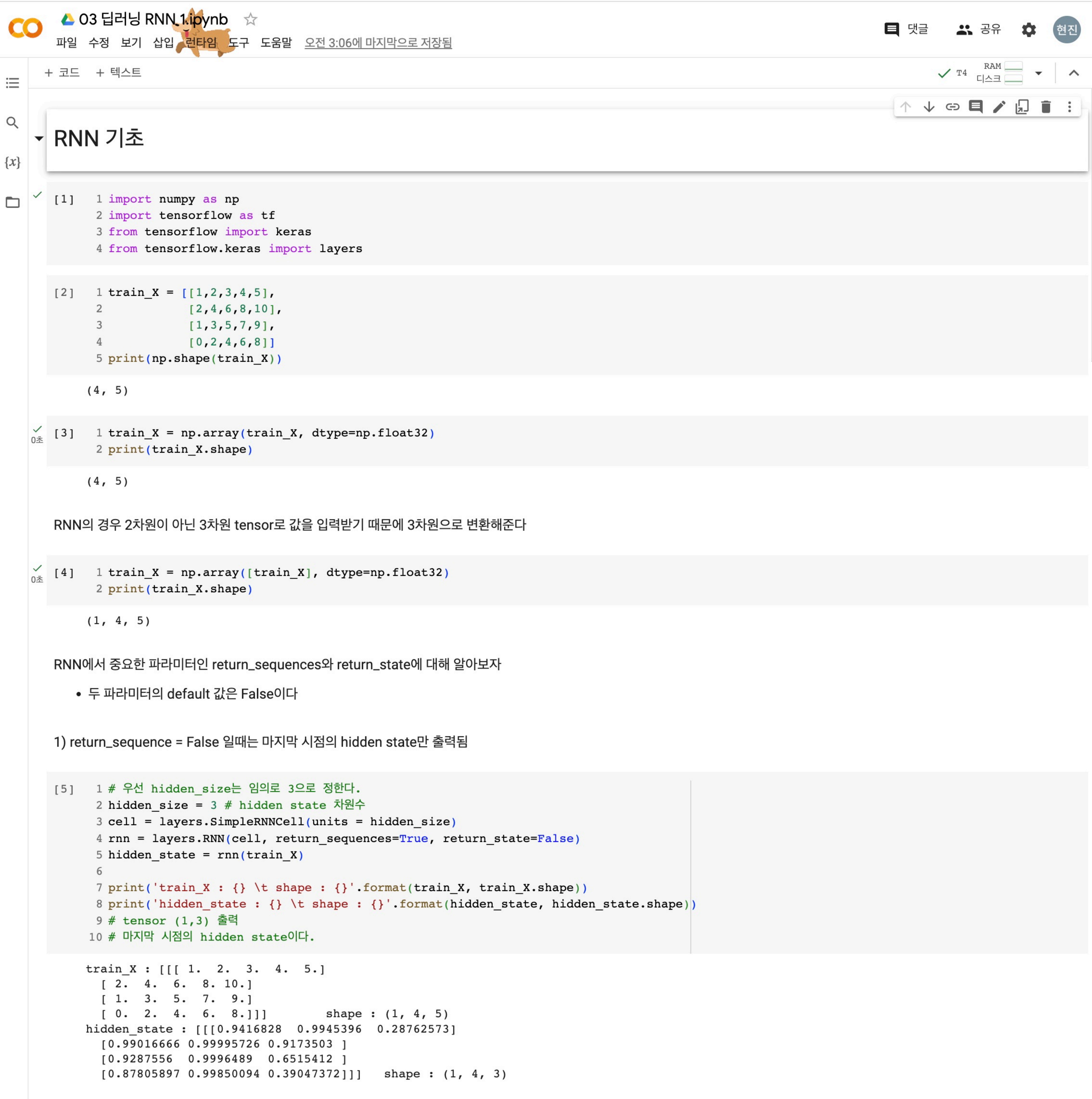

03 딥러닝 RNN 1.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 오전 3:06에 마지막으로 저장됨
댓글
공유
현진
+ 코드 + 텍스트
T4
RAM
디스크

