CREATING TENSORS WITH

tf.constant

tf.Variable

tf.zeros

tf.ones

TENSORFLOW

A Multidimensional array

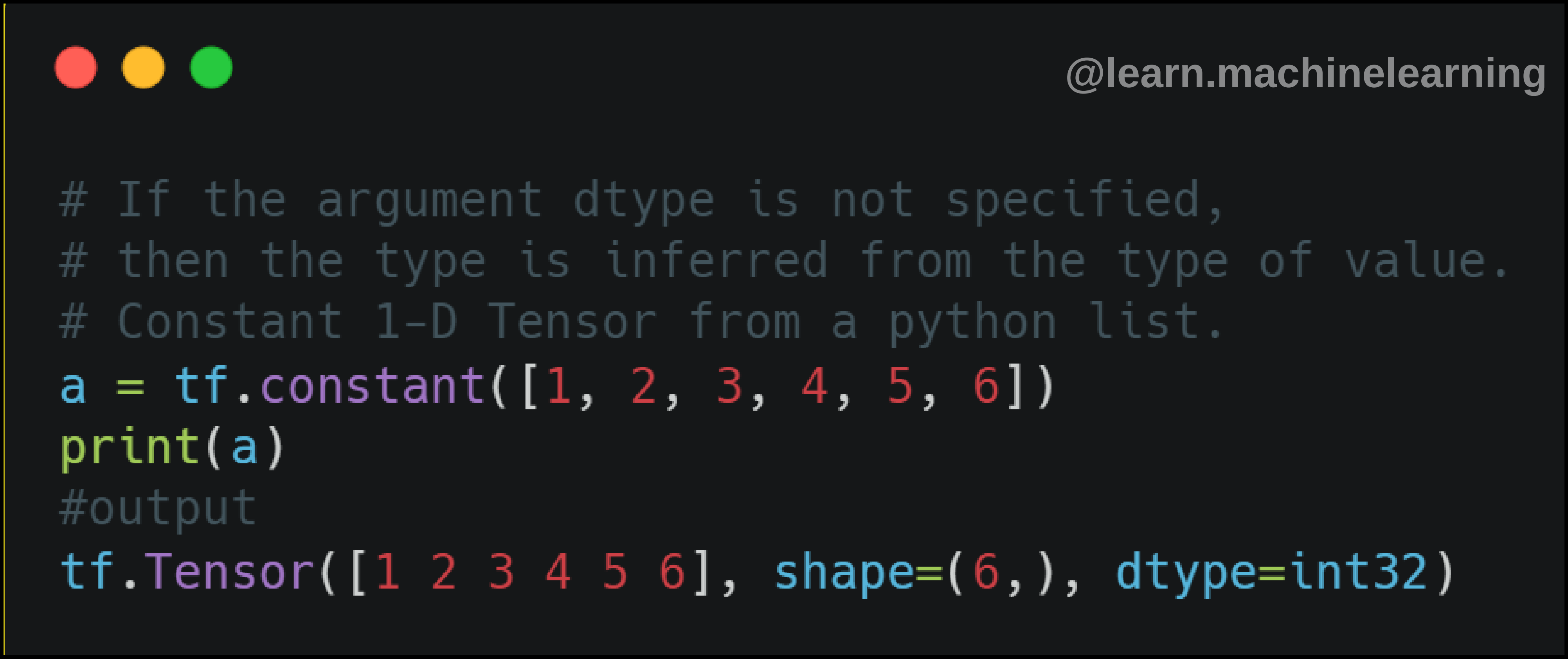
A Graph of operations

tf.constant

It creates a constant tensor from a tensor-like object. that does not change

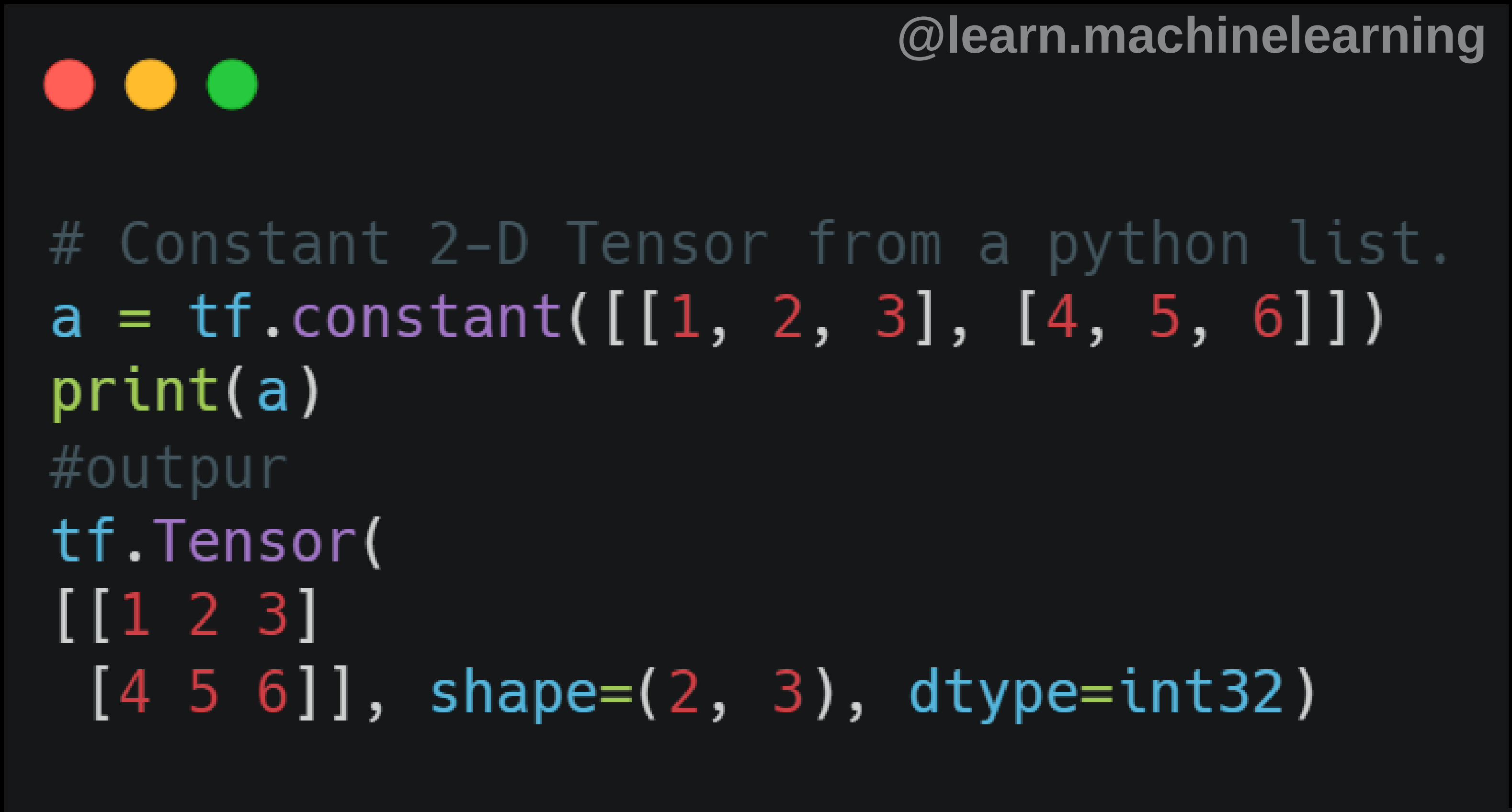
tf.constant(value, dtype=None, shape=None, name='Const')

- **value:** A constant value (or list) of output type dtype.
- **dtype:** The type of the elements of the resulting tensor.
- **shape:** Optional dimensions of resulting tensor.
- **name:** Optional name for the tensor.

