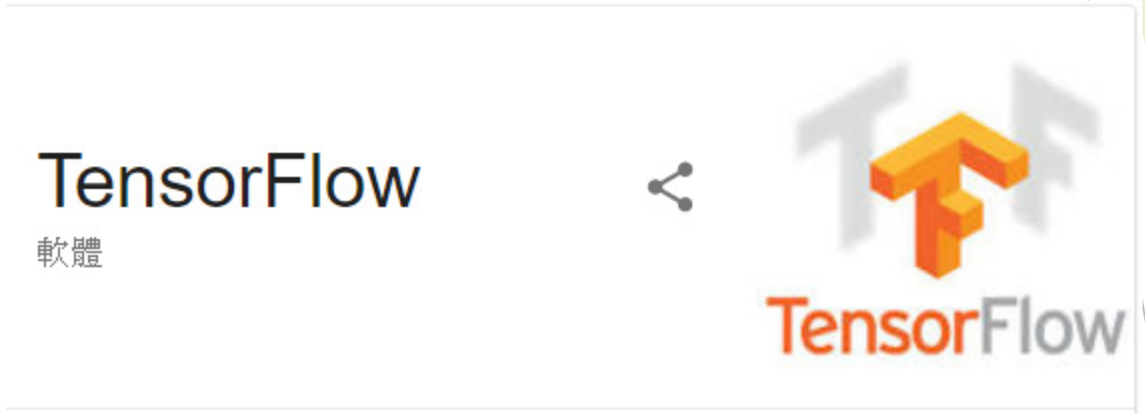
Tensorflow

Outline

- Tensorflow
- Dataflow graphs
- Tensorflow basic operation
- Homework

Tensorflow

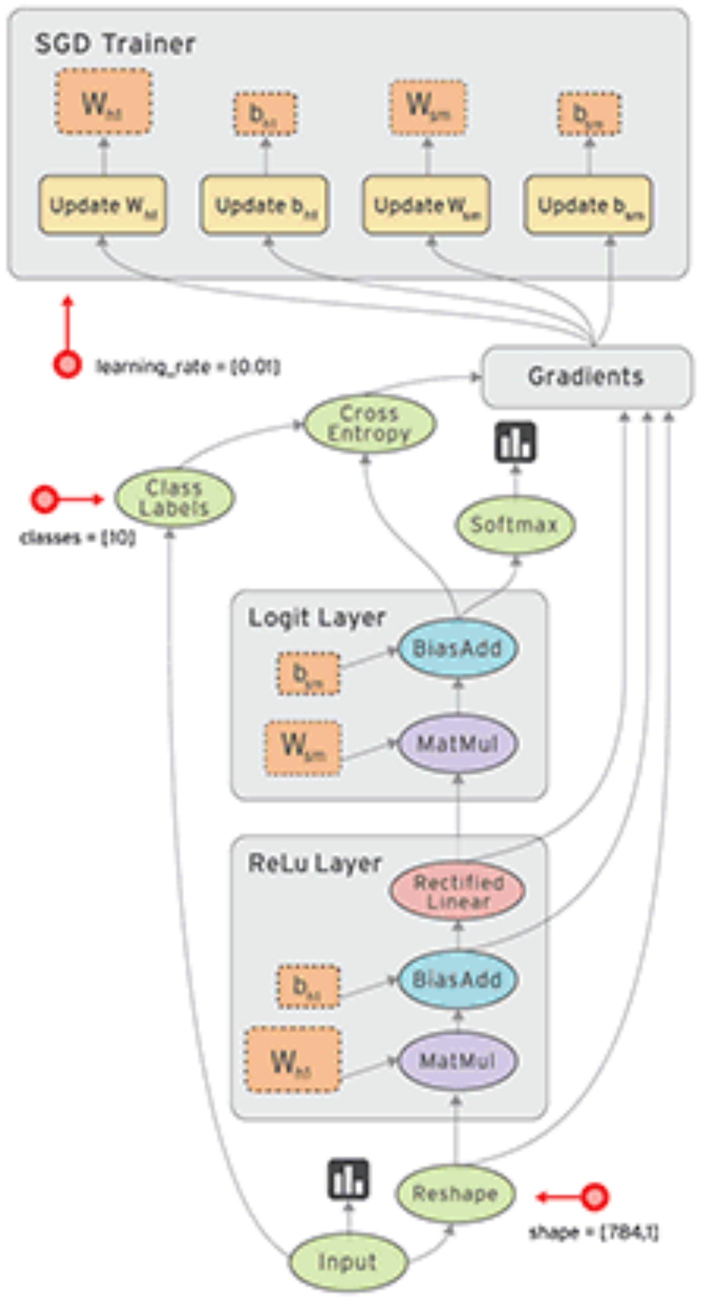
- open-source software
- It is a symbolic math library
- machine learning
- production at Google


TensorFlow
軟體

Dataflow graphs

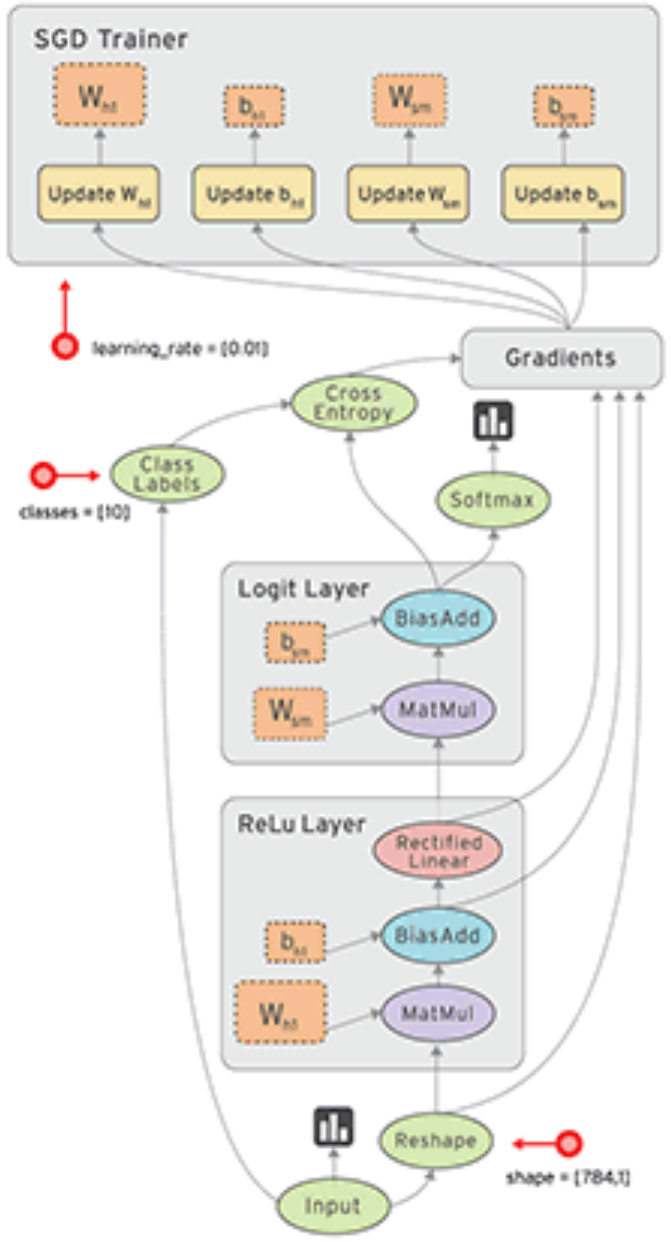
- TensorFlow is calculated using data flow graphs
- First need to create a dataflow diagram
- Put our data (tensor) in the dataflow graph Computation
- Nodes represent math operations in the diagram
- Edges represent multidimensional data (tensors).
- When Training the model ,tensors continually from a node

flow to another node in data flow diagrams



- The graph is composed of two types of objects :

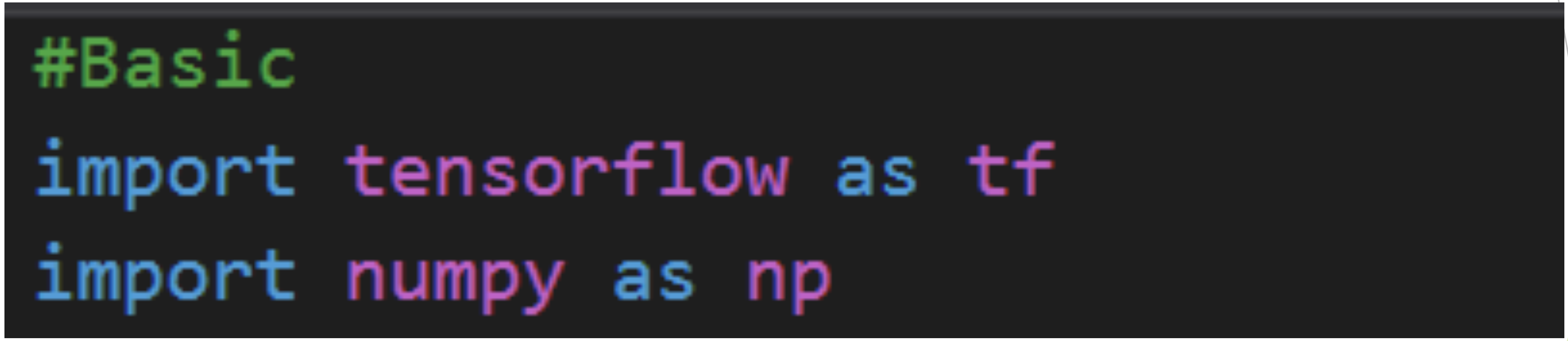
1. Node : A Operations (ops), calculations that consume and produce
2. Edge : A Tensors ,Values that will flow through the graph



