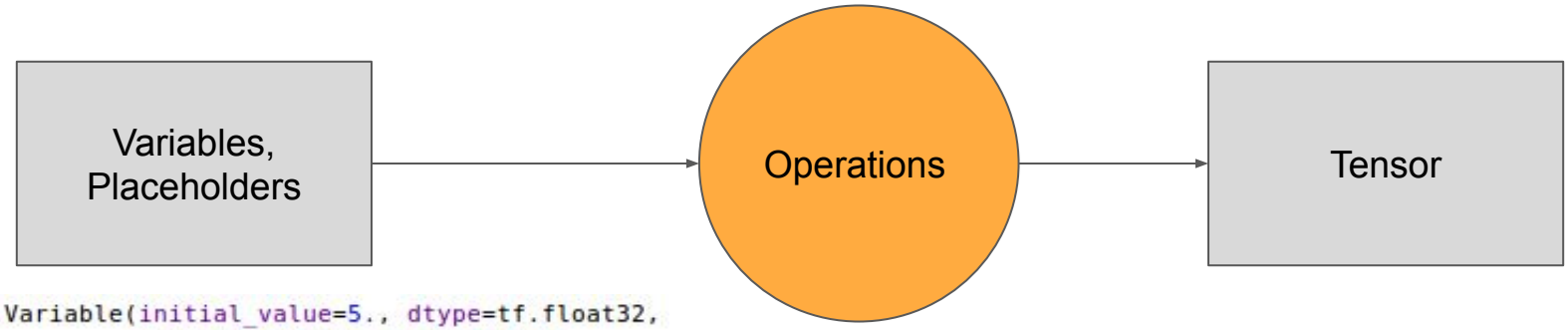
TensorFlow

Курс “Практическое применение по TensorFlow”
Шигапова Фирюза Зинатуллаевна
1-й семестр, 2019 г.

https://github.com/Firyuza/TensorFlowPractice

Tensor

```
output = tf.multiply(temp, temp)
```




```
temp = tf.Variable(initial_value=5., dtype=tf.float32,
                   trainable=False, name='temp')
```

Quiz. What will be printed?

```
g1 = tf.Graph()
with g1.as_default():
    op1 = tf.multiply(2, 3)

with tf.Graph().as_default() as g2:
    op2 = tf.subtract(3, 1)

nrof_operations = len(tf.get_default_graph().get_operations())

print(nrof_operations)
```

TF program. Graph

```
graph = tf.Graph()
with graph.as_default():
    v1_ph = tf.placeholder(shape=[DIM], dtype=tf.float32, name='vector1')
    v2_ph = tf.placeholder(shape=[DIM], dtype=tf.float32, name='vector2')
```

TF program. Tensors

```
graph = tf.Graph()
with graph.as_default():
    v1_ph = tf.placeholder(shape=[DIM], dtype=tf.float32, name='vector1')
    v2_ph = tf.placeholder(shape=[DIM], dtype=tf.float32, name='vector2')

    sub = tf.subtract(v1_ph, v2_ph)
    square = tf.square(sub)
    sum = tf.reduce_sum(square)

    distance = tf.sqrt(sum)
```

TF program. Session

```
if USE_GPU:
    gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=GPU_MEMORY_FRACTION)
    session_conf = tf.ConfigProto(allow_soft_placement=True,
                                  log_device_placement=True,
                                  gpu_options=gpu_options)
else:
    session_conf = tf.ConfigProto(
        device_count={'CPU': 1, 'GPU': 0},
        allow_soft_placement=True,
        log_device_placement=True
    )
session = tf.Session(graph=graph, config=session_conf)
```

TF program. Output

```
with session.as_default():
    distance_output = session.run(distance,
                                  feed_dict={v1_ph: data['v1'],
                                             v2_ph: data['v2']})
    print(distance_output)
```