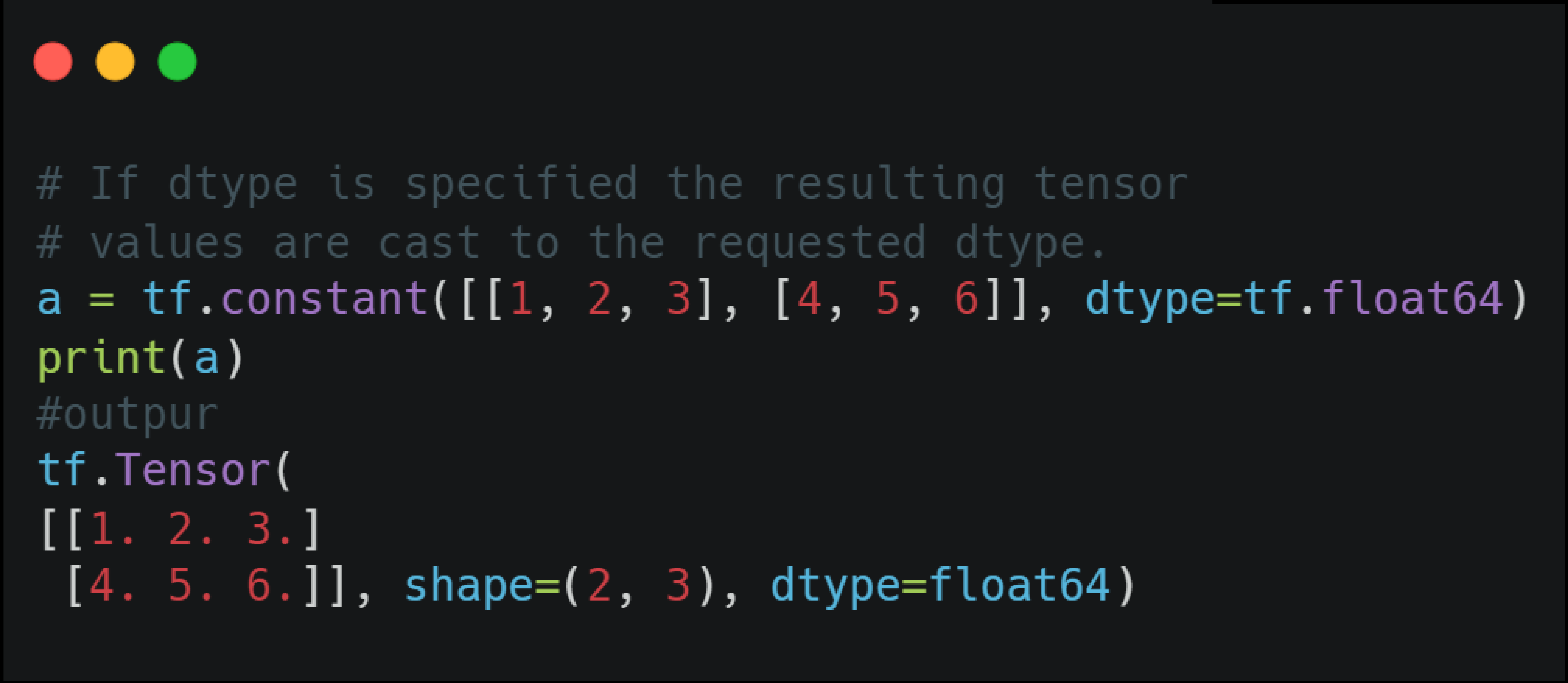


```
# If the argument dtype is not specified,
# then the type is inferred from the type of value.
# Constant 1-D Tensor from a python list.
a = tf.constant([1, 2, 3, 4, 5, 6])
print(a)
#output
tf.Tensor([1 2 3 4 5 6], shape=(6,), dtype=int32)
```

tf.constant



```
# Constant 2-D Tensor from a python list.
a = tf.constant([[1, 2, 3], [4, 5, 6]])
print(a)
#outpur
tf.Tensor(
[[1 2 3]
 [4 5 6]], shape=(2, 3), dtype=int32)
```



```
# If dtype is specified the resulting tensor
# values are cast to the requested dtype.
a = tf.constant([[1, 2, 3], [4, 5, 6]], dtype=tf.float64)
print(a)
#outpur
tf.Tensor(
[[1. 2. 3.]
 [4. 5. 6.]], shape=(2, 3), dtype=float64)
```

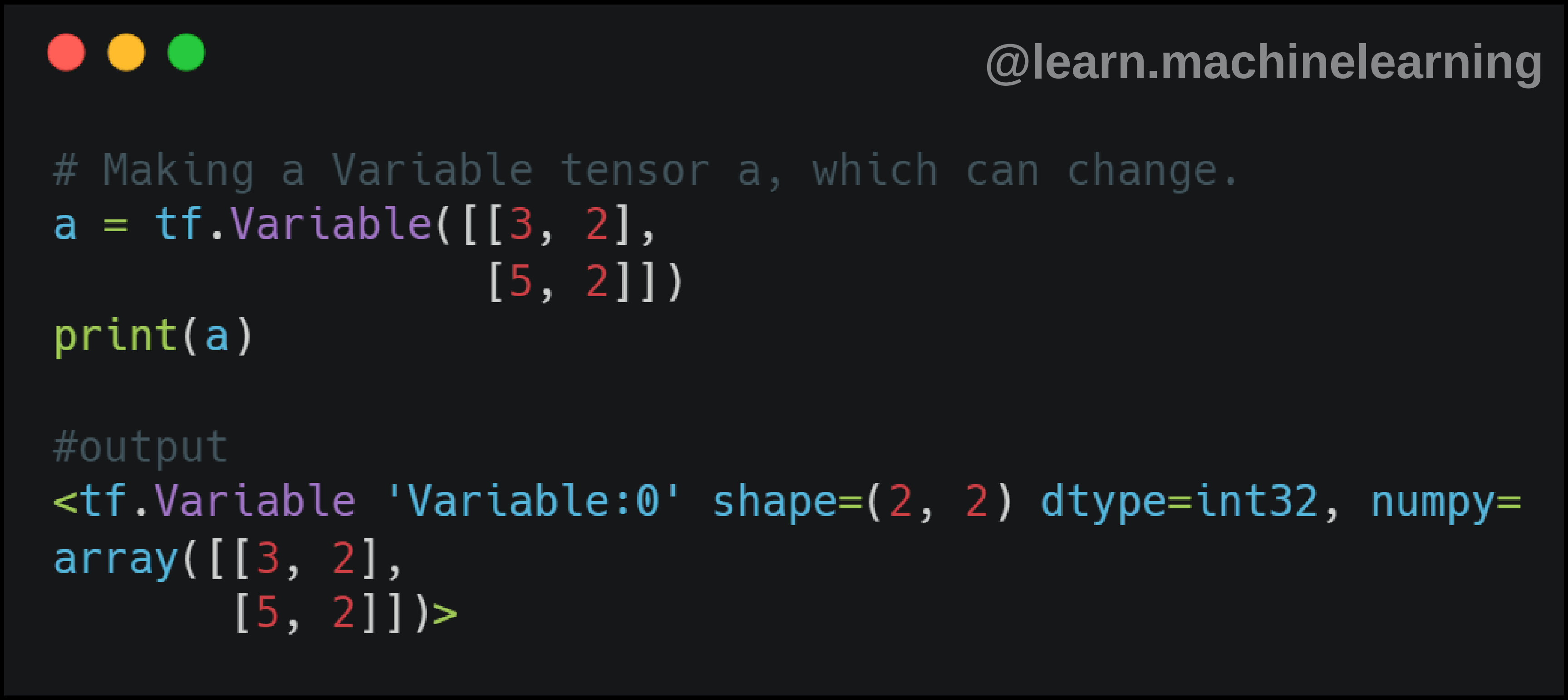
tf.Variable

@learn.machinelearning

```
tf.Variable( initial_value=None, trainable=None, validate_shape= True,
caching_device= None,    name=None, variable_def=None, dtype=
None, import_scope=None, constraint=None, synchronization=
tf.VariableSynchronization.AUTO, shape=None, aggregation=
tf.compat.v1.VariableAggregation.NONE)
```

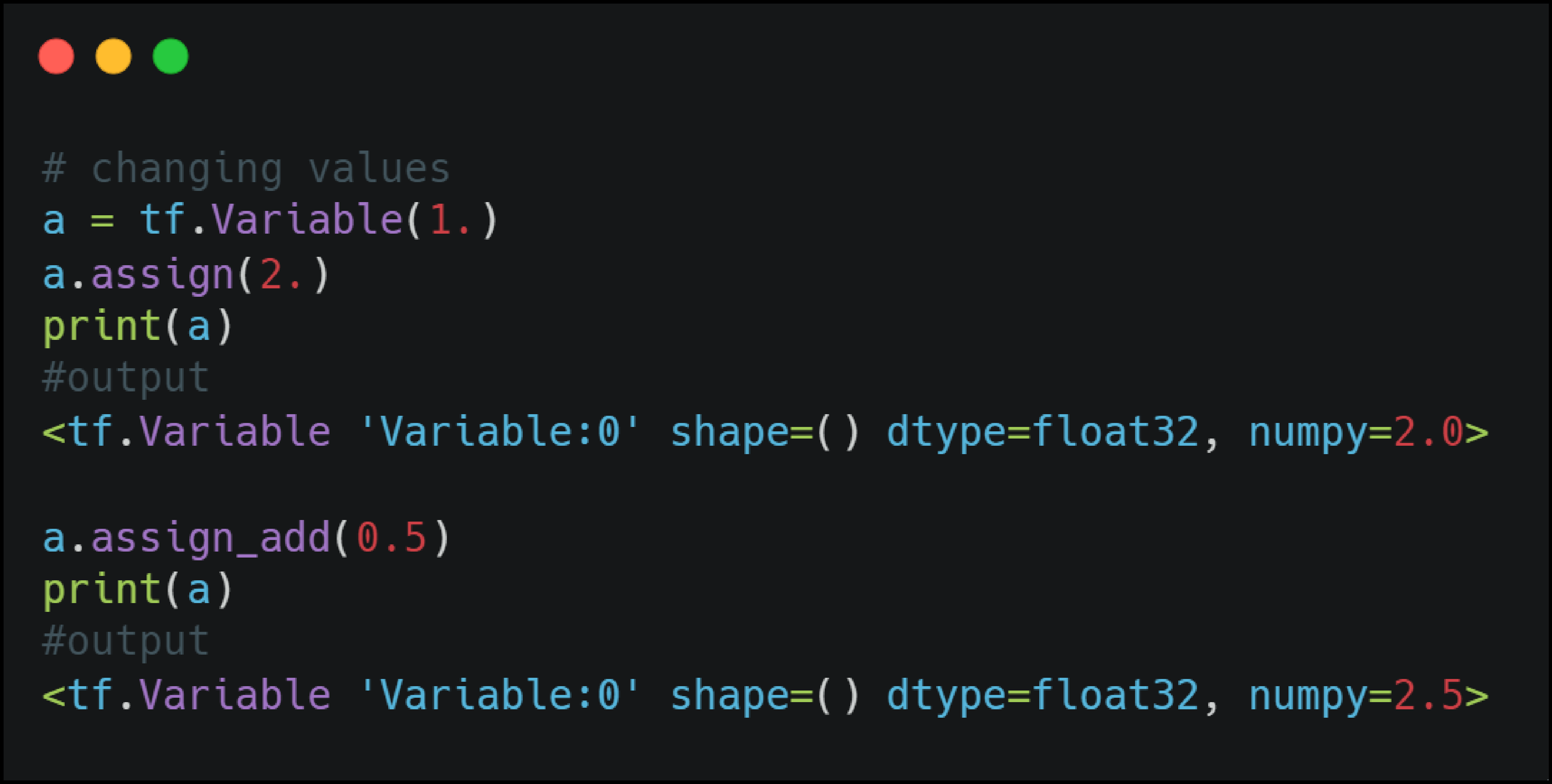
- A variable maintains shared, persistent state manipulated by a program.
- The Variable() constructor requires an initial value for the variable, which can be a Tensor of any type and shape. This initial value defines the type and shape of the variable. After construction, the type and shape of the variable are fixed. The value can be changed using one of the assign methods.

tf.Variable



```
# Making a Variable tensor a, which can change.
a = tf.Variable([[3, 2],
                 [5, 2]])
print(a)

#output
<tf.Variable 'Variable:0' shape=(2, 2) dtype=int32, numpy=
array([[3, 2],
       [5, 2]])>
```



```
# changing values
a = tf.Variable(1.)
a.assign(2.)
print(a)
#output
<tf.Variable 'Variable:0' shape=() dtype=float32, numpy=2.0>

a.assign_add(0.5)
print(a)
#output
<tf.Variable 'Variable:0' shape=() dtype=float32, numpy=2.5>
```