Tensorflow basic operation

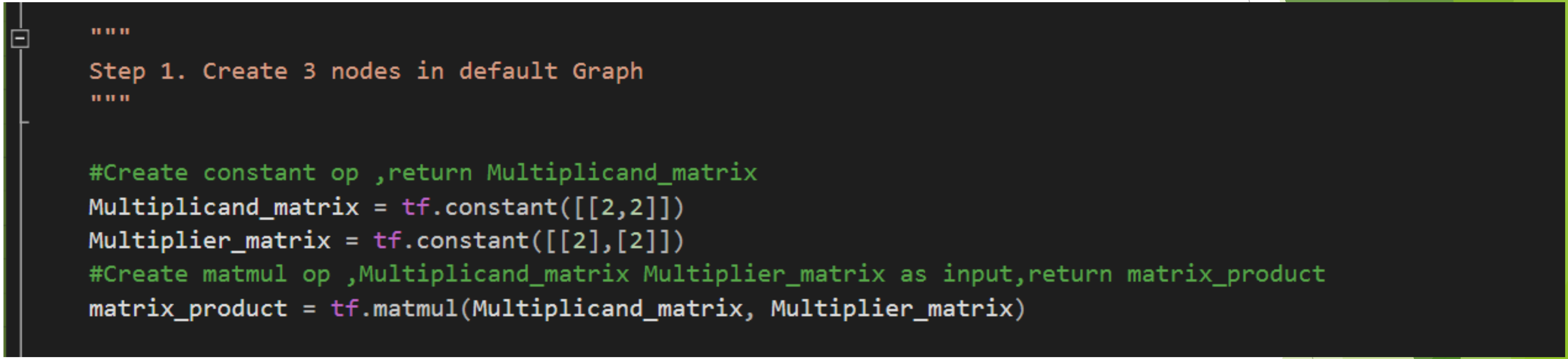
- Import
- Create nodes(op)

import

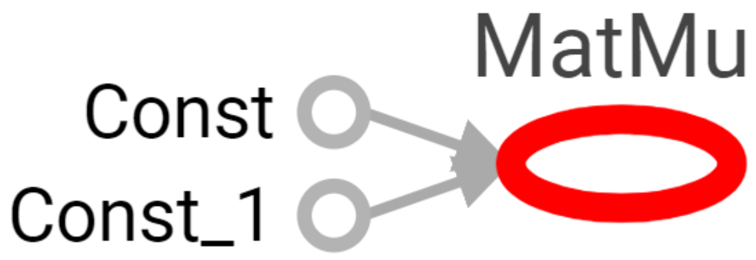


```
#Basic
import tensorflow as tf
import numpy as np
```

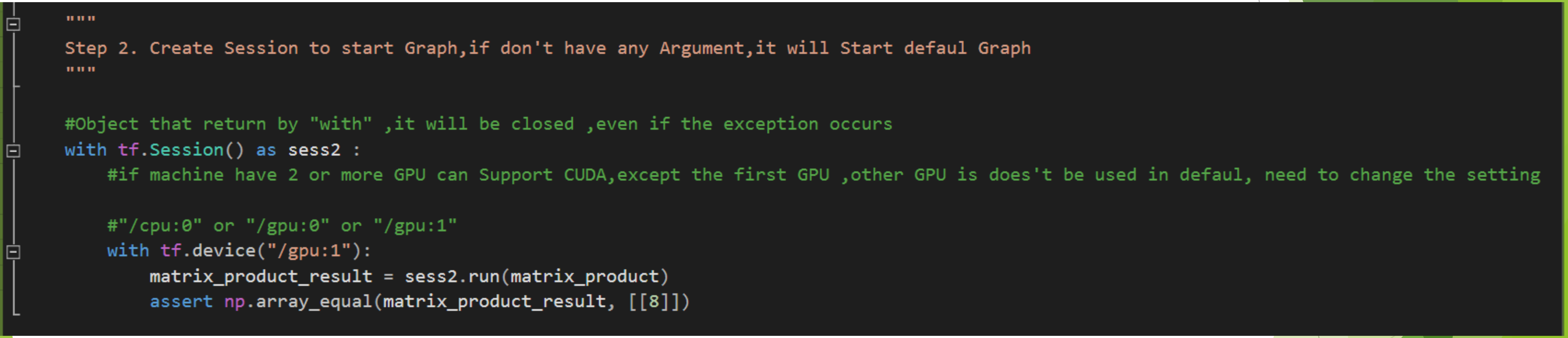
Create nodes(op)



```
"""
Step 1. Create 3 nodes in default Graph
"""

#Create constant op ,return Multiplicand_matrix
Multiplicand_matrix = tf.constant([[2,2]])
Multiplier_matrix = tf.constant([[2],[2]])
#Create matmul op ,Multiplicand_matrix Multiplier_matrix as input,return matrix_product
matrix_product = tf.matmul(Multiplicand_matrix, Multiplier_matrix)
```

