```
from numpy import array
from numpy import argmax
from tensorflow.keras.utils import to_categorical
```

```
from numpy import array
from numpy import argmax
from tensorflow.keras.utils import to_categorical
```

```
data = [9,5,6,0,3,4,2,1,0,7,8,9,6]
data = array(data)
print(data)
```

```
[9 5 6 0 3 4 2 1 0 7 8 9 6]
```

```
# one hot encode
encoded = to_categorical(data)
print(encoded)
```

```
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]]
```

```
# importing required libraries
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.utils import to_categorical
import matplotlib.image as mpimg
from keras.datasets import mnist
import matplotlib.pyplot as plt
from matplotlib import pyplot
import numpy as np
```

```
# loading the required train and test MNIST dataset
(train_data_x, train_data_y), (test_data_x, test_data_y) = mnist.load_data()
```

```
print(train_data_x.shape[0], train_data_x.shape[1], train_data_x.shape[2])
```

```
60000 28 28
```

```
print(test_data_x.shape[0], test_data_x.shape[1], test_data_x.shape[2])
```

```
10000 28 28
```

```
# flattenning each of (28*28) images to a 784 vector of pixels using reshape()
pixel_num = train_data_x.shape[1] * train_data_x.shape[2]
train_data_x = train_data_x.reshape((train_data_x.shape[0], pixel_num)).astype('float32')
test_data_x = test_data_x.reshape((test_data_x.shape[0], pixel_num)).astype('float32')
```

```
# normalizing inputs from (0-255) to (0-1) by dividing each value by 255 which is the maximum pixel value on a grayscale
train_data_x = train_data_x / 255
test_data_x = test_data_x / 255
```

```
# one hot encoding of outputs
# transforming vector of class integers into a binary matrix
train_data_y = to_categorical(train_data_y)
test_data_y = to_categorical(test_data_y)
class_num = test_data_y.shape[1]
```

```
# Plotting first 10 dataset images
for i in range(9):
 plt.imshow(train_data_x[i].reshape(28, 28), cmap = plt.cm.binary)
 #show the plot
 plt.show()
```

[4]