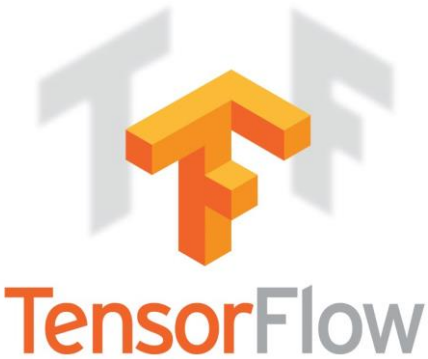
TensorFlow Tutorial

전현호

Contents

1. What is TensorFlow?
2. Install TensorFlow
3. Data type
4. Linear regression + Plot
5. MNIST
6. Multi Layer Perceptron (NN, FCN)
7. Convolutional NN

What is TensorFlow?



다차원 배열

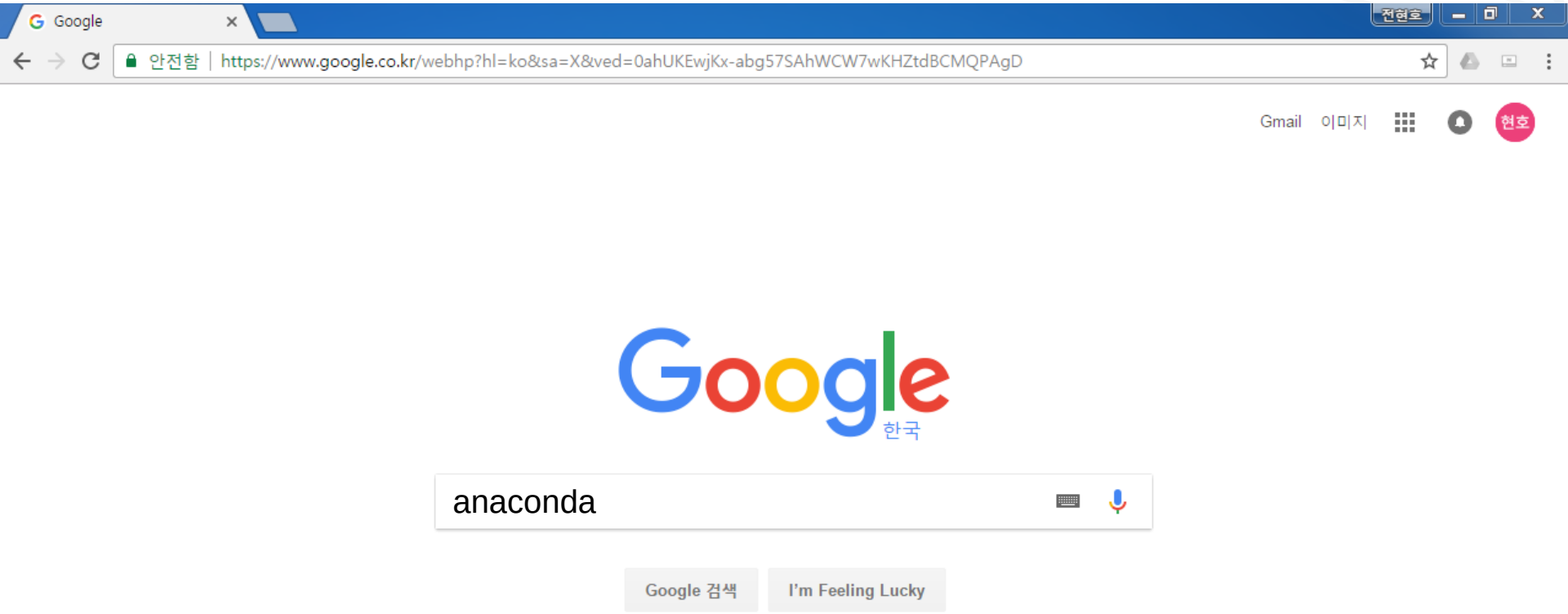
- 텐서플로우(TensorFlow) : 구글에서 개발한 기계 학습 및 딥러닝용 오픈소스 라이브러리
- 왜 텐서플로우를 사용해야 하는가?
  - 구현이 쉬움 (연산, 신경망 구조)
  - 여러 CPU 및 GPU에서 동작이 가능함
  - 데이터 및 구조를 시각적으로 확인할 수 있음 (TensorBoard)
  - 기타 등등 (뒤에서 설명)

TensorFlow Dev Summit - 17.02.16

- 텐서플로우 1.0 공식 버전 릴리즈
- 빨라짐 : 8GPUs + inception v3 모델을 사용할 경우 7.3배 속도 향상
  64GPUs + Inception v3 모델을 사용할 경우 58배 속도 향상
- 유연해짐 : 더 고급 수준의 API 추가 및 딥러닝 라이브러리중 하나인 Keras와 완벽한 호환성을 제공하는 새로운 tf.keras 모듈이 추가됨
- 기타 등등
- Python tf_upgrade.py --infile *InputFile* --outfile *OutputFile*
- pip install tensorflow로 대부분 환경에서 바로 업데이트 가능