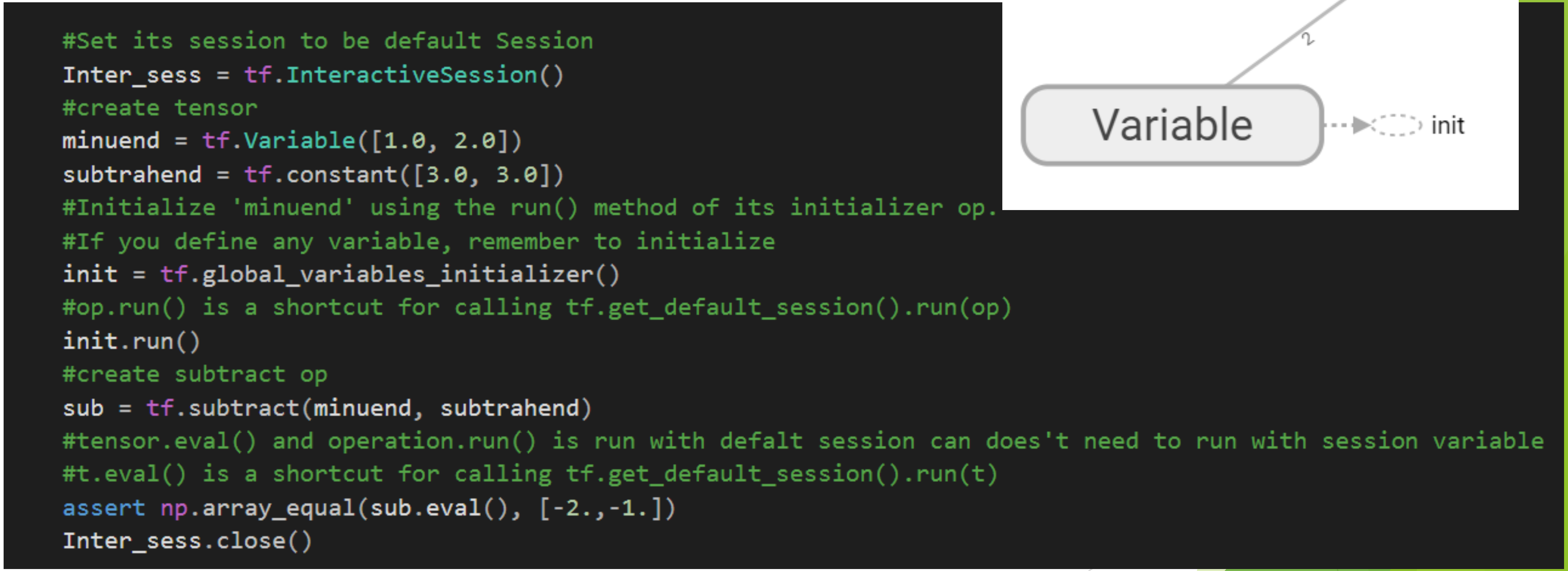


With as

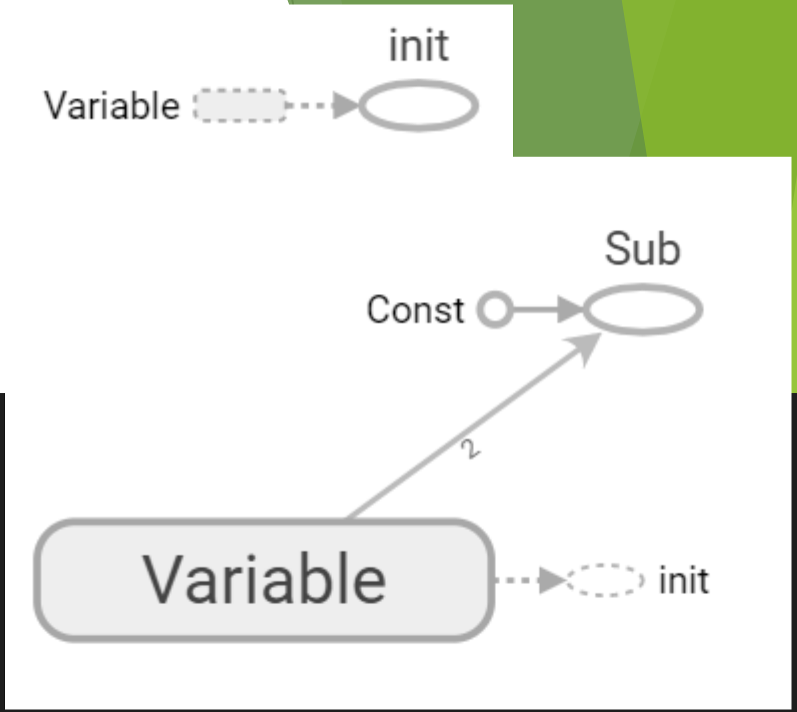


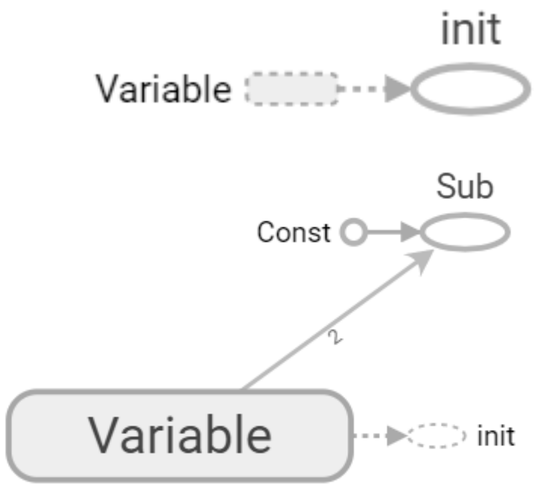
```
"""
Step 2. Create Session to start Graph,if don't have any Argument,it will Start defaul Graph
"""

#Object that return by "with" ,it will be closed ,even if the exception occurs
with tf.Session() as sess2 :
    #if machine have 2 or more GPU can Support CUDA,except the first GPU ,other GPU is does't be used in defaul, need to change the setting

    #"/cpu:0" or "/gpu:0" or "/gpu:1"
    with tf.device("/gpu:1"):
        matrix_product_result = sess2.run(matrix_product)
        assert np.array_equal(matrix_product_result, [[8]])
```

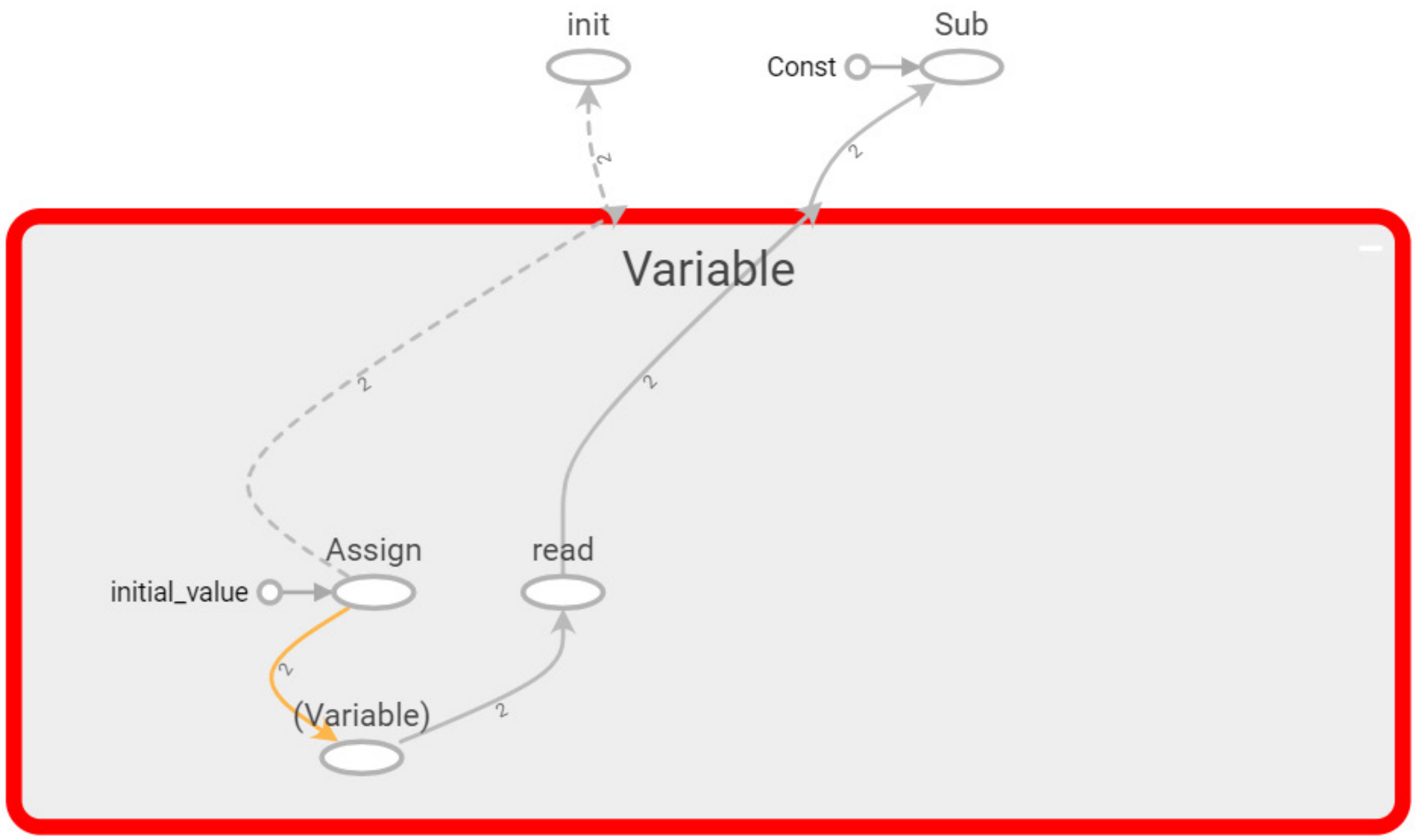
Avoid only one variable to hold the Session



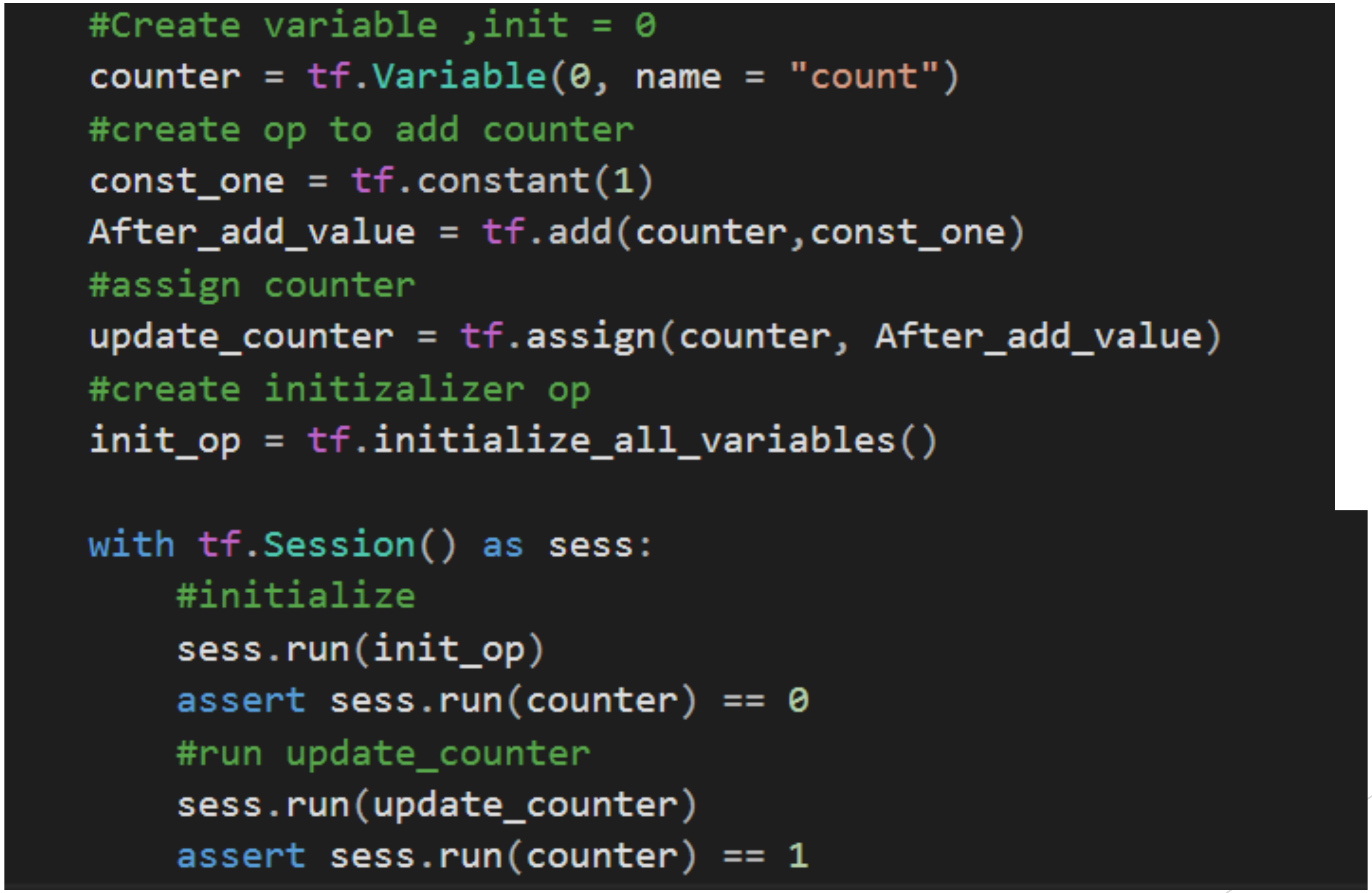
```
#Set its session to be default Session
Inter_sess = tf.InteractiveSession()
#create tensor
minuend = tf.Variable([1.0, 2.0])
subtrahend = tf.constant([3.0, 3.0])
#Initialize 'minuend' using the run() method of its initializer op.
#If you define any variable, remember to initialize
init = tf.global_variables_initializer()
#op.run() is a shortcut for calling tf.get_default_session().run(op)
init.run()
#create subtract op
sub = tf.subtract(minuend, subtrahend)
#tensor.eval() and operation.run() is run with defalt session can does't need to run with session variable
#t.eval() is a shortcut for calling tf.get_default_session().run(t)
assert np.array_equal(sub.eval(), [-2.,-1.])
Inter_sess.close()
```



