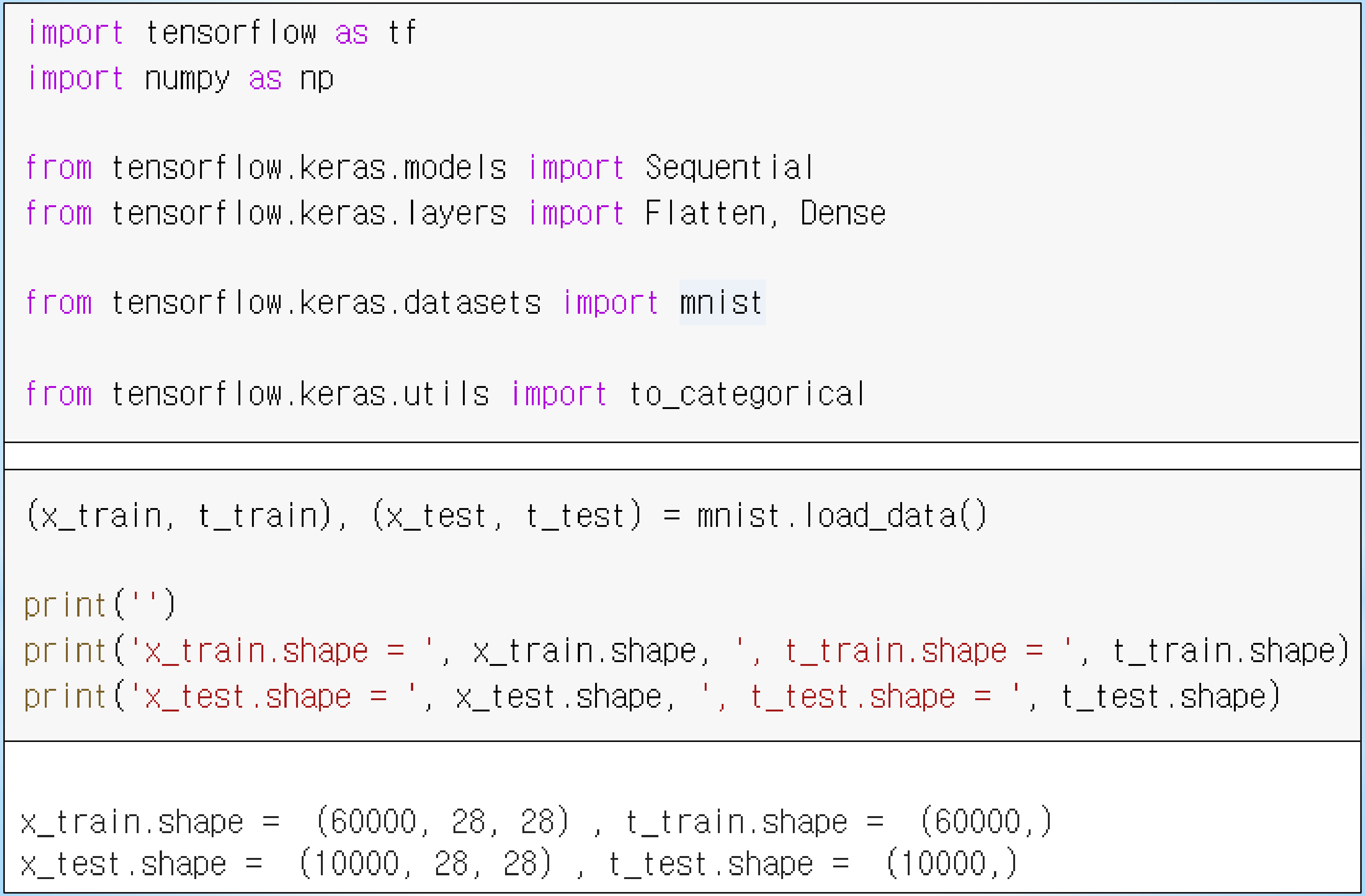
TensorFlow
2.0

- Neural Network (MNIST example) -

박성호 (neowizard2018@gmail.com)

