# Getting started with TensorFlow Tensors

The central unit of data in TensorFlow is the **tensor**. A tensor consists of a set of primitive values shaped into an array of any number of dimensions. A tensor's **rank** is its number of dimensions. Here are some examples of tensors:

3 # a rank 0 tensor; this is a scalar with shape []  
[1., 2., 3.] # a rank 1 tensor; this is a vector with shape [3]  
[[1., 2., 3.], [4., 5., 6.]] # a rank 2 tensor; a matrix with shape [2, 3]  
[[[1., 2., 3.]], [[7., 8., 9.]]] # a rank 3 tensor with shape [2, 1, 3]

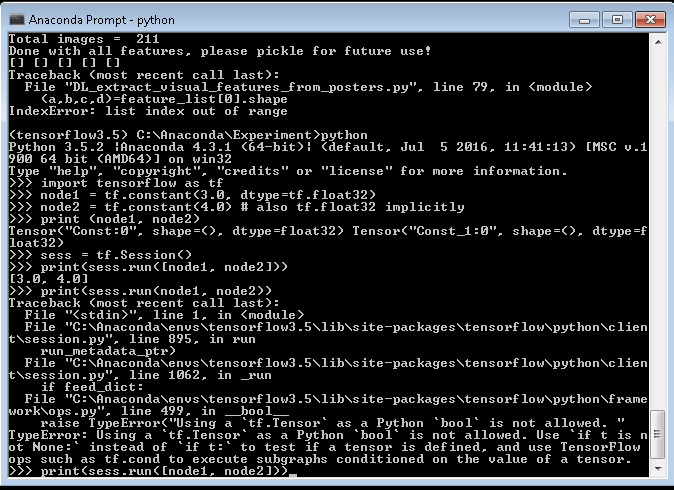
You might think of TensorFlow Core programs as consisting of two discrete sections:

1. Building the computational graph.
2. Running the computational graph.

import tensorflow as tf

node1 = tf.constant(3.0, dtype=tf.float32)  
node2 = tf.constant(4.0) # also tf.float32 implicitly  
print(node1, node2)

sess = tf.Session()  
print(sess.run([node1, node2]))

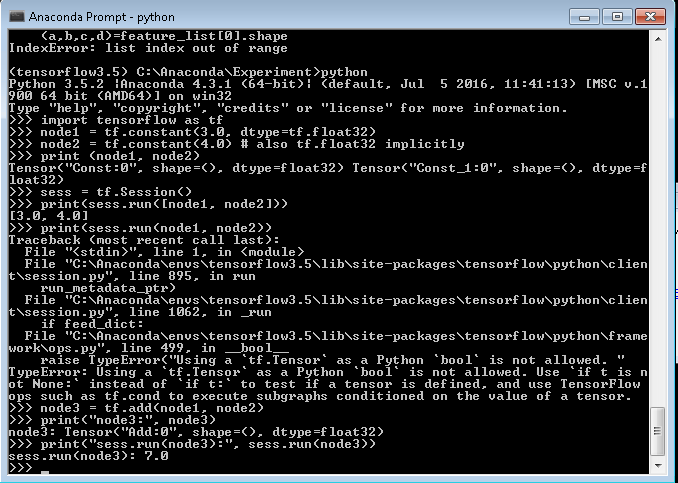


import tensorflow as tf

node1 = tf.constant(3.0, dtype=tf.float32)  
node2 = tf.constant(4.0) # also tf.float32 implicitly  
print(node1, node2)

sess = tf.Session()  
print(sess.run([node1, node2]))

node3 = tf.add(node1, node2)  
print("node3:", node3)  
print("sess.run(node3):", sess.run(node3))