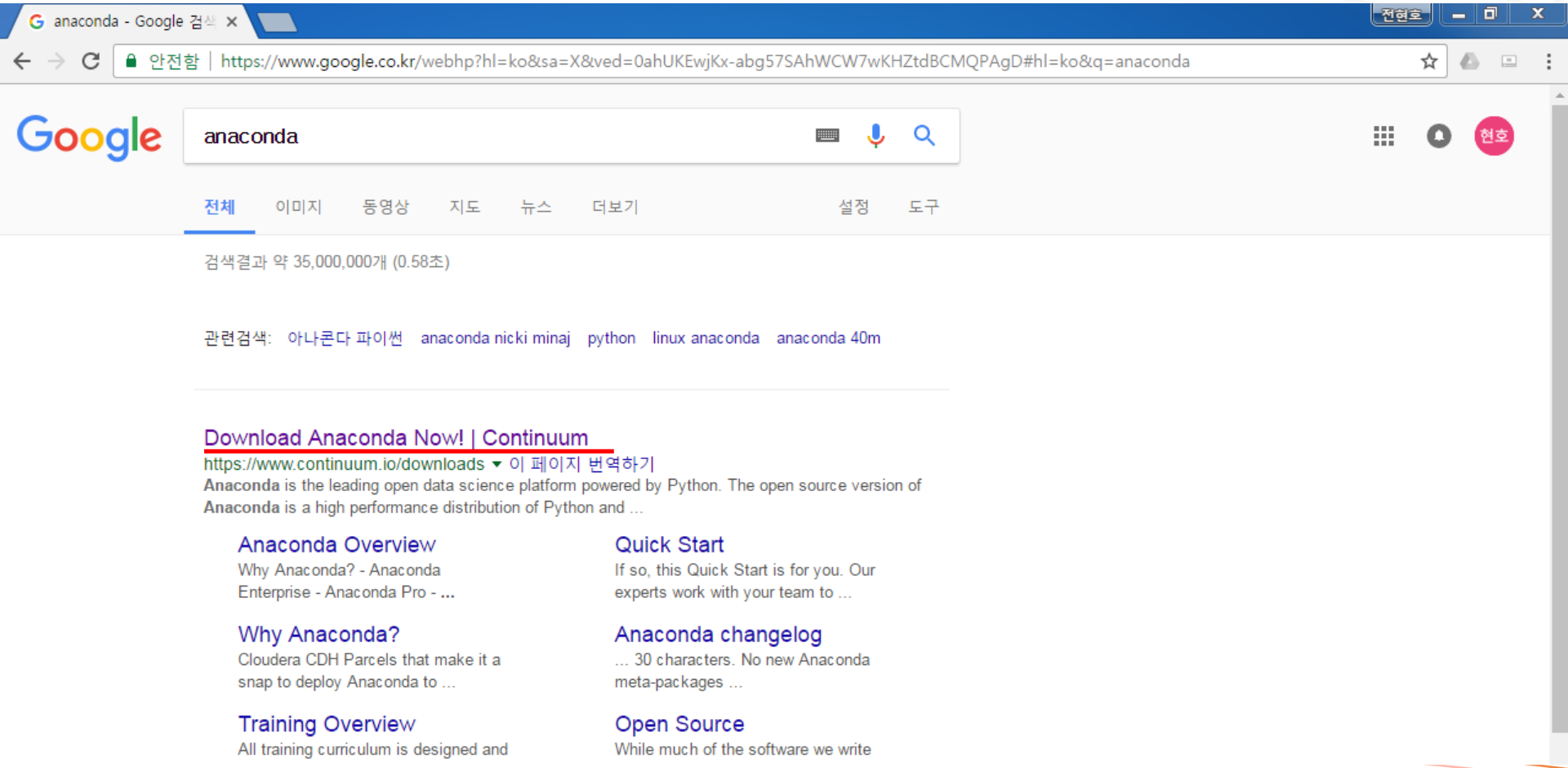
Install TensorFlow

1. Anaconda3 4.2.0 버전 설치 (Python 3.5)



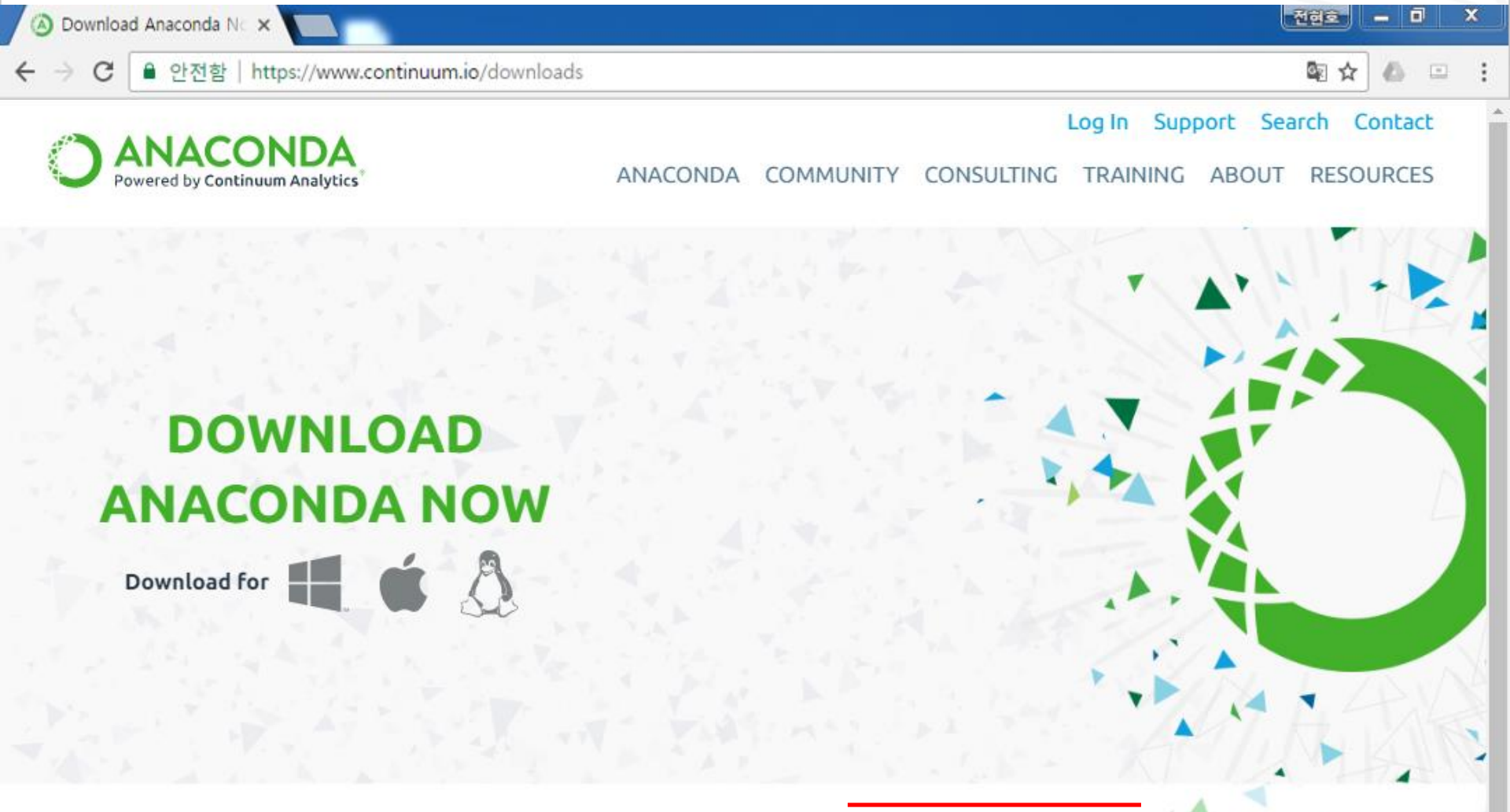
Install TensorFlow

1. Anaconda3 4.2.0 버전 설치 (Python 3.5)



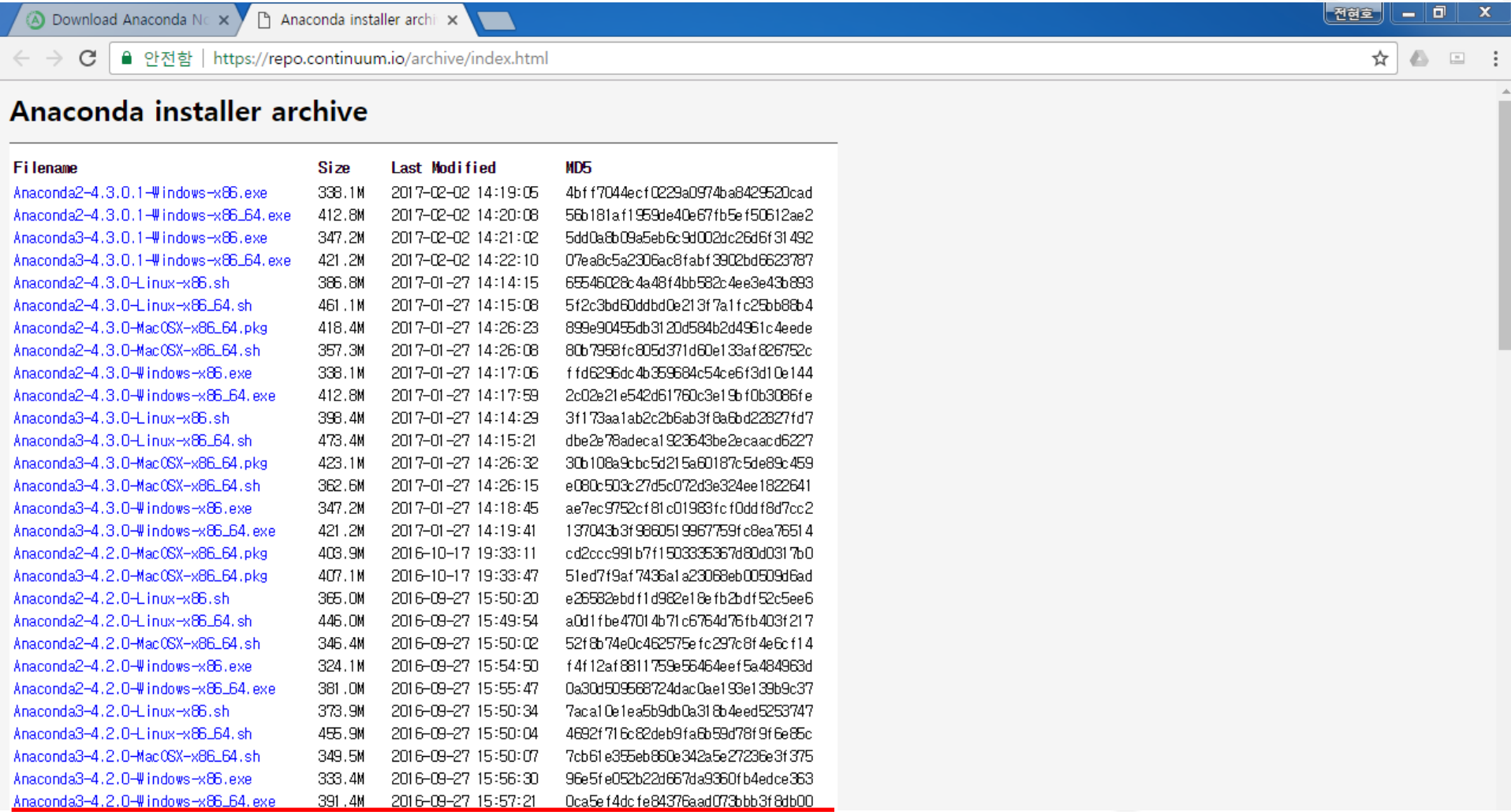
Install TensorFlow

1. Anaconda3 4.2.0 버전 설치 (Python 3.5)



Install TensorFlow

1. Anaconda3 4.2.0 버전 설치 (Python 3.5)



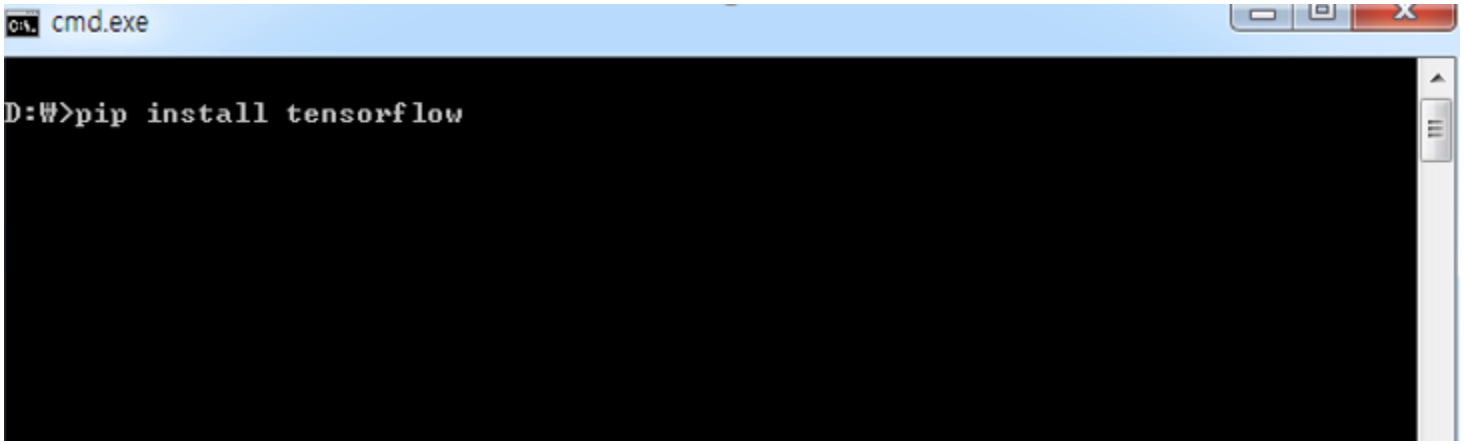
Anaconda installer archive

Filename	Size	Last Modified	MD5
Anaconda2-4.3.0.1-Windows-x86.exe	338.1M	2017-02-02 14:19:05	4bff7044ecf0229a0974ba8429520cad
Anaconda2-4.3.0.1-Windows-x86_64.exe	412.8M	2017-02-02 14:20:08	56b181af1959de40e67fb5ef50612ae2
Anaconda3-4.3.0.1-Windows-x86.exe	347.2M	2017-02-02 14:21:02	5dd0a8b09a5eb6c9d002dc26d6f31492
Anaconda3-4.3.0.1-Windows-x86_64.exe	421.2M	2017-02-02 14:22:10	07ea8c5a2306ac8fabf3902bd6623787
Anaconda2-4.3.0-Linux-x86.sh	386.8M	2017-01-27 14:14:15	65546028c4a48f4bb582c4ee3e43b893
Anaconda2-4.3.0-Linux-x86_64.sh	461.1M	2017-01-27 14:15:08	5f2c3bd60ddbd0e213f7a1fc25bb88b4
Anaconda2-4.3.0-MacOSX-x86_64.pkg	418.4M	2017-01-27 14:26:23	899e90455db3120d584b2d4961c4eede
Anaconda2-4.3.0-MacOSX-x86_64.sh	357.3M	2017-01-27 14:26:08	80b7958fc805d371d60e133af826752c
Anaconda2-4.3.0-Windows-x86.exe	338.1M	2017-01-27 14:17:06	ffd6296dc4b359684c54ce6f3d10e144
Anaconda2-4.3.0-Windows-x86_64.exe	412.8M	2017-01-27 14:17:59	2c02e21e542d61760c3e19bf0b3086fe
Anaconda3-4.3.0-Linux-x86.sh	398.4M	2017-01-27 14:14:29	3f173aa1ab2c2b6ab3f8a6bd22827fd7
Anaconda3-4.3.0-Linux-x86_64.sh	473.4M	2017-01-27 14:15:21	dbe2e78adeca1923643be2ecaacd6227
Anaconda3-4.3.0-MacOSX-x86_64.pkg	423.1M	2017-01-27 14:26:32	30b108a9cbc5d215a60187c5de89c459
Anaconda3-4.3.0-MacOSX-x86_64.sh	362.6M	2017-01-27 14:26:15	e080c503c27d5c072d3e324ee1822641
Anaconda3-4.3.0-Windows-x86.exe	347.2M	2017-01-27 14:18:45	ae7ec9752cf81c01983fcf0ddf8d7cc2
Anaconda3-4.3.0-Windows-x86_64.exe	421.2M	2017-01-27 14:19:41	137043b3f986051996759fc8ea76514
Anaconda2-4.2.0-MacOSX-x86_64.pkg	403.9M	2016-10-17 19:33:11	cd2ccc991b7f1503335367d80d03f7b0
Anaconda3-4.2.0-MacOSX-x86_64.pkg	407.1M	2016-10-17 19:33:47	51ed7f9af7436a1a23068eb00509d6ad
Anaconda2-4.2.0-Linux-x86.sh	365.0M	2016-09-27 15:50:20	e26582ebdf1d982e18efb2bdf52c5ee6
Anaconda2-4.2.0-Linux-x86_64.sh	446.0M	2016-09-27 15:49:54	a0d1fbe47014b71c6764d76fb403f217
Anaconda2-4.2.0-MacOSX-x86_64.sh	346.4M	2016-09-27 15:50:02	52f8b74e0c462575efc297c8f4e6cf14
Anaconda2-4.2.0-Windows-x86.exe	324.1M	2016-09-27 15:54:50	f4f12af8811759e56464eef5a484963d
Anaconda2-4.2.0-Windows-x86_64.exe	381.0M	2016-09-27 15:55:47	0a30d509568724dac0ae193e139b9c37
Anaconda3-4.2.0-Linux-x86.sh	373.9M	2016-09-27 15:50:34	7aca10e1ea5b9db0a318b4eed5253747
Anaconda3-4.2.0-Linux-x86_64.sh	455.9M	2016-09-27 15:50:04	4692f716c82deb9fa6b59d78f9f6e85c
Anaconda3-4.2.0-MacOSX-x86_64.sh	349.5M	2016-09-27 15:50:07	7cb61e355eb860e342a5e27236e3f375
Anaconda3-4.2.0-Windows-x86.exe	333.4M	2016-09-27 15:56:30	96e5fe052b22d667da9360fb4edce363
Anaconda3-4.2.0-Windows-x86_64.exe	391.4M	2016-09-27 15:57:21	0ca5ef4dcfe84376aad073bbb3f8db00

Install TensorFlow

2. TensorFlow 설치 – CPU 버전

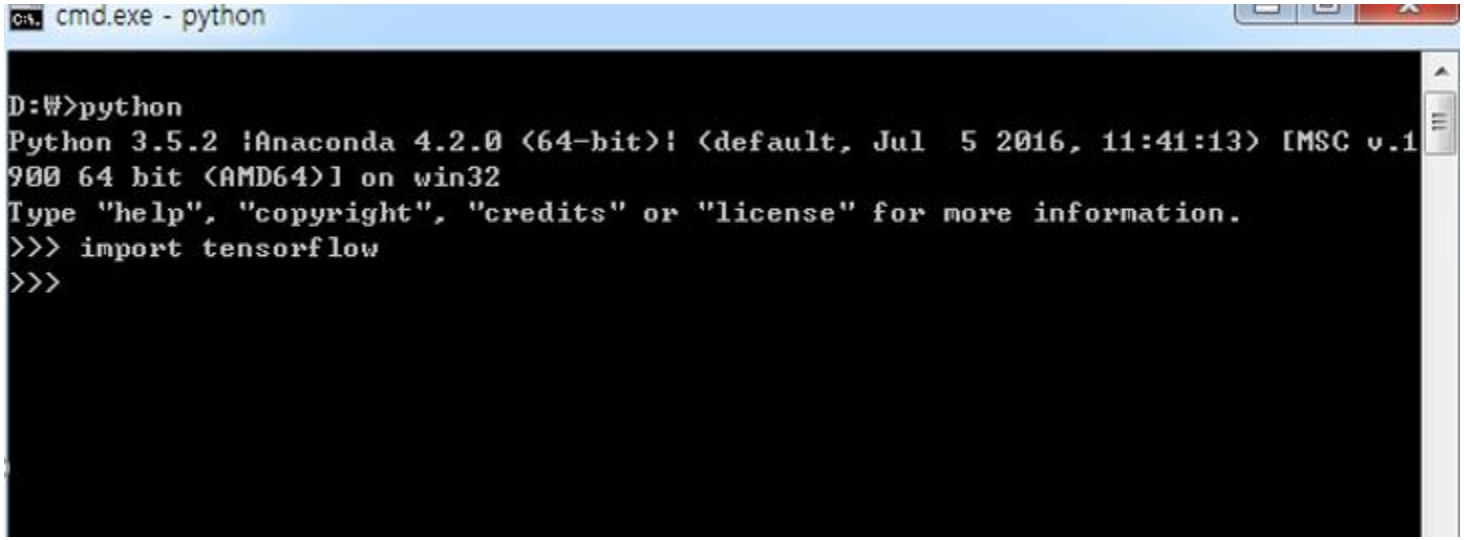
- cmd창에 pip install tensorflow 입력



Install TensorFlow

3. 설치 확인

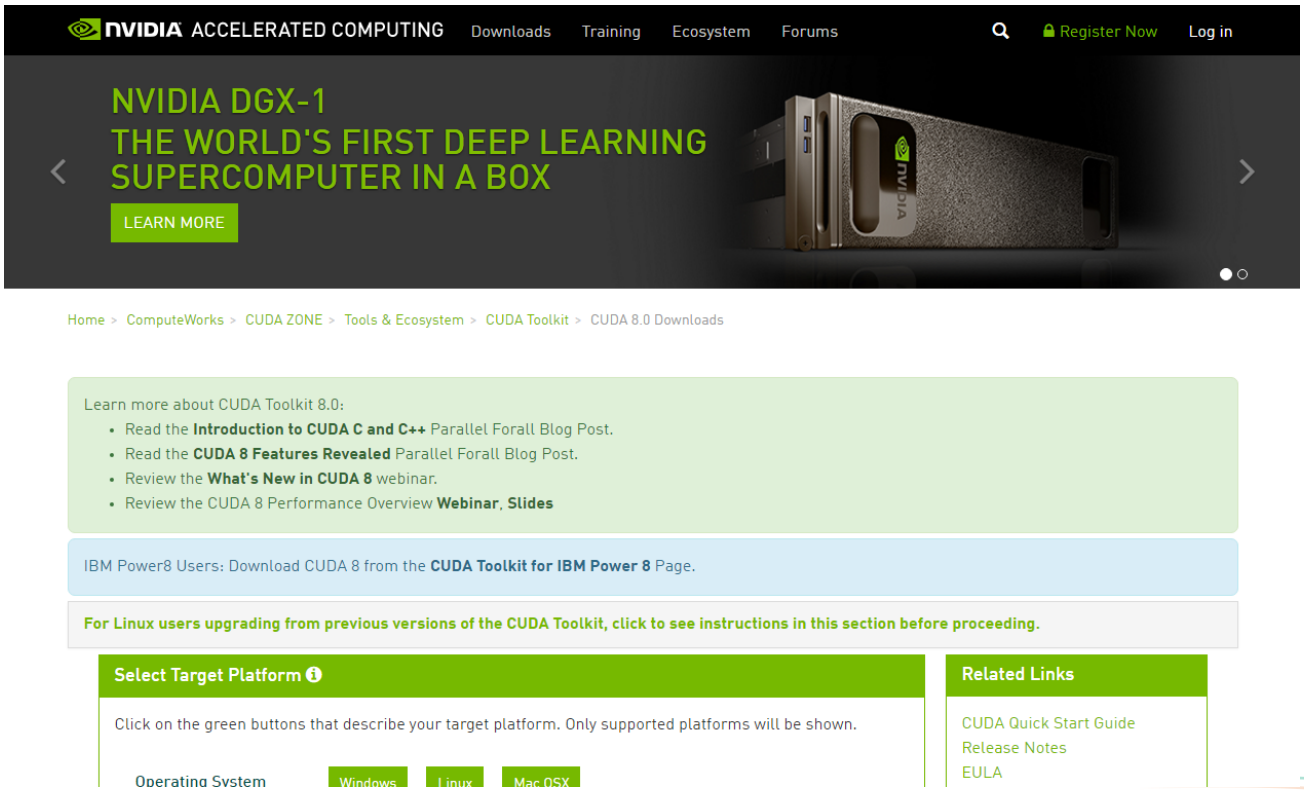
- Python을 실행시킨 후 import tensorflow를 통해 에러가 나는지 확인



Install TensorFlow

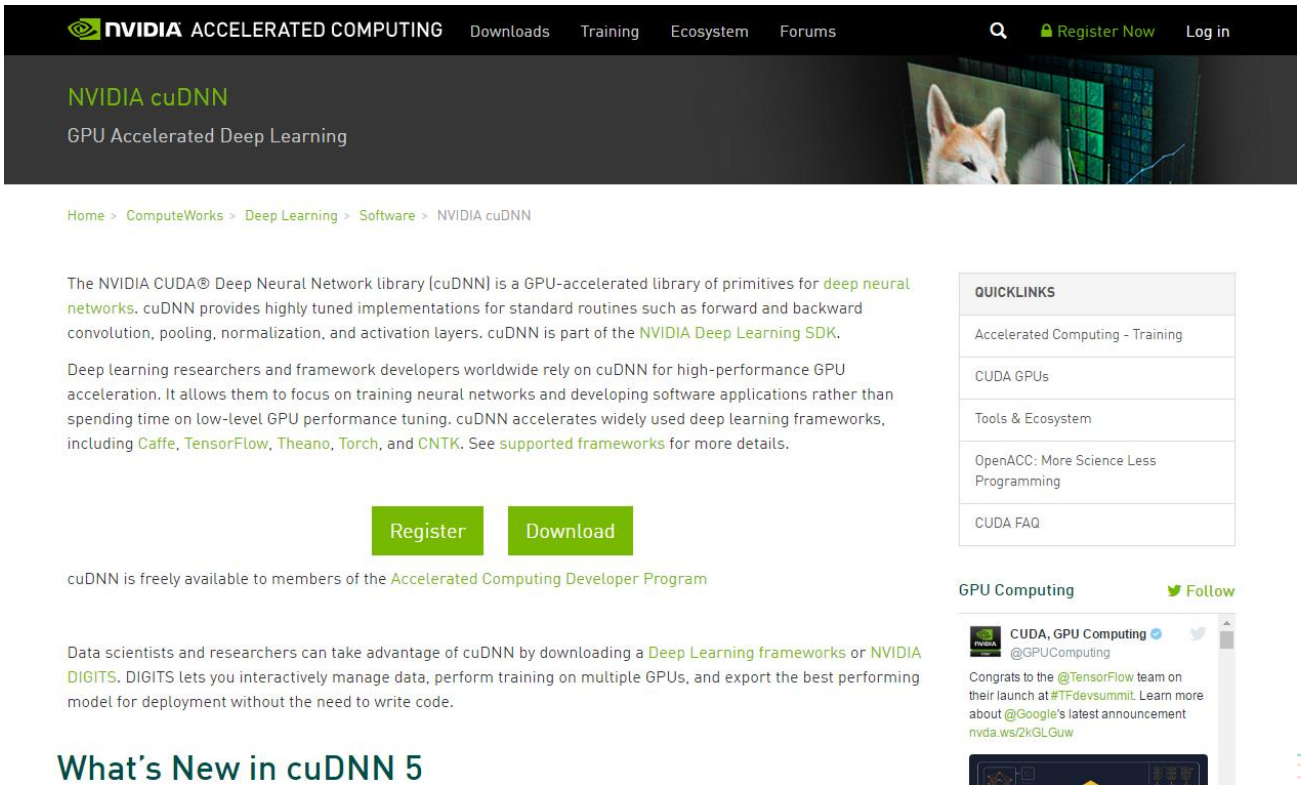
- GPU 버전 설치 과정

1. CUDA 8.0 버전 설치 (https://developer.nvidia.com/cuda-downloads)



Install TensorFlow

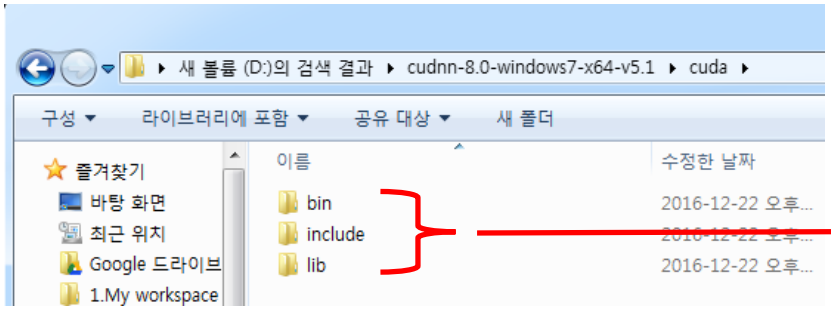
2. cuDNN 5.1 버전 설치 (https://developer.nvidia.com/cudnn)
   간단한 등록 과정을 요구함

