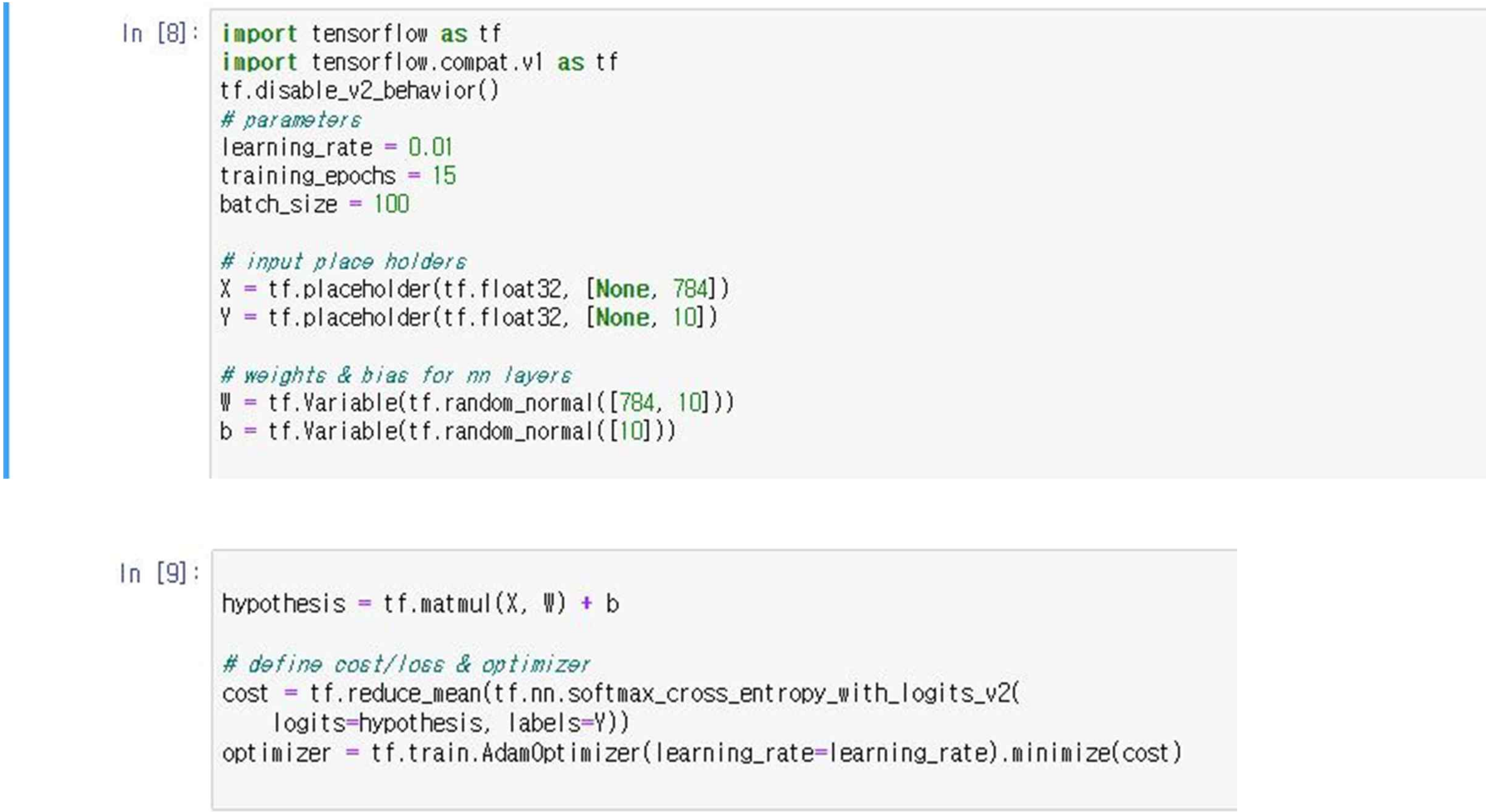
MNIST 손글씨 분류

2016010735 최연석

2017010715 허지혜

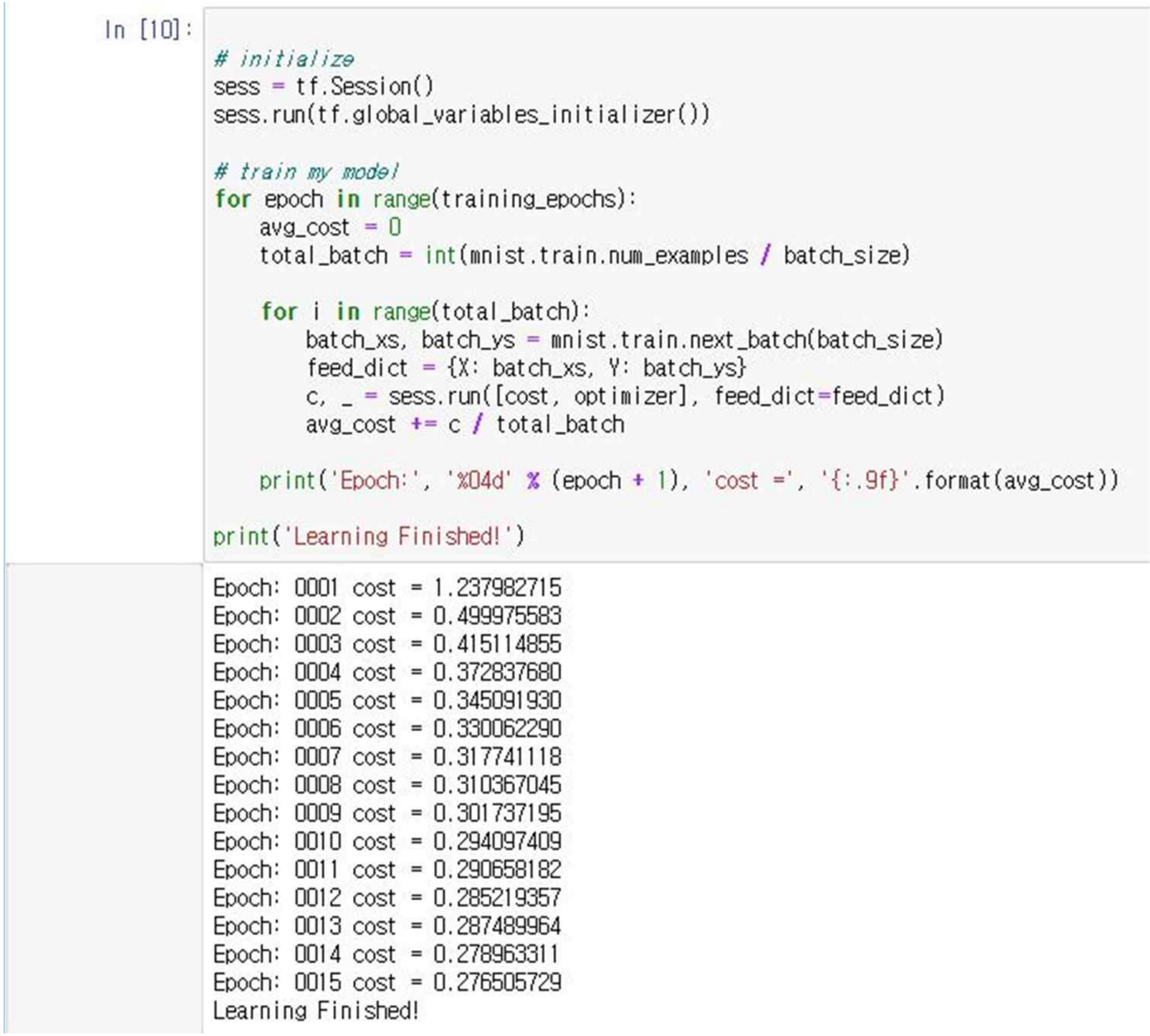


In [8]:

```
import tensorflow as tf
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
# parameters
learning_rate = 0.01
training_epochs = 15
batch_size = 100

# input place holders
X = tf.placeholder(tf.float32, [None, 784])
Y = tf.placeholder(tf.float32, [None, 10])

# weights & bias for nn layers
W = tf.Variable(tf.random_normal([784, 10]))
b = tf.Variable(tf.random_normal([10]))
```

