In [3]:
```
# These are all the modules we'll be using later. Make sure you can import them
# before proceeding further.
from __future__ import print_function
import imageio
import matplotlib.pyplot as plt
import numpy as np
import os
import sys
import tarfile
from IPython.display import display, Image
from sklearn.linear_model import LogisticRegression
from six.moves.urllib.request import urlretrieve
from six.moves import cPickle as pickle

# Config the matplotlib backend as plotting inline in IPython
%matplotlib inline
```

In [4]:

```
url = 'https://commondatastorage.googleapis.com/books1000/'
last_percent_reported = None
data_root = '.\Downloads' # Change me to store data elsewhere

def download_progress_hook(count, blockSize, totalSize):
  """A hook to report the progress of a download. This is mostly intended for users with
  slow internet connections. Reports every 5% change in download progress.
  """
  global last_percent_reported
  percent = int(count * blockSize * 100 / totalSize)

  if last_percent_reported != percent:
    if percent % 5 == 0:
      sys.stdout.write("%s%%" % percent)
      sys.stdout.flush()
    else:
      sys.stdout.write(".")
      sys.stdout.flush()

    last_percent_reported = percent

def maybe_download(filename, expected_bytes, force=False):
  """Download a file if not present, and make sure it's the right size."""
  dest_filename = os.path.join(data_root, filename)
  if force or not os.path.exists(dest_filename):
    print('Attempting to download:', filename)
    filename, _ = urlretrieve(url + filename, dest_filename, reporthook=download_progress_hook)
    print('\nDownload Complete!')
  statinfo = os.stat(dest_filename)
  if statinfo.st_size == expected_bytes:
    print('Found and verified', dest_filename)
  else:
    raise Exception(
      'Failed to verify ' + dest_filename + '. Can you get to it with a browser?')
  return dest_filename

train_filename = maybe_download('notMNIST_large.tar.gz', 247336696)
test_filename = maybe_download('notMNIST_small.tar.gz', 8458043)
```

```
Found and verified .\Downloads\notMNIST_large.tar.gz
Found and verified .\Downloads\notMNIST_small.tar.gz
```

In [9]:
```
num_classes = 10
np.random.seed(133)

def maybe_extract(filename, force=False):
  root = os.path.splitext(os.path.splitext(filename)[0])[0]  # remove .tar.gz
  if os.path.isdir(root) and not force:
    # You may override by setting force=True.
    print('%s already present - Skipping extraction of %s.' % (root, filename))
  else:
    print('Extracting data for %s. This may take a while. Please wait.' % root)
    tar = tarfile.open(filename)
    sys.stdout.flush()
    tar.extractall(data_root)
    tar.close()
  data_folders = [
    os.path.join(root, d) for d in sorted(os.listdir(root))
    if os.path.isdir(os.path.join(root, d))]
  if len(data_folders) != num_classes:
    raise Exception(
      'Expected %d folders, one per class. Found %d instead.' % (
        num_classes, len(data_folders)))
  print(data_folders)
  return data_folders

train_folders = maybe_extract(train_filename)
test_folders = maybe_extract(test_filename)
```

```
.\Downloads\notMNIST_large already present - Skipping extraction of .\Downloads\notMNIST_large.tar.gz.
['.\\Downloads\\notMNIST_large\\A', '.\\Downloads\\notMNIST_large\\B', '.\\Downloads\\notMNIST_large\\C', '.\\D
ownloads\\notMNIST_large\\D', '.\\Downloads\\notMNIST_large\\E', '.\\Downloads\\notMNIST_large\\F', '.\\Downloa
ds\\notMNIST_large\\G', '.\\Downloads\\notMNIST_large\\H', '.\\Downloads\\notMNIST_large\\I', '.\\Downloads\\no
tMNIST_large\\J']
.\Downloads\notMNIST_small already present - Skipping extraction of .\Downloads\notMNIST_small.tar.gz.
['.\\Downloads\\notMNIST_small\\A', '.\\Downloads\\notMNIST_small\\B', '.\\Downloads\\notMNIST_small\\C', '.\\D
ownloads\\notMNIST_small\\D', '.\\Downloads\\notMNIST_small\\E', '.\\Downloads\\notMNIST_small\\F', '.\\Downloa
ds\\notMNIST_small\\G', '.\\Downloads\\notMNIST_small\\H', '.\\Downloads\\notMNIST_small\\I', '.\\Downloads\\no
tMNIST_small\\J']
```

In [6]:
```
Image(filename=data_root+'\\notMNIST_large\\A\\a2F6b28udHRm.png')
```

Out[6]: 

In [7]:
```
Image(filename=data_root+'\\notMNIST_large\\A\\a3JvZWdlciAwNl81NS50dGY=.png')
```

Out[7]: 

