Python: NotMNIST Data Importer and Startup Code

There are 2 databases for notMNIST: a full 529114 database and a sub 18724 sample database.   This is a function to read the databases and store then in a compressible byte format.

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mp

def notMNIST_read (number_of_files,file):

    c = 10                          #classes A-J (10)
    m = number_of_files             #samples number of files in the small DB

    trainX = np.zeros((m,28*28))      #images are 28*28
    trainY = np.zeros((m,c))

    ix = 0
    for (root, dirs, files) in os.walk(file):
        for f in files:
            if f.endswith('.png'):
                try:
                    img = mp.imread(os.path.join(root,f))
                    trainX[ix,:]=np.asarray(img).reshape(28*28)
                    folder = os.path.split(root)[-1]
                    letter = ord(folder)-ord('A')      #A=0, B=1,..., J=9
                    trainY[ix,letter]=1
                    print("processing letter %s, file number %d"%(folder, ix),'\r', end="", f
lush=True)
                    ix+=1
                except:
                    pass
    if m==18724:
        np.save('notMNIST_small_trainX', (trainX[:ix,:]*255).astype('uint8'))
        np.save('notMNIST_small_trainY', (trainY[:ix,:]).astype('uint8'))
    elif m==529114:
        np.save('notMNIST_Large_trainX', (trainX[:ix,:]*255).astype('uint8'))
        np.save('notMNIST_Large_trainY', (trainY[:ix,:]).astype('uint8'))

    return trainX, trainY
```

Example, call the function and read the small database files stored in c:\location\small

```
trainX,trainY = notMNIST_read(18724,"C:/location/small")
```

To not have to call the function again, and just read the saved databases:

```
trainX = np.load('notMNIST_small_trainX.npy')/255
trainY = np.load('notMNIST_small_trainY.npy')/1
```

Function to display an image from the dataset :

```
def display_data(ix,X=trainX,Y=trainY):
    plt.imshow(X[ix,:].reshape(28,28),cmap='gray')
    array=['A','B','C','D','E','F','G','H','I','J']
    plt.title(array[np.argmax(Y[ix,:])])
```

Function to normalize (whiten) the data (0 mean, 1 std dev):

```
def normalizeX (X):
    Xm   = np.mean(X,axis=1,keepdims=True)
    Xstd = np.std(X,axis=1,keepdims=True)
    Xnan = np.where(Xstd==0)
    Xstd[Xnan]=1
    Xm  [Xnan]=0
    X -= Xm
    X /= Xstd
    return X
```

Split the data into training, validation, testing:

```
def trainsplit(X,Y,percentV=0.2, percentT=0.2):
    m = X.shape[0]
    ix = list(range(m))
    np.random.shuffle (ix)                #randomize the DB index
    X = X[ix,:]
    Y = Y[ix,:]

    m_train = np.floor(m*(1-percentV-percentT)).astype(int) #cut off for training
    m_val   = m_train + np.floor(m*percentV).astype(int)    #cut off for validation data

    X_train = X[       :m_train,:]
    X_val   = X[m_train:m_val,  :]
    X_test  = X[  m_val:,       :]

    Y_train = Y[       :m_train,:]
    Y_val   = Y[m_train:m_val,  :]
    Y_test  = Y[  m_val:,       :]

    return X_train, Y_train, X_val, Y_val, X_test, Y_test
```