In [9]:

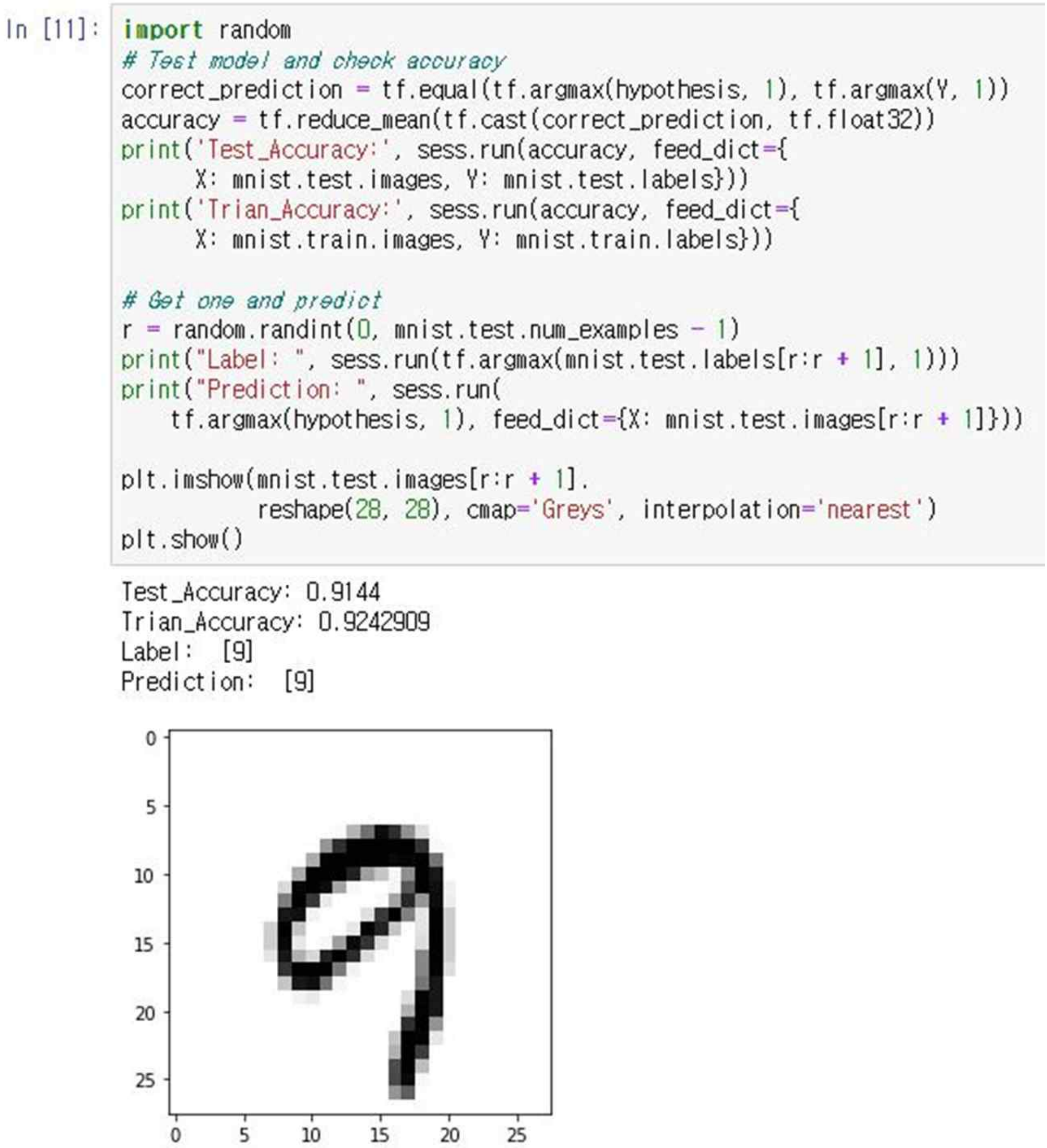
```

hypothesis = tf.matmul(X, W) + b

# define cost/loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(
    logits=hypothesis, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
```



