```
from keras.datasets import mnist

from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam

import random
np.random.seed(0)
```

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

In [ ]:
```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

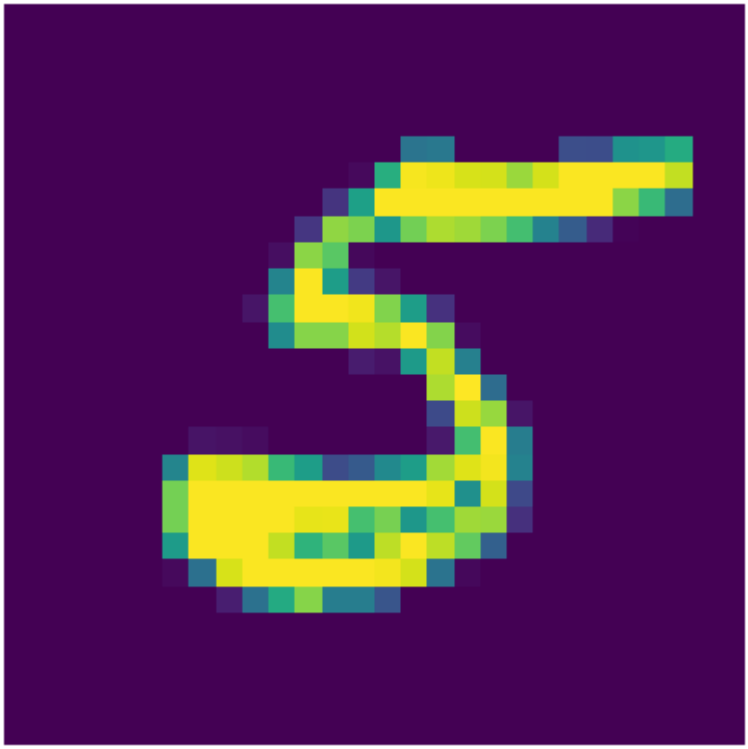
In [ ]:
```
x_train
```

Out[ ]:
```
array([[[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       ...,

