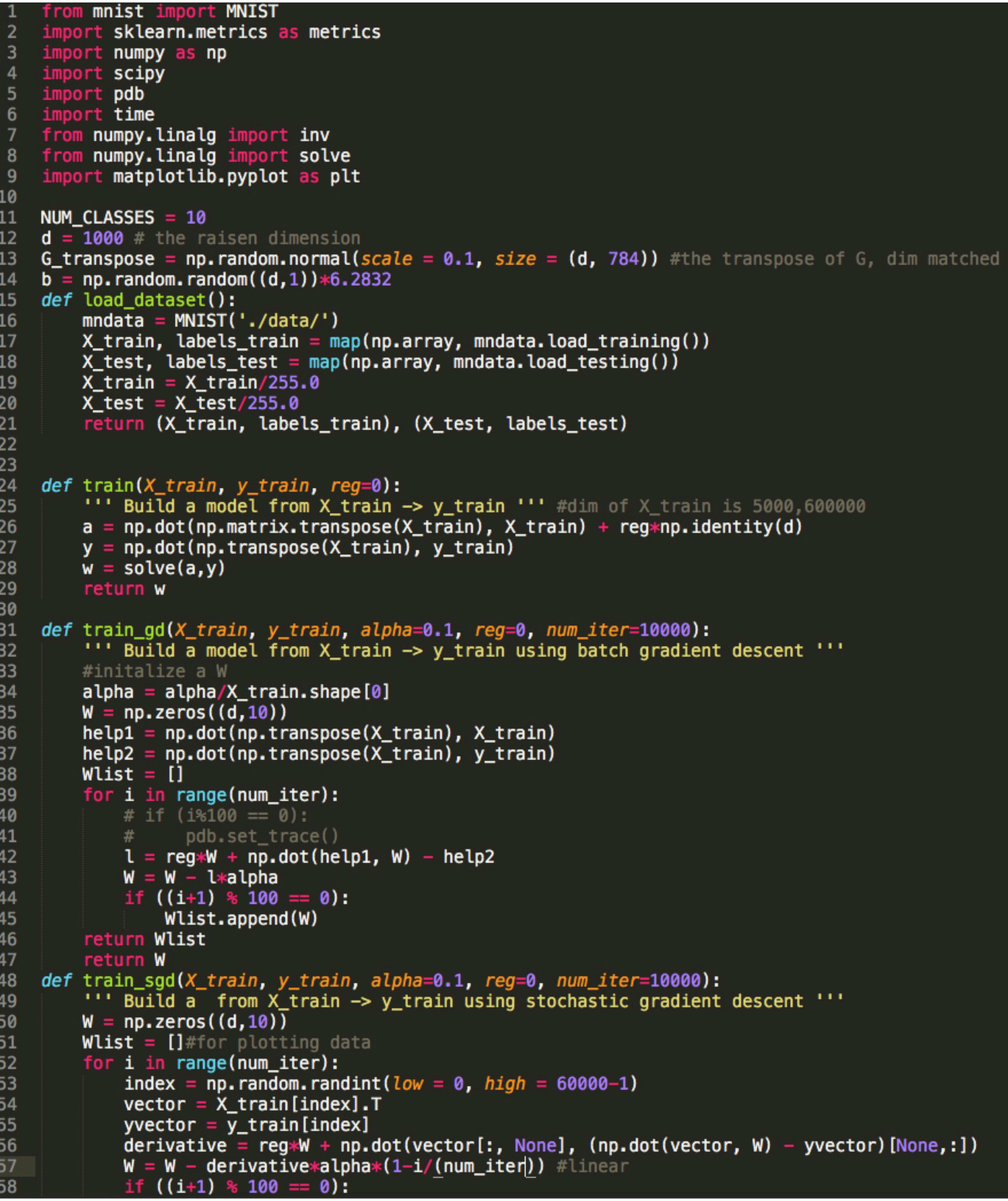

```
from mnist import MNIST
import sklearn.metrics as metrics
import numpy as np
import scipy
import pdb
import time
from numpy.linalg import inv
from numpy.linalg import solve
import matplotlib.pyplot as plt

NUM_CLASSES = 10
d = 1000 # the raisen dimension
G_transpose = np.random.normal(scale = 0.1, size = (d, 784)) #the transpose of G, dim matched
b = np.random.random((d,1))*6.2832
def load_dataset():
    mndata = MNIST('./data/')
    X_train, labels_train = map(np.array, mndata.load_training())
    X_test, labels_test = map(np.array, mndata.load_testing())
    X_train = X_train/255.0
    X_test = X_test/255.0
    return (X_train, labels_train), (X_test, labels_test)


def train(X_train, y_train, reg=0):
    ''' Build a model from X_train -> y_train ''' #dim of X_train is 5000,600000
    a = np.dot(np.matrix.transpose(X_train), X_train) + reg*np.identity(d)
    y = np.dot(np.transpose(X_train), y_train)
    w = solve(a,y)
    return w

def train_gd(X_train, y_train, alpha=0.1, reg=0, num_iter=10000):
    ''' Build a model from X_train -> y_train using batch gradient descent '''
    #initalize a W
    alpha = alpha/X_train.shape[0]
    W = np.zeros((d,10))
    help1 = np.dot(np.transpose(X_train), X_train)
    help2 = np.dot(np.transpose(X_train), y_train)
    Wlist = []
    for i in range(num_iter):
        # if (i%100 == 0):
        #     pdb.set_trace()
        l = reg*W + np.dot(help1, W) - help2
        W = W - l*alpha
        if ((i+1) % 100 == 0):
            Wlist.append(W)
    return Wlist
    return W
def train_sgd(X_train, y_train, alpha=0.1, reg=0, num_iter=10000):
    ''' Build a  from X_train -> y_train using stochastic gradient descent '''
    W = np.zeros((d,10))
    Wlist = []#for plotting data
    for i in range(num_iter):
        index = np.random.randint(low = 0, high = 60000-1)
        vector = X_train[index].T
        yvector = y_train[index]
        derivative = reg*W + np.dot(vector[:, None], (np.dot(vector, W) - yvector)[None,:])
        W = W - derivative*alpha*(1-i/(num_iter)) #linear
        if ((i+1) % 100 == 0):
```