In [10]:

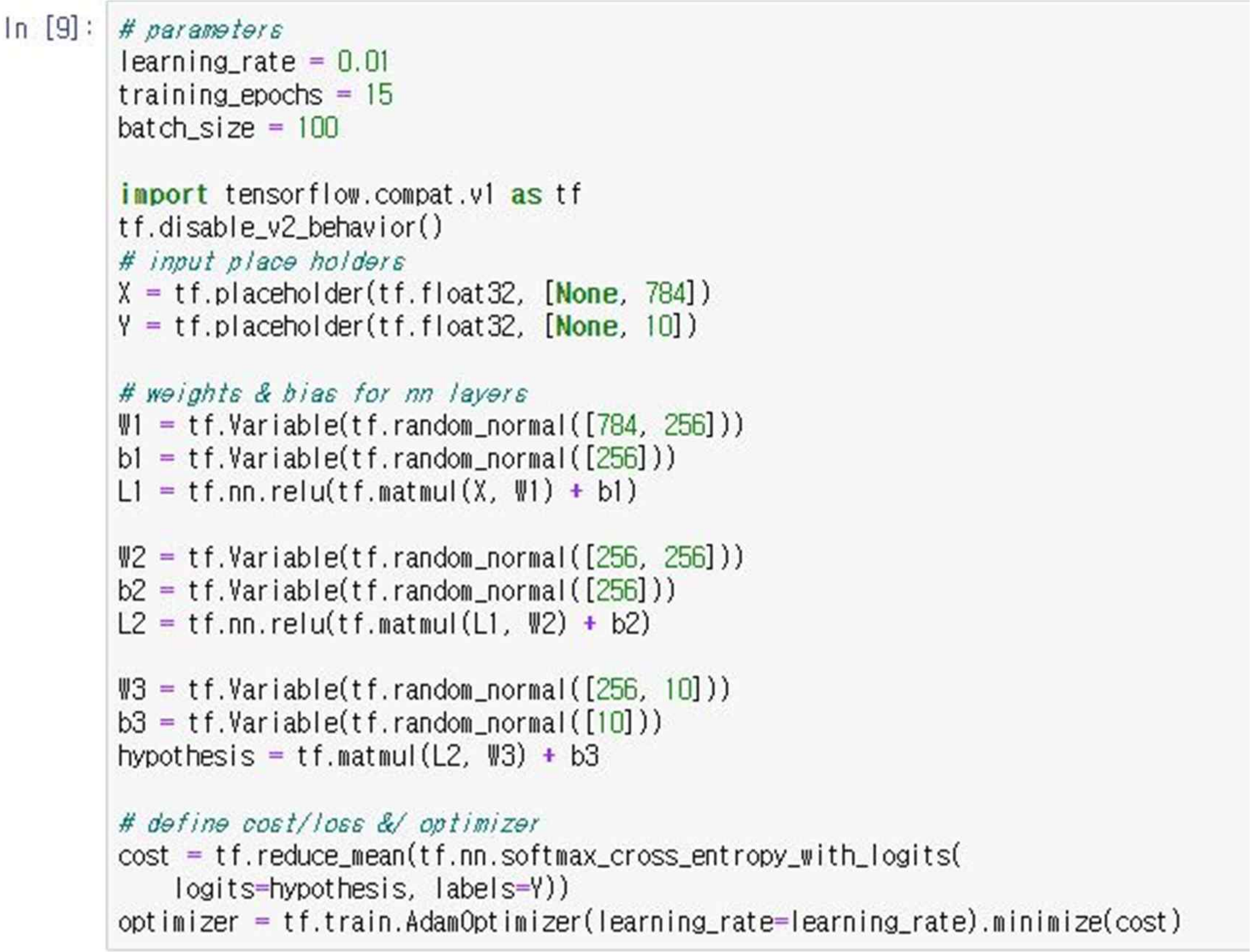
```
# initialize
sess = tf.Session()
sess.run(tf.global_variables_initializer())

# train my model
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        feed_dict = {X: batch_xs, Y: batch_ys}
        c, _ = sess.run([cost, optimizer], feed_dict=feed_dict)
        avg_cost += c / total_batch

    print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))

print('Learning Finished!')
```

```
Epoch: 0001 cost = 1.237982715
Epoch: 0002 cost = 0.499975583
Epoch: 0003 cost = 0.415114855
Epoch: 0004 cost = 0.372837680
Epoch: 0005 cost = 0.345091930
Epoch: 0006 cost = 0.330062290
Epoch: 0007 cost = 0.317741118
Epoch: 0008 cost = 0.310367045
Epoch: 0009 cost = 0.301737195
Epoch: 0010 cost = 0.294097409
Epoch: 0011 cost = 0.290658182
Epoch: 0012 cost = 0.285219357
Epoch: 0013 cost = 0.287489964
Epoch: 0014 cost = 0.278963311
Epoch: 0015 cost = 0.276505729
Learning Finished!
```