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]]], dtype=uint8)
```

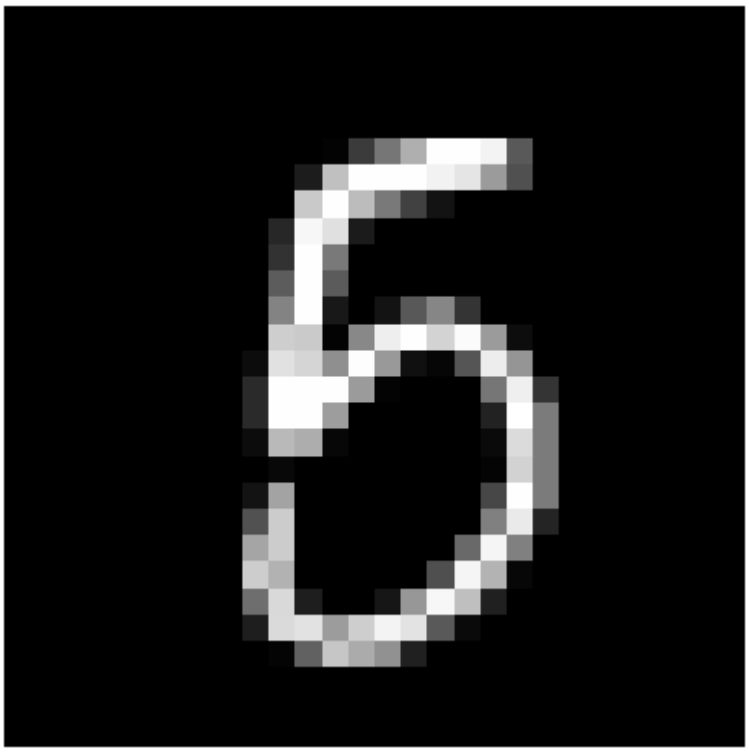
In [ ]:
```
set(y_train)
```

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

In [ ]:
```
plt.imshow(x_train[y_train==5][100])
plt.axis('off')
plt.show()
```



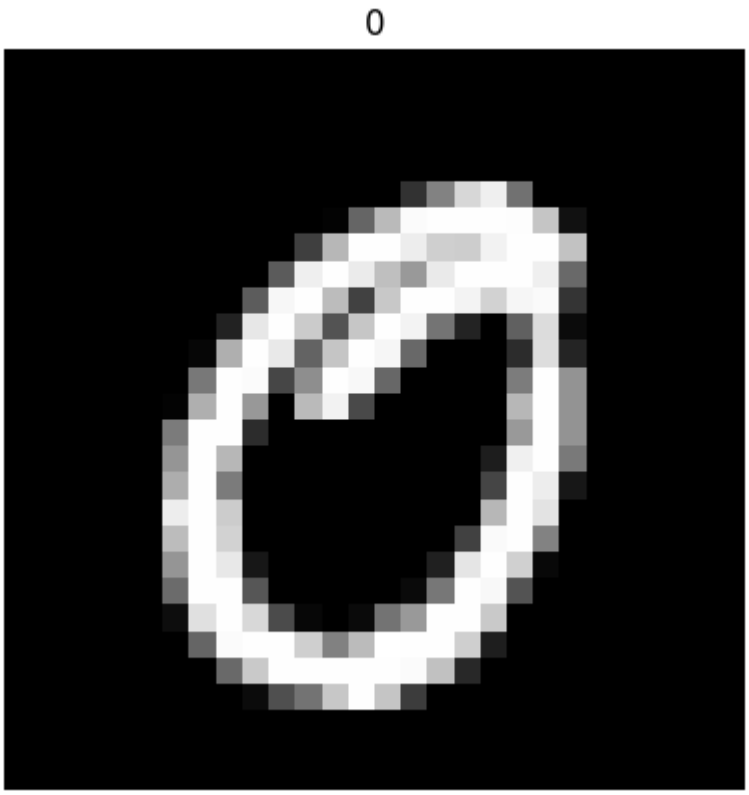
In [ ]:
```
plt.imshow(x_train[y_train==5][5000],cmap='gray')
