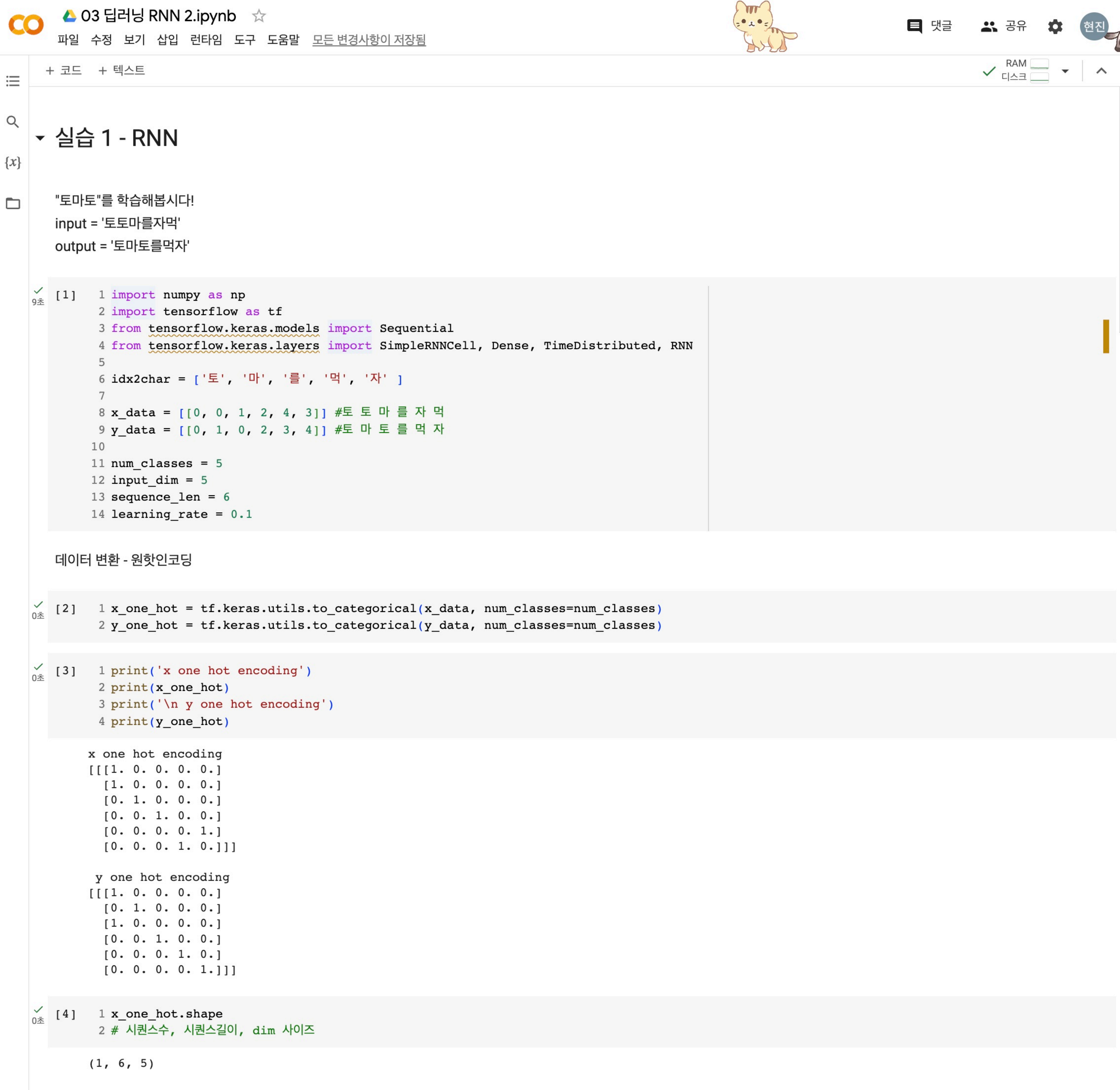


실습 1 - RNN

"토마토"를 학습해봅시다!

input = '토토마를자먹'

output = '토마토를먹자'

