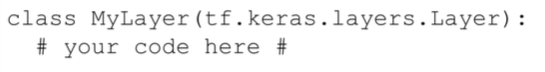
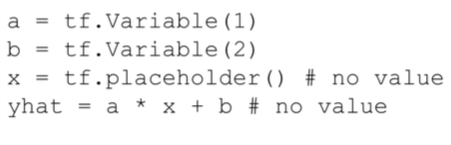
1. Differences between Tensorflow 1.x and 2.x

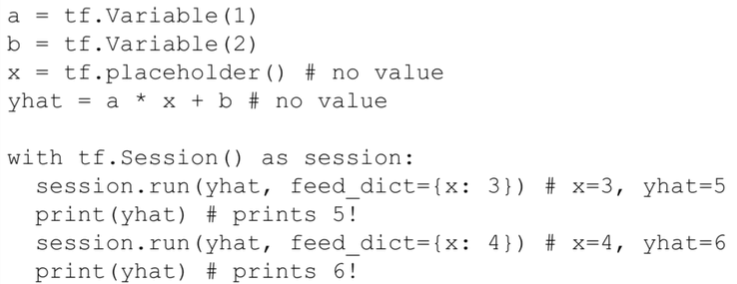
* The Keras API – same from user-side
* Keras is the standard for Tensorflow 2.0
* In Tensorflow 1.x, you could use a convolution layer from
* tf.layers.conv2d
* tf.contrib.layers.conv2d
* etc.
* No more tf.contrib
* Building custom layers
* You can still build your own
* Just conform to the Keras API using subclassing



* Eager execution
* Eager execution is on by default – no more sessions
* Now we don’t have to wait for ‘in the future’ – just a fancy way of saying it’ll work like regular Numpy / Python (and PyTorch)
* During the various iterations of Tensorflow 1.x, a Googler implemented eager execution so TF would work more like PyTorch – it is available in later versions >< had to turn this in on yourself
* In TF 2.0, eager is on by default
* Tensorflow 1.x:
* When you say ‘c = a + b’ in Tensorflow 1.x, you are defining a computation graph
* You are more computing the value of x now, you are telling Tensorflow how to compute it in the future
* Makes more sense in the context of placeholders
* Placeholders



* Sessions
  + You use a session to give the placeholders real values, so that you can calculate other values in the graph



* Efficiency:
* Why was Tensorflow the way it was before?
* Theano – the original GPU-enabled deep learning library for Python
* You would compile the graph, and that graph would be fast
* This is where Keras’ model.compile() comes from
* Without sessions / graphs, Tensorflow is less efficient
* The solution in TF2.0 is the @tf.function decorator
* Summary:
* Keras API is now the standard
* Other layers gone
* Tf.contrib is gone
* To create your own layers and models, subclass the Keras layer/model
* Sessions are gone
* Eager execution is enabled

1. Constants and Basic computation
2. Variables and Gradient Tape
3. Build your own custom model