Exercise 2

In the course you learned how to do classification using Fashion MNIST, a data set containing items of clothing. There's another, similar dataset called MNIST which has items of handwriting -- the digits 0 through 9.

Write an MNIST classifier that trains to 99% accuracy or above, and does it without a fixed number of epochs -- i.e. you should stop training once you reach that level of accuracy.

Some notes:

1. It should succeed in less than 10 epochs, so it is okay to change epochs to 10, but nothing larger
2. When it reaches 99% or greater it should print out the string "Reached 99% accuracy so cancelling training!"
3. If you add any additional variables, make sure you use the same names as the ones used in the class

```
import tensorflow as tf

# lets define the callback here
class endTrainingCallback(tf.keras.callbacks.Callback):

  def on_epoch_end(self, epoch, logs={}):
    if(logs.get('acc') > 0.99):
      print("Reached 99% accuracy! We are done training")
      self.model.stop_training = True


mnist = tf.keras.datasets.mnist

#callbacks = endTrainingCallback() for now I don't want to use the callback function

(x_train, y_train),(x_test, y_test) = mnist.load_data()

# values between 0 - 1 are easier for the neural network
x_train, x_test = x_train / 255.0, x_test / 255.0

# Here let's define our neural network
model = tf.keras.models.Sequential([
  # We want to build a convolutional neural network
    tf.keras.layers.Flatten(input_shape=(28,28)),
    tf.keras.layers.Dense(512, activation=tf.nn.relu), # relu ensures we are passing v
    tf.keras.layers.Dense(10, activation=tf.nn.softmax) # picks the largest value off
])

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

# training neural network with data (60K images)
model.fit(x_train, y_train, epochs=15)

# evaluate our neural network on the test data (10K images)
results = model.evaluate(x_test, y_test)

print("The accuracy of the neural network is %.2f percent" % (results[1] * 100))
```

```
Epoch 1/15
60000/60000 [==============================] - 22s 362us/sample - loss: 0.2047
Epoch 2/15
60000/60000 [==============================] - 21s 358us/sample - loss: 0.0819
Epoch 3/15
60000/60000 [==============================] - 21s 355us/sample - loss: 0.0545
Epoch 4/15
60000/60000 [==============================] - 22s 367us/sample - loss: 0.0374
Epoch 5/15
60000/60000 [==============================] - 22s 367us/sample - loss: 0.0275
Epoch 6/15
60000/60000 [==============================] - 22s 359us/sample - loss: 0.0217
Epoch 7/15
60000/60000 [==============================] - 21s 354us/sample - loss: 0.0175
Epoch 8/15
60000/60000 [==============================] - 21s 351us/sample - loss: 0.0132
Epoch 9/15
60000/60000 [==============================] - 21s 356us/sample - loss: 0.0132
Epoch 10/15
60000/60000 [==============================] - 21s 353us/sample - loss: 0.0107
Epoch 11/15
60000/60000 [==============================] - 22s 361us/sample - loss: 0.0102
Epoch 12/15
60000/60000 [==============================] - 21s 350us/sample - loss: 0.0090
Epoch 13/15
60000/60000 [==============================] - 21s 357us/sample - loss: 0.0090
Epoch 14/15
60000/60000 [==============================] - 22s 363us/sample - loss: 0.0089
Epoch 15/15
60000/60000 [==============================] - 22s 362us/sample - loss: 0.0078
10000/10000 [==============================] - 1s 73us/sample - loss: 0.0911 -
The accuracy of the neural network is 98.19 percent
```