In [1]:

```
import numpy as np
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.datasets import mnist
import matplotlib.pyplot as plt

```

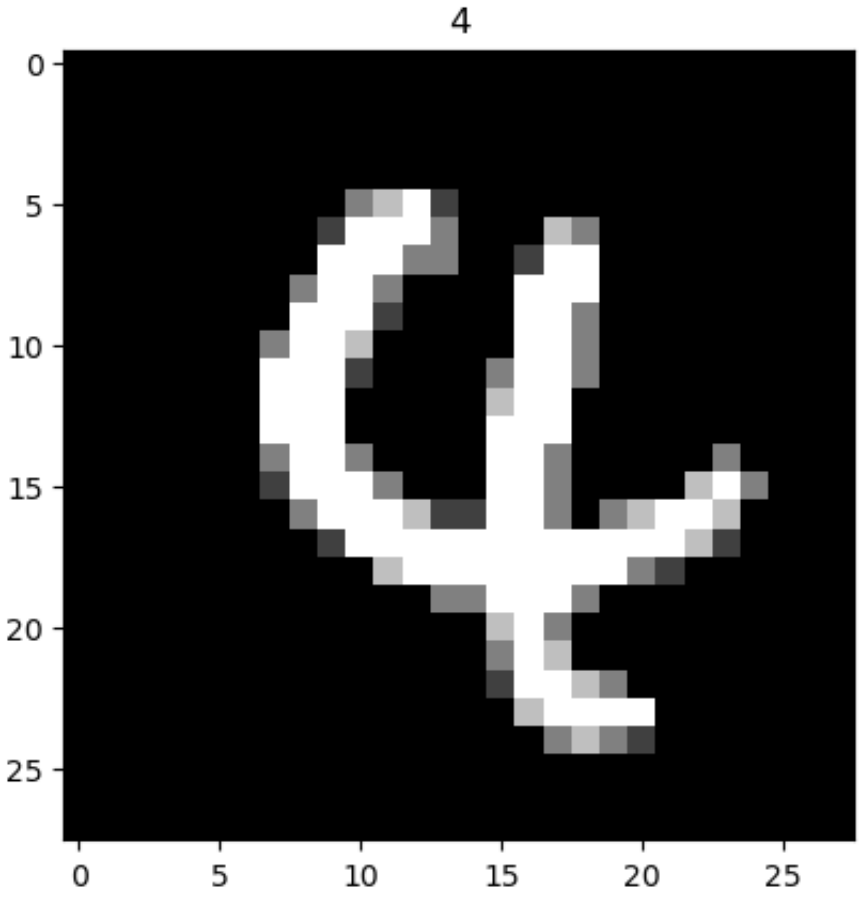
Let's Get a feel for training and testing data

getting a feel for the data is very important so you know how to work with it. We've randomly initialized some numbers and print their values. It's also important to note how the outputs are stored (are they one-hot encoded) or printed in base ten in a normal array. Clarifying through this well help us better understand our data and see what possible conversions we may have to do!

In [2]:

```
# y = w x + b
# Load the data and split it between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

randomIndx = np.random.randint(60000)
print(f"Random Index: {randomIndx}")
plt.imshow(x_train[randomIndx].reshape(28,28), cmap = 'gray')
plt.title(y_train[randomIndx])
print(f"Shape of image: {np.shape(x_train[randomIndx])}")
```

```
Random Index: 27694
Shape of image: (28, 28)
```



We should note a couple things:

Size of the images ar (28,28). Note the single channel. There are no 3 Dimensions. Knowing this will be helpful when we set up our Network. The outputs (y_test and y_train) are both column arrays that contain the VALUE of the output. This is helpful because now we know how to manage conversion of this data (if we have to ... we haven't made that decision yet).