[1]
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNNCell, Dense, TimeDistributed, RNN

idx2char = ['토', '마', '를', '먹', '자' ]

x_data = [[0, 0, 1, 2, 4, 3]] #토 토 마 를 자 먹
y_data = [[0, 1, 0, 2, 3, 4]] #토 마 토 를 먹 자

num_classes = 5
input_dim = 5
sequence_len = 6
learning_rate = 0.1
```

데이터 변환 - 원핫인코딩

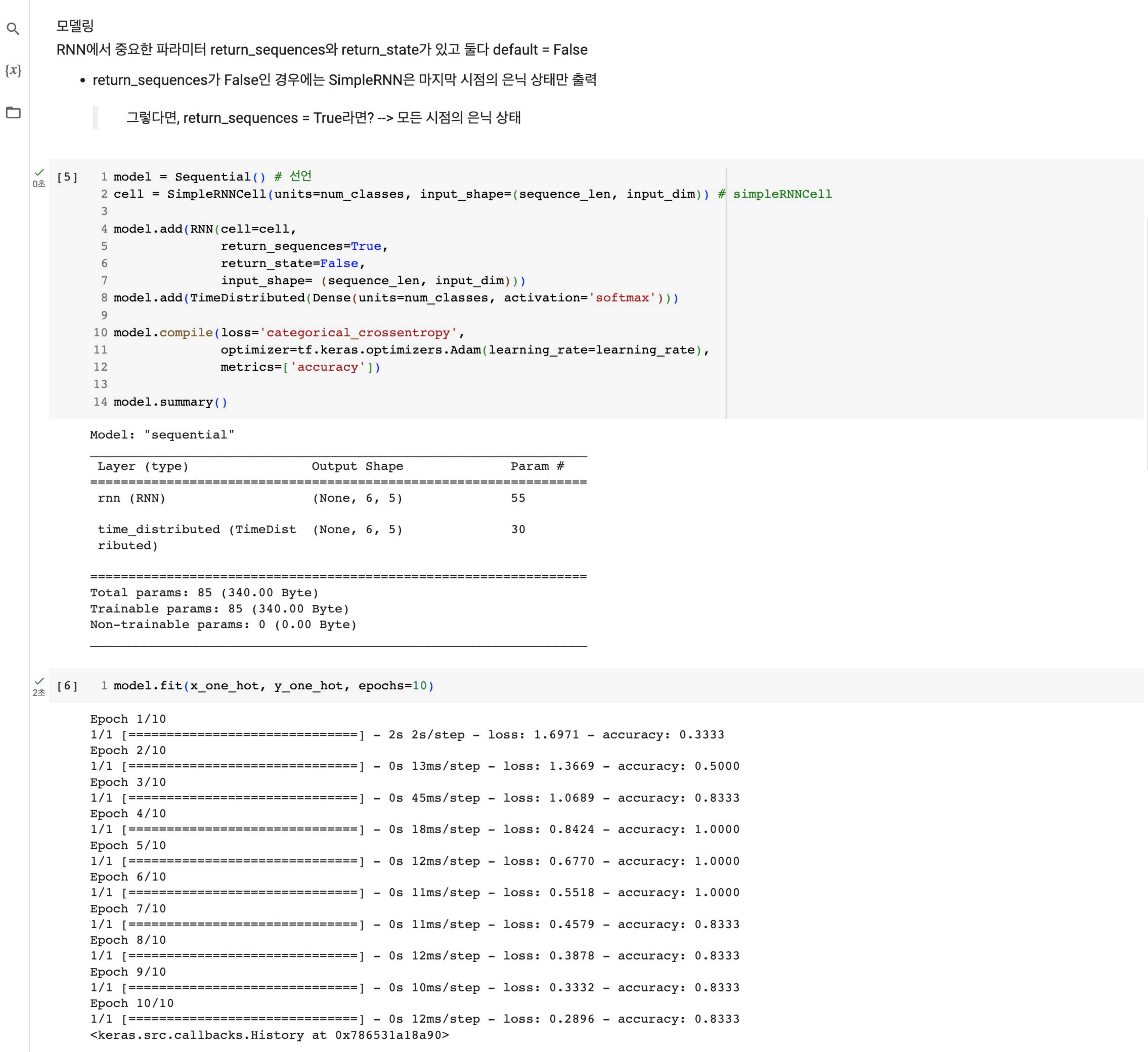
[2]
```
x_one_hot = tf.keras.utils.to_categorical(x_data, num_classes=num_classes)
y_one_hot = tf.keras.utils.to_categorical(y_data, num_classes=num_classes)
```

[3]
```
print('x one hot encoding')
print(x_one_hot)
print('\n y one hot encoding')
print(y_one_hot)
```

