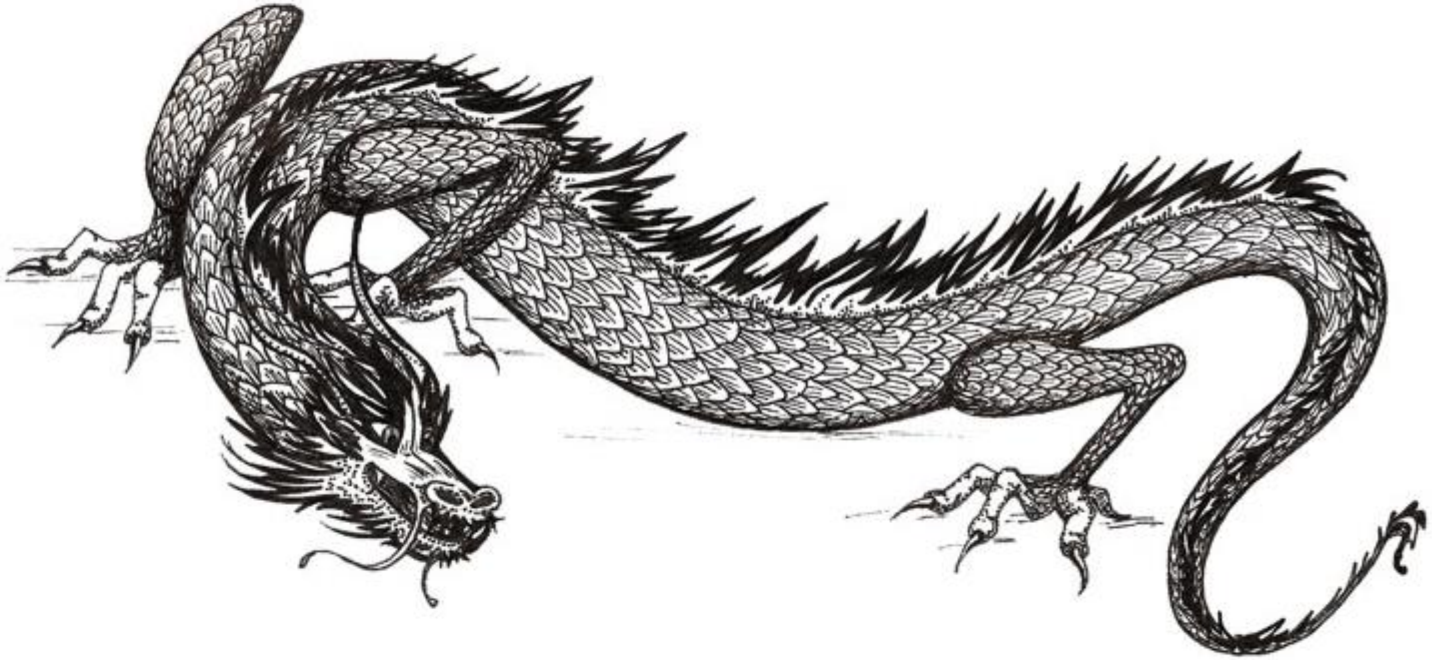


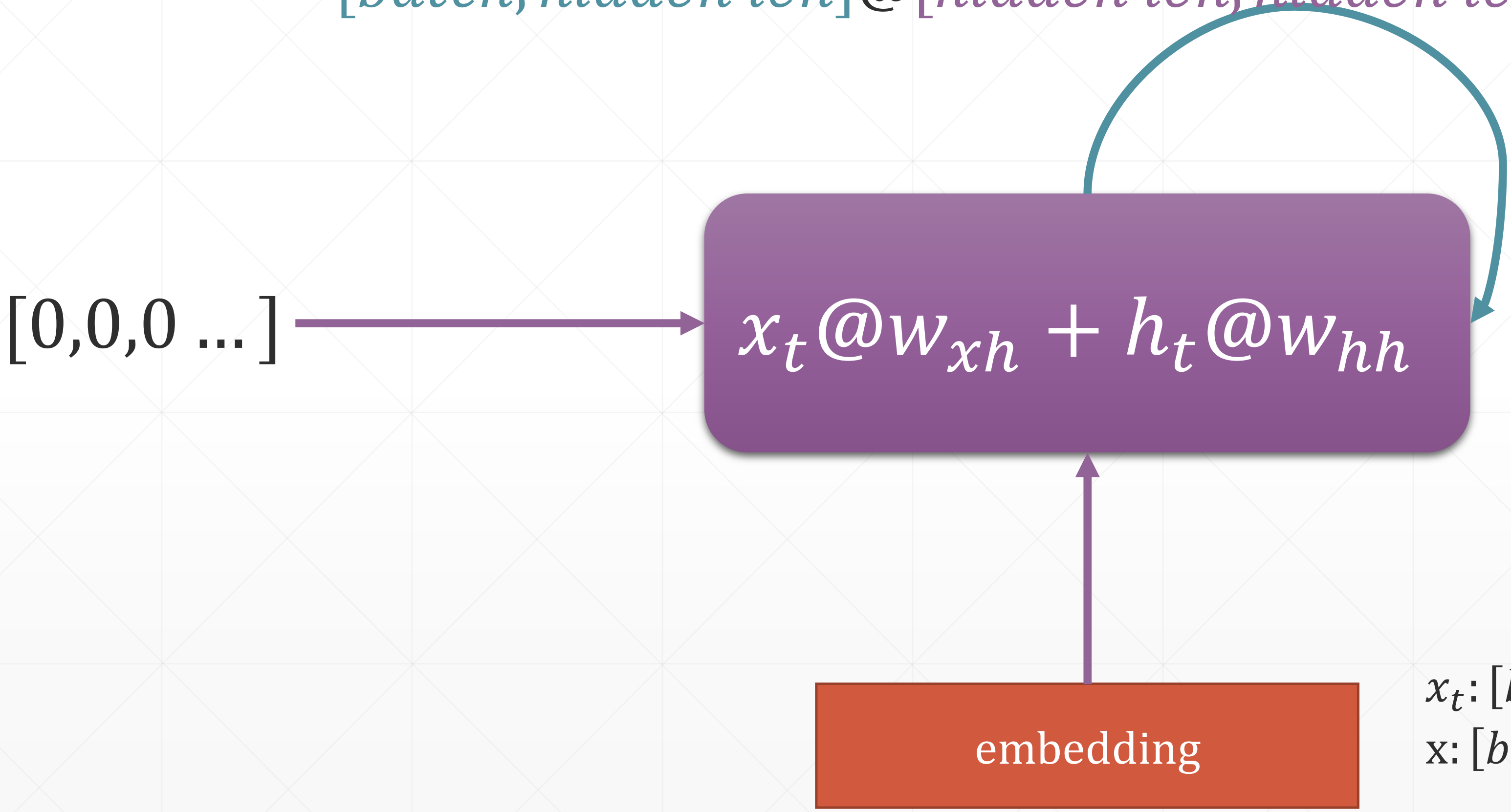
Simple RNN Layer

主讲：龙良曲

Recap

$$[batch, feature\ len]@[feature\ len,\ \ hidden\ len]\ \ +$$

$$[batch, hidden\ len]@[hidden\ len, hidden\ len]$$



$[0,0,0\ \ldots]$

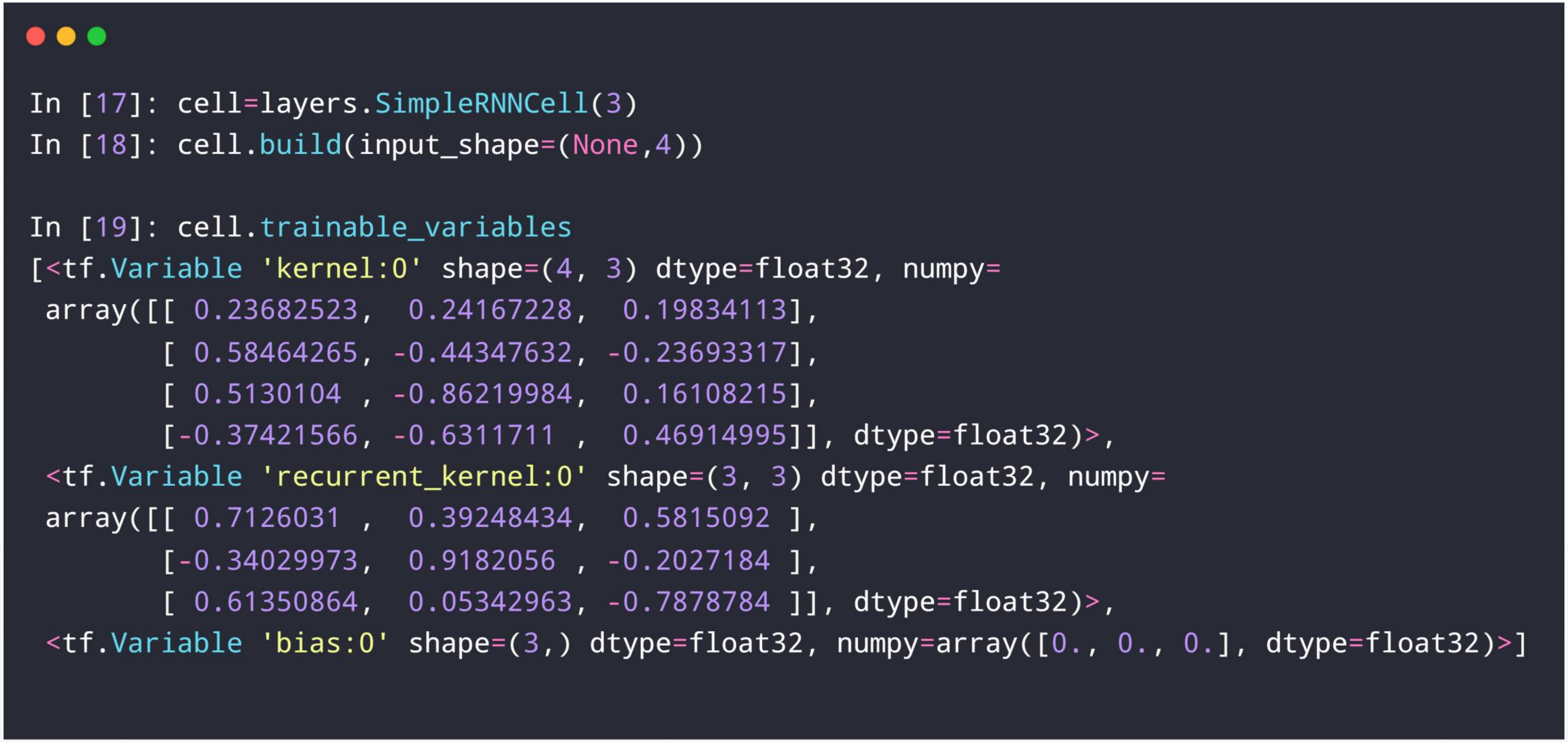
$x_t@w_{xh} + h_t@w_{hh}$

embedding

x_t: $[batch, feature\ len]$

x: $[batch, seq\ len, feature\ len]$