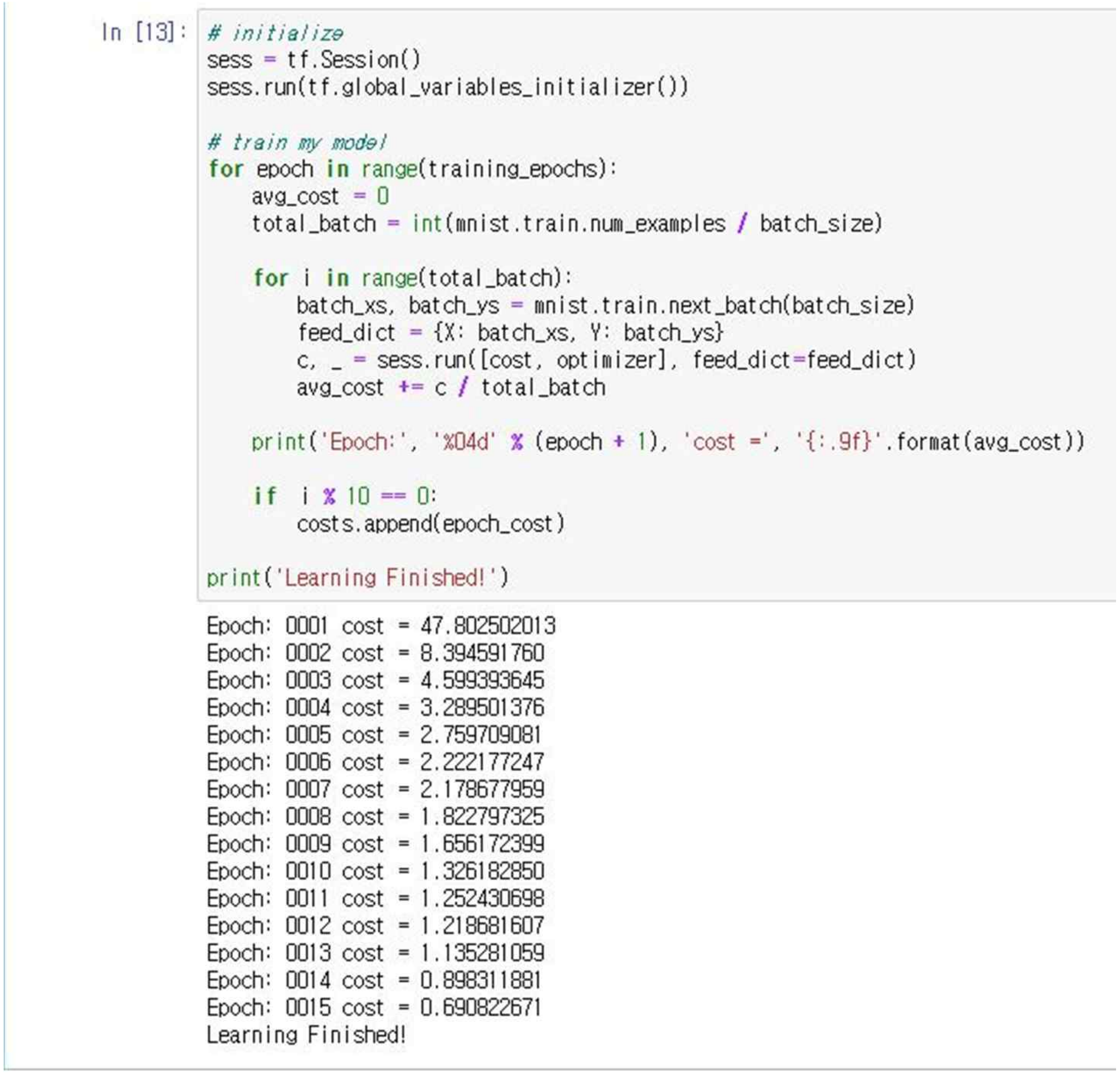
```
In [11]: import random
         # Test model and check accuracy
         correct_prediction = tf.equal(tf.argmax(hypothesis, 1), tf.argmax(Y, 1))
         accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
         print('Test_Accuracy:', sess.run(accuracy, feed_dict={
               X: mnist.test.images, Y: mnist.test.labels}))
         print('Trian_Accuracy:', sess.run(accuracy, feed_dict={
               X: mnist.train.images, Y: mnist.train.labels}))

         # Get one and predict
         r = random.randint(0, mnist.test.num_examples - 1)
         print("Label: ", sess.run(tf.argmax(mnist.test.labels[r:r + 1], 1)))
         print("Prediction: ", sess.run(
             tf.argmax(hypothesis, 1), feed_dict={X: mnist.test.images[r:r + 1]}))

         plt.imshow(mnist.test.images[r:r + 1].
                   reshape(28, 28), cmap='Greys', interpolation='nearest')
         plt.show()
```

```
Test_Accuracy: 0.9144
Trian_Accuracy: 0.9242909
Label:  [9]
Prediction:  [9]
```