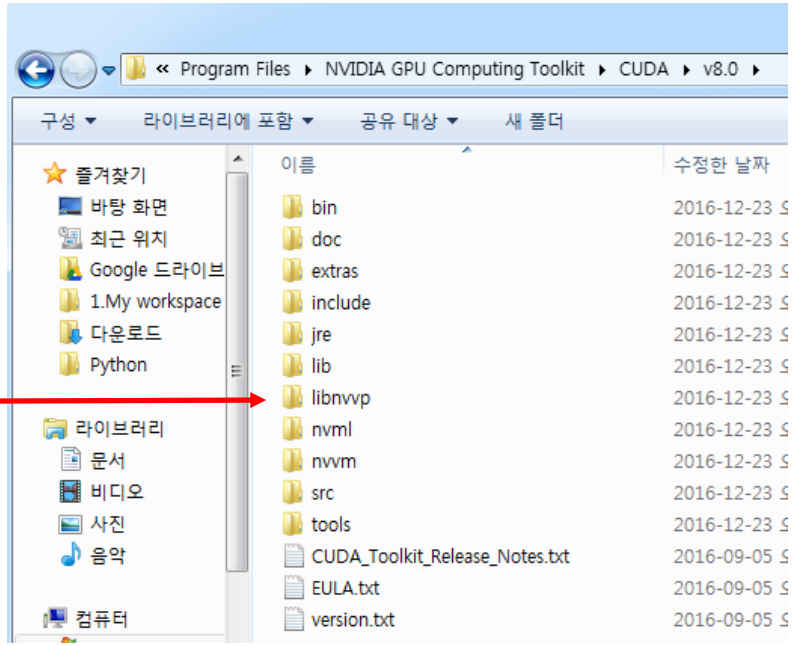


Install TensorFlow

3. cuDNN 파일의 압축풀기

4. 결과로 나오는 bin, include, lib 폴더를 CUDA가 설치된 경로로 접근하여 붙여넣기



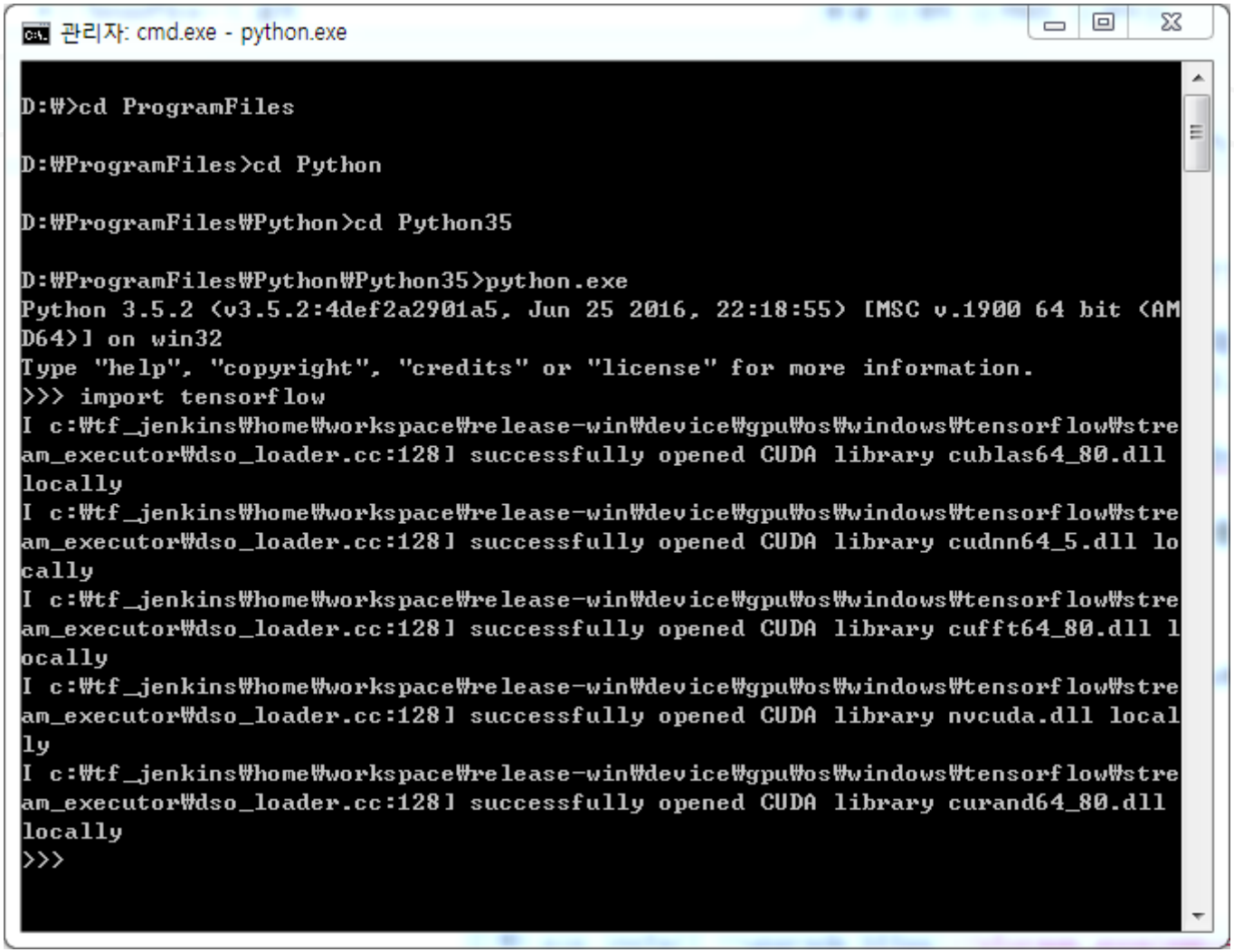


Install TensorFlow

5. Tensorflow 설치 - GPU 버전

   - cmd창에 pip install tensorflow-gpu 입력
   - CPU버전과 마찬가지로 설치가 잘 되었는지 확인

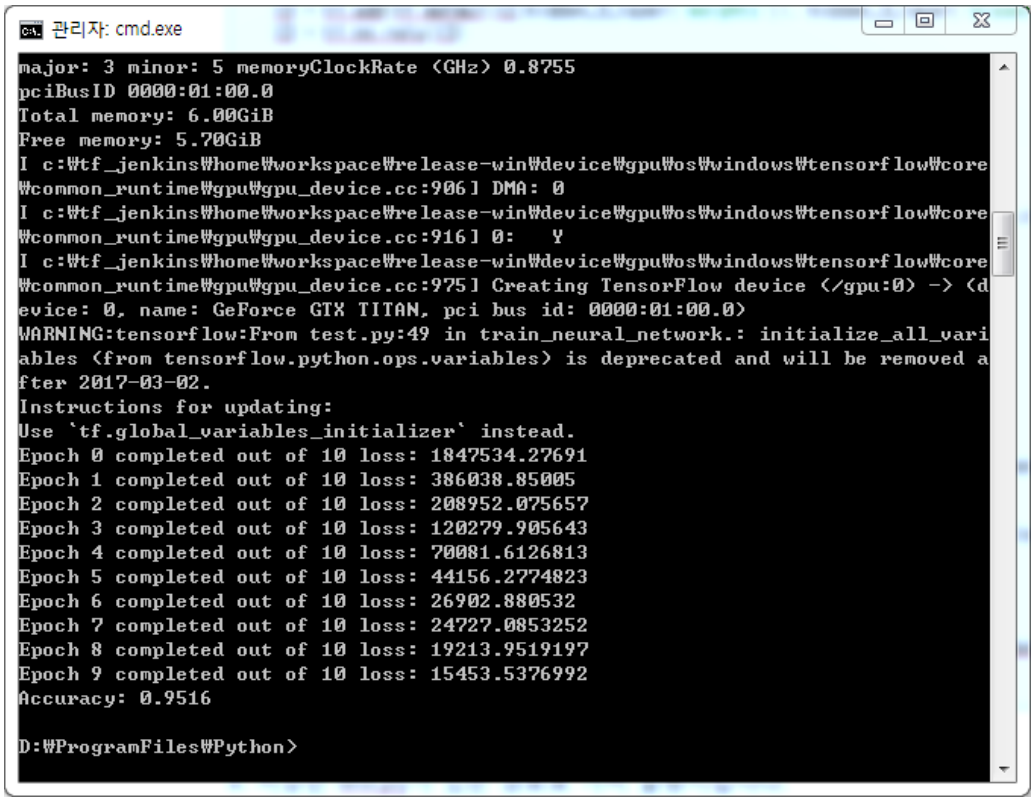
Install TensorFlow

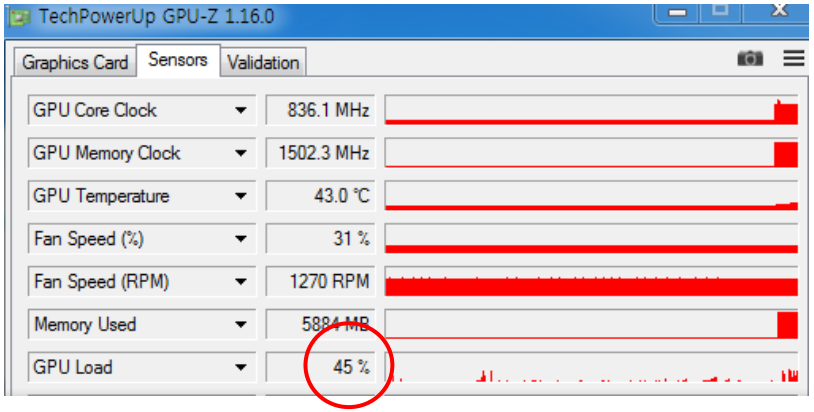


6. quit()을 통해 python 종료

Install TensorFlow

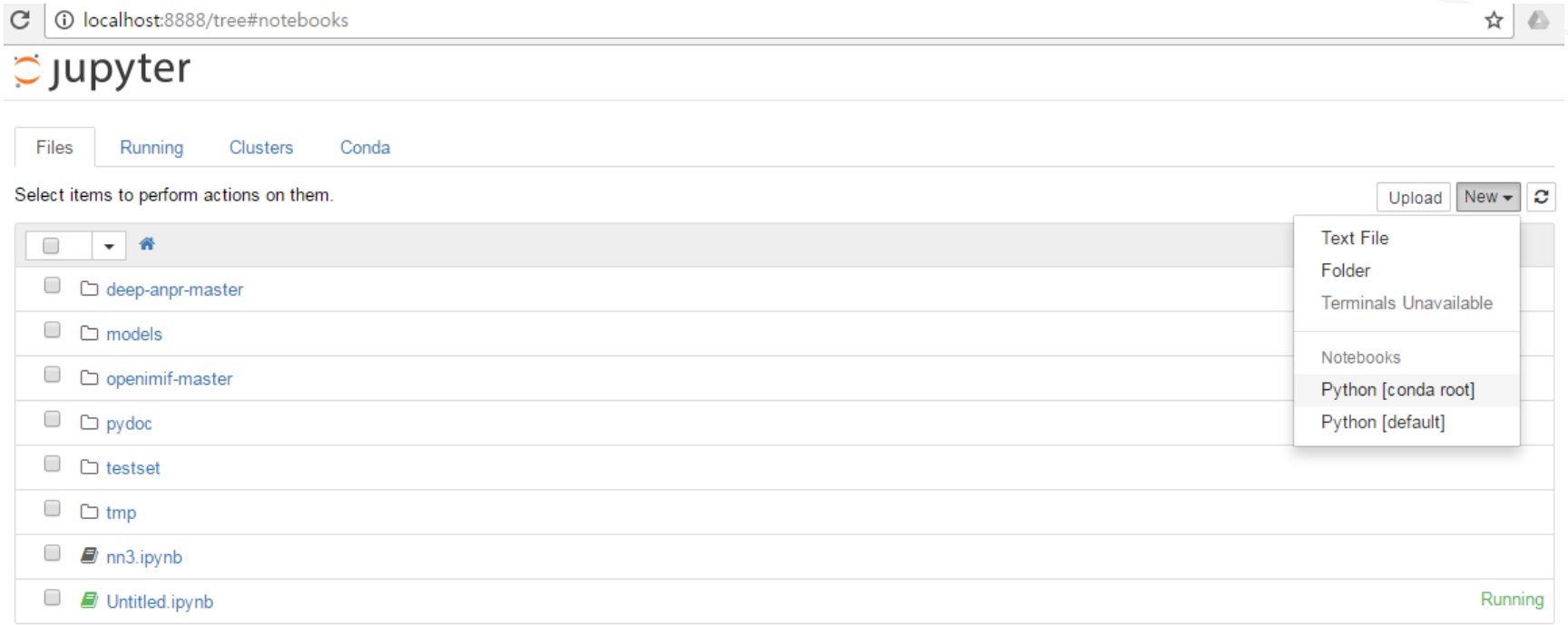
- MNIST 예제를 학습시켜 동작 확인 (3 hidden layer)





Data type

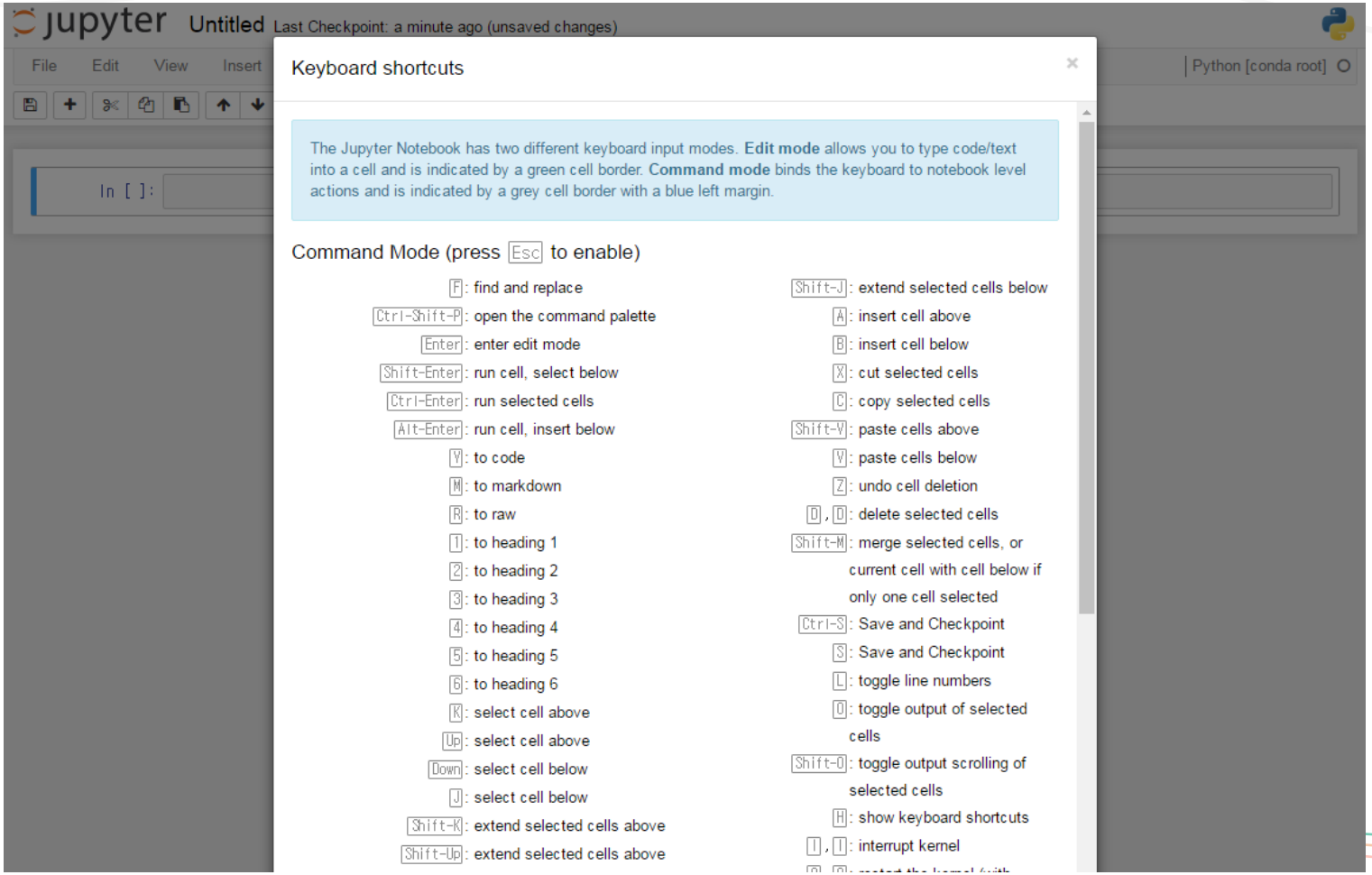
- cmd - jupyter notebook



※주의 : 경로상에 관련 파일이 있지 않을 경우 404 에러 발생

Data type

- jupyter notebook - h



Data type

※ https://www.tensorflow.org/versions/r0.12/api_docs/python/

Inputs and Readers > Placeholders

Members	
`tf.placeholder(dtype, shape=None, name=None)`	Inserts a placeholder for a tensor that will be always fed.

Variables > Variables

Members	
`class tf.Variable`	See the [Variables How To] (../../how_tos/variables/index.md) for a high
`tf.Variable.__init__(initial_value=None, trainable=True, collections=None, validate_shape=True, caching_device=None, name=None, variable_def=None, dtype=None, expected_shape=None, import_scope=None)`	Creates a new variable with value initial_value.

