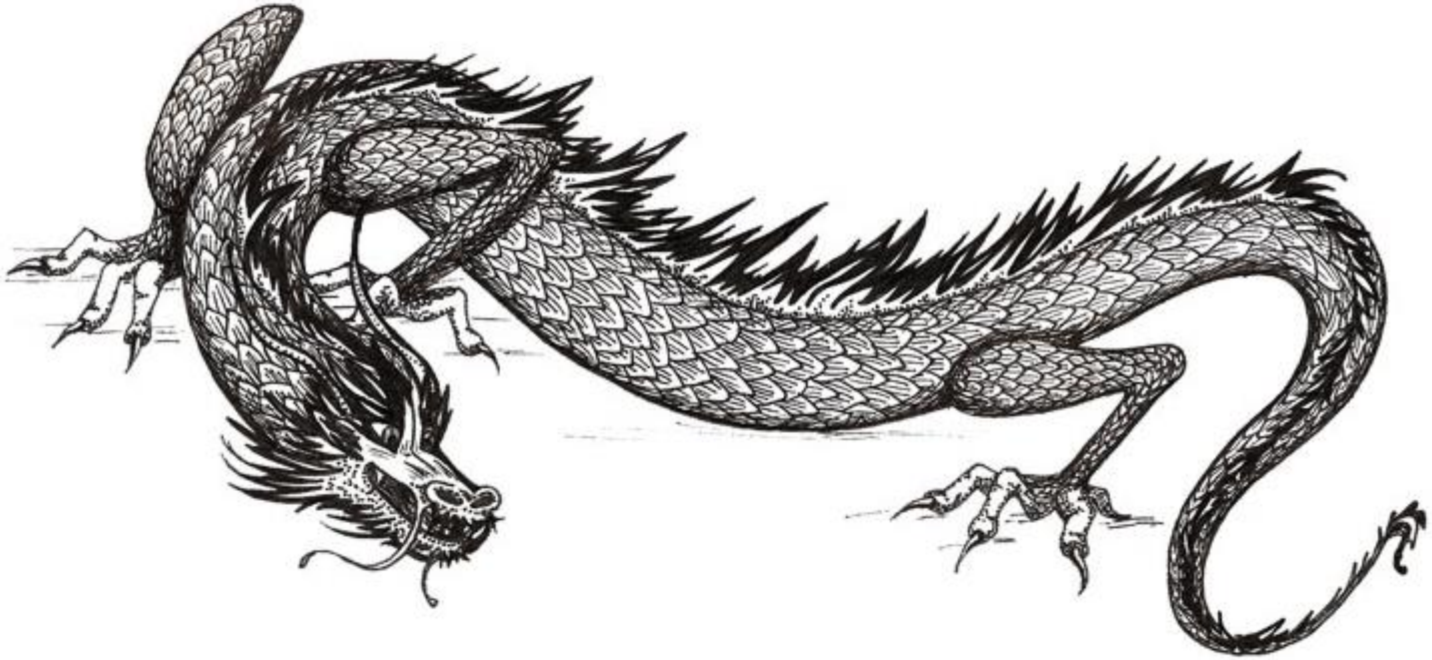


数据集加载

主讲：龙良曲

Outline

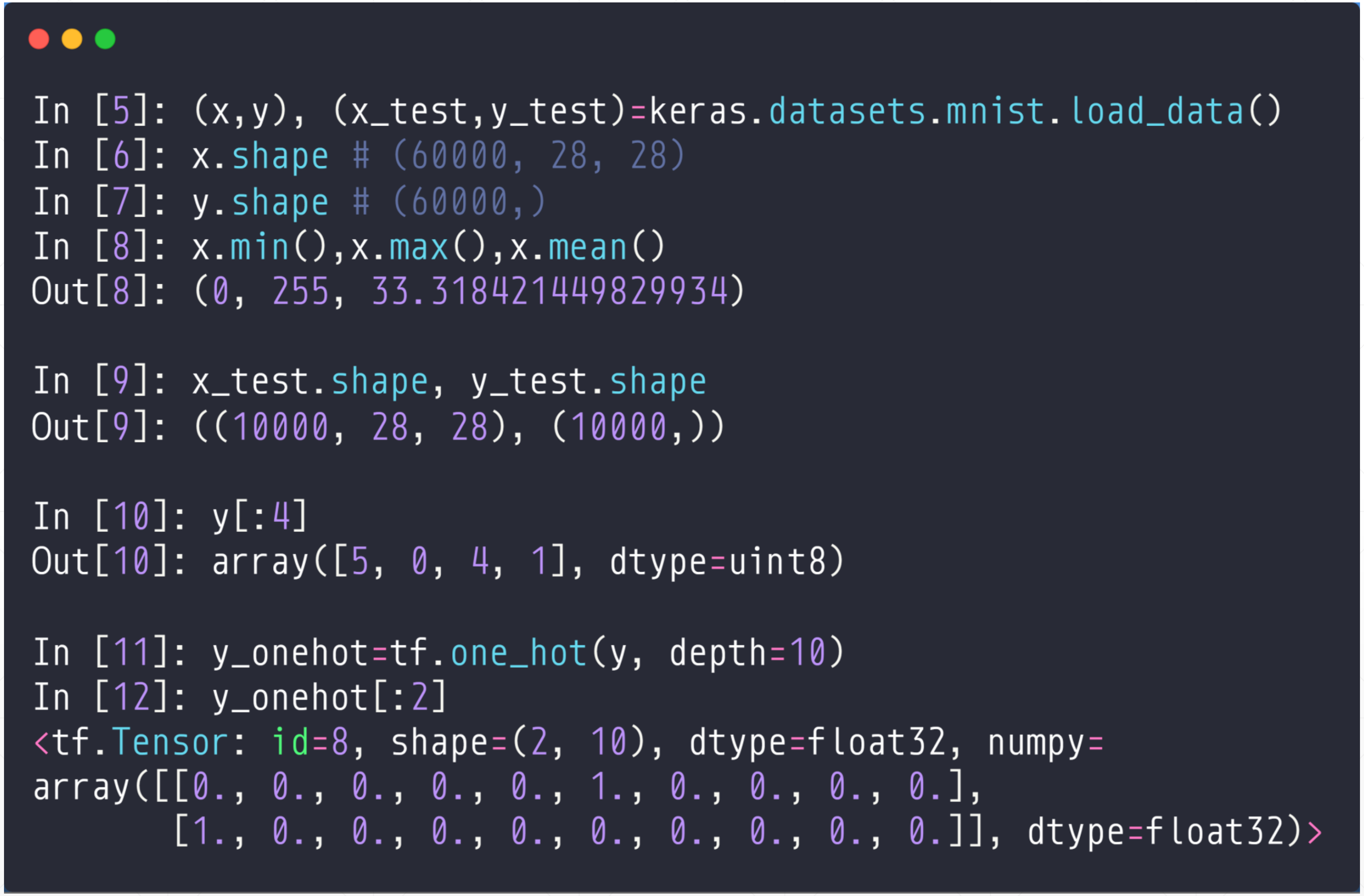
- keras.datasets
- tf.data.Dataset.from_tensor_slices
  - shuffle
  - map
  - batch
  - repeat
- we will talk *Input Pipeline* later

keras.datasets

- **boston housing**
  - Boston housing price regression dataset.
- **mnist/fashion mnist**
  - MNIST/Fashion-MNIST dataset.
- **cifar10/100**
  - small images classification dataset.
- **imdb**
  - sentiment classification dataset.

