AIDL: PROJECTS

Session 2 (2019/05/07): Deep dive into Tensorflow

Instructors

Issey Masuda Mora

Santi Puch Giner

Projects

Specs

Evaluation:

- Project report: 40%
- Presentation: 30%
- Advisor qualification: 30%

Requirements:

- Program a model
- Train a model
- Inference pipeline up & running! (even if the results are bull****)

Classes

Note on previous labs

- This will NOT assume any tech skill learned from them
- Starting from scratch
- Forget about them (during these sessions :P)

In-class exercises

- Not tests
- Meant to force you to think & apply what has been taught
- Difficult to finish in-class. Finish them at home if you want to get all the juice out of it
- Accumulative
- Not a single solution

A possible solution will be pushed to the repo after class

Personal motivation: why am I here?

- I hated all the way through college
- Dislike current education system
- Refactor my own skills
- You should be able to pass a DL interview!

Can I do something BETTER?

Personal motivation: edu system

XIV century

XXI century

Personal motivation: edu system

How to get there:

- Provide feedback
- Ask questions

The DL engineer quadrant

Programmers that understand DL as lego blocks

\+

Implementation

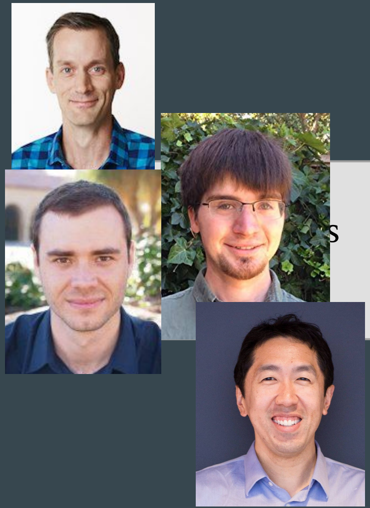
Follow the gradient!

Theoretical

\-

\+

\-



Random people

The

PhD

The 5 minute recap

- The content of each class will be based on all the previous lessons
- 5 min recap at the beginning of each class

Previously on *AIDL: Projects*...

“Give them the tools to apply DL in the industry”

Subject goals

torch

Keras

theano

DEEPLEARNING4J

Caffe2

TensorLayer

TensorFlow

PYTORCH

Chainer

Lasagne

Caffe

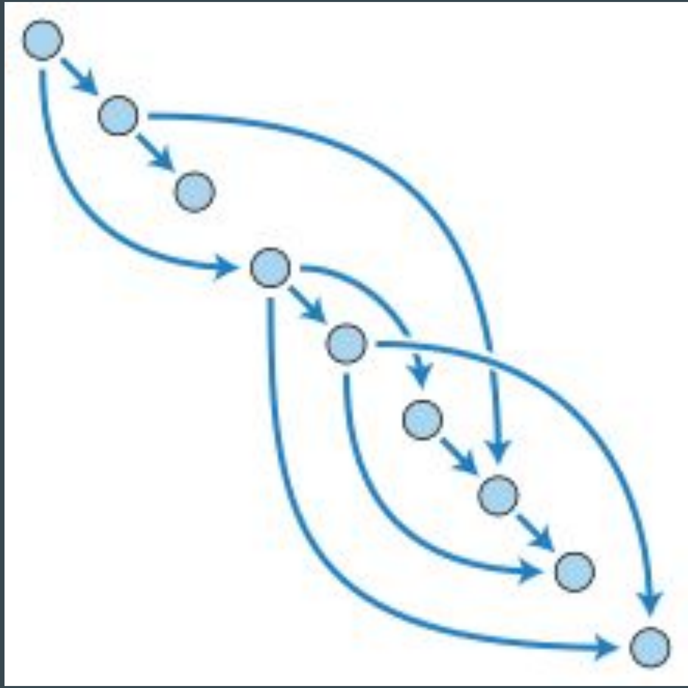
Microsoft CNTK

Sonnet

mxnet

Directed Acyclic Graph (DAG)

- Edged are directed from one node to another
- Nodes have a topological ordering
- No closed loops!



Computational graph

Static computational graph: Define-and-Run methodology.

