```
import tensorflow as tf
from keras.models import Sequential
from keras.datasets import mnist
import matplotlib.pyplot as plt
import numpy as np
import random
```

```
(x_train,y_train),(x_test,y_test)=mnist.load_data()
x_train=x_train/255
x_test=x_test/255
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-dataset
11490434/11490434 [==============================] - 0s 0us/step
```

```
import keras
model=keras.Sequential()
model.add(keras.layers.Flatten(input_shape=(28,28)))
model.add(keras.layers.Dense(128,activation='relu'))
model.add(keras.layers.Dense(10,activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics=["Accu
```

```
H=model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=5)
```

```
Epoch 1/5
1875/1875 [==============================] - 18s 9ms/step - loss: 0.6331 - Accur
Epoch 2/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.3334 - Accura
Epoch 3/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2857 - Accura
Epoch 4/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2569 - Accura
```

```
Epoch 5/5
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2353 - Accura
```

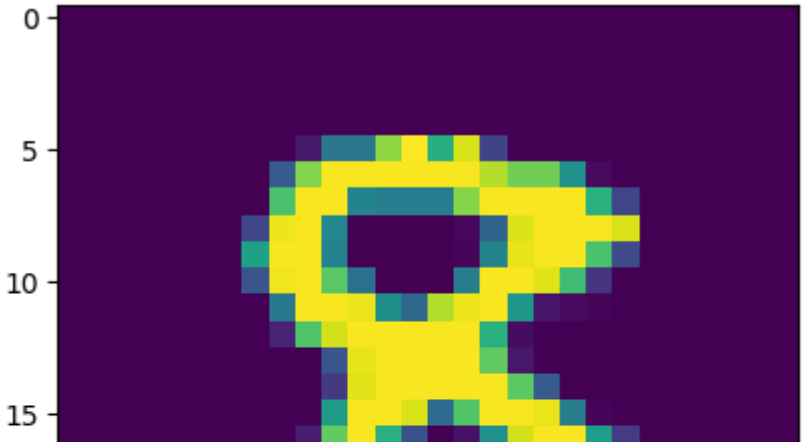
```
test_loss,test_acc=model.evaluate(x_test,y_test)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.2212 - Accuracy
```

```
print("Loss=%.3f"%test_loss)
print("Accuracy=%.3f"%test_acc)
```

```
Loss=0.221
Accuracy=0.938
```

```
n=random.randint(0,999)
plt.imshow(x_test[n])
plt.show()
```




```
prediction=model.predict(x_test)
print("The handwritten number in the image is %d"%np.argmax(prediction[n]))
```

```
313/313 [==============================] - 1s 3ms/step
The handwritten number in the image is 8
```