input dim, hidden dim

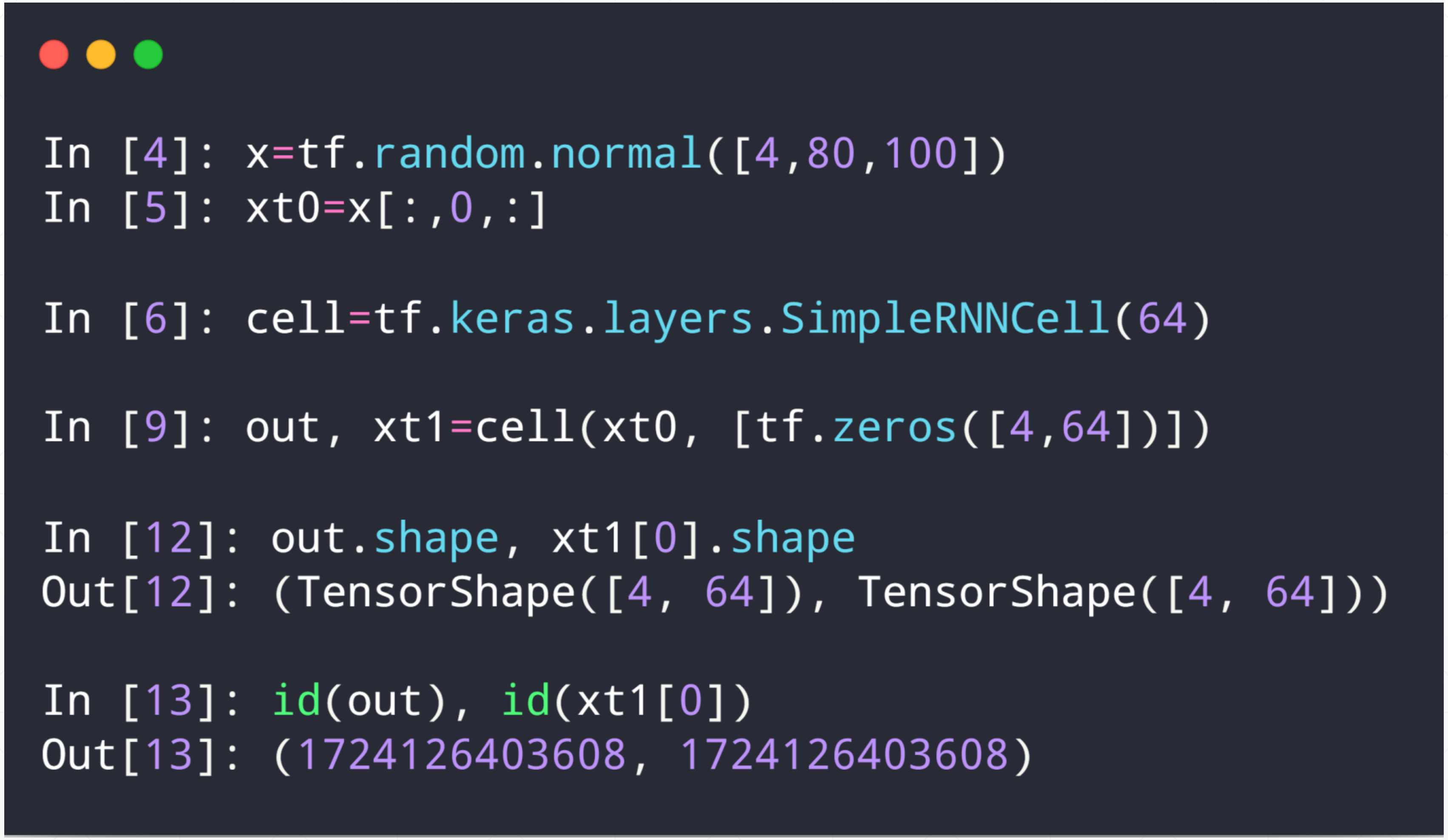


```
In [17]: cell=layers.SimpleRNNCell(3)
In [18]: cell.build(input_shape=(None,4))

In [19]: cell.trainable_variables
[<tf.Variable 'kernel:0' shape=(4, 3) dtype=float32, numpy=
 array([[ 0.23682523,  0.24167228,  0.19834113],
        [ 0.58464265, -0.44347632, -0.23693317],
        [ 0.5130104 , -0.86219984,  0.16108215],
        [-0.37421566, -0.6311711 ,  0.46914995]], dtype=float32)>,
 <tf.Variable 'recurrent_kernel:0' shape=(3, 3) dtype=float32, numpy=
 array([[ 0.7126031 ,  0.39248434,  0.5815092 ],
        [-0.34029973,  0.9182056 , -0.2027184 ],
        [ 0.61350864,  0.05342963, -0.7878784 ]], dtype=float32)>,
 <tf.Variable 'bias:0' shape=(3,) dtype=float32, numpy=array([0., 0., 0.], dtype=float32)>]
```