In [3]:

```python
# y = w x + b
# Load the data and split it between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# a two-layer net using the Dense layer and Sequential Model.
model = keras.Sequential([
    layers.Dense(512 , activation="relu"),
    layers.Dense(12, activation ="softmax")
])

# compile the network and training the neural network.
model.compile(optimizer='rmsprop', loss='sparse_categorical_cro
              metrics=['accuracy'])

# Scale images to the [0, 1] range.
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1]*x_
x_train = x_train / 255.0
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1]*x_test
x_test = x_test / 255.0


model.fit(x_train, y_train , epochs = 20 , batch_size= 2048)
model.save('kwon_mn_mnist')

# Testin the network with the first 12 samples in the test set.
test_digits = x_test[:12]
# Using predict() function to test.
predictions = model.predict(test_digits)
```

```
Epoch 1/20
 1/30 [>.............................] - ETA: 4s - loss: 2.4999 -
accuracy: 0.1118

2022-11-07 23:05:56.817259: W tensorflow/core/platform/profile_uti
ls/cpu_utils.cc:128] Failed to get CPU frequency: 0 Hz

30/30 [==============================] - 1s 13ms/step - loss: 0.68
79 - accuracy: 0.8124
Epoch 2/20
30/30 [==============================] - 0s 12ms/step - loss: 0.32
45 - accuracy: 0.9060
Epoch 3/20
30/30 [==============================] - 0s 12ms/step - loss: 0.24
90 - accuracy: 0.9287
Epoch 4/20
30/30 [==============================] - 0s 13ms/step - loss: 0.20
29 - accuracy: 0.9433
Epoch 5/20
30/30 [==============================] - 0s 14ms/step - loss: 0.17
33 - accuracy: 0.9504
Epoch 6/20
30/30 [==============================] - 0s 13ms/step - loss: 0.14
91 - accuracy: 0.9573
Epoch 7/20
30/30 [==============================] - 0s 13ms/step - loss: 0.12
91 - accuracy: 0.9637
Epoch 8/20
30/30 [==============================] - 0s 14ms/step - loss: 0.11
```

```
30/30 [==============================] - 0s 14ms/step - loss: 0.11
31 - accuracy: 0.9677
Epoch 9/20
30/30 [==============================] - 0s 13ms/step - loss: 0.10
00 - accuracy: 0.9704
Epoch 10/20
30/30 [==============================] - 0s 14ms/step - loss: 0.08
95 - accuracy: 0.9749
Epoch 11/20
30/30 [==============================] - 0s 13ms/step - loss: 0.07
81 - accuracy: 0.9779
Epoch 12/20
30/30 [==============================] - 0s 13ms/step - loss: 0.07
10 - accuracy: 0.9800
Epoch 13/20
30/30 [==============================] - 0s 13ms/step - loss: 0.06
40 - accuracy: 0.9814
Epoch 14/20
30/30 [==============================] - 0s 14ms/step - loss: 0.05
75 - accuracy: 0.9838
Epoch 15/20
30/30 [==============================] - 0s 14ms/step - loss: 0.05
17 - accuracy: 0.9855
Epoch 16/20
30/30 [==============================] - 0s 14ms/step - loss: 0.04
70 - accuracy: 0.9868
Epoch 17/20
30/30 [==============================] - 0s 13ms/step - loss: 0.04
23 - accuracy: 0.9890
Epoch 18/20
30/30 [==============================] - 0s 14ms/step - loss: 0.03
85 - accuracy: 0.9897
Epoch 19/20
30/30 [==============================] - 0s 15ms/step - loss: 0.03
54 - accuracy: 0.9901
Epoch 20/20
30/30 [==============================] - 0s 14ms/step - loss: 0.03
19 - accuracy: 0.9917
INFO:tensorflow:Assets written to: kwon_mn_mnist/assets
1/1 [==============================] - 0s 32ms/step
```

In [4]:

```python
# evaluate the model
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 0.07337881624698639
Test accuracy: 0.9760000109672546
```

