

Chapter01.

Tensorflow basic

작성자 : 김진성

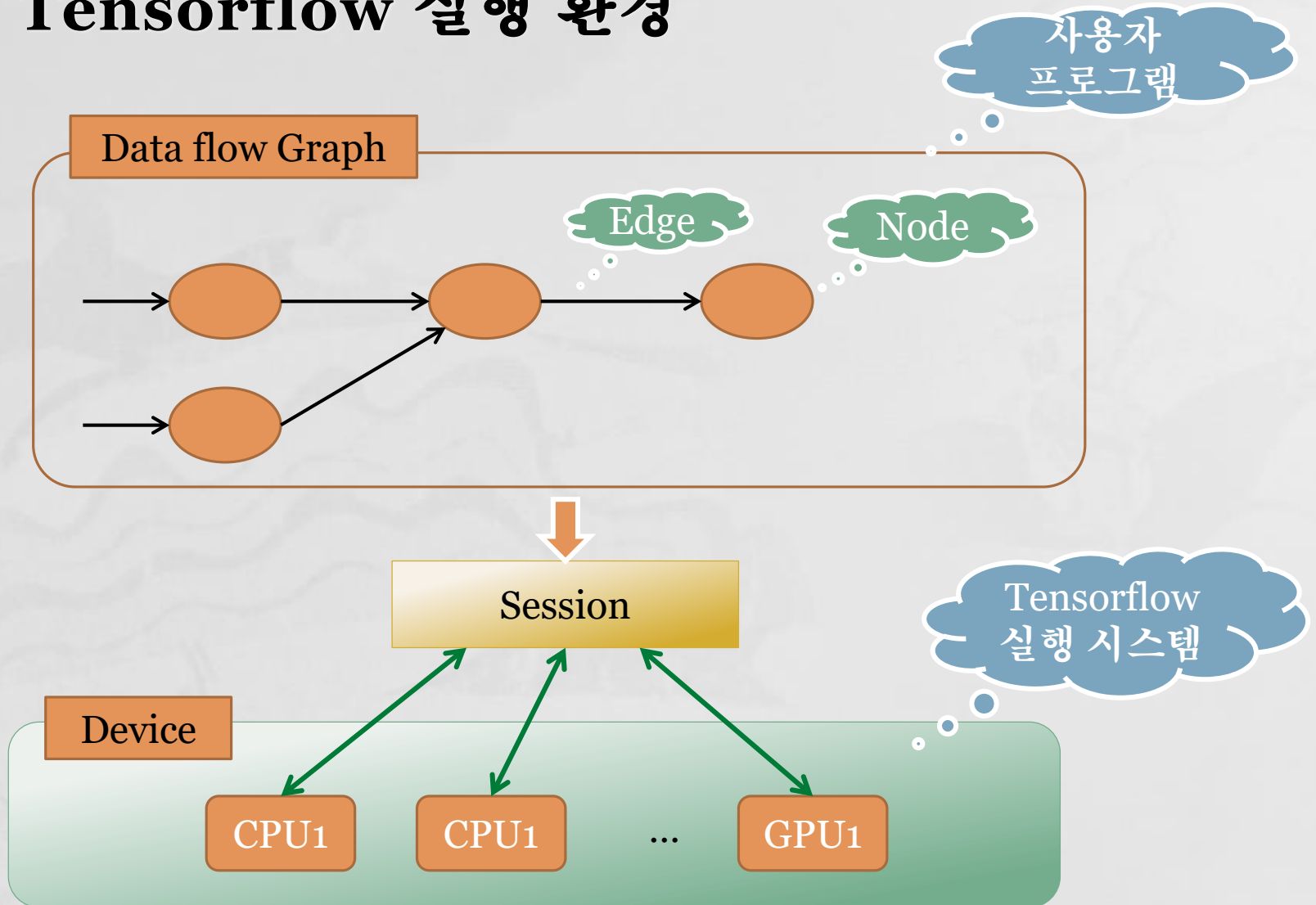
목차

1. Tensorflow 실행 환경
2. Tensorflow 프로그래밍 모델
3. Tensorboard
4. Session에서 변수 값 할당 방법
5. Tensorflow 2.x 주요 특징

1. Tensorflow 실행 환경



- Graph로 연산을 수행하는 프로그래밍 시스템
 - Graph : 점과 선(node와 edge)로 이루어진 수학적 구조
 - Node : Graph를 구성하는 사각형/타원 → 연산자(Operation)
 - Edge : Graph를 구성하는 선 → 텐서(Tensor) : 다차원배열
 - Session 상(내)에서 실행
- Session : 사용자 프로그램 + 실행 시스템 연결
 - 사용자 프로그램(Graph 작업)을 Device에 배정 및 실행 메서드(run) 제공
 - Device : Tensorflow 실행 시스템으로 사용자 프로그램 구동 장비(CPU,GPU)