Pros:

- Speed: can spend a long time optimizing the graph
- Memory: can predict and allocate all mem ahead of time

Cons:

- Inflexible: once compiled, it cannot be modified at run time
- Debugging: graph representation doesn't match code
- Static: more difficult to support flexible sized inputs

Dynamic computational graph: Define-by-Run strategy

Pros:

- Can change structure of NN at runtime (add layers, change shape, etc.)
- Support flexible sized inputs
- Easier to debug
- More natural to code

Cons:

- Compile at runtime, can be slower
- Dynamic batching more difficult

Surprise test!

Learning

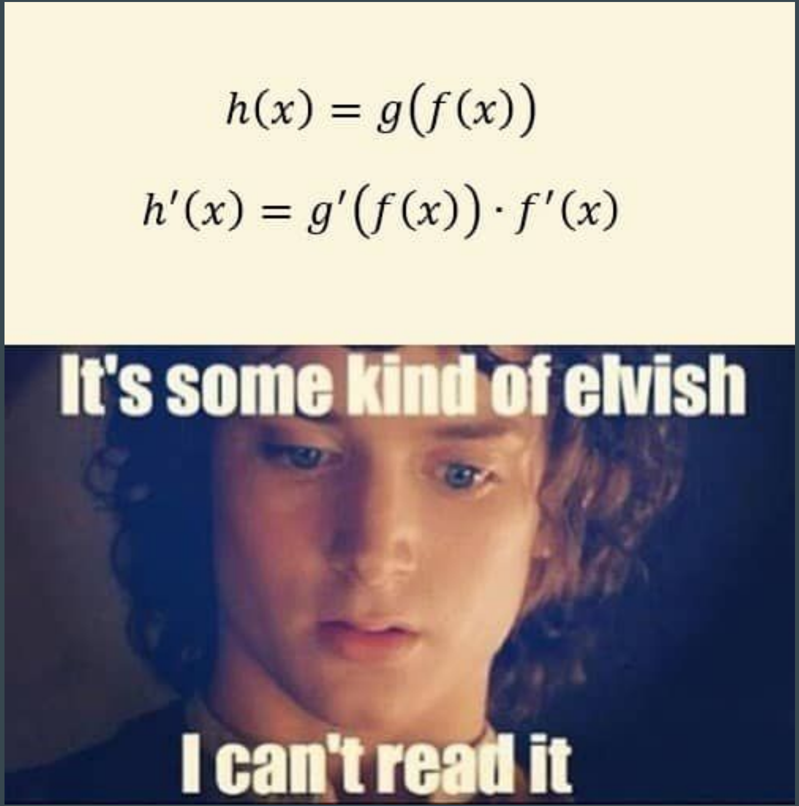
Learning step:

1. Forward pass the inputs through the model
2. Compute loss / objective function
3. Use optimization algorithm to update trainable variables:
    a. Compute the gradients using BACKPROPAGATION
    b. Use the gradients to update the model