In [5]:

```
print(predictions)
tempList = []

#this loop iterates through y_test and converts a format so we
for row in y_test:
    tempClass = np.argmax(row)
    tempList.append(tempClass)


y_testClasses = np.array(tempList)
print(y_testClasses)
```

```
[[1.44472347e-06 2.99101650e-08 6.57276405e-06 2.91571225e-04
  1.41589105e-08 8.45729289e-07 7.71096104e-11 9.99664426e-01
  1.26563773e-05 2.24084652e-05 9.99307516e-13 8.92310206e-13]
 [5.54974822e-09 3.56796190e-05 9.99951005e-01 5.42535508e-06
  4.94209545e-13 2.92059508e-06 1.51445434e-07 4.04054875e-14
  4.75412162e-06 3.17934151e-11 7.81208032e-17 3.88316802e-16]
 [6.36759205e-06 9.96804118e-01 3.55096534e-04 2.83386125e-05
  1.82314398e-04 1.71336087e-05 3.64993030e-05 1.15164113e-03
  1.40762329e-03 1.08614095e-05 1.75649917e-09 1.98630157e-09]
 [9.99937057e-01 3.08657788e-09 2.59330500e-06 9.87204345e-08
  5.36258199e-07 3.10805422e-06 3.46652596e-05 1.32735750e-05
  3.56874743e-08 8.63077548e-06 2.88278346e-14 1.18985386e-14]
 [1.16696647e-05 1.04700444e-08 1.24195449e-05 8.47948286e-07
  9.94035184e-01 4.92480922e-06 8.43140424e-06 1.36689603e-04
  3.24466309e-05 5.75744687e-03 9.85034901e-11 8.44351603e-11]
 [3.20264292e-07 9.98297632e-01 4.87143006e-06 1.86719899e-06
  2.45769552e-05 5.08002280e-08 1.13112371e-07 1.59170525e-03
  7.80668925e-05 7.76395268e-07 2.22883344e-12 1.70134132e-12]
 [9.84833637e-09 9.36114404e-08 1.77022889e-06 9.53496695e-08
  9.92149115e-01 1.12618309e-05 2.20118932e-06 1.08620488e-05
  7.58226402e-03 2.42309630e-04 5.15564241e-12 1.95912175e-11]
 [1.35047799e-07 5.28885357e-05 4.64089462e-05 7.65813747e-04
  2.57141143e-03 2.45264149e-04 5.37173790e-08 1.32264031e-04
  1.43821482e-04 9.96041894e-01 1.33902375e-10 1.04873894e-10]
 [4.56454444e-07 2.09420975e-07 6.93887821e-04 1.33437525e-05
  8.93958262e-04 6.68453813e-01 3.24644804e-01 8.45350030e-08
  5.15084714e-03 1.48546416e-04 4.62052922e-13 5.51766965e-13]
 [4.95026109e-09 3.70093643e-11 2.21228547e-09 1.12585201e-06
  1.12112286e-02 4.05268281e-08 1.27036015e-09 5.06468001e-04
  4.27739316e-04 9.87853527e-01 3.04550866e-15 2.91926665e-15]
 [9.99982476e-01 8.04013800e-10 1.17394729e-05 1.40877612e-08
  8.15570000e-10 5.98221959e-07 1.24219071e-06 8.60893351e-07
  6.72937404e-08 2.97836027e-06 1.41606172e-14 7.83408914e-15]
 [1.49785637e-05 9.62539604e-10 4.47148068e-06 8.61413199e-08
  1.43998959e-05 1.84176242e-05 9.99748290e-01 3.55776457e-08
  1.99144939e-04 6.70651801e-08 1.31050693e-12 2.96967474e-13]]
[0 0 0 ... 0 0 0]
```

In [6]:

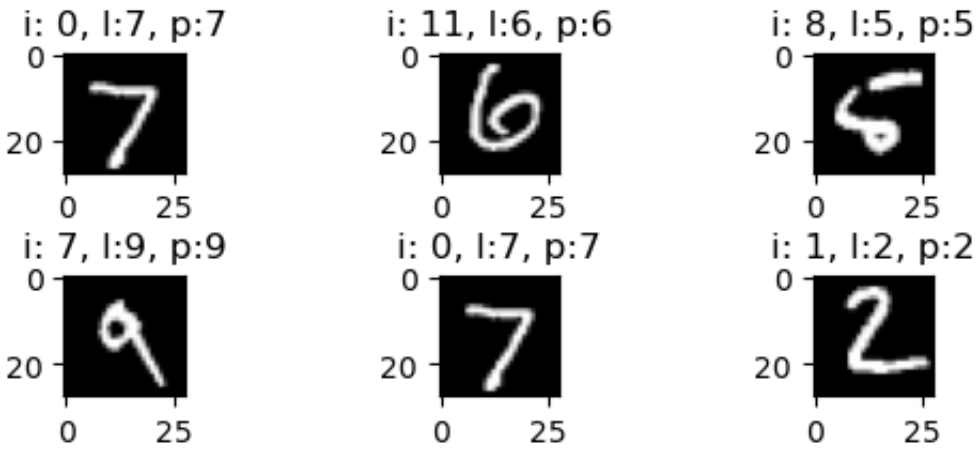
```
wrongPredictionIndexes = []
i = 0
for actual, prediction in zip('y_testClasses', predictions):
    if actual != prediction:
        wrongPredictionIndexes.append(i)
    i = i + 1

```

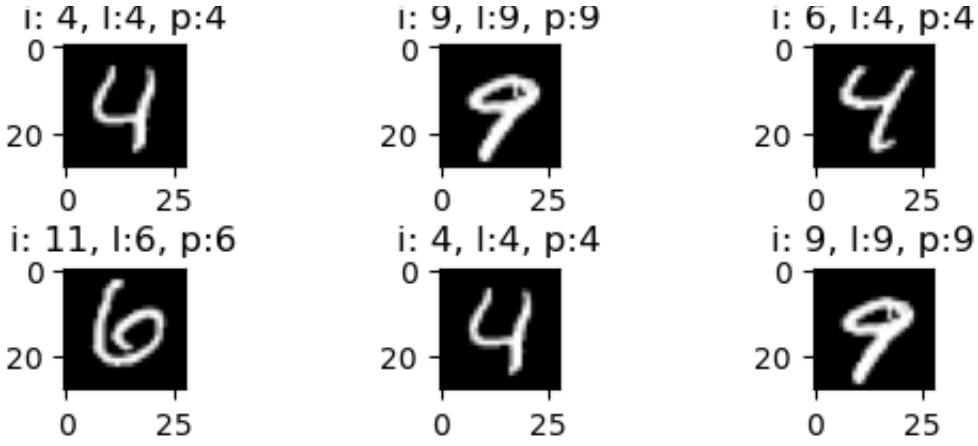
```
/var/folders/nj/5n17tpm16j1dszf8dzt_qfj00000gn/T/ipykernel_3539/52
0354650.py:4: FutureWarning: elementwise comparison failed; return
ing scalar instead, but in the future will perform elementwise com
parison
  if actual != prediction:
```