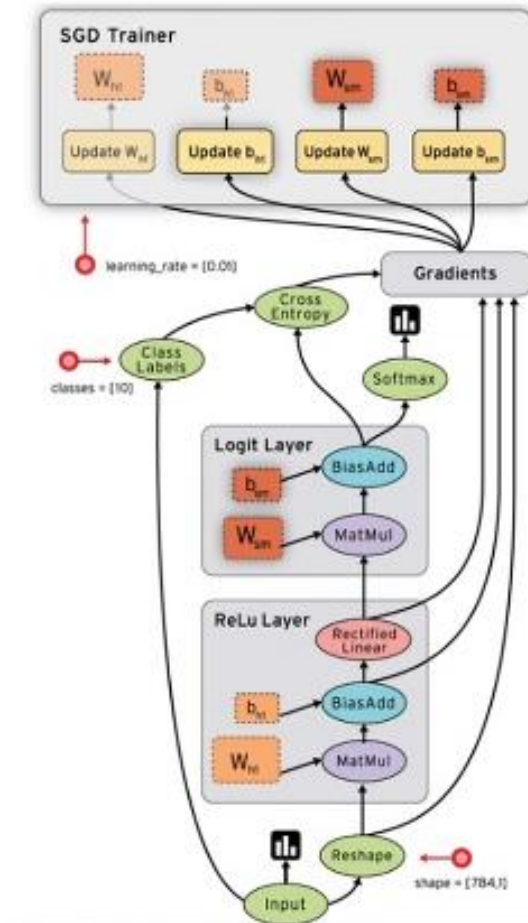
● Tensorflow 실행 환경



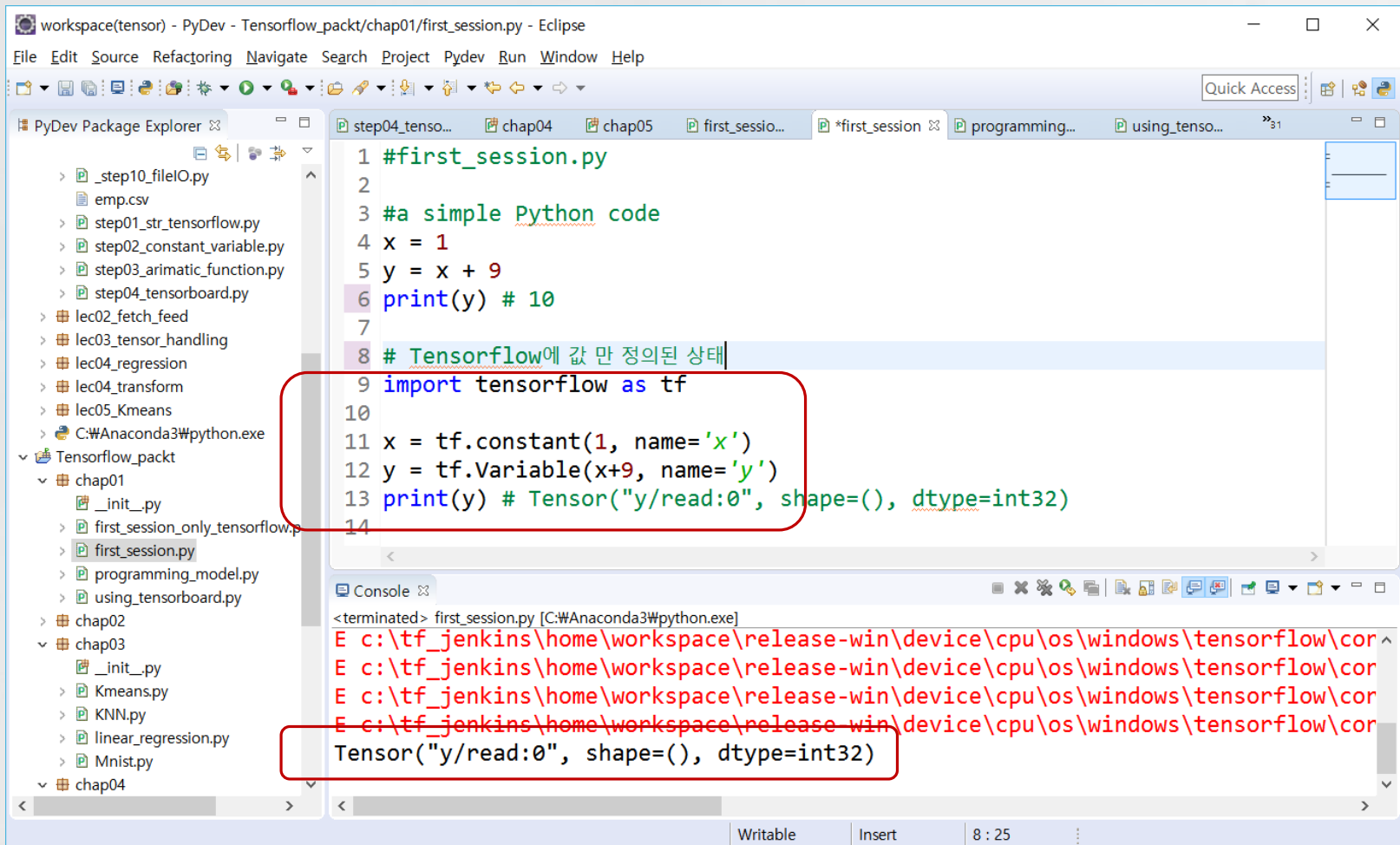
● Data Flow Graph

Node & Edge

- **Node** 
Mathematical operations
- **Edge** 
multidimensional data arrays
(tensors)



● Session 미 생성 : y변수 정의(값 할당)



```
workspace(tensor) - PyDev - Tensorflow_packt/chap01/first_session.py - Eclipse
File Edit Source Refactoring Navigate Search Project Pydev Run Window Help

PyDev Package Explorer
  > _step10_fileIO.py
  > emp.csv
  > step01_str_tensorflow.py
  > step02_constant_variable.py
  > step03_arimatic_function.py
  > step04_tensorboard.py
  > lec02_fetch_feed
  > lec03_tensor_handling
  > lec04_regression
  > lec04_transform
  > lec05_Kmeans
  > C:\Anaconda3\python.exe
  > Tensorflow_packt
    > chap01
      > _init_.py
      > first_session_only_tensorflow.p
      > first_session.py
      > programming_model.py
      > using_tensorboard.py
    > chap02
    > chap03
      > _init_.py
      > Kmeans.py
      > KNN.py
      > linear_regression.py
      > Mnist.py
    > chap04

1 #first_session.py
2
3 #a simple Python code
4 x = 1
5 y = x + 9
6 print(y) # 10
7
8 # TensorFlow에 값 만 정의된 상태
9 import tensorflow as tf
10
11 x = tf.constant(1, name='x')
12 y = tf.Variable(x+9, name='y')
13 print(y) # Tensor("y/read:0", shape=(), dtype=int32)
14

Console
<terminated> first_session.py [C:\Anaconda3\python.exe]
E c:\tf_jenkins\home\workspace\release-win\device\cpu\os\windows\tensorflow\cor
E c:\tf_jenkins\home\workspace\release-win\device\cpu\os\windows\tensorflow\cor
E c:\tf_jenkins\home\workspace\release-win\device\cpu\os\windows\tensorflow\cor
E c:\tf_jenkins\home\workspace\release-win\device\cpu\os\windows\tensorflow\cor
Tensor("y/read:0", shape=(), dtype=int32)
```

● Session 생성 : y변수 값 초기화, 연산

The screenshot shows the Eclipse IDE interface with the following components:

- PyDev Package Explorer (Left):** Displays a project structure for 'Tensorflow_packt'. The 'chap01' folder is expanded, showing files like '_init_.py', 'first_session_only_tensorflow.p', 'first_session.py', 'programming_model.py', and 'using_tensorboard.py'. 'first_session.py' is selected.
- Editor (Center):** Displays the code in 'first_session.py'. The code includes comments in Korean and Python syntax for creating a TensorFlow session and initializing a variable.