RNN 기초

[1]
```python
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

[2]
```python
train_X = [[1,2,3,4,5],
           [2,4,6,8,10],
           [1,3,5,7,9],
           [0,2,4,6,8]]
print(np.shape(train_X))
```

```
(4, 5)
```

[3]
```python
train_X = np.array(train_X, dtype=np.float32)
print(train_X.shape)
```

```
(4, 5)
```

RNN의 경우 2차원이 아닌 3차원 tensor로 값을 입력받기 때문에 3차원으로 변환해준다

[4]
```python
train_X = np.array([train_X], dtype=np.float32)
print(train_X.shape)
```

```
(1, 4, 5)
```

RNN에서 중요한 파라미터인 return_sequences와 return_state에 대해 알아보자

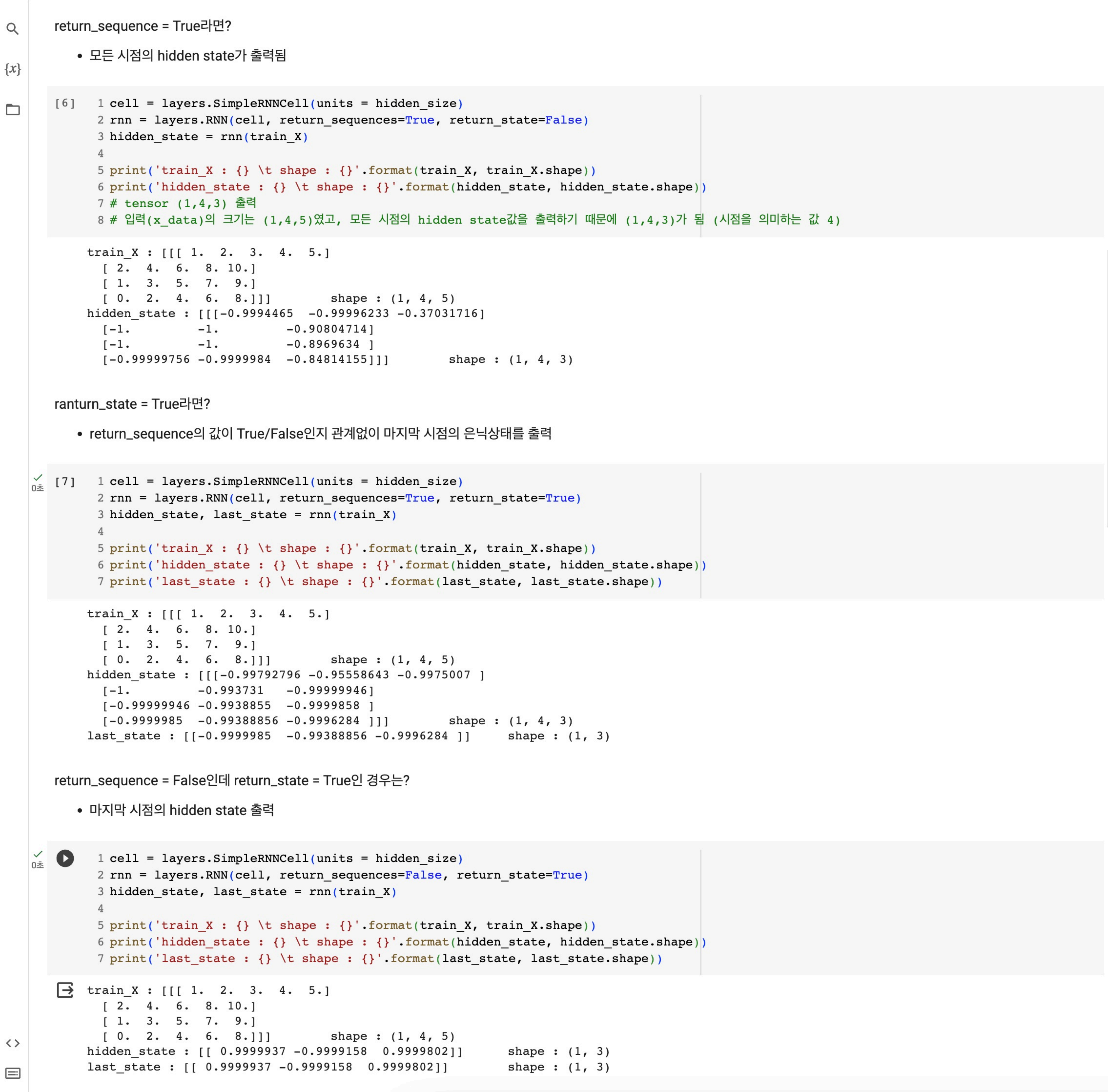
- 두 파라미터의 default 값은 False이다

1) return_sequence = False 일때는 마지막 시점의 hidden state만 출력됨

[5]
```python
# 우선 hidden_size는 임의로 3으로 정한다.
hidden_size = 3 # hidden state 차원수
cell = layers.SimpleRNNCell(units = hidden_size)
rnn = layers.RNN(cell, return_sequences=True, return_state=False)
hidden_state = rnn(train_X)

