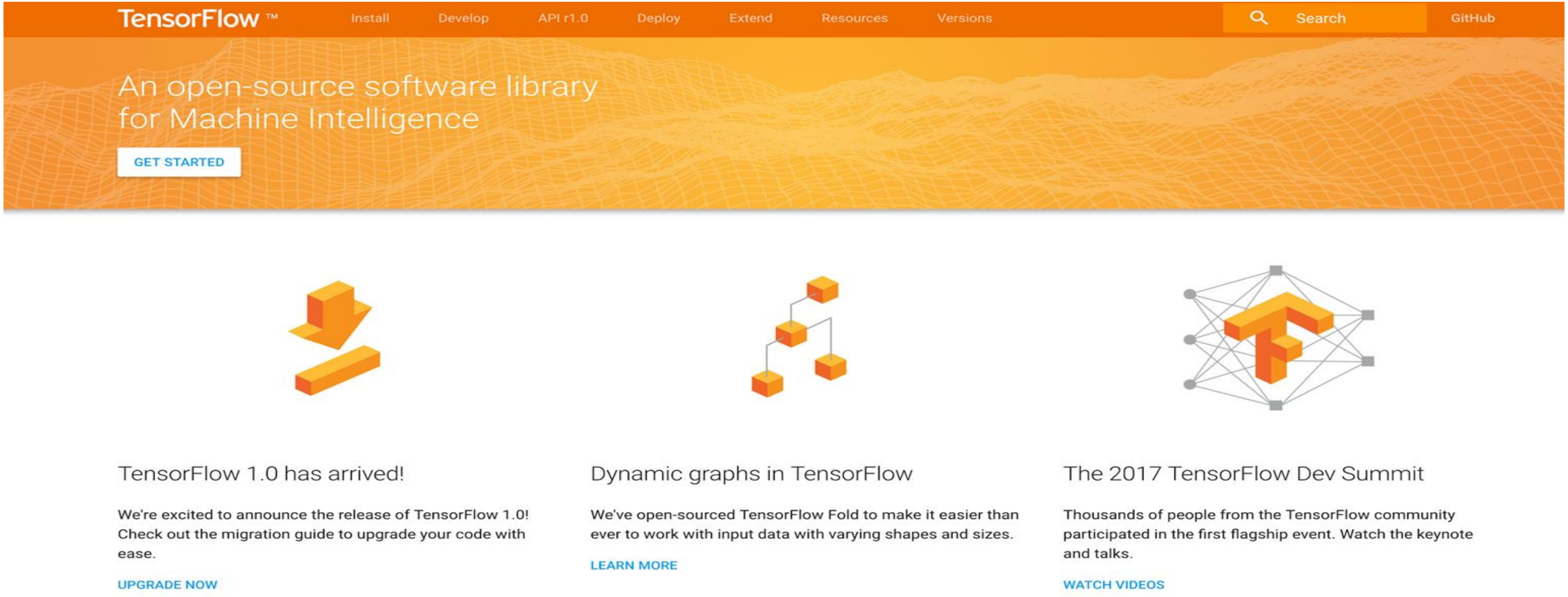
TensorFlow Basics

Lecture 01



An open-source software library for Machine Intelligence

TensorFlow 1.0 has arrived!

We're excited to announce the release of TensorFlow 1.0! Check out the migration guide to upgrade your code with ease.

Dynamic graphs in TensorFlow

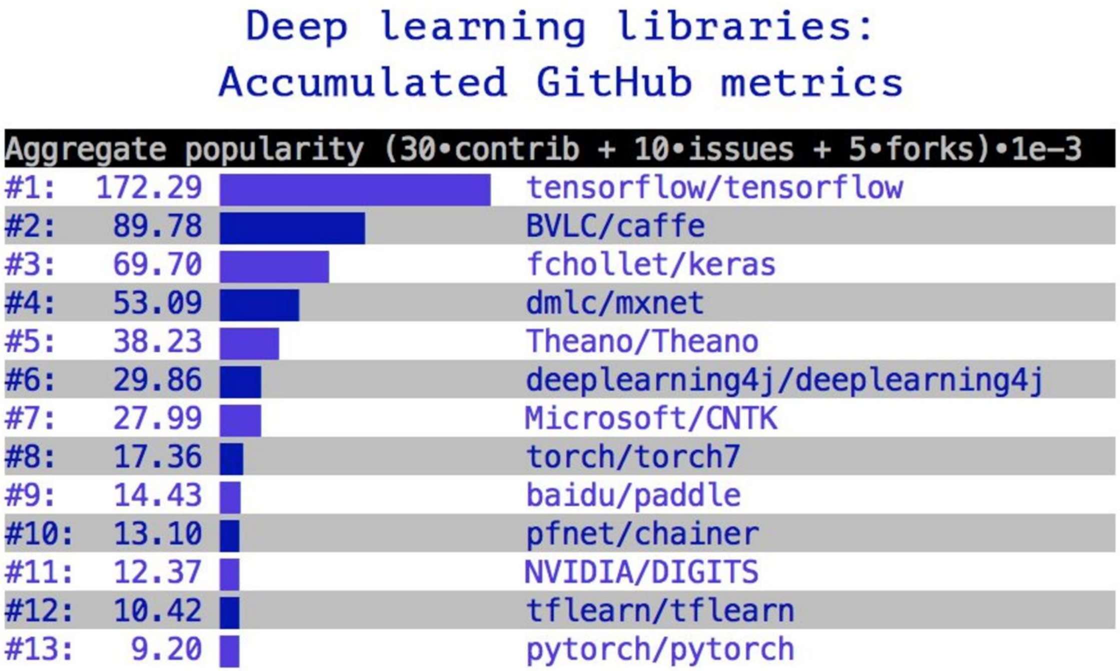
We've open-sourced TensorFlow Fold to make it easier than ever to work with input data with varying shapes and sizes.

The 2017 TensorFlow Dev Summit

Thousands of people from the TensorFlow community participated in the first flagship event. Watch the keynote and talks.