```
x one hot encoding
[[[1. 0. 0. 0. 0.]
  [1. 0. 0. 0. 0.]
  [0. 1. 0. 0. 0.]
  [0. 0. 1. 0. 0.]
  [0. 0. 0. 0. 1.]
  [0. 0. 0. 1. 0.]]]

 y one hot encoding
[[[1. 0. 0. 0. 0.]
  [0. 1. 0. 0. 0.]
  [1. 0. 0. 0. 0.]
  [0. 0. 1. 0. 0.]
  [0. 0. 0. 1. 0.]
  [0. 0. 0. 0. 1.]]]
```

[4]
```
x_one_hot.shape
# 시퀀스수, 시퀀스길이, dim 사이즈
```

```
(1, 6, 5)
```



모델링

RNN에서 중요한 파라미터 return_sequences와 return_state가 있고 둘다 default = False

- return_sequences가 False인 경우에는 SimpleRNN은 마지막 시점의 은닉 상태만 출력

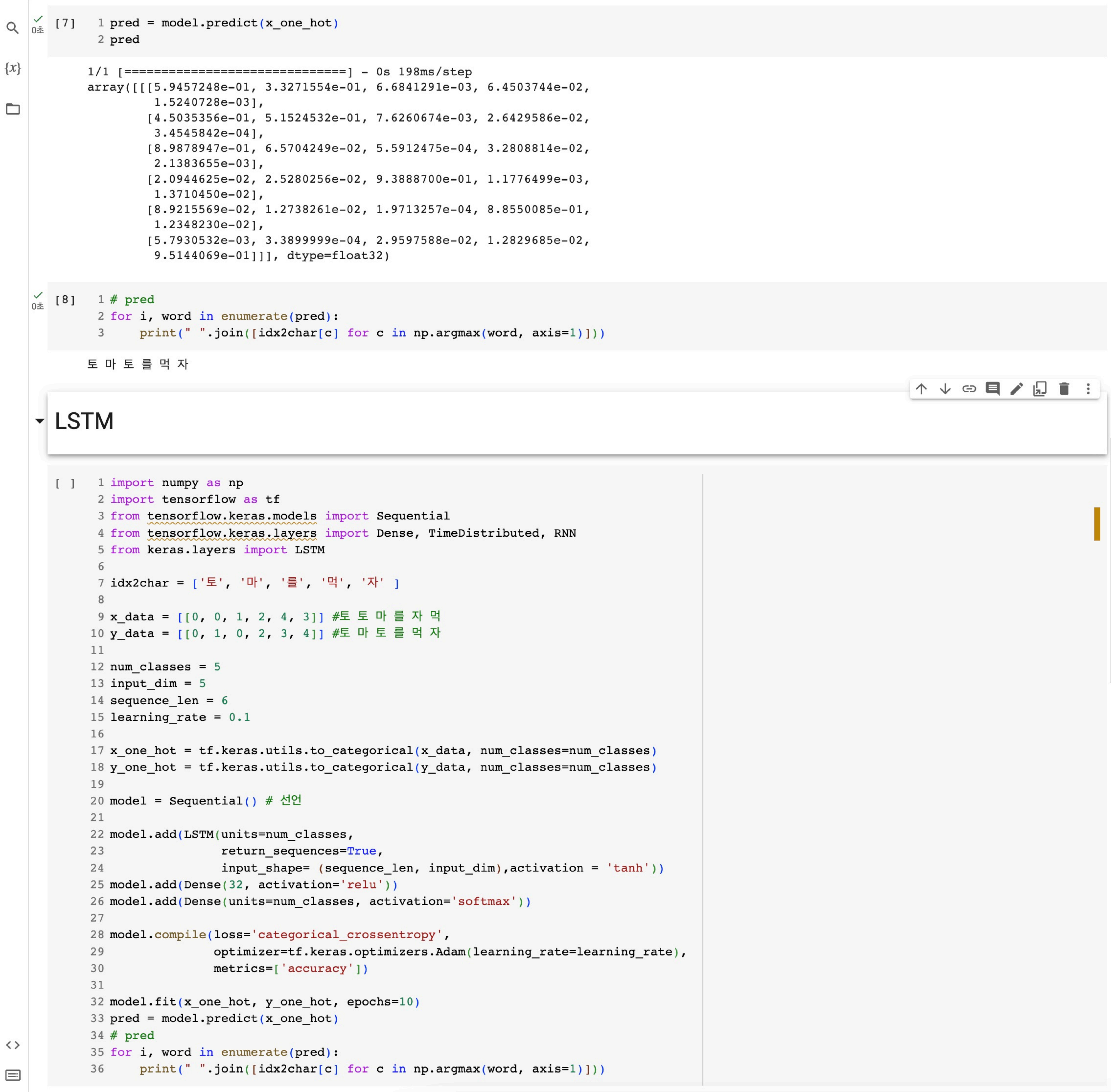
> 그렇다면, return_sequences = True라면? --> 모든 시점의 은닉 상태