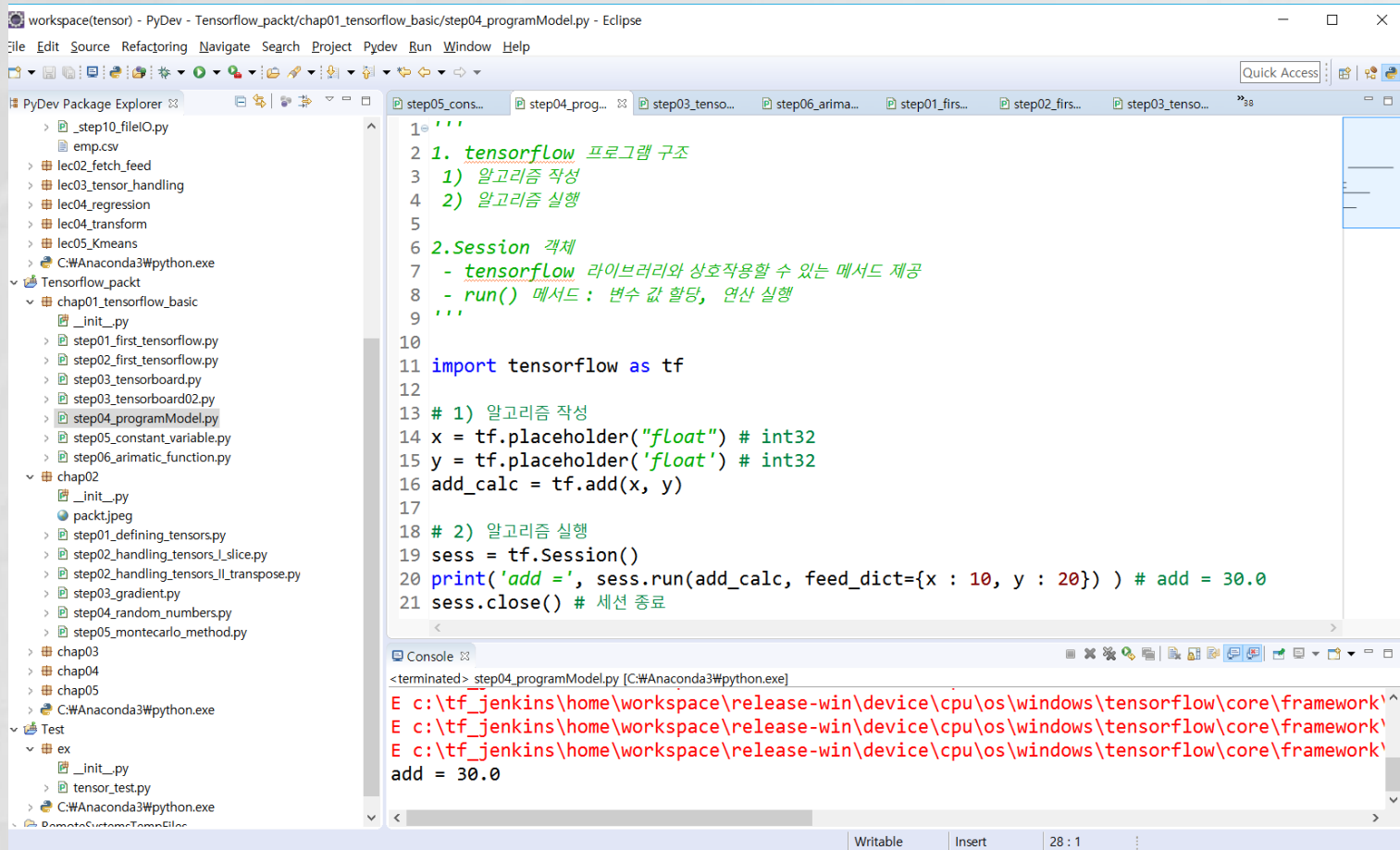
```
3 #a simple Python code
4 x = 1
5 y = x + 9
6 print(y) # 10
7
8 # Tensorflow에 값 만 정의된 상태
9 import tensorflow as tf
10
11 x = tf.constant(1, name='x')
12 y = tf.Variable(x+9, name='y')
13 #print(y) # Tensor("y/read:0", shape=(), dtype=int32)
14 init = tf.global_variables_initializer()
15
16 # Session 생성, 값 초기화, y변수 연산
17 sess = tf.Session()
18 sess.run(init)
19 print('y=', sess.run(y))
20
```

Lines 16-19 are highlighted with a red box. Line 12 contains a syntax error: `y = tf.Variable(x+9, name='y')` should be `y = tf.Variable(x+9, dtype=tf.int32, name='y')`.
- Console (Bottom):** Shows the execution output for 'first_session.py'. It displays two error messages (likely from a previous run) and the final output `y= 10`, which is also highlighted with a red box.

```
<terminated> first_session.py [C:\Anaconda3\python.exe]
E c:\tf_jenkins\home\workspace\release-win\device\cpu\os\windows\tensorflow\cor
E c:\tf_jenkins\home\workspace\release-win\device\cpu\os\windows\tensorflow\cor
y= 10
```

2. Tensorflow 프로그래밍 모델

◦ 알고리즘 작성과 알고리즘 실행 영역 분리



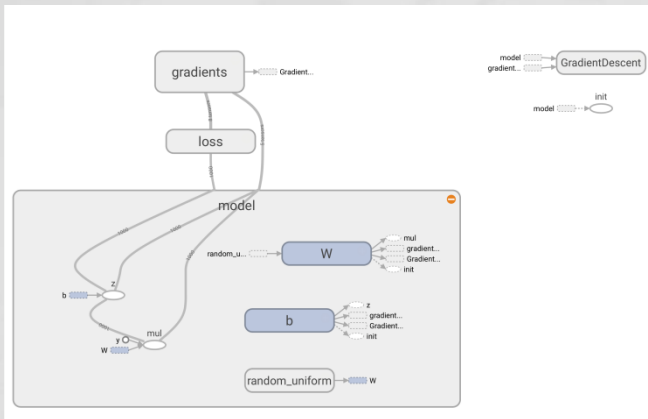
```
workspace(tensor) - PyDev - Tensorflow_packt/chap01_tensorflow_basic/step04_programModel.py - Eclipse
File Edit Source Refactoring Navigate Search Project Pydev Run Window Help
PyDev Package Explorer
> _step10_fileIO.py
  emp.csv
> lec02_fetch_feed
  lec03_tensor_handling
  lec04_regression
  lec04_transform
  lec05_Kmeans
  C:\Anaconda3\python.exe
Tensorflow_packt
  chap01_tensorflow_basic
    _init_.py
    step01_first_tensorflow.py
    step02_first_tensorflow.py
    step03_tensorboard.py
    step03_tensorboard02.py
    step04_programModel.py
    step05_constant_variable.py
    step06_arimatic_function.py
  chap02
    _init_.py
    packt.jpeg
    step01_defining_tensors.py
    step02_handling_tensors_slice.py
    step02_handling_tensors_ll_transpose.py
    step03_gradient.py
    step04_random_numbers.py
    step05_montecarlo_method.py
  chap03
  chap04
  chap05
  C:\Anaconda3\python.exe
Test
  ex
    _init_.py
    tensor_test.py
  C:\Anaconda3\python.exe
  RemoteSystemTensorFlow

1= '''
2 1. tensorflow 프로그램 구조
3 1) 알고리즘 작성
4 2) 알고리즘 실행
5
6 2.Session 객체
7 - tensorflow 라이브러리와 상호작용할 수 있는 메서드 제공
8 - run() 메서드: 변수 값 할당, 연산 실행
9 '''
10
11 import tensorflow as tf
12
13 # 1) 알고리즘 작성
14 x = tf.placeholder("float") # int32
15 y = tf.placeholder('float') # int32
16 add_calc = tf.add(x, y)
17
18 # 2) 알고리즘 실행
19 sess = tf.Session()
20 print('add =', sess.run(add_calc, feed_dict={x : 10, y : 20})) # add = 30.0
21 sess.close() # 세션 종료

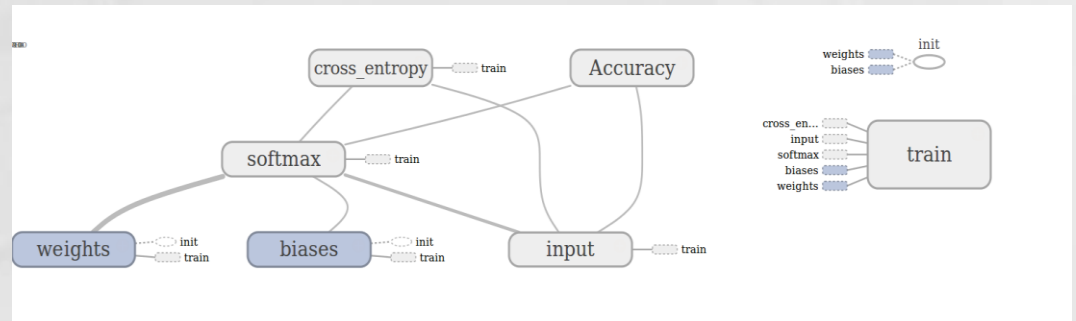
Console
<terminated> _step04_programModel.py [C:\Anaconda3\python.exe]
E c:\tf_jenkins\home\workspace\release-win\device\cpu\os\windows\tensorflow\core\framework\
E c:\tf_jenkins\home\workspace\release-win\device\cpu\os\windows\tensorflow\core\framework\
E c:\tf_jenkins\home\workspace\release-win\device\cpu\os\windows\tensorflow\core\framework\
add = 30.0
```


3. Tensorboard

- Data flow Graph 분석을 위한 시각화 제공
- Node의 계층적 분류와 상세화 제공
- Interactive 시각화 기능 제공
- 특정 부분 확대 및 축소 → 변수의 통계정보 제공