https://www.tensorflow.org

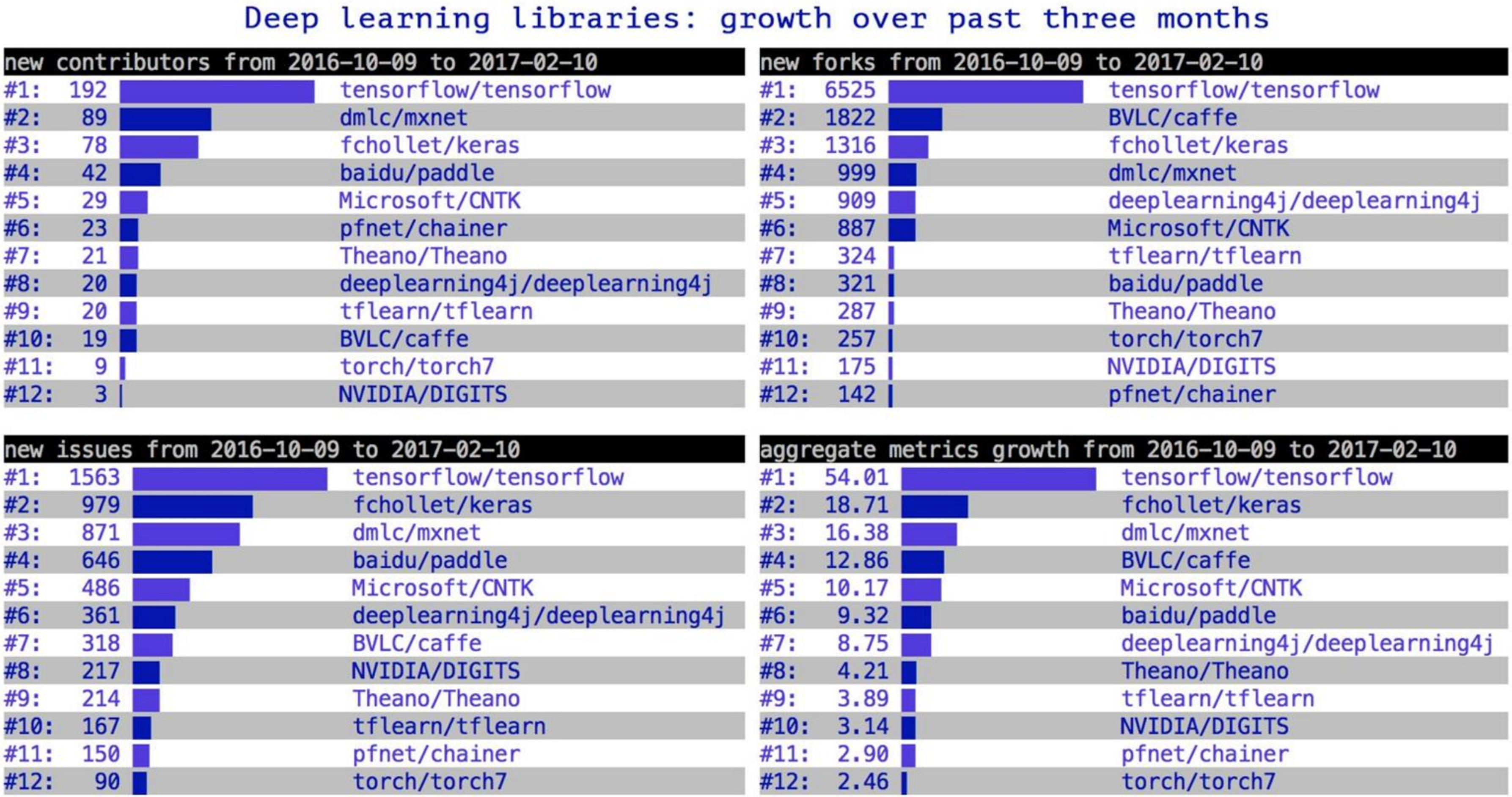
TensorFlow



Deep learning libraries:
Accumulated GitHub metrics

Aggregate popularity (30•contrib + 10•issues + 5•forks)•1e-3		
#1:	172.29	tensorflow/tensorflow
#2:	89.78	BVLC/caffe
#3:	69.70	fchollet/keras
#4:	53.09	dmlc/mxnet
#5:	38.23	Theano/Theano
#6:	29.86	deeplearning4j/deeplearning4j
#7:	27.99	Microsoft/CNTK
#8:	17.36	torch/torch7
#9:	14.43	baidu/paddle
#10:	13.10	pfnet/chainer
#11:	12.37	NVIDIA/DIGITS
#12:	10.42	tflearn/tflearn
#13:	9.20	pytorch/pytorch

https://twitter.com/fchollet/status/830499993450450944/



Deep learning libraries: growth over past three months

```
new contributors from 2016-10-09 to 2017-02-10
#1:   192  tensorflow/tensorflow
#2:    89  dmlc/mxnet
#3:    78  fchollet/keras
#4:    42  baidu/paddle
#5:    29  Microsoft/CNTK
#6:    23  pfnet/chainer
#7:    21  Theano/Theano
#8:    20  deeplearning4j/deeplearning4j
#9:    20  tflearn/tflearn
#10:   19  BVLC/caffe
#11:    9  torch/torch7
#12:    3  NVIDIA/DIGITS
```

```
new forks from 2016-10-09 to 2017-02-10
#1:   6525  tensorflow/tensorflow
#2:   1822  BVLC/caffe
#3:   1316  fchollet/keras
#4:    999  dmlc/mxnet
#5:    909  deeplearning4j/deeplearning4j
#6:    887  Microsoft/CNTK
#7:    324  tflearn/tflearn
#8:    321  baidu/paddle
#9:    287  Theano/Theano
#10:   257  torch/torch7
#11:   175  NVIDIA/DIGITS
#12:   142  pfnet/chainer
```

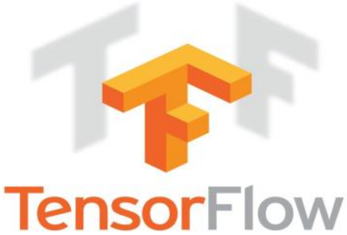
```
new issues from 2016-10-09 to 2017-02-10
#1:   1563  tensorflow/tensorflow
#2:    979  fchollet/keras
#3:    871  dmlc/mxnet
#4:    646  baidu/paddle
#5:    486  Microsoft/CNTK
#6:    361  deeplearning4j/deeplearning4j
#7:    318  BVLC/caffe
#8:    217  NVIDIA/DIGITS
#9:    214  Theano/Theano
#10:   167  tflearn/tflearn
#11:   150  pfnet/chainer
#12:    90  torch/torch7
```

```
aggregate metrics growth from 2016-10-09 to 2017-02-10
#1:   54.01  tensorflow/tensorflow
#2:   18.71  fchollet/keras
#3:   16.38  dmlc/mxnet
#4:   12.86  BVLC/caffe
#5:   10.17  Microsoft/CNTK
#6:    9.32  baidu/paddle
#7:    8.75  deeplearning4j/deeplearning4j
#8:    4.21  Theano/Theano
#9:    3.89  tflearn/tflearn
#10:   3.14  NVIDIA/DIGITS
#11:   2.90  pfnet/chainer
#12:   2.46  torch/torch7
```

https://twitter.com/fchollet/status/830499993450450944/

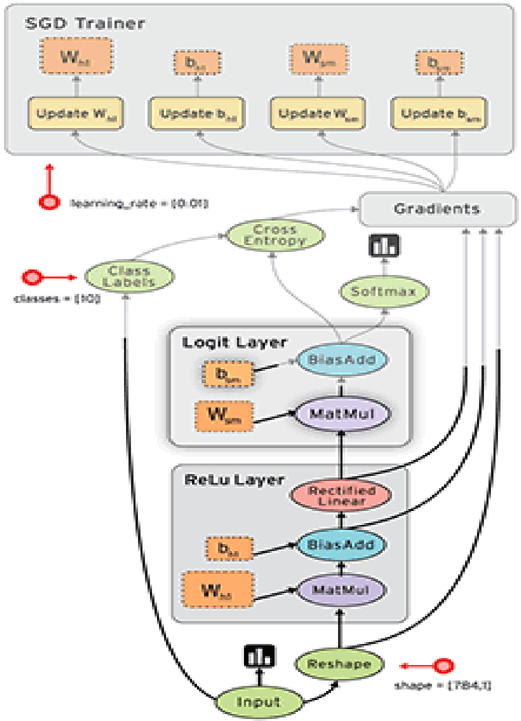
TensorFlow

- TensorFlow™ is an **open source software library** for **numerical computation** using **data flow graphs**.
- Python



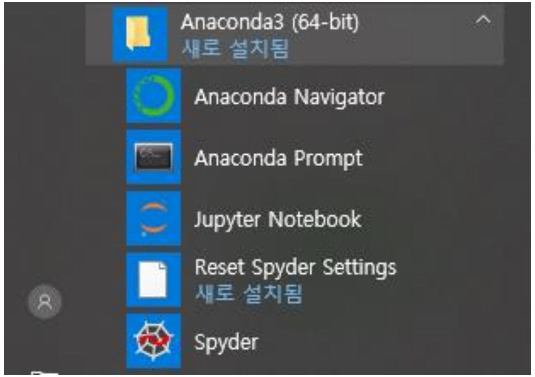
What is a Data Flow Graph?

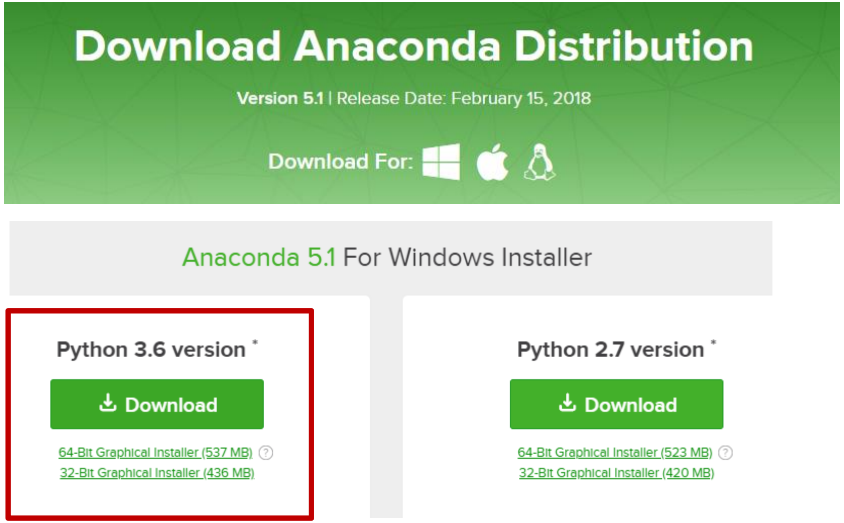
- Nodes in the graph represent mathematical operations
- Edges represent the multidimensional data arrays (tensors) communicated between them.



Installing Anaconda

- Installing Anaconda




Download Anaconda Distribution

Version 5.1 | Release Date: February 15, 2018

Download For:

Anaconda 5.1 For Windows Installer

Python 3.6 version *

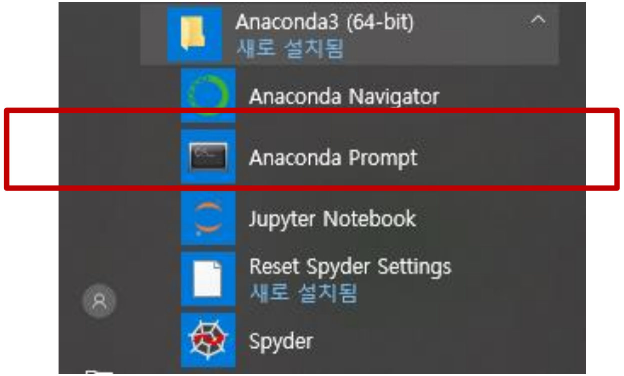
64-Bit Graphical Installer (537 MB)
32-Bit Graphical Installer (436 MB)

