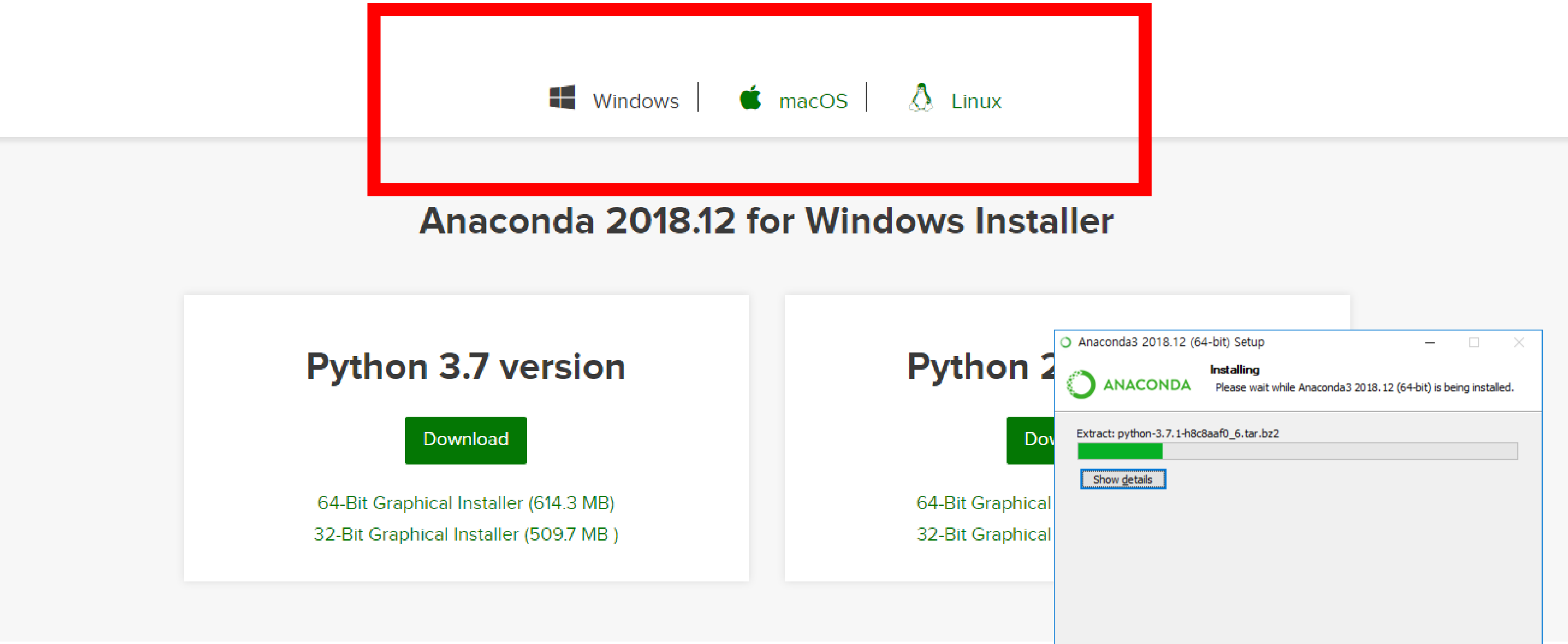
1. configuration and basic

AILAB

Hanyang Univ.

anaconda 설치

- https://www.anaconda.com/distribution/



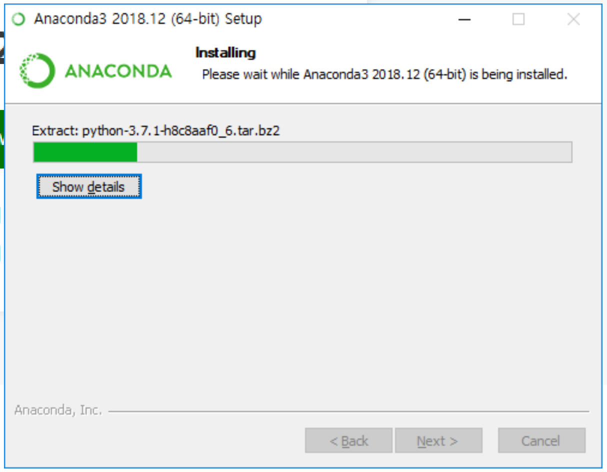
Anaconda 2018.12 for Windows Installer

Python 3.7 version

64-Bit Graphical Installer (614.3 MB)
32-Bit Graphical Installer (509.7 MB)

Python 2

64-Bit Graphical
32-Bit Graphical



Installing
Please wait while Anaconda3 2018.12 (64-bit) is being installed.

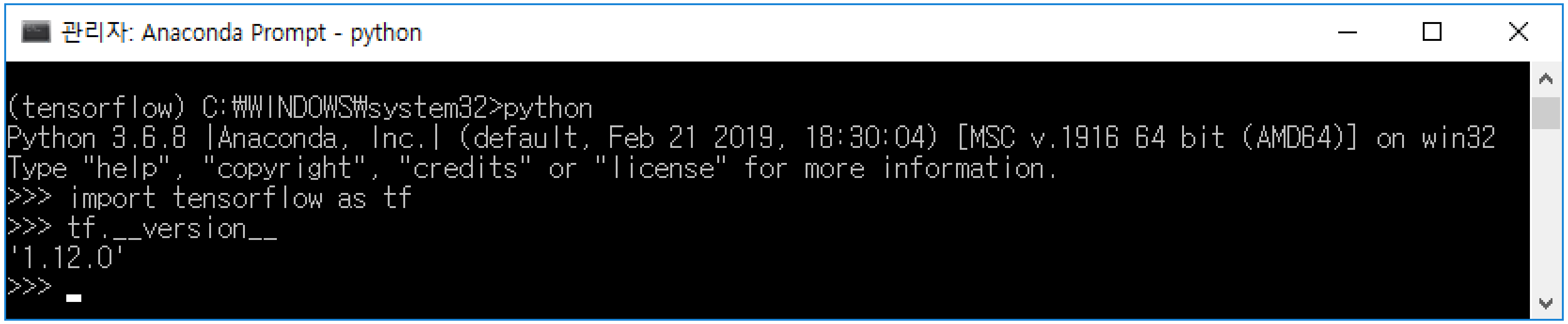
Extract: python-3.7.1-h8c8aaf0_6.tar.bz2

tensorflow 설치

- anaconda prompt 관리자 권한으로 실행
- conda create –n tensorflow python=3.6
- activate tensorflow
- conda install tensorflow==1.12.0