Constants, Sequences, and Random Values > Constant Value Tensors

Members	
`tf.zeros(shape, dtype=tf.float32, name=None)`	Creates a tensor with all elements set to zero.
`tf.zeros_like(tensor, dtype=None, name=None, optimize=True)`	Creates a tensor with all elements set to zero.
`tf.ones(shape, dtype=tf.float32, name=None)`	Creates a tensor with all elements set to 1.
`tf.ones_like(tensor, dtype=None, name=None, optimize=True)`	Creates a tensor with all elements set to 1.
`tf.fill(dims, value, name=None)`	Creates a tensor filled with a scalar value.
`tf.constant(value, dtype=None, shape=None, name='Const', verify_shape=False)`	Creates a constant tensor.

Data type

- Code

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

In [3]: ph = tf.placeholder(dtype = tf.float32, shape=[3,3])
        var = tf.Variable([1, 2, 3, 4, 5], dtype = tf.float32)
        const = tf.constant([10, 20, 30, 40, 50], dtype = tf.float32)

In [5]: print(ph)

        Tensor("Placeholder_1:0", shape=(3, 3), dtype=float32)

In [6]: print(var)

        Tensor("Variable/read:0", shape=(5,), dtype=float32)

In [7]: print(const)

        Tensor("Const:0", shape=(5,), dtype=float32)