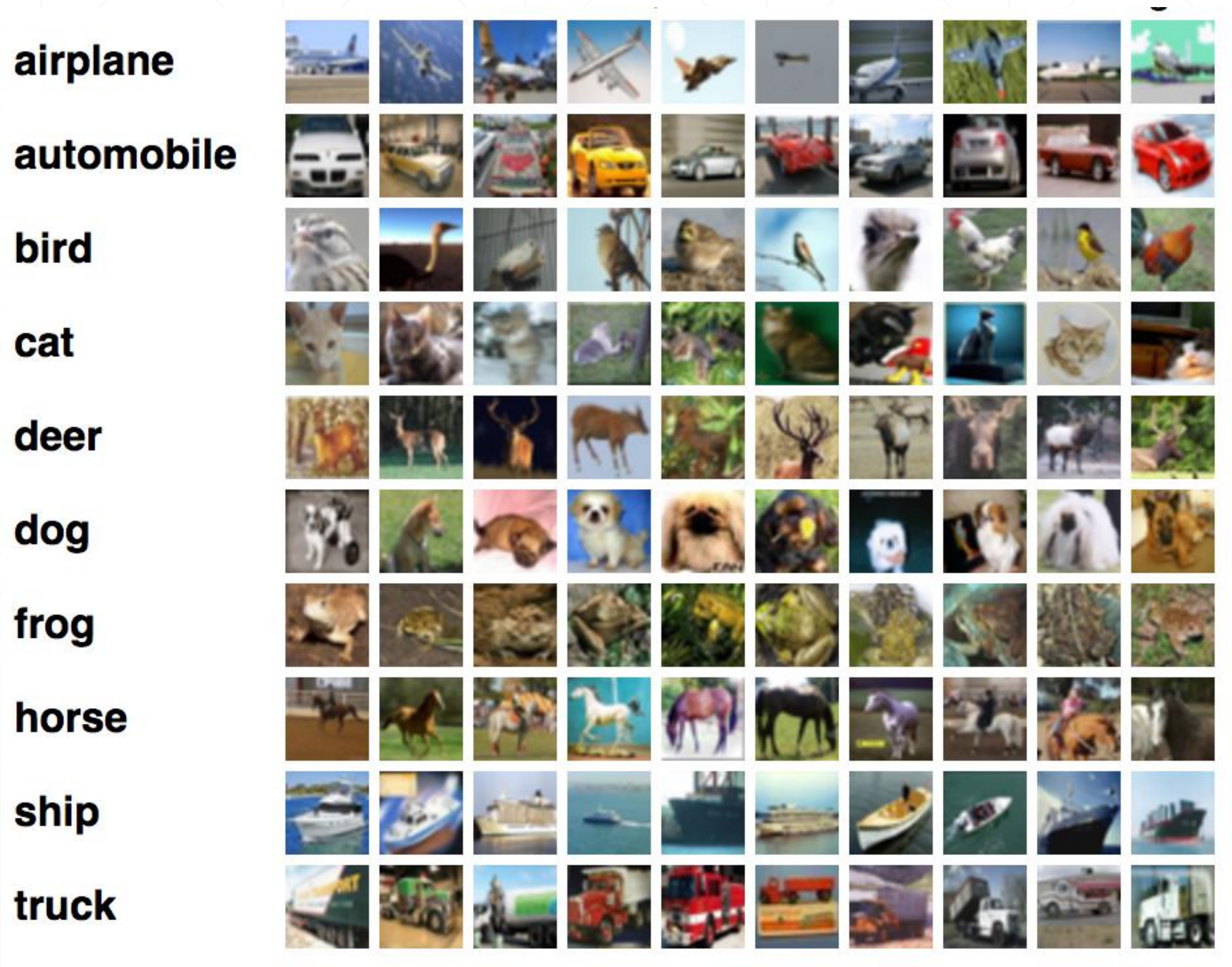
MNIST

MNIST

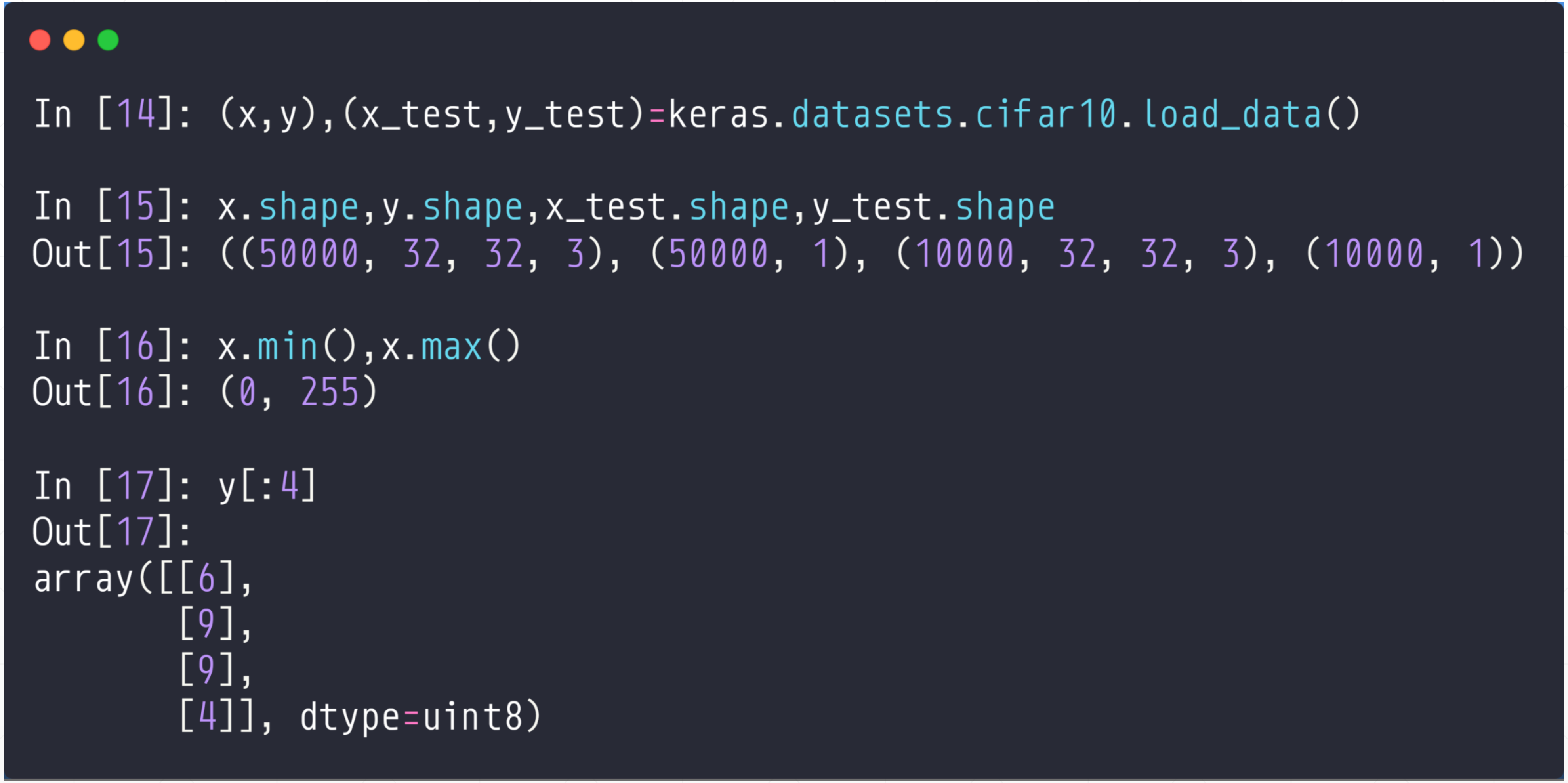


```
In [5]: (x,y), (x_test,y_test)=keras.datasets.mnist.load_data()
In [6]: x.shape # (60000, 28, 28)
In [7]: y.shape # (60000,)
In [8]: x.min(),x.max(),x.mean()
Out[8]: (0, 255, 33.318421449829934)

In [9]: x_test.shape, y_test.shape
Out[9]: ((10000, 28, 28), (10000,))

In [10]: y[:4]
Out[10]: array([5, 0, 4, 1], dtype=uint8)

In [11]: y_onehot=tf.one_hot(y, depth=10)
In [12]: y_onehot[:2]
<tf.Tensor: id=8, shape=(2, 10), dtype=float32, numpy=
array([[0., 0., 0., 0., 0., 1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)>
```