print('train_X : {} \t shape : {}'.format(train_X, train_X.shape))
print('hidden_state : {} \t shape : {}'.format(hidden_state, hidden_state.shape))
# tensor (1,3) 출력
# 마지막 시점의 hidden state이다.
```

```
train_X : [[[ 1.  2.  3.  4.  5.]
  [ 2.  4.  6.  8. 10.]
  [ 1.  3.  5.  7.  9.]
  [ 0.  2.  4.  6.  8.]]]        shape : (1, 4, 5)
hidden_state : [[[0.9416828  0.9945396  0.28762573]
  [0.99016666 0.99995726 0.9173503 ]
  [0.9287556  0.9996489  0.6515412 ]
  [0.87805897 0.99850094 0.39047372]]]    shape : (1, 4, 3)
```



return_sequence = True라면?

- 모든 시점의 hidden state가 출력됨

```
cell = layers.SimpleRNNCell(units = hidden_size)
rnn = layers.RNN(cell, return_sequences=True, return_state=False)
hidden_state = rnn(train_X)

print('train_X : {} \t shape : {}'.format(train_X, train_X.shape))
print('hidden_state : {} \t shape : {}'.format(hidden_state, hidden_state.shape))
# tensor (1,4,3) 출력
# 입력(x_data)의 크기는 (1,4,5)였고, 모든 시점의 hidden state값을 출력하기 때문에 (1,4,3)가 됨 (시점을 의미하는 값 4)
```

```
train_X : [[[ 1.  2.  3.  4.  5.]
  [ 2.  4.  6.  8. 10.]
  [ 1.  3.  5.  7.  9.]
  [ 0.  2.  4.  6.  8.]]]        shape : (1, 4, 5)
hidden_state : [[[-0.9994465  -0.99996233 -0.37031716]
  [-1.         -1.         -0.90804714]
  [-1.         -1.         -0.8969634 ]
  [-0.99999756 -0.9999984  -0.84814155]]]        shape : (1, 4, 3)
```

ranturn_state = True라면?

- return_sequence의 값이 True/False인지 관계없이 마지막 시점의 은닉상태를 출력

```
cell = layers.SimpleRNNCell(units = hidden_size)
rnn = layers.RNN(cell, return_sequences=True, return_state=True)
hidden_state, last_state = rnn(train_X)

print('train_X : {} \t shape : {}'.format(train_X, train_X.shape))
print('hidden_state : {} \t shape : {}'.format(hidden_state, hidden_state.shape))
print('last_state : {} \t shape : {}'.format(last_state, last_state.shape))
```