In [9]:

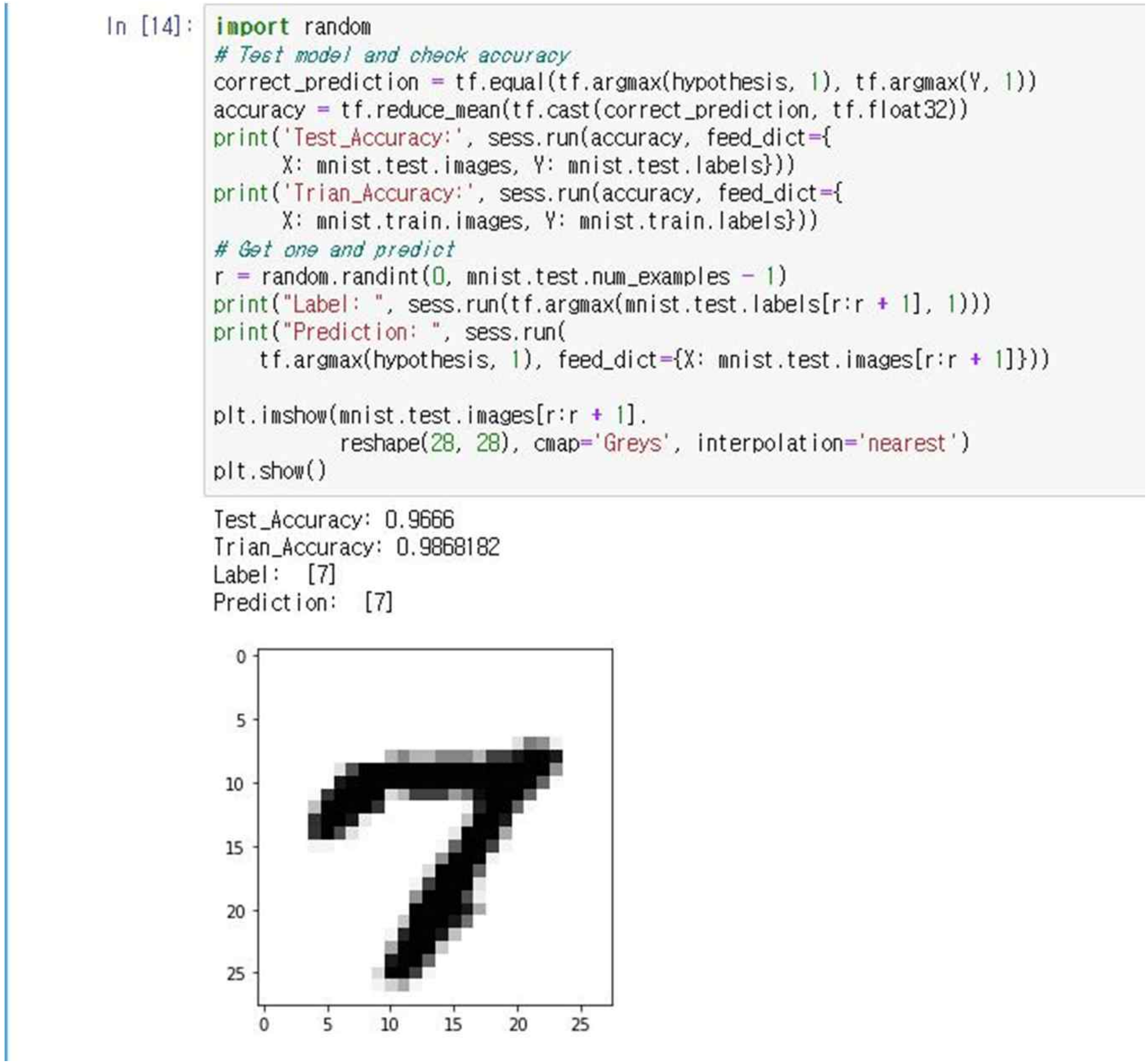
```
# parameters
learning_rate = 0.01
training_epochs = 15
batch_size = 100

import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
# input place holders
X = tf.placeholder(tf.float32, [None, 784])
Y = tf.placeholder(tf.float32, [None, 10])

# weights & bias for nn layers
W1 = tf.Variable(tf.random_normal([784, 256]))
b1 = tf.Variable(tf.random_normal([256]))
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)

W2 = tf.Variable(tf.random_normal([256, 256]))
b2 = tf.Variable(tf.random_normal([256]))
L2 = tf.nn.relu(tf.matmul(L1, W2) + b2)

W3 = tf.Variable(tf.random_normal([256, 10]))
b3 = tf.Variable(tf.random_normal([10]))
hypothesis = tf.matmul(L2, W3) + b3

# define cost/loss &/ optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    logits=hypothesis, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
```



In [13]:

```
# initialize
sess = tf.Session()
sess.run(tf.global_variables_initializer())

# train my model
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        feed_dict = {X: batch_xs, Y: batch_ys}
        c, _ = sess.run([cost, optimizer], feed_dict=feed_dict)
        avg_cost += c / total_batch

    print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))

    if  i % 10 == 0:
        costs.append(epoch_cost)

print('Learning Finished!')
```

