In [1]:
```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
(X_train, y_train),(X_test,y_test)=keras.datasets.mnist.load_data()
```

In [3]:
```
len(X_train)
```

Out[3]: 60000

In [4]:
```
len(X_test)
```

Out[4]: 10000

In [5]:
```
X_train[0].shape #the shape of the immages 28*28 pixels
```

Out[5]: (28, 28)

In [6]:
```
X_train[0]
```

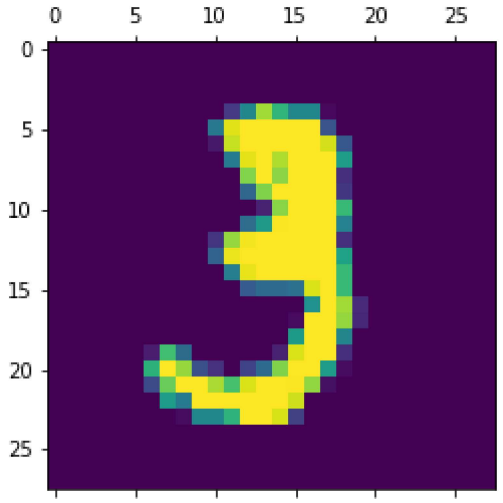
```
Out[6]: array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
         18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
        253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
        253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
        253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
        205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
         90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
        190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
        253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
        241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
```

```
  0,   0],
[ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
  0,   0],
[ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,
  0,   0],
[ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,
  0,   0],
[ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,
  0,   0],
[ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,
  0,   0],
[ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,
 148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,
  0,   0],
[ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,
 253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,
  0,   0],
[ 0,   0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,
 253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0],
[ 0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253, 253,
 195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0],
[ 0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244, 133,
  11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0],
[ 0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0],
[ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0],
[ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  0,   0],
[ 0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

```
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0]], dtype=uint8)
```

In [7]:
```
plt.matshow(X_train[10])
```

Out[7]:
```
<matplotlib.image.AxesImage at 0x1f800679908>
```

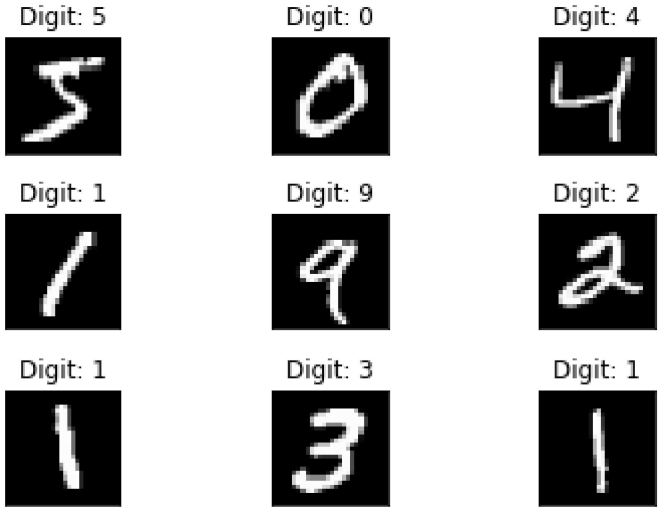


In [8]:
```
y_train[:5]
```

Out[8]:
```
array([5, 0, 4, 1, 9], dtype=uint8)
```

X_train contains 60,000 training images' data each of size 28x28 and y_train contains their corresponding labels. Similarly, X_test contains 10,000 testing images' data each of dimension 28x28 and y_test contains their corresponding labels

In [9]:
```
import matplotlib.pyplot as plt
fig = plt.figure()
for i in range(9):
 plt.subplot(3,3,i+1)
 plt.tight_layout()
 plt.imshow(X_train[i], cmap='gray', interpolation='none')
 plt.title("Digit: {}".format(y_train[i]))
 plt.xticks([])
 plt.yticks([])
```



In [10]:
```
# scaling the data is required
X_train = X_train/255
X_test = X_test/255
```

Using the convolutional neural network for the above dataset

In [11]:
```
from tensorflow.keras import layers, models
```

In [12]:
```
X_train = X_train.reshape(-1,28,28,1)   #training set
X_test = X_test.reshape(-1,28,28,1)   #testing set
#we should convert the 2-d matrix to a 1-d array
```

Try to run the code for the following composition and check if the accuracy is improved. Predict the corresponding results

In [14]:
```
convolutional_neural_network = models.Sequential([
    layers.Conv2D(filters=25, kernel_size=(3, 3), activation='relu', input_shape=(28,28,1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

In [ ]:

In [ ]: