In [ ]:
```
Shirsath Vaishnavi
Roll No: 61
```

In [100]:
```
!pip install keras
```

```
Requirement already satisfied: keras in c:\users\praja\anaconda3\lib\site-packages (2.10.0)
```

In [101]:
```
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
```

In [102]:
```
(x_train, y_train),(x_test, y_test) = mnist.load_data()
```

In [103]:
```
x_train.shape
```

Out[103]:
```
(60000, 28, 28)
```

In [104]:
```
x_test.shape
```

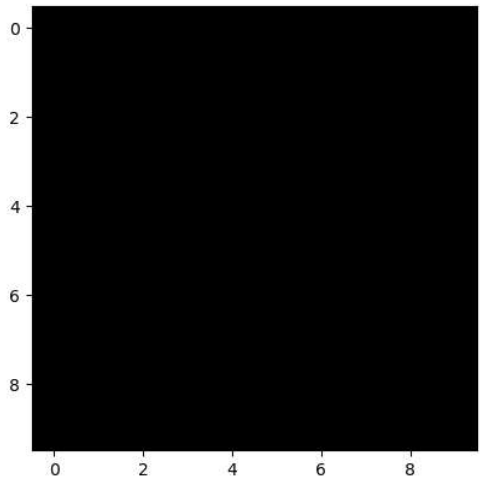
Out[104]:
```
(10000, 28, 28)
```

In [105]:
```
import matplotlib.pyplot as plt
```

In [106]:
```
x = np.zeros(100).reshape(10,10)
```

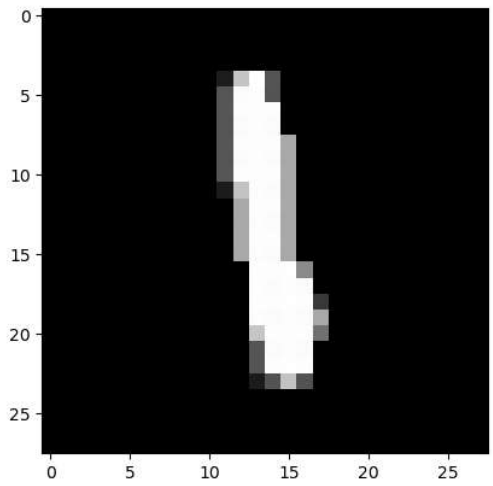
In [107]:
```
plt.imshow(x, cmap='gray')
```

Out[107]:
```
<matplotlib.image.AxesImage at 0x205a11d0430>
```



In [108]:
```
plt.imshow(x_train[200], cmap='gray')
```

Out[108]:
```
<matplotlib.image.AxesImage at 0x205a1204a30>
```

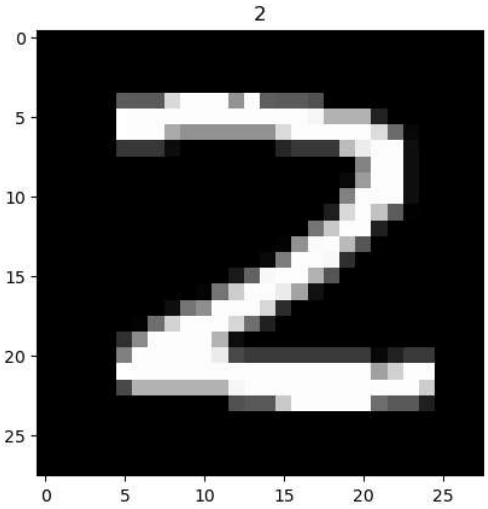


In [109]:
```
y_train[200]
```

Out[109]:
```
1
```

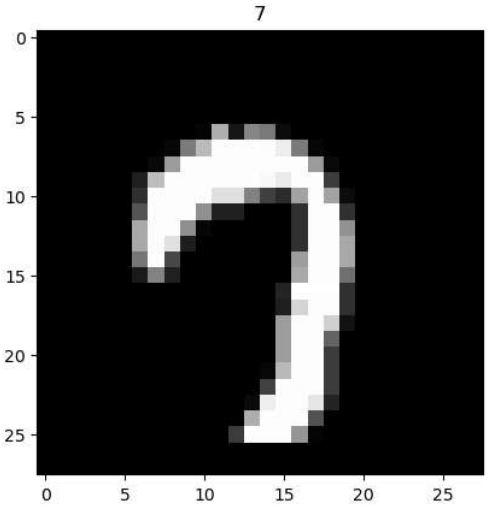
In [110]:
```
plt.imshow(x_train[220], cmap='gray')
plt.title(y_train[220])
```