plt.axis('off')
plt.show()
```

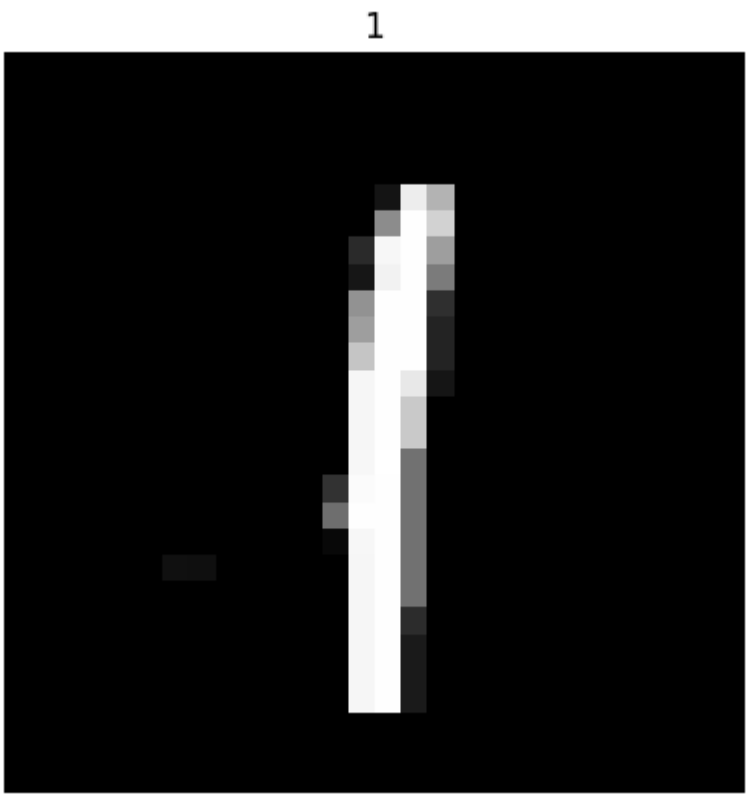


In [ ]:
```
for i in range(0,10):
  print(i,':',len(x_train[y_train==i]))
```

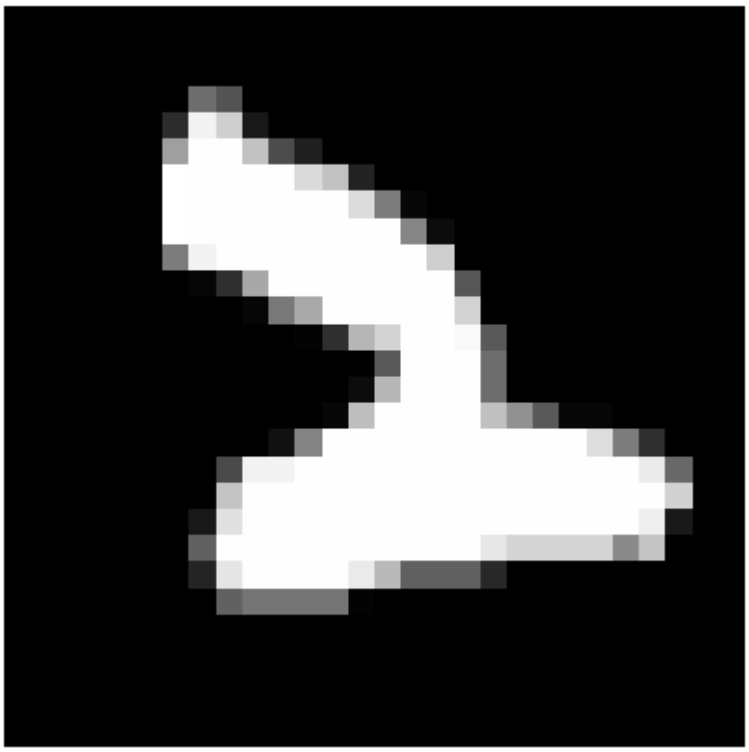
```
0 : 5923
1 : 6742
2 : 5958
3 : 6131
4 : 5842
5 : 5421
6 : 5918
7 : 6265
8 : 5851
9 : 5949
```

In [ ]:
```
for i in range(0,10):