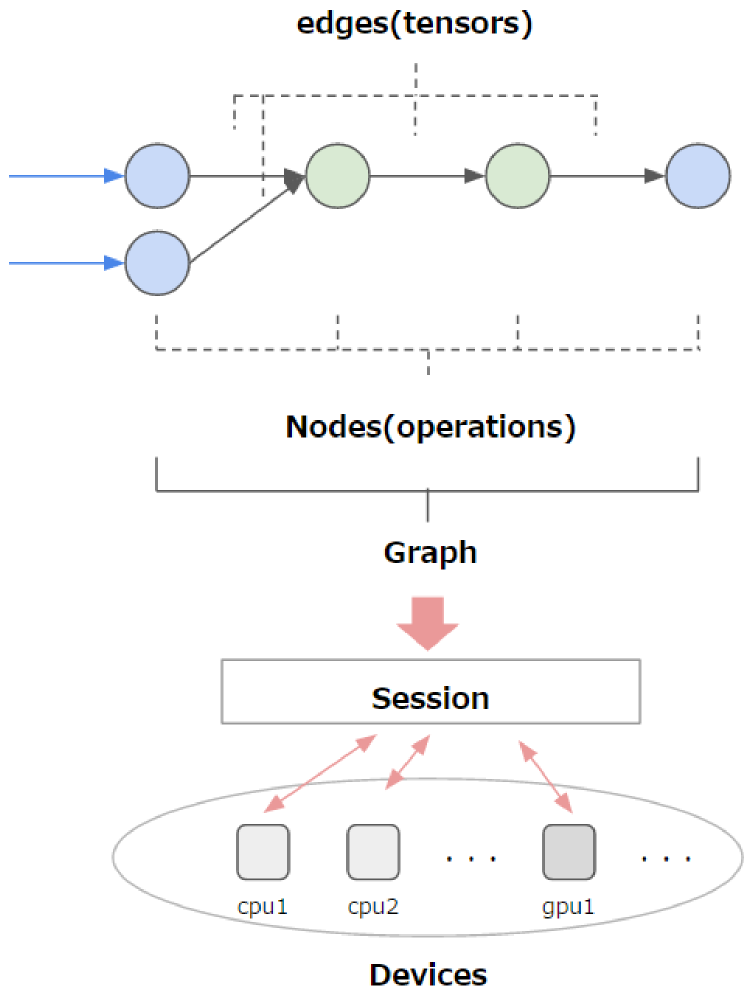
tensorflow 작동확인

- python
- import tensorflow as tf
- tf.__version__

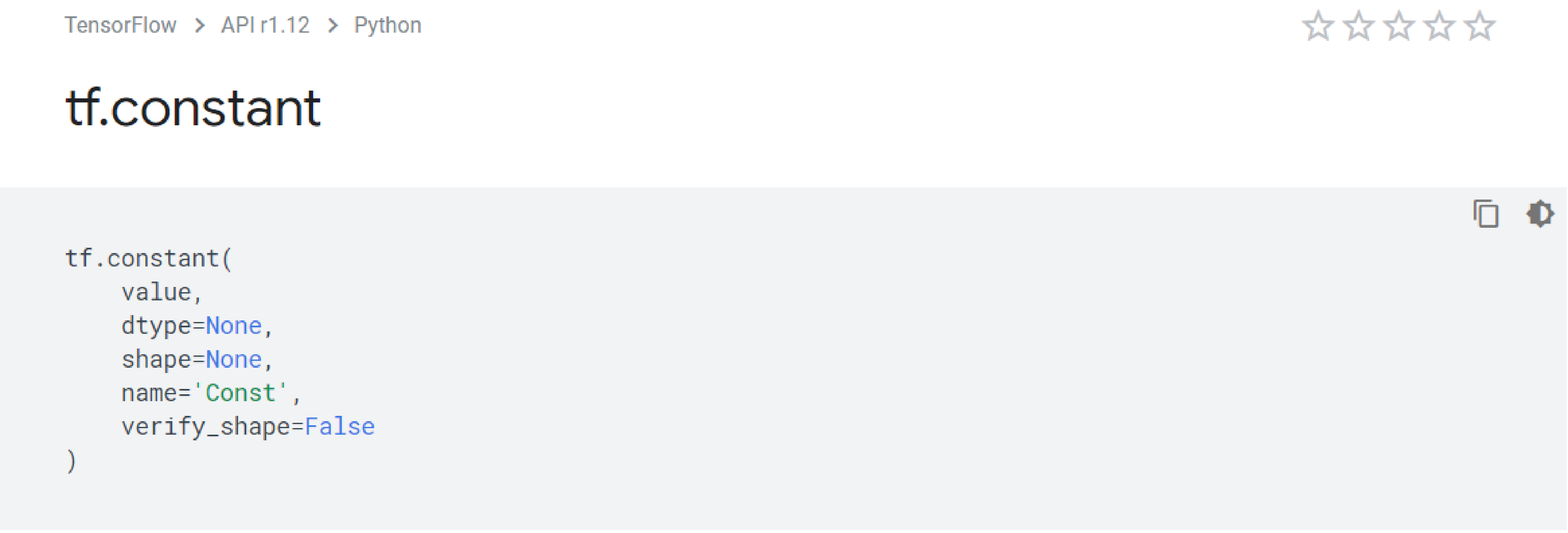


```
(tensorflow) C:₩WINDOWS₩system32>python
Python 3.6.8 |Anaconda, Inc.| (default, Feb 21 2019, 18:30:04) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> tf.__version__
'1.12.0'
>>>
```