Example: Linear regression

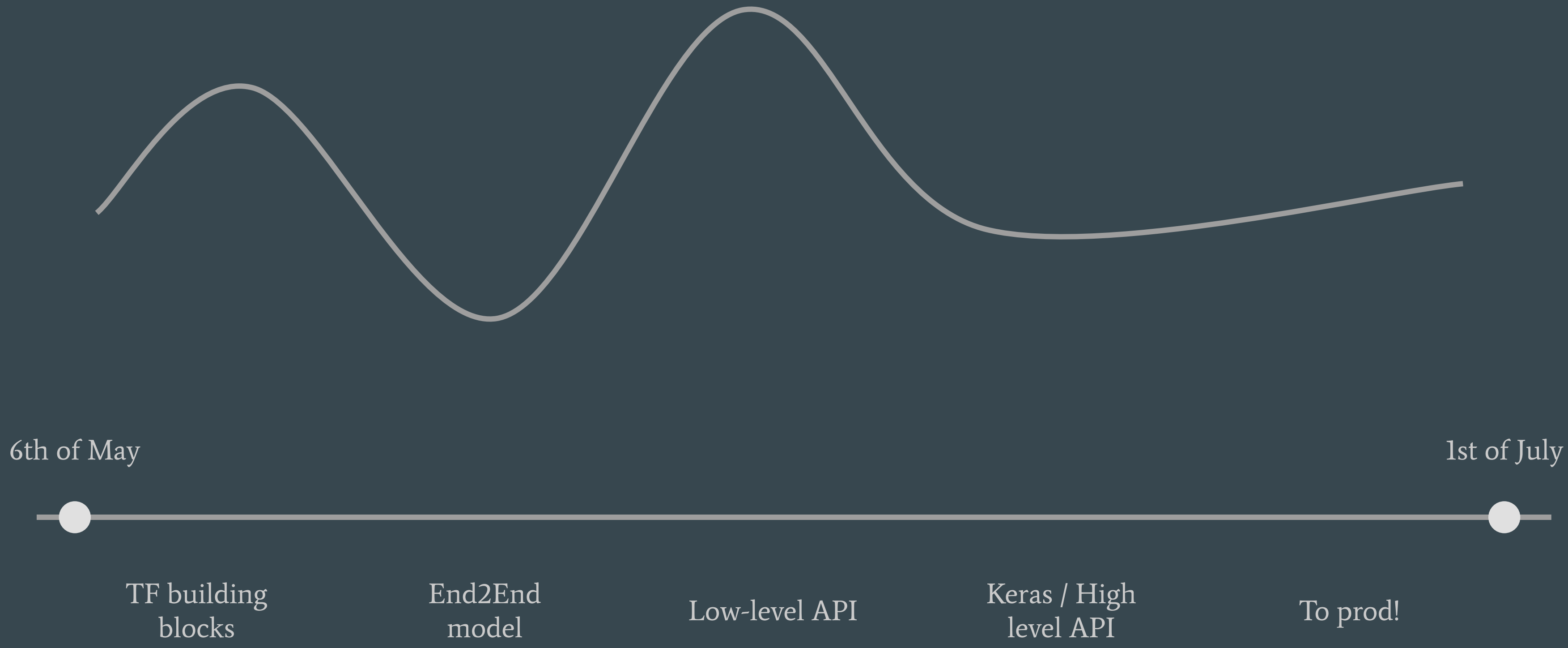
Understanding backprop

- Key ingredient: chain rule
- Local derivatives do not depend on final loss
- **The local derivatives tell us what is going to be held in memory!**



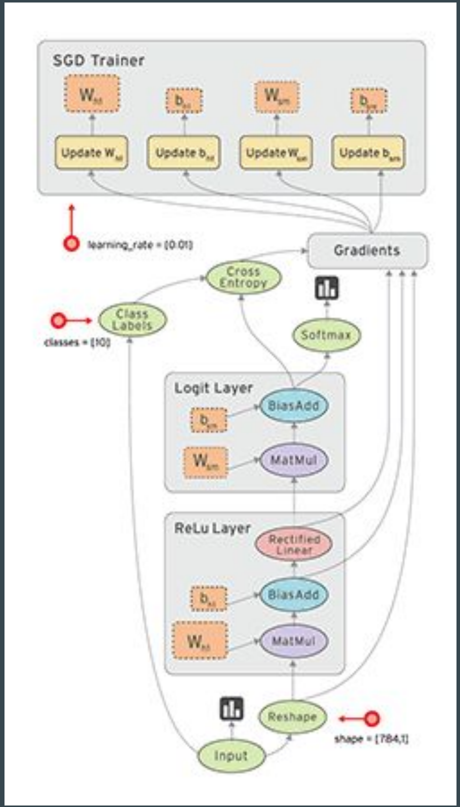
Tensorflow: a ~~gentle~~ introduction

Tensorflow


6th of May
1st of July
TF building blocks
End2End model
Low-level API
Keras / High level API
To prod!

It’s all about graphs

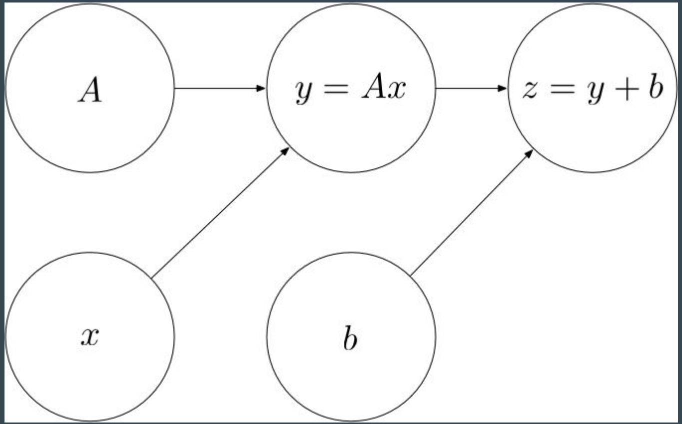
Tensorflow is a symbolic computational library that uses **static computational graphs** (define-and-run) to represent the models