tf.Zeros and tf.ones

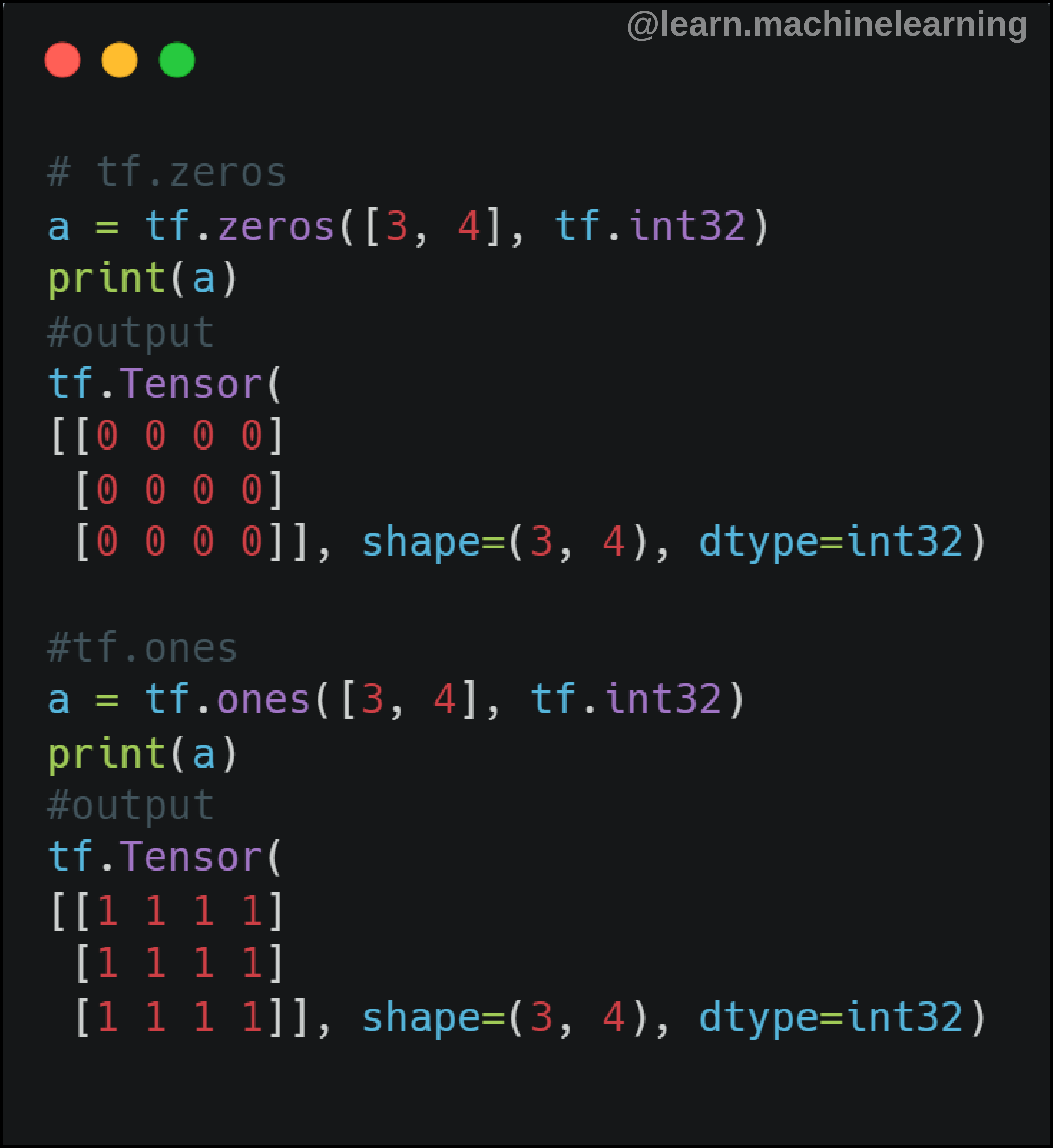
@learn.machinelearning

- Creating tensors with just tf.constant and tf.Variable can be tedious if you want to create big tensors. Imagine you want to create random noise – well, you could do that by making a tensor with tf.zeros or tf.ones.
- All we need to specify is the shape in the format shape=[rows, columns] and a dtype, if it matters at all. The number of rows and columns are arbitrary, and you could in principle create 4K images (as noise).

```
tf.zeros(shape, dtype=tf.dtypes.float32, name=None)
```

```
tf.ones(shape, dtype=tf.dtypes.float32, name=None)
```

tf.Zeros and tf.ones



```
# tf.zeros
a = tf.zeros([3, 4], tf.int32)
print(a)
#output
tf.Tensor(
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]], shape=(3, 4), dtype=int32)

#tf.ones
a = tf.ones([3, 4], tf.int32)
print(a)
#output
tf.Tensor(
[[1 1 1 1]
 [1 1 1 1]
 [1 1 1 1]], shape=(3, 4), dtype=int32)
```