Getting Started with TensorFlow Summary
Source: https://www.tensorflow.org/get_started/get_started

Tensors are the central unit of data in TensorFlow. They consist of a set of primitive values shaped into an array of any number of dimensions (matrices). You must first construct your computational graph. A computational graph is a series of TensorFlow operations arranged into a graph of nodes. These nodes take zero or more tensors as inputs and produces a tensor as an output. If your node is a constant (tf.constant), it does not take any inputs and it outputs the constant value stored inside of it when the computational graph is ran within a session. If your node is an operation, it can take Tensor nodes as inputs and perform operations on them.
Inputs can also be placeholders, which promise to provide values later when declared within an operation (tf.placeholder(tf.float32)).
Variables are declared in TensorFlow with initial value and its type such as:
W = tf.Variable([.3], tf.float32). Variables must be initialized before you call a session run unlike constants. You can use tf.assign to change the value provided to tf.variable.
The loss value can be computed directly within the session run parameter.
Using optimizers in TensorFlow is easy, specifically gradient descent (use optimizer = tf.train.GradientDescentOptimizer(stepSize) function and then call train = optimizer.minimize(loss) and then iterate your sess.run(train, x, y) and at this point you can retrieve your [W,b].

Question 2: Reporting the Learned Regression Function

For this question, my reported values were as follows:

```
Training Loss:  {'loss': 0.085015491, 'global_step': 2000}
Testing Loss:  {'loss': 0.086240709, 'global_step': 2000}
w1:  [[ 1.99998057]]
w2:  [[-2.99982572]]
b:  [ 0.71055716]
```

Notice that both my training and testing loss are below 0.1.
For my input_fn and input_fn_eval, I chose my batch_size to be 150 and num_epochs to be 2000.
To fit my model, I chose my steps to be 2000.
I found these parameter values to be the most efficient for my loss values.

MNIST For ML Beginners Summary
Source: https://www.tensorflow.org/get_started/mnist/beginners

MNIST is a simple computer vision dataset, consisting of images that belong to labels/classifications.
In the example given in this section, mnist.train.images is a tensor with the shape [55000, 784] where the first dimension is an index into the list of images and the second dimension is the index for each pixel in each image. It is 784 because each image is 28x28 and they are flattened into a vector with 28x28=784 numbers.
mnist.train.labels is a [55000, 10] array of floats where the first index is the amount of labels for each corresponding image and the second index is the label 0-9.
We use a softmax regression model for this problem, which has two steps: first, we add up the evidence of our input being in certain classes, and then we convert that evidence into probabilities. For adding up the evidence of our input, we do a weighted sum of the pixel intensities: it is positive if it is evidence in favor and negative otherwise.

Modifications to mnist_softmax.py:

Changed step size for GradientDescentOptimizer(0.65)
Changed loop range under #Train to (5500)
Changed mnist.train.next_batch size to (400)

Output:

```
(tensorflow) eduroam-169-233-193-96:Supplemental_Material ricardomunoz$
python mnist_softmax.py
Extracting /tmp/tensorflow/mnist/input_data/train-images-idx3-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/train-labels-idx1-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/t10k-images-idx3-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/t10k-labels-idx1-ubyte.gz
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow
library wasn't compiled to use SSE4.1 instructions, but these are
available on your machine and could speed up CPU computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow
library wasn't compiled to use SSE4.2 instructions, but these are
available on your machine and could speed up CPU computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow
library wasn't compiled to use AVX instructions, but these are available
on your machine and could speed up CPU computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow
library wasn't compiled to use AVX2 instructions, but these are available
on your machine and could speed up CPU computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow
library wasn't compiled to use FMA instructions, but these are available
on your machine and could speed up CPU computations.
0.926
```

TensorFlow Mechanics 101
Source: https://www.tensorflow.org/get_started/mnist/mechanics

We must first prepare the data of the MNIST, a classic problem in machine learning. Each 28x28 pixel image of handwritten digits determine which digit the image represents. The placeholder_inputs() creates tf.placeholder operations that define the shape of the inputs, as well as the batch_size to the rest of the graph and into which the actual training examples will be fed into.
Build the graph: interference() builds the graph as far as is required for running the network forward to make predictions, loss() adds to the interference graph the operations required to generate loss and training() adds to the loss graph the operations required to compute and apply gradients. What I found useful in this tutorial is the Visualize the Status section. When the events files are updated and written, TensorBoard may be run against the training folder to display the values from the summaries (tf.summary.FileWriter). Also, in the Evaluate the Model section, I learned that during the thousands of steps, the code attempts to evaluate the model against both the training and test datasets. Specifically, the do_eval() function is called three times: for training, validation, and test datasets.