In [ ]:
```

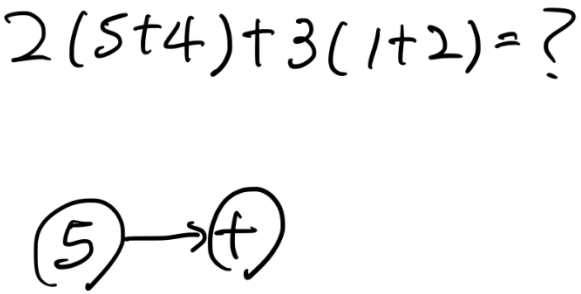
Data type

- 그래프 생성

$2(5+4)+3(1+2)=?$

Data type

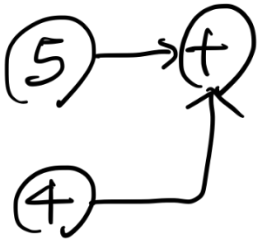
- 그래프 생성



Data type

- 그래프 생성

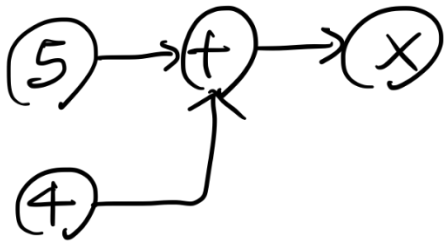
2(5+4)+3(1+2)=?



Data type

- 그래프 생성

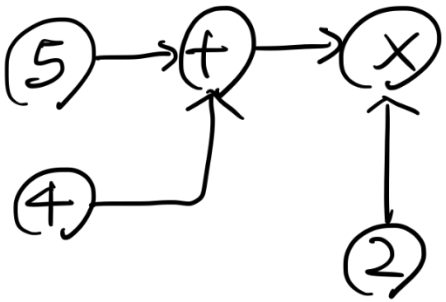
2(5+4)+3(1+2)=?



Data type

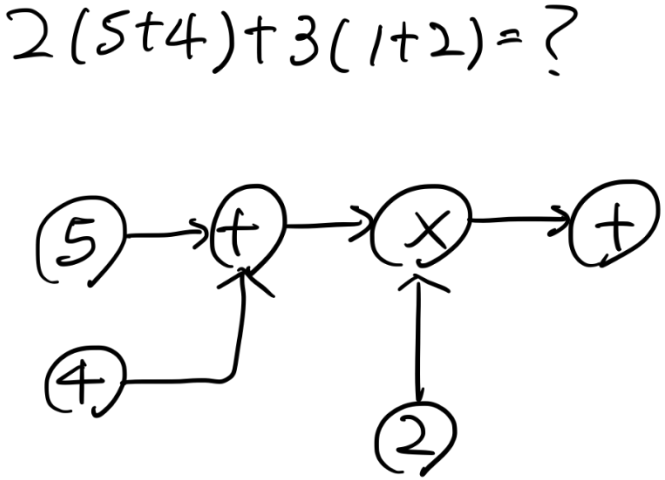
- 그래프 생성

$2(5+4)+3(1+2)=?$



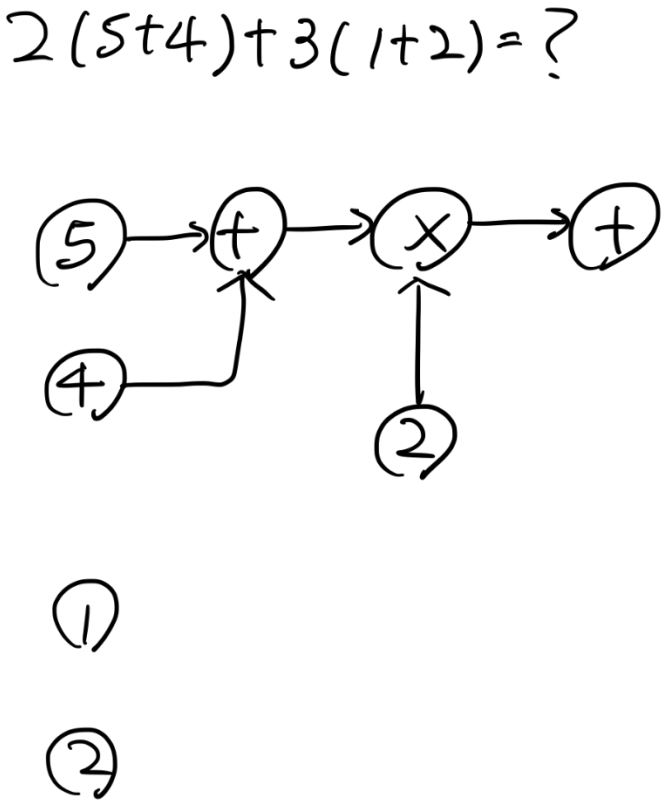
Data type

- 그래프 생성



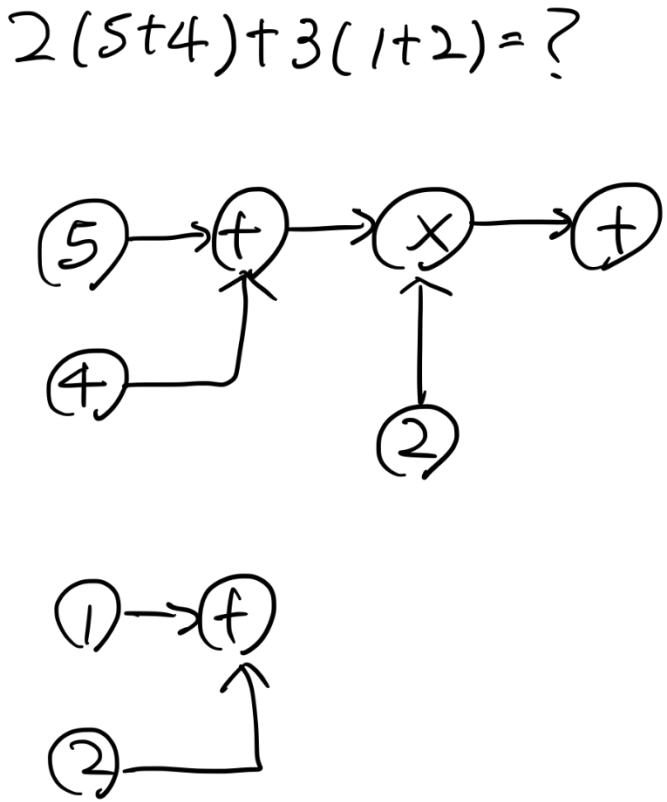
Data type

- 그래프 생성



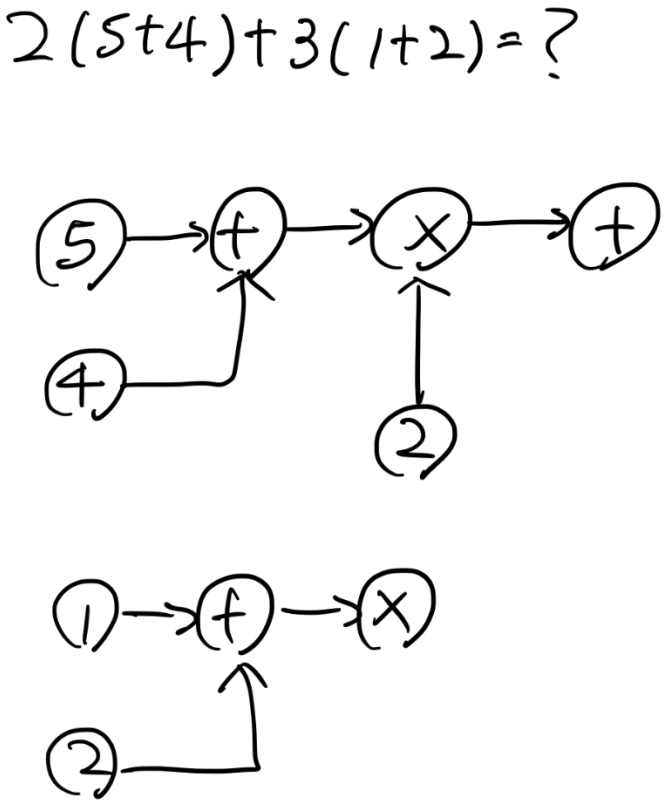
Data type

- 그래프 생성



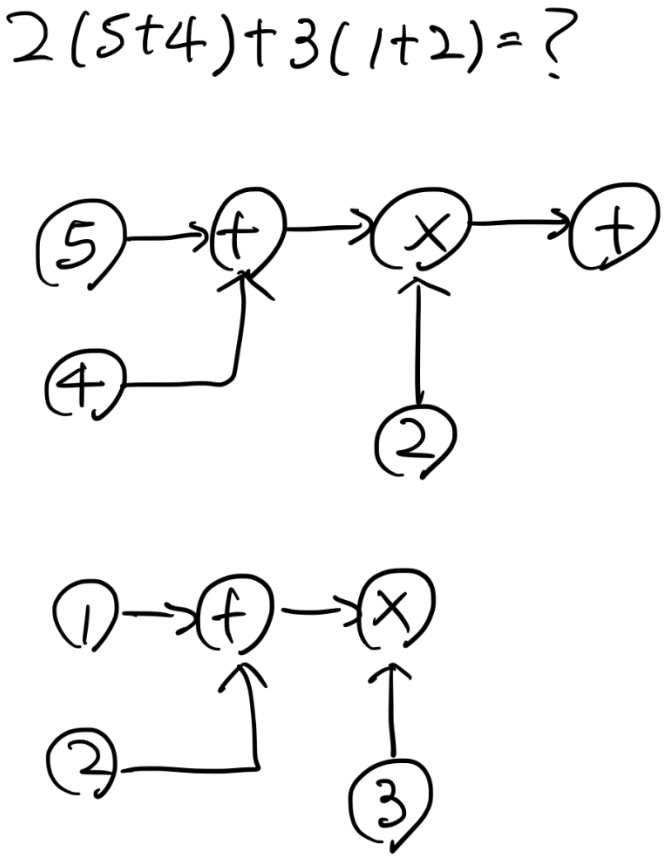
Data type

- 그래프 생성



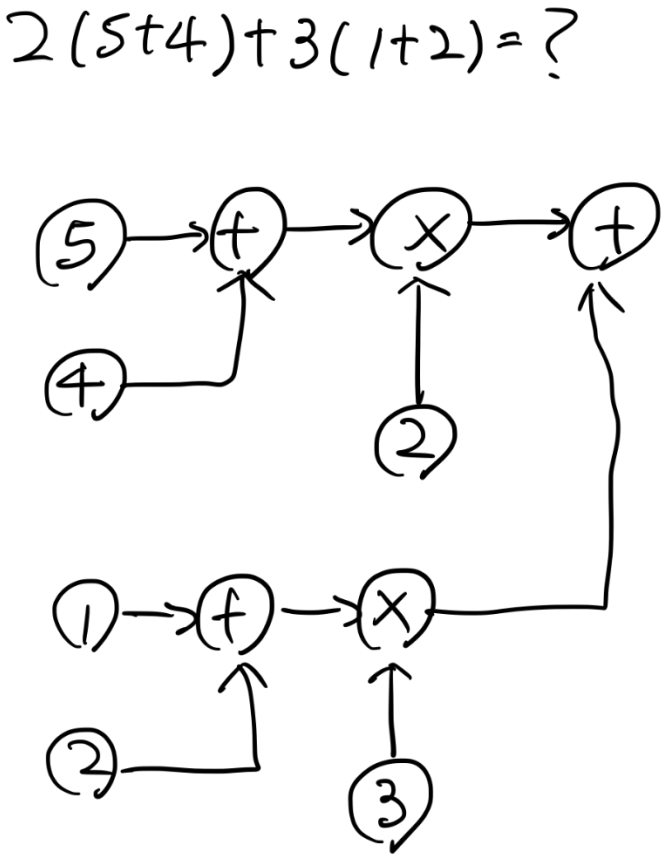
Data type

- 그래프 생성



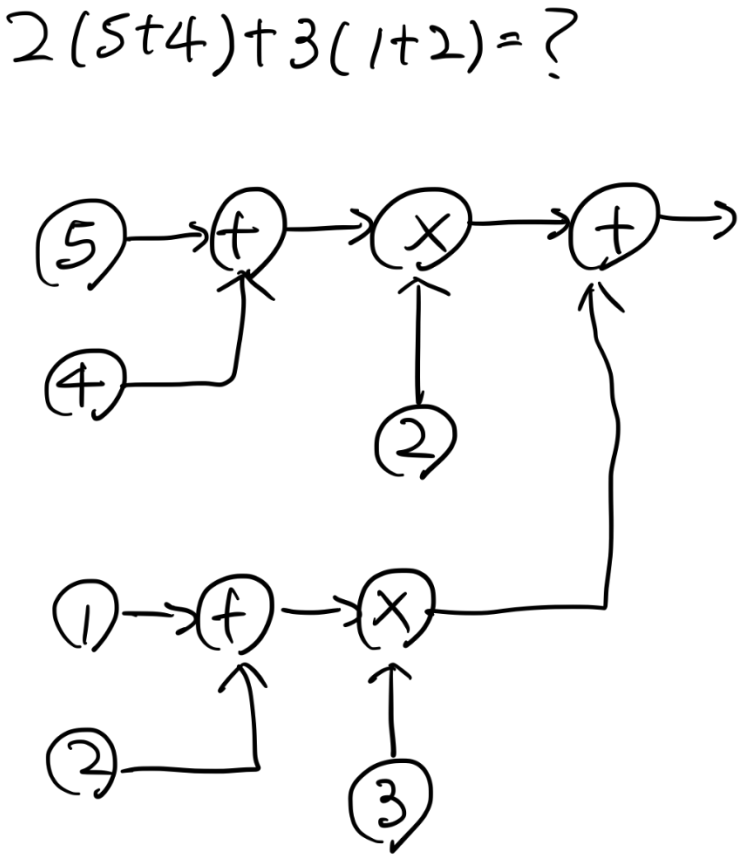
Data type

- 그래프 생성



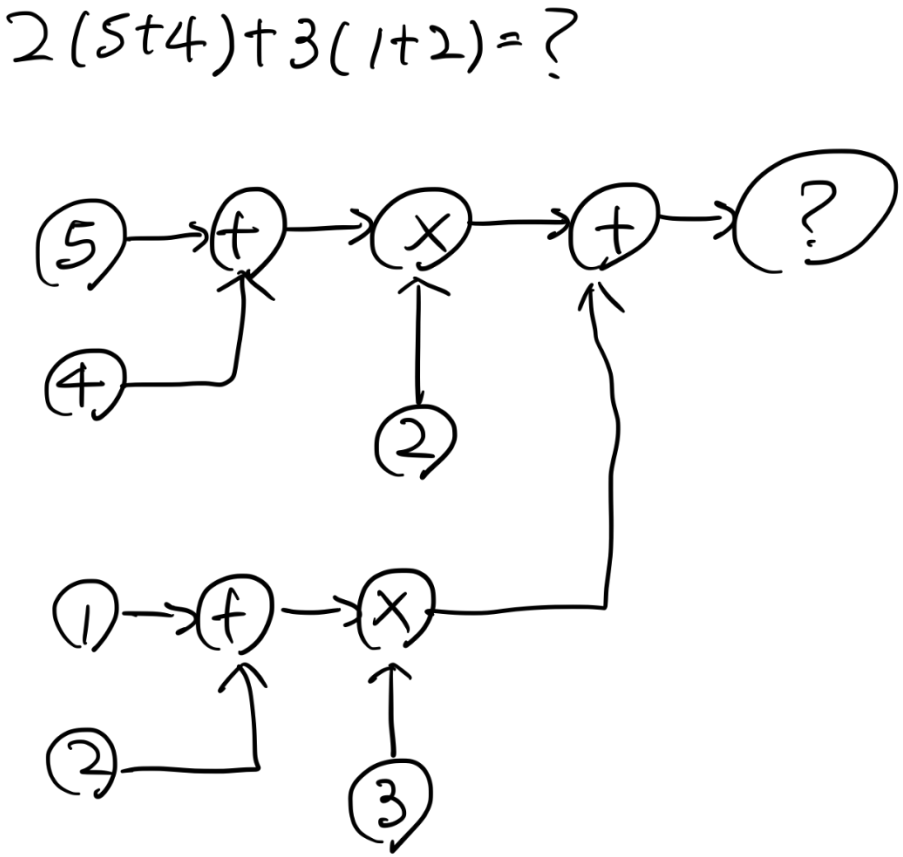
Data type

- 그래프 생성



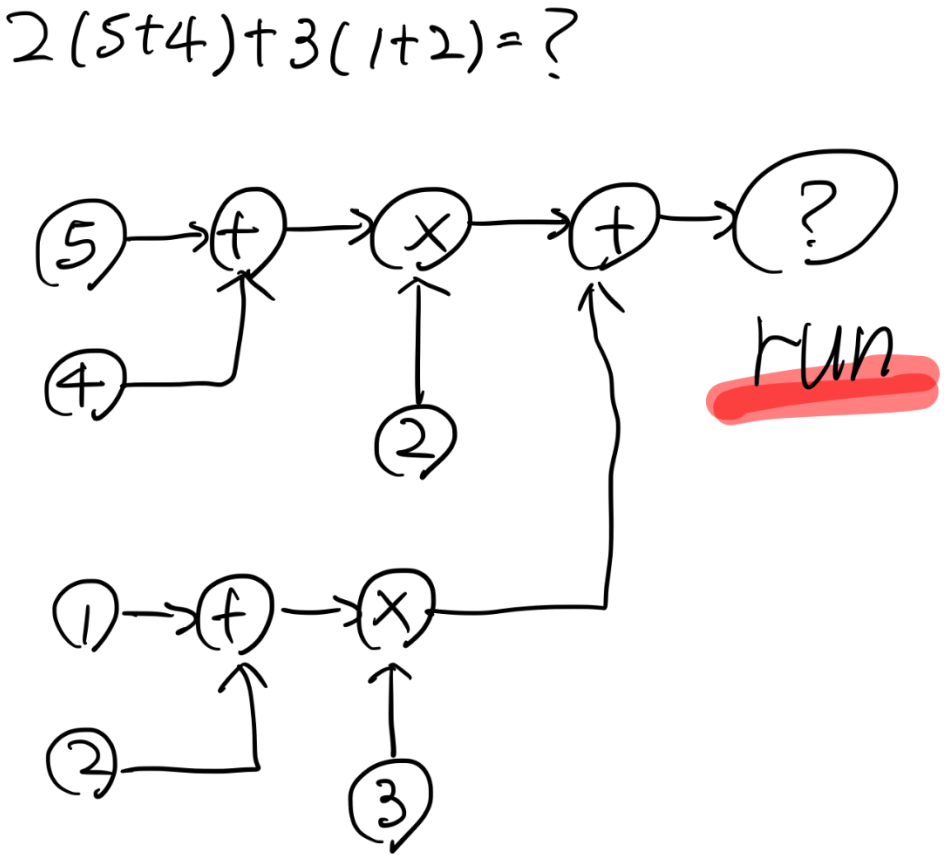
Data type

- 그래프 생성



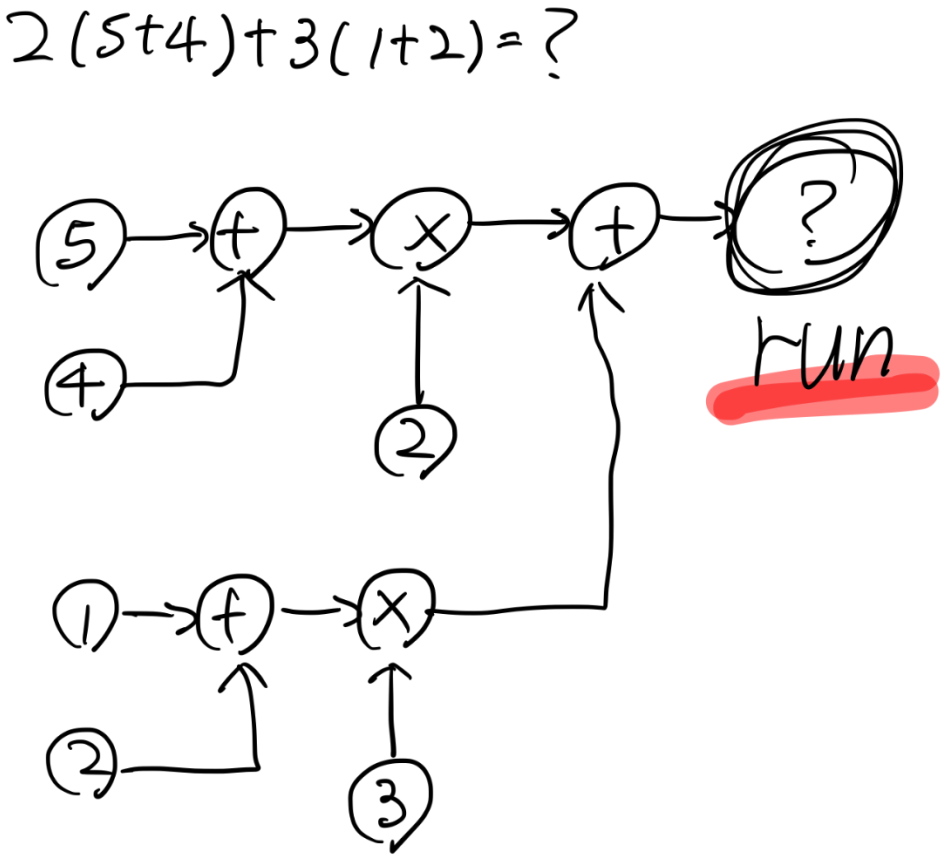
Data type

- 그래프 생성



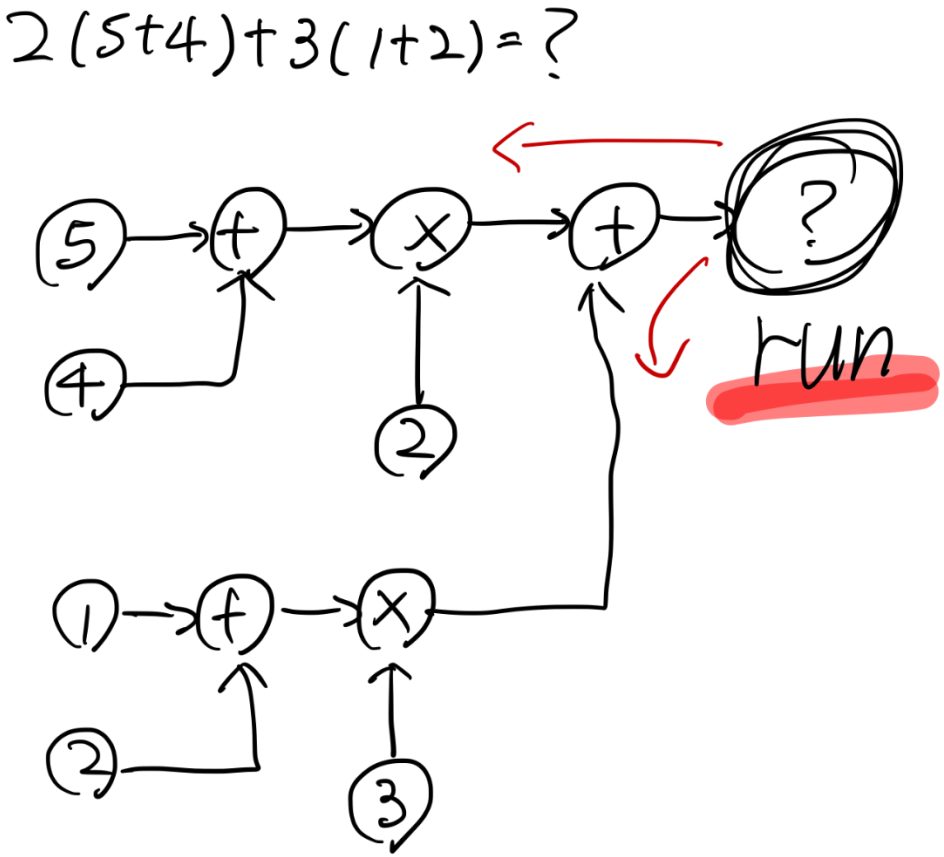
Data type

- 그래프 생성



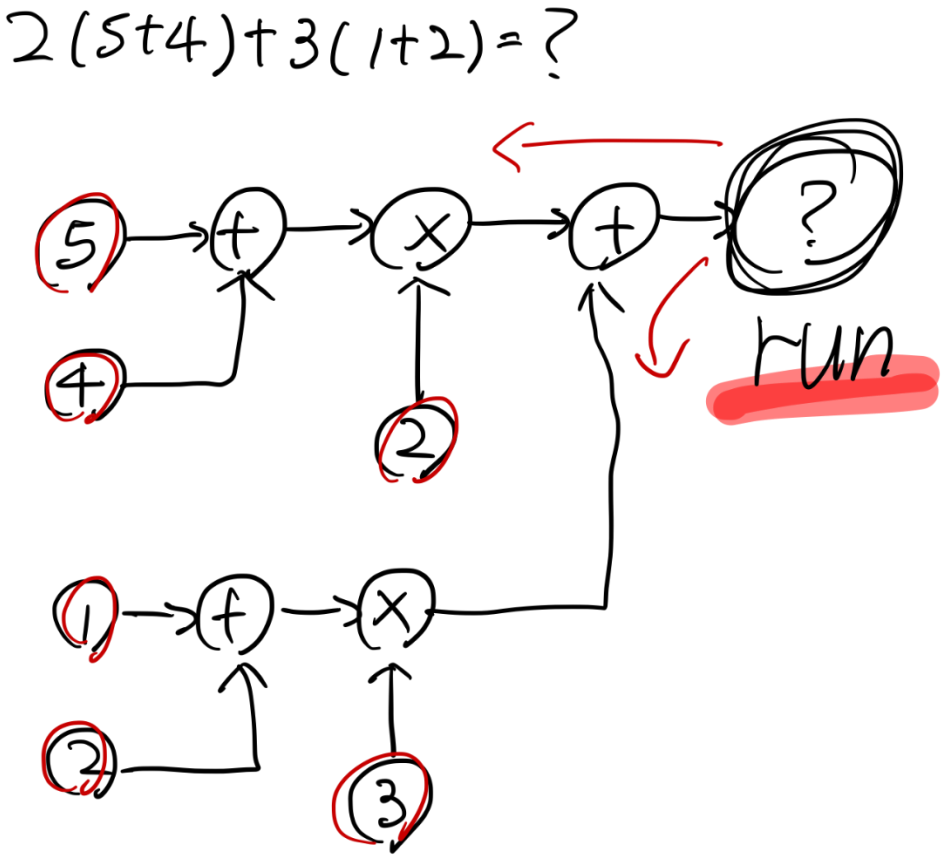
Data type

- 그래프 생성



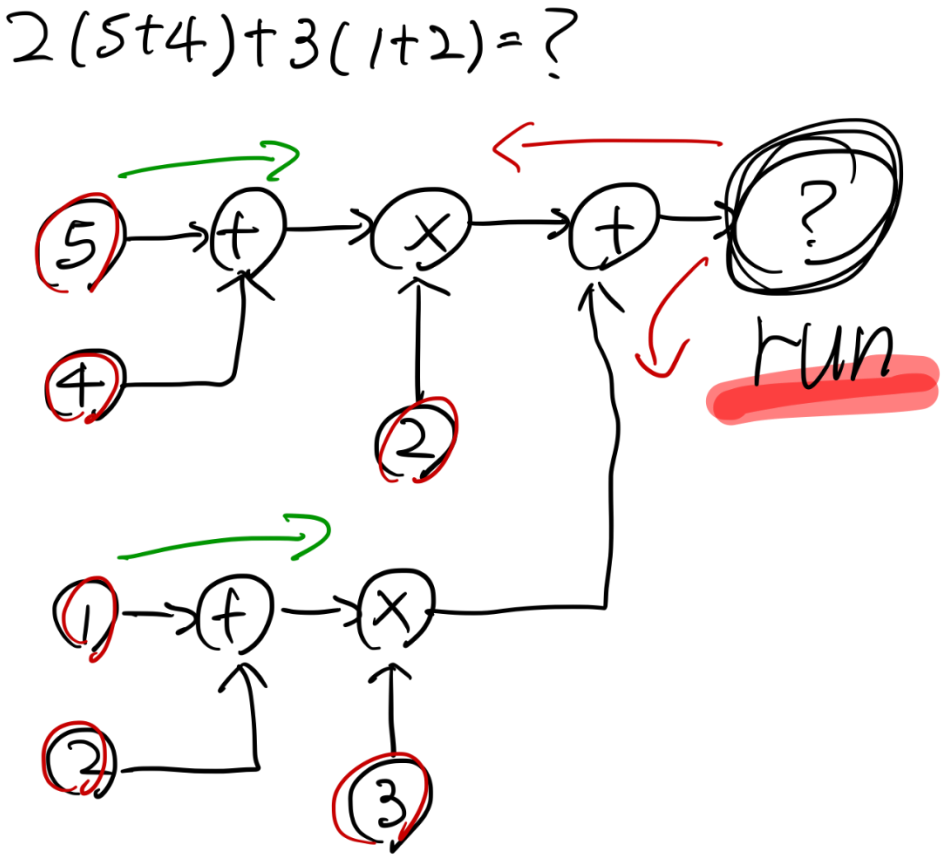
Data type

- 그래프 생성



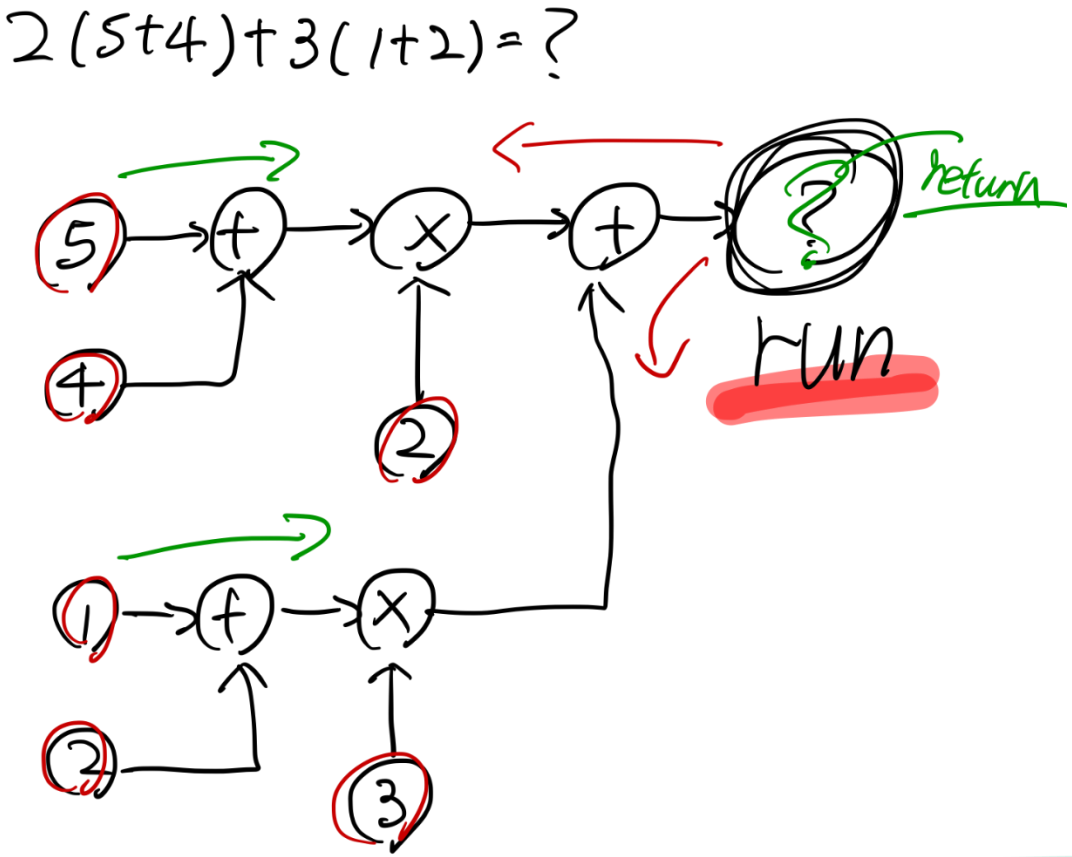
Data type

- 그래프 생성



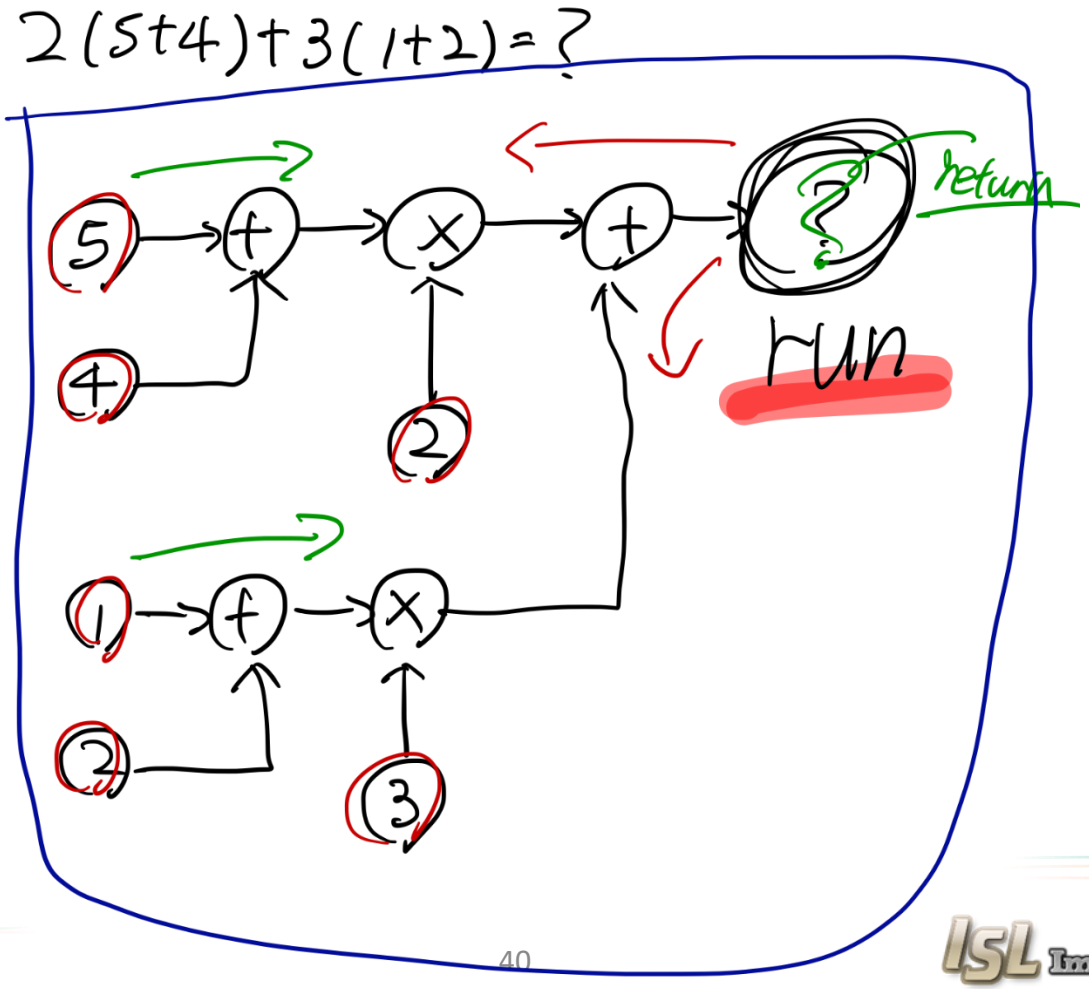
Data type

- 그래프 생성



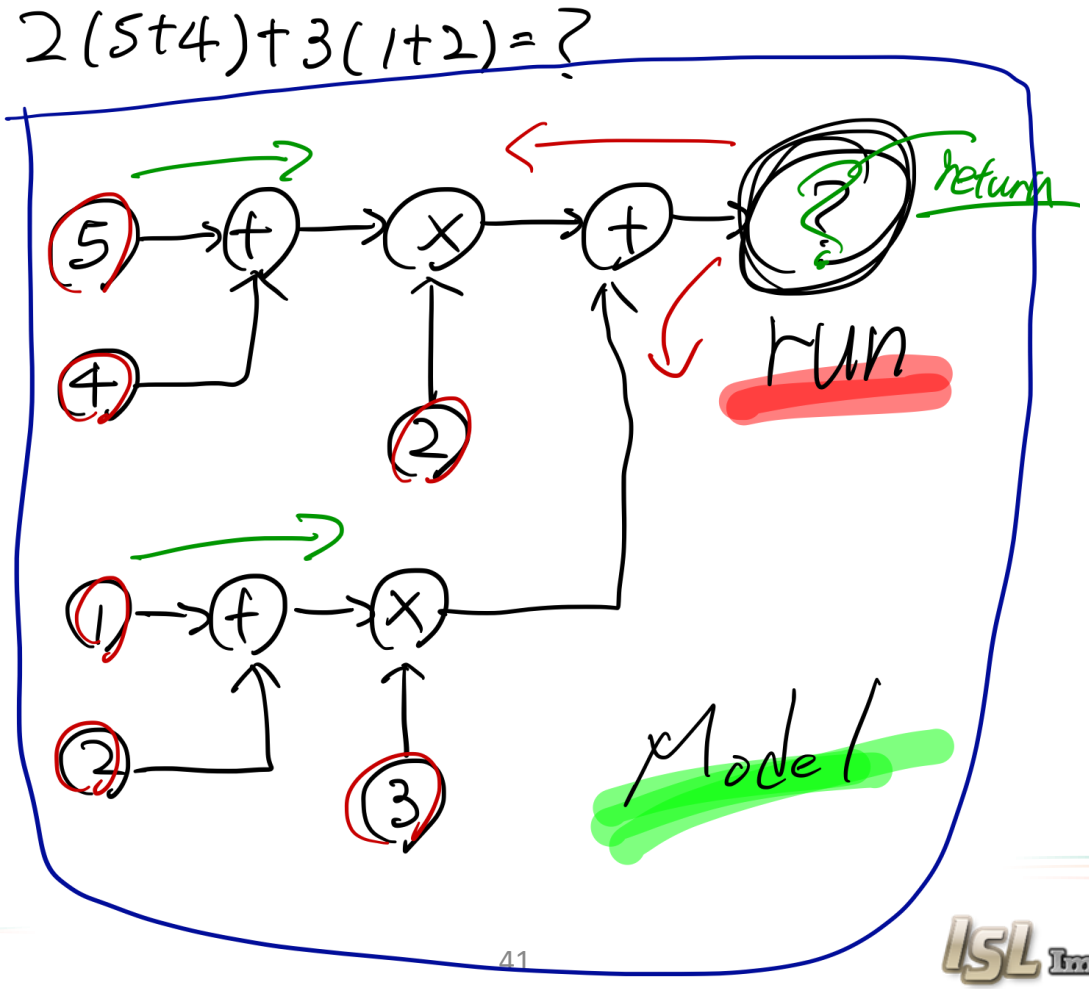
Data type

- 그래프 생성



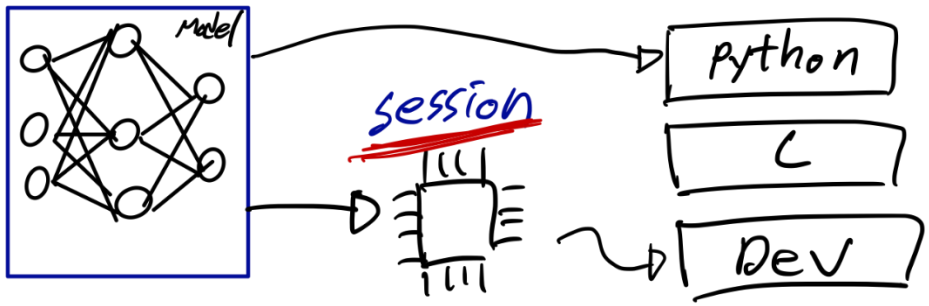
Data type

- 그래프 생성



Data type

- 세션



- Python이 C기반이기 때문에 상호간의 호환성이 좋음
- Python에서 생성된 모델은 디바이스에서 실제 구조가 만들어짐. 이 과정을 '세션에 보낸다.' 또는 '세션에 올린다.' 고 표현함.
- 하드웨어 단계에서 구현되었기 때문에 매우 빠른 연산이 가능해짐 -> TF의 장점

Data type

- 테스트 - constant

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

In [2]:
```
ph = tf.placeholder(dtype = tf.float32, shape=[3,3])
var = tf.Variable([1, 2, 3, 4, 5], dtype = tf.float32)
const = tf.constant([10, 20, 30, 40, 50], dtype = tf.float32)
```

In [3]:
```
print(ph)
```