  plt.imshow(x_train[y_train==i][np.random.randint(0,5000)], cmap='gray')
  plt.title(str(i))
  plt.axis('off')
  plt.show()
```

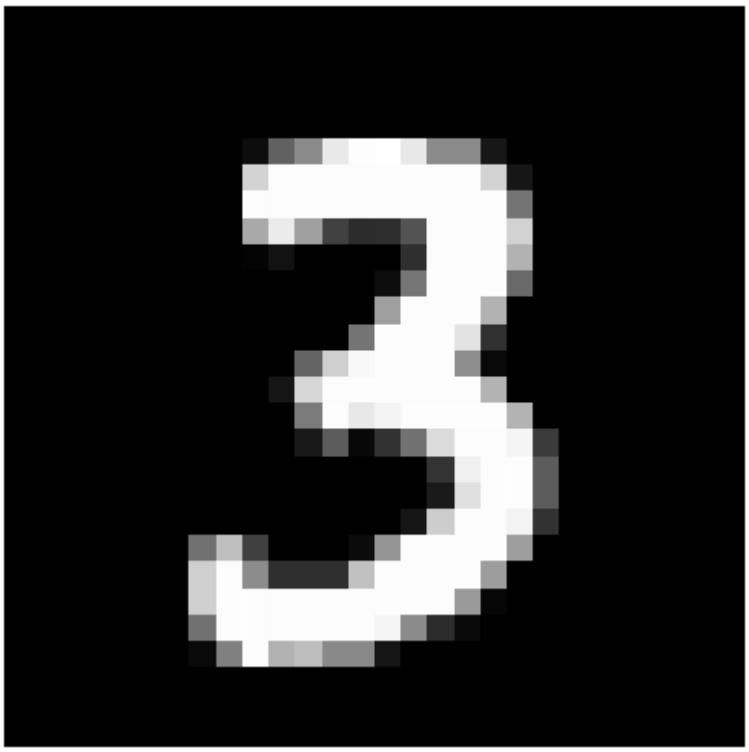




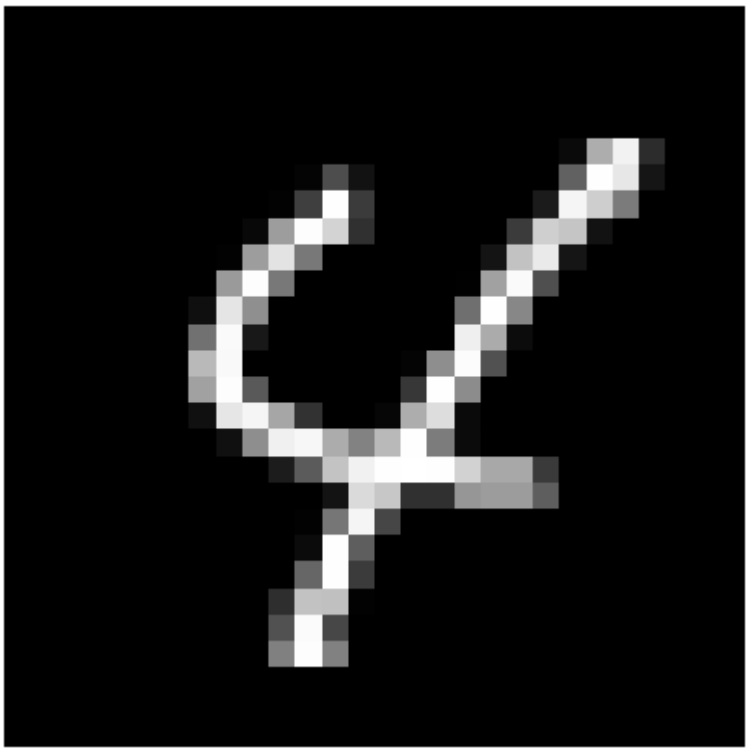
2



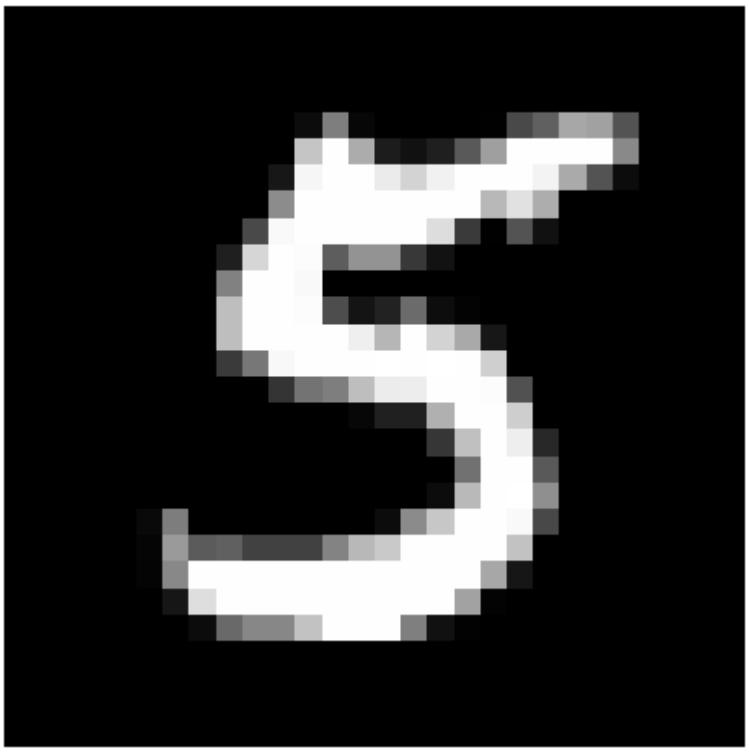
3



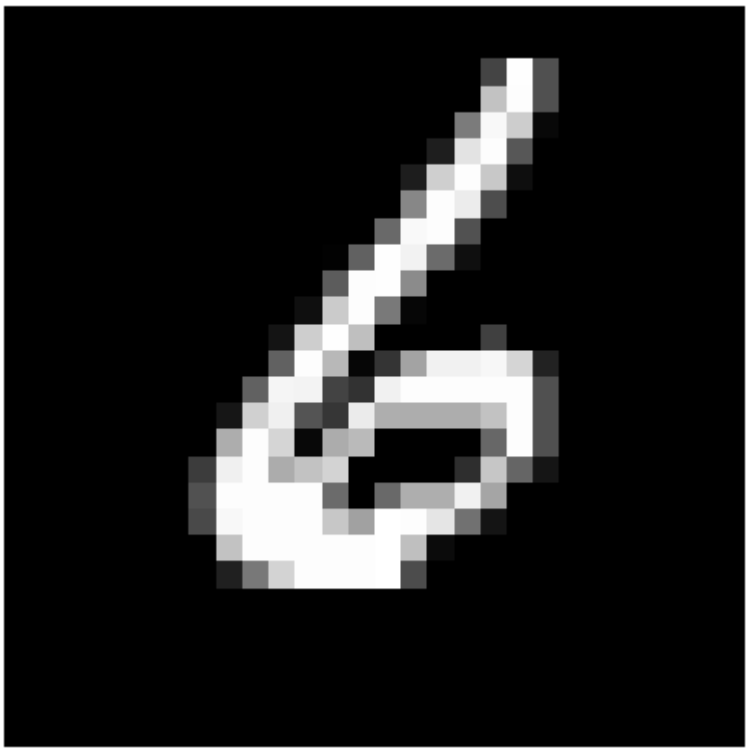
4



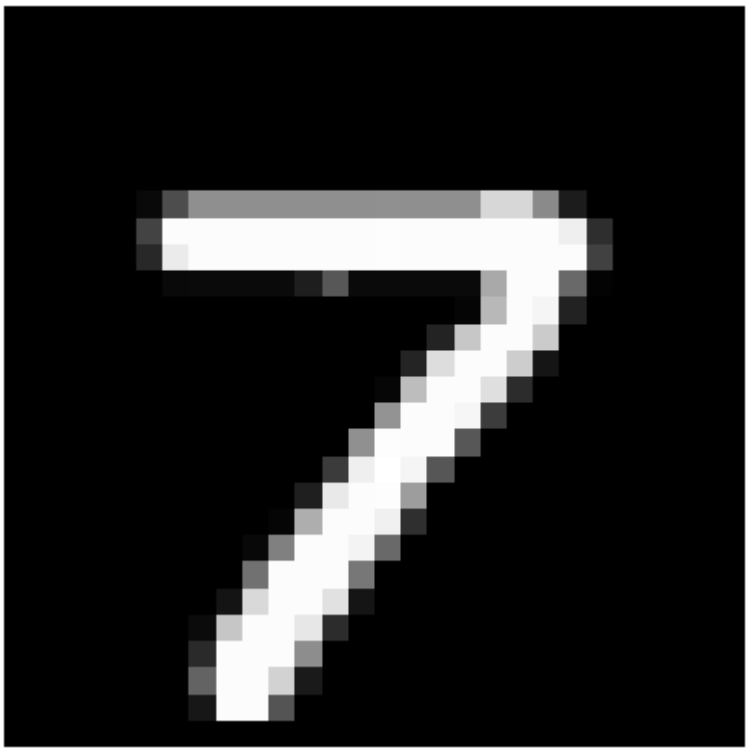
5



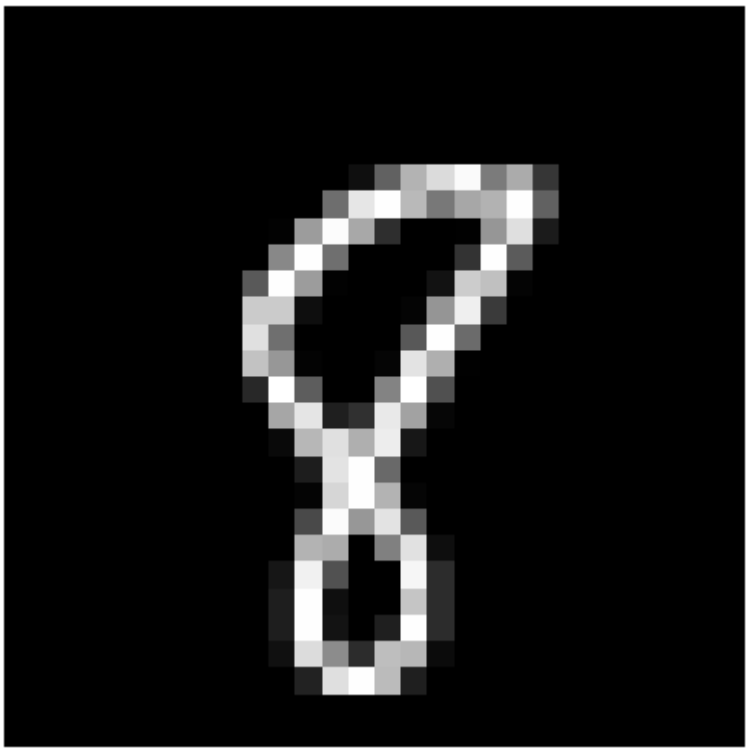