import tensorflow as tf

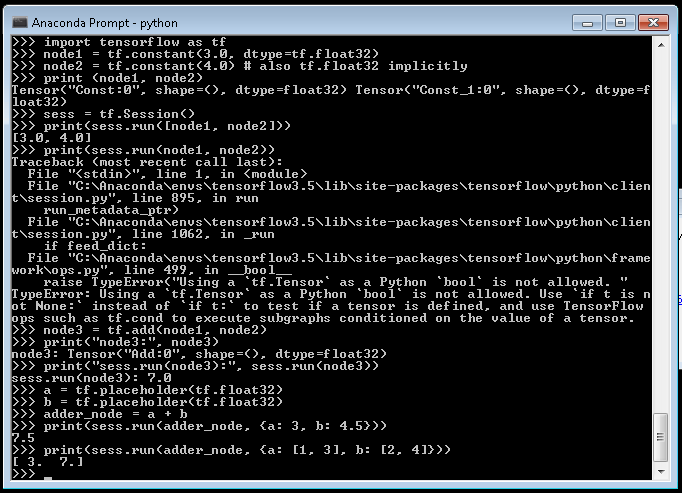
node1 = tf.constant(3.0, dtype=tf.float32)  
node2 = tf.constant(4.0) # also tf.float32 implicitly  
print(node1, node2)

sess = tf.Session()  
print(sess.run([node1, node2]))

node3 = tf.add(node1, node2)  
print("node3:", node3)  
print("sess.run(node3):", sess.run(node3))

a = tf.placeholder(tf.float32)  
b = tf.placeholder(tf.float32)  
adder\_node = a + b  # + provides a shortcut for tf.add(a, b)

print(sess.run(adder\_node, {a: 3, b: 4.5}))  
print(sess.run(adder\_node, {a: [1, 3], b: [2, 4]}))



import tensorflow as tf

node1 = tf.constant(3.0, dtype=tf.float32)  
node2 = tf.constant(4.0) # also tf.float32 implicitly  
print(node1, node2)

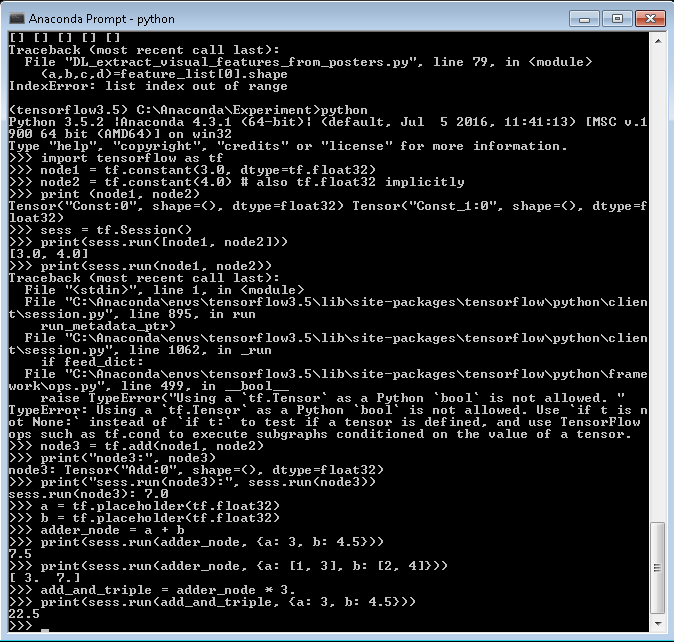
sess = tf.Session()  
print(sess.run([node1, node2]))

node3 = tf.add(node1, node2)  
print("node3:", node3)  
print("sess.run(node3):", sess.run(node3))

a = tf.placeholder(tf.float32)  
b = tf.placeholder(tf.float32)  
adder\_node = a + b  # + provides a shortcut for tf.add(a, b)

print(sess.run(adder\_node, {a: 3, b: 4.5}))  
print(sess.run(adder\_node, {a: [1, 3], b: [2, 4]}))

add\_and\_triple = adder\_node \* 3.  
print(sess.run(add\_and\_triple, {a: 3, b: 4.5}))