CIFAR10/100


airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

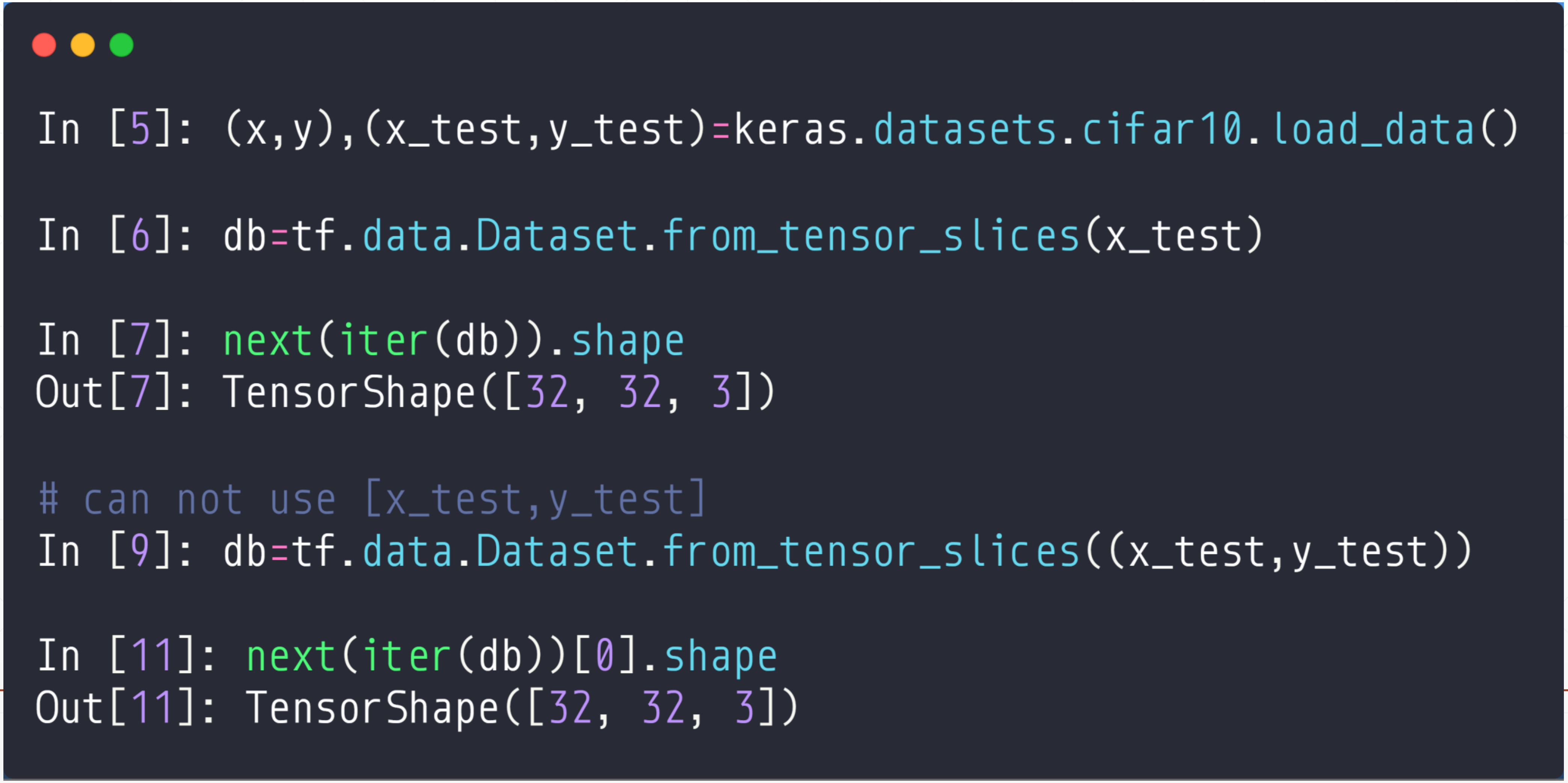
CIFAR10/100



```
In [14]: (x,y),(x_test,y_test)=keras.datasets.cifar10.load_data()

In [15]: x.shape,y.shape,x_test.shape,y_test.shape
Out[15]: ((50000, 32, 32, 3), (50000, 1), (10000, 32, 32, 3), (10000, 1))

In [16]: x.min(),x.max()
Out[16]: (0, 255)

In [17]: y[:4]
Out[17]:
array([[6],
       [9],
       [9],
       [4]], dtype=uint8)
```

