讀入 MNIST

在 TensorFlow 套件中讀入 MNIST 同樣是很容易的，不論是訓練資料或者測試資料，都有分 `images` 與 `labels` 屬性

In [10]:

```
#import tensorflow, 無論 CPU 或 GPU 版本都是 import tensorflow as tf
import tensorflow as tf
import numpy as np

#將MNIST 手寫數字資料讀進來
mnist = tf.keras.datasets.mnist

# mnist 的load_data()會回傳已經先分割好的training data 和 testing data
# 並且將每個 pixel 的值從 Int 轉成 floating point 同時做normalize(這是很常見的preprocessing)
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

# 檢視結構
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
print("---")

print(len(x_train)) #training data 總共有60000張圖片
print(x_train[0].shape) #每張圖片(拿第一張當樣本)大小為 28x28

print("---")
# 檢視一個觀測值

#print(x_train[1, :])
print(y_train[1])    # 第一張訓練圖片的真實答案
print(np.argmax(y_train[1]))
```

```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
---
60000
(28, 28)
---
0
0
```

`MNIST` 的圖片是 28 像素 x 28 像素，每一張圖片就可以用 28 x 28 = 784 個數字來紀錄，因此 `print(x_train.shape)` 的輸出告訴我們有 55,000 張訓練圖片，每張圖片都有 784 個數字；而 `print(y_train.shape)` 的輸出告訴我們的是這 55,000 張訓練圖片的真實答案，`print(np.argmax(y_train[1]))` 的輸出告訴我們第一張訓練圖片的真實答案為 3。

我們也可以使用 `matplotlib.pyplot` 把第一張訓練圖片印出來看看

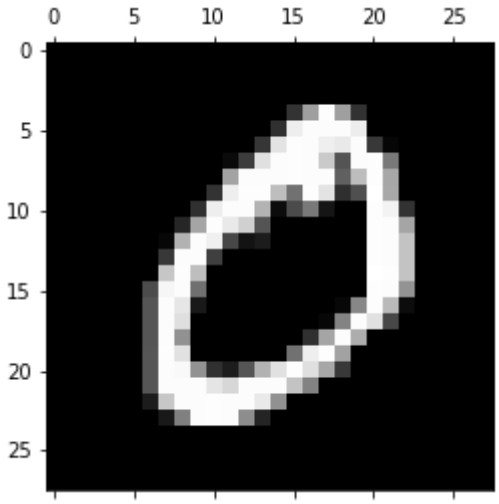
In [11]:

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline

# 讀入 MNIST
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

# 印出來看看
first_train_img = np.reshape(x_train[1, :], (28, 28))
plt.matshow(first_train_img, cmap = plt.get_cmap('gray'))
plt.show()
```



Softmax 函數

我們需要透過 `Softmax` 函數將分類器輸出的分數（Evidence）轉換為機率（Probability），然後依據機率作為預測結果的輸出，可想而知深度學習模型的輸出層會是一個 Softmax 函數。

Cross-entropy

不同於我們先前使用 `Mean Squared Error` 定義 `Loss`，在這個深度學習模型中我們改用 `Cross-entropy` 來定義 `Loss`。

TensorFlow 實作

我們建立一個可以利用 TensorBoard 檢視的深度學習模型，實作手寫數字辨識的分類器。

In [13]:

```
# 開始搭建model
# 利用 "Sequential" 把每層 layer 疊起來

# input 大小為 28 x 28

# 最後的 Dense(10) 且 activation 用 softmax
# 代表最後 output 為 10 個 class （0~9）的機率
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

# model每層定義好後需要經過compile
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [14]:

```
# 將搭好的 model 去 fit 我們的 training data
# 並evalutate 在 testing data 上
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test, verbose=2)
```

```
Train on 60000 samples
Epoch 1/5
60000/60000 [==============================] - 4s 63us/sample - loss: 0.2992
- accuracy: 0.9132
Epoch 2/5
60000/60000 [==============================] - 3s 49us/sample - loss: 0.1434
- accuracy: 0.9578
Epoch 3/5
60000/60000 [==============================] - 3s 48us/sample - loss: 0.1080
- accuracy: 0.9675
Epoch 4/5
60000/60000 [==============================] - 3s 50us/sample - loss: 0.0890
- accuracy: 0.9722
Epoch 5/5
60000/60000 [==============================] - 3s 49us/sample - loss: 0.0753
- accuracy: 0.9762
10000/1 - 0s - loss: 0.0368 - accuracy: 0.9789
```

Out[14]:

```
[0.06959879066096619, 0.9789]
```

針對 MNIST 資料建立了一個神經網絡模型，達到 92% 的準確率，同時我們也用了 TensorBoard 來視覺化。