```python
model = Sequential() # 선언
cell = SimpleRNNCell(units=num_classes, input_shape=(sequence_len, input_dim)) # simpleRNNCell

model.add(RNN(cell=cell,
              return_sequences=True,
              return_state=False,
              input_shape= (sequence_len, input_dim)))
model.add(TimeDistributed(Dense(units=num_classes, activation='softmax')))

model.compile(loss='categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate),
              metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 rnn (RNN)                   (None, 6, 5)              55        
                                                                 
 time_distributed (TimeDist  (None, 6, 5)              30        
 ributed)                                                        
                                                                 
=================================================================
Total params: 85 (340.00 Byte)
Trainable params: 85 (340.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```python
model.fit(x_one_hot, y_one_hot, epochs=10)
```

```
Epoch 1/10
1/1 [==============================] - 2s 2s/step - loss: 1.6971 - accuracy: 0.3333
Epoch 2/10
1/1 [==============================] - 0s 13ms/step - loss: 1.3669 - accuracy: 0.5000
Epoch 3/10
1/1 [==============================] - 0s 45ms/step - loss: 1.0689 - accuracy: 0.8333
Epoch 4/10
1/1 [==============================] - 0s 18ms/step - loss: 0.8424 - accuracy: 1.0000
Epoch 5/10
1/1 [==============================] - 0s 12ms/step - loss: 0.6770 - accuracy: 1.0000
Epoch 6/10
1/1 [==============================] - 0s 11ms/step - loss: 0.5518 - accuracy: 1.0000
Epoch 7/10
1/1 [==============================] - 0s 11ms/step - loss: 0.4579 - accuracy: 0.8333
Epoch 8/10
1/1 [==============================] - 0s 12ms/step - loss: 0.3878 - accuracy: 0.8333
Epoch 9/10
1/1 [==============================] - 0s 10ms/step - loss: 0.3332 - accuracy: 0.8333
Epoch 10/10
1/1 [==============================] - 0s 12ms/step - loss: 0.2896 - accuracy: 0.8333
<keras.src.callbacks.History at 0x786531a18a90>
```