MNIST Example

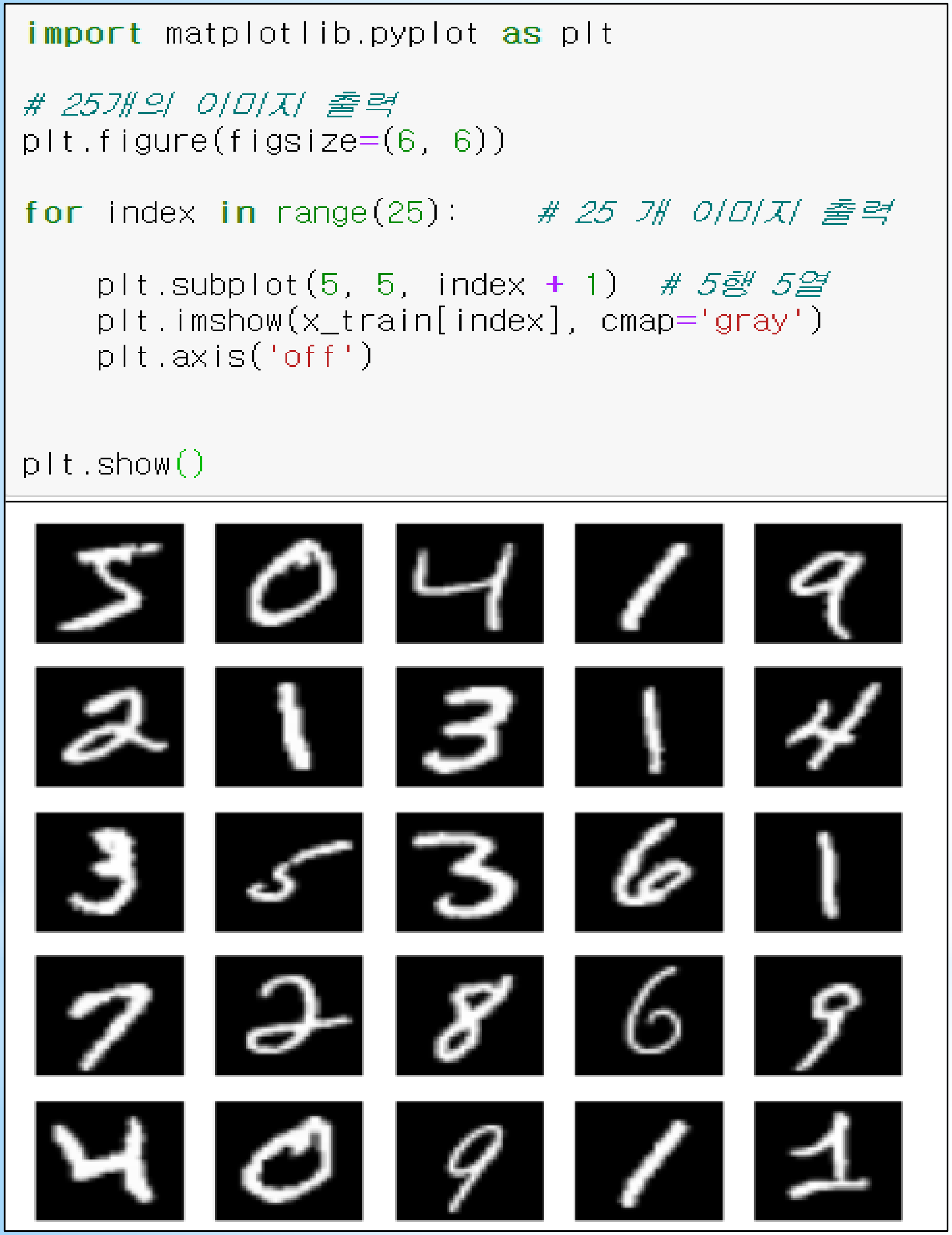
[1] MNIST Data 생성 및 확인



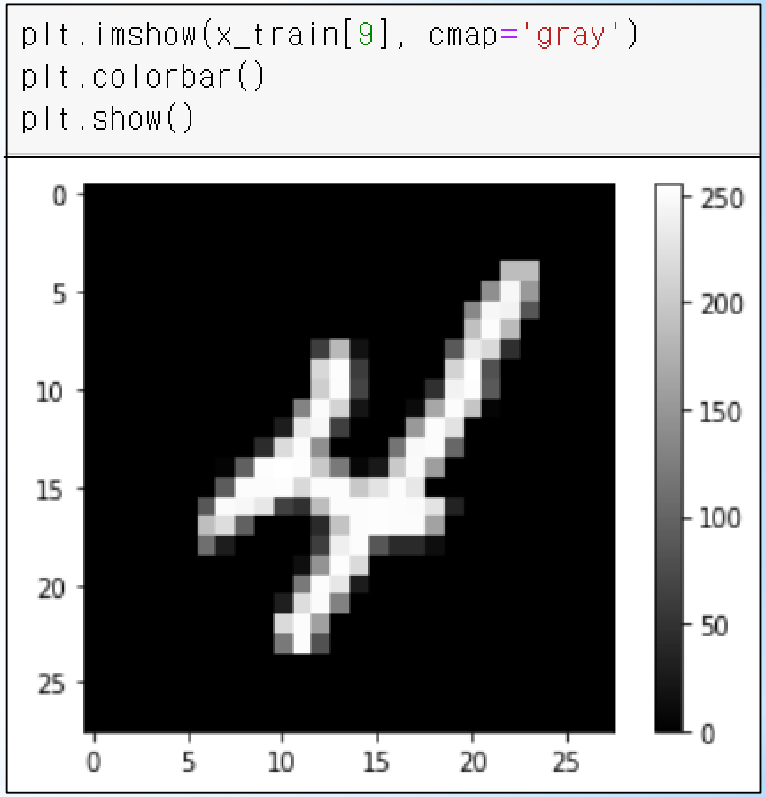
```python
import tensorflow as tf
import numpy as np

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense

from tensorflow.keras.datasets import mnist

from tensorflow.keras.utils import to_categorical
```

```python
(x_train, t_train), (x_test, t_test) = mnist.load_data()

print('')
print('x_train.shape = ', x_train.shape, ', t_train.shape = ', t_train.shape)
print('x_test.shape = ', x_test.shape, ', t_test.shape = ', t_test.shape)
```

```

x_train.shape =  (60000, 28, 28) , t_train.shape =  (60000,)
x_test.shape =  (10000, 28, 28) , t_test.shape =  (10000,)
```