tensorflow 기본 - Session



tensorflow 기본 - constant



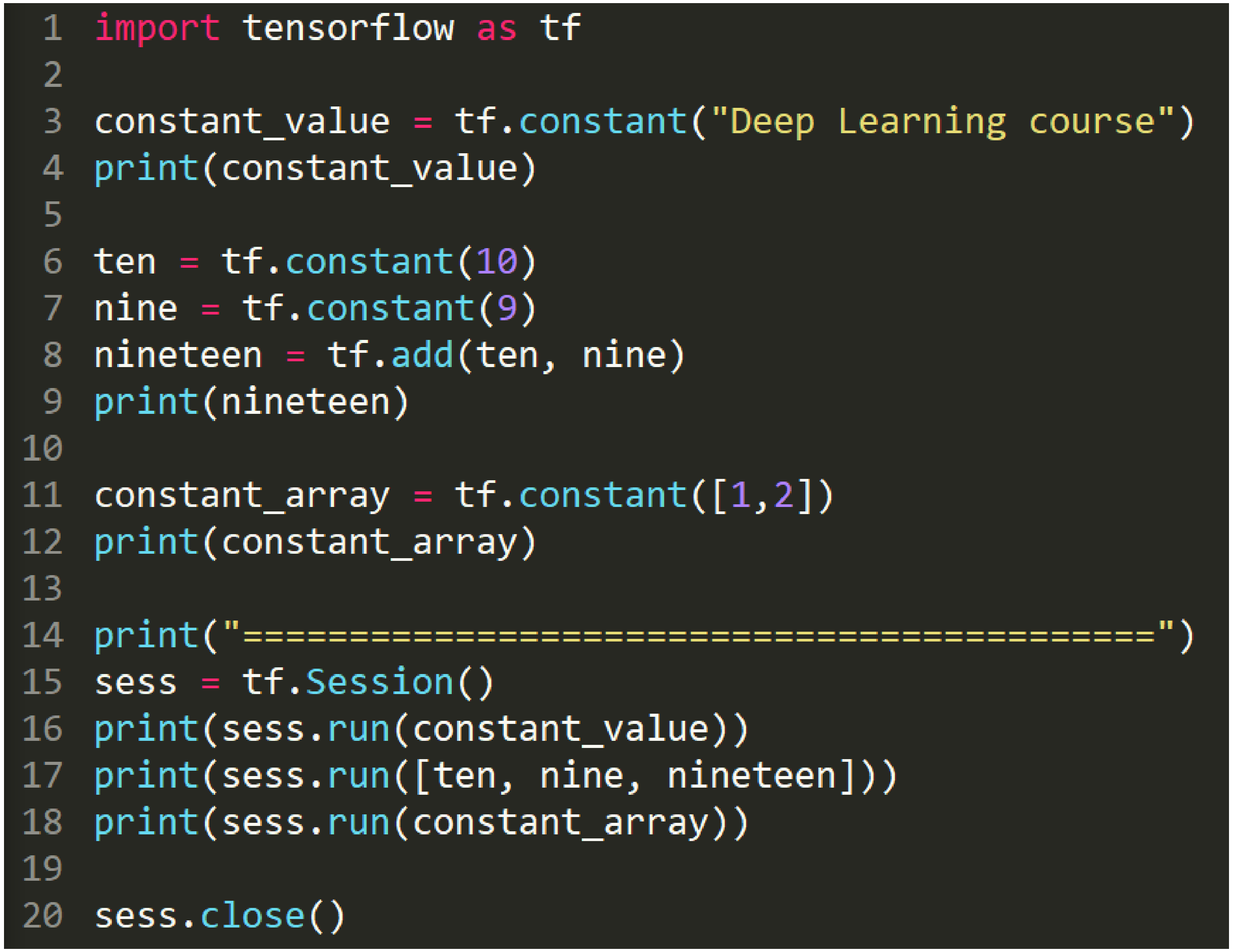
TensorFlow > API r1.12 > Python

tf.constant

```
tf.constant(
    value,
    dtype=None,
    shape=None,
    name='Const',
    verify_shape=False
)
```

https://www.tensorflow.org/versions/r1.12/api_docs/python/tf/constant

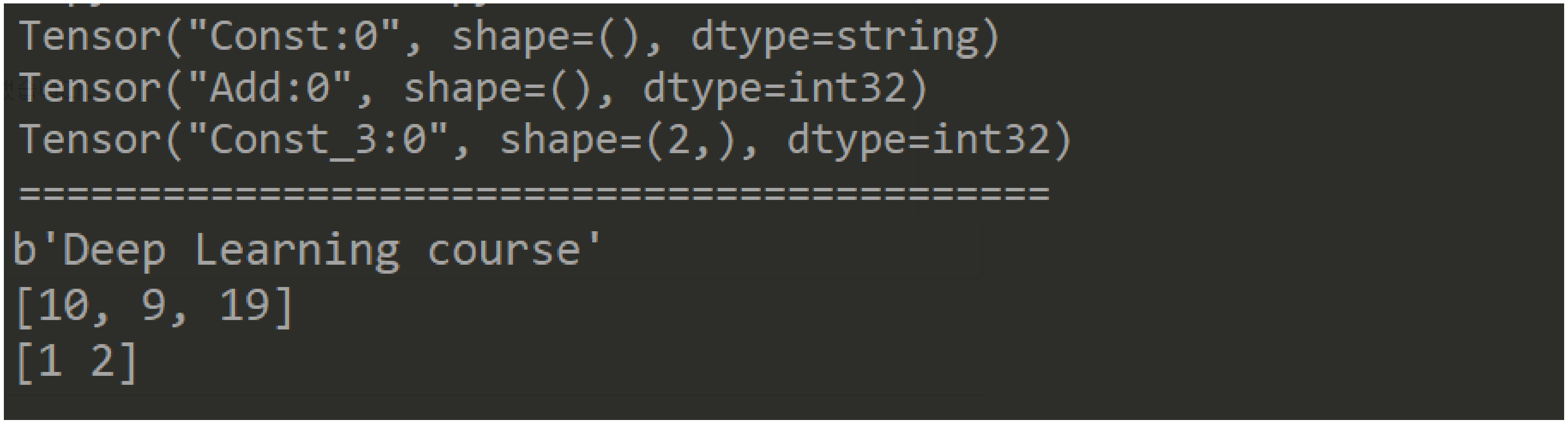
tensorflow 기본 - constant



```python
import tensorflow as tf

constant_value = tf.constant("Deep Learning course")
print(constant_value)

ten = tf.constant(10)
nine = tf.constant(9)
nineteen = tf.add(ten, nine)
print(nineteen)

constant_array = tf.constant([1,2])
print(constant_array)

print("==========================================")
sess = tf.Session()
print(sess.run(constant_value))
print(sess.run([ten, nine, nineteen]))
print(sess.run(constant_array))

sess.close()
```