Out[110]:
```
Text(0.5, 1.0, '2')
```



In [111]:
```
plt.imshow(x_test[220], cmap='gray')
plt.title(y_test[220])
```

Out[111]:
```
Text(0.5, 1.0, '7')
```



In [112]:
```
x = np.array([[2,3,5],[8,9,0]])
```

In [113]:
```
x
```

Out[113]:
```
array([[2, 3, 5],
       [8, 9, 0]])
```

In [114]:
```
x.shape
```

Out[114]:
```
(2, 3)
```

In [115]:
```
x = x.flatten()
x
```

Out[115]:
```
array([2, 3, 5, 8, 9, 0])
```

In [116]:
```
x.shape
```

Out[116]:
```
(6,)
```

In [117]:
```
img = x_train[3]
```

In [118]:
```
img.shape
```

Out[118]:
```
(28, 28)
```

In [119]:
```
img = img.flatten()
img.shape
```

Out[119]: (784,)

In [120]:
```
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
```

In [121]:
```
x_train.shape
```

Out[121]: (60000, 784)

In [122]:
```
x = np.array([8,6,5,7,0,3,4,2])
```

In [123]:
```
x/8
```

Out[123]: array([1.   , 0.75 , 0.625, 0.875, 0.   , 0.375, 0.5 , 0.25])

In [124]:
```
x_train = x_train / 255
```

In [125]:
```
x_test = x_test / 255
```

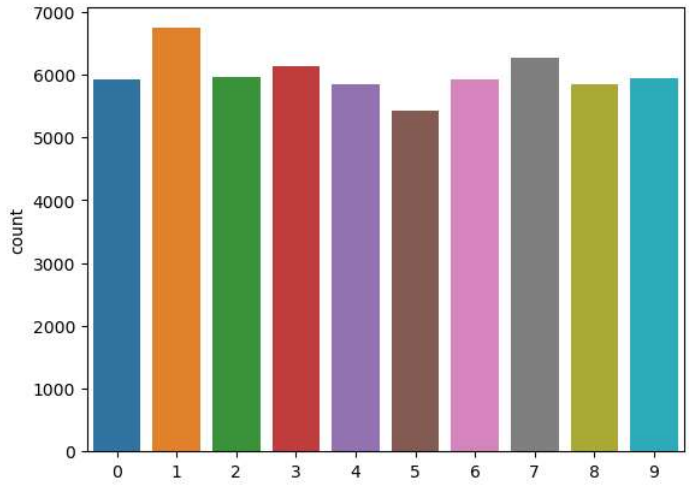
In [126]:
```
set(y_train)
```

Out[126]: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

In [127]:
```
import seaborn as sns
```

In [128]:
```
sns.countplot(x = y_train)
```

Out[128]: <AxesSubplot:ylabel='count'>



In [129]:
```
from collections import Counter
Counter(y_train)
```

Out[129]:
```
Counter({5: 5421,
         0: 5923,
         4: 5842,
         1: 6742,
         9: 5949,
         2: 5958,
         3: 6131,
         6: 5918,
         7: 6265,
         8: 5851})
```

In [130]:
```
from keras.utils import to_categorical
```

In [131]:
```
x = [0,2,2,1,0,1,2]
```

In [132]:
```
to_categorical(x)
```

Out[132]:
```
array([[1., 0., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]], dtype=float32)
```

In [133]:
```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

In [134]:
```
y_train.shape
```

Out[134]:
```
(60000, 10)
```

In [135]:
```
y_test.shape
```

Out[135]:
```
(10000, 10)
```

Define the network architecture

In [136]:
```
# Object of neural network
model = Sequential()