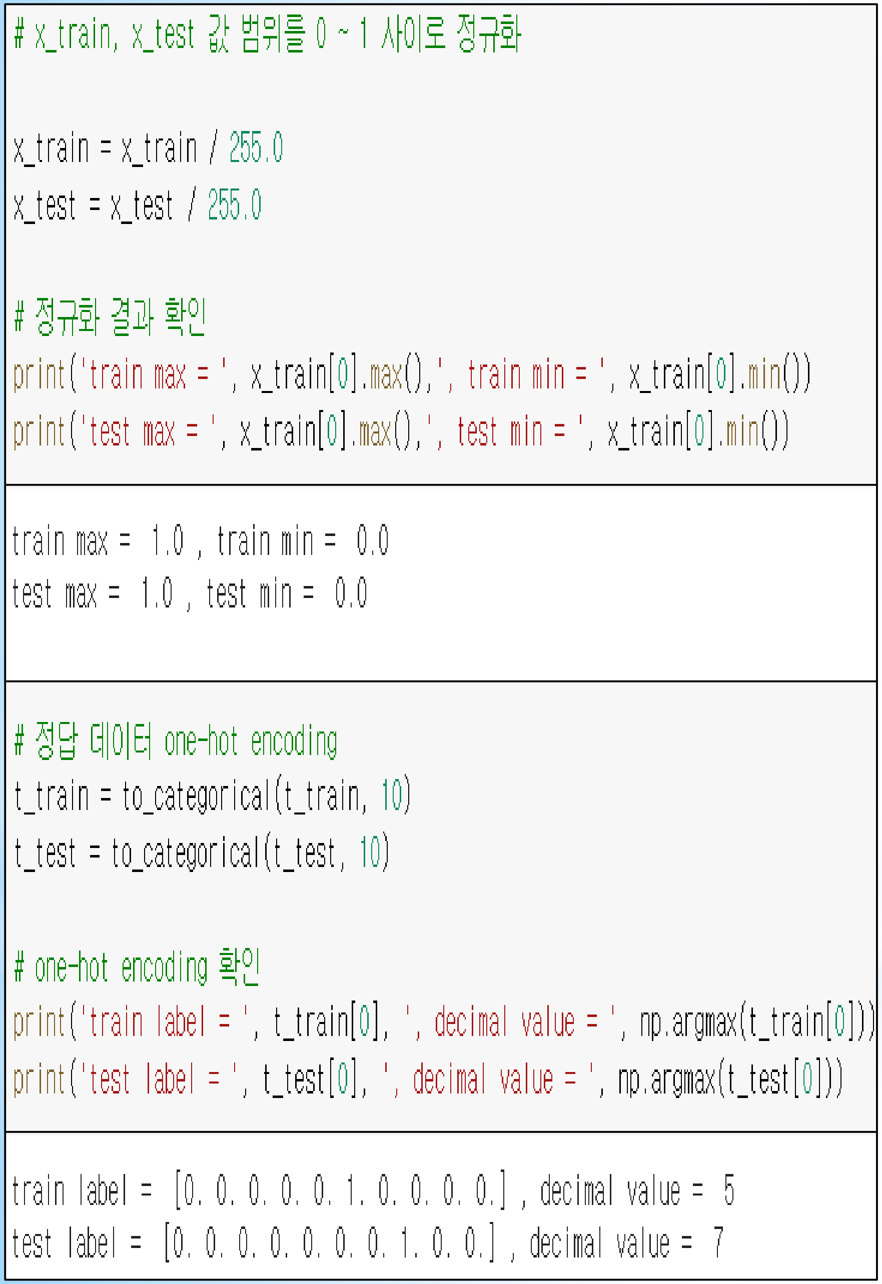


```
import matplotlib.pyplot as plt

# 25개의 이미지 출력
plt.figure(figsize=(6, 6))

for index in range(25):     # 25 개 이미지 출력

    plt.subplot(5, 5, index + 1)  # 5행 5열
    plt.imshow(x_train[index], cmap='gray')
    plt.axis('off')


plt.show()
```



```
plt.imshow(x_train[9], cmap='gray')
plt.colorbar()
plt.show()
```