```
Tensor("Placeholder:0", shape=(3, 3), dtype=float32)
```

In [4]:
```
print(var)
```

```
Tensor("Variable/read:0", shape=(5,), dtype=float32)
```

In [5]:
```
print(const)
```

```
Tensor("Const:0", shape=(5,), dtype=float32)
```

In [6]:
```
a = tf.constant([2])
b = tf.constant([3])
c = tf.constant([5])
d = a*b+c
print(d)
```

```
Tensor("add:0", shape=(1,), dtype=int32)
```

Data type

- 테스트 - constant

In [6]:
```
a = tf.constant([2])
b = tf.constant([3])
c = tf.constant([5])
d = a*b+c
print(d)
```

```
Tensor("add:0", shape=(1,), dtype=int32)
```

In [7]:
```
sess = tf.Session()
res = sess.run(d)
print(res)
```

```
[11]
```

In [ ]:

Data type

- 테스트 - Variable

```
In [7]: sess = tf.Session()
        res = sess.run(d)
        print(res)

        [11]

In [8]: var_a = tf.Variable([2])
        var_b = tf.Variable([10])
        var_c = tf.Variable([7])
        var_d = var_a * var_b + var_c
        print(var_d)

        Tensor("add_1:0", shape=(1,), dtype=int32)
```

Data type

- 테스트 - Variable

```
In [9]: var_res = sess.run(var_d)
---------------------------------------------------------------------------
FailedPreconditionError                    Traceback (most recent call last)
D:₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩client₩session.py in _do_call(sel
   1020     try:
-> 1021       return fn(*args)
   1022     except errors.OpError as e:

D:₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩client₩session.py in _run_fn(sess
etadata)
   1002                                  feed_dict, fetch_list, target_list,
-> 1003                                  status, run_metadata)
   1004

D:₩Anaconda3₩lib₩contextlib.py in __exit__(self, type, value, traceback)
     65             try:
---> 66                 next(self.gen)
     67             except StopIteration:

D:₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩framework₩errors_impl.py in raise
    469           compat.as_text(pywrap_tensorflow.TF_Message(status)),

In [ ]:
```

Variable은 별도의 초기화가 필요함!

Data type

- 테스트 - Variable

```
D:₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩framework₩errors_impl.py in raise
```

```
In [10]: init = tf.global_variables_initializer()
sess.run(init)
var_res = sess.run(var_d)
print(var_res)
```

```
[27]
```

```
In [ ]:
```

Data type

- 테스트 - placeholder

```
value1 = 10
value2 = 50
value3 = 14

ph_d = ph_a * ph_b + ph_c

tensor_map = {ph_a : value1, ph_b : value2, ph_c : value3}

print(ph_a)
print(ph_b)
print(ph_c)
print(ph_d)
```

```
Tensor("Placeholder_1:0", dtype=float32)
Tensor("Placeholder_2:0", dtype=float32)
Tensor("Placeholder_3:0", dtype=float32)
Tensor("add_2:0", dtype=float32)
```

In [12]:

```
ph_res = sess.run(ph_d, feed_dict=tensor_map)
print(ph_res)
```

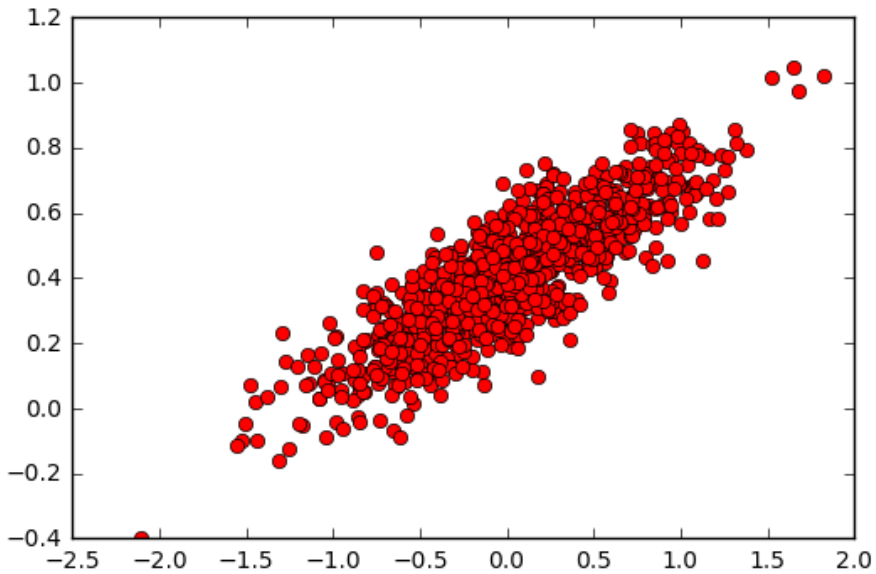
```
514.0
```

In [ ]:

Linear regression + Plot

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
```

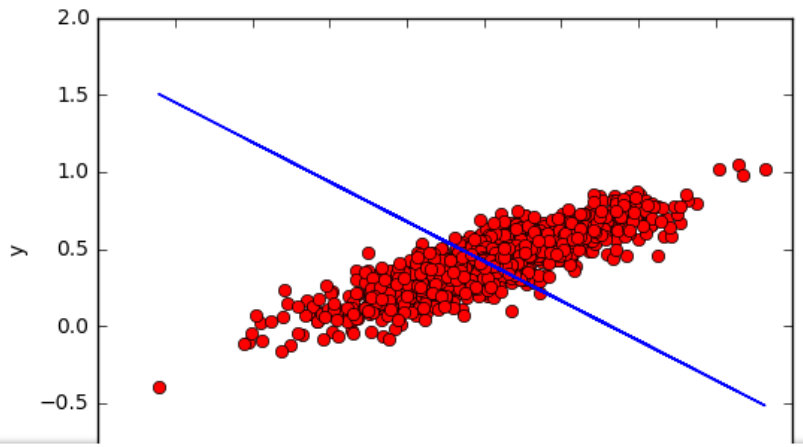
In [5]:
```
num_point = 1000
vectors_set = []
for i in range(num_point):
    x1 = np.random.normal(0.0, 0.55)
    y1 = x1 * 0.3 + 0.4 + np.random.normal(0,0 + 0.1)
    vectors_set.append([x1, y1])

x_data = [v[0] for v in vectors_set]
y_data = [v[1] for v in vectors_set]

plt.plot(x_data, y_data, 'ro')
plt.show()
```



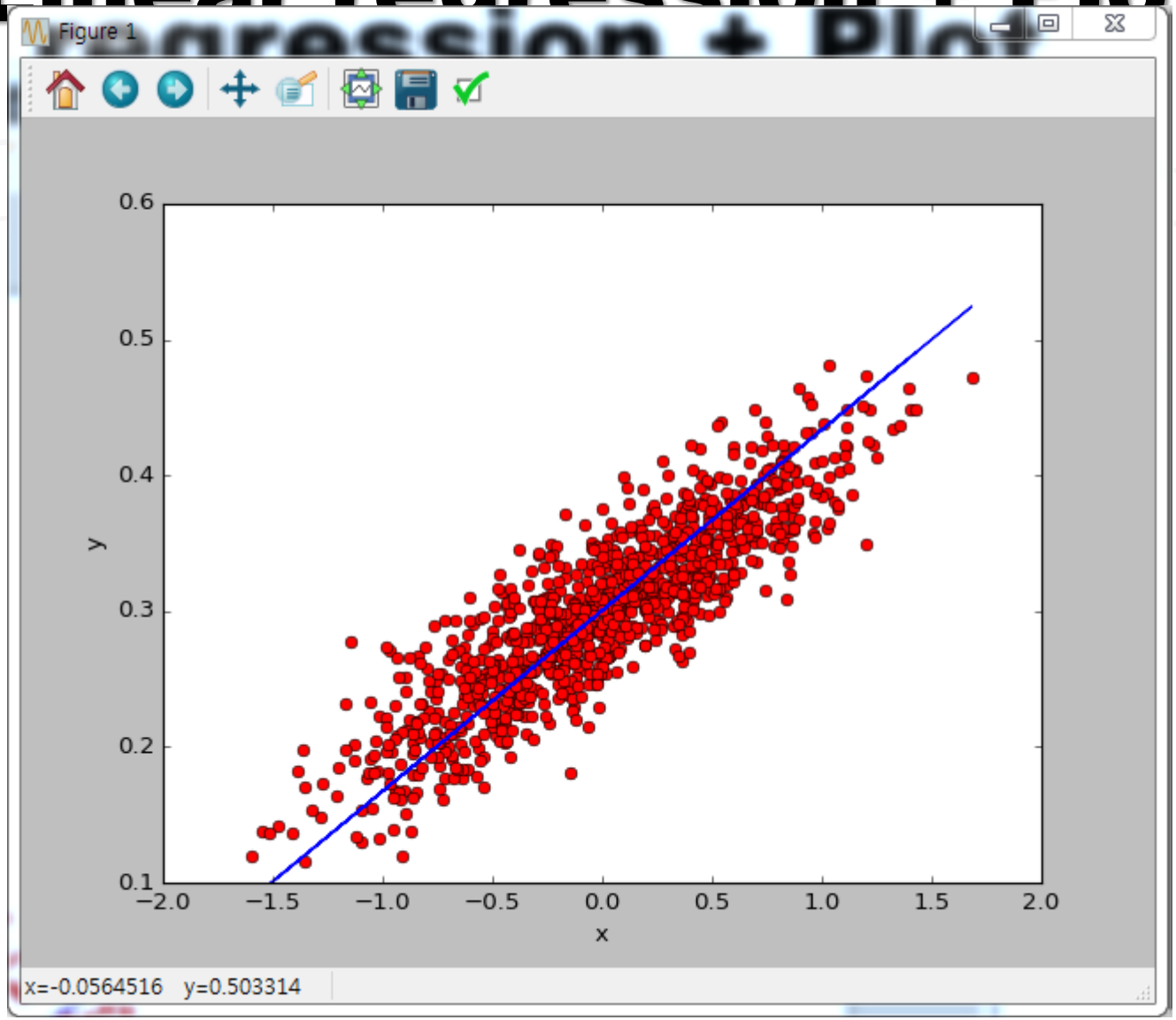
Linear regression + Plot

In [6]:
```
W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))
y = W * x_data + b

loss = tf.reduce_mean(tf.square(y - y_data)) # cost function
optimizer = tf.train.GradientDescentOptimizer(0.5) #학습 파라미터
train = optimizer.minimize(loss)
#모델(그래프) 설계 완료

#변수 초기화 및 세션 열기
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for step in range(10):
    sess.run(train)
    print(step, sess.run(W), sess.run(b), sess.run(loss))

    #데이터 분포
    plt.plot(x_data, y_data, 'ro')
    #직선(pred)
    plt.plot(x_data, sess.run(W) * x_data + sess.run(b))

    plt.xlabel('x')
    plt.ylabel('y')
    plt.show()
```

```
0 [-0.51449585] [ 0.42089418] 0.212177
```