Computational graph & Dataflow

Programs are represented as directed graphs with data flowing through them where:

- **Nodes**: Operations of the program → **tf.Operation**
- **Edges**: Data flowing through the graph → **tf.Tensor**

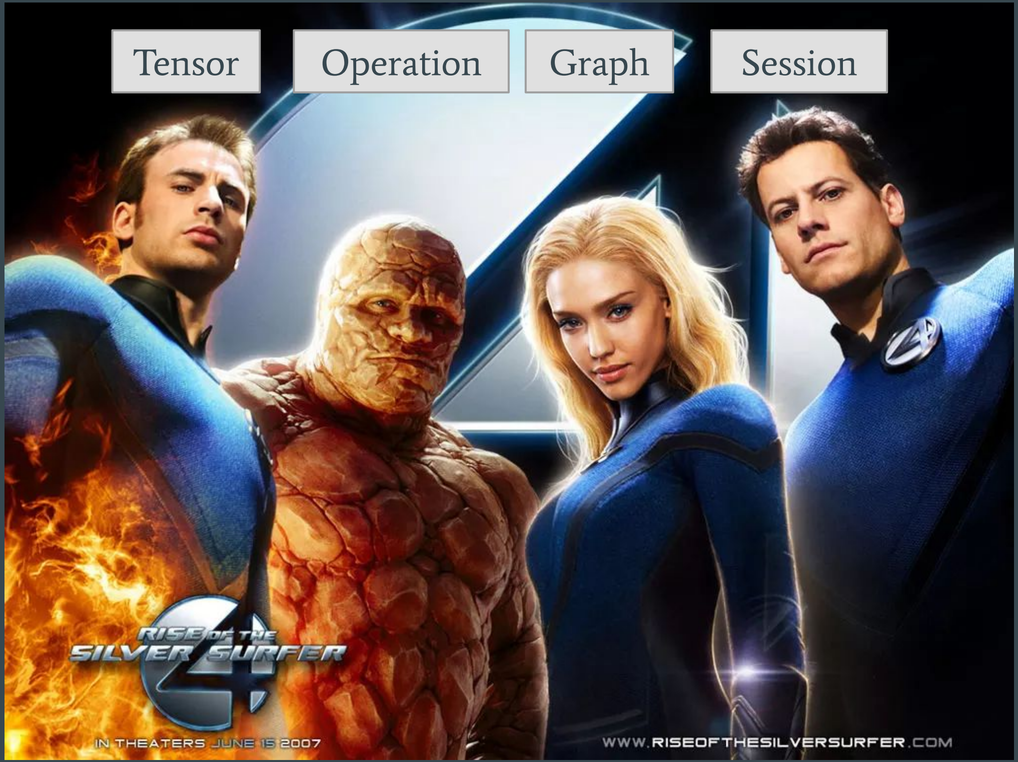


From symbolic to real numbers

Two phases:

- **Definition phase**: build the computational graph → **tf.Graph**
- **Execution phase**: interact with the graph feeding data and fetching results.
  Run a subgraph of the original graph & change its state → **tf.Session**