SimpleRNNCell

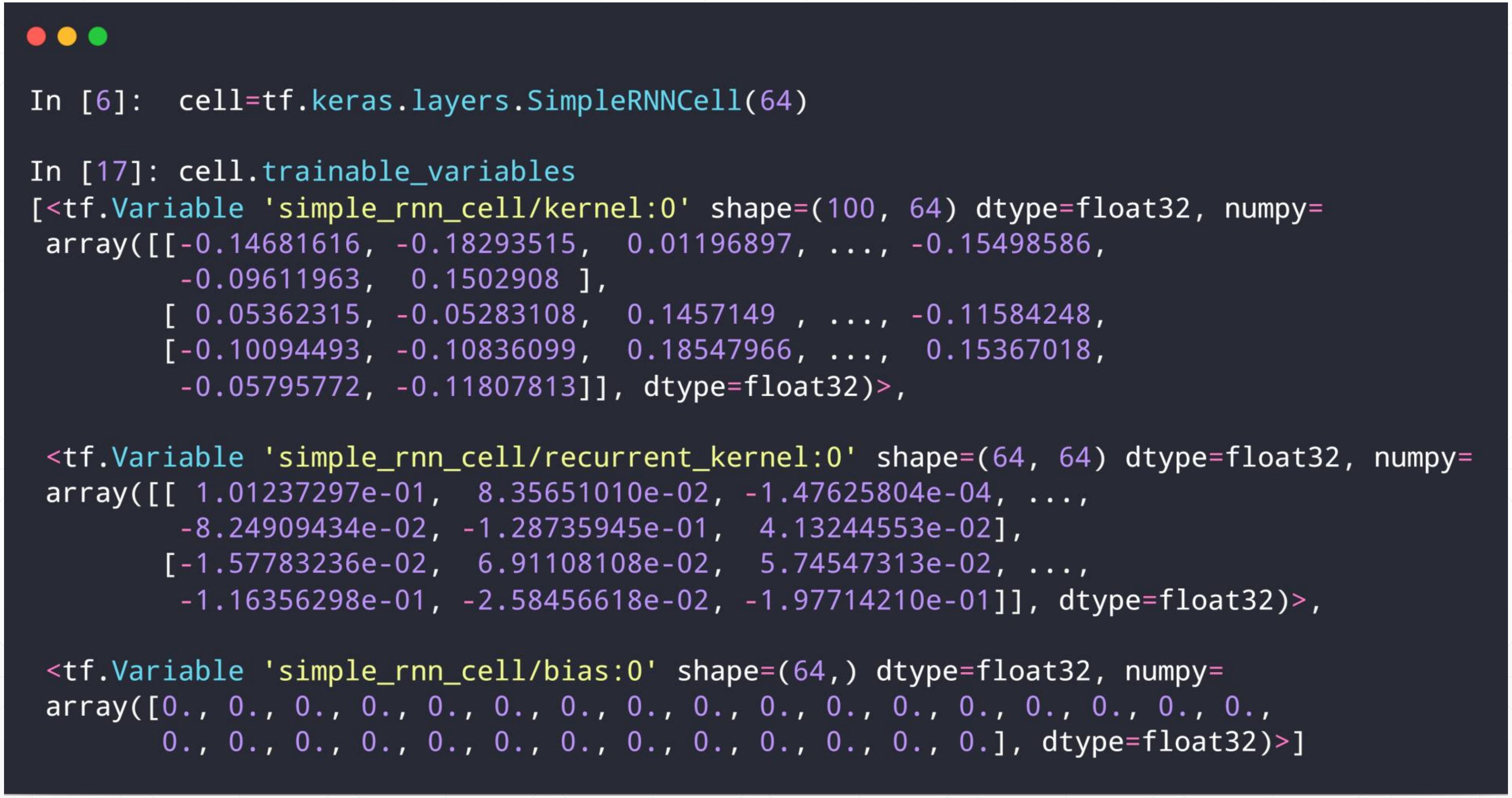
- $out, h_1 \;=\; call(x, h_0)$
  - x: $[b, seq\ len,\ word\ vec]$
  - h_0/h_1: $[b,\ h\ \mathrm{dim}]$
  - out: $[b,\ h\ \mathrm{dim}]$