6



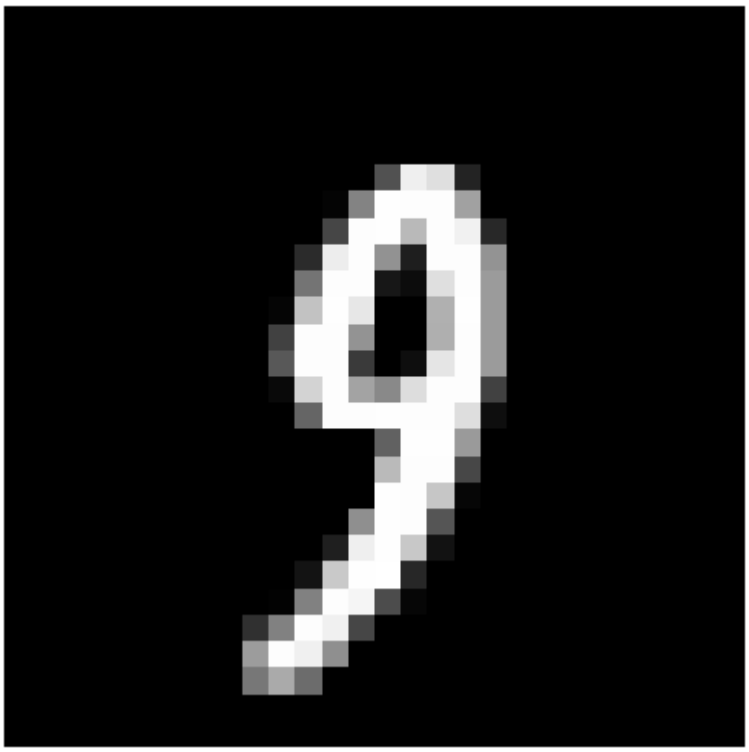
7



8



9



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

In [ ]:
```
y_train = to_categorical(y_train,10)
y_test = to_categorical(y_test)
```

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

In [ ]:
```
x_train
```

Out[ ]:
```
array([[[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       ...,