```
train_X : [[[ 1.  2.  3.  4.  5.]
  [ 2.  4.  6.  8. 10.]
  [ 1.  3.  5.  7.  9.]
  [ 0.  2.  4.  6.  8.]]]        shape : (1, 4, 5)
hidden_state : [[[-0.99792796 -0.95558643 -0.9975007 ]
  [-1.         -0.993731   -0.99999946]
  [-0.99999946 -0.9938855  -0.9999858 ]
  [-0.9999985  -0.99388856 -0.9996284 ]]]        shape : (1, 4, 3)
last_state : [[-0.9999985  -0.99388856 -0.9996284 ]]      shape : (1, 3)
```

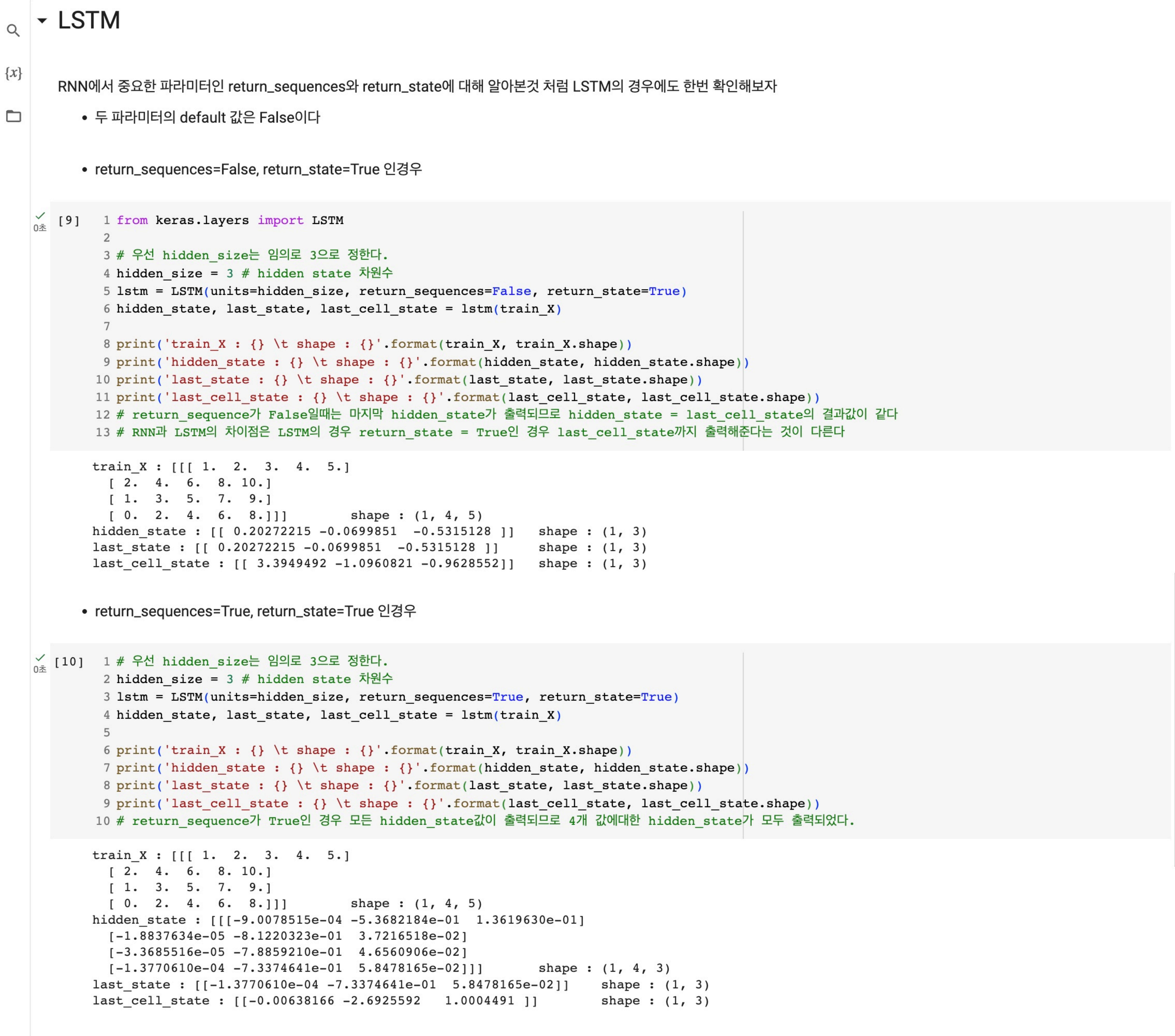
return_sequence = False인데 return_state = True인 경우는?

- 마지막 시점의 hidden state 출력