Python 2.7 version *

64-Bit Graphical Installer (523 MB)
32-Bit Graphical Installer (420 MB)

https://www.anaconda.com/download/#windows

Installing TensorFlow

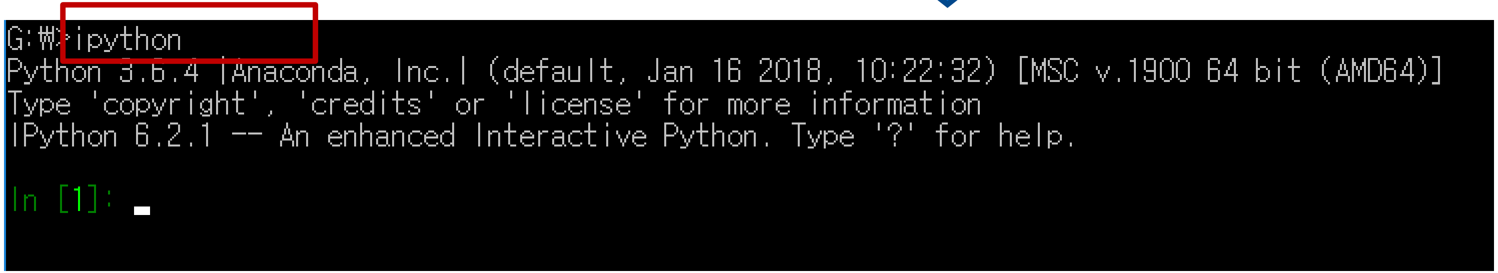


※ pip
파이썬 패키지를 관리하는 소프트웨어

G:\>pip install tensorflow

or

G:\>pip install --upgrade tensorflow

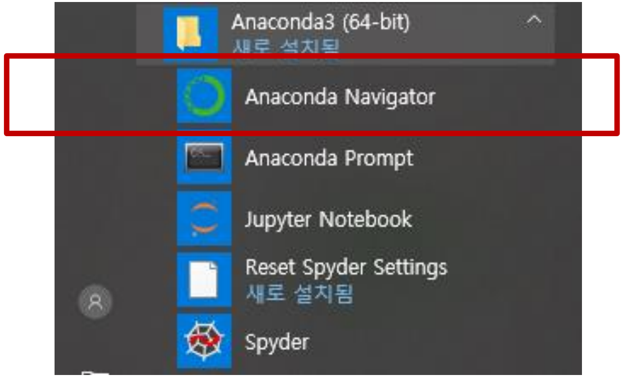


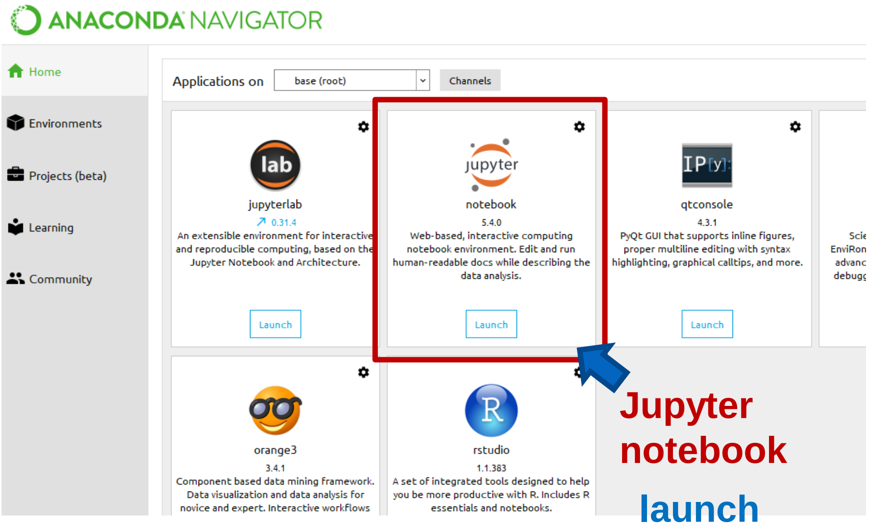
```
G:₩>ipython
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
```

Test in Anaconda Prompt

```
In [1]: import tensorflow as tf

In [2]: tf.__version__
Out[2]: '1.7.0‘
In [3]: hello = tf.constant("Hello, TensorFlow!")

In [4]: sess = tf.Session()

In [5]: print(sess.run(hello))
b'Hello, TensorFlow!'

In [6]: exit
```

Test in Anaconda Navigator





An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

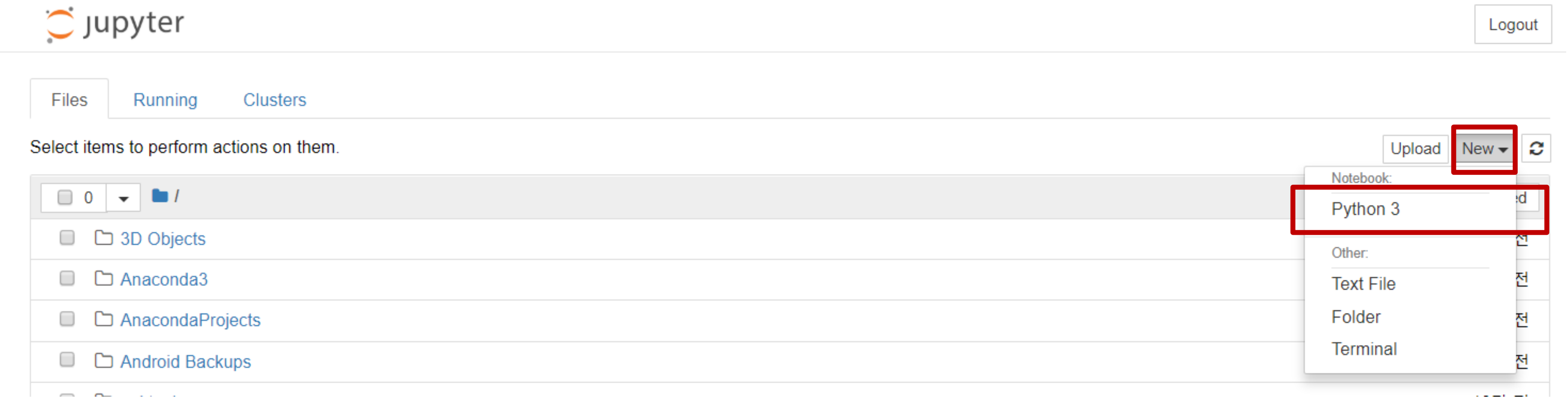
Jupyter notebook

launch

※ Jupyter notebook

머신러닝이나 데이터분석 용도로 파이썬을 사용하는 경우에 주로 사용하는 툴로써 가벼우며 코드를 실행 및 수정이 간편

Test in Anaconda Navigator



Select items to perform actions on them.

3D Objects

Anaconda3

AnacondaProjects

Android Backups

Test in Jupyter notebook

```
In [1]: import tensorflow as tf

In [2]: tf.__version__
Out[2]: '1.7.0‘
In [3]: hello = tf.constant("Hello, TensorFlow!")

In [4]: sess = tf.Session()

In [5]: print(sess.run(hello))
b'Hello, TensorFlow!'
```

shift + enter

b’String’ ‘**b**’ indicates *Bytes literals*

ex01_1.ipynb

Test in Jupyter notebook (1/4)

```
import tensorflow as tf
3 # a rank 0 tensor; this is a scalar with shape []
3

[1. ,2., 3.] # a rank 1 tensor; this is a vector with shape [3]
[1.0, 2.0, 3.0]

[[1., 2., 3.], [4., 5., 6.]] # a rank 2 tensor; a matrix with shape [2, 3]
[[1.0, 2.0, 3.0], [4.0, 5.0, 6.0]]

[[[1., 2., 3.]], [[7., 8., 9.]]] # a rank 3 tensor with shape [2, 1, 3]
[[[1.0, 2.0, 3.0]], [[7.0, 8.0, 9.0]]]

node1 = tf.constant(3.0, tf.float32)  # 상수를 할당, 변수를 할당할 때는  tf.Variable() 사용
Node2 = tf.constant(4.0) # 상수, also tf.float32 implicitly
node3 = tf.add(node1, node2) # 연산자

print("노드 1:", node1, "노드 2:", node2)
print("노드 3: ", node3)
노드 1: Tensor("Const_18:0", shape=(), dtype=float32) 노드 2: Tensor("Const_19:0", shape=(), dtype=float32)
노드 3:  Tensor("Add_8:0", shape=(), dtype=float32)
```

ex01_1.ipynb

Test in Jupyter notebook (2/4)

```
① sess = tf.Session()
② print("sess.run(node1, node2): ", sess.run([node1, node2]))
print("sess.run(node3): ", sess.run(node3))
sess.run(node1, node2):  [3.0, 4.0]
sess.run(node3):  7.0

a = tf.placeholder(tf.float32) # 플레이스홀더
b = tf.placeholder(tf.float32) # 플레이스홀더
adder_node = a + b  # + provides a shortcut for tf.add(a, b)
print(sess.run(adder_node, feed_dict={a: 3, b: 4.5}))
print(sess.run(adder_node, feed_dict={a: [1,3], b: [2, 4]}))
7.5
[3. 7.]

add_and_triple = adder_node * 3.
print(sess.run(add_and_triple, feed_dict={a: 3, b:4.5}))
22.5
```

ex01_1.ipynb

Test in Jupyter notebook (3/4)

```
import tensorflow as tf
import numpy as np  #행렬 라이브러리 제공

a = tf.constant([1,2,3])
b = tf.constant([[10, 20, 30], [100, 200, 300]])
c = tf.add(a,b)

print(c) # 만약 Session 을 열고 run 하지 않고 c를 프린트 한다면...
1 with tf.Session() as sess:
2     print (sess.run(c))
```

```
Tensor("Add_4:0", shape=(2, 3), dtype=int32)
```

```
[[ 11  22  33]
 [101 202 303]]
```