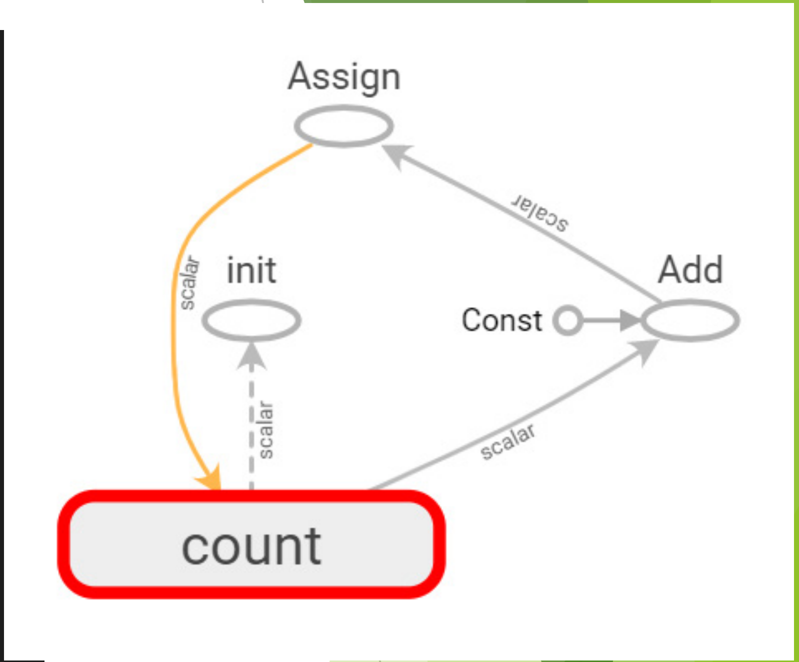

init
Variable
Sub
Const
2
Variable
init

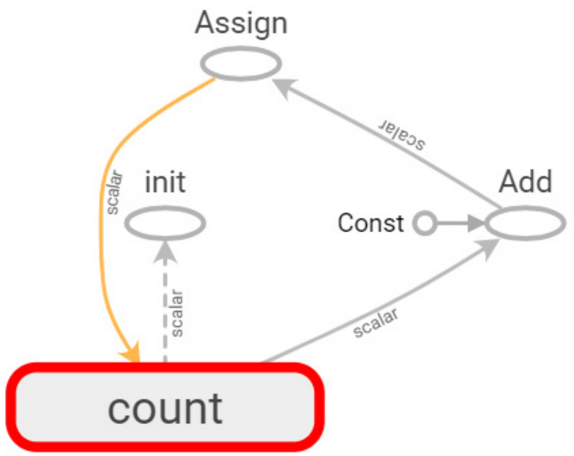

init
Sub
Const
2
2
Variable
2
2
Assign
read
initial_value
2
(Variable)
2

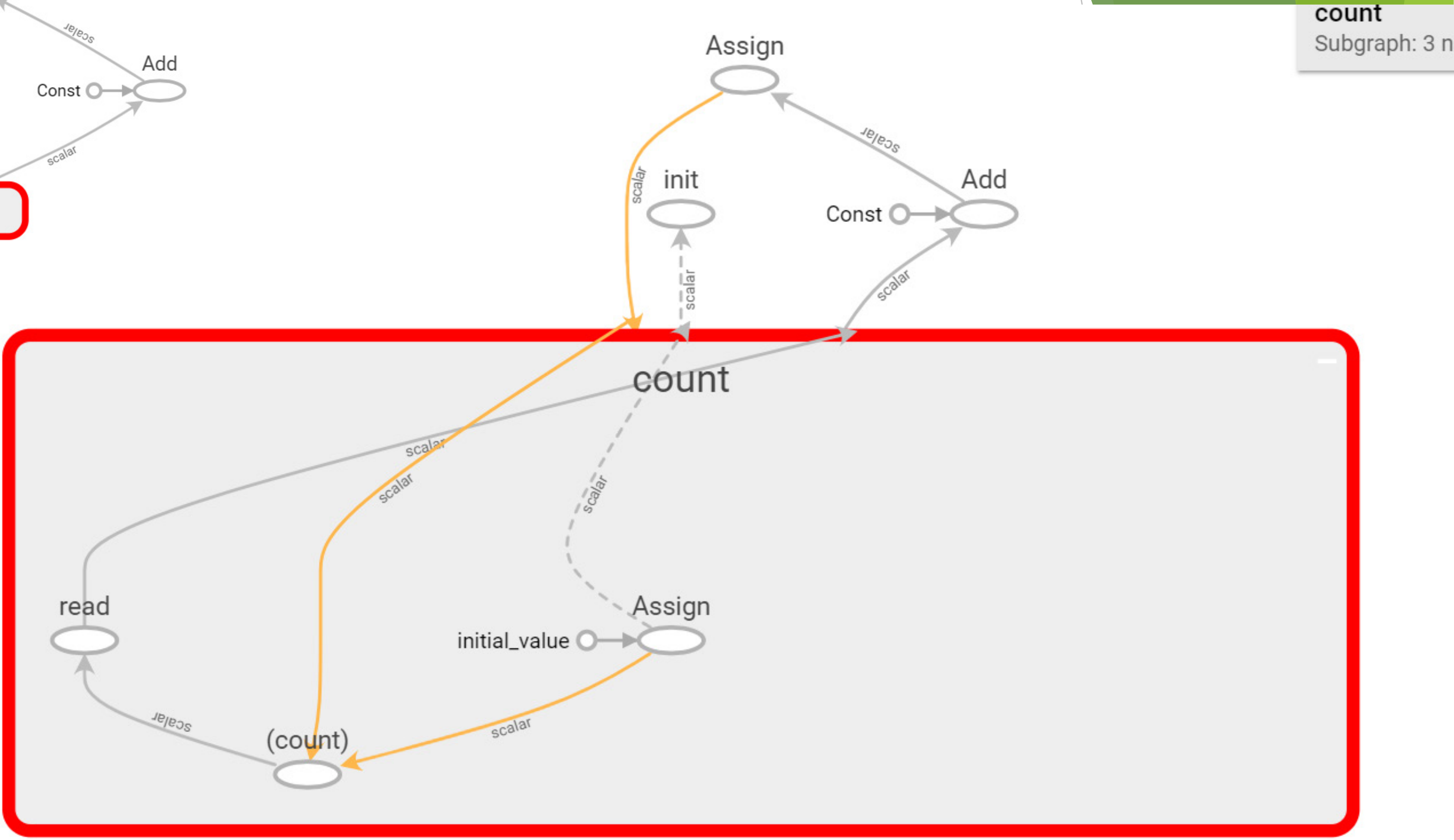
Update tensor by run training Graph



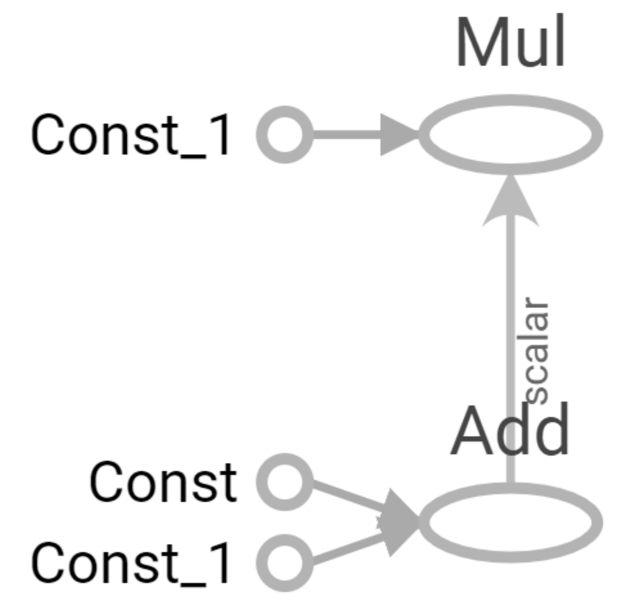
```
#Create variable ,init = 0
counter = tf.Variable(0, name = "count")
#create op to add counter
const_one = tf.constant(1)
After_add_value = tf.add(counter,const_one)
#assign counter
update_counter = tf.assign(counter, After_add_value)
#create initizalizer op
init_op = tf.initialize_all_variables()

with tf.Session() as sess:
    #initialize
    sess.run(init_op)
    assert sess.run(counter) == 0
    #run update_counter
    sess.run(update_counter)
    assert sess.run(counter) == 1
```

