understanding of the underlying learning algorithms are unlikely to work well. In the rest of this section, we will start paring back the abstraction, a process we will continue throughout the rest of the book.

TensorFlow Eager

The TensorFlow team recently added a new experimental module, TensorFlow Eager, that enables users to run TensorFlow calculations imperatively. In time, this module will likely become the preferred entry mode for new programmers learning TensorFlow. However, at the timing of writing, this module is still very new with many rough edges. As a result, we won't teach you about Eager mode, but encourage you to check it out for yourself.

It's important to emphasize that much of TensorFlow will remain declarative even after Eager matures, so it's worth learning declarative TensorFlow regardless.

TensorFlow Graphs

Any computation in TensorFlow is represented as an instance of a `tf.Graph` object. Such a graph consists of a set of instances of `tf.Tensor` objects and `tf.Operation` objects. We have covered `tf.Tensor` in some detail, but what are `tf.Operation` objects? You have already seen them over the course of this chapter. A call to an operation like `tf.matmul` creates a `tf.Operation` instance to mark the need to perform the matrix multiplication operation.

When a `tf.Graph` is not explicitly specified, TensorFlow adds tensors and operations to a hidden global `tf.Graph` instance. This instance can be fetched by `tf.get_default_graph()` (Example 2-22).

Example 2-22. Getting the default TensorFlow graph

```
>>> tf.get_default_graph()
<tensorflow.python.framework.ops.Graph>
```

It is possible to specify that TensorFlow operations should be performed in graphs other than the default. We will demonstrate examples of this in future chapters.

TensorFlow Sessions

In TensorFlow, a `tf.Session()` object stores the context under which a computation is performed. At the beginning of this chapter, we used `tf.InteractiveSession()` to set up an environment for all TensorFlow computations. This call created a hidden global context for all computations performed. We then used `tf.Tensor.eval()` to