```
cell = layers.SimpleRNNCell(units = hidden_size)
rnn = layers.RNN(cell, return_sequences=False, return_state=True)
hidden_state, last_state = rnn(train_X)

print('train_X : {} \t shape : {}'.format(train_X, train_X.shape))
print('hidden_state : {} \t shape : {}'.format(hidden_state, hidden_state.shape))
print('last_state : {} \t shape : {}'.format(last_state, last_state.shape))
```

```
train_X : [[[ 1.  2.  3.  4.  5.]
  [ 2.  4.  6.  8. 10.]
  [ 1.  3.  5.  7.  9.]
  [ 0.  2.  4.  6.  8.]]]        shape : (1, 4, 5)
hidden_state : [[ 0.9999937 -0.9999158  0.9999802]]      shape : (1, 3)
last_state : [[ 0.9999937 -0.9999158  0.9999802]]        shape : (1, 3)
```



LSTM

RNN에서 중요한 파라미터인 return_sequences와 return_state에 대해 알아본것 처럼 LSTM의 경우에도 한번 확인해보자

- 두 파라미터의 default 값은 False이다

- return_sequences=False, return_state=True 인경우

```python
from keras.layers import LSTM

# 우선 hidden_size는 임의로 3으로 정한다.
hidden_size = 3 # hidden state 차원수
lstm = LSTM(units=hidden_size, return_sequences=False, return_state=True)
hidden_state, last_state, last_cell_state = lstm(train_X)