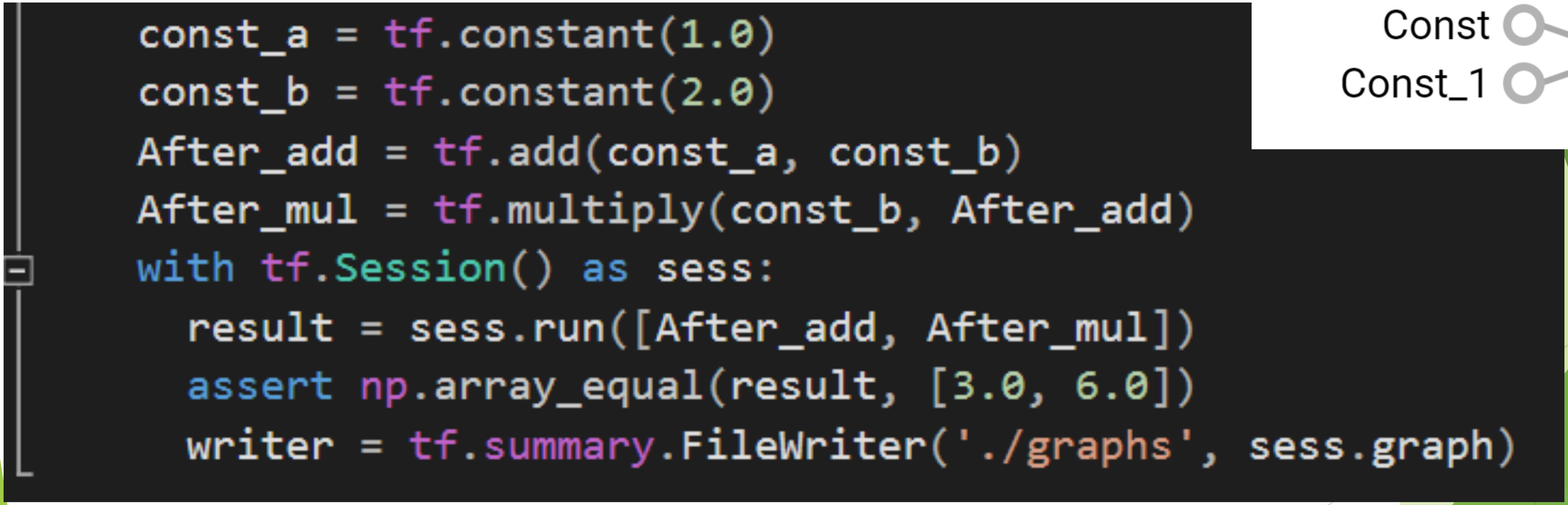



Assign
scalar
init
Add
Const
scalar
scalar
scalar
count


count
Subgraph: 3 n
Assign
scalar
init
Add
Const
scalar
scalar
scalar
count
scalar
scalar
scalar
read
Assign
initial_value
scalar
(count)
scalar

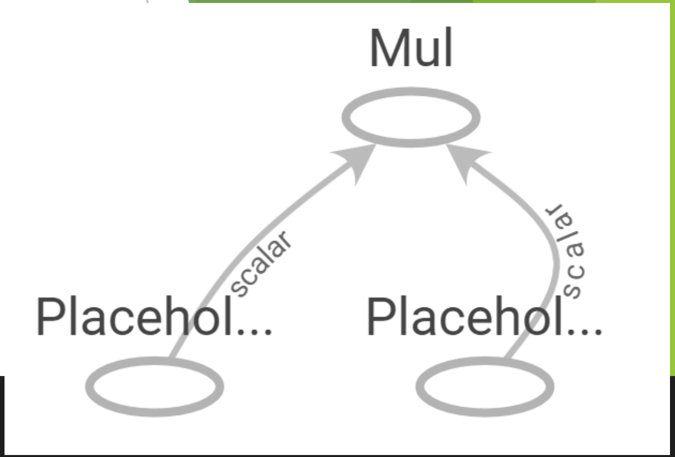
receive more than 1 tensor by use "Fetch"

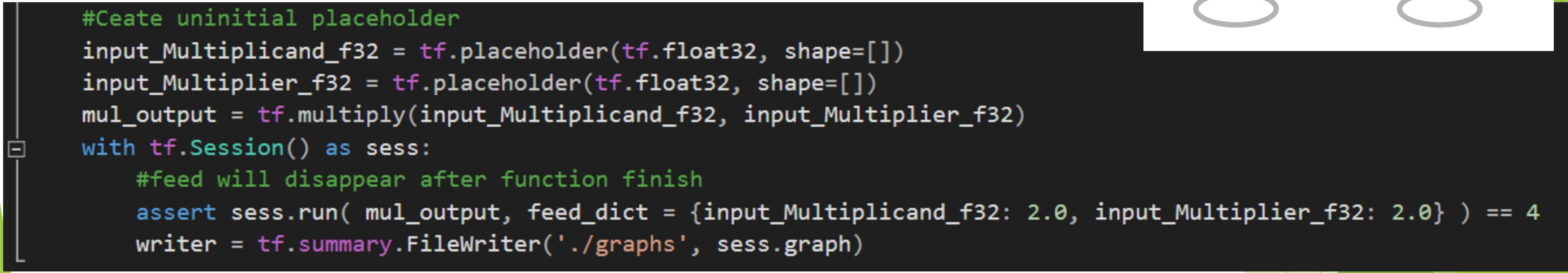




```
    const_a = tf.constant(1.0)
    const_b = tf.constant(2.0)
    After_add = tf.add(const_a, const_b)
    After_mul = tf.multiply(const_b, After_add)
    with tf.Session() as sess:
        result = sess.run([After_add, After_mul])
        assert np.array_equal(result, [3.0, 6.0])
        writer = tf.summary.FileWriter('./graphs', sess.graph)
```

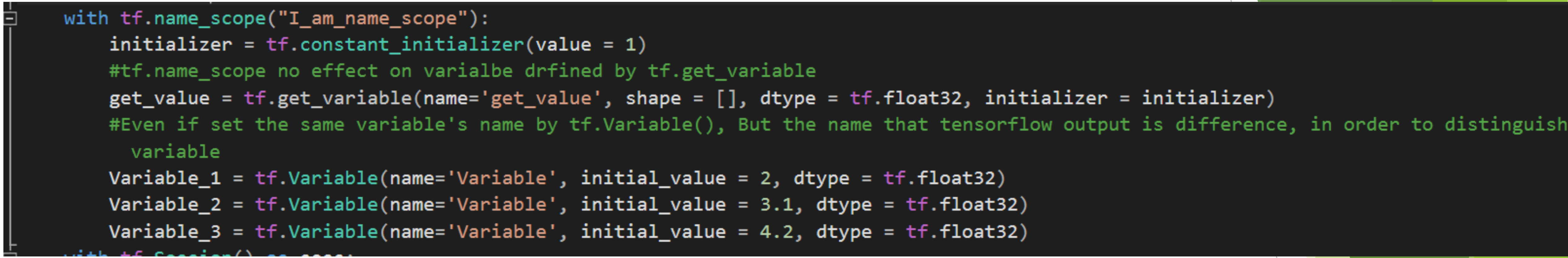
When Graph operating, Modify or insert the tensor