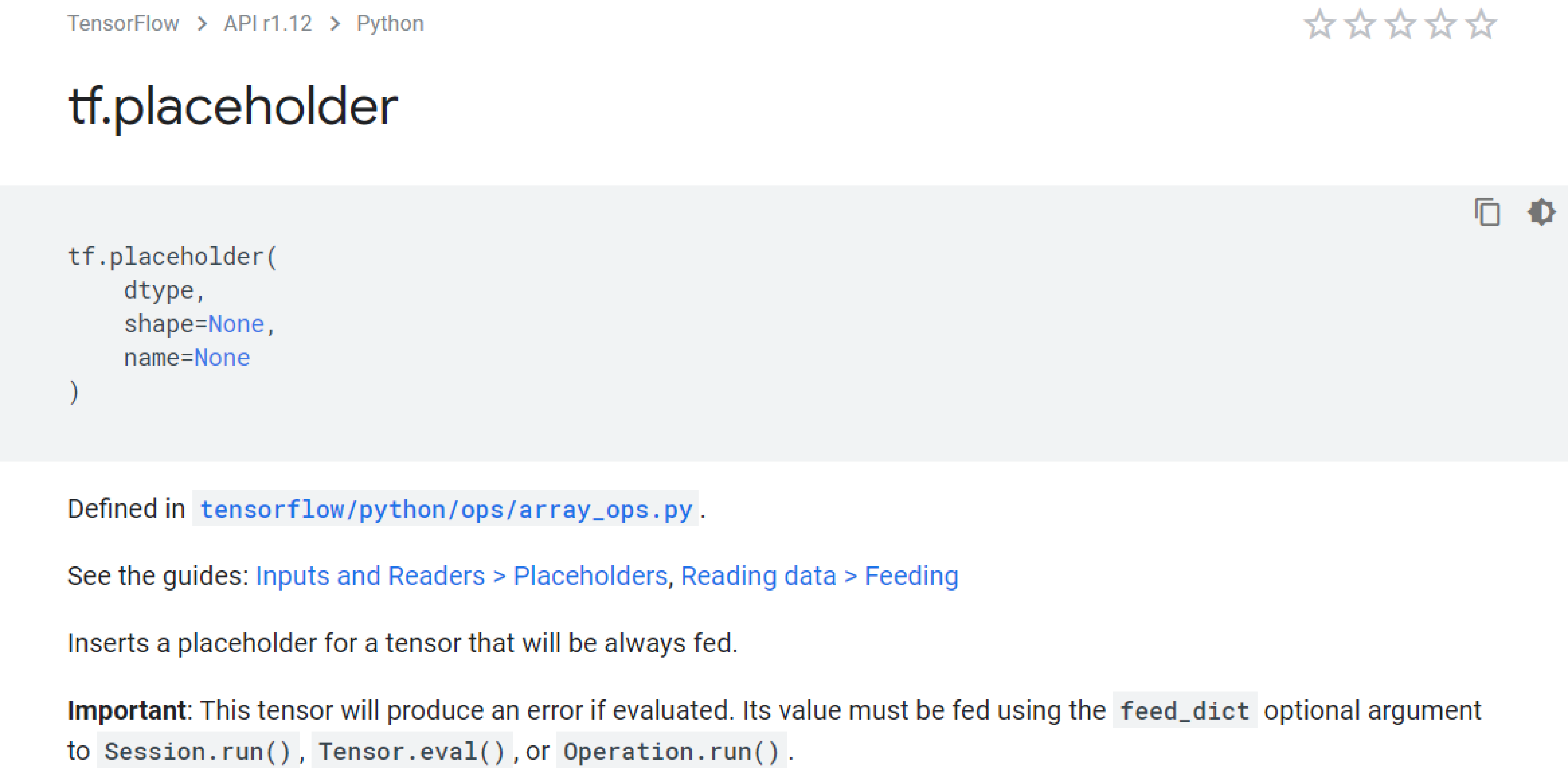
tensorflow 기본 - constant

Result :



```
Tensor("Const:0", shape=(), dtype=string)
Tensor("Add:0", shape=(), dtype=int32)
Tensor("Const_3:0", shape=(2,), dtype=int32)
========================================
b'Deep Learning course'
[10, 9, 19]
[1 2]
```

tensorflow 기본 - Placeholder



TensorFlow > API r1.12 > Python

☆☆☆☆☆

tf.placeholder

```
tf.placeholder(
    dtype,
    shape=None,
    name=None
)
```

Defined in `tensorflow/python/ops/array_ops.py`.

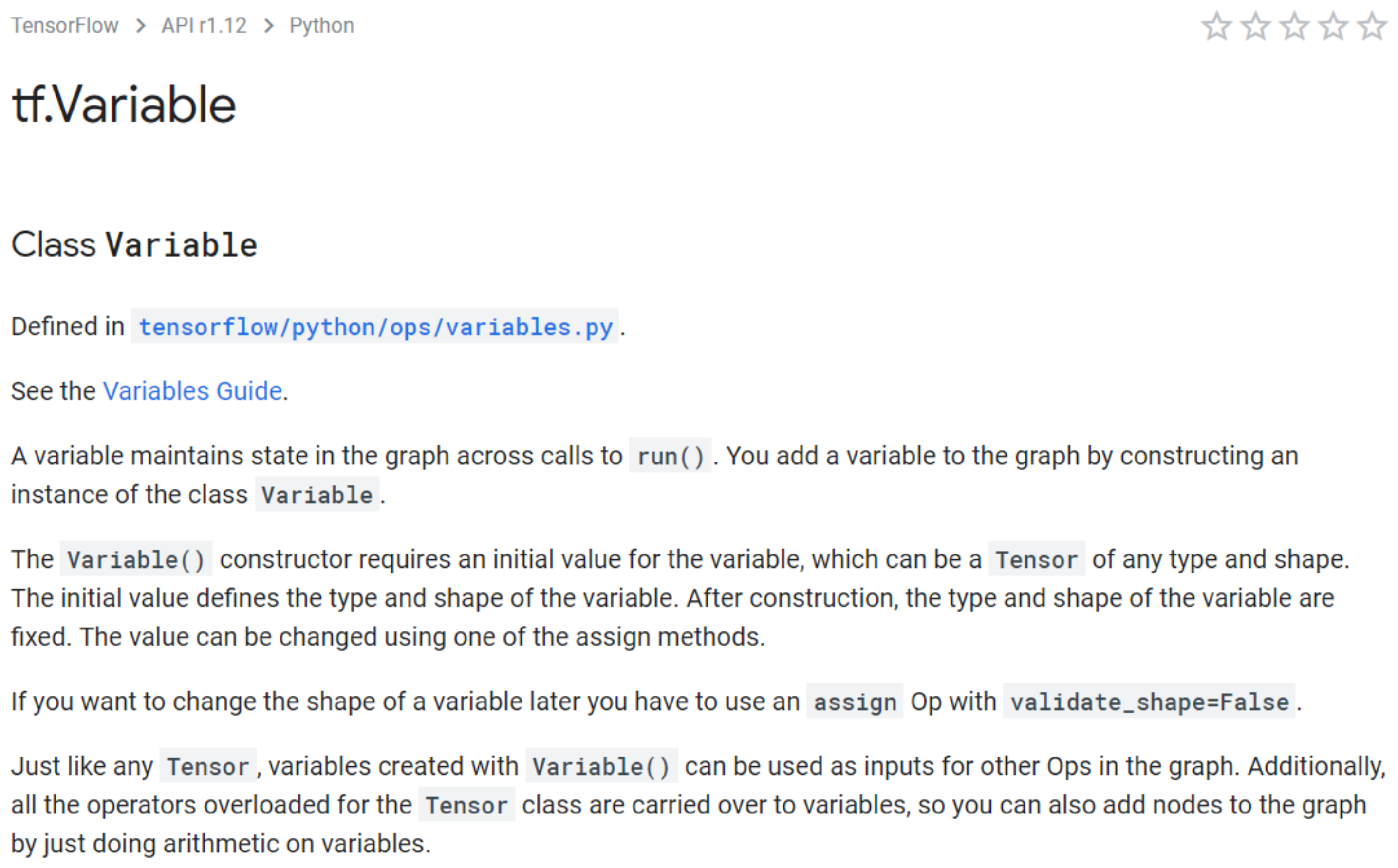
See the guides: Inputs and Readers > Placeholders, Reading data > Feeding

Inserts a placeholder for a tensor that will be always fed.

Important: This tensor will produce an error if evaluated. Its value must be fed using the `feed_dict` optional argument to `Session.run()`, `Tensor.eval()`, or `Operation.run()`.

https://www.tensorflow.org/versions/r1.12/api_docs/python/tf/placeholder

tensorflow 기본 - Variable



TensorFlow > API r1.12 > Python

☆☆☆☆☆

tf.Variable

Class `Variable`

Defined in `tensorflow/python/ops/variables.py`.

See the Variables Guide.

A variable maintains state in the graph across calls to `run()`. You add a variable to the graph by constructing an instance of the class `Variable`.

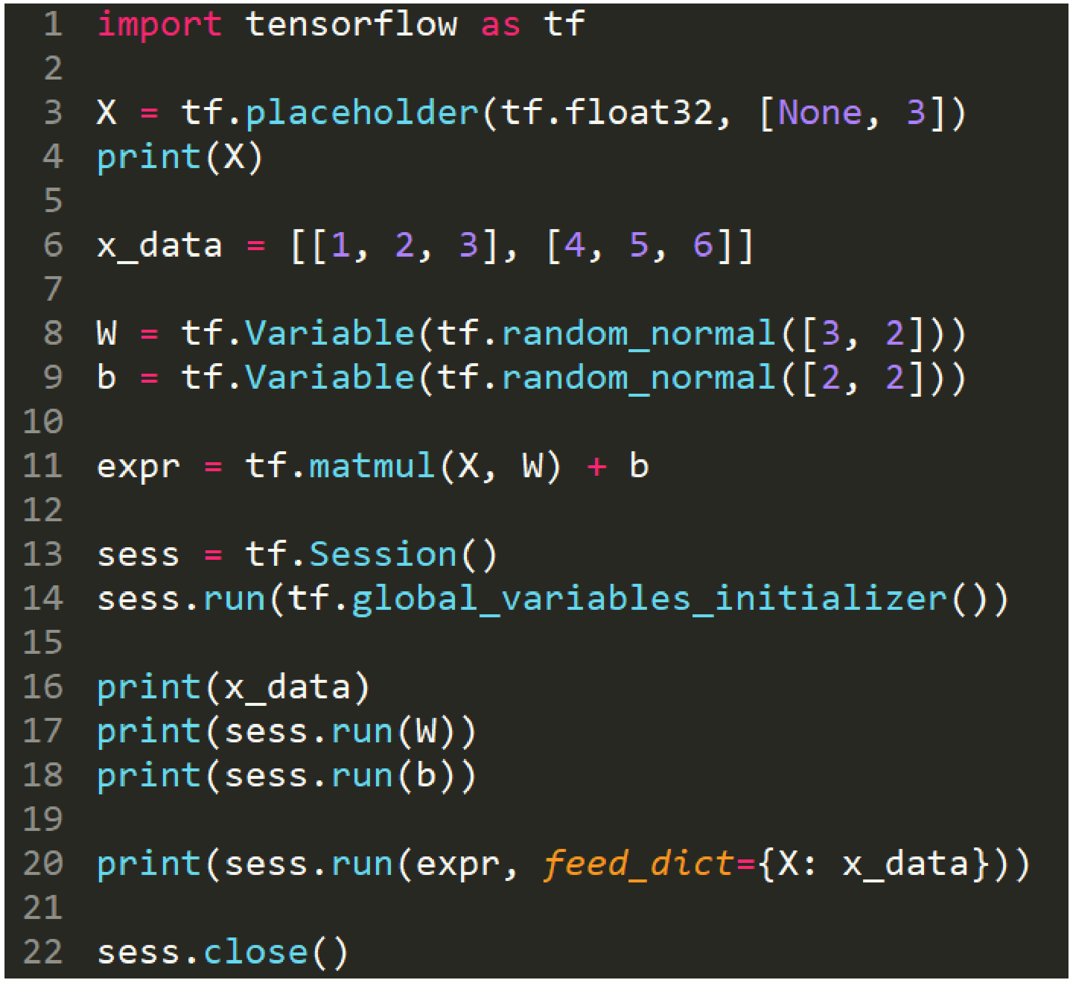
The `Variable()` constructor requires an initial value for the variable, which can be a `Tensor` of any type and shape. The initial value defines the type and shape of the variable. After construction, the type and shape of the variable are fixed. The value can be changed using one of the assign methods.

If you want to change the shape of a variable later you have to use an `assign` Op with `validate_shape=False`.

