In [1]:

```
import tensorflow as tf
```

In [2]:

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

In [3]:

```
x_train.shape
```

Out[3]:

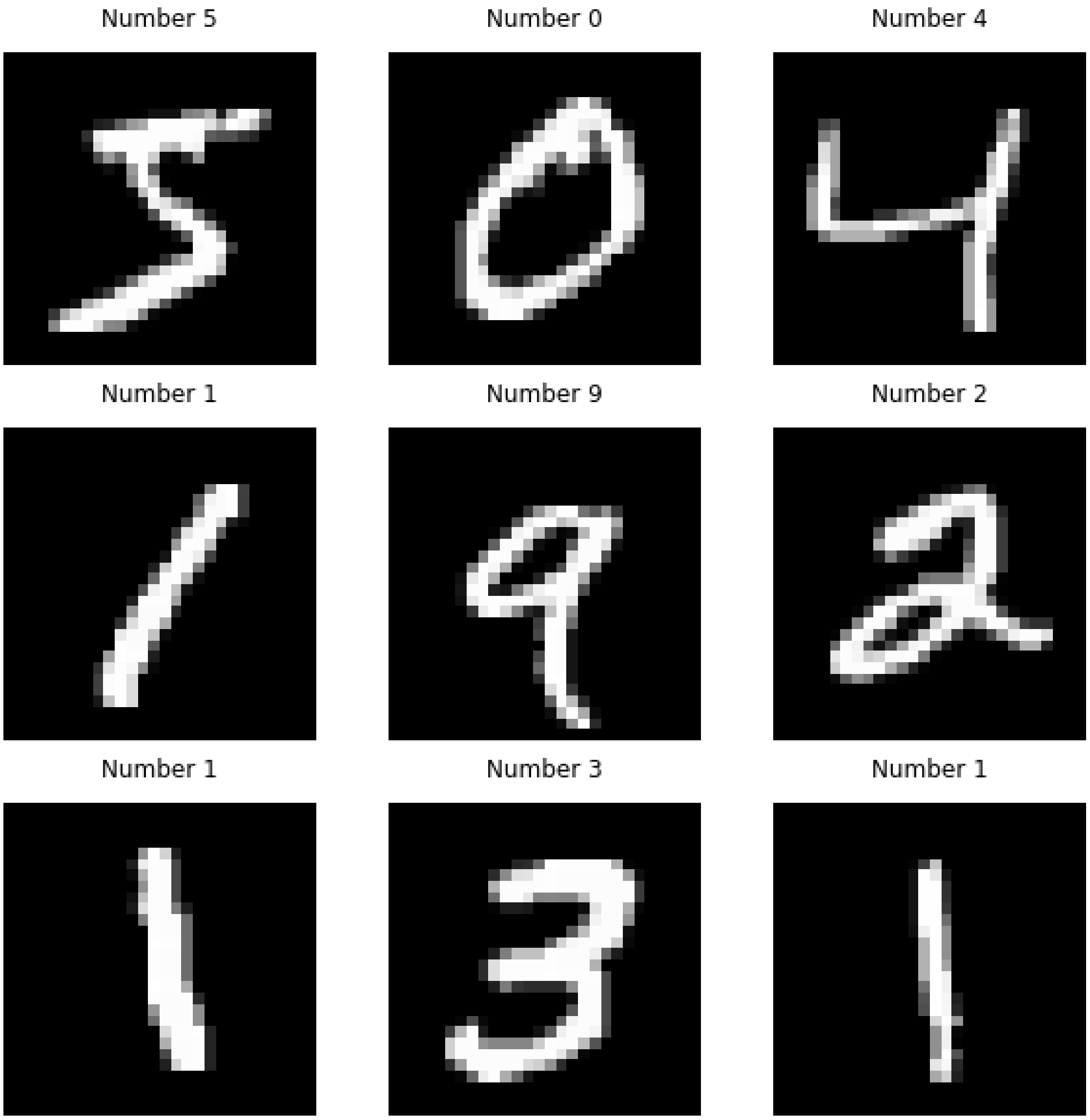
```
(60000, 28, 28)
```

In [4]:

```
import matplotlib.pyplot as plt
%matplotlib inline
fig, axs = plt.subplots(3, 3, figsize = (10, 10))
plt.gray()
for i, ax in enumerate(axs.flat):
    ax.matshow(x_train[i])
    ax.axis('off')
    ax.set_title('Number {}'.format(y_train[i]))
fig.show
```

