
                if len(copyInputV) != 0:
                    index = randint(0, len(copyInputV)-1)
                    self.feedforward(copyInputV[index])
                    self.backpropagate(copytargetV[index], learningRate)
                    del copyInputV[index]
                    del copytargetV[index]


    #helping method feed forward
    def feedforward (self, inputV):
        self.feed(inputV)
        self.calcLocalFieldsAndStates()

    #helping method backpropagate
    def backpropagate (self, target, learningRate):
        self.calcErrors(target)
        self.updateWeights(learningRate)

    # the derivative of the tangent hyperbolic function
    def gP(self, b):
        return (1- (np.tanh(b)*np.tanh(b)))

    #test with the validation set
    def validate(self,inputtestpatterns, testtargetV):
        sum = 0
        for i in range(0, len(testtargetV)):
            self.feedforward(inputtestpatterns[i])
            sum += self.validateTarget(testtargetV[i])
        c = (1.0/(2.0*len(testtargetV))) * sum
        return c

    def validateTarget(self, testtarget):
        sgno =  self.sgn(self.net[len(self.numNLayer)-1].statesV)
        sub = sgno - testtarget
        return abs(sub)

    def sgn(self, output):
        if (output < 0.0):
            return -1
        else:
            return 1


trainingset = open('training_set.csv', 'rt')
```



```python
data = np.loadtxt(trainingset, delimiter= ",")
inputpatterns = data[:, :2]
targetV = data[:, 2:3]

validatingset = open('validation_set.csv', 'rt')
testdata = np.loadtxt(validatingset, delimiter= ",")
inputtestpatterns = testdata[:, :2]
testtargetV = testdata[:, 2:3]

pn = PreceptronNetwork([2, 10, 6, 1])
pn.trainRandomly(inputpatterns, targetV, 50, 0.03)
c = pn.validate(inputtestpatterns, testtargetV)
print(c)


pd.DataFrame(pn.net[1].weightMatrix).to_csv('w1.csv', index=False)
pd.DataFrame(pn.net[2].weightMatrix).to_csv('w2.csv', index=False)
pd.DataFrame(pn.net[3].weightMatrix).to_csv('w3.csv', index=False)
pd.DataFrame(pn.net[1].thresholdV).to_csv('t1.csv', index=False)
pd.DataFrame(pn.net[2].thresholdV).to_csv('t2.csv', index=False)
pd.DataFrame(pn.net[3].thresholdV).to_csv('t3.csv', index=False)
```



```python
validatingset = open('validation_set.csv', 'rt')
testdata = np.loadtxt(validatingset, delimiter=",")
inputtestpatterns = testdata[:, :2]
testtargetV = testdata[:, 2:3]

pn = PreceptronNetwork([2, 10, 6, 1])
pn.trainRandomly(inputpatterns, targetV, 50, 0.03)
c = pn.validate(inputtestpatterns, testtargetV)
print(c)


pd.DataFrame(pn.net[1].weightMatrix).to_csv('w1.csv', index=Fal
pd.DataFrame(pn.net[2].weightMatrix).to_csv('w2.csv', index=Fal
pd.DataFrame(pn.net[3].weightMatrix).to_csv('w3.csv', index=Fal
pd.DataFrame(pn.net[1].thresholdV).to_csv('t1.csv', index=False
pd.DataFrame(pn.net[2].thresholdV).to_csv('t2.csv', index=False
pd.DataFrame(pn.net[3].thresholdV).to_csv('t3.csv', index=False
```

```
import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['C:\\Users\\razan\\Desktop\\ffr135 ann\\hm2Python', 'C:/Users/razan/Deskt

Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)]

Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] on win32
In[2]: runfile('C:/Users/razan/Desktop/ffr135 ann/hm2Python/hw23.py', wdir='C:/Users/razan
[0.1184]

In[3]:
```