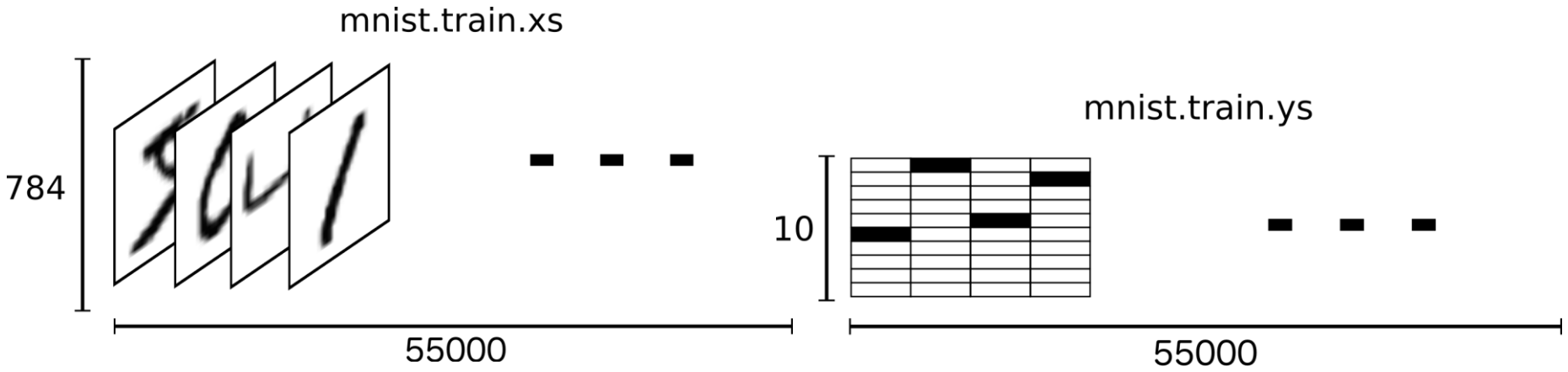
Linear regression + Plot



MNIST

- MNIST

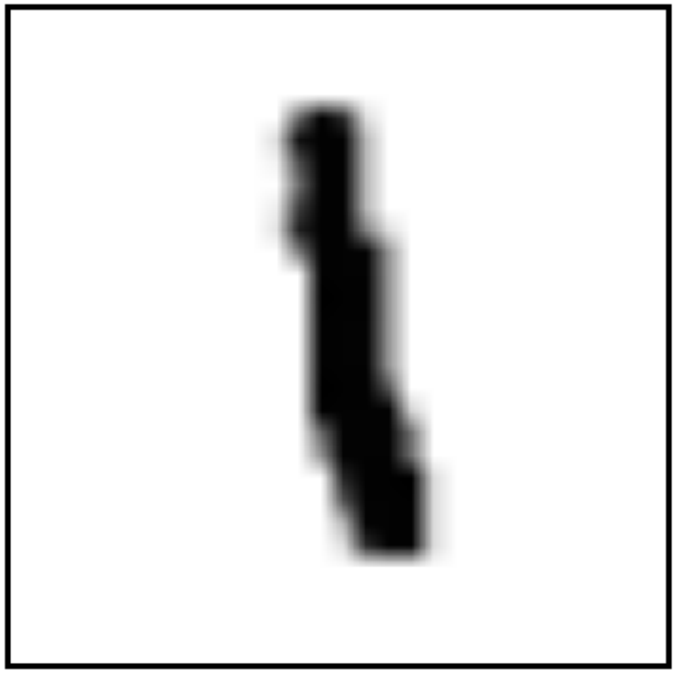
When one learns how to program, there's a tradition that the first thing you do is print "Hello World." Just like programming has Hello World, machine learning has MNIST.



MNIST

- MNIST

When one learns how to program, there's a tradition that the first thing you do is print "Hello World." Just like programming has Hello World, machine learning has MNIST.



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

MLP

- Input layer - 3 hidden layer – output layer

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("/tmp/data/", one_hot = True)

n_nodes_hl1 = 500
n_nodes_hl2 = 500
n_nodes_hl3 = 500

n_classes = 10
batch_size = 100

x = tf.placeholder('float', [None, 784])
y = tf.placeholder('float')


hidden_1_layer = {'weights':tf.Variable(tf.random_normal([784, n_nodes_hl1])),
                  'biases':tf.Variable(tf.random_normal([n_nodes_hl1]))}

hidden_2_layer = {'weights':tf.Variable(tf.random_normal([n_nodes_hl1, n_nodes_hl2])),
                  'biases':tf.Variable(tf.random_normal([n_nodes_hl2]))}

hidden_3_layer = {'weights':tf.Variable(tf.random_normal([n_nodes_hl2, n_nodes_hl3])),
                  'biases':tf.Variable(tf.random_normal([n_nodes_hl3]))}

output_layer = {'weights':tf.Variable(tf.random_normal([n_nodes_hl3, n_classes])),
                'biases':tf.Variable(tf.random_normal([n_classes])),}


l1 = tf.add(tf.matmul(x,hidden_1_layer['weights']), hidden_1_layer['biases'])
l1 = tf.nn.relu(l1)

l2 = tf.add(tf.matmul(l1,hidden_2_layer['weights']), hidden_2_layer['biases'])
l2 = tf.nn.relu(l2)

l3 = tf.add(tf.matmul(l2,hidden_3_layer['weights']), hidden_3_layer['biases'])
l3 = tf.nn.relu(l3)
```

MLP

- Input layer - 3 hidden layer – output layer

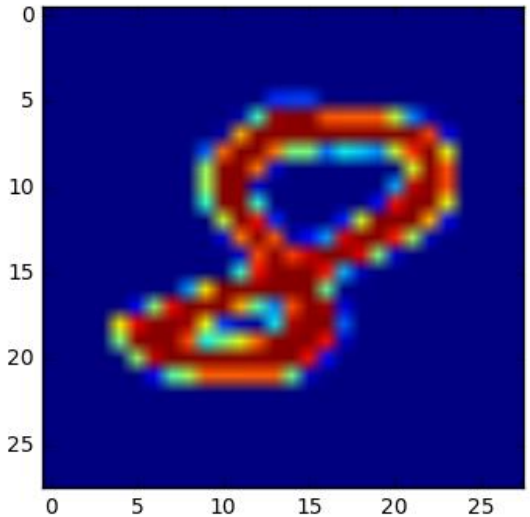
```
output = tf.matmul(l3,output_layer['weights']) + output_layer['biases']


prediction = output
cost = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits(prediction,y) )
optimizer = tf.train.AdamOptimizer().minimize(cost)

hm_epochs = 10
sess = tf.Session()
sess.run(tf.global_variables_initializer())

for epoch in range(hm_epochs):
    epoch_loss = 0
    for _ in range(int(mnist.train.num_examples/batch_size)):
        epoch_x, epoch_y = mnist.train.next_batch(batch_size)
        _, c = sess.run([optimizer, cost], feed_dict={x: epoch_x, y: epoch_y})
        epoch_loss += c

    print('Epoch', epoch + 1, 'completed out of',hm_epochs,'loss:',epoch_loss)

correct = tf.equal(tf.argmax(prediction, 1), tf.argmax(y, 1))

accuracy = tf.reduce_mean(tf.cast(correct, 'float'))
print('Accuracy:',sess.run(accuracy, feed_dict = {x:mnist.test.images, y:mnist.test.labels}))
```

```
Extracting /tmp/data/train-images-idx3-ubyte.gz
Extracting /tmp/data/train-labels-idx1-ubyte.gz
Extracting /tmp/data/t10k-images-idx3-ubyte.gz
Extracting /tmp/data/t10k-labels-idx1-ubyte.gz
Epoch 1 completed out of 10 loss: 1607590.13811
Epoch 2 completed out of 10 loss: 395378.241097
Epoch 3 completed out of 10 loss: 212075.841237
Epoch 4 completed out of 10 loss: 124842.024561
Epoch 5 completed out of 10 loss: 73650.398987
Epoch 6 completed out of 10 loss: 45980.5750535
Epoch 7 completed out of 10 loss: 29941.5561811
Epoch 8 completed out of 10 loss: 23685.460072
Epoch 9 completed out of 10 loss: 19176.7452043
Epoch 10 completed out of 10 loss: 17872.4552318
Accuracy: 0.9494
```

MLP

- Input layer - 3 hidden layer – output layer

```
import numpy as np
s = 61
arr = np.array(mnist.test.images[s])
arr.shape = (28, 28)

import matplotlib.pyplot as plt

plt.imshow(arr)
plt.show()

arr.shape = (1, 784)
print('GT   : ', np.argmax(mnist.test.labels[s]))
print('Pred : ',np.argmax(sess.run(output, feed_dict={x:arr})))
```

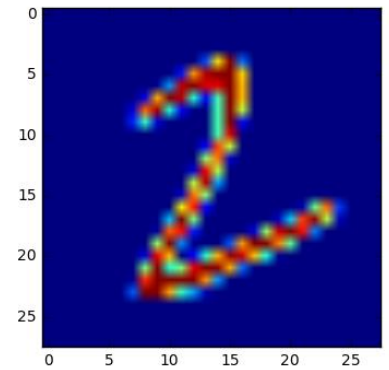



```
GT   : 8
Pred : 8
```

MLP

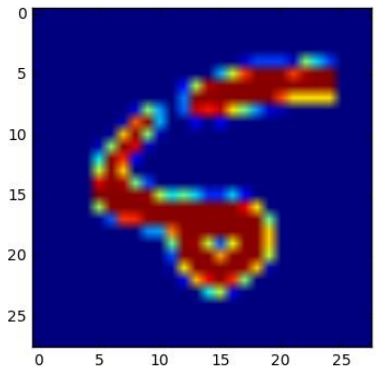
- Failure

```
import numpy as np
s = 38
arr = np.array(mnist.test.images[s])
arr.shape = (28, 28)

import matplotlib.pyplot as plt

plt.imshow(arr)
plt.show()

arr.shape = (1, 784)
print('GT   : ', np.argmax(mnist.test.labels[s]))
print('Pred : ',np.argmax(sess.run(output, feed_dict={x:arr})))
```



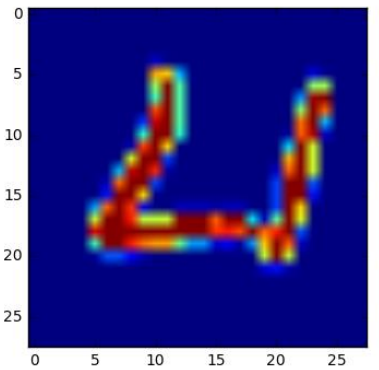
```
GT   : 2
Pred : 1
```

```
import numpy as np
s = 8
arr = np.array(mnist.test.images[s])
arr.shape = (28, 28)

import matplotlib.pyplot as plt

plt.imshow(arr)
plt.show()

arr.shape = (1, 784)
print('GT   : ', np.argmax(mnist.test.labels[s]))
print('Pred : ',np.argmax(sess.run(output, feed_dict={x:arr})))
```



```
GT   : 5
Pred : 6
```

```
import numpy as np
s = 33
arr = np.array(mnist.test.images[s])
arr.shape = (28, 28)

import matplotlib.pyplot as plt

plt.imshow(arr)
plt.show()

arr.shape = (1, 784)
print('GT   : ', np.argmax(mnist.test.labels[s]))
print('Pred : ',np.argmax(sess.run(output, feed_dict={x:arr})))
```

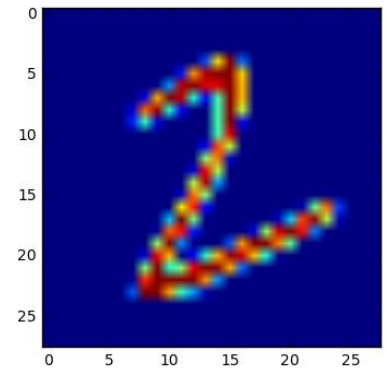


```
GT   : 4
Pred : 0
```

MLP

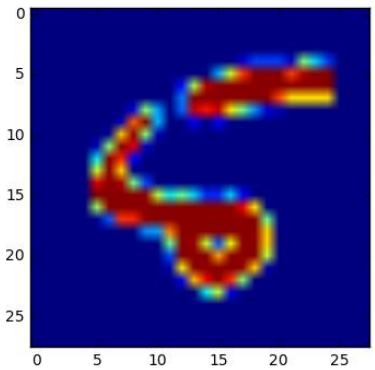
- Failure

```
import numpy as np
s = 38
arr = np.array(mnist.test.images[s])
arr.shape = (28, 28)

import matplotlib.pyplot as plt

plt.imshow(arr)
plt.show()

arr.shape = (1, 784)
print('GT   : ', np.argmax(mnist.test.labels[s]))
print('Pred : ',np.argmax(sess.run(output, feed_dict={x:arr})))
```



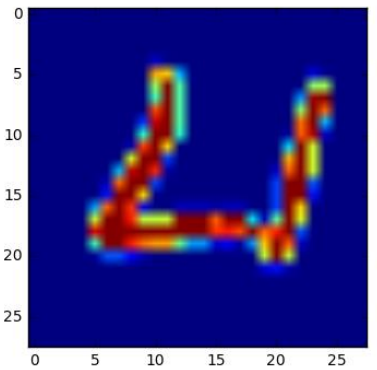
GT : 2
Pred : 1

```
import numpy as np
s = 8
arr = np.array(mnist.test.images[s])
arr.shape = (28, 28)

import matplotlib.pyplot as plt

plt.imshow(arr)
plt.show()

arr.shape = (1, 784)
print('GT   : ', np.argmax(mnist.test.labels[s]))
print('Pred : ',np.argmax(sess.run(output, feed_dict={x:arr})))
```



GT : 5
Pred : 6

```
import numpy as np
s = 33
arr = np.array(mnist.test.images[s])
arr.shape = (28, 28)

import matplotlib.pyplot as plt

plt.imshow(arr)
plt.show()

arr.shape = (1, 784)
print('GT   : ', np.argmax(mnist.test.labels[s]))
print('Pred : ',np.argmax(sess.run(output, feed_dict={x:arr})))
```



GT : 4
Pred : 0

CNN (ConvNet)

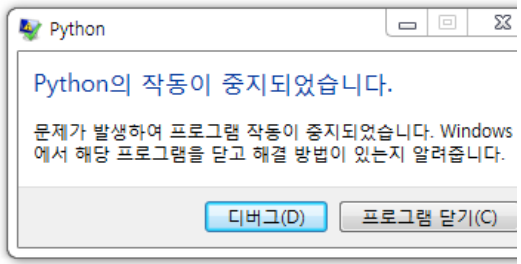
- 버그

In [13]:
```
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

In [*]:
```
for i in range(20000):#20000
    batch = mnist.train.next_batch(50)
    if i % 1000 == 0:
        train_accuracy = sess.run(accuracy, feed_dict={
                x:batch[0], y_: batch[1], keep_prob: 1.0})
        print("step %d, training accuracy %g"%(i, train_accuracy))
    sess.run(train_step,feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})
```

```
step 0, training accuracy 0.12
step 1000, training accuracy 0.94
step 2000, training accuracy 0.96
step 3000, training accuracy 1
step 4000, training accuracy 0.98
step 5000, training accuracy 1
step 6000, training accuracy 0.96
step 7000, training accuracy 1
step 8000, training accuracy 0.98
step 9000, training accuracy 0.96
step 10000, training accuracy 0.98
step 11000, training accuracy 1
step 12000, training accuracy 1
step 13000, training accuracy 1
step 14000, training accuracy 0.98
step 15000, training accuracy 1
step 16000, training accuracy 1
step 17000, training accuracy 1
step 18000, training accuracy 1
step 19000, training accuracy 1
```