Out[4]:

```
<bound method Figure.show of <Figure size 720x720 with 9 Axes>>
```



Number 5

Number 0

Number 4

Number 1

Number 9

Number 2

Number 1

Number 3

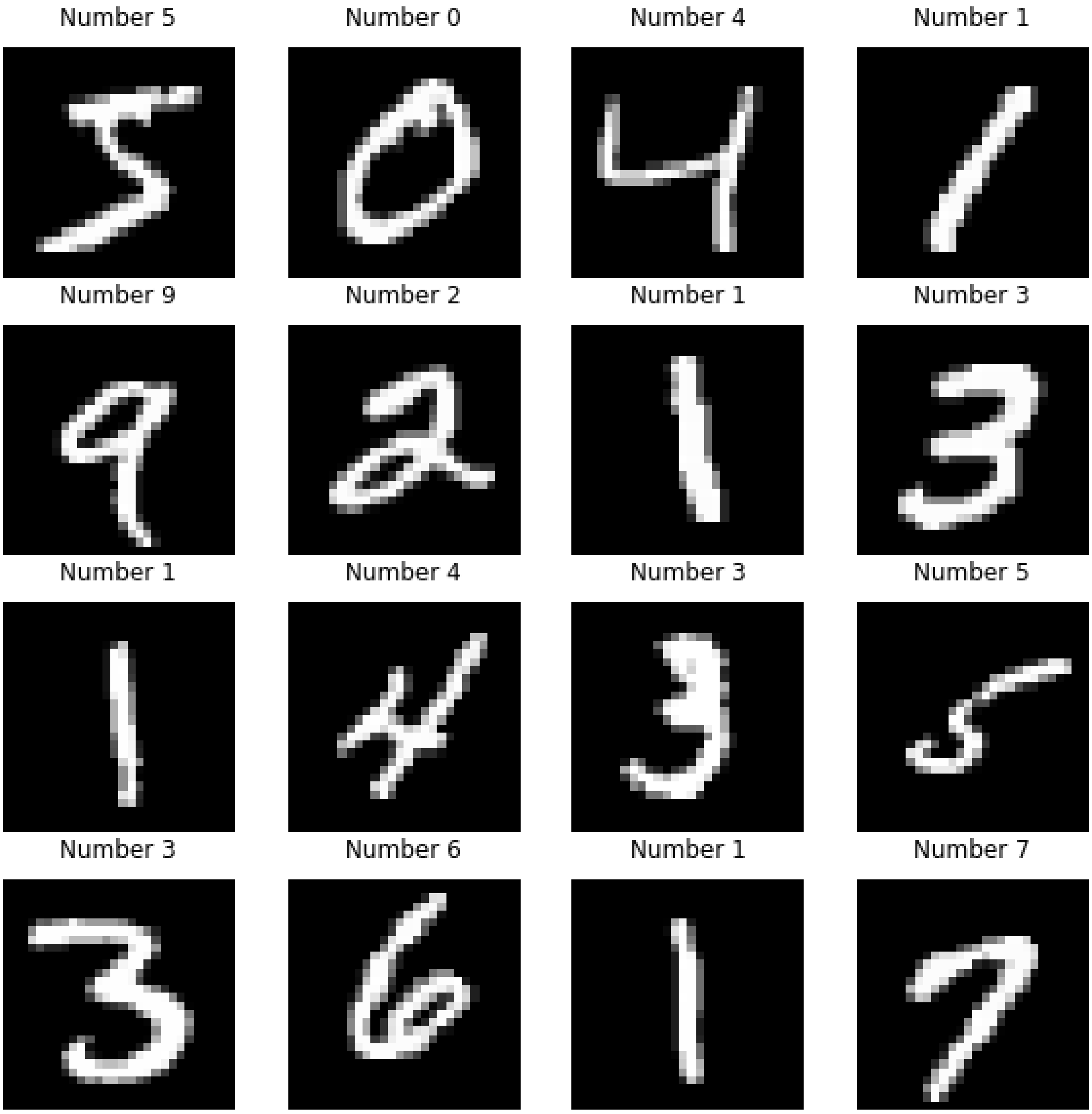
Number 1

In [5]:

```
import matplotlib.pyplot as plt
%matplotlib inline
fig, axs = plt.subplots(4, 4, figsize = (10, 10))
plt.gray()
for i, ax in enumerate(axs.flat):
    ax.matshow(x_train[i])
    ax.axis('off')
    ax.set_title('Number {}'.format(y_train[i]))
fig.show
```

Out[5]:

```
<bound method Figure.show of <Figure size 720x720 with 16 Axes>>
```



In [6]:

```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
input_shape = (28, 28, 1)
```

In [7]:

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255

print('x_train shape', x_train.shape)
print('NUmber of images in x_train', x_train.shape[0])
print('NUmber of images in x_test', x_test.shape[0])
```

```
x_train shape (60000, 28, 28, 1)
NUmber of images in x_train 60000
NUmber of images in x_test 10000
```

In [8]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
model = Sequential()
model.add(Conv2D(28, kernel_size=(3,3), input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation=tf.nn.relu))
model.add(Dropout(0.2))
model.add(Dense(10,activation=tf.nn.softmax))
```

In [9]:

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x=x_train,y=y_train, epochs=1)
```

```
1875/1875 [==============================] - 28s 15ms/step - loss: 0.2053 - accurac
y: 0.9378
```

Out[9]:

```
<tensorflow.python.keras.callbacks.History at 0x1f380466d60>
```

In [10]:

```
model.evaluate(x_test, y_test)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.0762 - accuracy: 0.
9752
```

Out[10]:

```
[0.07617513090372086, 0.9751999974250793]
```

In [ ]:

In [ ]: