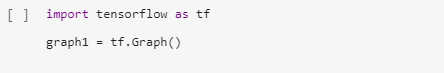
Worksheet 1

Import a module or library

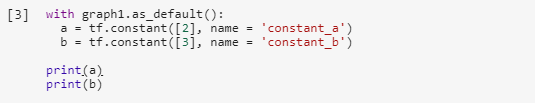
As we said before, TensorFlow works as a graph computational model. Let's create our first graph which we named as **graph1**.



Create nodes and tensors

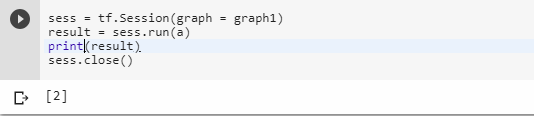
Now lets create a node. First, add 2 constants to our graph. Calling functions, **tf.constant([value], name=’constant\_name’).** By calling this function, it constructs new **tf.Operation** and **tf.Tensor** objects and add them to the graph. As mentioned, each **tf.Operation** is a node and each **tf.Tensor** is an edge of the graph.

**Notice:**  tf.constant([2], name="constant\_a") creates a new tf.Operation named "constant\_a" and returns a reference of that value, **tf.Tensor,** named "constant\_a:0".



Creating a Session

By just printing out the variables **a** and **b ,** from above, they only returns reference of the node or tensors. To see the value of each nodes, we have to create a Tensorflow session.



### Defining multidimensional arrays using Tensorflow

As we briefly said before, a zero dimension can be seen as a point, a single object or a single item.

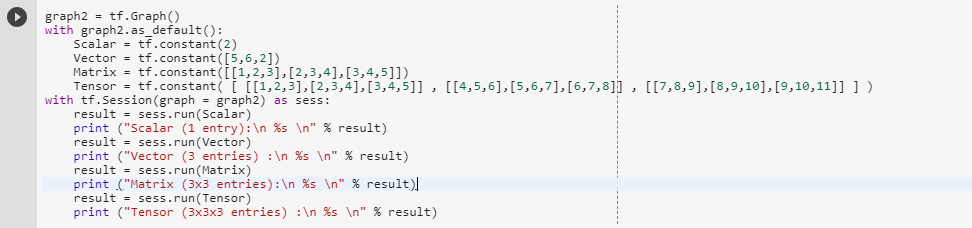
The First dimension as a line or One dimensional array as numbers along this line.

The Second dimension or two dimensions can be seen as a surface or a matrix.

The Third dimension can be seen as a volume.

The Fourth dimension can be seen as the hyperspace or spacetime, a volume varying through time, or an infinite series of volumes along an infinite line.

Okay, now let’s create multidimensional arrays in tensorflow.



You will now get the idea of multidimensional arrays in tensorflow and whenever you want to create a node, it has to be Created inside **the graph** and Run it on **the session** to see the value**.**

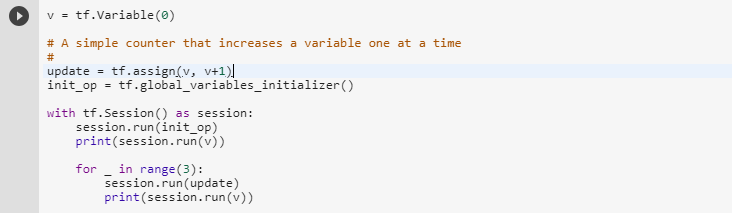
### Variables

Question: Why do we need to use variables when we have tensors?

Answer: Tensors cannot be updated during each run (which is created using **tf.constant()** ) and

When a variable is created, TensorFlow adds **tf.Operation** to your graph (same as creating nodes). This operation stores a writable tensor value that persists between **tf.Session.run** calls.

Okay, let’s look at it.



You must initialize all global variables before using them by the function, **tf.global\_variables\_initializer( )**

You can update your variable by using **tf.assign(value, value\_to\_update).**

### Placeholders

After working with variables to some extent, lets try feeding data outside of a tensorflow graph.

We will use placeholders to feed data to the graph from the outside of the graph.

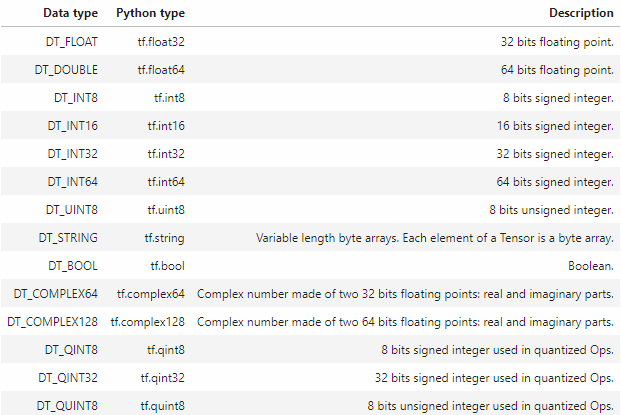
What are placeholders and what do they do?

Placeholders can be seen as “holes” in your model.

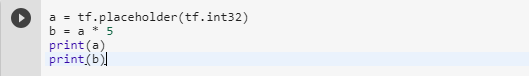
“Holes” to which you will pass the data and can create them by using **tf.placeholder(datatype)**.

**Datatype = integers, floating, points, strings, Booleans or 8, 16, 32, 64 bits.**

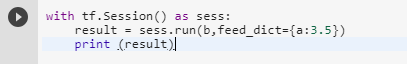
Below are the data types in python.



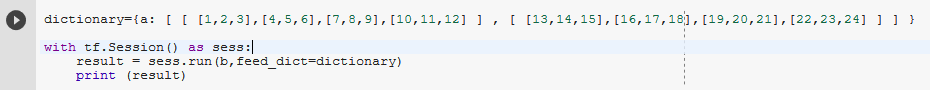
Now we will define placeholder by using **tf.placeholder()** and do a multiplication.



Then, feed that data to the graph.



You can create a multidimensional dictionary and feed into the graph like this.

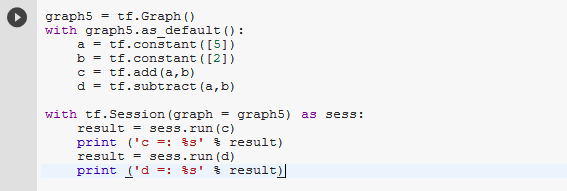


### Operation

Operations are nodes that represent the mathematical operations over the tensor on a graph. These operations can be any kind of functions, like add and subtract tensor or maybe an activation function.

**tf.constant, tf.matmul, tf.add, tf.nn.signmoid** are some of the operations in Tensorflow. These are like functions in python but operate directly over tensors and each one does a specific thing.

Note: **tf.nn.sigmoid** is an activation function, it’s a little more complicated, but this function helps learning models to evaluate what kind of information is good or not.



More operations can be found at: <https://www.tensorflow.org/versions/r0.9/api_docs/python/index.html>