```
from keras.datasets import mnist
Data = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datase
11493376/11490434 [==============================] - 0s 0us/step
```

```
((x_train,y_train)),(x_test,y_test)= Data
```

```
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')
```

```
#Normalising value from 0-255 to 0-1

x_train = x_train/255
x_test= x_test/255
```

```
from keras.utils import np_utils

print(y_test.shape)

y_train=np_utils.to_categorical(y_train)
y_test=np_utils.to_categorical(y_test)

num_classes= y_test.shape[1]
print(y_test.shape)
```

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

Double-click (or enter) to edit

```
from keras.models import Sequential
from keras.layers import Dense
```

```
 model= Sequential ()
model.add(Dense(32,input_dim=28*28,activation='relu'))
model.add(Dense(64,activation= 'relu'))
model.add(Dense(10,activation='softmax'))
```

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

```
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: 27,882
Trainable params: 27,882
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(x_train, y_train, epochs=10, batch_size=100)
```

```
Epoch 1/10
600/600 [==============================] - 2s 3ms/step - loss: 0.7728 - accurac
Epoch 2/10
600/600 [==============================] - 2s 3ms/step - loss: 0.2050 - accurac
Epoch 3/10
600/600 [==============================] - 2s 3ms/step - loss: 0.1511 - accurac
Epoch 4/10
600/600 [==============================] - 2s 3ms/step - loss: 0.1257 - accurac
Epoch 5/10
600/600 [==============================] - 2s 3ms/step - loss: 0.1057 - accurac
Epoch 6/10
600/600 [==============================] - 2s 3ms/step - loss: 0.0910 - accurac
Epoch 7/10
600/600 [==============================] - 2s 3ms/step - loss: 0.0816 - accurac
Epoch 8/10
600/600 [==============================] - 2s 3ms/step - loss: 0.0729 - accurac
Epoch 9/10
600/600 [==============================] - 2s 3ms/step - loss: 0.0636 - accurac
Epoch 10/10
600/600 [==============================] - 2s 4ms/step - loss: 0.0619 - accurac
<tensorflow.python.keras.callbacks.History at 0x7f75f7d68f90>
```

```
score= model.evaluate(x_test, y_test)
print(score)
```

```
313/313 [==============================] - 0s 1ms/step - loss: 0.1039 - accurac
[0.10392550379037857, 0.9689000248908997]
```