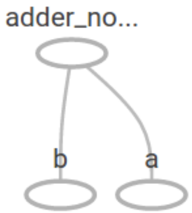
ex01_1.ipynb

Test in Jupyter notebook (4/4)

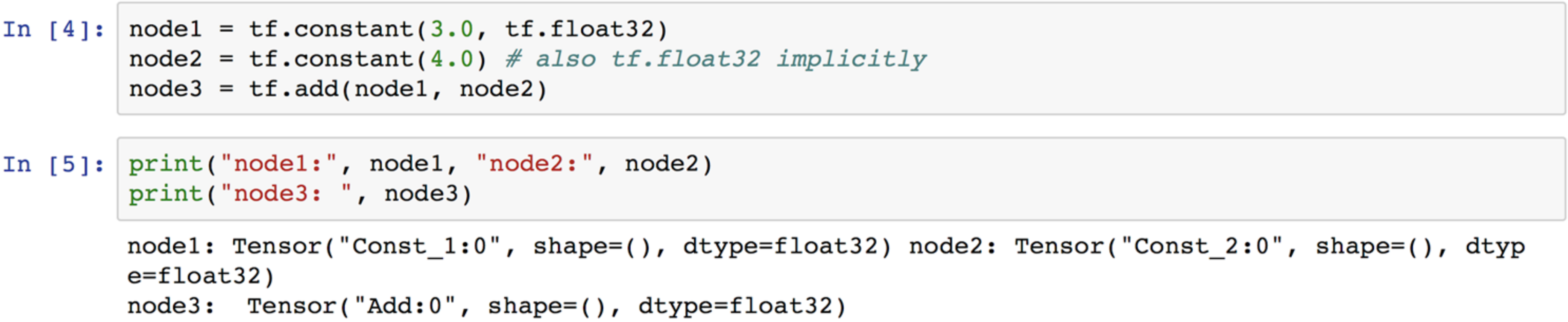
```
import tensorflow as tf
import numpy as np

t = tf.zeros([3,])
① with tf.Session() as sess:
②     print (sess.run(t),"\n")

# tf.random_normal   정규분포 난수로 텐서 생성
# tf.random_uniform  균등분포 난수로 텐서 생성

t1 = tf.random_normal([1] ) #shape, mean, standard deviation
t2 = tf.random_normal([2,2])
t3 = tf.random_uniform([1], -3, 3) #shape, mean, standard deviation
t4 = tf.random_uniform([2,2], -4, 3)

① with tf.Session() as sess:
②     print ("t1 : ", sess.run(t1))
      print ("t2 : ", sess.run(t2), "\n")
      print ("t3 : ", sess.run(t3))
      print ("t4 : ", sess.run(t4))
```

```
[0. 0. 0.]

t1 :  [-1.5805651]
t2 :  [[0.5062811  1.9646689 ]
 [0.31635258 0.5122979 ]]

t3 :  [2.7569304]
t4 :  [[-1.891392   1.7238994]
 [ 2.0100794  1.1356215]]
```

Computational Graph



import tensorflow as tf

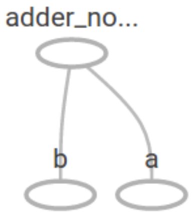


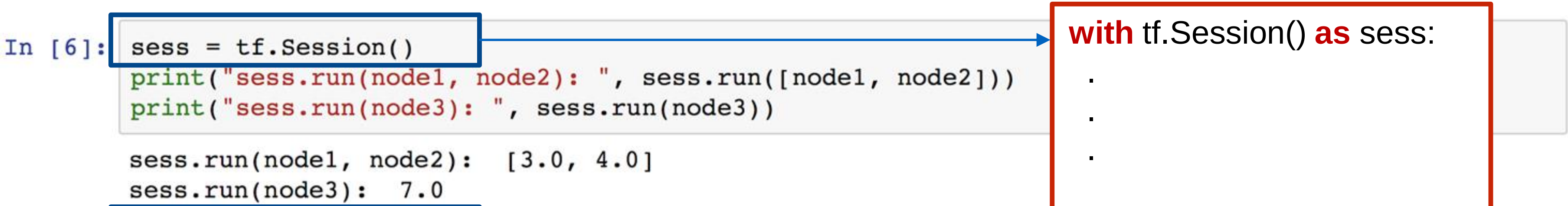
```
In [4]: node1 = tf.constant(3.0, tf.float32)
        node2 = tf.constant(4.0) # also tf.float32 implicitly
        node3 = tf.add(node1, node2)
```

```
In [5]: print("node1:", node1, "node2:", node2)
        print("node3: ", node3)
```

```
node1: Tensor("Const_1:0", shape=(), dtype=float32) node2: Tensor("Const_2:0", shape=(), dtyp
e=float32)
node3:  Tensor("Add:0", shape=(), dtype=float32)
```

Computational Graph





```
In [6]: sess = tf.Session()
        print("sess.run(node1, node2): ", sess.run([node1, node2]))
        print("sess.run(node3): ", sess.run(node3))

        sess.run(node1, node2):  [3.0, 4.0]
        sess.run(node3):  7.0
```

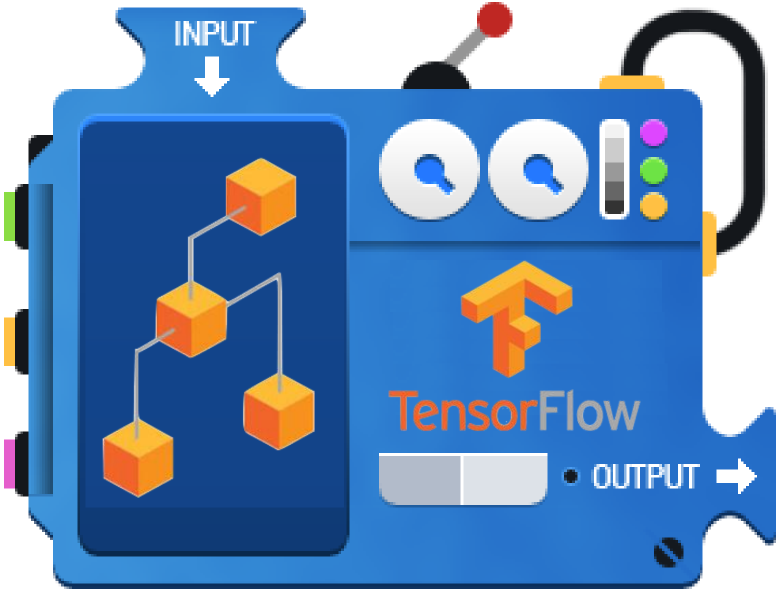
```
sess.close()
```

```
with tf.Session() as sess:
  .
  .
  .
```

TensorFlow Mechanics

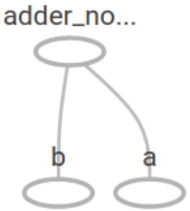
2 feed data and run graph (operation)
sess.run (op)

1 Build graph using
TensorFlow operations

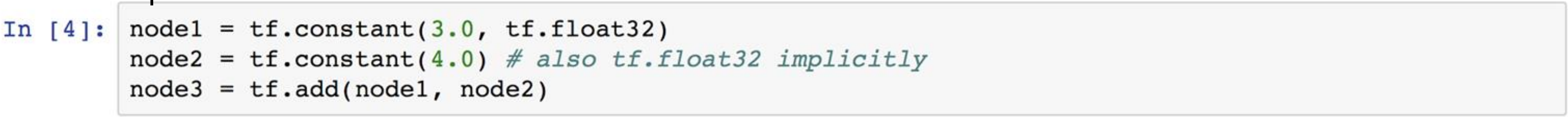


3 update variables
in the graph
(and return values)

Computational Graph



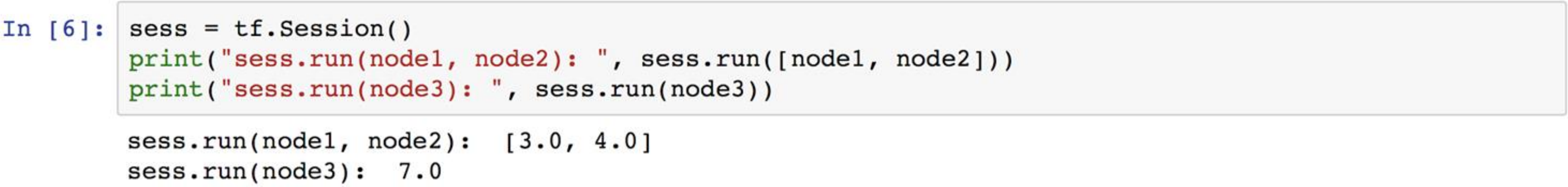
1 Build graph (tensors) using TensorFlow operations



```
In [4]: node1 = tf.constant(3.0, tf.float32)
        node2 = tf.constant(4.0) # also tf.float32 implicitly
        node3 = tf.add(node1, node2)
```