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]])
```

In [ ]:
```
y_train
```

Out[ ]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.]])
```

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

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

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

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

Out[ ]:
```
(60000, 784)
```

In [ ]:
```
#Model Training
model = Sequential()
#input and first hidden layer
model.add(Dense(35,input_dim=784,activation='relu'))
#hidden layers
model.add(Dense(15,activation='relu'))
model.add(Dense(25,activation='relu'))
#output layer
model.add(Dense(10,activation='softmax'))
model.compile(Adam(learning_rate=0.01),loss='categorical_crossentropy',metrics=['accuracy'])
```

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

Model: "sequential_2"

Layer (type)	Output Shape	Param #
dense_8 (Dense)	(None, 35)	27,475
dense_9 (Dense)	(None, 15)	540
dense_10 (Dense)	(None, 25)	400
dense_11 (Dense)	(None, 10)	260

Total params: 28,675 (112.01 KB)

Trainable params: 28,675 (112.01 KB)

Non-trainable params: 0 (0.00 B)

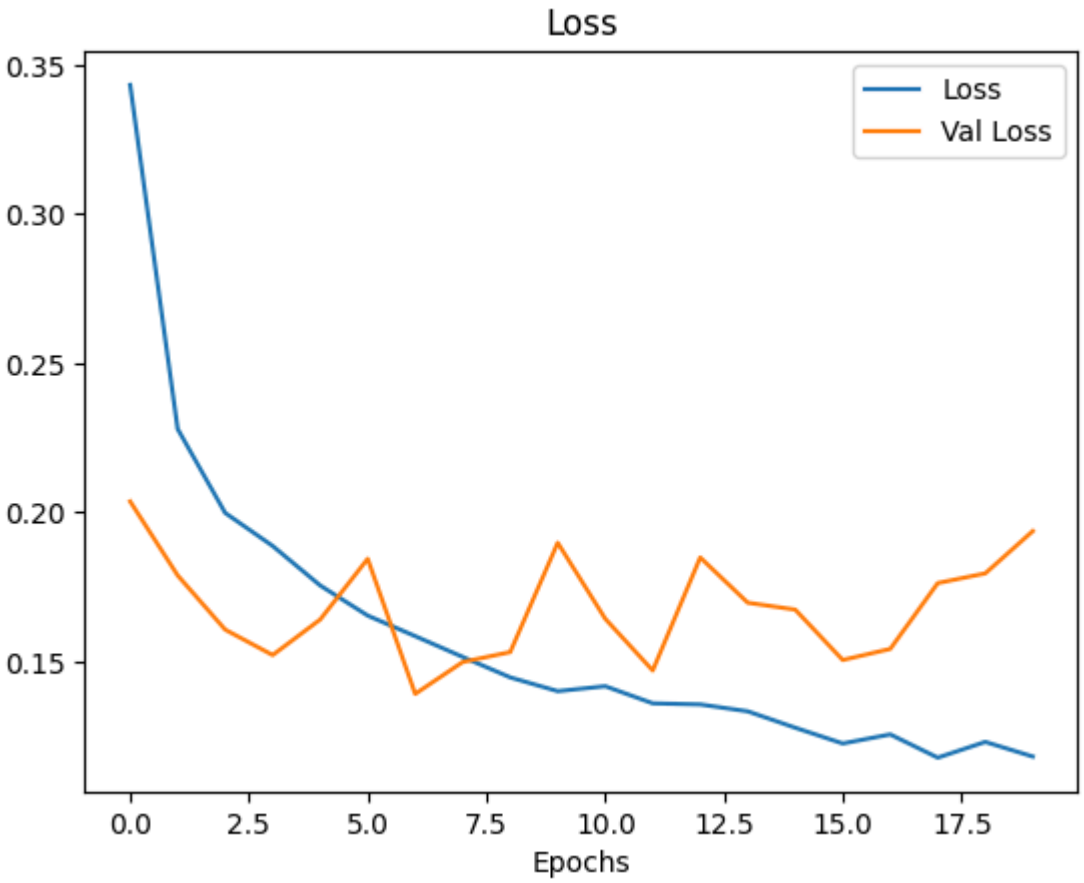
In [ ]:
```
h = model.fit(x_train,y_train,validation_split=0.1,epochs=20)
```

```
Epoch 1/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 8s 3ms/step - accuracy: 0.8378 - loss: 0.5062 - val_accuracy:
0.9410 - val_loss: 0.2036
Epoch 2/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.9334 - loss: 0.2295 - val_accuracy:
0.9490 - val_loss: 0.1788
Epoch 3/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9447 - loss: 0.1987 - val_accuracy:
0.9565 - val_loss: 0.1606
Epoch 4/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.9498 - loss: 0.1845 - val_accuracy:
0.9562 - val_loss: 0.1521
Epoch 5/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9515 - loss: 0.1745 - val_accuracy:
0.9578 - val_loss: 0.1640
Epoch 6/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9563 - loss: 0.1585 - val_accuracy:
0.9517 - val_loss: 0.1844
Epoch 7/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 10s 6ms/step - accuracy: 0.9583 - loss: 0.1581 - val_accuracy:
0.9602 - val_loss: 0.1391
Epoch 8/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9617 - loss: 0.1464 - val_accuracy:
0.9617 - val_loss: 0.1498
Epoch 9/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9614 - loss: 0.1455 - val_accuracy:
0.9627 - val_loss: 0.1531
Epoch 10/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.9636 - loss: 0.1385 - val_accuracy:
0.9548 - val_loss: 0.1897
Epoch 11/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9627 - loss: 0.1433 - val_accuracy:
0.9612 - val_loss: 0.1643
Epoch 12/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 13s 5ms/step - accuracy: 0.9644 - loss: 0.1345 - val_accuracy:
0.9640 - val_loss: 0.1469
Epoch 13/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.9658 - loss: 0.1302 - val_accuracy:
0.9573 - val_loss: 0.1848
Epoch 14/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9663 - loss: 0.1328 - val_accuracy:
0.9623 - val_loss: 0.1696
Epoch 15/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - accuracy: 0.9684 - loss: 0.1266 - val_accuracy:
0.9628 - val_loss: 0.1673
Epoch 16/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 8s 5ms/step - accuracy: 0.9684 - loss: 0.1221 - val_accuracy:
0.9637 - val_loss: 0.1504
Epoch 17/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 10s 5ms/step - accuracy: 0.9698 - loss: 0.1222 - val_accuracy:
0.9642 - val_loss: 0.1542
Epoch 18/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9727 - loss: 0.1100 - val_accuracy:
0.9652 - val_loss: 0.1762
Epoch 19/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.9674 - loss: 0.1276 - val_accuracy:
0.9612 - val_loss: 0.1795
Epoch 20/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9724 - loss: 0.1079 - val_accuracy:
0.9608 - val_loss: 0.1937
```