CPU버전 삭제로 해결



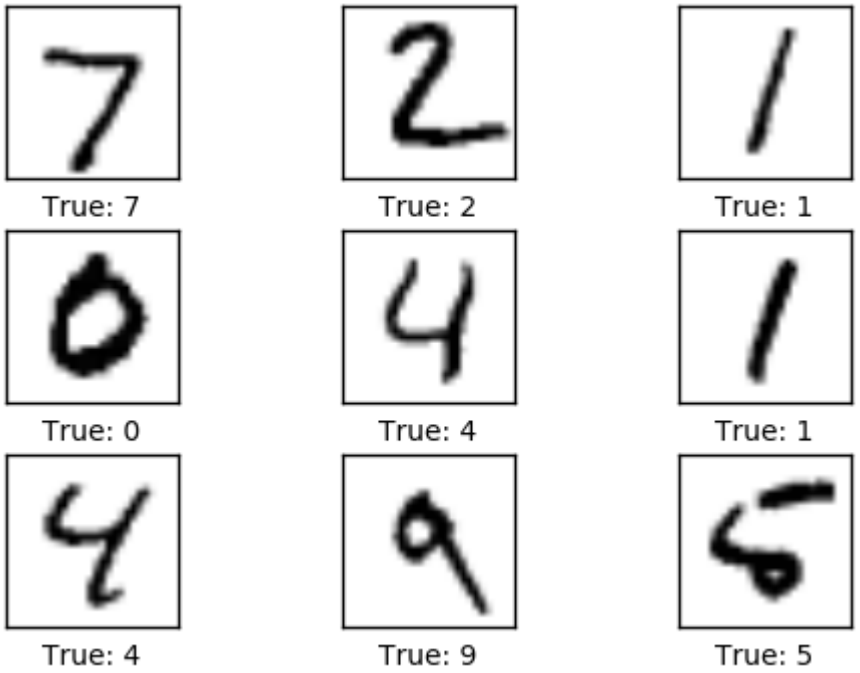
In [*]:
```
print("test accuracy %g"% sess.run(
        accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels, keep_prob: 1.0}))
```

CNN (ConvNet)

```
In [12]: cross_entropy = -tf.reduce_sum(y_*tf.log(y_conv))
         train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
         correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
         accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))

In [13]: sess = tf.Session()
         sess.run(tf.global_variables_initializer())

In [17]: for i in range(20000):#20000
             batch = mnist.train.next_batch(50)
             if i % 1000 == 0:

In [19]: print("Accuracy : ", sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels, keep_prob: 1.0}))
         Accuracy :  0.9927

         step 0, training accuracy 0.94
         step 1000, training accuracy 1
         step 2000, training accuracy 1
         step 3000, training accuracy 1
         step 4000, training accuracy 1
         step 5000, training accuracy 0.98
         step 6000, training accuracy 1
         step 7000, training accuracy 1
         step 8000, training accuracy 1
         step 9000, training accuracy 1
         step 10000, training accuracy 1
         step 11000, training accuracy 1
         step 12000, training accuracy 1
         step 13000, training accuracy 1
         step 14000, training accuracy 1
         step 15000, training accuracy 1
         step 16000, training accuracy 1
         step 17000, training accuracy 1
         step 18000, training accuracy 1
         step 19000, training accuracy 1
```

Q&A

example

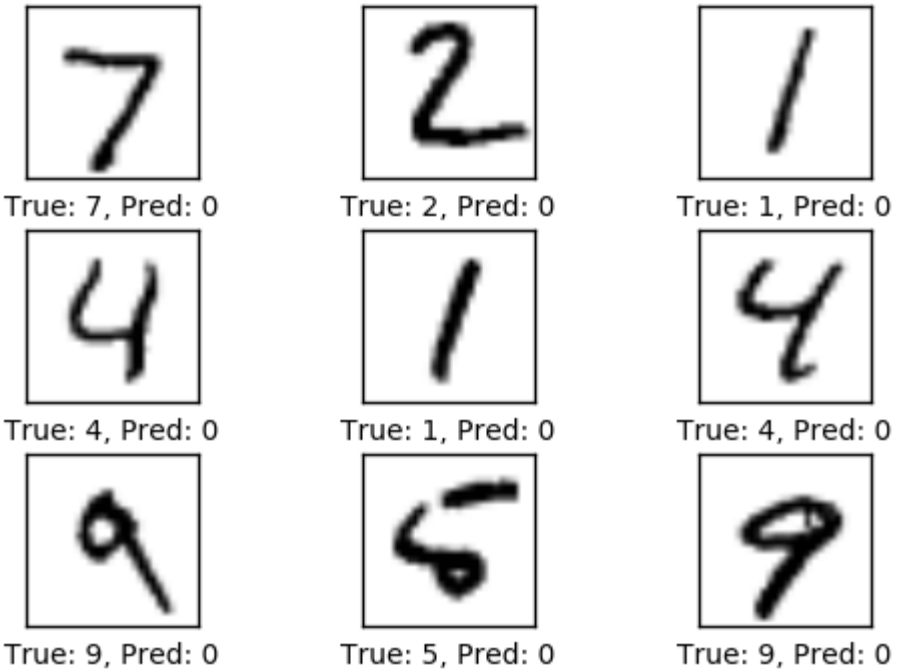


Before any opt

```
print_accuracy()
```

```
Accuracy on test-set: 9.8%
```

```
plot_example_errors()
```



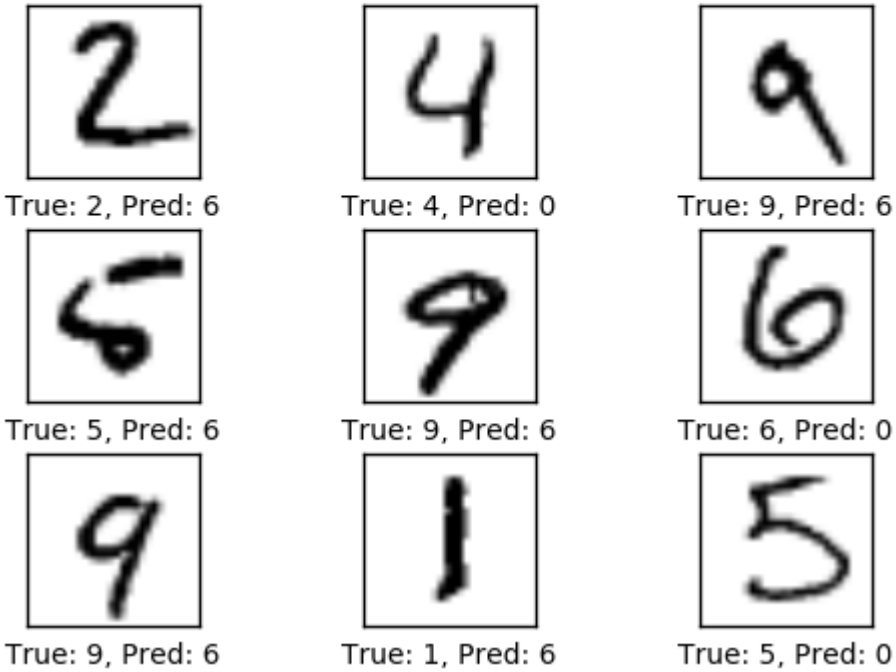
After 1 opt iter

```
optimize(num_iterations=1)
```

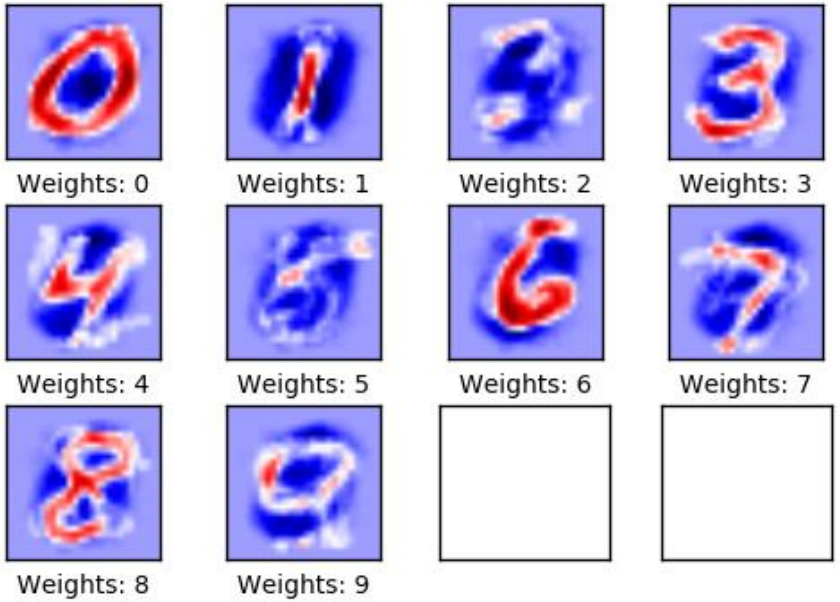
```
print_accuracy()
```

```
Accuracy on test-set: 40.7%
```

```
plot_example_errors()
```




```
plot_weights()
```



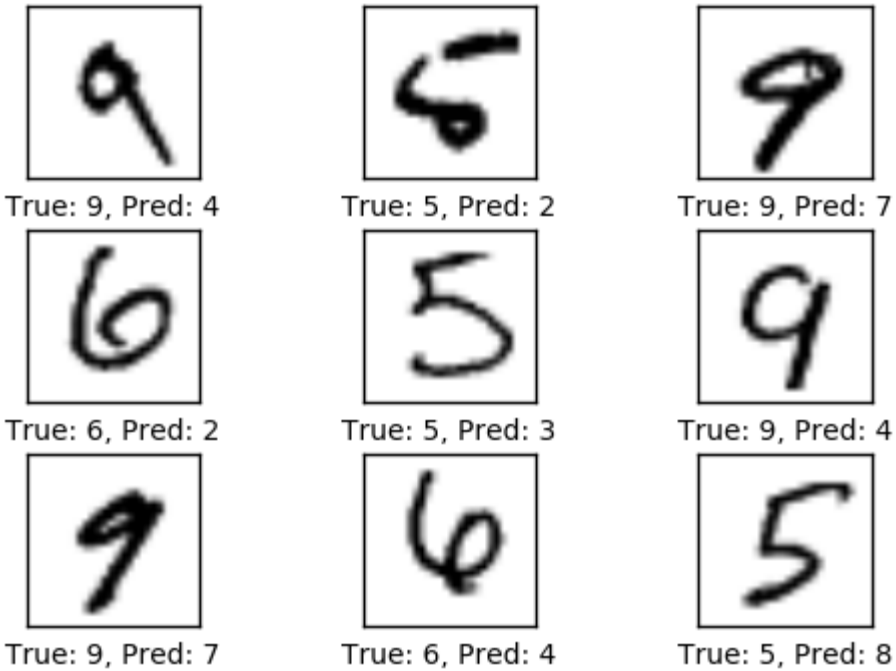
After 10 opt iter

```
# We have already performed 1 iteration.
optimize(num_iterations=9)
```

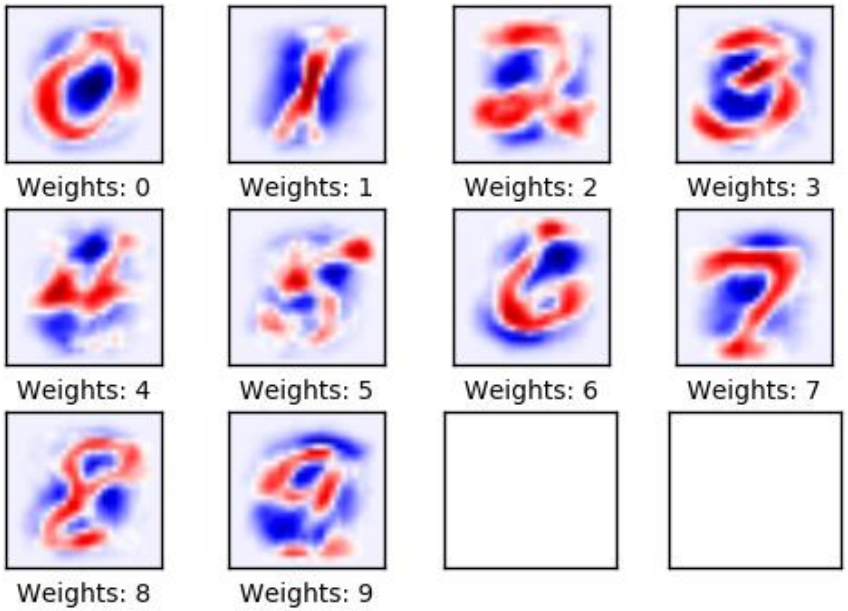
```
print_accuracy()
```

```
Accuracy on test-set: 78.2%
```

```
plot_example_errors()
```




```
plot_weights()
```



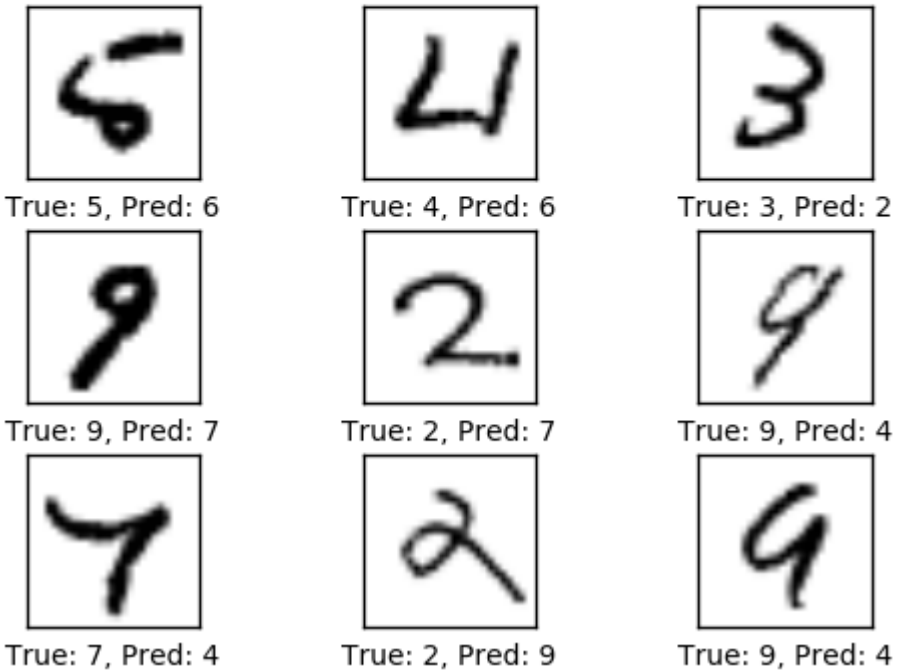
After 1000 opt iter

```
# We have already performed 10 iterations.
optimize(num_iterations=990)
```

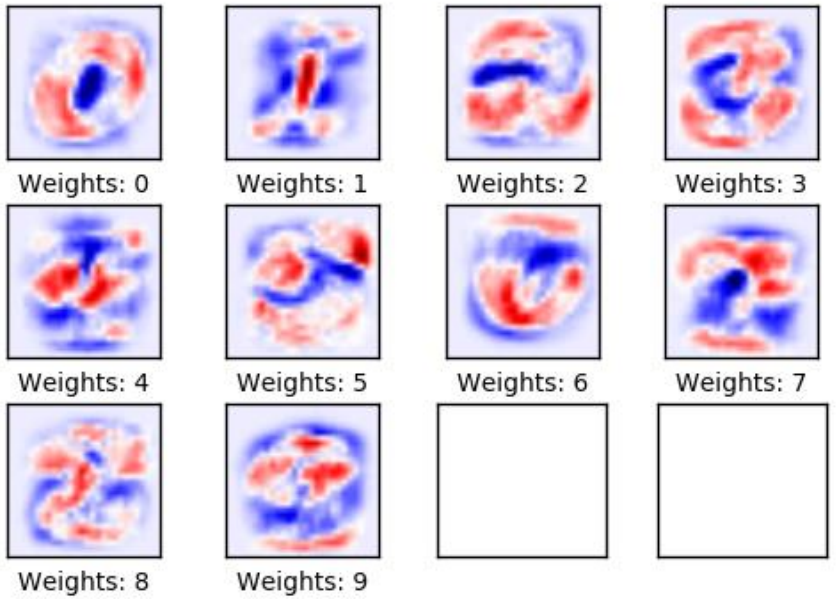
```
print_accuracy()
```

```
Accuracy on test-set: 91.7%
```

```
plot_example_errors()
```




```
plot_weights()
```



After 1000 opt iter

```
print_confusion_matrix()
```

```
[[ 957    0    3    2    0    5   11    1    1    0]
 [   0 1108    2    2    1    2    4    2   14    0]
 [   4    9  914   19   15    5   13   14   35    4]
 [   1    0   16  928    0   28    2   14   13    8]
 [   1    1    3    2  939    0   10    2    6   18]
 [  10    3    3   33   10  784   17    6   19    7]
 [   8    3    3    2   11   14  915    1    1    0]
 [   3    9   21    9    7    1    0  959    2   17]
 [   8    8    8   38   11   40   14   18  825    4]
 [  11    7    1   13   75   13    1   39    4  845]]
```