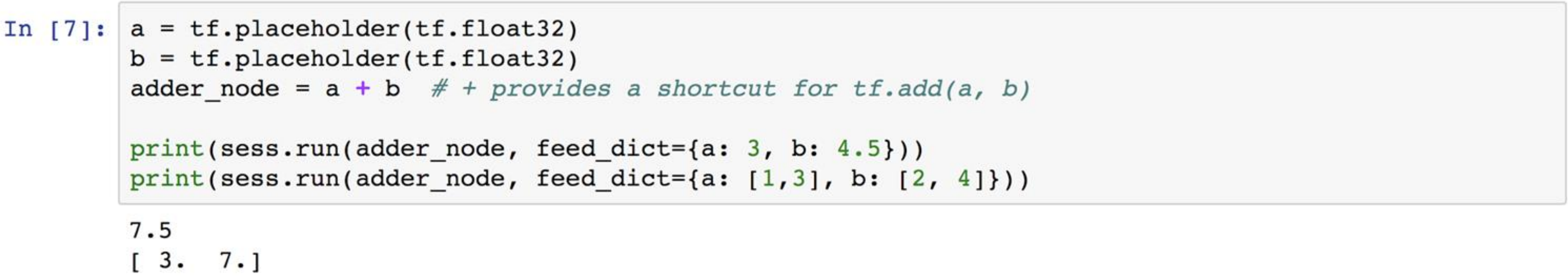
2 feed data and run graph (operation)
sess.run (op)

3 update variables in the graph
(and return values)

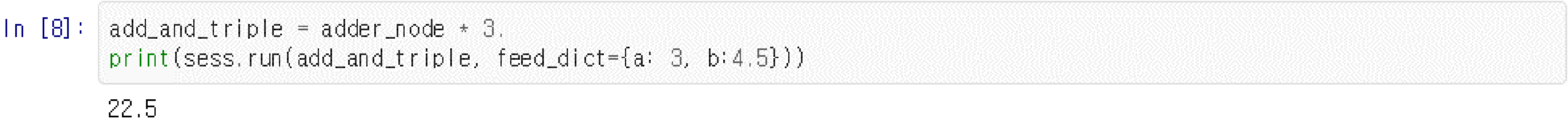


```
In [6]: sess = tf.Session()
        print("sess.run(node1, node2): ", sess.run([node1, node2]))
        print("sess.run(node3): ", sess.run(node3))

        sess.run(node1, node2):  [3.0, 4.0]
        sess.run(node3):  7.0
```

Placeholder



```
In [7]: 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, feed_dict={a: 3, b: 4.5}))
        print(sess.run(adder_node, feed_dict={a: [1,3], b: [2, 4]}))

        7.5
        [ 3.  7.]
```



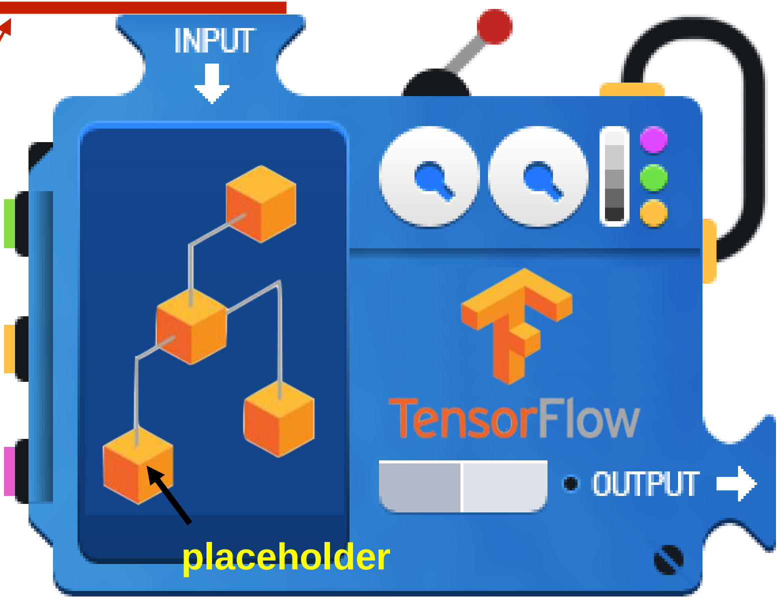
```
In [8]: add_and_triple = adder_node * 3.
        print(sess.run(add_and_triple, feed_dict={a: 3, b:4.5}))

        22.5
```

TensorFlow Mechanics

2 feed data and run graph (operation)
sess.run (op, feed_dict={x: x_data})

placeholder

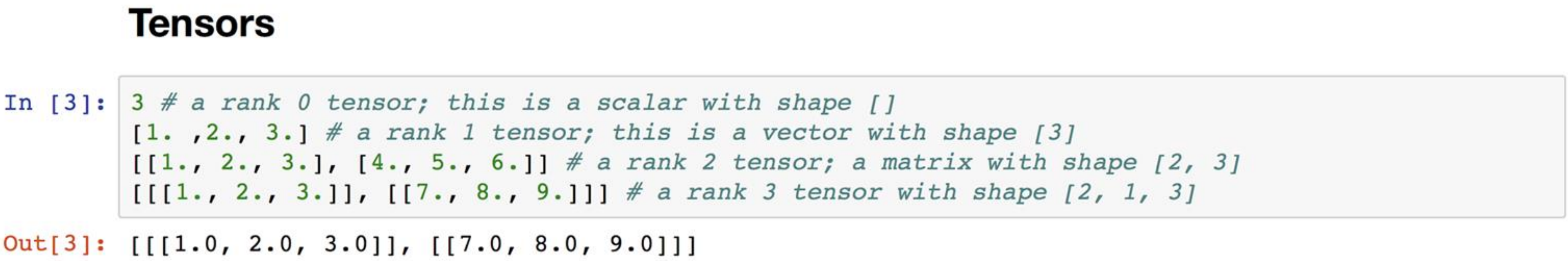
1 Build graph using TensorFlow operations



3 update variables in the graph (and return values)

WWW.MATHWAREHOUSE.COM

Everything is **Tensor**

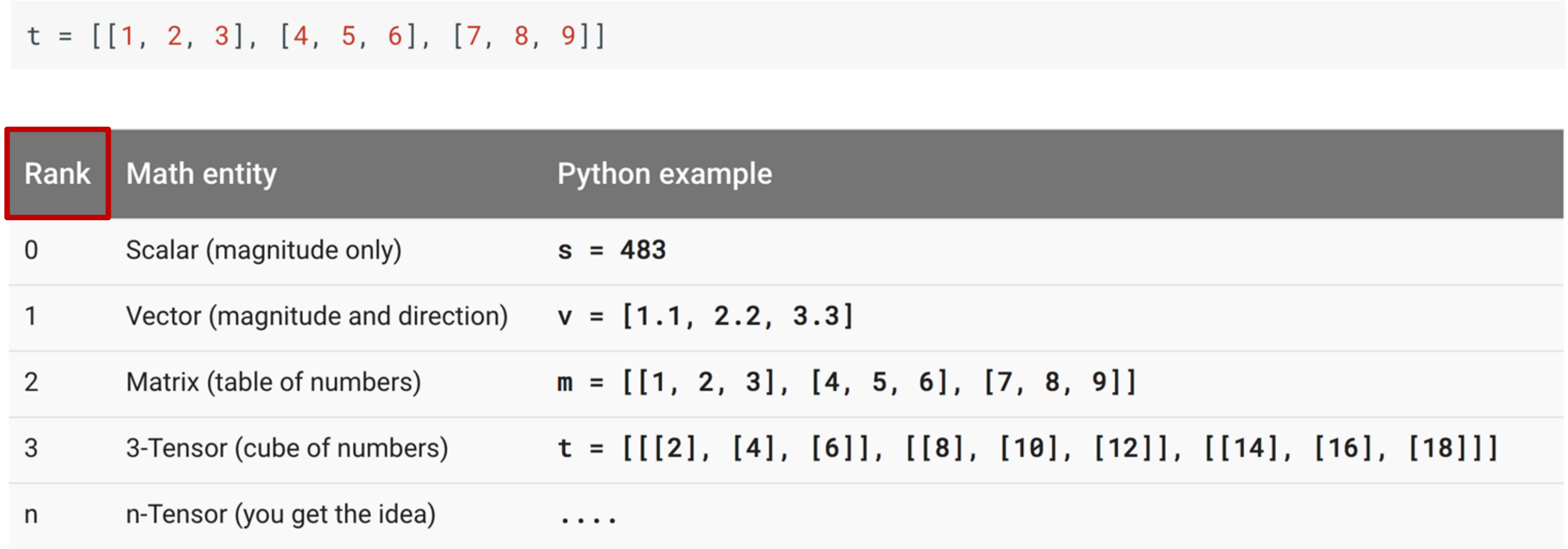


Tensors

```
In [3]: 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]

Out[3]: [[[1.0, 2.0, 3.0]], [[7.0, 8.0, 9.0]]]
```

```
t = tf.Constant([1., 2., 3.])
```