[2] 데이터 전처리



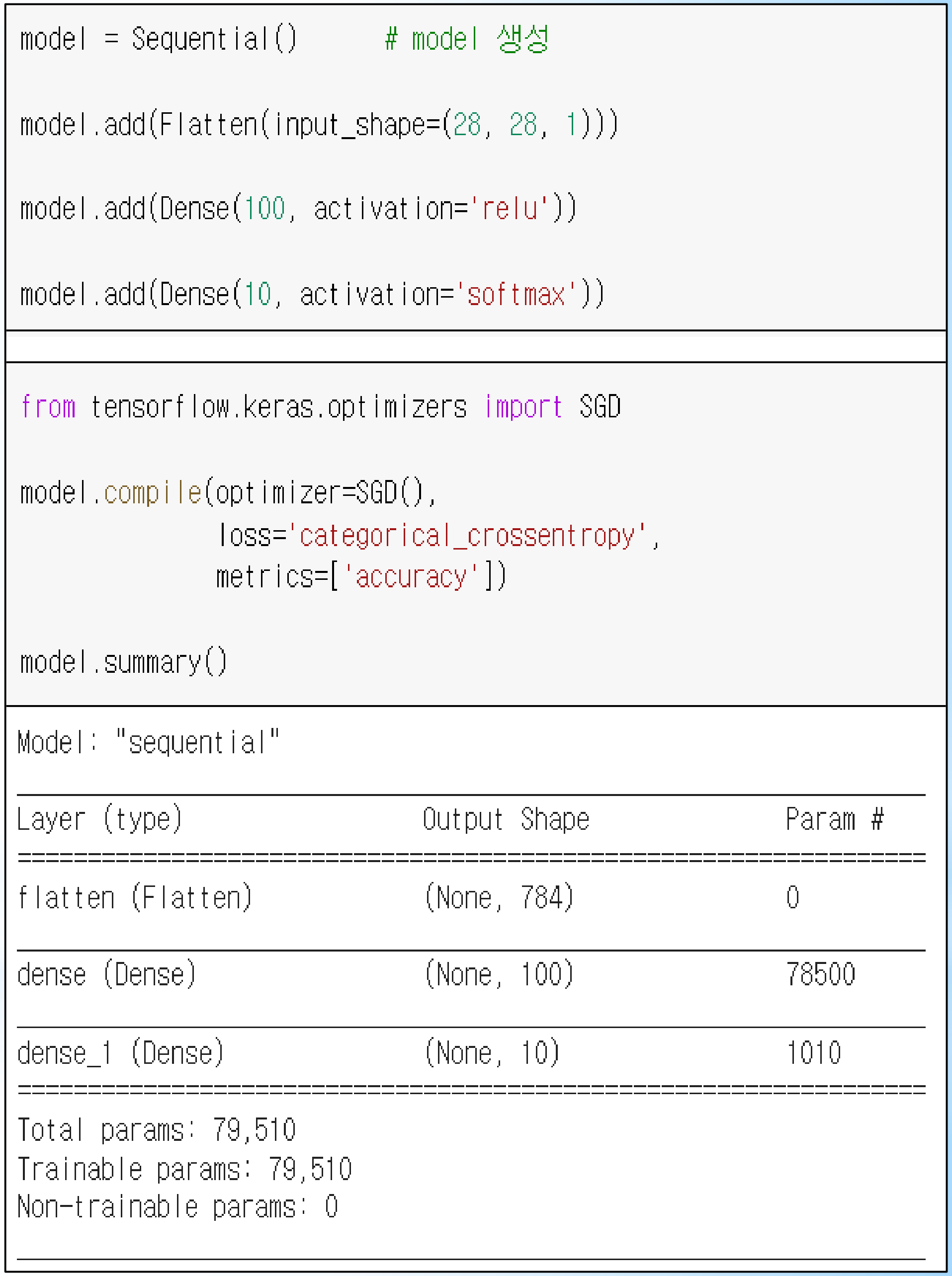
```python
# x_train, x_test 값 범위를 0 ~ 1 사이로 정규화

x_train = x_train / 255.0
x_test = x_test / 255.0

# 정규화 결과 확인
print('train max = ', x_train[0].max(),', train min = ', x_train[0].min())
print('test max = ', x_train[0].max(),', test min = ', x_train[0].min())
```

```
train max =  1.0 , train min =  0.0
test max =  1.0 , test min =  0.0
```

```python
# 정답 데이터 one-hot encoding
t_train = to_categorical(t_train, 10)
t_test = to_categorical(t_test, 10)

# one-hot encoding 확인
print('train label = ', t_train[0], ', decimal value = ', np.argmax(t_train[0]))
print('test label = ', t_test[0], ', decimal value = ', np.argmax(t_test[0]))
```

```
train label =  [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.] , decimal value =  5
test label =  [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.] , decimal value =  7
```