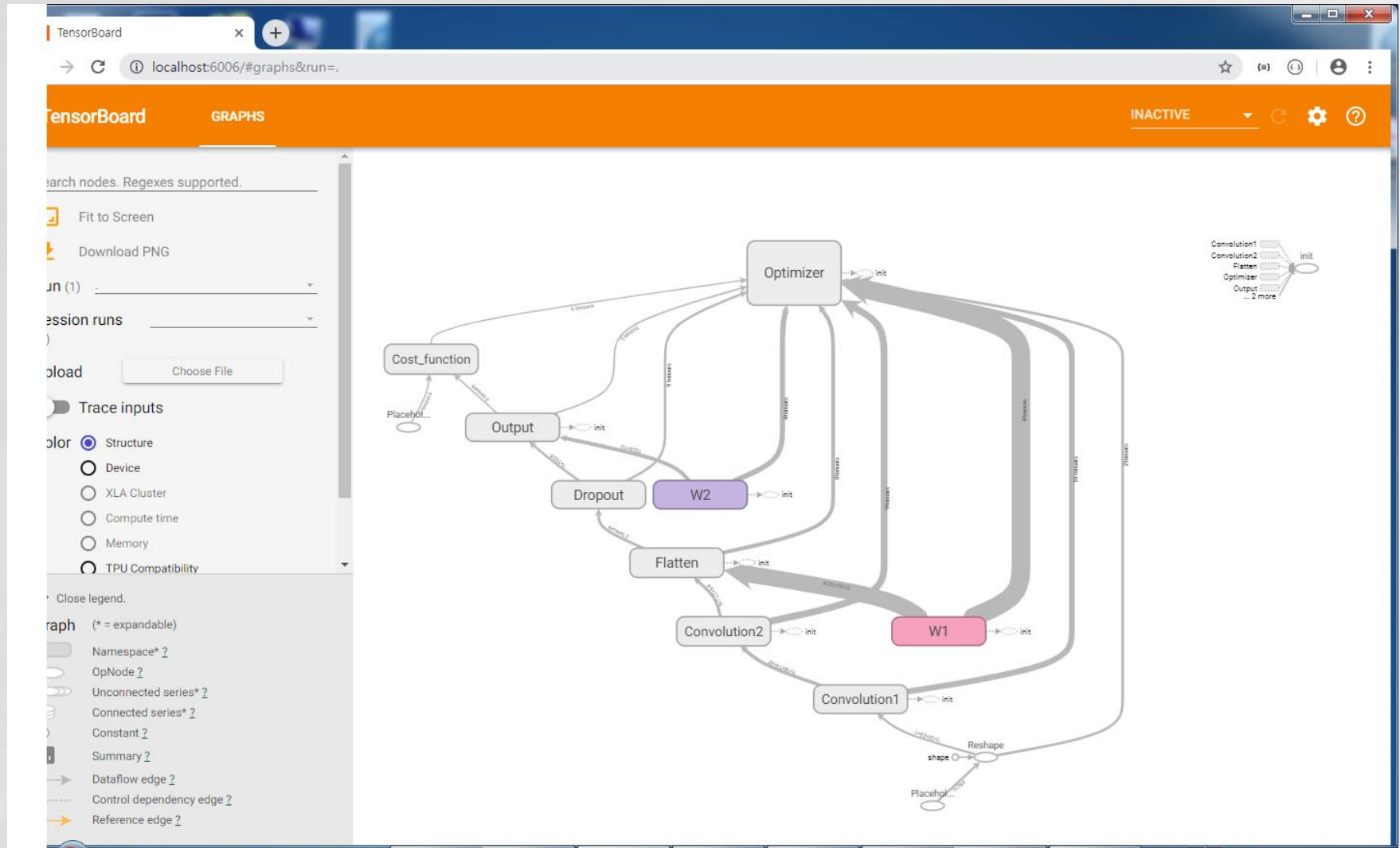


Regression Analysis



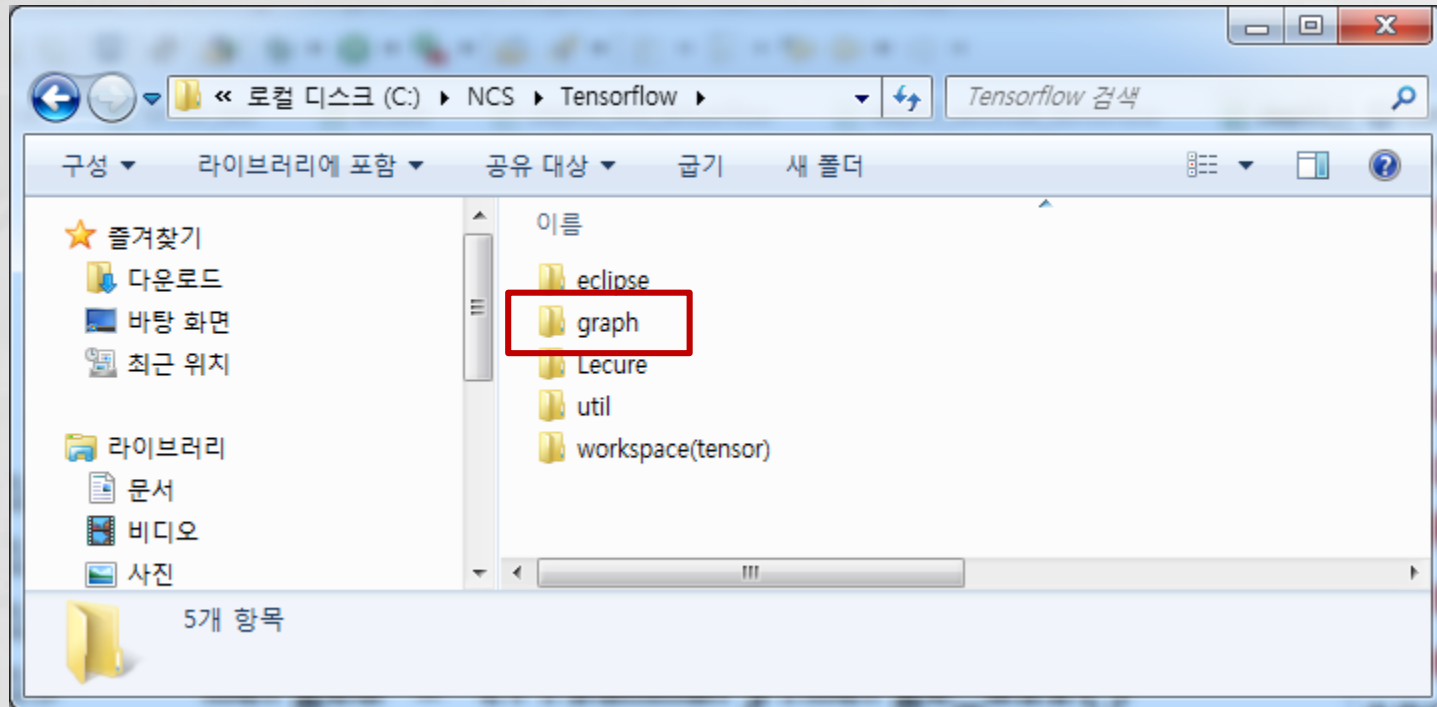
Artificial Neural Network

● CNN Model layer



● Graph 폴더 생성

✓ 주의 : Graph만 독립적으로 생성되는 폴더 생성



● Tensorboard 사용

```
workspace(tensor) - PyDev - chapter01_tensorflow_basic/lecture/step03_tensorboard.py - Eclipse
File Edit Source Refactoring Navigate Search Project Pydev Run Window Help

ex tensor_test lecture step01_first_tensorflow step02_seconde_tensorflow step03_tensorboard step04_programModel step03_tensorboard02

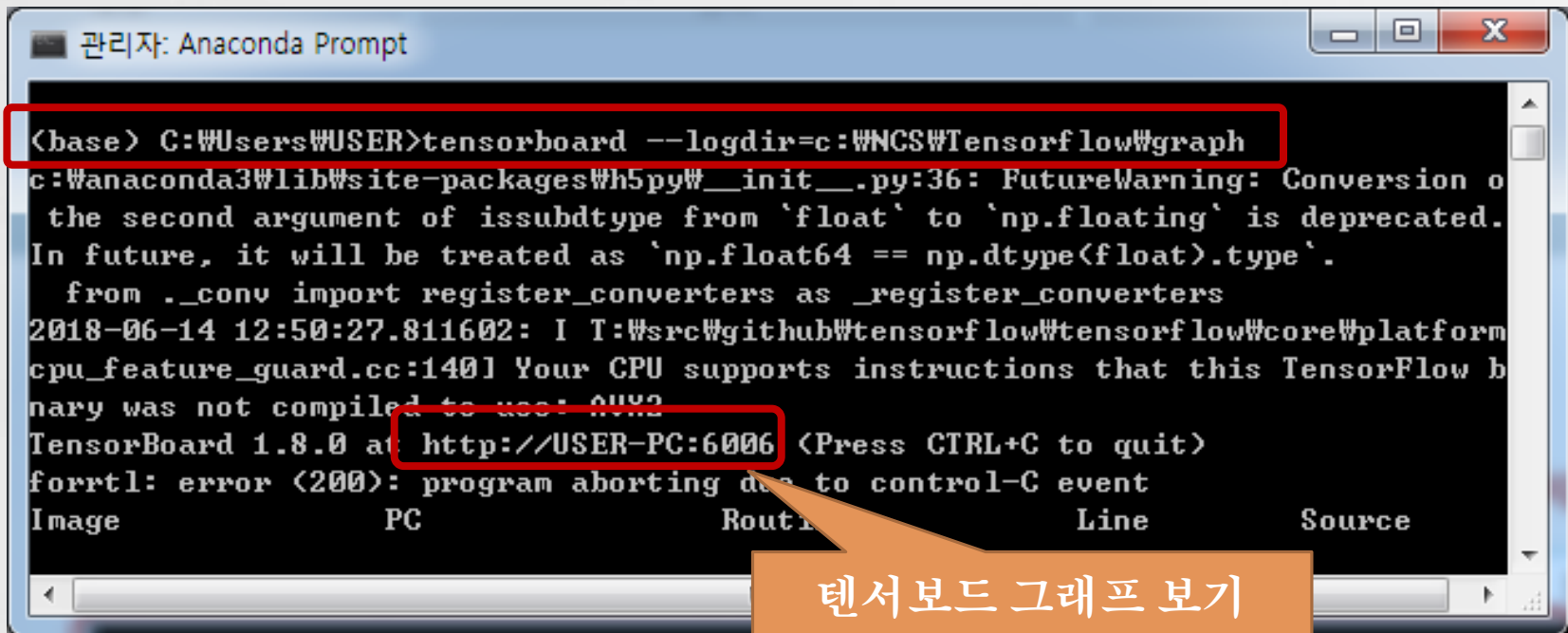
5 a = tf.constant(10,name="a")
6 b = tf.constant(90,name="b")
7 y = tf.Variable(a+b*2,name='y') # a + (b*2)
8
9 model = tf.initialize_all_variables()
10
11 # sess = tf.Session()
12 with tf.Session() as sess: # sess.close() 생략
13     merged = tf.summary.merge_all()
14     writer = tf.summary.FileWriter(r"C:\NCS\Tensorflow\graph", sess.graph)
15     sess.run(model)
16     print(sess.run(y)) # 190
17
18 # cmd 실행 코드
19 #C:\Anaconda3\Scripts>tensorboard --logdir=c:\ncs\tensorflow
20 # Starting TenforBoard b'41' on port 6006
21
22
```