Tensor Ranks



```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Rank	Math entity	Python example
0	Scalar (magnitude only)	`s = 483`
1	Vector (magnitude and direction)	`v = [1.1, 2.2, 3.3]`
2	Matrix (table of numbers)	`m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`
3	3-Tensor (cube of numbers)	`t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]`
n	n-Tensor (you get the idea)	`....`

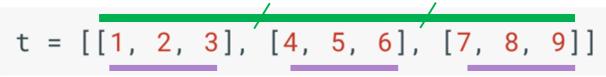
ex01_rank.ipynb

Tensor Ranks

```
import tensorflow as tf
c0 = tf.constant(10) # rank 0
c1 = tf.constant([10]) # rank 0
c2 = tf.constant([1,2])  # rank 1
c3 = tf.constant([1,2,3])  # rank 1
c4 = tf.constant([[1,2,3],[4,5,6]])  # rank 2
c5 = tf.constant([[1,2,3],[4,5,6],[7,8,9]])  # rank 2
c6 = tf.constant([[[2],[4]],[[8],[10]],[[12],[14]],[[18],[20]]])  # rank 3
c7 = tf.constant([[[2],[4],[6]],[[8],[10],[12]],[[14],[16],[18]]])  # rank 3

with tf.Session() as sess:
    print("c0 value : ",sess.run((c0)), ",\t c0 rank : ",sess.run(tf.rank(c0)))

    print('='*50)
    print("c1 value : ",sess.run((c1)), ",\t c1 rank : ",sess.run(tf.rank(c1)))

    print('='*50)
    print("c2 value : ",sess.run((c2)), ",\t c2 rank : ",sess.run(tf.rank(c2)))

    print('='*50)
    print("c3 value : ",sess.run((c3)), ",\t c3 rank : ",sess.run(tf.rank(c3)))

    print('='*50)
    print("c4 value : ")
    print(sess.run((c4)), ",\t c4 rank : ",sess.run(tf.rank(c4)))

    print('='*50)
    print("c5 value : ")
    print(sess.run((c5)), ",\t c5 rank : ",sess.run(tf.rank(c5)))

    print('='*50)
    print("c6 value : ")
    print(sess.run((c6)), ",\t c6 rank : ",sess.run(tf.rank(c6)))

    print('='*50)
    print("c7 value : ")
    print(sess.run((c7)), ",\t c7 rank : ",sess.run(tf.rank(c7)))
```

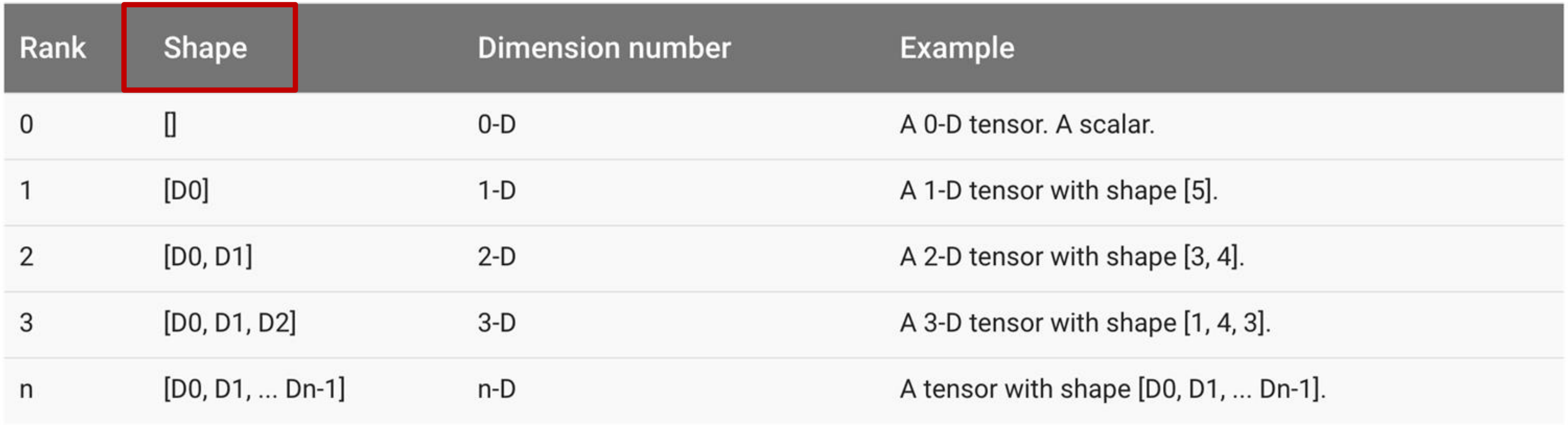
```
c0 value :  10 ,         c0 rank :  0
==================================================
c1 value :  [10] ,       c1 rank :  1
==================================================
c2 value :  [1 2] ,      c2 rank :  1
==================================================
c3 value :  [1 2 3] ,    c3 rank :  1
==================================================
c4 value :
[[1 2 3]
 [4 5 6]] ,              c4 rank :  2
==================================================
c5 value :
[[1 2 3]
 [4 5 6]
 [7 8 9]] ,              c5 rank :  2
==================================================
c6 value :
[[[ 2]
  [ 4]]

 [[ 8]
  [10]]

 [[12]
  [14]]

 [[18]
  [20]]] ,               c6 rank :  3
==================================================
c7 value :
[[[ 2]
  [ 4]
  [ 6]]

 [[ 8]
  [10]
  [12]]

 [[14]
  [16]
  [18]]] ,               c7 rank :  3
```

Tensor Shapes



```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Shapes ?
[3,3]



Rank	Shape	Dimension number	Example
0	[]	0-D	A 0-D tensor. A scalar.
1	[D0]	1-D	A 1-D tensor with shape [5].
2	[D0, D1]	2-D	A 2-D tensor with shape [3, 4].
3	[D0, D1, D2]	3-D	A 3-D tensor with shape [1, 4, 3].
n	[D0, D1, ... Dn-1]	n-D	A tensor with shape [D0, D1, ... Dn-1].

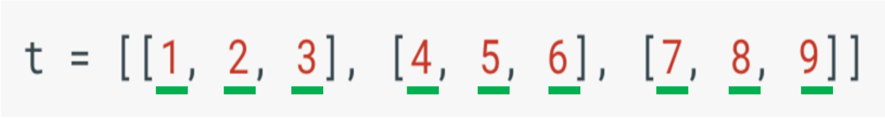
ex01_shape.ipynb

Tensor Shapes

```
import tensorflow as tf
c0 = tf.constant(10) # rank 0
c1 = tf.constant([10]) # rank 0
c2 = tf.constant([1,2])  # rank 1
c3 = tf.constant([1,2,3])  # rank 1
c4 = tf.constant([[1,2,3],[4,5,6]])  # rank 2
c5 = tf.constant([[1,2,3],[4,5,6],[7,8,9]])  # rank 2
c6 = tf.constant([[[2],[4]],[[8],[10]],[[12],[14]],[[18],[20]]])  # rank 3
c7 = tf.constant([[[2],[4],[6]],[[8],[10],[12]],[[14],[16],[18]]])  # rank 3

with tf.Session() as sess:
    print("c0 value : ",sess.run((c0)), ",\t c0 shape : ",sess.run(tf.shape(c0)))

    print('='*50)
    print("c1 value : ",sess.run((c1)), ",\t c1 shape : ",sess.run(tf.shape(c1)))

    print('='*50)
    print("c2 value : ",sess.run((c2)), ",\t c2 shape : ",sess.run(tf.shape(c2)))

    print('='*50)
    print("c3 value : ",sess.run((c3)), ",\t c3 shape : ",sess.run(tf.shape(c3)))

    print('='*50)
    print("c4 value : ")
    print(sess.run((c4)), ",\t c4 shape : ",sess.run(tf.shape(c4)))

    print('='*50)
    print("c5 value : ")
    print(sess.run((c5)), ",\t c5 shape : ",sess.run(tf.shape(c5)))

    print('='*50)
    print("c6 value : ")
    print(sess.run((c6)), ",\t c6 shape : ",sess.run(tf.shape(c6)))

    print('='*50)
    print("c7 value : ")
    print(sess.run((c7)), ",\t c7 shape : ",sess.run(tf.shape(c7)))
```

```
c0 value :  10 ,         c0 shape :  []
==================================================
c1 value :  [10] ,       c1 shape :  [1]
==================================================
c2 value :  [1 2] ,      c2 shape :  [2]
==================================================
c3 value :  [1 2 3] ,    c3 shape :  [3]
==================================================
c4 value :
[[1 2 3]
 [4 5 6]] ,              c4 shape :  [2 3]
==================================================
c5 value :
[[1 2 3]
 [4 5 6]
 [7 8 9]] ,              c5 shape :  [3 3]
==================================================
c6 value :
[[[ 2]
  [ 4]]

 [[ 8]
  [10]]

 [[12]
  [14]]

 [[18]
  [20]]] ,               c6 shape :  [4 2 1]
==================================================
c7 value :
[[[ 2]
  [ 4]
  [ 6]]

 [[ 8]
  [10]
  [12]]

 [[14]
  [16]
  [18]]] ,               c7 shape :  [3 3 1]
```