Single layer RNN Cell

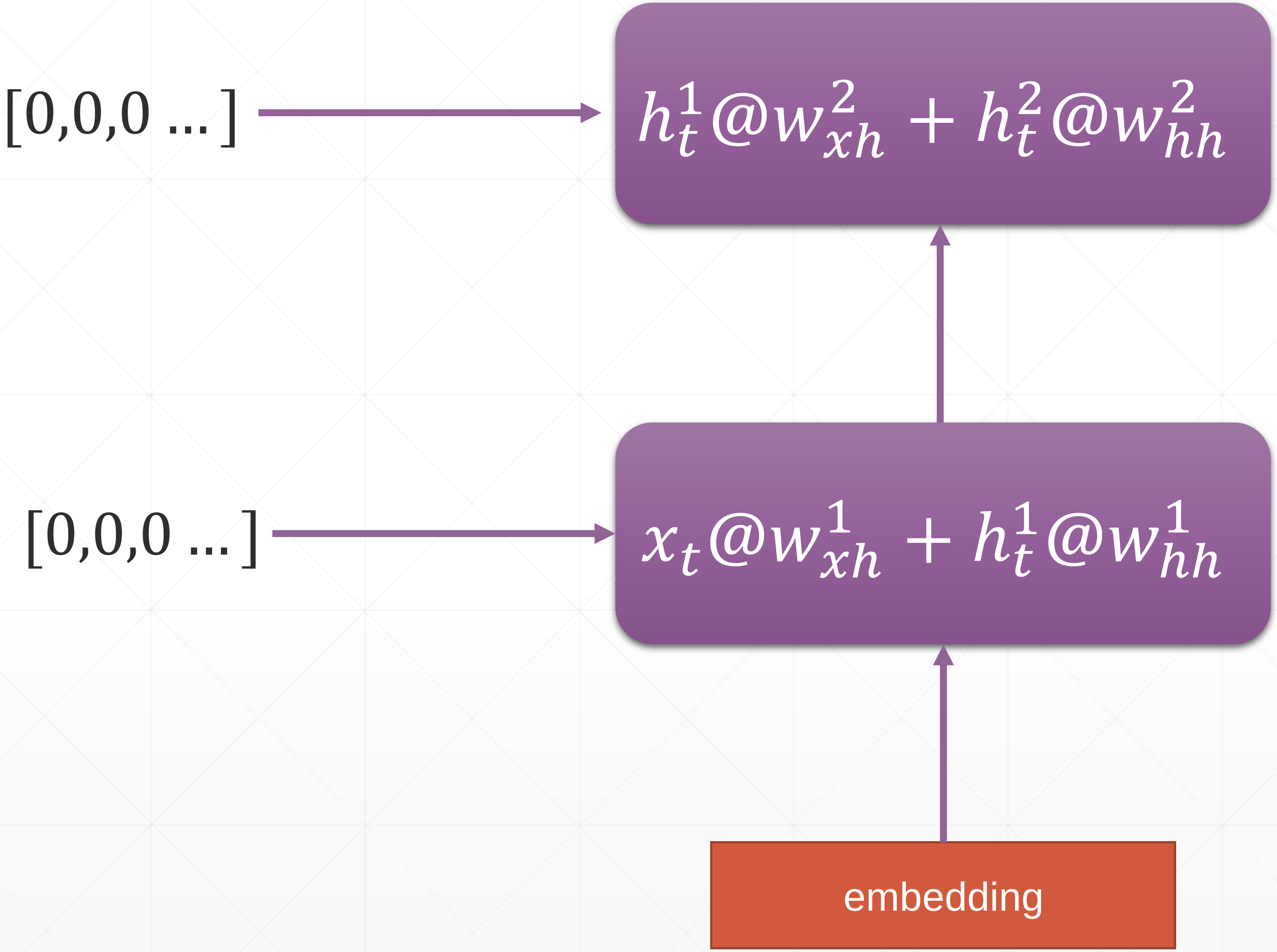


```
In [4]: x=tf.random.normal([4,80,100])
In [5]: xt0=x[:,0,:]

In [6]: cell=tf.keras.layers.SimpleRNNCell(64)

In [9]: out, xt1=cell(xt0, [tf.zeros([4,64])])

In [12]: out.shape, xt1[0].shape
Out[12]: (TensorShape([4, 64]), TensorShape([4, 64]))

In [13]: id(out), id(xt1[0])
Out[13]: (1724126403608, 1724126403608)
```