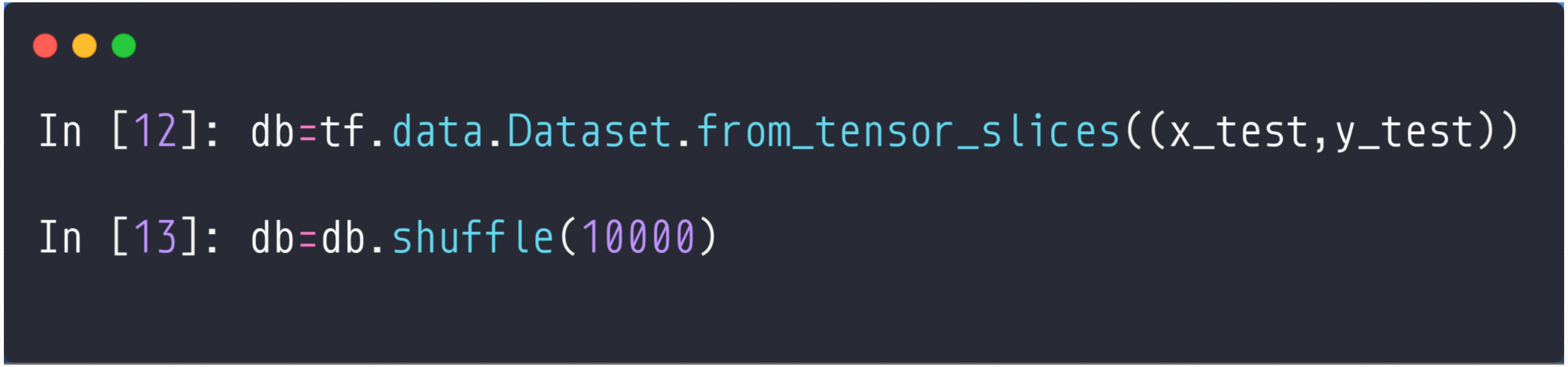
tf.data.Dataset

- from_tensor_slices()



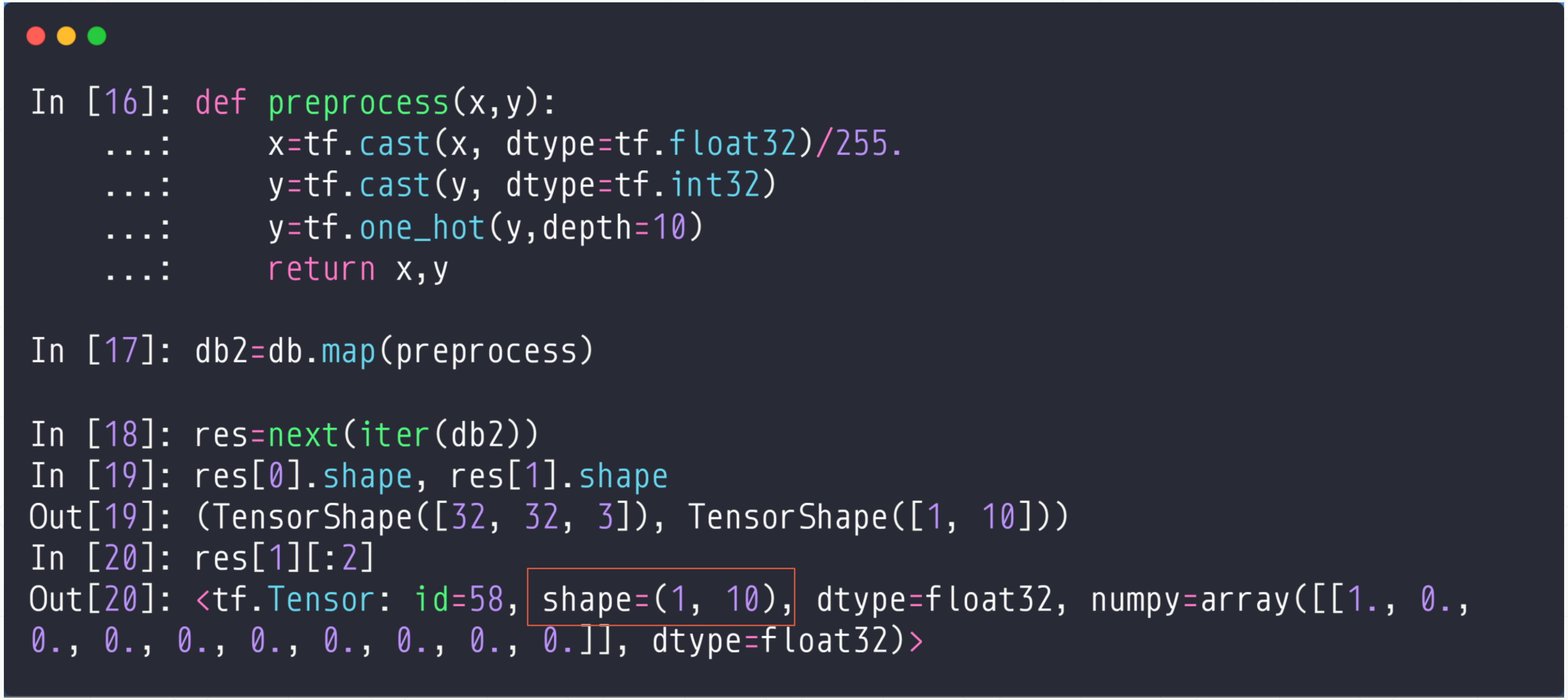
```
In [5]: (x,y),(x_test,y_test)=keras.datasets.cifar10.load_data()

In [6]: db=tf.data.Dataset.from_tensor_slices(x_test)

In [7]: next(iter(db)).shape
Out[7]: TensorShape([32, 32, 3])

# can not use [x_test,y_test]
In [9]: db=tf.data.Dataset.from_tensor_slices((x_test,y_test))

In [11]: next(iter(db))[0].shape
Out[11]: TensorShape([32, 32, 3])
```