텐서보드 데이터 플로 그래프 생성

Writable Insert 16 : 29

- Tensorboard 로그파일 실행

Anaconda3 Prompt> tensorboard --logdir=c:\ncs\tensorflow\graph



```
(base) C:\Users\USER>tensorboard --logdir=c:\ncs\tensorflow\graph
c:\anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of
the second argument of issubdtype from 'float' to 'np.floating' is deprecated.
In future, it will be treated as 'np.float64 == np.dtype(float).type'.
  from ._conv import register_converters as _register_converters
2018-06-14 12:50:27.811602: I T:\src\github\tensorflow\tensorflow\core\platform
cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow b
nary was not compiled to use: AVX2
TensorBoard 1.8.0 at http://USER-PC:6006 (Press CTRL+C to quit)
forrtl: error (200): program aborting due to control-C event
Image                PC                Routi                Line                Source
```

텐서보드 그래프 보기



● Tensorboard2 사용

workspace(tensor) - PyDev - Tensorflow/lab1_basic/step02_tensorboard_test.py - Eclipse

File Edit Source Refactoring Navigate Search Project Pydev Run Window Help

PyDev Package Explorer

- Tensorflow
 - lab1_basic
 - __init__.py
 - step01_tensorflow_test.py
 - step02_tensorboard_test.py
 - lab10_mnist_NN
 - lab11_mnist_CNN
 - lab12_RNN
 - lab2_linear_regression
 - __init__.py
 - lab-02-1-linear_regression
 - lab-02-2-linear_regression
 - lab-02-3-linear_regression
 - lab3_minimizing_cost
 - lab4_multi_variable
 - lab5_logistic_regression
 - lab6_softmax
 - lab7_mnist_introduction
 - lab8_tensor_manipulation
 - lab9_xor_NN
- C:\Anaconda3\python.exe
- RemoteSystemsTempFiles

lab-02-1-linear_regression

```
1
2 import tensorflow as tf
3
4 a = tf.add(1, 2)
5 b = tf.multiply(a, 3) #tf.mul(a, 3)
6 c = tf.add(4, 5)
7 d = tf.multiply(c, 6)
8 e = tf.multiply(4, 5)
9 f = tf.div(c, 6)
10 g = tf.add(b, d)
11 h = tf.multiply(g, f)
12
13 with tf.Session() as sess :
14     merged = tf.summary.merge_all()
15     writer = tf.summary.FileWriter(r"C:\NCS\Tensorflow", sess.graph)
16     #sess.run(h)
17     print('h =', sess.run(h)) # h = 63
18     writer.close()
```

텐서보드 데이터플로우 그래프 생성

Console

```
<terminated> step02_tensorboard_test.py [C:\Anaconda3\python.exe]
h = 63
E c:\tf_jenkins\home\workspace\release-win\device\cpu\os\windows\tensorflow\cor
E c:\tf_jenkins\home\workspace\release-win\device\cpu\os\windows\tensorflow\cor
E c:\tf_jenkins\home\workspace\release-win\device\cpu\os\windows\tensorflow\cor
E c:\tf_jenkins\home\workspace\release-win\device\cpu\os\windows\tensorflow\cor
```

Writable Insert 18 : 1

● Tensorboard2 그래프 보기

The image shows two overlapping windows of the TensorBoard2 web application. The background window is on the 'SCALARS' tab, and the foreground window is on the 'GRAPHS' tab.

TensorBoard2 SCALARS Tab (Background Window):

- Address bar: 192.168.0.26:6006
- Navigation bar: TensorBoard, SCALARS, IMAGES, AUDIO, GRAPHS, DISTRIBUTIONS, HISTOGRAMS, EMBEDDINGS
- Left sidebar:
 - Write a regex to create a tag group
 - ☐ Split on underscores
 - ☐ Data download links
 - Tooltip sorting method: default
 - Smoothing: 0.6
 - Horizontal Axis: STEP, RELATIVE, WALL
 - Runs: TOGGLE ALL RUNS
 - c:\NCS\Tensorflow