Just like any `Tensor`, variables created with `Variable()` can be used as inputs for other Ops in the graph. Additionally, all the operators overloaded for the `Tensor` class are carried over to variables, so you can also add nodes to the graph by just doing arithmetic on variables.

https://www.tensorflow.org/api_docs/python/tf/Variable

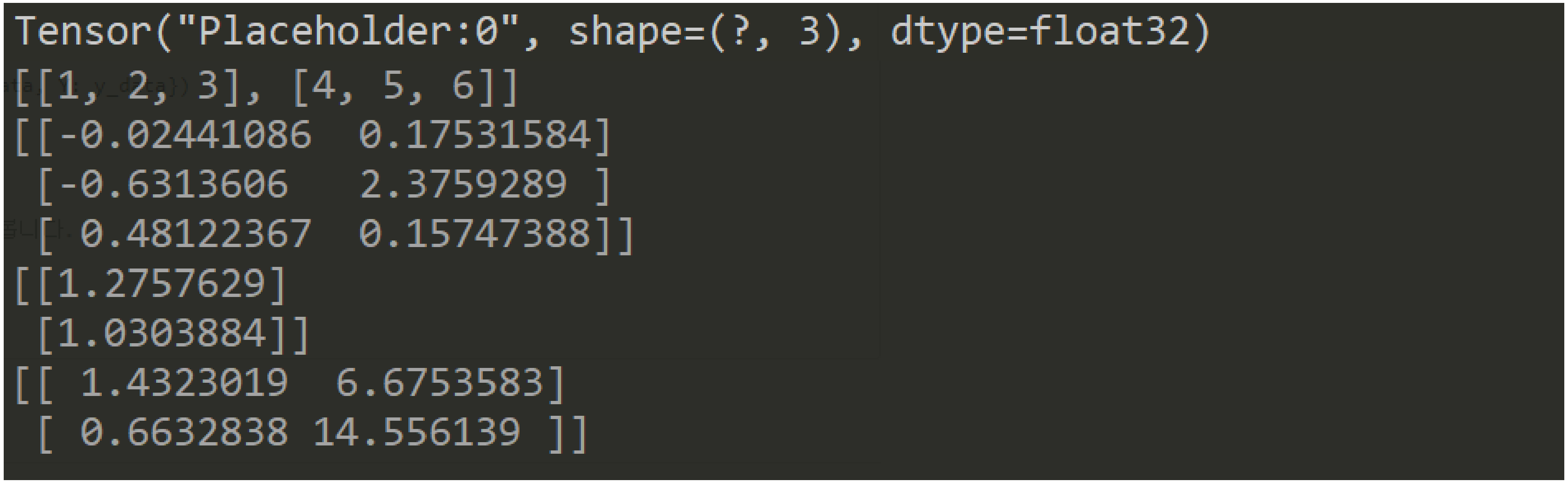
tensorflow 기본 – Placeholder, Variable



```python
import tensorflow as tf

X = tf.placeholder(tf.float32, [None, 3])
print(X)

x_data = [[1, 2, 3], [4, 5, 6]]

W = tf.Variable(tf.random_normal([3, 2]))
b = tf.Variable(tf.random_normal([2, 2]))

expr = tf.matmul(X, W) + b

sess = tf.Session()
sess.run(tf.global_variables_initializer())

print(x_data)
print(sess.run(W))
print(sess.run(b))

print(sess.run(expr, feed_dict={X: x_data}))

sess.close()
```

tensorflow 기본 – Placeholder, Variable

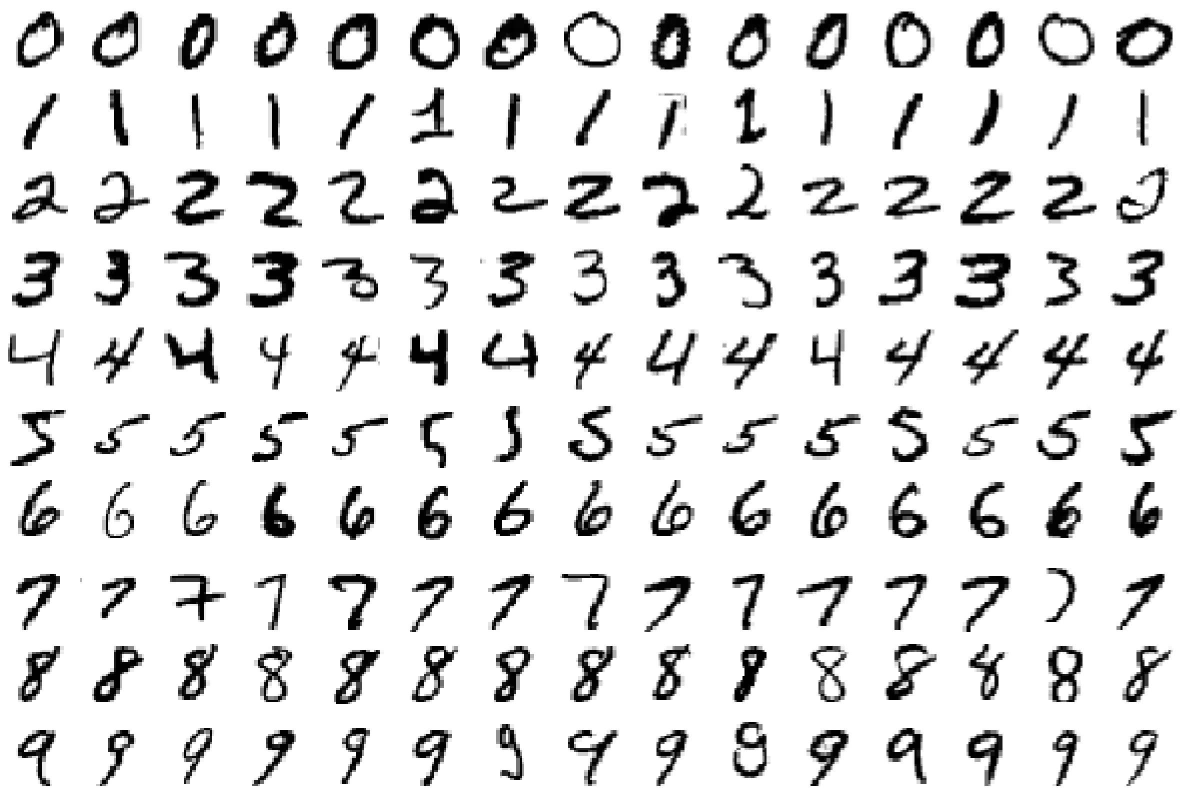
Result :



