```
#for questions 8-9

import math
import numpy as np
import random

def createData(f, n):
    matx = np.zeros(shape=(n,3))
    vecty = np.zeros(shape=(n,1))
    for i in range(0, n):
        x = np.matrix([1, random.uniform(-1, 1), random.uniform(-1, 1)])
        matx[i] = x
        vecty[i] = 1 if (x[0,2] > f(x[0,1])) else -1

    return matx, vecty


def setup(n):
    #Set up the target function f
    x = random.uniform(-1, 1), random.uniform(-1, 1)
    y = random.uniform(-1, 1), random.uniform(-1, 1)
    z = np.polyfit(x, y, 1)
    f = np.poly1d(z)

    #create data
    matx, vecty = createData(f, n)

    return f, matx, vecty


def logReg():
    f, matx, vecty = setup(100)
    vectw = np.zeros(shape=(1, 3))

    epochCount = 0
    #each run of the while loop is an epoch
    while True:
        epochCount += 1

        #save w^(t-1) for later comparison
        oldVectw = vectw.copy()

        #pick a random permutation of indices
        indices = np.random.permutation(list(range(100)))

        #go through each data point
        for i in indices:
            vectx = matx[i, :]
            y = vecty[i, :]

            #calculate gradient on the point and adjust vectw
            grad = -(y * vectx) / (1 + math.exp(y * np.dot(vectw, vectx)))
            vectw -= 0.01 * grad

        #Check for termination condition
        if np.linalg.norm(oldVectw - vectw) < 0.01:
            break

    #Calculate E_out
    testMatx, testVecty = createData(f, 1000)
    totalError = 0
    for i in range(1000):
        testVectx = testMatx[i, :]
        testy = testVecty[i, :]
        totalError += math.log(1 + math.exp(-testy * np.dot(vectw, testVectx)))
    avgError = totalError / 1000
```

```
    return avgError, epochCount


totalEOut = 0
totalEpochCount = 0
for i in range(0, 100):
    avgError, epochCount = logReg()
    totalEOut += avgError
    totalEpochCount += epochCount
print(totalEOut / 100)
print(totalEpochCount / 100)
```