TF program

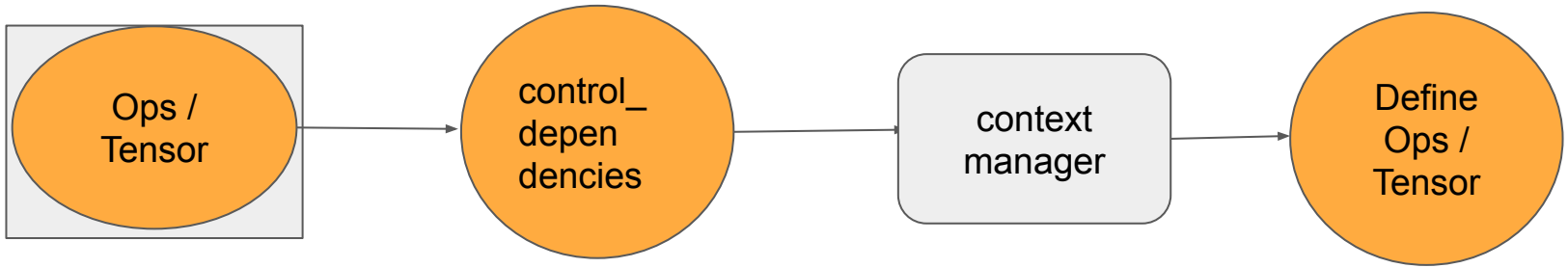
```
import tensorflow as tf

DIM = 3
USE_GPU = True
GPU_MEMORY_FRACTION = 1.0

def run(data):
    graph = tf.Graph()
    with graph.as_default():
        v1_ph = tf.placeholder(shape=[DIM], dtype=tf.float32, name='vector1')
        v2_ph = tf.placeholder(shape=[DIM], dtype=tf.float32, name='vector2')

        sub = tf.subtract(v1_ph, v2_ph)
        square = tf.square(sub)
        sum = tf.reduce_sum(square)

        distance = tf.sqrt(sum)

    if USE_GPU:
        gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=GPU_MEMORY_FRACTION)
        session_conf = tf.ConfigProto(allow_soft_placement=True,
                                      log_device_placement=True,
                                      gpu_options=gpu_options)
    else:
        session_conf = tf.ConfigProto(
            device_count={'CPU': 1, 'GPU': 0},
            allow_soft_placement=True,
            log_device_placement=True
        )
    session = tf.Session(graph=graph, config=session_conf)

    with session.as_default():
        distance_output = session.run(distance,
                                      feed_dict={v1_ph: data['v1'],
                                                 v2_ph: data['v2']})
        print(distance_output)

    return

run({'v1': [1, 2, 3],
     'v2': [4, 5, 6]})
```

TF program batch. Graph

```
graph = tf.Graph()
with graph.as_default():
    v1_ph = tf.placeholder(shape=[None, DIM], dtype=tf.float32, name='vector1')
    v2_ph = tf.placeholder(shape=[None, DIM], dtype=tf.float32, name='vector2')
```

Продолжите самостоятельно

control_dependencies



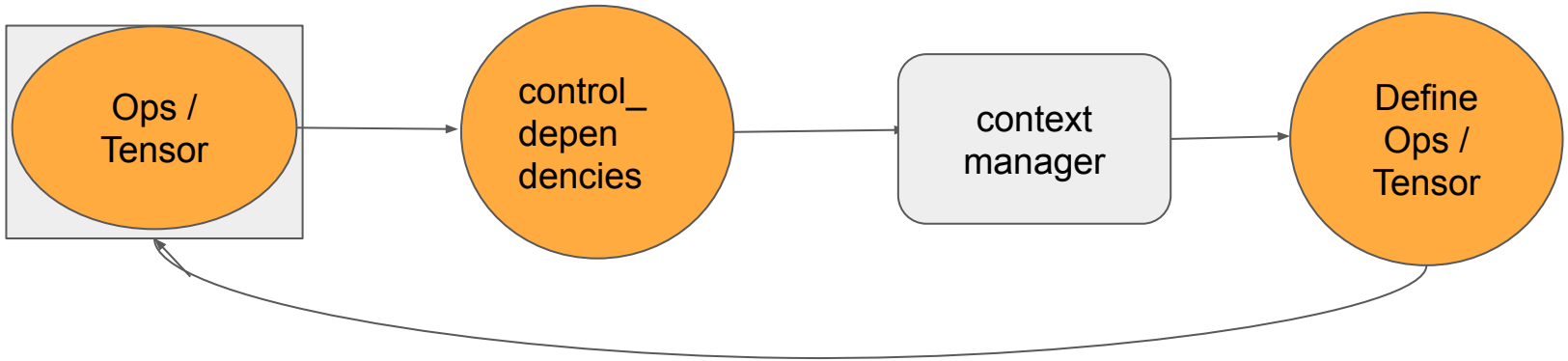

```
with g.control_dependencies([a, b, c]):
  # `d` and `e` will only run after `a`, `b`, and `c` have executed.
  d = ...
  e = ...
```