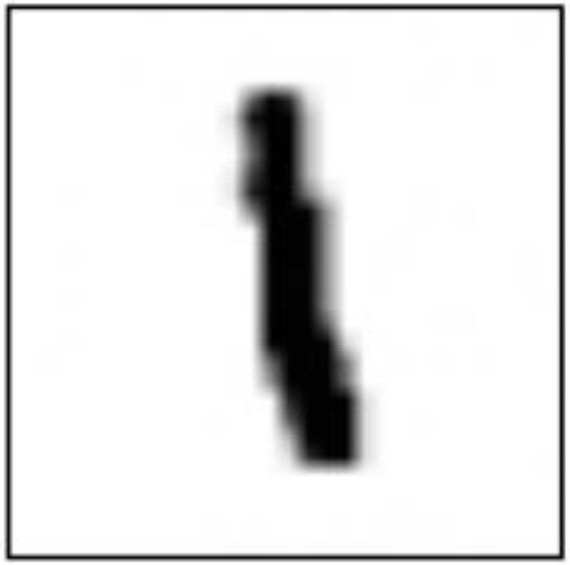
```
Tensor("Placeholder:0", shape=(?, 3), dtype=float32)
[[1, 2, 3], [4, 5, 6]]
[[-0.02441086  0.17531584]
 [-0.6313606   2.3759289 ]
 [ 0.48122367  0.15747388]]
[[1.2757629]
 [1.0303884]]
[[ 1.4323019  6.6753583]
 [ 0.6632838 14.556139 ]]
```

dataset

- MNIST



MNIST



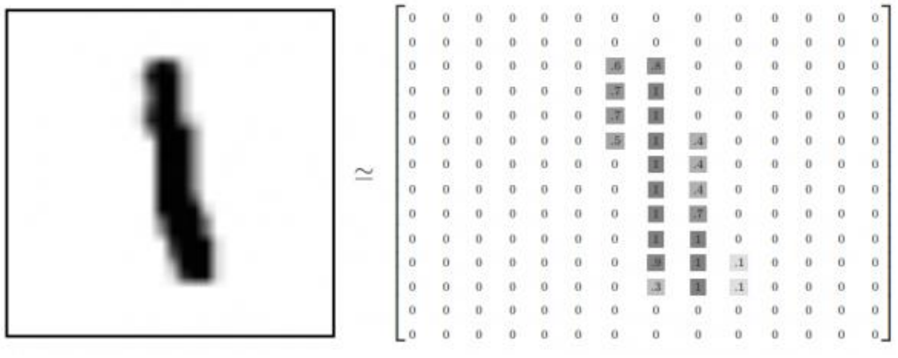
$$\cong \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&.9&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&.3&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

이미지 : **784차원**의 벡터 ex) [0, 0, 0, 0, ……. .7, 1, 0, 0, 0, ………]
라벨 : **0 ~ 9**

MNIST 사용법

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
#mnist dataset 설치, one_hot 방식으로


x, y = mnist.train.next_batch(n)
#x : n만큼의 random 한 mnist 이미지 데이터 (n, 784 차원 벡터)
#y : n만큼의 random 한 mnist 이미지 데이터의 label (n, 10 차원 벡터)

mnist_image = np.array(x).reshape((28, 28))
#x (n, 284) 차원의 벡터 -> (n, 28, 28) 차원의 벡터로 reshape
```

MNIST

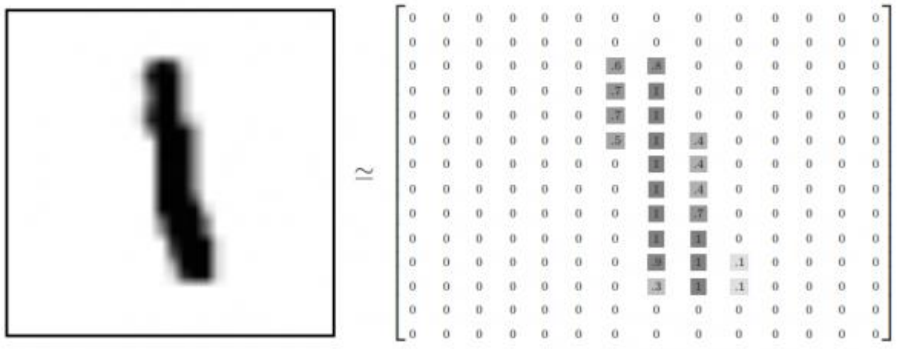


x, y = mnist.train.next_batch(n)

#x : n만큼의 random 한 mnist 이미지 데이터
#y : n만큼의 random 한 mnist 이미지 데이터의 label

x : **[n, 784] 차원** 벡터 ex) [[0, 0, 0, 0, ……. .7, 1, 0, 0, 0, ………], […], […], …]
y : **[n, 10] 차원** 벡터 ex) [[0, 1, 0, 0, 0, 0, 0, 0, 0, 0], […], [..], …]

MNIST



x, y = mnist.train.next_batch(n)

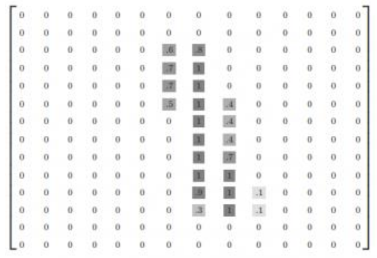
#x : n만큼의 random 한 mnist 이미지 데이터
#y : n만큼의 random 한 mnist 이미지 데이터의 label

x : **[n, 784] 차원** 벡터 ex) [[0, 0, 0, 0, ……. .7, 1, 0, 0, 0, ………], […], […], …]
y : **[n, 10] 차원** 벡터 ex) [[0, 1, 0, 0, 0, 0, 0, 0, 0, 0], […], [..], …]

mnist_image = np.array(x).reshape((28, 28))

#x (n, 284) 차원의 벡터 -> (n, 28, 28) 차원의 np array 로 reshape

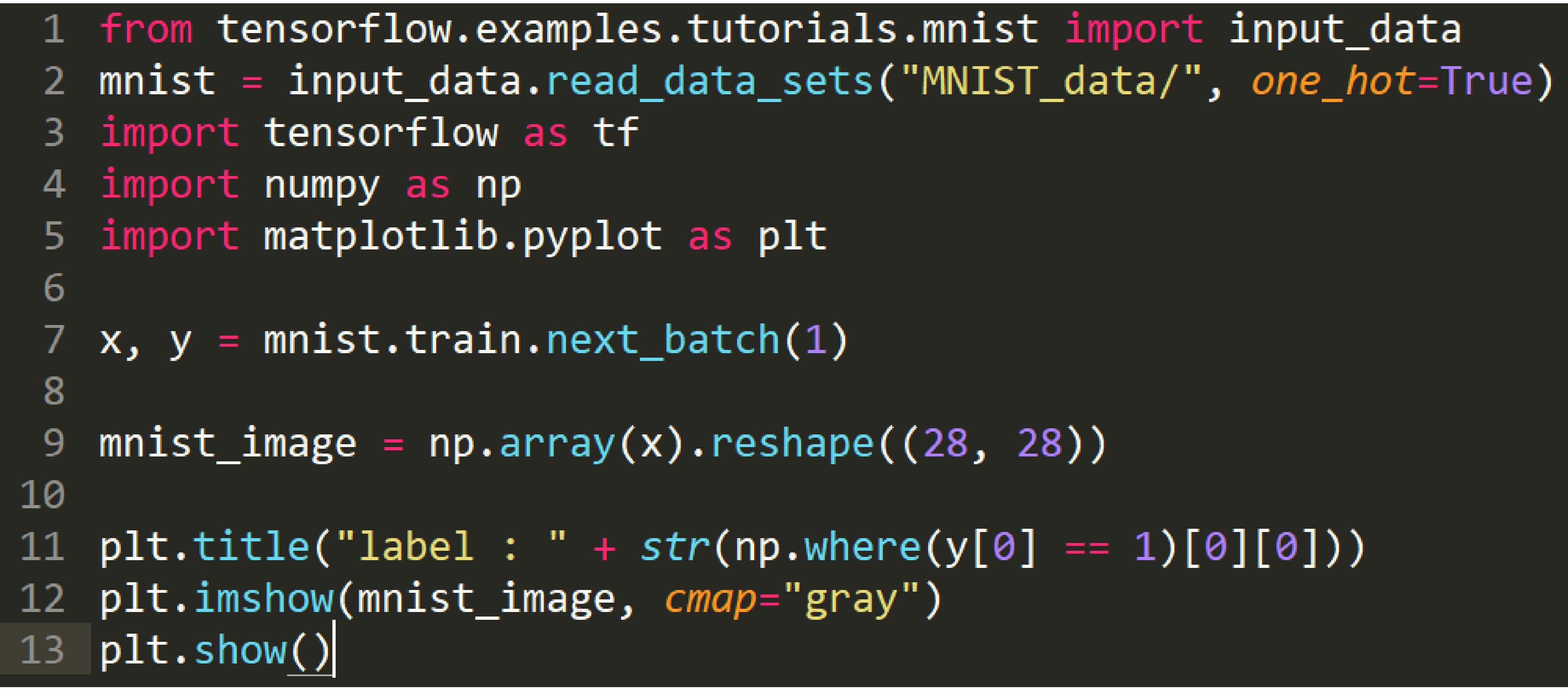
mnist_image : **[n, 28, 28] 차원** 벡터 ex) [[…], […], […], […], …]



이미지 라이브러리 설치

- pip install matplotlib #이미지를 확인 할 수 있도록

MNIST 출력 소스