print('train_X : {} \t shape : {}'.format(train_X, train_X.shape))
print('hidden_state : {} \t shape : {}'.format(hidden_state, hidden_state.shape))
print('last_state : {} \t shape : {}'.format(last_state, last_state.shape))
print('last_cell_state : {} \t shape : {}'.format(last_cell_state, last_cell_state.shape))
# return_sequence가 False일때는 마지막 hidden_state가 출력되므로 hidden_state = last_cell_state의 결과값이 같다
# RNN과 LSTM의 차이점은 LSTM의 경우 return_state = True인 경우 last_cell_state까지 출력해준다는 것이 다른다
```

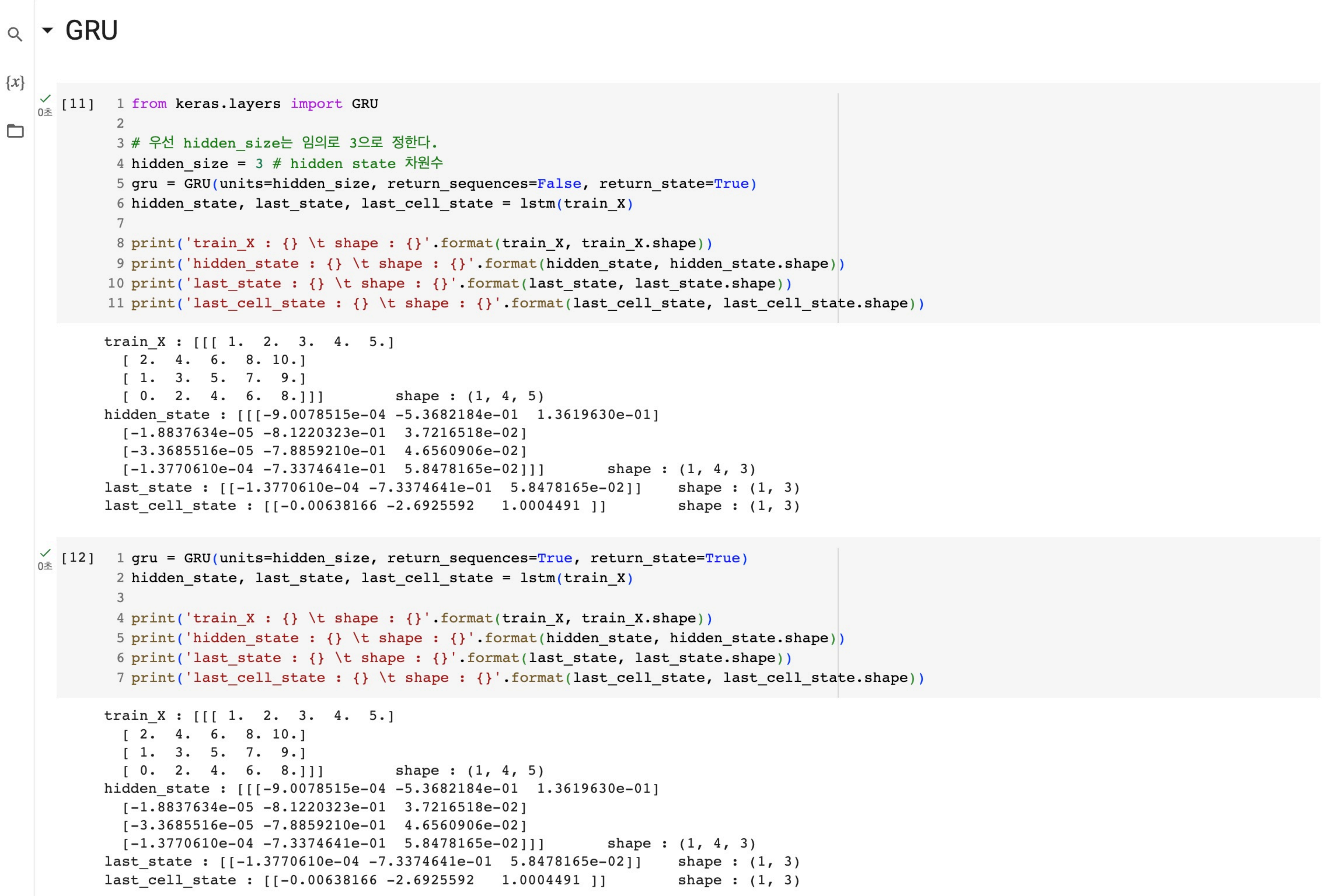
```
train_X : [[[ 1.  2.  3.  4.  5.]
  [ 2.  4.  6.  8. 10.]
  [ 1.  3.  5.  7.  9.]
  [ 0.  2.  4.  6.  8.]]]        shape : (1, 4, 5)
hidden_state : [[ 0.20272215 -0.0699851  -0.5315128 ]]   shape : (1, 3)
last_state : [[ 0.20272215 -0.0699851  -0.5315128 ]]     shape : (1, 3)
last_cell_state : [[ 3.3949492 -1.0960821 -0.9628552]]   shape : (1, 3)
```

- return_sequences=True, return_state=True 인경우

```python
# 우선 hidden_size는 임의로 3으로 정한다.
hidden_size = 3 # hidden state 차원수
lstm = LSTM(units=hidden_size, return_sequences=True, return_state=True)
hidden_state, last_state, last_cell_state = lstm(train_X)

print('train_X : {} \t shape : {}'.format(train_X, train_X.shape))
print('hidden_state : {} \t shape : {}'.format(hidden_state, hidden_state.shape))
print('last_state : {} \t shape : {}'.format(last_state, last_state.shape))
print('last_cell_state : {} \t shape : {}'.format(last_cell_state, last_cell_state.shape))
# return_sequence가 True인 경우 모든 hidden_state값이 출력되므로 4개 값에대한 hidden_state가 모두 출력되었다.
```