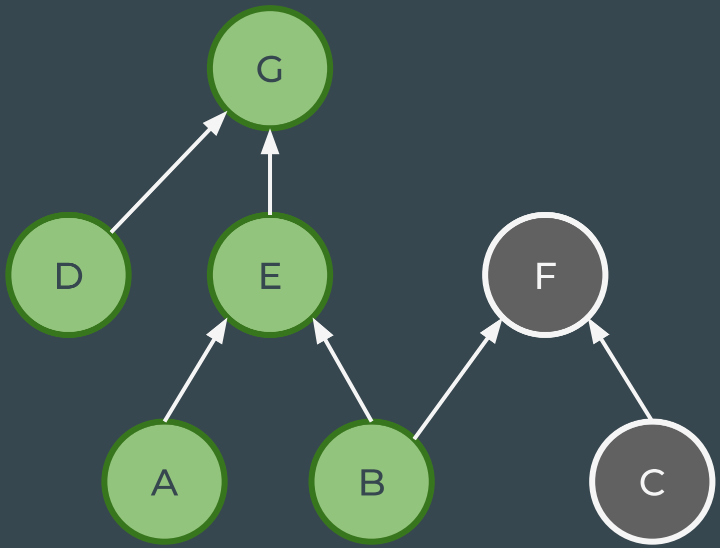
Fantastic four



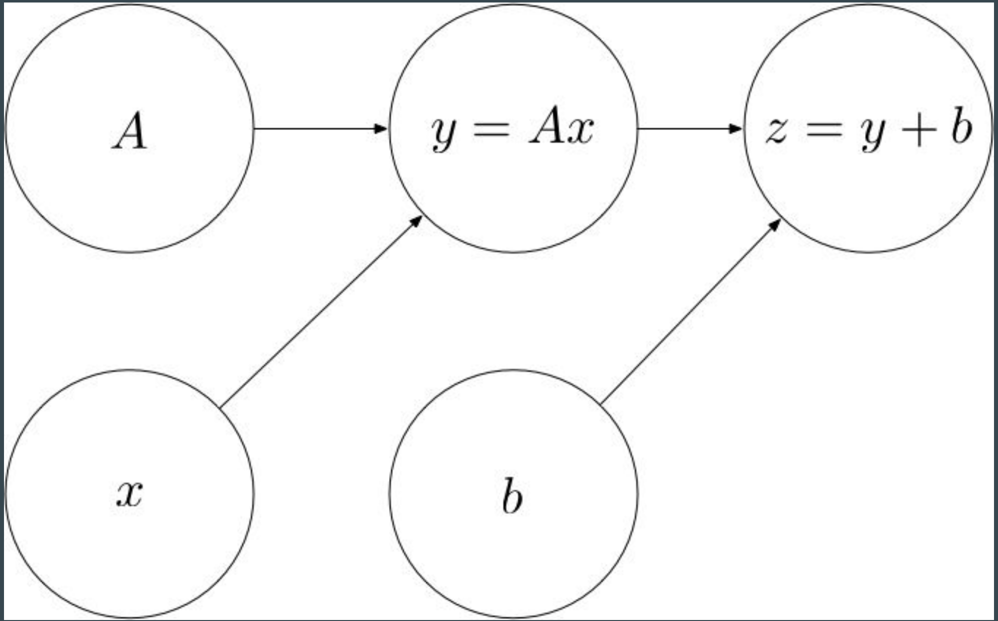
Computation paths

Data goes in/out in the graph through:

- IN: feeding → feed_dict
- OUT: fetching



Micro example I: the graph



Micro example II: the code

```
import tensorflow as tf
import random

graph = tf.Graph()
with graph.as_default():
    x = tf.placeholder(tf.float32, shape=[], name='x')
    A = tf.get_variable('A', shape=[], dtype=tf.float32,
initializer=tf.initializers.random_normal())
    b = tf.random_normal(shape=[], dtype=tf.float32, name='b')
    y = A * x
    z = y + b

with tf.Session(graph=graph) as sess:
    sess.run(tf.global_variables_initializer())
    result = sess.run(z, feed_dict={x: random.random()})
```