```python
pred = model.predict(x_one_hot)
pred
```

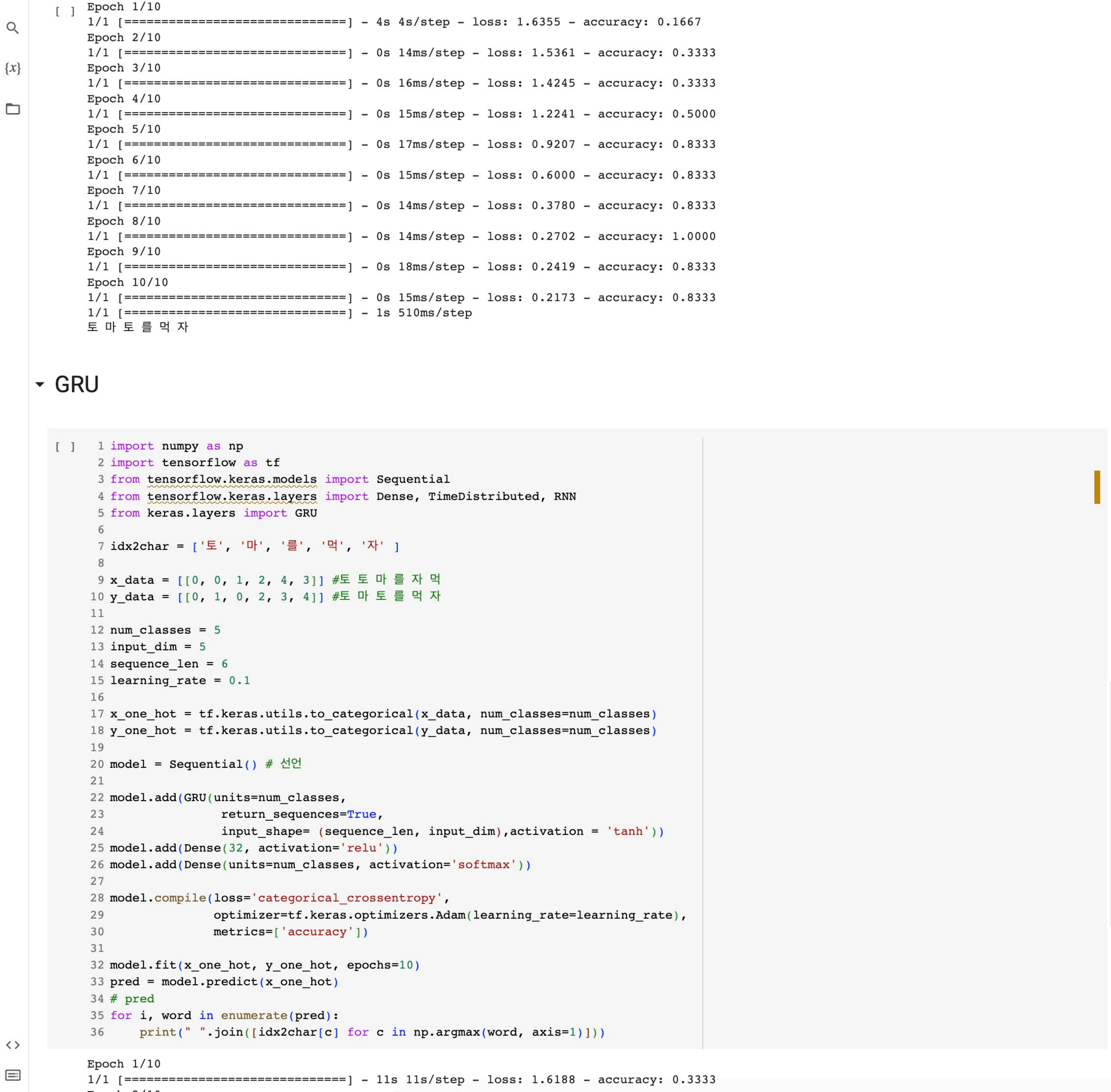
```
1/1 [==============================] - 0s 198ms/step
array([[[5.9457248e-01, 3.3271554e-01, 6.6841291e-03, 6.4503744e-02,
         1.5240728e-03],
        [4.5035356e-01, 5.1524532e-01, 7.6260674e-03, 2.6429586e-02,
         3.4545842e-04],
        [8.9878947e-01, 6.5704249e-02, 5.5912475e-04, 3.2808814e-02,
         2.1383655e-03],
        [2.0944625e-02, 2.5280256e-02, 9.3888700e-01, 1.1776499e-03,
         1.3710450e-02],
        [8.9215569e-02, 1.2738261e-02, 1.9713257e-04, 8.8550085e-01,
         1.2348230e-02],
        [5.7930532e-03, 3.3899999e-04, 2.9597588e-02, 1.2829685e-02,
         9.5144069e-01]]], dtype=float32)
```