In [7]:

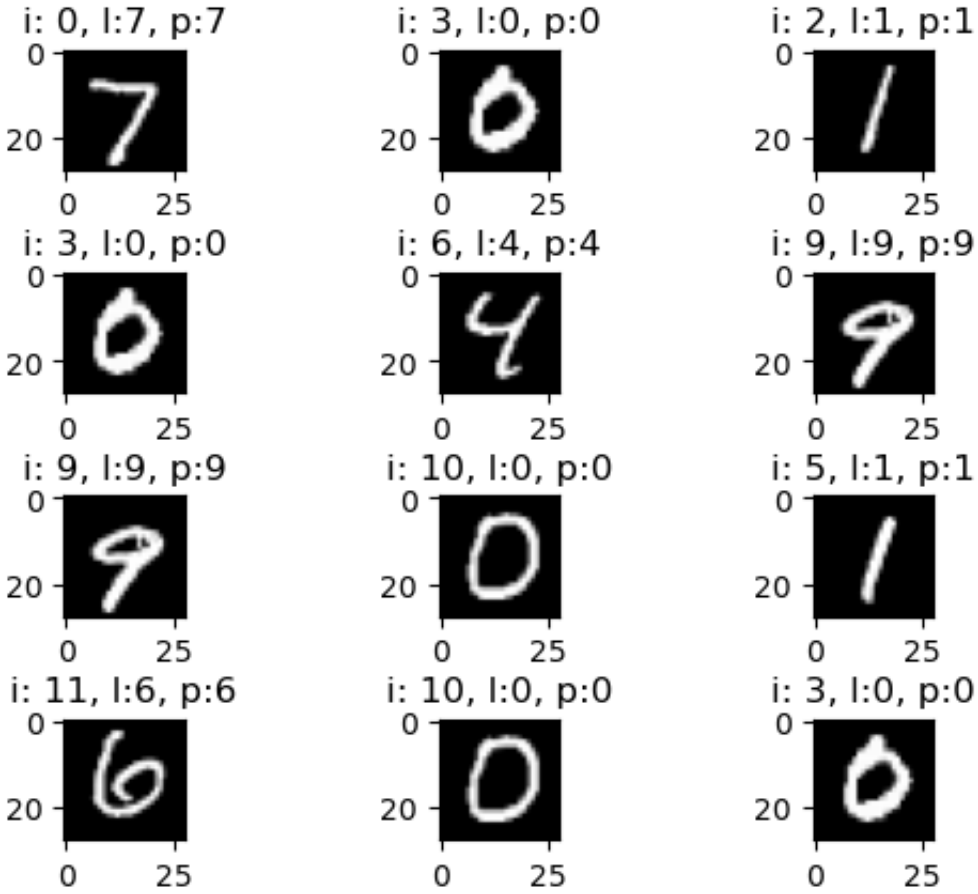
```
# plotting 12 tests
print(f' Example -1')
randomIntList = np.random.randint(len(wrongPredictionIndexes),
plot =1
plt.figure()
for randomNum in randomIntList:
    plt.subplot(4,3,plot)
    imageIndex = wrongPredictionIndexes[randomNum]
    plt.title(f'i: {imageIndex}, l:{ y_test[imageIndex]}, p:{pr
    plt.subplots_adjust(hspace=0.85)
    plt.imshow(x_test[imageIndex].reshape(28,28), cmap = 'gray'
    plot = plot+1
plt.show()

print(f' Example -2')
randomIntList = np.random.randint(len(wrongPredictionIndexes),
plot =1
plt.figure()
for randomNum in randomIntList:
    plt.subplot(4,3,plot)
    imageIndex = wrongPredictionIndexes[randomNum]
    plt.title(f'i: {imageIndex}, l:{ y_test[imageIndex]}, p:{pr
    plt.subplots_adjust(hspace=0.85)
    plt.imshow(x_test[imageIndex].reshape(28,28), cmap = 'gray'
    plot = plot+1
plt.show()

```

```
 Example -1
```





Example -2



In [ ]:
```
1
```