[3] 모델 구축 및 컴파일

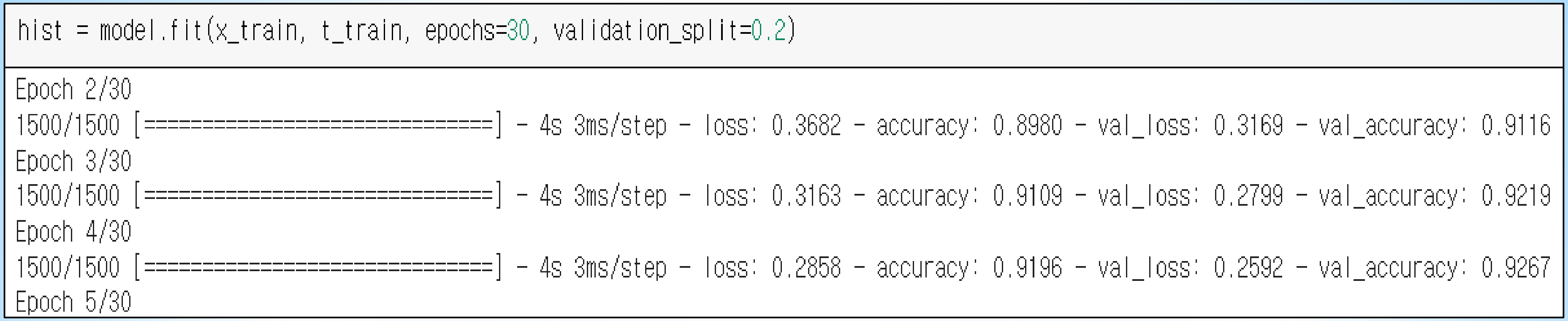


```python
model = Sequential()       # model 생성

model.add(Flatten(input_shape=(28, 28, 1)))

model.add(Dense(100, activation='relu'))

model.add(Dense(10, activation='softmax'))
```

```python
from tensorflow.keras.optimizers import SGD

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

model.summary()
```

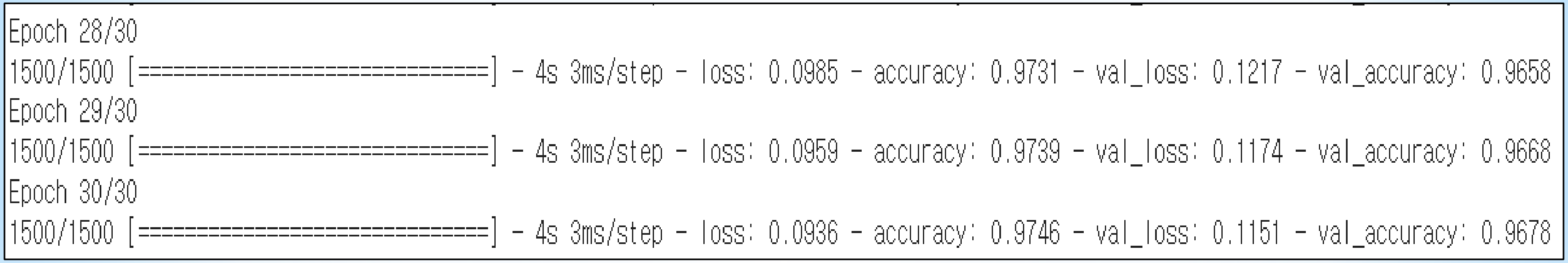
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 100)               78500
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1010
=================================================================
Total params: 79,510
Trainable params: 79,510
Non-trainable params: 0
_________________________________________________________________
```

[4] 모델 학습



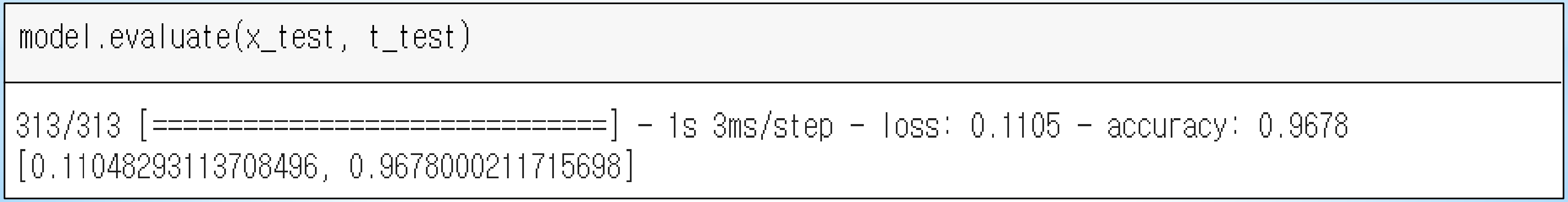
```
hist = model.fit(x_train, t_train, epochs=30, validation_split=0.2)
```