In [10]:
```
image_size = 28  # Pixel width and height.
pixel_depth = 255.0  # Number of levels per pixel.

def load_letter(folder, min_num_images):
  """Load the data for a single letter label."""
  image_files = os.listdir(folder) ## it counts all files in the folder
  dataset = np.ndarray(shape=(len(image_files), image_size, image_size),
                       dtype=np.float32) #it creates a 3d array
  print(folder)
  num_images = 0
  for image in image_files:
    image_file = os.path.join(folder, image)
    try:
      image_data = (imageio.imread(image_file).astype(float) -
                    pixel_depth / 2) / pixel_depth
      if image_data.shape != (image_size, image_size):
        raise Exception('Unexpected image shape: %s' % str(image_data.shape))
      dataset[num_images, :, :] = image_data
      num_images = num_images + 1
    except (IOError, ValueError) as e:
      print('Could not read:', image_file, ':', e, '- it\'s ok, skipping.')

  dataset = dataset[0:num_images, :, :]
  if num_images < min_num_images:
    raise Exception('Many fewer images than expected: %d < %d' %
                    (num_images, min_num_images))

  print('Full dataset tensor:', dataset.shape)
  print('Mean:', np.mean(dataset))
  print('Standard deviation:', np.std(dataset))
  return dataset

def maybe_pickle(data_folders, min_num_images_per_class, force=False):
  dataset_names = []
  for folder in data_folders:                #for each letter's folder
    set_filename = folder + '.pickle'        #create a file with appropriate letter's name
    dataset_names.append(set_filename)       #add them to a returned set
    if os.path.exists(set_filename) and not force:
      # You may override by setting force=True.
      print('%s already present - Skipping pickling.' % set_filename)
    else:
      print('Pickling %s.' % set_filename)
```

```
      dataset = load_letter(folder, min_num_images_per_class) #it loads a letter from folder to a 3D array
      try:
        with open(set_filename, 'wb') as f:
          pickle.dump(dataset, f, pickle.HIGHEST_PROTOCOL)    #it dumps a 3D array to a file
      except Exception as e:
        print('Unable to save data to', set_filename, ':', e)

  return dataset_names

train_datasets = maybe_pickle(train_folders, 45000) # it creates 3D array for all letters in a train dataset
test_datasets = maybe_pickle(test_folders, 1800)# it creates 3D array for all letters in a test dataset
```

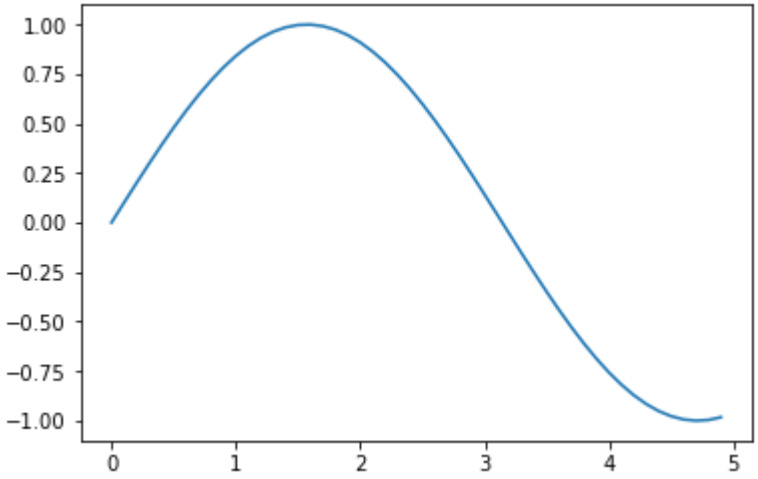
```
.\Downloads\notMNIST_large\A.pickle already present - Skipping pickling.
.\Downloads\notMNIST_large\B.pickle already present - Skipping pickling.
.\Downloads\notMNIST_large\C.pickle already present - Skipping pickling.
.\Downloads\notMNIST_large\D.pickle already present - Skipping pickling.
.\Downloads\notMNIST_large\E.pickle already present - Skipping pickling.
.\Downloads\notMNIST_large\F.pickle already present - Skipping pickling.
.\Downloads\notMNIST_large\G.pickle already present - Skipping pickling.
.\Downloads\notMNIST_large\H.pickle already present - Skipping pickling.
.\Downloads\notMNIST_large\I.pickle already present - Skipping pickling.
.\Downloads\notMNIST_large\J.pickle already present - Skipping pickling.
.\Downloads\notMNIST_small\A.pickle already present - Skipping pickling.
.\Downloads\notMNIST_small\B.pickle already present - Skipping pickling.
.\Downloads\notMNIST_small\C.pickle already present - Skipping pickling.
.\Downloads\notMNIST_small\D.pickle already present - Skipping pickling.
.\Downloads\notMNIST_small\E.pickle already present - Skipping pickling.
.\Downloads\notMNIST_small\F.pickle already present - Skipping pickling.
.\Downloads\notMNIST_small\G.pickle already present - Skipping pickling.
.\Downloads\notMNIST_small\H.pickle already present - Skipping pickling.
.\Downloads\notMNIST_small\I.pickle already present - Skipping pickling.
.\Downloads\notMNIST_small\J.pickle already present - Skipping pickling.
```

