Introduction to Broadcasting

Broadcasting is a term (introduced by NumPy) for when a tensor system's matrices and vectors of different sizes can be added together. These rules allow for conveniences like adding a vector to every row of a matrix. Broadcasting rules can be quite complex, so we will not dive into a formal discussion of the rules. It's often easier to experiment and see how the broadcasting works (Example 2-18).

Example 2-18. Examples of broadcasting

```
>>> a = tf.ones((2, 2))
>>> a.eval()
array([[ 1.,  1.],
       [ 1.,  1.]], dtype=float32)
>>> b = tf.range(0, 2, 1, dtype=tf.float32)
>>> b.eval()
array([ 0.,  1.], dtype=float32)
>>> c = a + b
>>> c.eval()
array([[ 1.,  2.],
       [ 1.,  2.]], dtype=float32)
```

Notice that the vector `b` is added to every row of matrix `a`. Notice another subtlety; we explicitly set the `dtype` for `b`. If the `dtype` isn't set, TensorFlow will report a type error. Let's see what would have happened if we hadn't set the `dtype` (Example 2-19).

Example 2-19. TensorFlow doesn't perform implicit type casting

```
>>> b = tf.range(0, 2, 1)
>>> b.eval()
array([0, 1], dtype=int32)
>>> c = a + b
ValueError: Tensor conversion requested dtype float32 for Tensor with dtype int32:
'Tensor("range_2:0", shape=(2,), dtype=int32)
```

Unlike languages like C, TensorFlow doesn't perform implicit type casting under the hood. It's often necessary to perform explicit type casts when doing arithmetic operations.

Imperative and Declarative Programming

Most situations in computer science involve imperative programming. Consider a simple Python program (Example 2-20).