W, b



```
In [6]:  cell=tf.keras.layers.SimpleRNNCell(64)

In [17]: cell.trainable_variables
[<tf.Variable 'simple_rnn_cell/kernel:0' shape=(100, 64) dtype=float32, numpy=
 array([[-0.14681616, -0.18293515,  0.01196897, ..., -0.15498586,
         -0.09611963,  0.1502908 ],
        [ 0.05362315, -0.05283108,  0.1457149 , ..., -0.11584248,
        [-0.10094493, -0.10836099,  0.18547966, ...,  0.15367018,
         -0.05795772, -0.11807813]], dtype=float32)>,

 <tf.Variable 'simple_rnn_cell/recurrent_kernel:0' shape=(64, 64) dtype=float32, numpy=
 array([[ 1.01237297e-01,  8.35651010e-02, -1.47625804e-04, ...,
         -8.24909434e-02, -1.28735945e-01,  4.13244553e-02],
        [-1.57783236e-02,  6.91108108e-02,  5.74547313e-02, ...,
         -1.16356298e-01, -2.58456618e-02, -1.97714210e-01]], dtype=float32)>,

 <tf.Variable 'simple_rnn_cell/bias:0' shape=(64,) dtype=float32, numpy=
 array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.], dtype=float32)>]
```



[0,0,0 …]