```python
# pred
for i, word in enumerate(pred):
    print(" ".join([idx2char[c] for c in np.argmax(word, axis=1)]))
```

```
토 마 토 를 먹 자
```

LSTM

```python
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, TimeDistributed, RNN
from keras.layers import LSTM

idx2char = ['토', '마', '를', '먹', '자' ]

x_data = [[0, 0, 1, 2, 4, 3]] #토 토 마 를 자 먹
y_data = [[0, 1, 0, 2, 3, 4]] #토 마 토 를 먹 자

num_classes = 5
input_dim = 5
sequence_len = 6
learning_rate = 0.1

x_one_hot = tf.keras.utils.to_categorical(x_data, num_classes=num_classes)
y_one_hot = tf.keras.utils.to_categorical(y_data, num_classes=num_classes)

model = Sequential() # 선언

model.add(LSTM(units=num_classes,
               return_sequences=True,
               input_shape= (sequence_len, input_dim),activation = 'tanh'))
model.add(Dense(32, activation='relu'))
model.add(Dense(units=num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate),
              metrics=['accuracy'])

model.fit(x_one_hot, y_one_hot, epochs=10)
pred = model.predict(x_one_hot)
# pred
for i, word in enumerate(pred):
    print(" ".join([idx2char[c] for c in np.argmax(word, axis=1)]))
```



```
Epoch 1/10
1/1 [==============================] - 4s 4s/step - loss: 1.6355 - accuracy: 0.1667
Epoch 2/10
1/1 [==============================] - 0s 14ms/step - loss: 1.5361 - accuracy: 0.3333
Epoch 3/10
1/1 [==============================] - 0s 16ms/step - loss: 1.4245 - accuracy: 0.3333
Epoch 4/10
1/1 [==============================] - 0s 15ms/step - loss: 1.2241 - accuracy: 0.5000
Epoch 5/10
1/1 [==============================] - 0s 17ms/step - loss: 0.9207 - accuracy: 0.8333
Epoch 6/10
1/1 [==============================] - 0s 15ms/step - loss: 0.6000 - accuracy: 0.8333
Epoch 7/10
1/1 [==============================] - 0s 14ms/step - loss: 0.3780 - accuracy: 0.8333
Epoch 8/10
1/1 [==============================] - 0s 14ms/step - loss: 0.2702 - accuracy: 1.0000
Epoch 9/10
1/1 [==============================] - 0s 18ms/step - loss: 0.2419 - accuracy: 0.8333
Epoch 10/10
1/1 [==============================] - 0s 15ms/step - loss: 0.2173 - accuracy: 0.8333
1/1 [==============================] - 1s 510ms/step
토 마 토 를 먹 자
```

GRU

```python
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, TimeDistributed, RNN
from keras.layers import GRU

idx2char = ['토', '마', '를', '먹', '자' ]

x_data = [[0, 0, 1, 2, 4, 3]] #토 토 마 를 자 먹
y_data = [[0, 1, 0, 2, 3, 4]] #토 마 토 를 먹 자

num_classes = 5
input_dim = 5
sequence_len = 6
learning_rate = 0.1

x_one_hot = tf.keras.utils.to_categorical(x_data, num_classes=num_classes)
y_one_hot = tf.keras.utils.to_categorical(y_data, num_classes=num_classes)

model = Sequential() # 선언

model.add(GRU(units=num_classes,
               return_sequences=True,
               input_shape= (sequence_len, input_dim),activation = 'tanh'))
model.add(Dense(32, activation='relu'))
model.add(Dense(units=num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate),
              metrics=['accuracy'])

model.fit(x_one_hot, y_one_hot, epochs=10)
pred = model.predict(x_one_hot)
# pred
for i, word in enumerate(pred):
    print(" ".join([idx2char[c] for c in np.argmax(word, axis=1)]))
```

```
Epoch 1/10
1/1 [==============================] - 11s 11s/step - loss: 1.6188 - accuracy: 0.3333
```