In [45]:
```
a_picle_filaname = train_datasets[0]
print (a_picle_filaname)
input = open(a_picle_filaname, 'rb')
_3Darray_A = pickle.load(input)
input.close()
print (_3Darray_A.ndim, '\t', _3Darray_A.shape)
print ('============================================ point to print in a screen of image at index 0 ============
print (_3Darray_A[0][0][4])
print ('============================================ array[0] of pixels of image at index 0 ====================
print (_3Darray_A[0][0])
print ('============================================ array[1] of pixels of image at index 0 ====================
print (_3Darray_A[0][1])
print ('============================================ Image at index 0 ==========================================
print (_3Darray_A[0])
print ('============================================ Full set of images ========================================
#print (_3Darray_A)
```

```
.\Downloads\notMNIST_large\A.pickle
3        (52909, 28, 28)
============================================ point to print in a screen of image at index 0 =============
============
-0.484314
============================================ array[0] of pixels of image at index 0 =====================
============
[-0.5        -0.5        -0.5        -0.5        -0.48431373 -0.5
 -0.19019608  0.46862745  0.49215686  0.5         0.5         0.5         0.5
  0.5         0.5         0.5         0.5         0.5         0.5
  0.49215686  0.46862745 -0.19019608 -0.5        -0.48431373 -0.5        -0.5
 -0.5        -0.5       ]
============================================ array[1] of pixels of image at index 0 =====================
============
[-0.5        -0.5        -0.5        -0.48823529 -0.5        -0.30392158
  0.43333334  0.5         0.49215686  0.5         0.5         0.5         0.5
  0.5         0.5         0.5         0.5         0.5         0.5
  0.49215686  0.5         0.43333334 -0.30392158 -0.5        -0.48823529
 -0.5        -0.5        -0.5       ]
```

In [48]:
```
x = np.arange(0, 5, 0.1);
y = np.sin(x)
plt.plot(x, y)
```

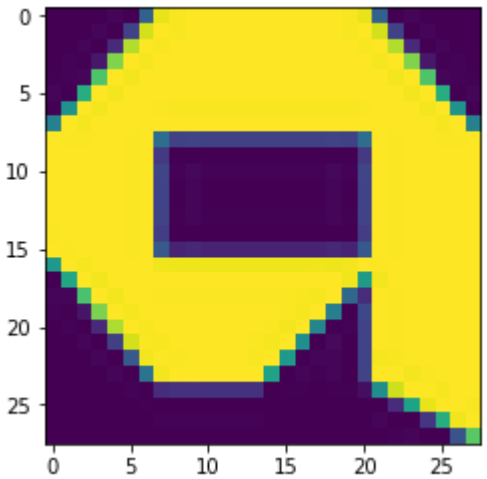
Out[48]:
```
[<matplotlib.lines.Line2D at 0x1e348563198>]
```



In [102]:
```
for i in range (10):
    x = _3Darray_A[i]
    print (i, " = ", x.shape)
    imageio.imwrite(data_root + '\\imageio\\picle_A_' + str(i) + '.png', x[:])

y = _3Darray_A[0]
plt.imshow(y)
```

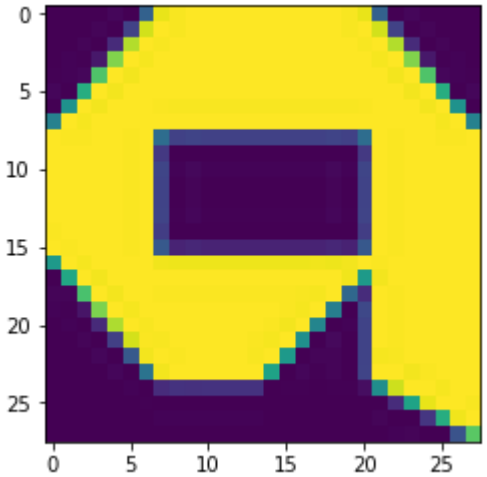
```
0  =  (28, 28)
1  =  (28, 28)
2  =  (28, 28)
3  =  (28, 28)
4  =  (28, 28)
5  =  (28, 28)
6  =  (28, 28)
7  =  (28, 28)
8  =  (28, 28)
9  =  (28, 28)

C:\Other_IT\Anaconda\lib\site-packages\imageio\core\util.py:104: UserWarning: Conversion from float32 to uint8,
range [-0.5, 0.5]
  'range [{2}, {3}]'.format(dtype_str, out_type.__name__, mi, ma))
```

Out[102]: <matplotlib.image.AxesImage at 0x1e34b9f3828>



In [103]:
```
image_data = (imageio.imread(data_root + '\\notMNIST_large\\A\\a29ydW5pc2hpLnR0Zg==.png').astype(float) -
                    pixel_depth / 2) / pixel_depth
plt.imshow(image_data)
```

Out[103]:
```
<matplotlib.image.AxesImage at 0x1e34ba82a90>
```



In [ ]: