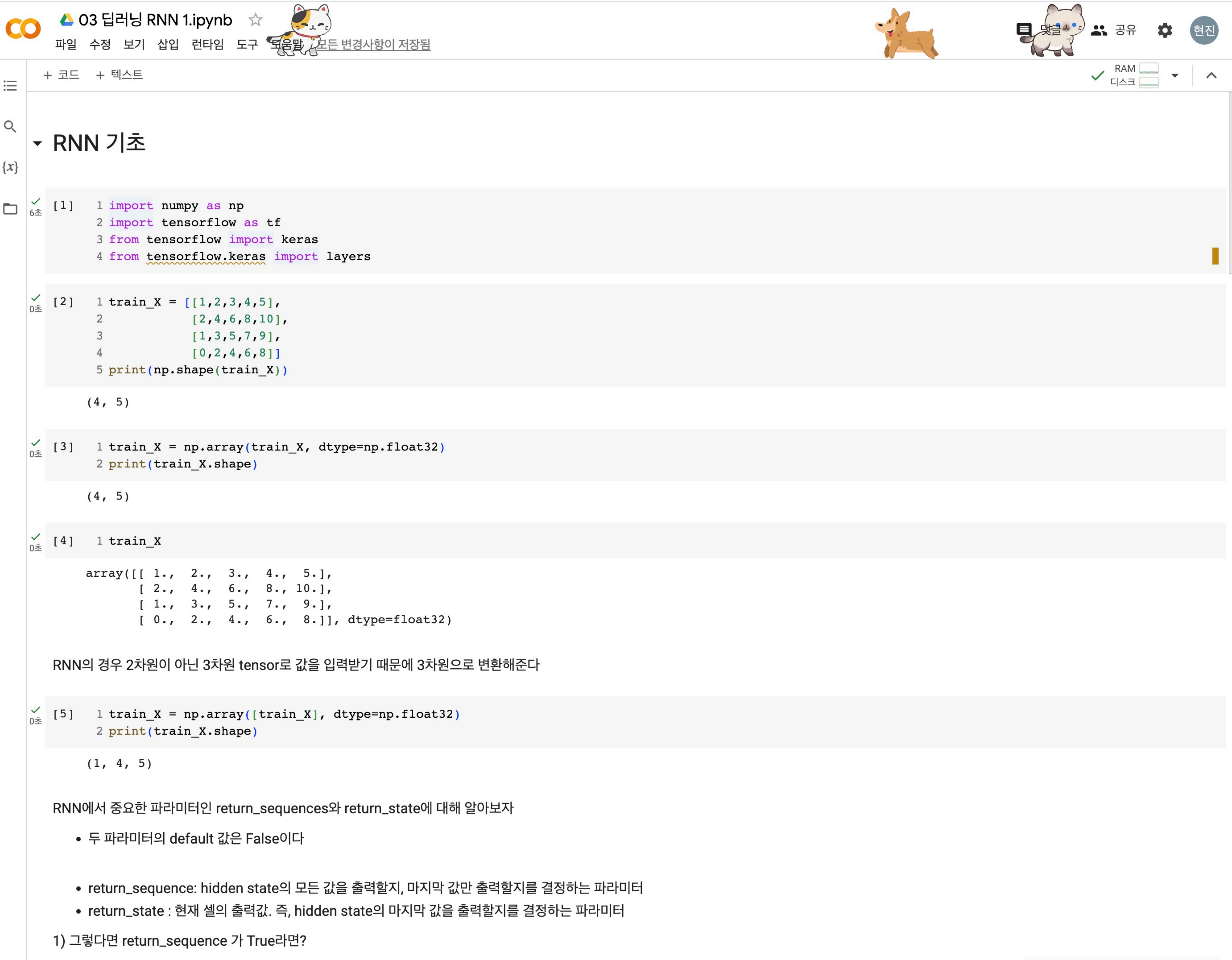

03 딥러닝 RNN 1.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유 현진
+ 코드 + 텍스트
RAM
디스크

RNN 기초

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

[2]
```
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]
```
train_X = np.array(train_X, dtype=np.float32)
print(train_X.shape)
```

```
(4, 5)
```

[4]
```
train_X
```

```
array([[ 1.,  2.,  3.,  4.,  5.],
       [ 2.,  4.,  6.,  8., 10.],
       [ 1.,  3.,  5.,  7.,  9.],
       [ 0.,  2.,  4.,  6.,  8.]], dtype=float32)
```

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

[5]
```
train_X = np.array([train_X], dtype=np.float32)
print(train_X.shape)
```

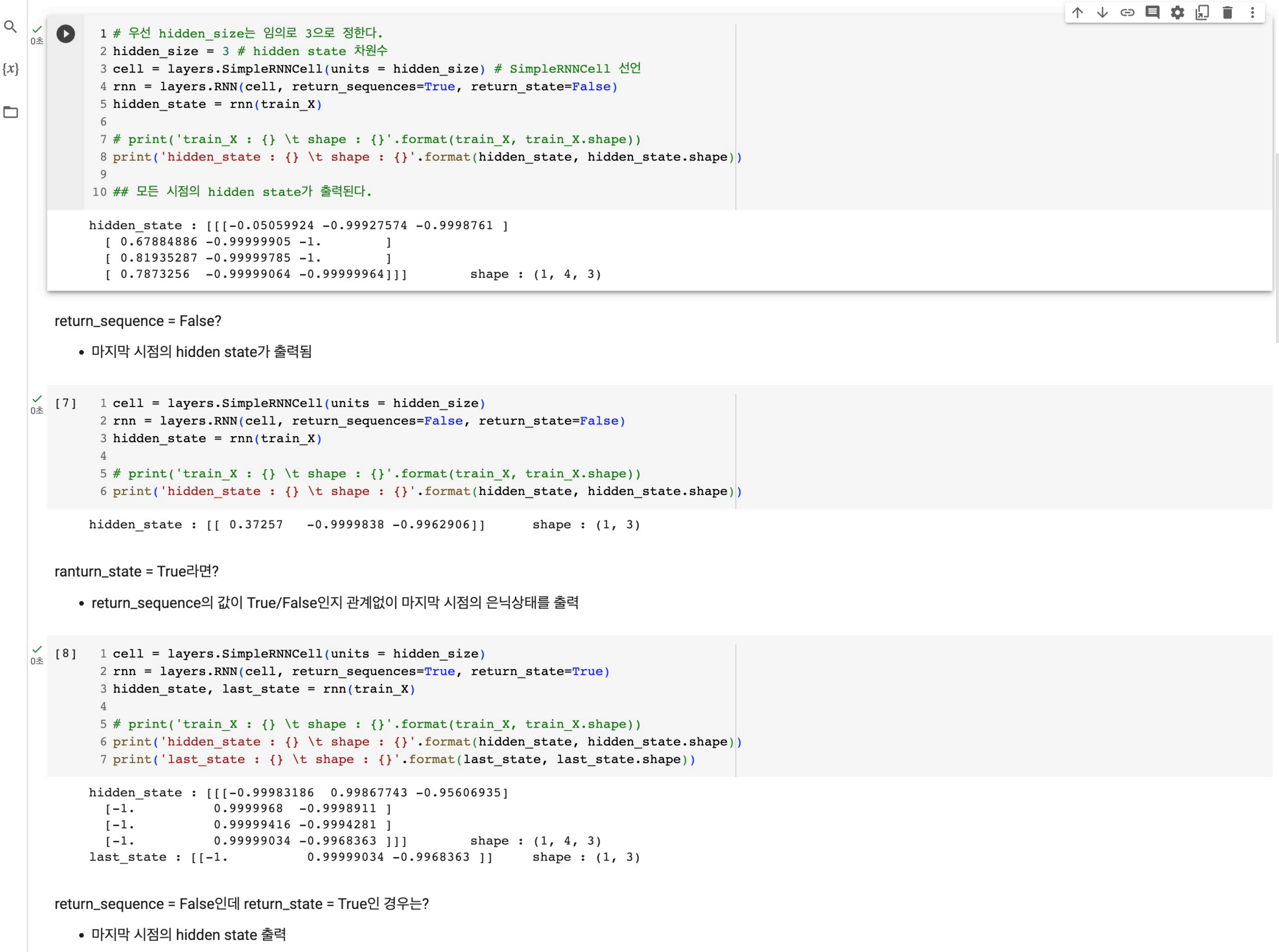
```
(1, 4, 5)
```

RNN에서 중요한 파라미터인 return_sequences와 return_state에 대해 알아보자

- 두 파라미터의 default 값은 False이다

- return_sequence: hidden state의 모든 값을 출력할지, 마지막 값만 출력할지를 결정하는 파라미터
- return_state : 현재 셀의 출력값. 즉, hidden state의 마지막 값을 출력할지를 결정하는 파라미터

1) 그렇다면 return_sequence 가 True라면?



```
# 우선 hidden_size는 임의로 3으로 정한다.
hidden_size = 3 # hidden state 차원수
cell = layers.SimpleRNNCell(units = hidden_size) # SimpleRNNCell 선언
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))

## 모든 시점의 hidden state가 출력된다.
```

```
hidden_state : [[[-0.05059924 -0.99927574 -0.9998761 ]
  [ 0.67884886 -0.99999905 -1.        ]
  [ 0.81935287 -0.99999785 -1.        ]
  [ 0.7873256  -0.99999064 -0.99999964]]]        shape : (1, 4, 3)
```

return_sequence = False?

- 마지막 시점의 hidden state가 출력됨

```
cell = layers.SimpleRNNCell(units = hidden_size)
rnn = layers.RNN(cell, return_sequences=False, 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))
```

```
hidden_state : [[ 0.37257   -0.9999838 -0.9962906]]      shape : (1, 3)
```

ranturn_state = True라면?

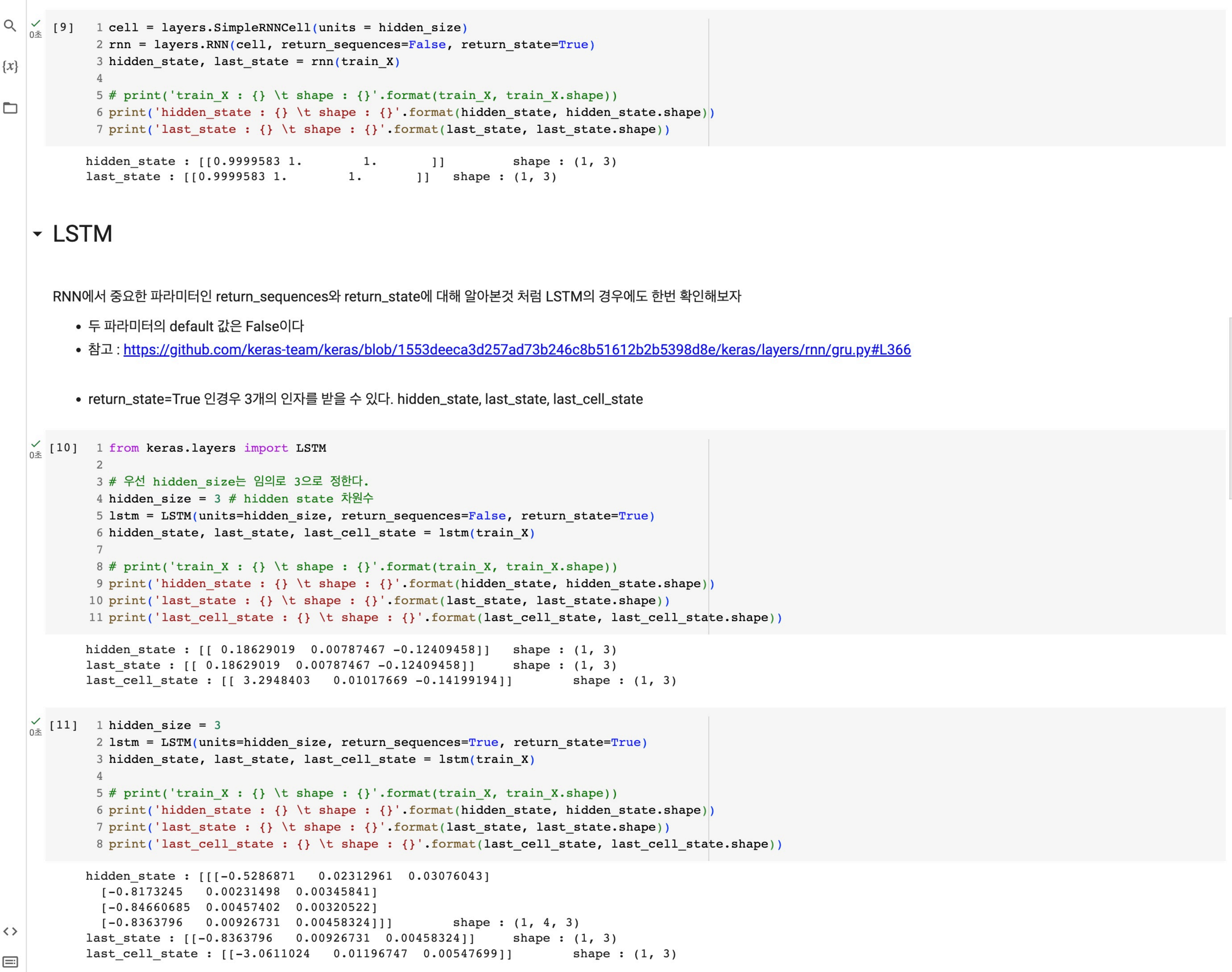
- return_sequence의 값이 True/False인지 관계없이 마지막 시점의 은닉상태를 출력

```
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))
```

```
hidden_state : [[[-0.99983186  0.99867743 -0.95606935]
  [-1.          0.9999968  -0.9998911 ]
  [-1.          0.99999416 -0.9994281 ]
  [-1.          0.99999034 -0.9968363 ]]]        shape : (1, 4, 3)
last_state : [[-1.          0.99999034 -0.9968363 ]]      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))
```

```
hidden_state : [[0.9999583 1.        1.       ]]         shape : (1, 3)
last_state : [[0.9999583 1.        1.       ]]   shape : (1, 3)
```

LSTM

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

- 두 파라미터의 default 값은 False이다
- 참고 : https://github.com/keras-team/keras/blob/1553deeca3d257ad73b246c8b51612b2b5398d8e/keras/layers/rnn/gru.py#L366

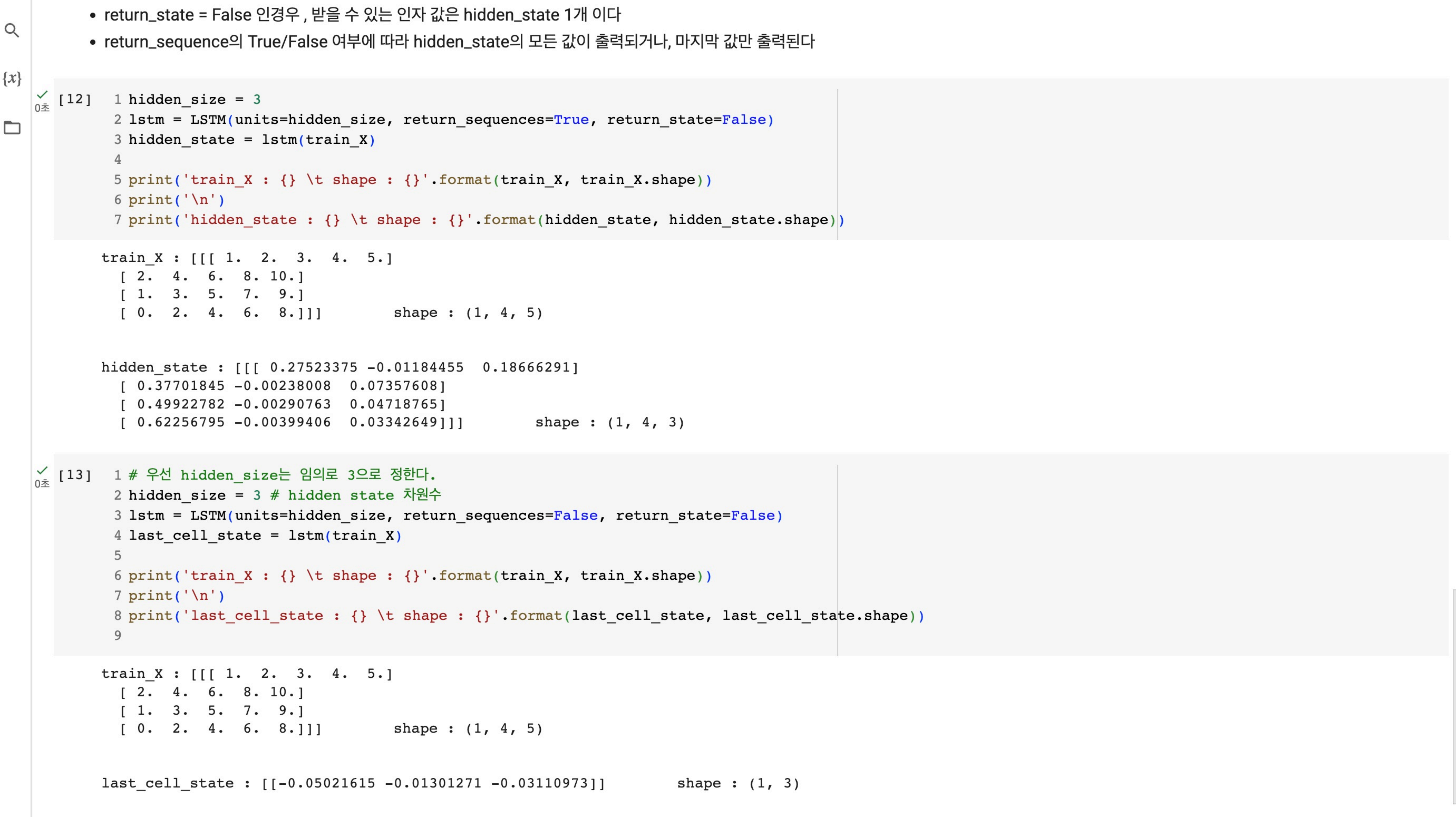
- return_state=True 인경우 3개의 인자를 받을 수 있다. hidden_state, last_state, last_cell_state

```
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))
```

```
hidden_state : [[ 0.18629019  0.00787467 -0.12409458]]   shape : (1, 3)
last_state : [[ 0.18629019  0.00787467 -0.12409458]]     shape : (1, 3)
last_cell_state : [[ 3.2948403   0.01017669 -0.14199194]]        shape : (1, 3)
```

```
hidden_size = 3
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))
```

```
hidden_state : [[[-0.5286871   0.02312961  0.03076043]
  [-0.8173245   0.00231498  0.00345841]
  [-0.84660685  0.00457402  0.00320522]
  [-0.8363796   0.00926731  0.00458324]]]        shape : (1, 4, 3)
last_state : [[-0.8363796   0.00926731  0.00458324]]     shape : (1, 3)
last_cell_state : [[-3.0611024   0.01196747  0.00547699]]        shape : (1, 3)
```



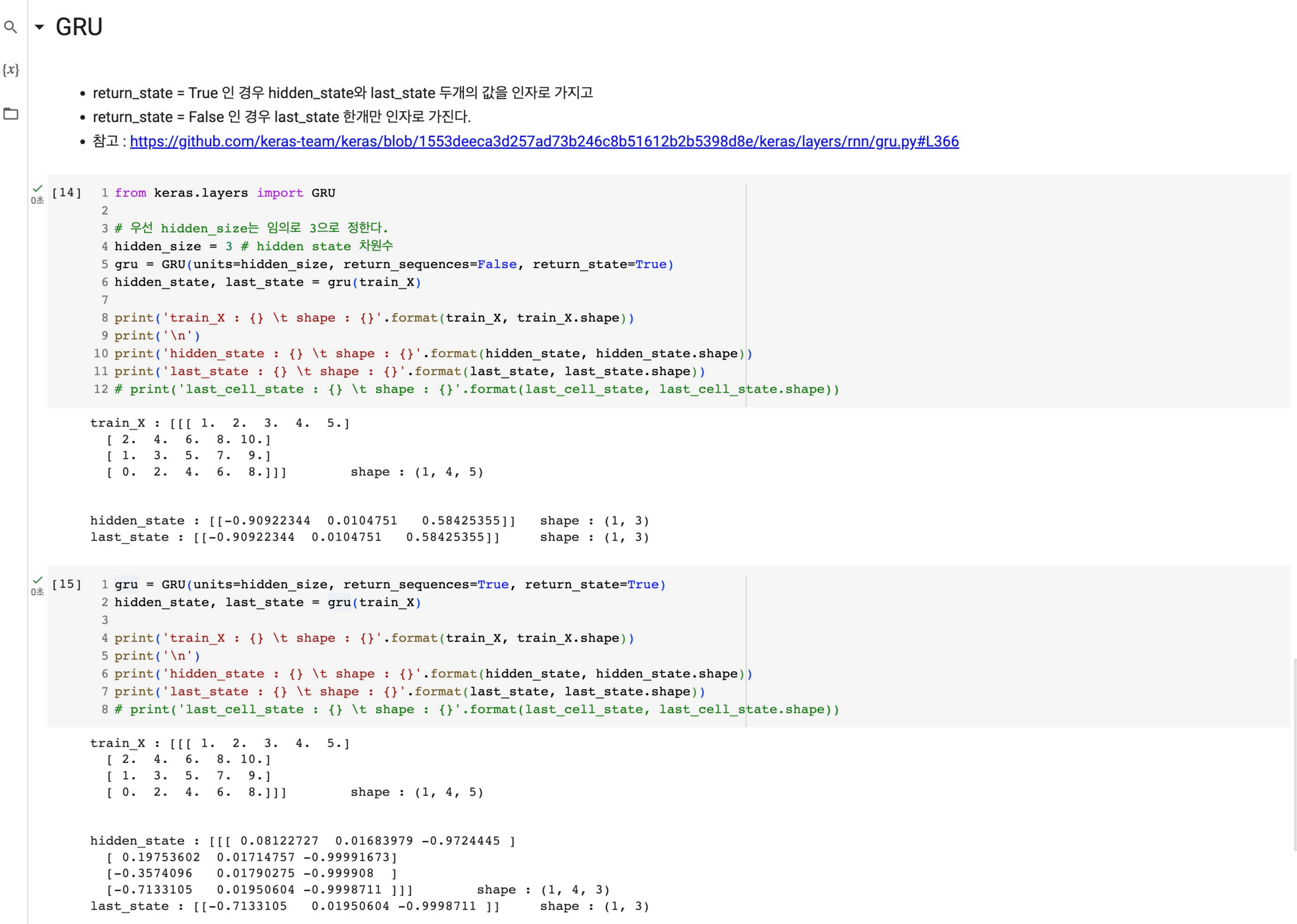
- return_state = False 인경우 , 받을 수 있는 인자 값은 hidden_state 1개 이다
- return_sequence의 True/False 여부에 따라 hidden_state의 모든 값이 출력되거나, 마지막 값만 출력된다