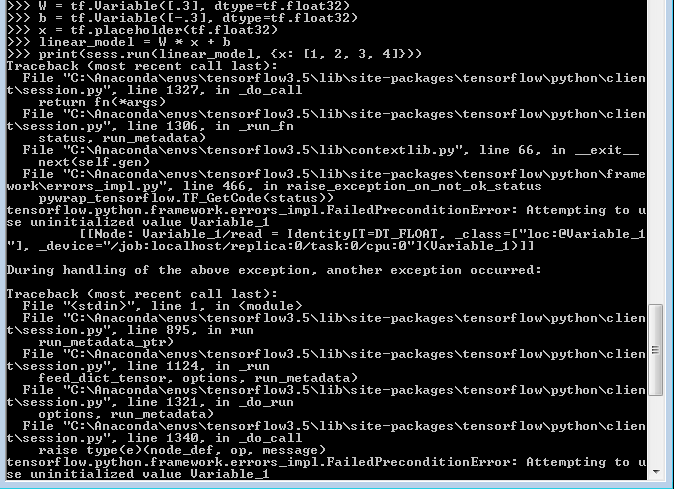


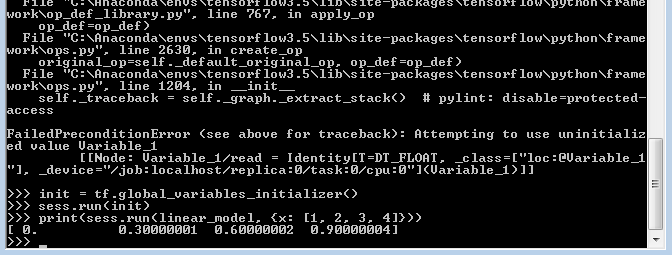
Machine Learning (Weight, Linear Regression)

W = tf.Variable([.3], dtype=tf.float32)  
b = tf.Variable([-.3], dtype=tf.float32)  
x = tf.placeholder(tf.float32)  
linear\_model = W \* x + b

init = tf.global\_variables\_initializer()  
sess.run(init)

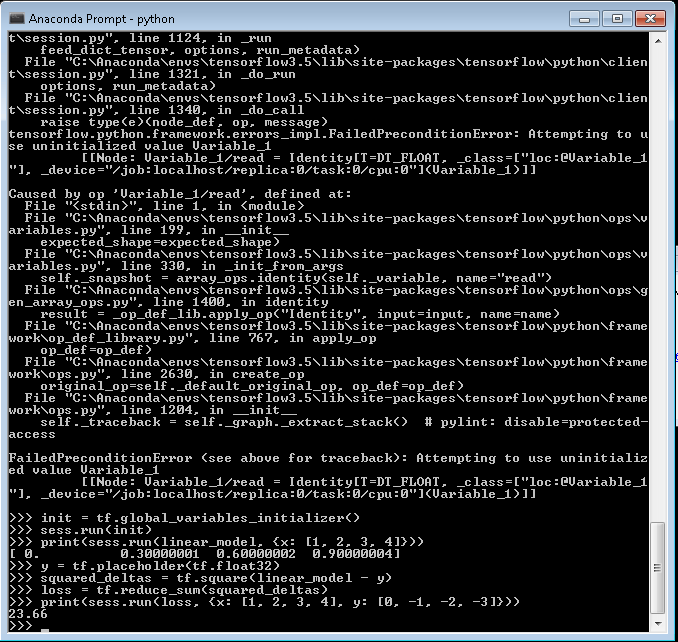
print(sess.run(linear\_model, {x: [1, 2, 3, 4]}))



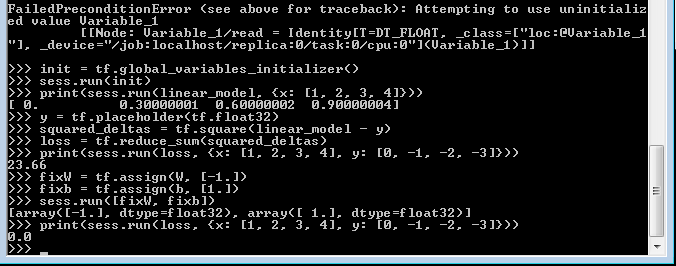


Loss Function

y = tf.placeholder(tf.float32)  
squared\_deltas = tf.square(linear\_model - y)  
loss = tf.reduce\_sum(squared\_deltas)  
print(sess.run(loss, {x: [1, 2, 3, 4], y: [0, -1, -2, -3]}))



fixW = tf.assign(W, [-1.])  
fixb = tf.assign(b, [1.])  
sess.run([fixW, fixb])  
print(sess.run(loss, {x: [1, 2, 3, 4], y: [0, -1, -2, -3]}))



Optimization

optimizer = tf.train.GradientDescentOptimizer(0.01)  
train = optimizer.minimize(loss)

sess.run(init) # reset values to incorrect defaults.  
for i in range(1000):  
  sess.run(train, {x: [1, 2, 3, 4], y: [0, -1, -2, -3]})  
  
print(sess.run([W, b]))