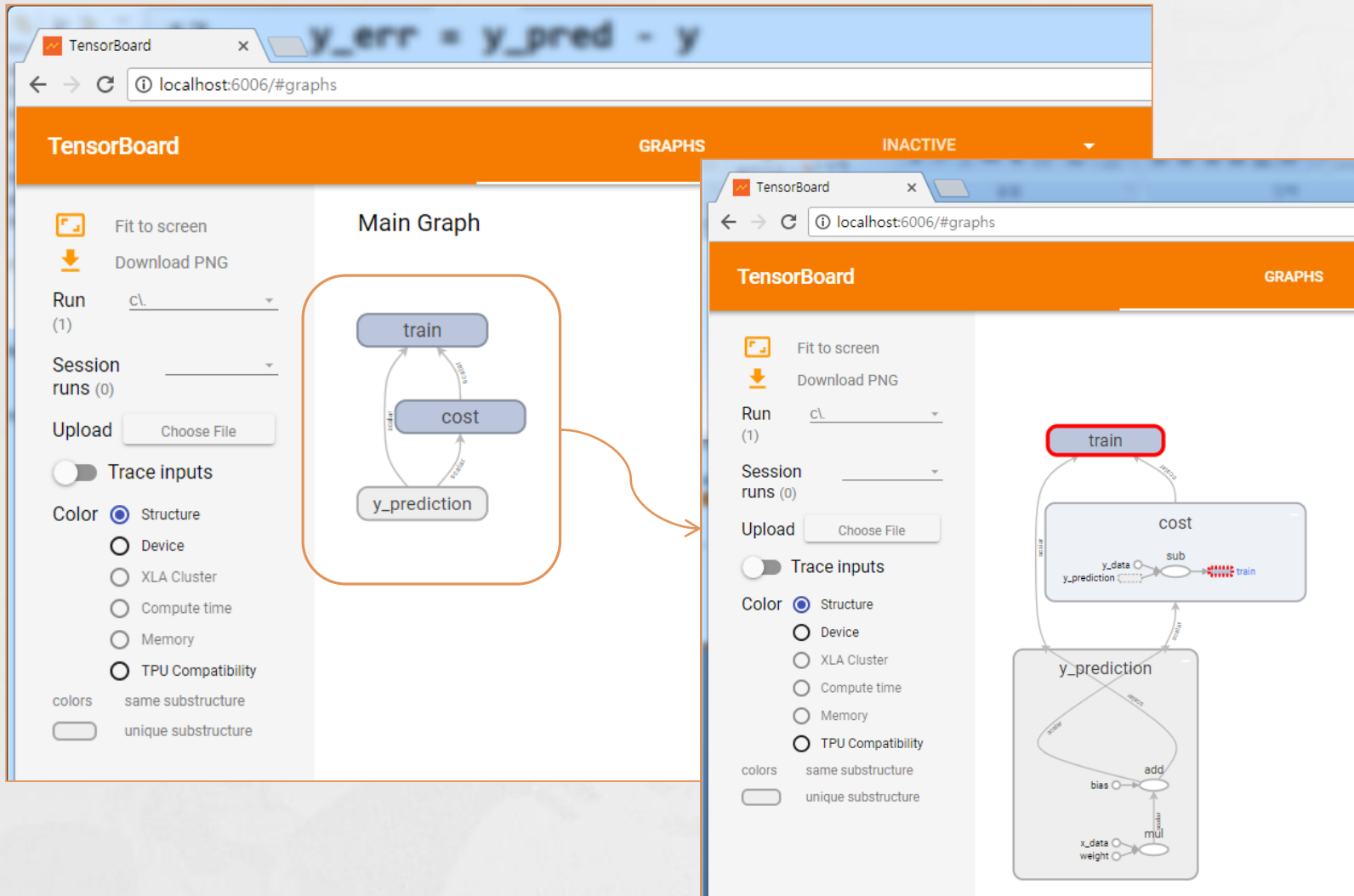
TensorBoard2 GRAPHS Tab (Foreground Window):

- Address bar: 192.168.0.26:6006
- Navigation bar: TensorBoard, SCALARS, IMAGES, AUDIO, **GRAPHS**, DISTRIBUTIONS, HISTOGRAMS, EMBEDDINGS
- Left sidebar:
 - Fit to screen
 - Download PNG
 - Run: cl.
 - Session runs: (0)
 - Upload: Choose File
 - Trace inputs: ☐
 - Color Graph:
 - ☒ Structure (* = expandable)
 - ☐ Device
 - Namespace*
 - same substructure
 - OpNode
 - unique substructure*
 - unconnected series*
 - Connected series*
 - Constant
 - Summary
 - Legend:
 - Dataflow edge
 - - - Control dependency edge
 - ← Reference edge

Graph Visualization:

The graph shows a computational flow with nodes and edges. Nodes include Mul_3, Add_2, Mul, Add, Mul_1, Add_1, div, and Mul_2. Edges represent dataflow, control dependency, and reference. The graph is a directed acyclic graph (DAG) showing the execution flow of operations.

● Tensorboard3(name_scope 이용) 그래프 보기



4. 변수 값 할당 방법

- ◉ Session의 변수 처리 : Fetch와 Feeds 방법
- ◉ Fetch 방법
 - 연산결과를 가져오는 방법
- ◉ Feed 방법
 - Placeholder에 값을 공급(feed)하여 실행하는 방법

❖ 변수 선언을 위한 tensorflow 코드 실행

```
import tensorflow as tf
```

```
# tf.placeholder: 계산을 실행할 때 입력값을 받는 변수로 사용
```

```
# None 은 크기가 정해지지 않았음을 의미합니다.
```

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

```
print(X)
```

```
# X 플레이스홀더에 넣을 값
```

```
# 플레이스홀더에서 설정한 것 처럼, 두번째 차원의 요소의 갯수는 3개
```