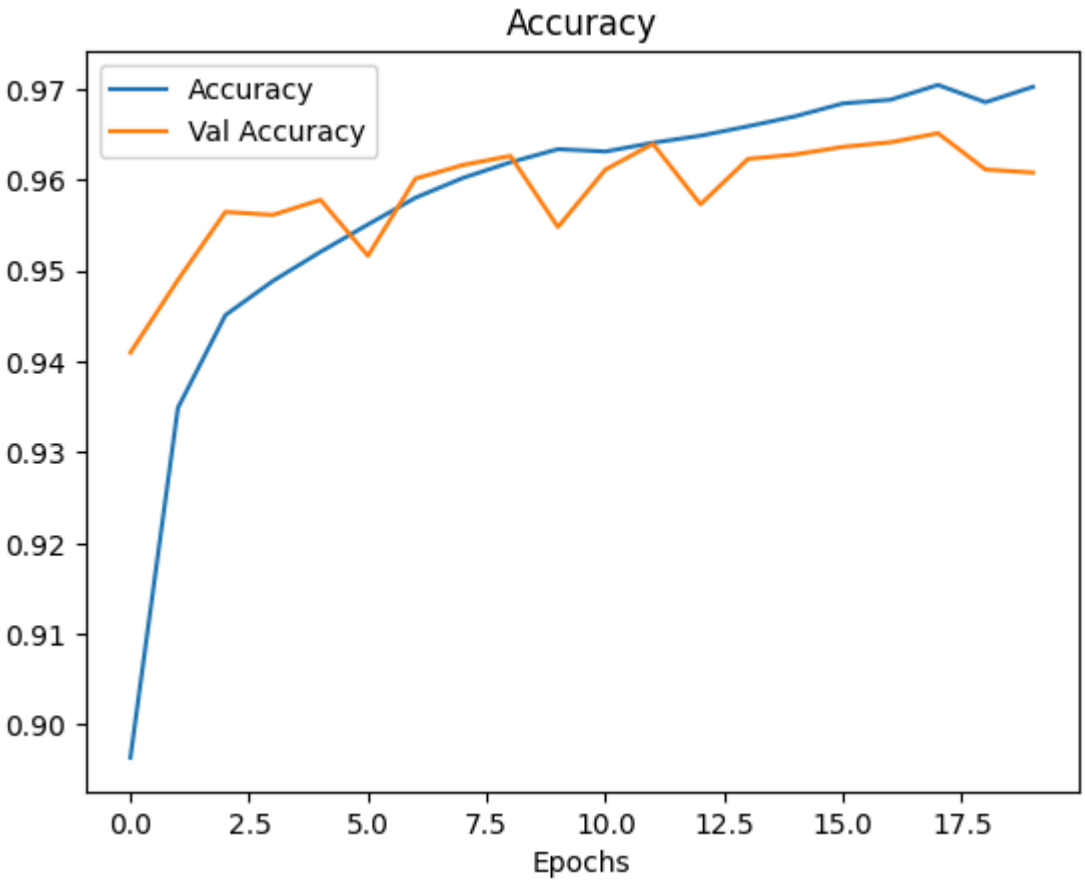
In [ ]:

```
plt.plot(h.history['loss'])
plt.plot(h.history['val_loss'])
plt.legend(['Loss','Val Loss'])
plt.title("Loss")
```

```
plt.xlabel("Epochs")
plt.show()
```



