

Lifecycle of a Node Value

When you evaluate a node, TensorFlow automatically determines the set of nodes that it depends on and it evaluates these nodes first. For example, consider the following code:

```
w = tf.constant(3)
x = w + 2
y = x + 5
z = x * 3

with tf.Session() as sess:
    print(y.eval()) # 10
    print(z.eval()) # 15
```

First, this code defines a very simple graph. Then it starts a session and runs the graph to evaluate `y`: TensorFlow automatically detects that `y` depends on `w`, which depends on `x`, so it first evaluates `w`, then `x`, then `y`, and returns the value of `y`. Finally, the code runs the graph to evaluate `z`. Once again, TensorFlow detects that it must first evaluate `w` and `x`. It is important to note that it will *not* reuse the result of the previous evaluation of `w` and `x`. In short, the preceding code evaluates `w` and `x` twice.

All node values are dropped between graph runs, except variable values, which are maintained by the session across graph runs (queues and readers also maintain some state, as we will see in [Chapter 12](#)). A variable starts its life when its initializer is run, and it ends when the session is closed.

If you want to evaluate `y` and `z` efficiently, without evaluating `w` and `x` twice as in the previous code, you must ask TensorFlow to evaluate both `y` and `z` in just one graph run, as shown in the following code:

```
with tf.Session() as sess:
    y_val, z_val = sess.run([y, z])
    print(y_val) # 10
    print(z_val) # 15
```



In single-process TensorFlow, multiple sessions do not share any state, even if they reuse the same graph (each session would have its own copy of every variable). In distributed TensorFlow (see [Chapter 12](#)), variable state is stored on the servers, not in the sessions, so multiple sessions can share the same variables.

Linear Regression with TensorFlow

TensorFlow operations (also called *ops* for short) can take any number of inputs and produce any number of outputs. For example, the addition and multiplication ops each take two inputs and produce one output. Constants and variables take no input