Tensor Size



```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Size ?

[9]

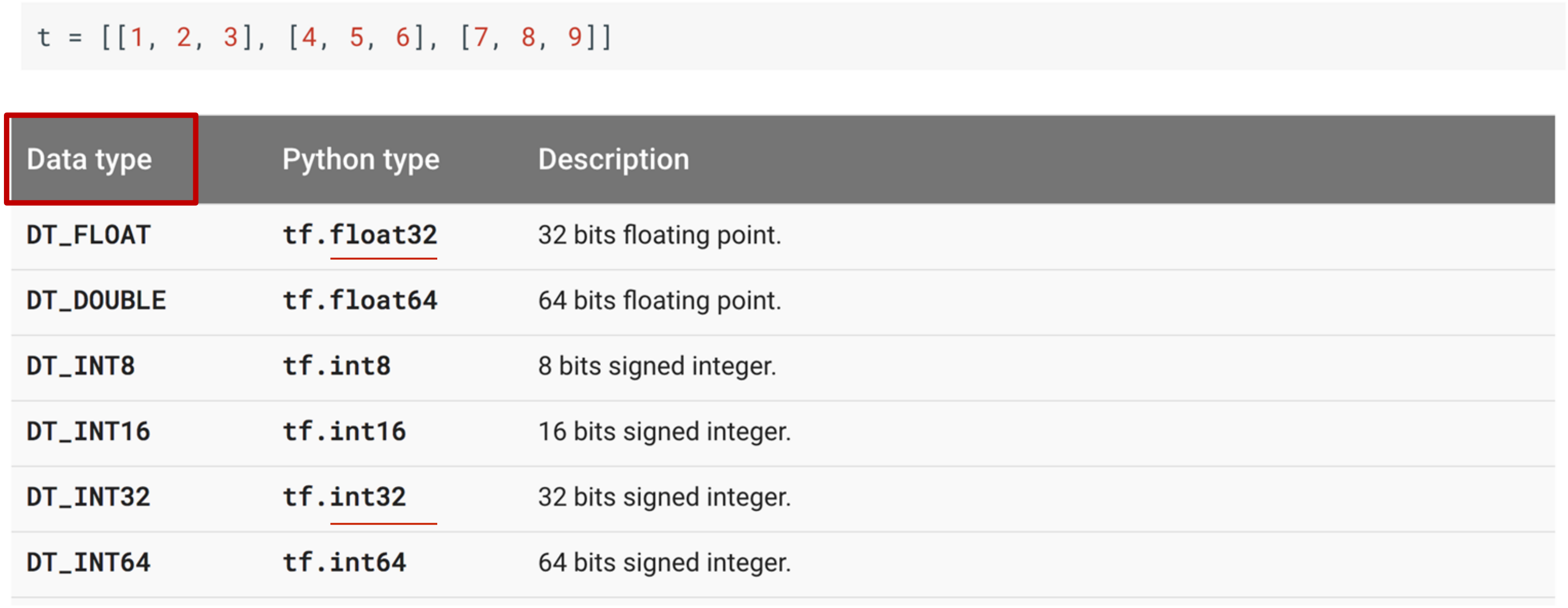
ex01_size.ipynb

Tensor Size

```
import tensorflow as tf
c0 = tf.constant(10) # rank 0
c1 = tf.constant([10]) # rank 0
c2 = tf.constant([1,2])  # rank 1
c3 = tf.constant([1,2,3])  # rank 1
c4 = tf.constant([[1,2,3],[4,5,6]])  # rank 2
c5 = tf.constant([[1,2,3],[4,5,6],[7,8,9]])  # rank 2
c6 = tf.constant([[[2],[4]],[[8],[10]],[[12],[14]],[[18],[20]]])  # rank 3
c7 = tf.constant([[[2],[4],[6]],[[8],[10],[12]],[[14],[16],[18]]])  # rank 3

with tf.Session() as sess:
   print("c0 value : ",sess.run((c0)), ",\t c0 size : ",sess.run(tf.size(c0)))

   print('='*50)
   print("c1 value : ",sess.run((c1)), ",\t c1 size : ",sess.run(tf.size(c1)))

   print('='*50)
   print("c2 value : ",sess.run((c2)), ",\t c2 size : ",sess.run(tf.size(c2)))

   print('='*50)
   print("c3 value : ",sess.run((c3)), ",\t c3 size : ",sess.run(tf.size(c3)))

   print('='*50)
   print("c4 value : ")
   print(sess.run((c4)), ",\t c4 size : ",sess.run(tf.size(c4)))

   print('='*50)
   print("c5 value : ")
   print(sess.run((c5)), ",\t c5 size : ",sess.run(tf.size(c5)))

   print('='*50)
   print("c6 value : ")
   print(sess.run((c6)), ",\t c6 size : ",sess.run(tf.size(c6)))

   print('='*50)
   print("c7 value : ")
   print(sess.run((c7)), ",\t c7 size : ",sess.run(tf.size(c7)))
```

```
c0 value :  10 ,          c0 size :  1
==================================================
c1 value :  [10] ,        c1 size :  1
==================================================
c2 value :  [1 2] ,       c2 size :  2
==================================================
c3 value :  [1 2 3] ,     c3 size :  3
==================================================
c4 value :
[[1 2 3]
 [4 5 6]] ,               c4 size :  6
==================================================
c5 value :
[[1 2 3]
 [4 5 6]
 [7 8 9]] ,               c5 size :  9
==================================================
c6 value :
[[[ 2]
  [ 4]]

 [[ 8]
  [10]]

 [[12]
  [14]]

 [[18]
  [20]]] ,                c6 size :  8
==================================================
c7 value :
[[[ 2]
  [ 4]
  [ 6]]

 [[ 8]
  [10]
  [12]]

 [[14]
  [16]
  [18]]] ,                c7 size :  9
```

Tensor Types



```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Data type	Python type	Description
`DT_FLOAT`	`tf.float32`	32 bits floating point.
`DT_DOUBLE`	`tf.float64`	64 bits floating point.
`DT_INT8`	`tf.int8`	8 bits signed integer.
`DT_INT16`	`tf.int16`	16 bits signed integer.
`DT_INT32`	`tf.int32`	32 bits signed integer.
`DT_INT64`	`tf.int64`	64 bits signed integer.

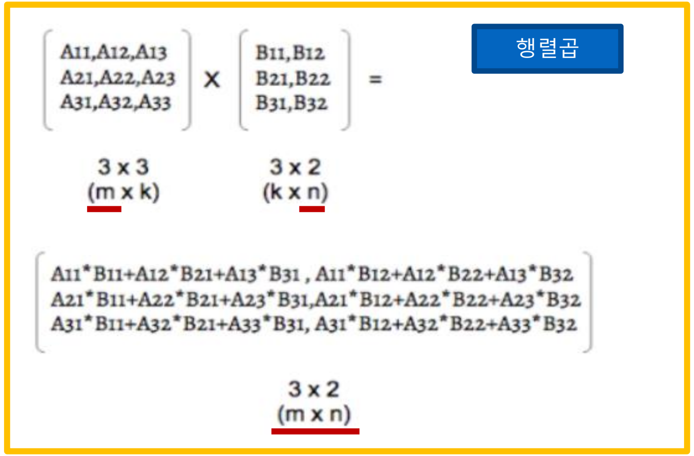
...

ex01_matmul.ipynb

tf.matmul

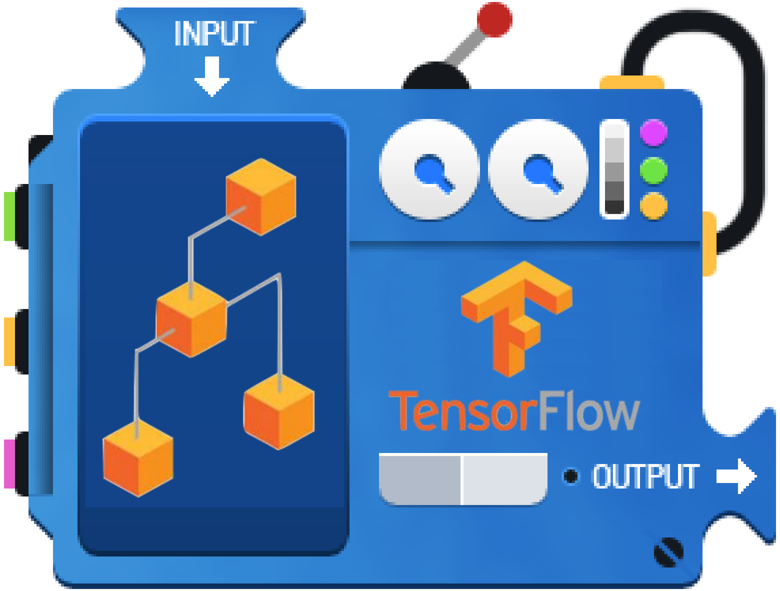
```
import tensorflow as tf
x = tf.constant([[1,2,3],[4,5,6]])
w = tf.constant([[2,3],[5,6],[50,60]])
b = tf.constant([100,200])
expr1 = tf.matmul(x, w)
expr2 = tf.matmul(x, w) + b

with tf.Session() as sess:
    print("\n x : \n",sess.run((x)))
    print("\n w : \n",sess.run((w)))
    print("\n b : \n",sess.run((b)))
    print("\n expr1 : \n",sess.run((expr1)))
    print("\n expr2 : \n",sess.run((expr2)))
```

```
x :
[[1 2 3]
[4 5 6]]

w :
[[ 2  3]
[ 5  6]
[50 60]]

b :
[100 200]

expr1 :
[[162 195]
[333 402]]

expr2 :
[[262 395]
[433 602]]
```


