In [ ]:
```
from keras.datasets import mnist
import matplotlib.pyplot as plt

data = mnist.load_data()
```

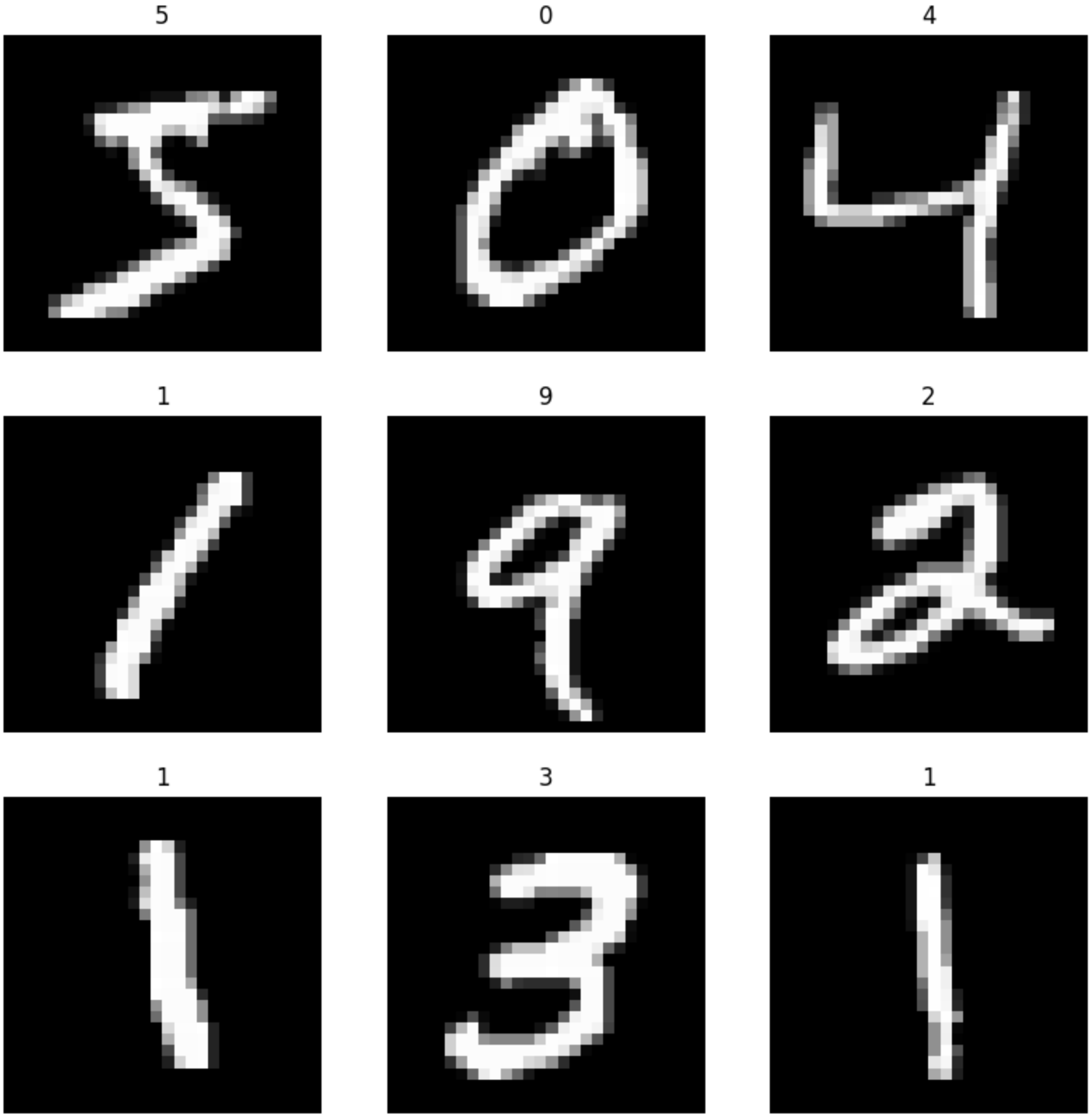
In [ ]:
```
(X_train, y_train), (X_test, y_test) = data
```

In [ ]:
```
X_train.shape
```

Out[ ]:
```
(60000, 28, 28)
```

In [ ]:
```
plt.figure(figsize=(10, 10))
for i in range(9):
      # Display the image
      ax = plt.subplot(3, 3, i + 1)
      plt.imshow(X_train[i], cmap='gray')
      # Print the label
      plt.title(y_train[i])
      plt.axis('off')

# Show the subplot
plt.show()
```



In [ ]:
```
X_train = X_train.reshape((X_train.shape[0], 28*28)).astype('float32')
X_test = X_test.reshape((X_test.shape[0], 28*28)).astype('float32')
```

In [ ]:
```
X_train = X_train / 255
X_test = X_test / 255
```

In [ ]:
```
from keras import utils


print(y_test.shape)

y_train = utils.to_categorical(y_train)
y_test = utils.to_categorical(y_test)

num_classes = y_test.shape[1]
```

```
print(y_test.shape)
```

```
(10000,)
(10000, 10)
```

In [ ]:
```
from keras.models import Sequential
from keras.layers import Dense
```

In [ ]:
```
model = Sequential()

model.add(Dense(32, input_dim = 28 * 28, activation= 'relu'))

model.add(Dense(64, activation = 'relu'))

model.add(Dense(10, activation  = 'softmax'))
```

In [ ]:
```
model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

In [ ]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                25120

 dense_1 (Dense)             (None, 64)                2112

 dense_2 (Dense)             (None, 10)                650

=================================================================
Total params: 27882 (108.91 KB)
Trainable params: 27882 (108.91 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:
```
model.fit(X_train, y_train, epochs= 10, batch_size = 100)
```

```
Epoch 1/10
600/600 [==============================] - 2s 2ms/step - loss: 0.4241 - accuracy: 0.8765
Epoch 2/10
600/600 [==============================] - 1s 2ms/step - loss: 0.1880 - accuracy: 0.9458
Epoch 3/10
600/600 [==============================] - 1s 2ms/step - loss: 0.1482 - accuracy: 0.9576
Epoch 4/10
600/600 [==============================] - 1s 2ms/step - loss: 0.1238 - accuracy: 0.9635
Epoch 5/10
600/600 [==============================] - 1s 1ms/step - loss: 0.1076 - accuracy: 0.9685
Epoch 6/10
600/600 [==============================] - 1s 2ms/step - loss: 0.0952 - accuracy: 0.9716
Epoch 7/10
600/600 [==============================] - 1s 2ms/step - loss: 0.0854 - accuracy: 0.9745
Epoch 8/10
600/600 [==============================] - 1s 2ms/step - loss: 0.0757 - accuracy: 0.9772
Epoch 9/10
600/600 [==============================] - 1s 2ms/step - loss: 0.0700 - accuracy: 0.9789
Epoch 10/10
600/600 [==============================] - 1s 2ms/step - loss: 0.0639 - accuracy: 0.9804
```

Out[ ]:
```
<keras.src.callbacks.History at 0x2a128de1f90>
```

In [ ]:
```
scores = model.evaluate(X_test, y_test)
print('Accuracy: ',scores[1] * 100)
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.0993 - accuracy: 0.9694
Accuracy:  96.93999886512756
```