.shuffle



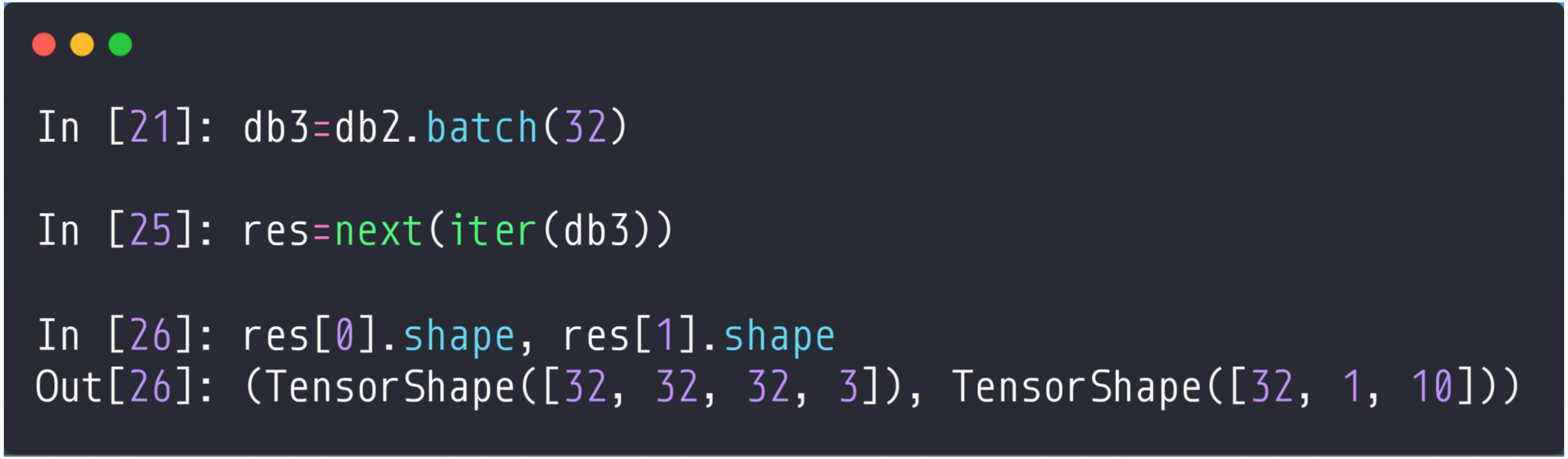
```
In [12]: db=tf.data.Dataset.from_tensor_slices((x_test,y_test))

In [13]: db=db.shuffle(10000)
```