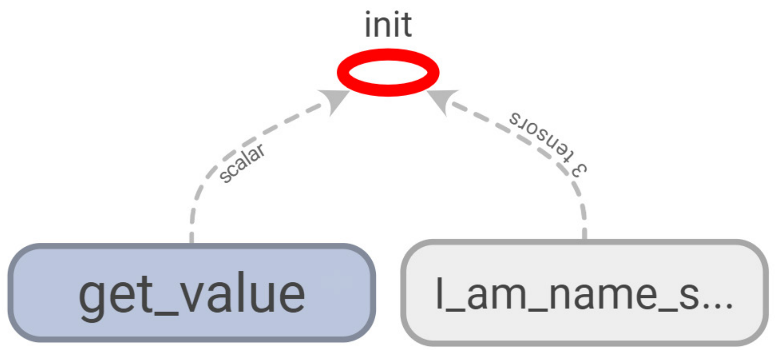


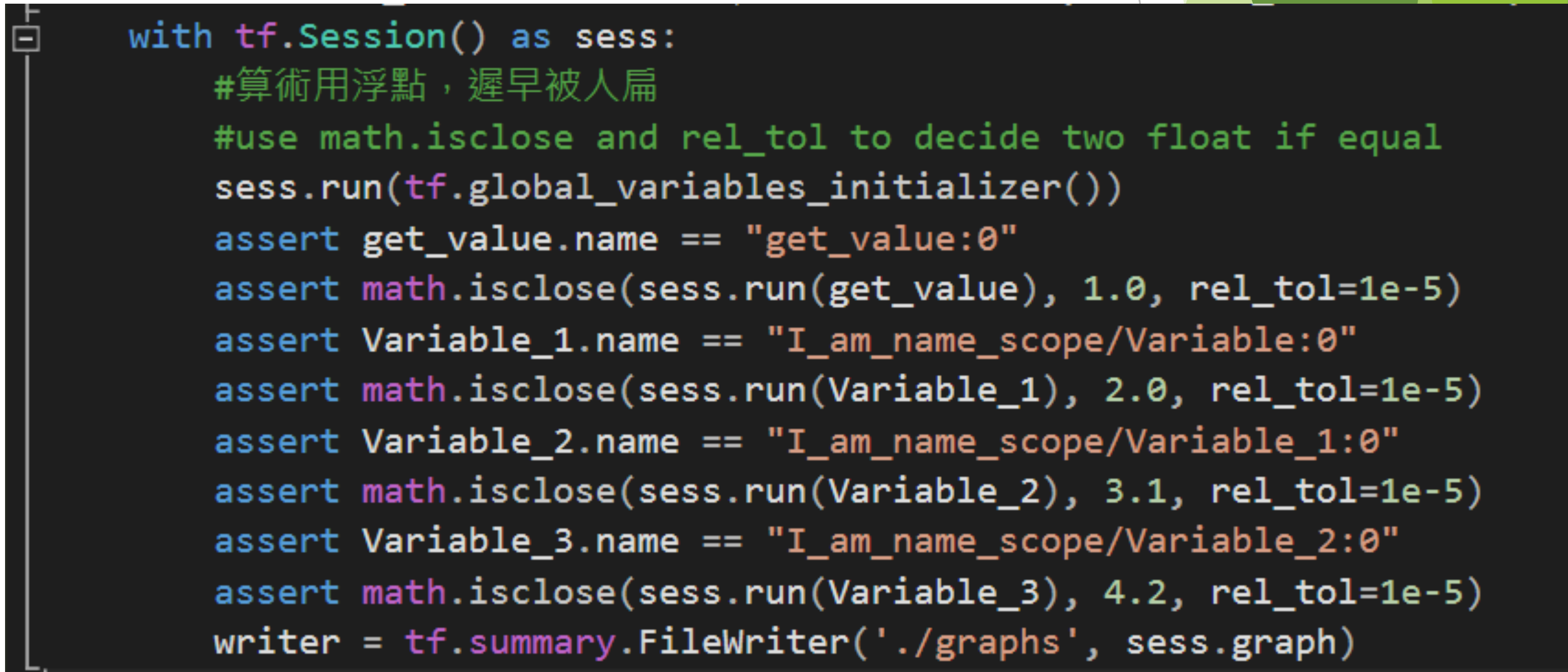
```
#Ceate uninitial placeholder
input_Multiplicand_f32 = tf.placeholder(tf.float32, shape=[])
input_Multiplier_f32 = tf.placeholder(tf.float32, shape=[])
mul_output = tf.multiply(input_Multiplicand_f32, input_Multiplier_f32)
with tf.Session() as sess:
    #feed will disappear after function finish
    assert sess.run( mul_output, feed_dict = {input_Multiplicand_f32: 2.0, input_Multiplier_f32: 2.0} ) == 4
    writer = tf.summary.FileWriter('./graphs', sess.graph)
```

name_scope

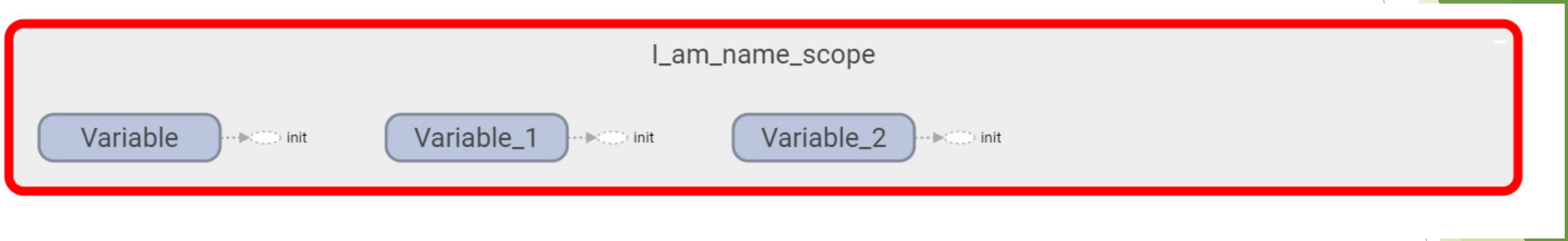


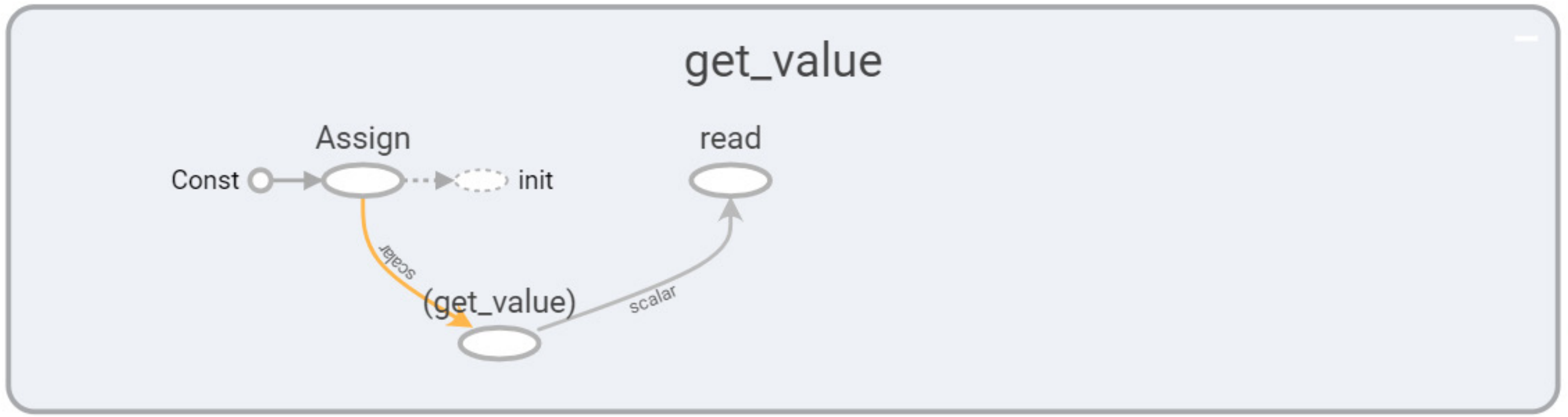
```
with tf.name_scope("I_am_name_scope"):
    initializer = tf.constant_initializer(value = 1)
    #tf.name_scope no effect on varialbe drfined by tf.get_variable
    get_value = tf.get_variable(name='get_value', shape = [], dtype = tf.float32, initializer = initializer)
    #Even if set the same variable's name by tf.Variable(), But the name that tensorflow output is difference, in order to distinguish
     variable
    Variable_1 = tf.Variable(name='Variable', initial_value = 2, dtype = tf.float32)
    Variable_2 = tf.Variable(name='Variable', initial_value = 3.1, dtype = tf.float32)
    Variable_3 = tf.Variable(name='Variable', initial_value = 4.2, dtype = tf.float32)
```





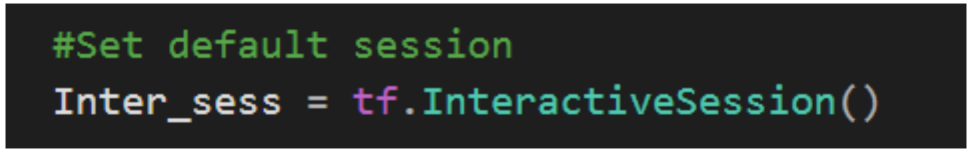
```
with tf.Session() as sess:
    #算術用浮點，遲早被人扁
    #use math.isclose and rel_tol to decide two float if equal
    sess.run(tf.global_variables_initializer())
    assert get_value.name == "get_value:0"
    assert math.isclose(sess.run(get_value), 1.0, rel_tol=1e-5)
    assert Variable_1.name == "I_am_name_scope/Variable:0"
    assert math.isclose(sess.run(Variable_1), 2.0, rel_tol=1e-5)
    assert Variable_2.name == "I_am_name_scope/Variable_1:0"
    assert math.isclose(sess.run(Variable_2), 3.1, rel_tol=1e-5)
    assert Variable_3.name == "I_am_name_scope/Variable_2:0"
    assert math.isclose(sess.run(Variable_3), 4.2, rel_tol=1e-5)
    writer = tf.summary.FileWriter('./graphs', sess.graph)
```