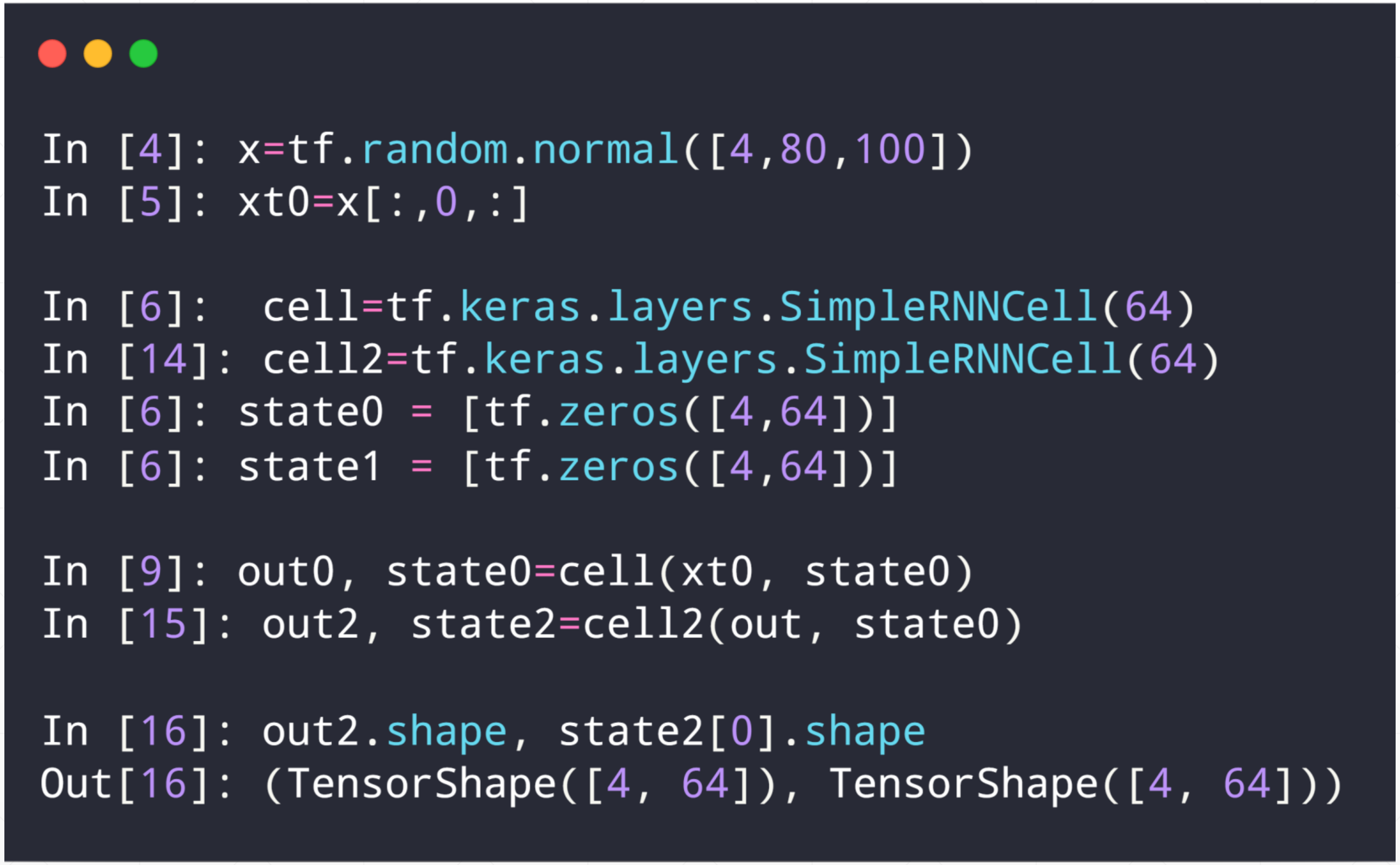
$h_t^1@w_{xh}^2 + h_t^2@w_{hh}^2$

[0,0,0 …]