행렬곱
A11,A12,A13
A21,A22,A23
A31,A32,A33
X
B11,B12
B21,B22
B31,B32
=
3 x 3
(m x k)
3 x 2
(k x n)
A11*B11+A12*B21+A13*B31 , A11*B12+A12*B22+A13*B32
A21*B11+A22*B21+A23*B31,A21*B12+A22*B22+A23*B32
A31*B11+A32*B21+A33*B31, A31*B12+A32*B22+A33*B32
3 x 2
(m x n)

TensorFlow Mechanics

2 feed data and run graph (operation)
sess.run (op, feed_dict={x: x_data})



1 Build graph using TensorFlow operations

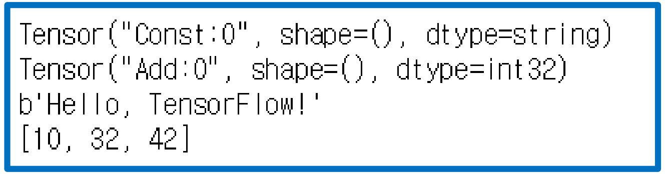
3 update variables in the graph (and return values)

WWW.MATHWAREHOUSE.COM

ex01_2.ipynb

TensorFlow Basic (summary)

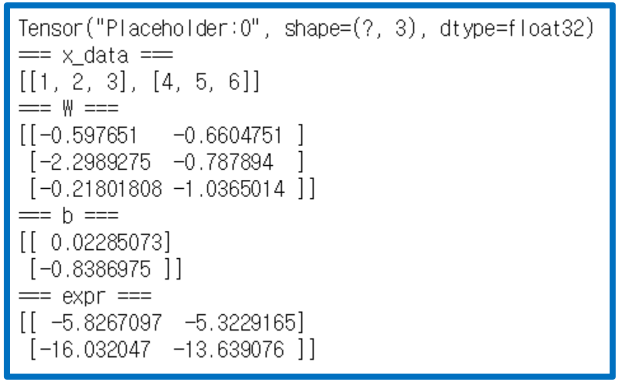
```
# 텐서플로우의 기본적인 구성을 익힙니다.
import tensorflow as tf

# tf.constant  상수를 의미합니다.
hello = tf.constant('Hello, TensorFlow!')
print(hello)

a = tf.constant(10)
b = tf.constant(32)
c = tf.add(a, b)  # a + b 로도 쓸 수 있음
print(c)

# 위에서 변수와 수식들을 정의했지만, 실행이 정의한 시점에서 실행되는 것은 아닙니다.
# 다음처럼 Session 객제와 run 메소드를 사용할 때 계산이 됩니다.
# 따라서 모델을 구성하는 것과, 실행하는 것을 분리하여 프로그램을 깔끔하게 작성할 수 있습니다.
# 그래프를 실행할 세션을 구성합니다.
sess = [redacted]
# sess.run: 설정한 텐서 그래프(변수나 수식 등등)를 실행합니다.
print([redacted](hello))
print([redacted]([a, b, c]))

# 세션을 닫습니다.
sess.close()
```



```
Tensor("Const:0", shape=(), dtype=string)
Tensor("Add:0", shape=(), dtype=int32)
b'Hello, TensorFlow!'
[10, 32, 42]
```

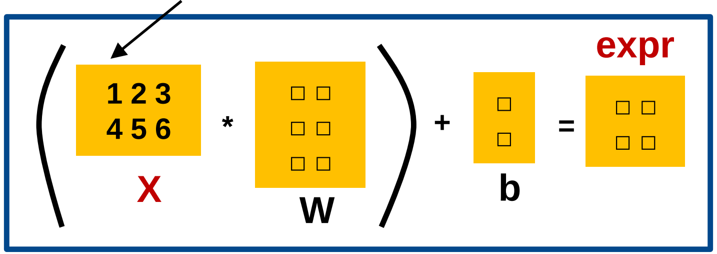
Placeholder & Variable (summary 1/2)

```
# 플레이스홀더와 변수의 개념을 익혀봅니다
import tensorflow as tf

# tf.placeholder: 계산을 실행할 때 입력값을 받는 변수로 사용합니다.
# None 은 크기가 정해지지 않았음을 의미합니다.
X = tf.[redacted](tf.float32, [None, 3]) # None 행 3 열 플레이스홀더 할당
print(X)

# X 플레이스홀더에 넣을 값 입니다.
# 플레이스홀더에서 설정한 것 처럼, 두번째 차원의 요소의 갯수는 3개 입니다.
x_data = [[1, 2, 3], [4, 5, 6]] # 2 행3 열구조

# tf.Variable: 그래프를 계산하면서 최적화 할 변수들입니다.
# tf.random_normal: 각 변수들의 초기값을 정규분포 랜덤 값으로 초기화합니다.
W = tf.[redacted](tf.random_normal([3, 2])) # 3 행 2 열  변수 할당
b = tf.[redacted](tf.random_normal([2, 1])) # 2 행 1 열  변수 할당

# 입력값과 변수들을 계산할 수식을 작성합니다.
# tf.matmul 처럼 mat* 로 되어 있는 함수로 행렬 계산을 수행합니다.
expr = tf.matmul([redacted], [redacted]) + b  # 2 행2 열 구조
```



```
Tensor("Placeholder:0", shape=(?, 3), dtype=float32)
=== x_data ===
[[1, 2, 3], [4, 5, 6]]
=== W ===
[[-0.597651   -0.6604751 ]
 [-2.2989275  -0.787894  ]
 [-0.21801808 -1.0365014 ]]
=== b ===
[[ 0.02285073]
 [-0.8386975 ]]
=== expr ===
[[ -5.8267097  -5.3229165]
 [-16.032047  -13.639076 ]]
```

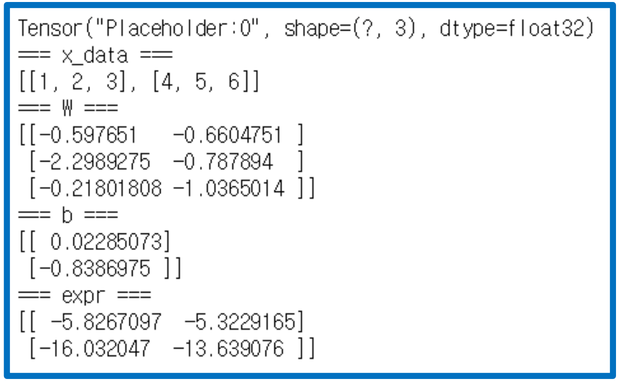
행 : none , 열 : 3



ex01_3.ipynb

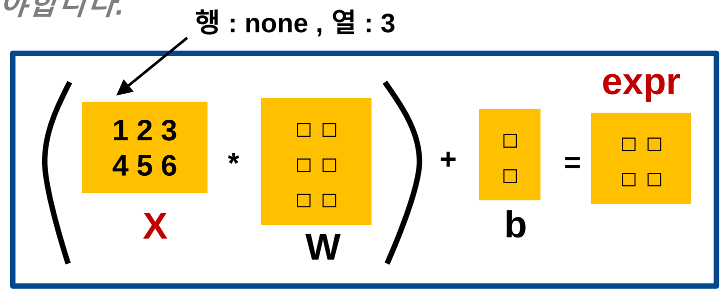
Placeholder & Variable (summary 2/2)

```
# 그래프를 실행할 세션을 구성합니다.
sess = [redacted]

# 위에서 설정한 Variable 들의 값들을 초기화 하기 위해
# 처음에 tf.global_variables_initializer 를 한 번 실행해야 합니다.
[redacted](tf.global_variables_initializer())

print("=== x_data ===")
print(x_data)
print("=== W ===")
print([redacted](W))
print("=== b ===")
print([redacted](b))
print("=== expr ===")

# expr 수식에는 X 라는 입력값이 필요합니다.
# 따라서 expr 실행시에는 이 변수에 대한 실제 입력값을 다음처럼 넣어줘야합니다.
print([redacted](expr, feed_dict={X: [redacted]}))
sess.close()
```



```
Tensor("Placeholder:0", shape=(?, 3), dtype=float32)
=== x_data ===
[[1, 2, 3], [4, 5, 6]]
=== W ===
[[-0.597651   -0.6604751 ]
 [-2.2989275  -0.787894  ]
 [-0.21801808 -1.0365014 ]]
=== b ===
[[ 0.02285073]
 [-0.8386975 ]]
=== expr ===
[[ -5.8267097  -5.3229165]
 [-16.032047  -13.639076 ]]
```

expr = tf.matmul(X, W) + b # 2행2열구조



행 : none , 열 : 3
expr
1 2 3
4 5 6
X
*
□ □
□ □
□ □
W
+
□
□
b
=
□ □
□ □