I_am_name_scope
Variable
init
Variable_1
init
Variable_2
init

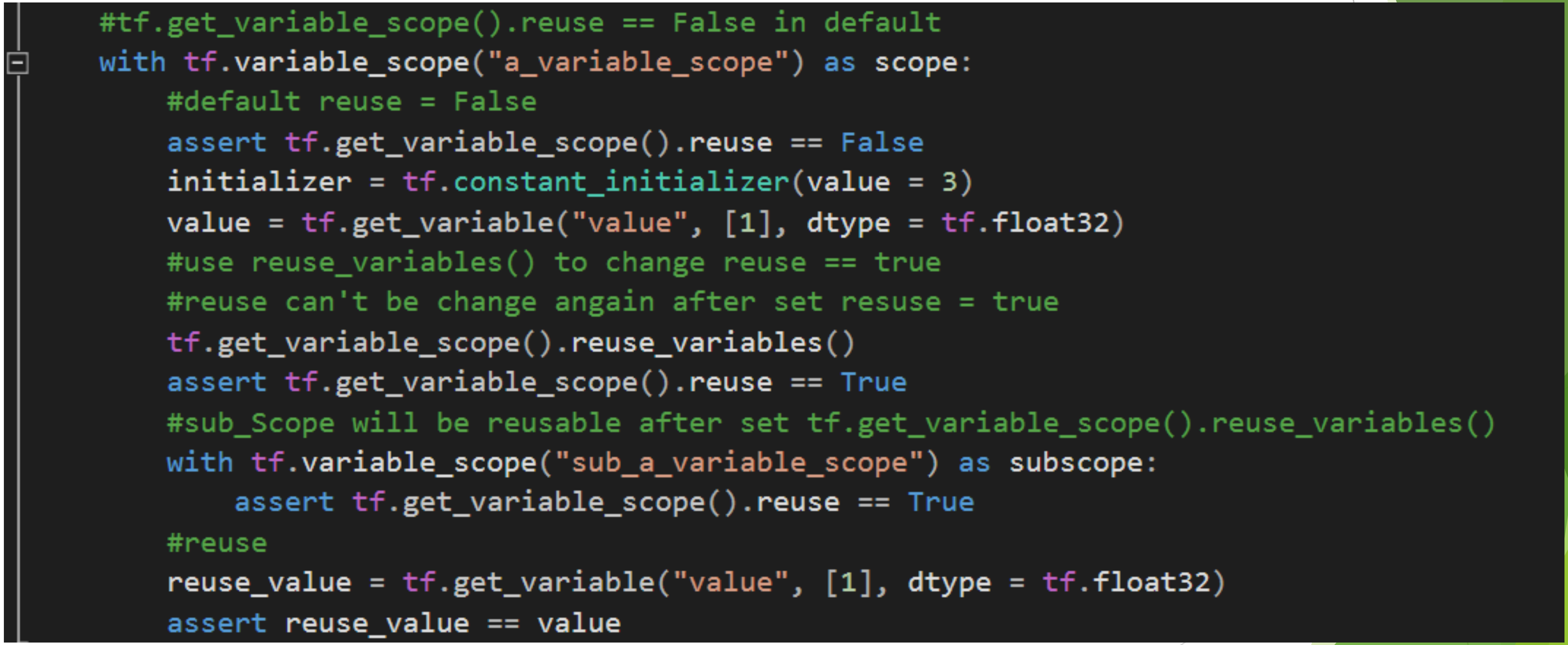

get_value
Assign
read
Const
init
scalar
(get_value)
scalar

reuse variable

- reuse variable by tf.variable_scope() and tf.get_variable()
- If decide to set reuse variable, need to use scope.reuse_variables(), otherwise, it will get error
- Why we need to reuse variable? When start training and test, Maybe two of them have different structure, but need have same weight and bias

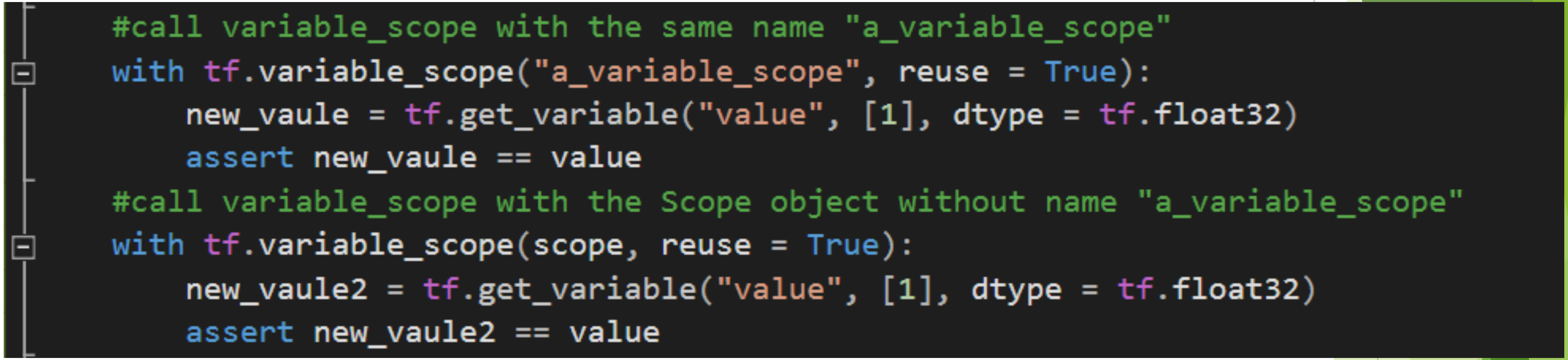


```
#Set default session
Inter_sess = tf.InteractiveSession()
```



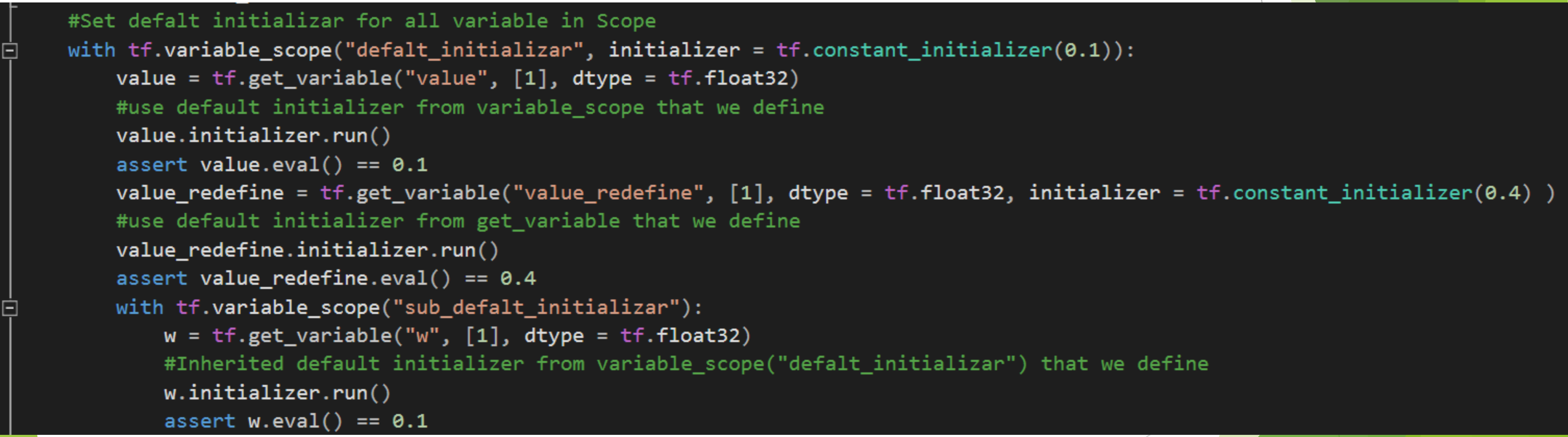
```
    #tf.get_variable_scope().reuse == False in default
    with tf.variable_scope("a_variable_scope") as scope:
        #default reuse = False
        assert tf.get_variable_scope().reuse == False
        initializer = tf.constant_initializer(value = 3)
        value = tf.get_variable("value", [1], dtype = tf.float32)
        #use reuse_variables() to change reuse == true
        #reuse can't be change angain after set resuse = true
        tf.get_variable_scope().reuse_variables()
        assert tf.get_variable_scope().reuse == True
        #sub_Scope will be reusable after set tf.get_variable_scope().reuse_variables()
        with tf.variable_scope("sub_a_variable_scope") as subscope:
            assert tf.get_variable_scope().reuse == True
        #reuse
        reuse_value = tf.get_variable("value", [1], dtype = tf.float32)
        assert reuse_value == value
```

Call scope again



```
#call variable_scope with the same name "a_variable_scope"
with tf.variable_scope("a_variable_scope", reuse = True):
    new_vaule = tf.get_variable("value", [1], dtype = tf.float32)
    assert new_vaule == value
#call variable_scope with the Scope object without name "a_variable_scope"
with tf.variable_scope(scope, reuse = True):
    new_vaule2 = tf.get_variable("value", [1], dtype = tf.float32)
    assert new_vaule2 == value
```