Fetching

Feeding

Definition phase

Execution phase

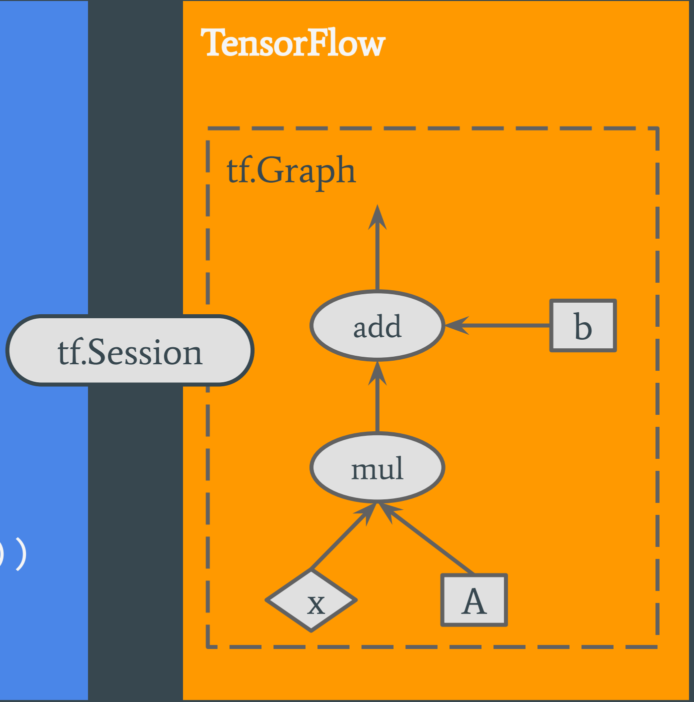
Micro example III: line by line

```
import tensorflow as tf
import random

graph = tf.Graph()
with graph.as_default():
    x = tf.placeholder(tf.float32, shape=[], name='x')
    A = tf.get_variable('A', shape=[], dtype=tf.float32,
initializer=tf.initializers.random_normal())
    b = tf.random_normal(shape=[], dtype=tf.float32, name='b')
    y = A * x
    z = y + b

with tf.Session(graph=graph) as sess:
    sess.run(tf.global_variables_initializer())
    result = sess.run(z, feed_dict={x: random.random()})
```

Python & TF

Python

```
graph = tf.Graph()
with graph.as_default():
    x = tf.placeholder(..., name='x')
    A = tf.get_variable('A', ...)
    b = tf.random_normal(..., name='b')
    y = A * x
    z = y + b

with tf.Session(graph=graph) as sess:
    sess.run(tf.global_variables_initializer())
    result = sess.run(z, feed_dict={x:
random.random()})
```



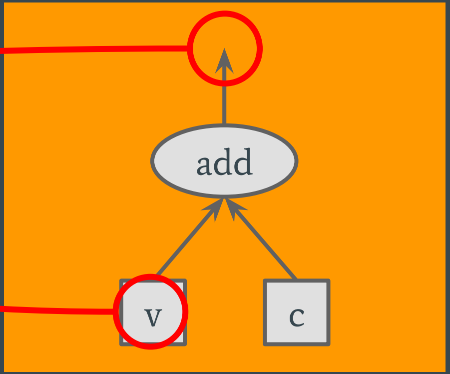
New family members

Special instances of tf.Tensor:

- **Variables** (tf.get_variable): holds values that can change from one session run to another
- **Placeholders** (tf.placeholder): represents a “data shell”, doesn’t know its value, only the shape and type. The real values are fed through the session (*feed_dict*)
- **Constants** (tf.constant): immutable value

Sanity check

```
v = tf.get_variable(‘v’, shape=[], initializer=1)
c = tf.constant(4)
v = tf.add(v, c) # same as ‘v + c’

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    result = sess.run(v)
```



Q: What’s the value of *result*?

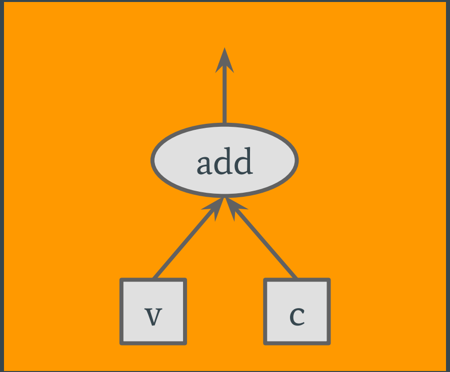
A: result = 5

Q: And what about the *graph variable v*?

A: v = 1

Sanity check II

```
v = tf.get_variable('v', shape=[], initializer=1)
c = tf.constant(4)
v_update = v.assign_add(c)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    result = sess.run(v_update)
```



Recap!

- tf.Session
  - sess.run()
- tf.Graph
  - graph.as_default()
- tf.placeholder
- tf.get_variable
  - assign, assign_add, assign_sub
- tf.constant
- Python operations overridden. TF provides custom functions too...
  - + == tf.add
  - x == tf.mul

Questions?