$x_t@w_{xh}^1 + h_t^1@w_{hh}^1$

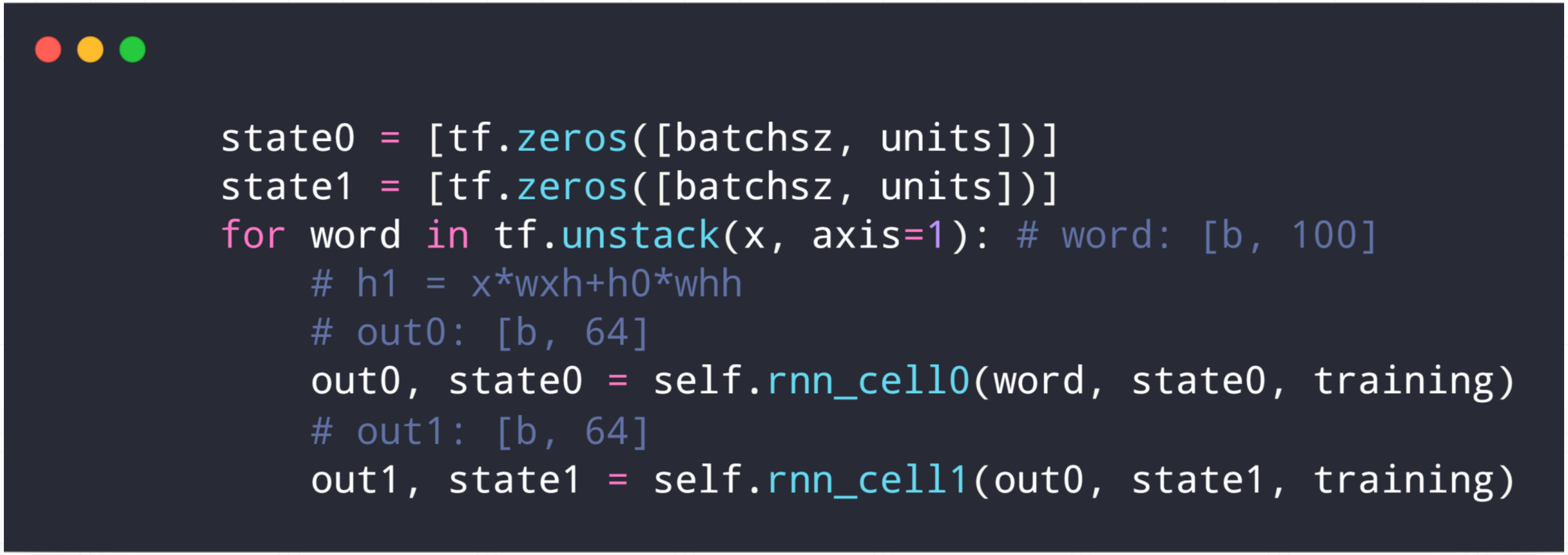
embedding

Multi-Layers RNN



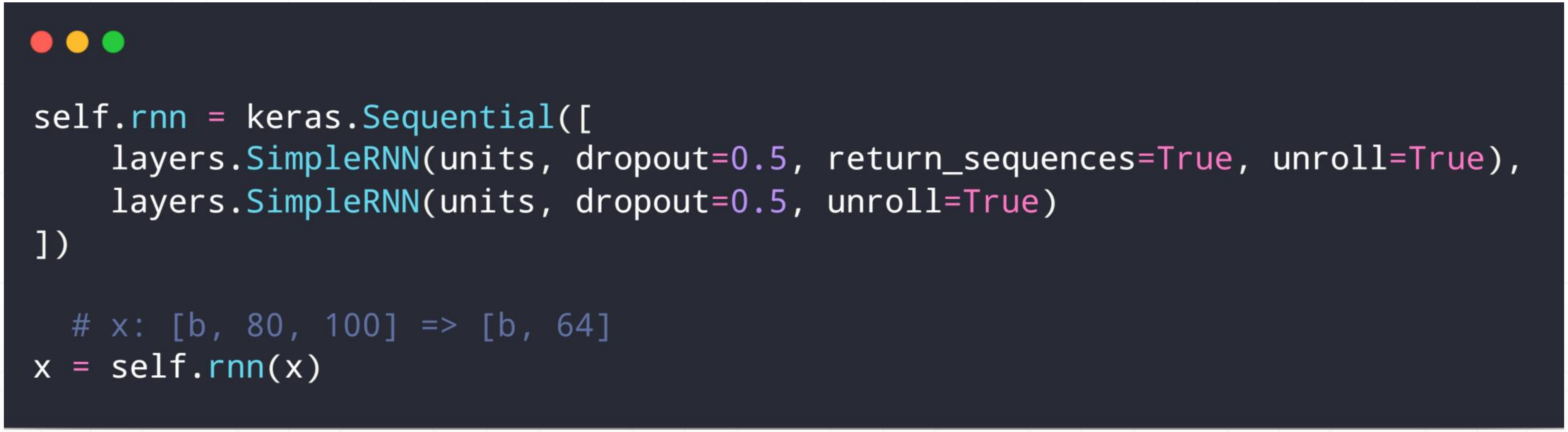
```
In [4]: x=tf.random.normal([4,80,100])
In [5]: xt0=x[:,0,:]

In [6]:  cell=tf.keras.layers.SimpleRNNCell(64)
In [14]: cell2=tf.keras.layers.SimpleRNNCell(64)
In [6]: state0 = [tf.zeros([4,64])]
In [6]: state1 = [tf.zeros([4,64])]

In [9]: out0, state0=cell(xt0, state0)
In [15]: out2, state2=cell2(out, state0)

In [16]: out2.shape, state2[0].shape
Out[16]: (TensorShape([4, 64]), TensorShape([4, 64]))
```

Multi-Layers RNN



```
state0 = [tf.zeros([batchsz, units])]
state1 = [tf.zeros([batchsz, units])]
for word in tf.unstack(x, axis=1): # word: [b, 100]
    # h1 = x*wxh+h0*whh
    # out0: [b, 64]
    out0, state0 = self.rnn_cell0(word, state0, training)
    # out1: [b, 64]
    out1, state1 = self.rnn_cell1(out0, state1, training)
```

RNN Layer



```
self.rnn = keras.Sequential([
    layers.SimpleRNN(units, dropout=0.5, return_sequences=True, unroll=True),
    layers.SimpleRNN(units, dropout=0.5, unroll=True)
])

  # x: [b, 80, 100] => [b, 64]
x = self.rnn(x)
```

下一课时

RNN实战

Thank You.