```
hidden_size = 3
lstm = LSTM(units=hidden_size, return_sequences=True, return_state=False)
hidden_state = lstm(train_X)

print('train_X : {} \t shape : {}'.format(train_X, train_X.shape))
print('\n')
print('hidden_state : {} \t shape : {}'.format(hidden_state, hidden_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 : [[[ 0.27523375 -0.01184455  0.18666291]
  [ 0.37701845 -0.00238008  0.07357608]
  [ 0.49922782 -0.00290763  0.04718765]
  [ 0.62256795 -0.00399406  0.03342649]]]        shape : (1, 4, 3)
```

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

print('train_X : {} \t shape : {}'.format(train_X, train_X.shape))
print('\n')
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)


last_cell_state : [[-0.05021615 -0.01301271 -0.03110973]]        shape : (1, 3)
```



GRU

- return_state = True 인 경우 hidden_state와 last_state 두개의 값을 인자로 가지고
- return_state = False 인 경우 last_state 한개만 인자로 가진다.
- 참고 : https://github.com/keras-team/keras/blob/1553deeca3d257ad73b246c8b51612b2b5398d8e/keras/layers/rnn/gru.py#L366

[14] 0초

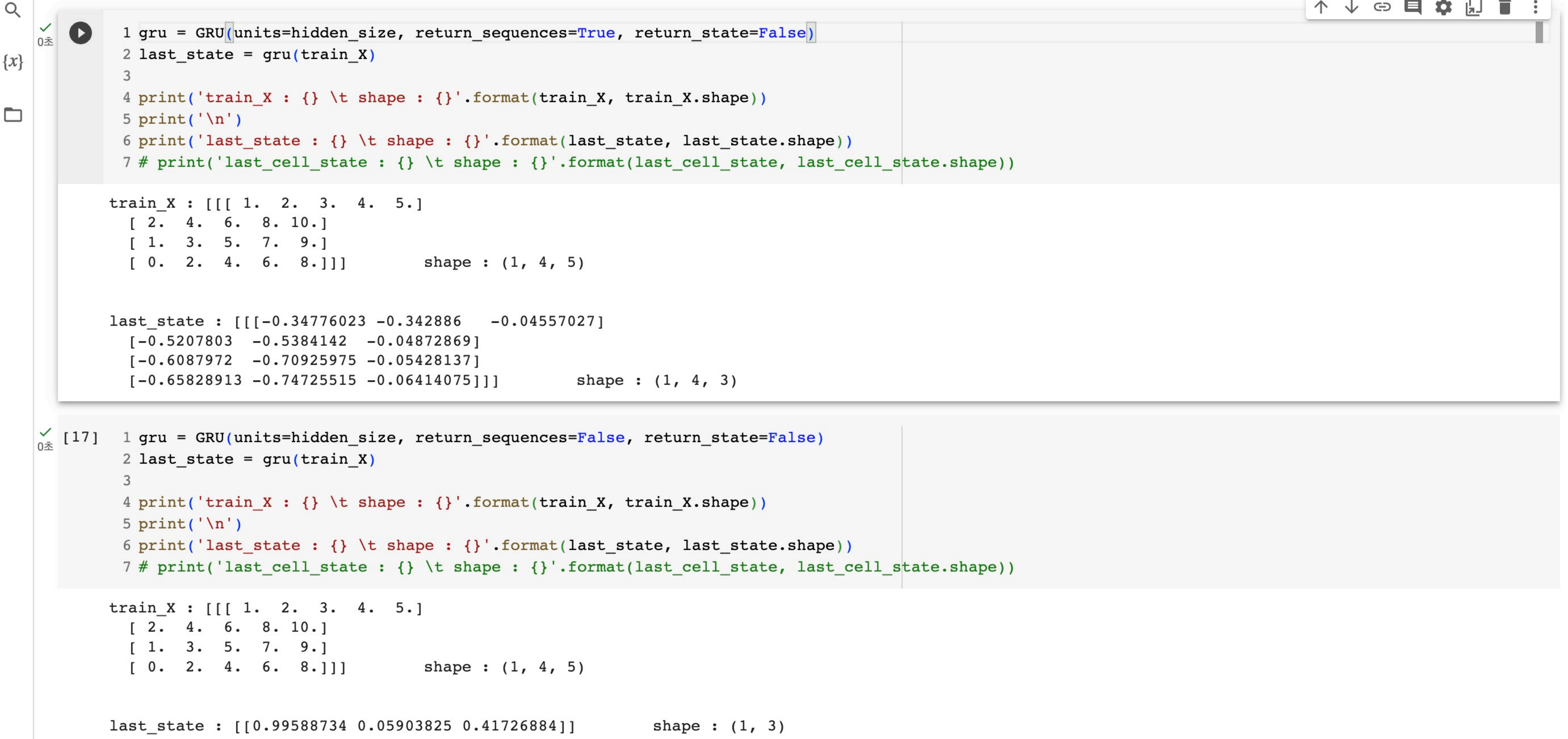
```python
from keras.layers import GRU

# 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 = gru(train_X)

print('train_X : {} \t shape : {}'.format(train_X, train_X.shape))
print('\n')
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 : [[-0.90922344  0.0104751   0.58425355]]     shape : (1, 3)
last_state : [[-0.90922344  0.0104751   0.58425355]]     shape : (1, 3)
```

[15] 0초

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

print('train_X : {} \t shape : {}'.format(train_X, train_X.shape))
print('\n')
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 : [[[ 0.08122727  0.01683979 -0.9724445 ]
  [ 0.19753602  0.01714757 -0.99991673]
  [-0.3574096   0.01790275 -0.999908  ]
  [-0.7133105   0.01950604 -0.9998711 ]]]        shape : (1, 4, 3)
last_state : [[-0.7133105   0.01950604 -0.9998711 ]]     shape : (1, 3)
```



```python
gru = GRU(units=hidden_size, return_sequences=True, return_state=False)
last_state = gru(train_X)

print('train_X : {} \t shape : {}'.format(train_X, train_X.shape))
print('\n')
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)


last_state : [[[-0.34776023 -0.342886   -0.04557027]
  [-0.5207803  -0.5384142  -0.04872869]
  [-0.6087972  -0.70925975 -0.05428137]
  [-0.65828913 -0.74725515 -0.06414075]]]        shape : (1, 4, 3)
```

```python
gru = GRU(units=hidden_size, return_sequences=False, return_state=False)
last_state = gru(train_X)

print('train_X : {} \t shape : {}'.format(train_X, train_X.shape))
print('\n')
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)


last_state : [[0.99588734 0.05903825 0.41726884]]        shape : (1, 3)
```