```
train_X : [[[ 1.  2.  3.  4.  5.]
  [ 2.  4.  6.  8. 10.]
  [ 1.  3.  5.  7.  9.]
  [ 0.  2.  4.  6.  8.]]]        shape : (1, 4, 5)
hidden_state : [[[-9.0078515e-04 -5.3682184e-01  1.3619630e-01]
  [-1.8837634e-05 -8.1220323e-01  3.7216518e-02]
  [-3.3685516e-05 -7.8859210e-01  4.6560906e-02]
  [-1.3770610e-04 -7.3374641e-01  5.8478165e-02]]]       shape : (1, 4, 3)
last_state : [[-1.3770610e-04 -7.3374641e-01  5.8478165e-02]]    shape : (1, 3)
last_cell_state : [[-0.00638166 -2.6925592   1.0004491 ]]        shape : (1, 3)
```



GRU

```python
from keras.layers import GRU

# 우선 hidden_size는 임의로 3으로 정한다.
hidden_size = 3 # hidden state 차원수
gru = GRU(units=hidden_size, return_sequences=False, return_state=True)
hidden_state, last_state, last_cell_state = lstm(train_X)

print('train_X : {} \t shape : {}'.format(train_X, train_X.shape))
print('hidden_state : {} \t shape : {}'.format(hidden_state, hidden_state.shape))
print('last_state : {} \t shape : {}'.format(last_state, last_state.shape))
print('last_cell_state : {} \t shape : {}'.format(last_cell_state, last_cell_state.shape))
```

```
train_X : [[[ 1.  2.  3.  4.  5.]
  [ 2.  4.  6.  8. 10.]
  [ 1.  3.  5.  7.  9.]
  [ 0.  2.  4.  6.  8.]]]        shape : (1, 4, 5)
hidden_state : [[[-9.0078515e-04 -5.3682184e-01  1.3619630e-01]
  [-1.8837634e-05 -8.1220323e-01  3.7216518e-02]
  [-3.3685516e-05 -7.8859210e-01  4.6560906e-02]
  [-1.3770610e-04 -7.3374641e-01  5.8478165e-02]]]       shape : (1, 4, 3)
last_state : [[-1.3770610e-04 -7.3374641e-01  5.8478165e-02]]    shape : (1, 3)
last_cell_state : [[-0.00638166 -2.6925592   1.0004491 ]]        shape : (1, 3)
```

```python
gru = GRU(units=hidden_size, return_sequences=True, return_state=True)
hidden_state, last_state, last_cell_state = lstm(train_X)

print('train_X : {} \t shape : {}'.format(train_X, train_X.shape))
print('hidden_state : {} \t shape : {}'.format(hidden_state, hidden_state.shape))
print('last_state : {} \t shape : {}'.format(last_state, last_state.shape))
print('last_cell_state : {} \t shape : {}'.format(last_cell_state, last_cell_state.shape))
```

```
train_X : [[[ 1.  2.  3.  4.  5.]
  [ 2.  4.  6.  8. 10.]
  [ 1.  3.  5.  7.  9.]
  [ 0.  2.  4.  6.  8.]]]        shape : (1, 4, 5)
hidden_state : [[[-9.0078515e-04 -5.3682184e-01  1.3619630e-01]
  [-1.8837634e-05 -8.1220323e-01  3.7216518e-02]
  [-3.3685516e-05 -7.8859210e-01  4.6560906e-02]
  [-1.3770610e-04 -7.3374641e-01  5.8478165e-02]]]       shape : (1, 4, 3)
last_state : [[-1.3770610e-04 -7.3374641e-01  5.8478165e-02]]    shape : (1, 3)
last_cell_state : [[-0.00638166 -2.6925592   1.0004491 ]]        shape : (1, 3)
```