# Input layer
model.add(Dense(784, input_shape=(784,),
         activation='relu'))

# Hidden Layer-1
model.add(Dense(256, activation='relu'))

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

In [137]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 784)               615440

 dense_4 (Dense)             (None, 256)               200960

 dense_5 (Dense)             (None, 10)                2570

=================================================================
Total params: 818,970
Trainable params: 818,970
Non-trainable params: 0
_________________________________________________________________
```

Compile the model

In [138]:
```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics = ['accuracy'])
```

Train the model

In [139]:
```
history = model.fit(x_train, y_train, epochs=10,
         batch_size=10)
```

```
Epoch 1/10
6000/6000 [==============================] - 19s 3ms/step - loss: 0.1883 - accuracy: 0.9422
Epoch 2/10
6000/6000 [==============================] - 20s 3ms/step - loss: 0.0917 - accuracy: 0.9719
Epoch 3/10
6000/6000 [==============================] - 17s 3ms/step - loss: 0.0655 - accuracy: 0.9806
Epoch 4/10
6000/6000 [==============================] - 19s 3ms/step - loss: 0.0565 - accuracy: 0.9828
Epoch 5/10
6000/6000 [==============================] - 19s 3ms/step - loss: 0.0457 - accuracy: 0.9859
Epoch 6/10
6000/6000 [==============================] - 17s 3ms/step - loss: 0.0400 - accuracy: 0.9880
Epoch 7/10
6000/6000 [==============================] - 18s 3ms/step - loss: 0.0361 - accuracy: 0.9901
Epoch 8/10
6000/6000 [==============================] - 18s 3ms/step - loss: 0.0354 - accuracy: 0.9903
Epoch 9/10
6000/6000 [==============================] - 18s 3ms/step - loss: 0.0319 - accuracy: 0.9919
Epoch 10/10
6000/6000 [==============================] - 18s 3ms/step - loss: 0.0309 - accuracy: 0.9918
```

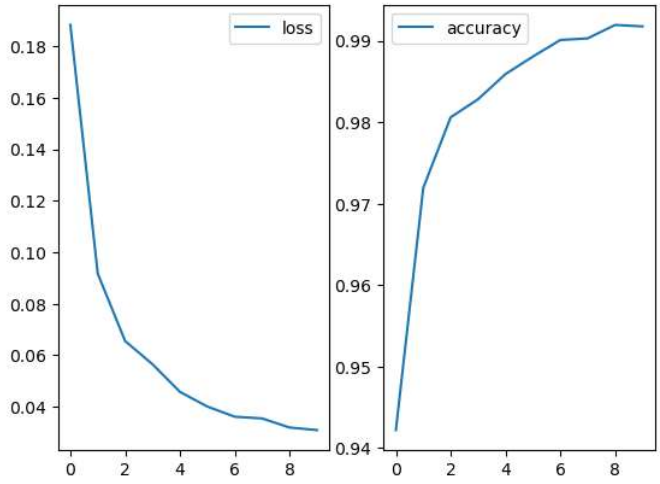
In [143]:
```
history.history
```

Out[143]:
```
{'loss': [0.1883452981710434,
  0.09167750179767609,
  0.06548676639795303,
  0.056462012231349945,
  0.045726992189884186,
  0.04003522917628288,
  0.036083586513996124,
  0.03541017696261406,
  0.03188195452094078,
  0.03089030273258686],
 'accuracy': [0.942216694355011,
  0.9719499945640564,
  0.9805999994277954,
  0.9828166961669922,
  0.9858999848365784,
  0.9880499839782715,
  0.9900833368301392,
  0.9902833104133606,
  0.9919333457946777,
  0.9917500019073486]}
```

In [144]:
```
plt.subplot(1,2,1)
plt.plot(history.history['loss'], label='loss')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['accuracy'],
         label='accuracy')
plt.legend()
```

Out[144]:
```
<matplotlib.legend.Legend at 0x205a74a7820>
```



In [142]:
```
model.evaluate(x_test, y_test, batch_size=1)
```

```
10000/10000 [==============================] - 5s 537us/step - loss: 0.1374 - accuracy: 0.9785
```

Out[142]:
```
[0.1374022215604782, 0.9785000085830688]
```