```
Epoch 2/30
1500/1500 [==============================] - 4s 3ms/step - loss: 0.3682 - accuracy: 0.8980 - val_loss: 0.3169 - val_accuracy: 0.9116
Epoch 3/30
1500/1500 [==============================] - 4s 3ms/step - loss: 0.3163 - accuracy: 0.9109 - val_loss: 0.2799 - val_accuracy: 0.9219
Epoch 4/30
1500/1500 [==============================] - 4s 3ms/step - loss: 0.2858 - accuracy: 0.9196 - val_loss: 0.2592 - val_accuracy: 0.9267
Epoch 5/30
```



```
Epoch 28/30
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0985 - accuracy: 0.9731 - val_loss: 0.1217 - val_accuracy: 0.9658
Epoch 29/30
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0959 - accuracy: 0.9739 - val_loss: 0.1174 - val_accuracy: 0.9668
Epoch 30/30
1500/1500 [==============================] - 4s 3ms/step - loss: 0.0936 - accuracy: 0.9746 - val_loss: 0.1151 - val_accuracy: 0.9678
```

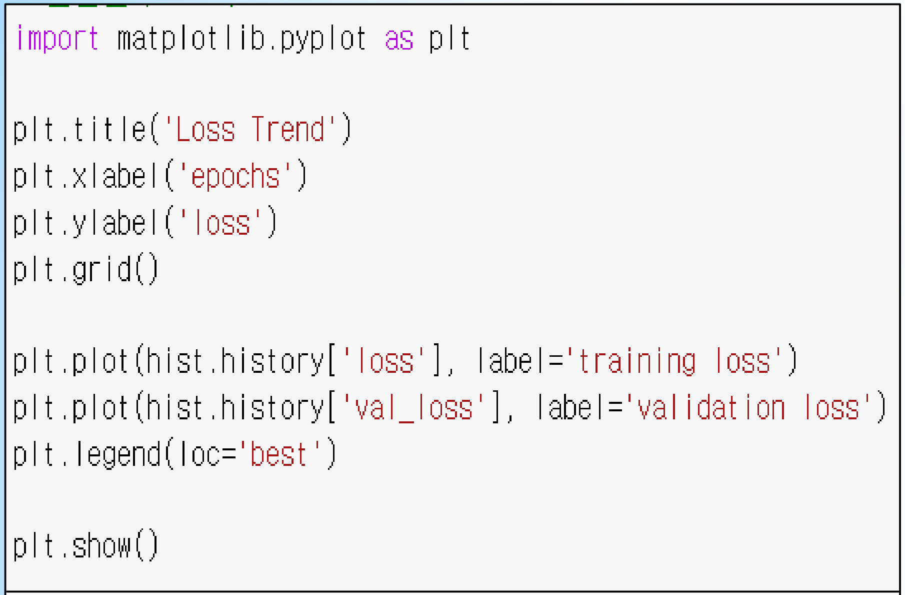
[5] 모델 (정확도) 평가



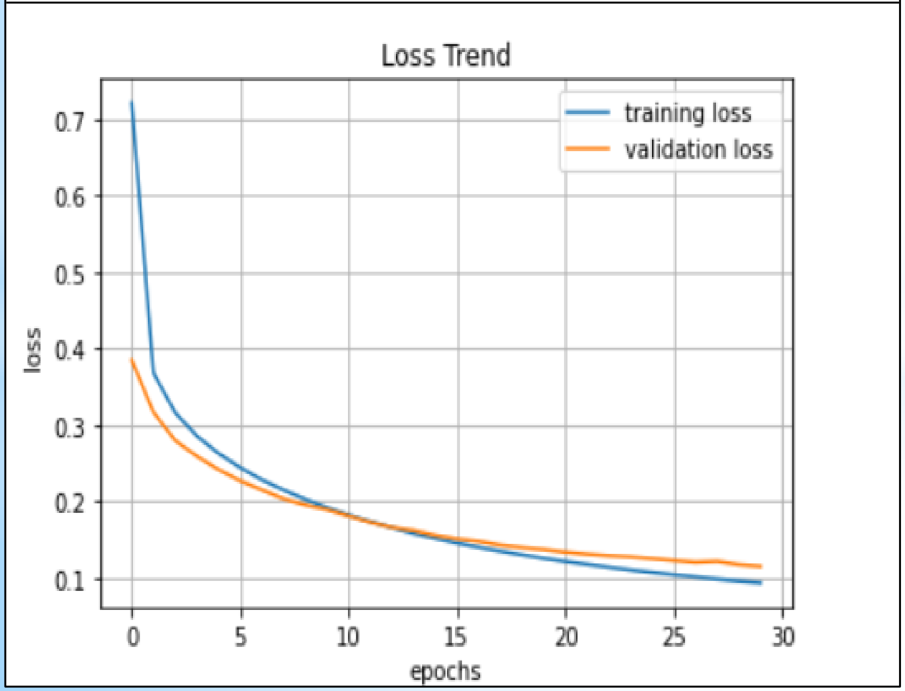
```
model.evaluate(x_test, t_test)
```

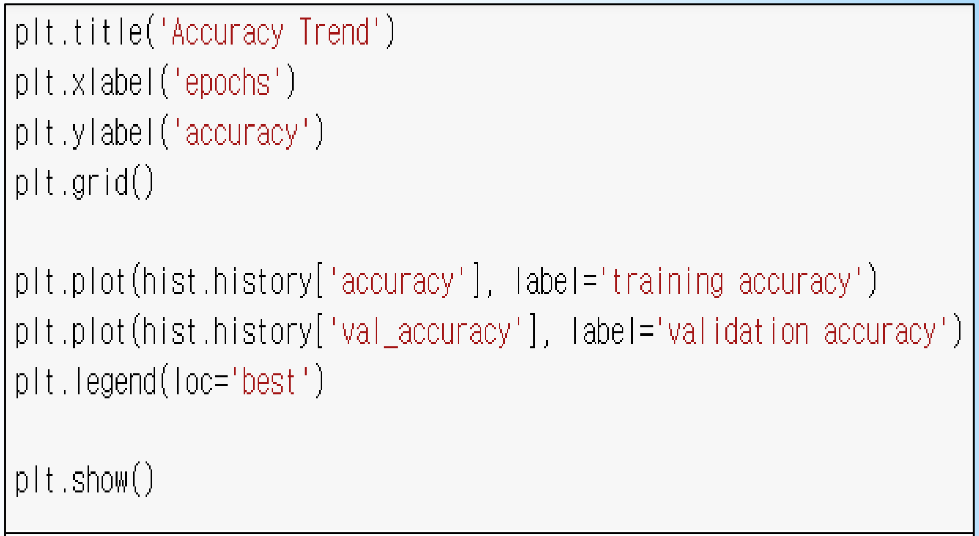
```
313/313 [==============================] - 1s 3ms/step - loss: 0.1105 - accuracy: 0.9678
[0.11048293113708496, 0.9678000211715698]
```

[6] 손실 및 정확도 추세

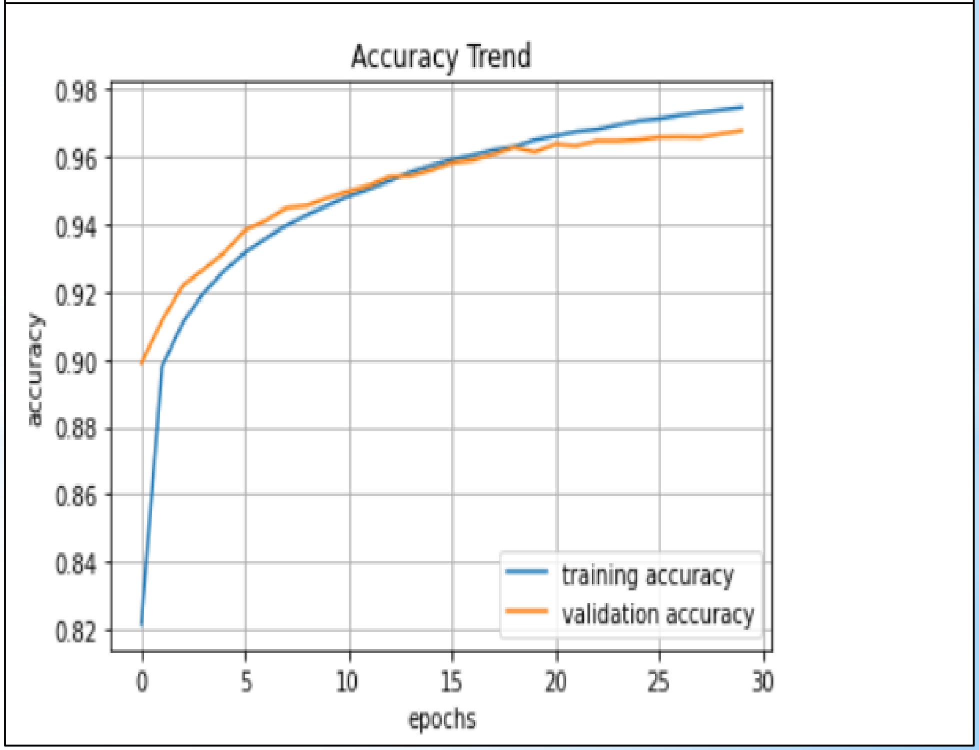