.map



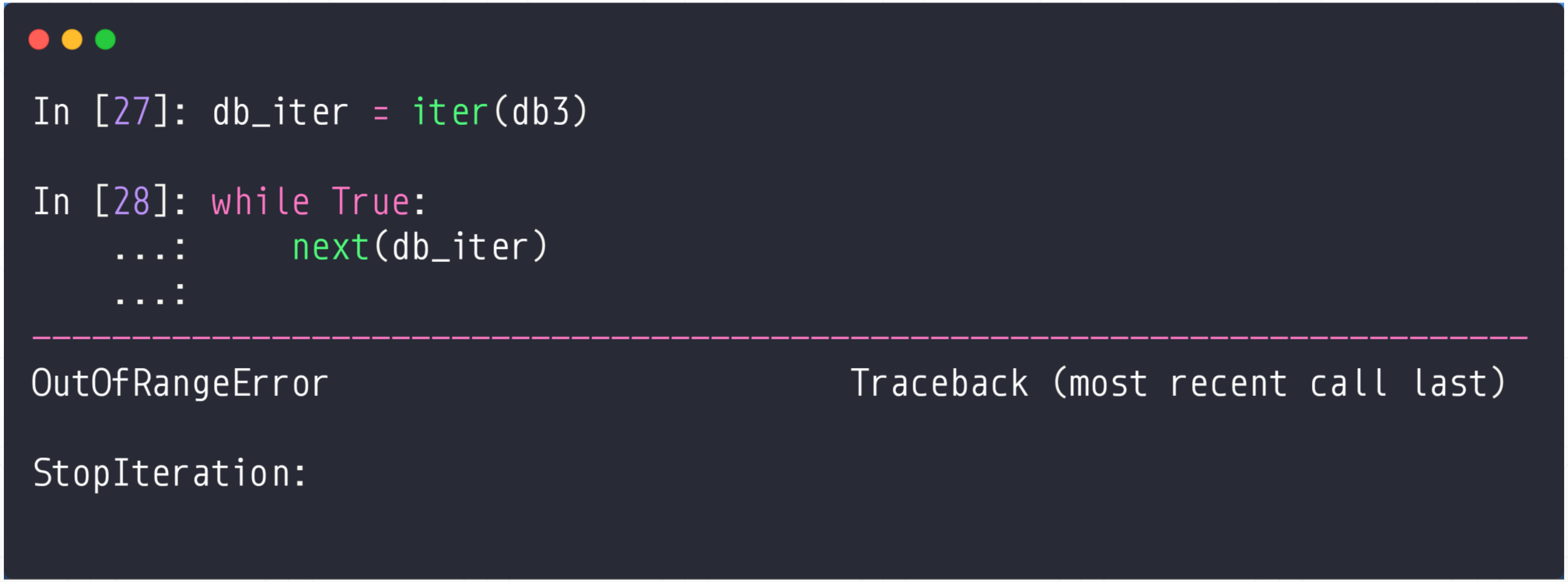
```
In [16]: def preprocess(x,y):
    ...:     x=tf.cast(x, dtype=tf.float32)/255.
    ...:     y=tf.cast(y, dtype=tf.int32)
    ...:     y=tf.one_hot(y,depth=10)
    ...:     return x,y

In [17]: db2=db.map(preprocess)

In [18]: res=next(iter(db2))
In [19]: res[0].shape, res[1].shape
Out[19]: (TensorShape([32, 32, 3]), TensorShape([1, 10]))
In [20]: res[1][:2]
Out[20]: <tf.Tensor: id=58, shape=(1, 10), dtype=float32, numpy=array([[1., 0.,
0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)>
```

.batch



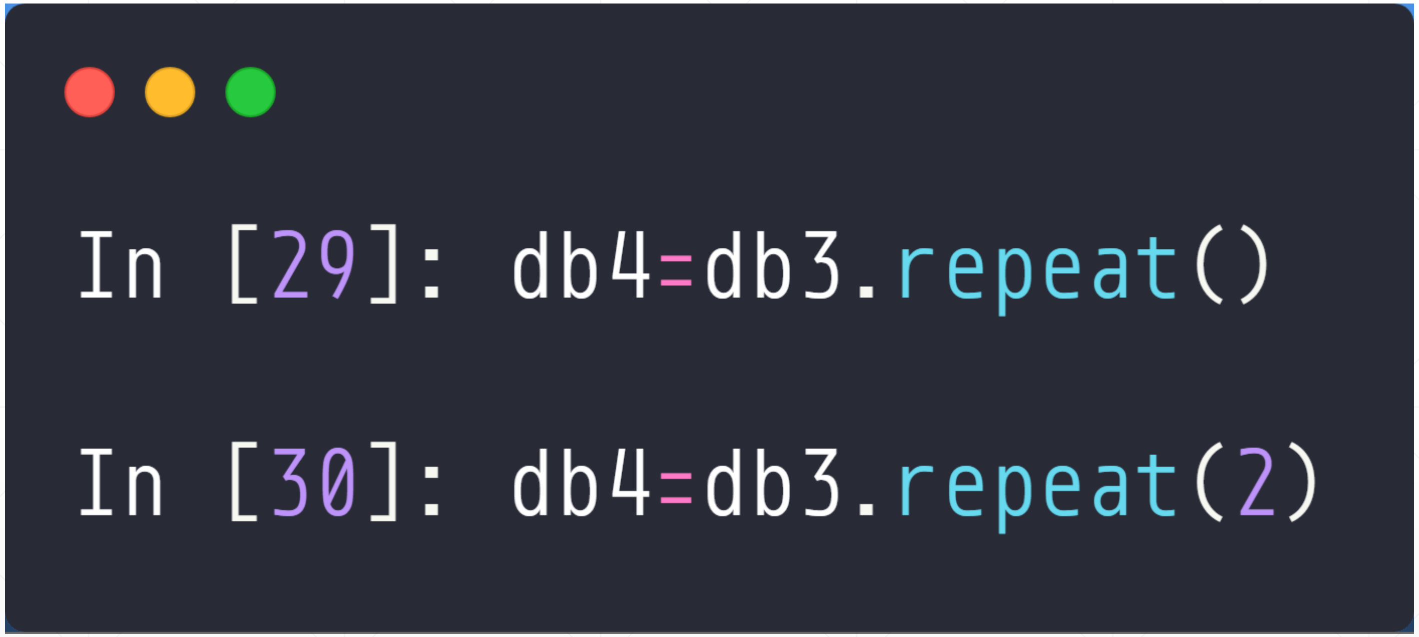
```
In [21]: db3=db2.batch(32)

In [25]: res=next(iter(db3))

In [26]: res[0].shape, res[1].shape
Out[26]: (TensorShape([32, 32, 32, 3]), TensorShape([32, 1, 10]))
```