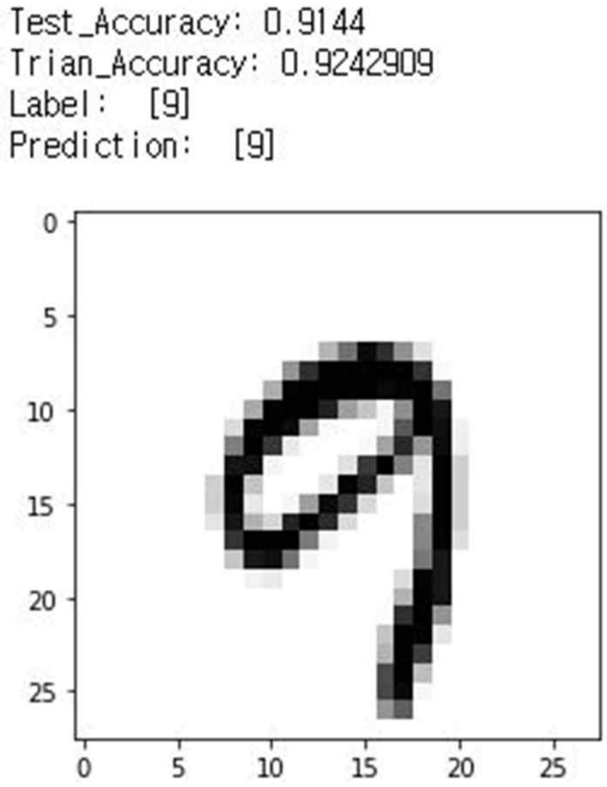
```
Epoch: 0001 cost = 47.802502013
Epoch: 0002 cost = 8.394591760
Epoch: 0003 cost = 4.599393645
Epoch: 0004 cost = 3.289501376
Epoch: 0005 cost = 2.759709081
Epoch: 0006 cost = 2.222177247
Epoch: 0007 cost = 2.178677959
Epoch: 0008 cost = 1.822797325
Epoch: 0009 cost = 1.656172399
Epoch: 0010 cost = 1.326182850
Epoch: 0011 cost = 1.252430698
Epoch: 0012 cost = 1.218681607
Epoch: 0013 cost = 1.135281059
Epoch: 0014 cost = 0.898311881
Epoch: 0015 cost = 0.690822671
Learning Finished!
```



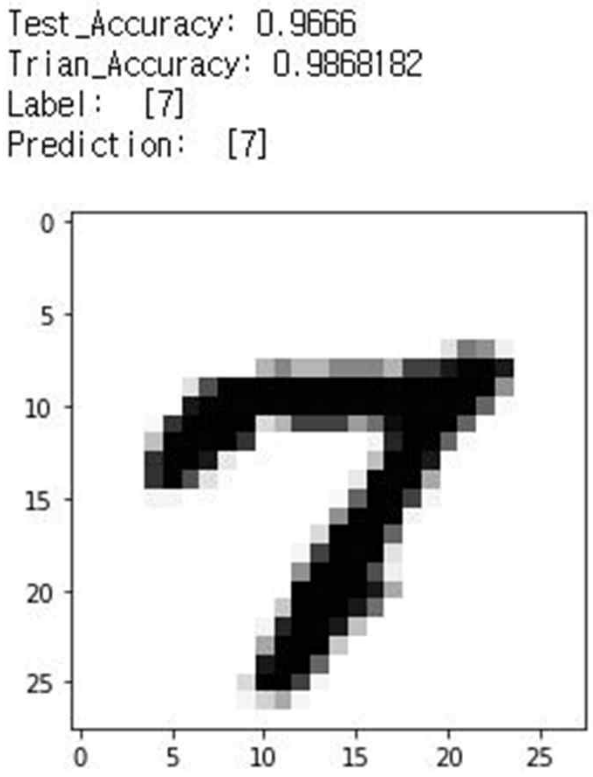
```python
import random
# Test model and check accuracy
correct_prediction = tf.equal(tf.argmax(hypothesis, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print('Test_Accuracy:', sess.run(accuracy, feed_dict={
      X: mnist.test.images, Y: mnist.test.labels}))
print('Trian_Accuracy:', sess.run(accuracy, feed_dict={
      X: mnist.train.images, Y: mnist.train.labels}))
# Get one and predict
r = random.randint(0, mnist.test.num_examples - 1)
print("Label: ", sess.run(tf.argmax(mnist.test.labels[r:r + 1], 1)))
print("Prediction: ", sess.run(
    tf.argmax(hypothesis, 1), feed_dict={X: mnist.test.images[r:r + 1]}))

plt.imshow(mnist.test.images[r:r + 1].
           reshape(28, 28), cmap='Greys', interpolation='nearest')
plt.show()
```

```
Test_Accuracy: 0.9666
Trian_Accuracy: 0.9868182
Label:  [7]
Prediction:  [7]
```

- 비교하기!



```
Test_Accuracy: 0.9144
Trian_Accuracy: 0.9242909
Label:  [9]
Prediction:  [9]
```



```
Test_Accuracy: 0.9666
Trian_Accuracy: 0.9868182
Label:  [7]
Prediction:  [7]
```

끝!