```
import matplotlib.pyplot as plt

plt.title('Loss Trend')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.grid()

plt.plot(hist.history['loss'], label='training loss')
plt.plot(hist.history['val_loss'], label='validation loss')
plt.legend(loc='best')

plt.show()
```

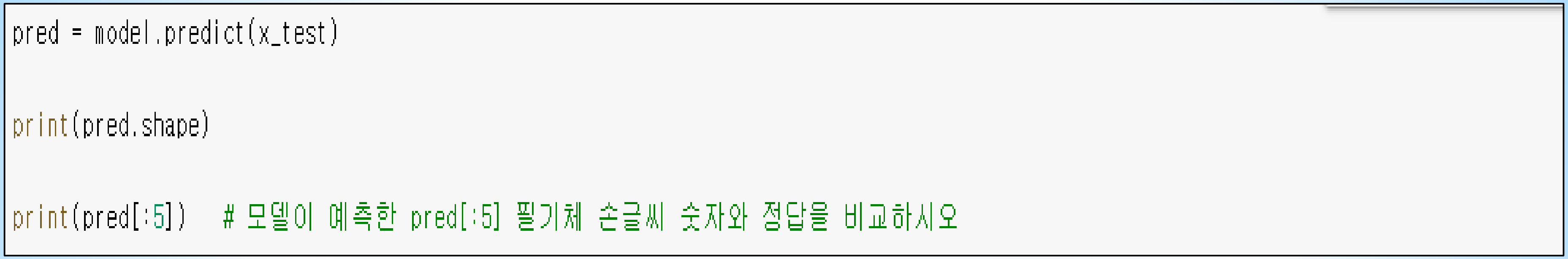




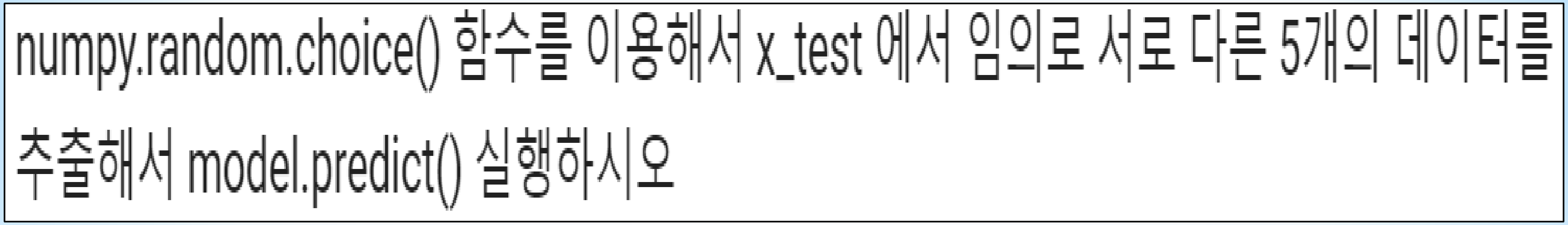
```
plt.title('Accuracy Trend')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.grid()

plt.plot(hist.history['accuracy'], label='training accuracy')
plt.plot(hist.history['val_accuracy'], label='validation accuracy')
plt.legend(loc='best')

plt.show()
```



[7] 예측



```
pred = model.predict(x_test)


print(pred.shape)


print(pred[:5])  # 모델이 예측한 pred[:5] 필기체 손글씨 숫자와 정답을 비교하시오
```



numpy.random.choice() 함수를 이용해서 x_test 에서 임의로 서로 다른 5개의 데이터를 추출해서 model.predict() 실행하시오