StopIteration



```
In [27]: db_iter = iter(db3)

In [28]: while True:
    ...:     next(db_iter)
    ...:
---------------------------------------------------------------------------
OutOfRangeError                           Traceback (most recent call last)

StopIteration:
```

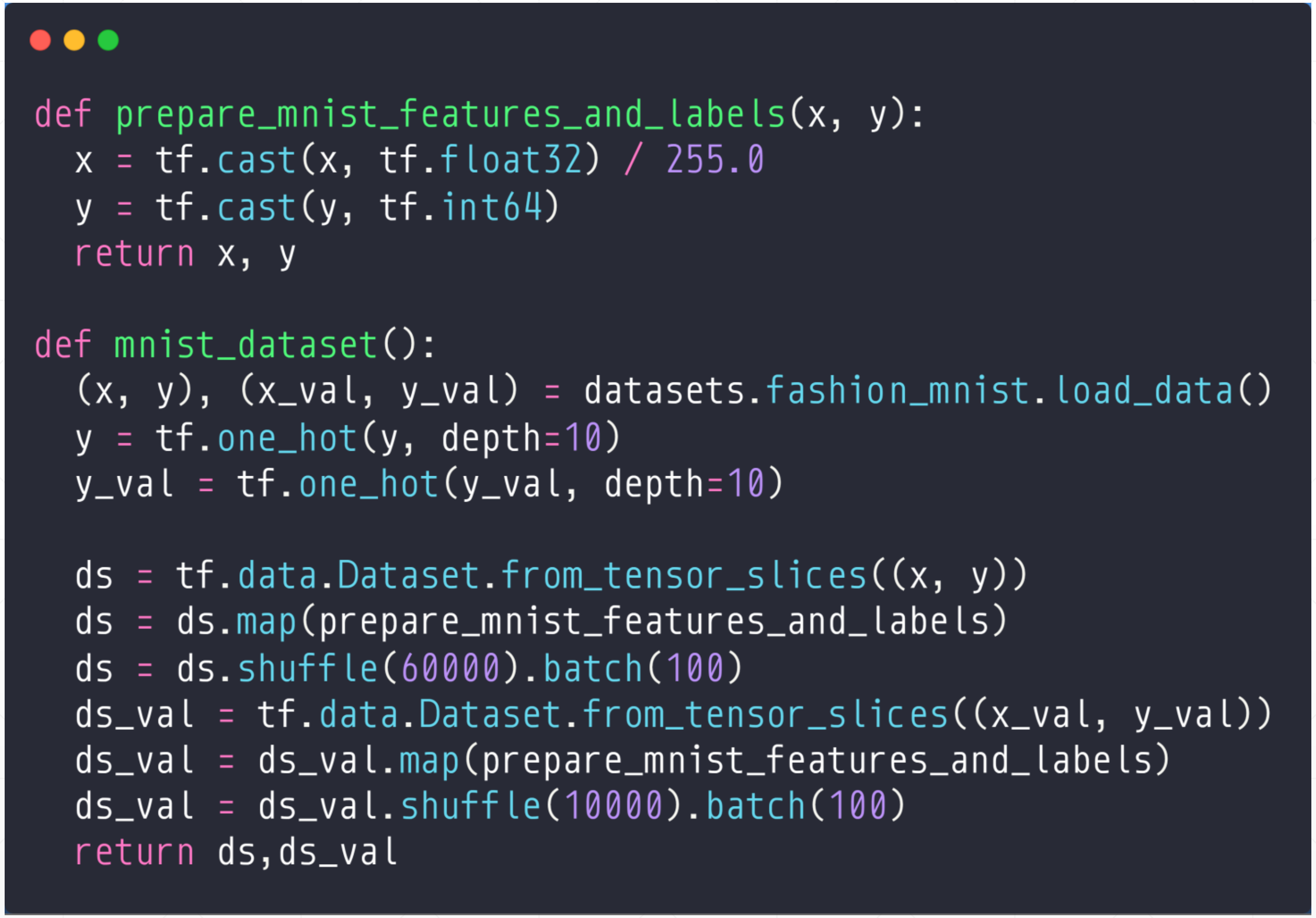
.repeat()



```
In [29]: db4=db3.repeat()

In [30]: db4=db3.repeat(2)
```

使用for x,y in db4时会自动迭代两次，若不传参数则会迭代无数次

For example



```
def prepare_mnist_features_and_labels(x, y):
  x = tf.cast(x, tf.float32) / 255.0
  y = tf.cast(y, tf.int64)
  return x, y

def mnist_dataset():
  (x, y), (x_val, y_val) = datasets.fashion_mnist.load_data()
  y = tf.one_hot(y, depth=10)
  y_val = tf.one_hot(y_val, depth=10)

  ds = tf.data.Dataset.from_tensor_slices((x, y))
  ds = ds.map(prepare_mnist_features_and_labels)
  ds = ds.shuffle(60000).batch(100)
  ds_val = tf.data.Dataset.from_tensor_slices((x_val, y_val))
  ds_val = ds_val.map(prepare_mnist_features_and_labels)
  ds_val = ds_val.shuffle(10000).batch(100)
  return ds,ds_val
```

下一课时

测试(张量)-实战

Thank You.