```python
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

x, y = mnist.train.next_batch(1)

mnist_image = np.array(x).reshape((28, 28))

plt.title("label : " + str(np.where(y[0] == 1)[0][0]))
plt.imshow(mnist_image, cmap="gray")
plt.show()
```

과제

10개의 랜덤 한 mnist 이미지와 라벨을 for 문을 이용하여 순차적으로 plt.show() 함수를 사용하여 출력하기

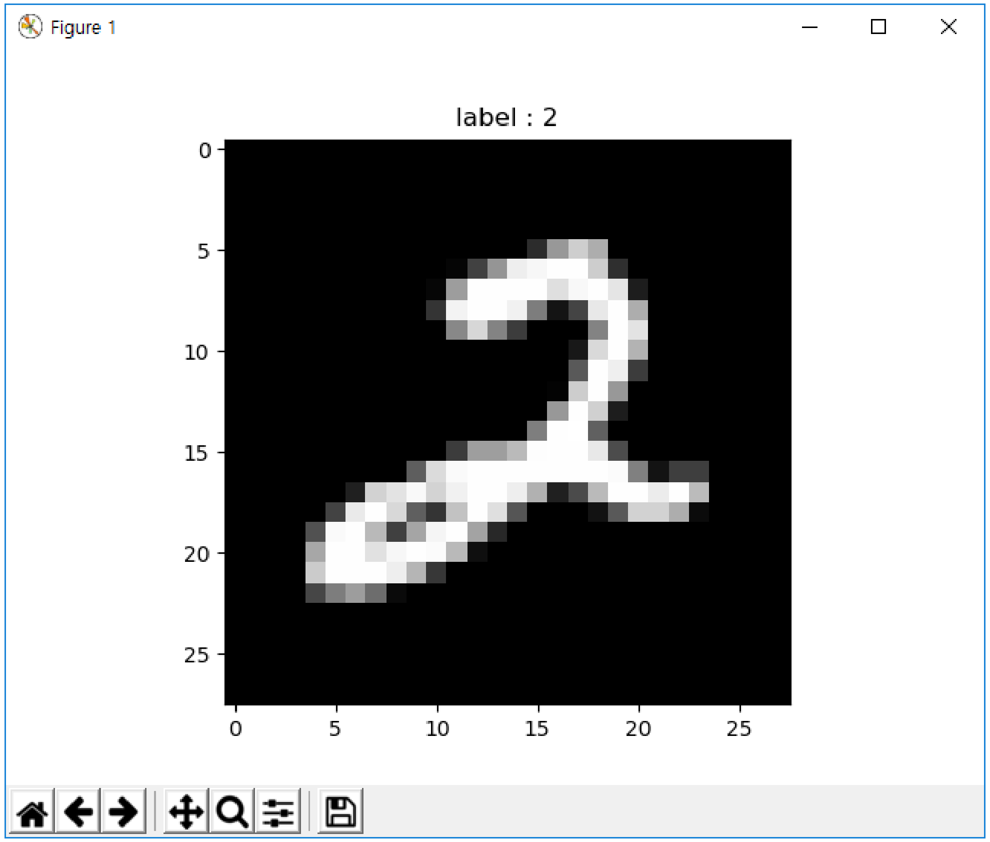


Figure 1
label : 2
0
5
10
15
20
25
0
5
10
15
20
25