In [ ]:
```
plt.plot(h.history['accuracy'])
plt.plot(h.history['val_accuracy'])
plt.legend(['Accuracy','Val Accuracy'])
plt.title("Accuracy")
plt.xlabel("Epochs")
plt.show()
```

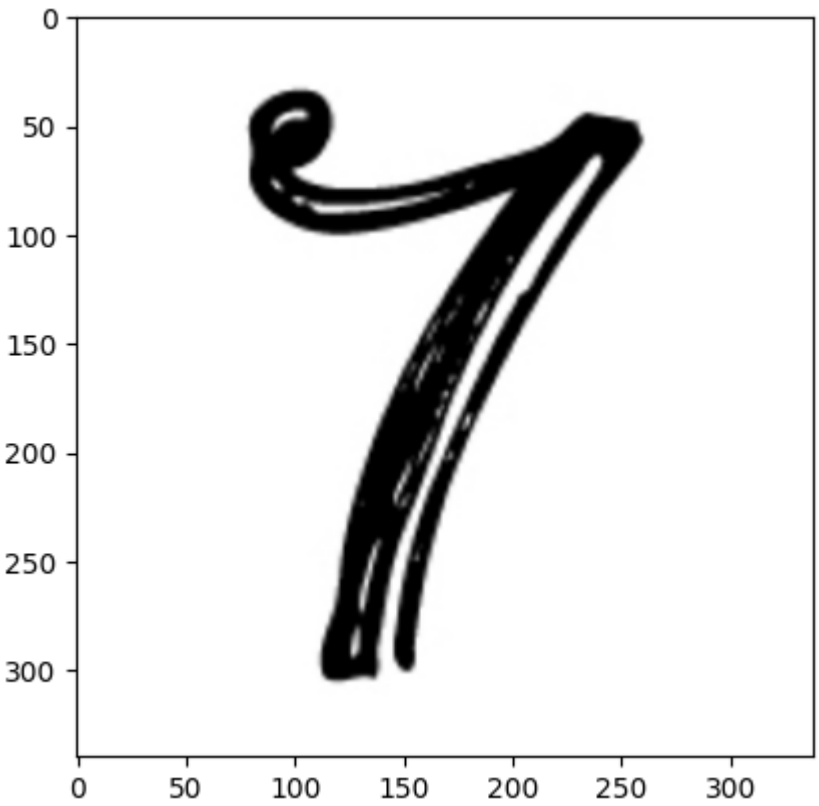


In [ ]:

Method 1 for Testing the Model

In [ ]:
```
import cv2
import matplotlib.pyplot as plt
```

In [ ]:
```
img = cv2.imread(r'/content/seven12.jpeg')
plt.imshow(img,cmap='gray')
plt.show()
```



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

Out[ ]:
```
(340, 338, 3)
```

In [ ]:
```
img = cv2.resize(img,(28,28))
img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
img = cv2.bitwise_not(img)

plt.imshow(img,cmap='gray')
plt.show()
```