Source: https://www.tensorflow.org/api_docs/python/tf/Graph#control_dependencies

control_dependencies

```
multiply_op = tf.multiply(2, 3)
subtract_op = tf.subtract(multiply_op, 3)

with tf.control_dependencies([multiply_op, subtract_op]):
    # Operations constructed here will be executed after
    # multiply_op, subtract_op
    divide_op = tf.divide(subtract_op, multiply_op)

output_op = tf.add(divide_op, divide_op)

session = tf.Session()
output = session.run(output_op)

print(output) # 1.0
```

control_dependencies



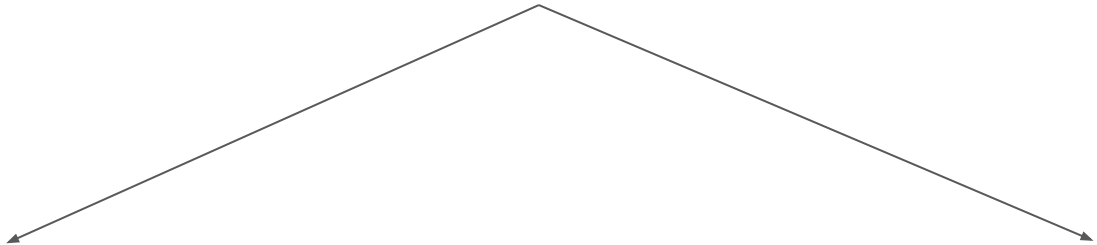

```
with g.control_dependencies([a, b, c]):
  # `d` and `e` will only run after `a`, `b`, and `c` have executed.
  d = ...
  e = ...
```

Source: https://www.tensorflow.org/api_docs/python/tf/Graph#control_dependencies

control_dependencies. Clean control dependency

```
multiply_op = tf.multiply(2, 3)
subtract_op = tf.subtract(multiply_op, 3)

with tf.control_dependencies([multiply_op, subtract_op]):
    # Operations constructed here will be executed after
    # multiply_op, subtract_op
    divide_op = tf.divide(subtract_op, multiply_op)
    with tf.control_dependencies(None):
        # Operations constructed here will not not wait
        # ANY operation
        output_op = tf.add(divide_op, divide_op)

session = tf.Session()
output = session.run(output_op)

print(output)  # 1.0
```

How to Debug (or Print)



tf.print() via *Session*

Eager Execution

tf.print

```
print_op = [tf.print("sub:", sub, output_stream=sys.stdout),
            tf.print("square:", square, output_stream=sys.stdout),
            tf.print("distance:", distance, output_stream=sys.stdout)]
```

```
with session.as_default():
    distance_output = session.run(distance,
                                  feed_dict={v1_ph: data['v1'],
                                             v2_ph: data['v2']})
```

```
with session.as_default():
    distance_output, _ = session.run([distance, print_op],
                                     feed_dict={v1_ph: data['v1'],
                                                v2_ph: data['v2']})
```

NB: _ is *None*

tf.print via control_dependencies

```
print_op = tf.print("distance:", distance, output_stream=sys.stdout)

with tf.control_dependencies([print_op]):
    distance = distance * 0.
```

1. Print ***distance*** value before it's multiplied by zero
2. session.run returns ***distance*** is zero

```
with session.as_default():
    distance_output, _ = session.run([distance, print_op],
                                     feed_dict={v1_ph: data['v1'],
                                                v2_ph: data['v2']})
    print(distance_output)
```

```
distance: 5.19615221
0.0
```

<= Print by ***tf.print***
<= ***distance_output***

Eager execution

- Use Python control flow instead of Graph control flow
- Works with Python objects and NumPy arrays (convert to tf.Tensor)
- tf.Tensor refers concrete value
- tf.Tensor.numpy() returns NumPy array
- Easy to control backpropagation
- *Program* state is *not* stored in *global collection* and managed by *tf.Session* anymore!

Eager execution

```
tf.enable_eager_execution()
```

```
tensor = tf.range(10)
tf.print("tensors:", tensor,  output_stream=sys.stdout)
# tensors: [0 1 2 ... 7 8 9]

tensor = tf.square(tensor)
tf.print("tensors:", tensor,  output_stream=sys.stdout)
# tensors: [0 1 4 ... 49 64 81]
```

```
tensor = tf.range(10)
print(tensor)
# tf.Tensor([0 1 2 3 4 5 6 7 8 9], shape=(10,), dtype=int32)
```

Eager execution vs. Placeholder

Not working!

```
RuntimeError: tf.placeholder() is not compatible with eager execution.
```