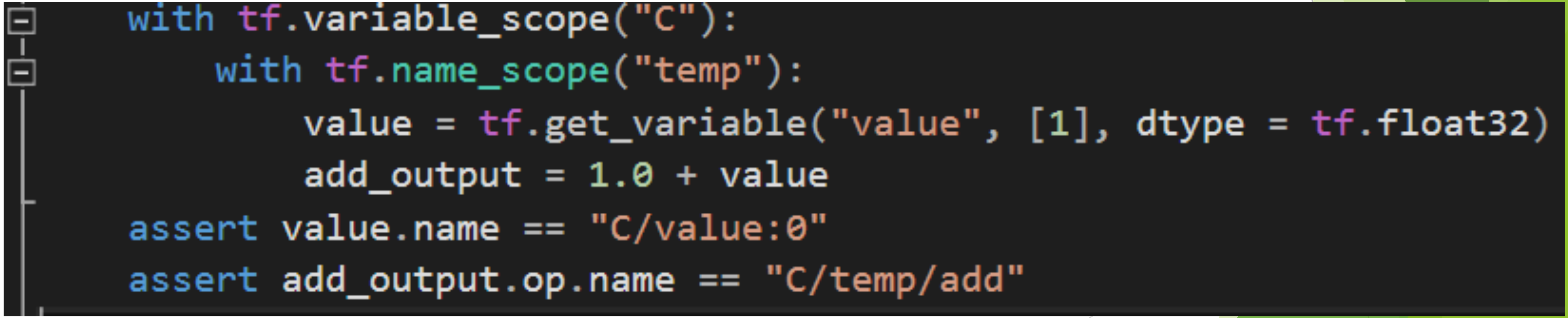
Set default initializer for all variable in Scope



```
#Set defalt initializar for all variable in Scope
with tf.variable_scope("defalt_initializar", initializer = tf.constant_initializer(0.1)):
    value = tf.get_variable("value", [1], dtype = tf.float32)
    #use default initializer from variable_scope that we define
    value.initializer.run()
    assert value.eval() == 0.1
    value_redefine = tf.get_variable("value_redefine", [1], dtype = tf.float32, initializer = tf.constant_initializer(0.4) )
    #use default initializer from get_variable that we define
    value_redefine.initializer.run()
    assert value_redefine.eval() == 0.4
    with tf.variable_scope("sub_defalt_initializar"):
        w = tf.get_variable("w", [1], dtype = tf.float32)
        #Inherited default initializer from variable_scope("defalt_initializar") that we define
        w.initializer.run()
        assert w.eval() == 0.1
```

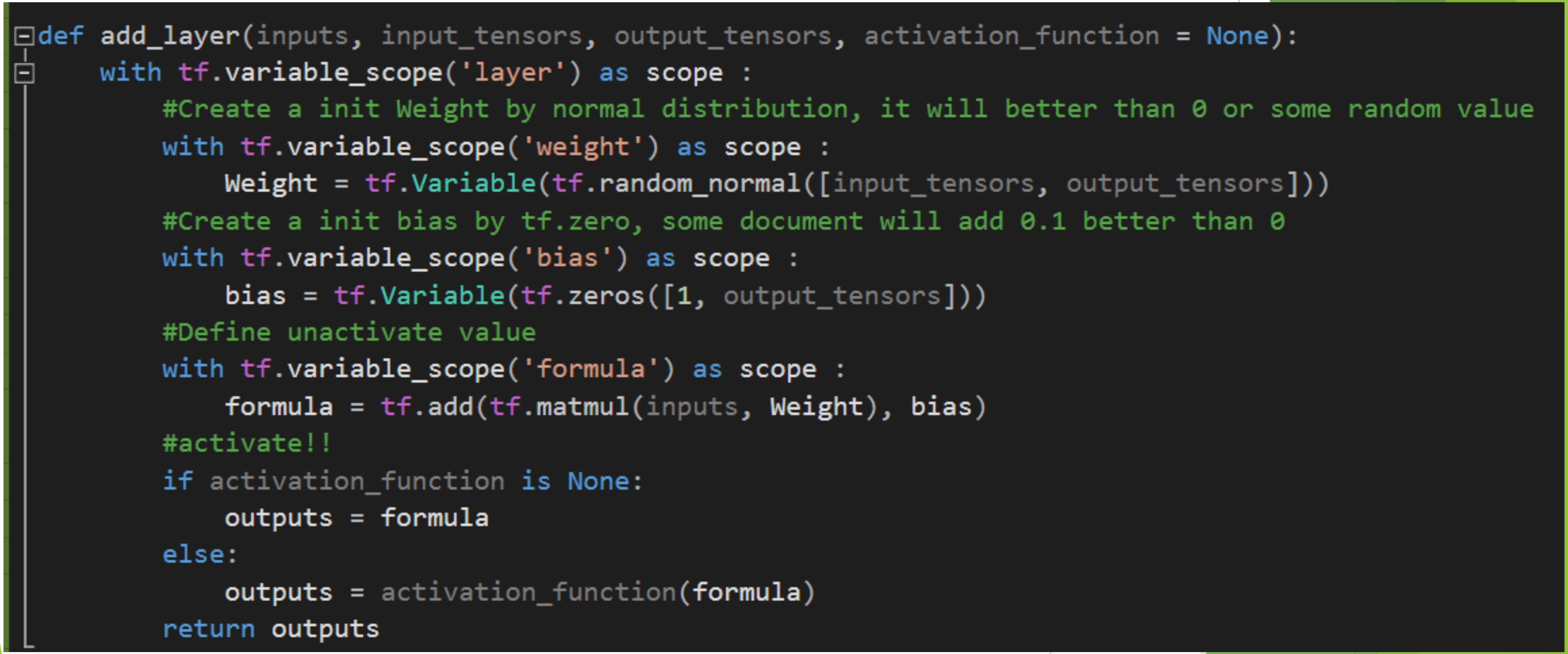
variable_scope and name_scope

- Both variable_scope and name_scope have the same effect on all operations as well as variables created using tf.Variable
- name_scope is ignored by tf.get_variable



```
with tf.variable_scope("C"):
    with tf.name_scope("temp"):
        value = tf.get_variable("value", [1], dtype = tf.float32)
        add_output = 1.0 + value
assert value.name == "C/value:0"
assert add_output.op.name == "C/temp/add"
```

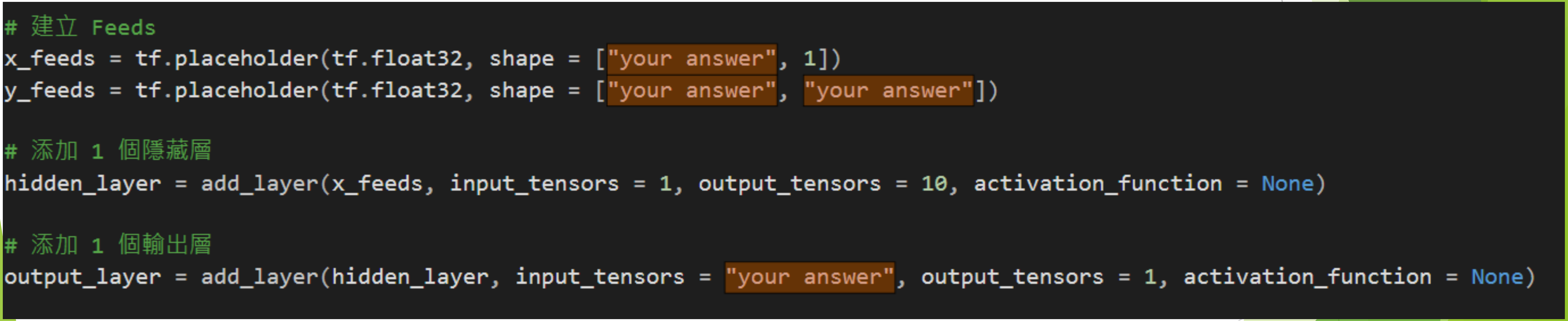
add layer



```
def add_layer(inputs, input_tensors, output_tensors, activation_function = None):
    with tf.variable_scope('layer') as scope :
        #Create a init Weight by normal distribution, it will better than 0 or some random value
        with tf.variable_scope('weight') as scope :
            Weight = tf.Variable(tf.random_normal([input_tensors, output_tensors]))
        #Create a init bias by tf.zero, some document will add 0.1 better than 0
        with tf.variable_scope('bias') as scope :
            bias = tf.Variable(tf.zeros([1, output_tensors]))
        #Define unactivate value
        with tf.variable_scope('formula') as scope :
            formula = tf.add(tf.matmul(inputs, Weight), bias)
        #activate!!
        if activation_function is None:
            outputs = formula
        else:
            outputs = activation_function(formula)
        return outputs
```

Homework

Simple network



```
# 建立 Feeds
x_feeds = tf.placeholder(tf.float32, shape = ["your answer", 1])
y_feeds = tf.placeholder(tf.float32, shape = ["your answer", "your answer"])

# 添加 1 個隱藏層
hidden_layer = add_layer(x_feeds, input_tensors = 1, output_tensors = 10, activation_function = None)

# 添加 1 個輸出層
output_layer = add_layer(hidden_layer, input_tensors = "your answer", output_tensors = 1, activation_function = None)
```

- Install tensorflow
- https://drive.google.com/file/d/1Xxkr62qn6CAfpdyk0oA_xbJFp4DCrm9k/view?usp=sharing
- Grade : 7 (one week)
- Grade : 6 (two week)
- ALL student completed in two week , I will get you all 8!!