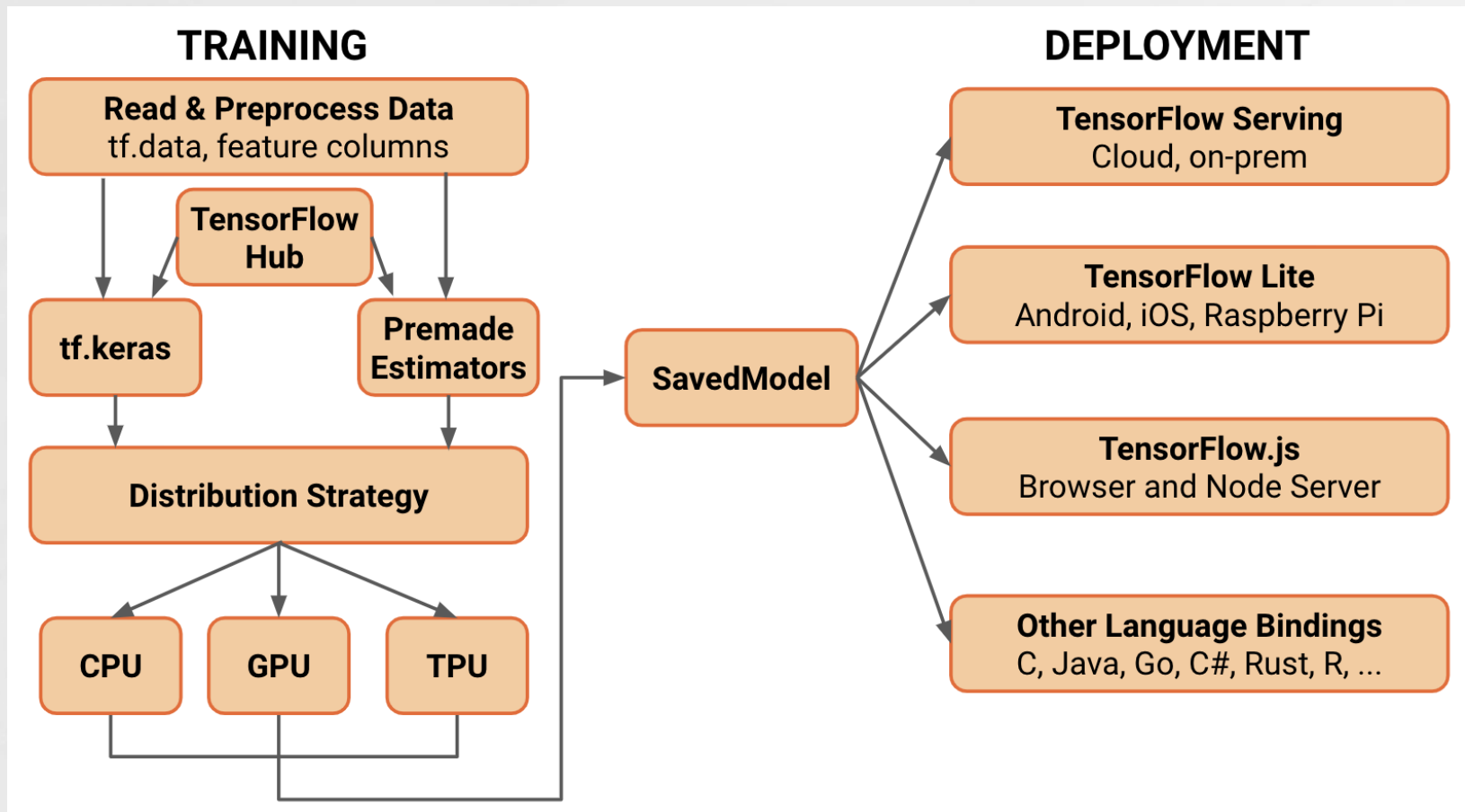
```
x_data = [[1, 2, 3]]
```

5. Tensorflow2.x 주요 특징

- 단순성과 편의성에 초점을 두고 업그레이드
- 즉시 실행(eager execution)
- Keras와 즉시 실행(eager execution) 쉬운 모델 작성
- 플랫폼에 독립적인 탄탄한(robust) 모델 배포
- 연구를 위한 강력한 실험법
- 중복된 API 정리 및 단순화
- Tensorflow 1.0 기능 제공
- tf.function 기능 : Python 문법으로 코드 작성
([TF2.0:Function, not Session](#) 참고)

참고 : <https://github.com/tensorflow/community/blob/master/rfcs/20180918-functions-not-sessions-20.md>

Tensorflow2.0 Architecture



1. **tf.data** 이용하여 데이터 불러오기

트레이닝 데이터는 tf.data로 작성된 입력 파이프라인으로 읽음
tf.feature_column 이용 feature 특징 읽음

2. **tf.keras**로 모델 작성, 학습, 검증(validate)하거나, **Premade Estimator** 사용

Keras는 텐서플로우의 나머지 부분과 긴밀히 통합, 접근 용이
텐서플로우의 허브 모듈을 이용한 Transfer learning으로 Keras나 Estimator
모델을 학습시킬 수 있음

3. **eager execution**으로 실행, **@tf.function** 사용

텐서플로우2.0은 쉬운 사용과 원활한 디버깅을 위해 기본적으로 eager
execution으로 실행 tf.function은 Python으로 Tensorflow 그래프 변환
Tensorflow1.x의 그래프 기반 실행의 장점(성능 최적화, 원격 실행 및
쉬운 직렬화, 내보내기, 배포 기능 등) 모두 유지

4. 분산 학습을 위해 **Distribution Strategy**를 사용

규모가 큰 머신러닝 학습을 위해 Distribution Strategy API 제공
모델의 정의를 바꾸지 않으면서 다른 하드웨어 구성에 쉽게 배포하고
학습시킬 수 있다. CPU, GPU, TPU와 같은 다양한 하드웨어 가속기 지원

5. **SavedModel** 내보내기

SavedModel을 TensorFlow Serving, TensorFlow Lite, TensorFlow.js,
TensorFlow Hub 등에서의 표준 교환 포맷으로 사용

tf.Variable()

Tensorflow ver1.x

```
import tensorflow.compat.v1 as tf # ver 1.x
tf.disable_v2_behavior() # ver 2.x 사용안함
```

```
x = tf.constant(1, name = 'x') # 상수 정의
y = tf.Variable(x + 9, name = 'y') # 변수 정의
```

```
# y변수 초기화 객체 생성
init = tf.global_variables_initializer()
```

```
''' 실행 영역 : Session 객체 '''
sess = tf.Session() # object
```

```
# 변수 실행 -> 초기값 할당
sess.run(init) # 모든 변수(y, z) 초기화
```

```
print('y=', sess.run(y)) # y= 10
```

Tensorflow ver2.x

```
import tensorflow as tf # ver 2.x import
```

```
# Python code 형식 지원
x = tf.constant(1.0) # x = 1
y = tf.Variable(x + 9.0) # 변수 정의
print('y=', y) # 변수 정보
# <tf.Variable 'Variable:0' shape=()
dtype=float32, numpy=10.0>
```

```
# 함수 장식자 적용
@tf.function # Python code 지원
def f():
    return y.read_value()
```

```
y_val = f()
print('y=', y_val) # v= tf.Tensor(10.0,
shape=(), dtype=float32)
```

tf.placeholder()

Tensorflow ver1.x

```
import tensorflow.compat.v1 as tf # ver 1.x
tf.disable_v2_behavior() # ver 2.x 사용안함

# 공급형 변수 정의
x = tf.placeholder(tf.float32)

# 식 정의
y = tf.square(x) # x참조
z = tf.add(x, y) # x,y 참조

sess = tf.Session ()

zo = sess.run(z, feed_dict = {x : 2. }) # x=2, y=4 # 6.0
print('zo=', zo)
z1 = sess.run (z, feed_dict = {x : 2 , y : 2. }) # 4.0
print('z1=', z1)
```

Tensorflow ver2.x

```
import tensorflow as tf # ver 2.x import

# 공급형 변수 : Python 함수의 인수로 받음
#x = tf.placeholder(tf.float32) # AttributeError:

# 함수장식자 + Python 함수(식 정의)
@tf.function
def calc_z1(x, y): # 인수 2개
    return tf.add(x, y)

@tf.function
def calc_z0(x): # 인수 1개
    return calc_z1(x, tf.square(x))

zo = calc_z0(2.)
print(zo) # tf.Tensor(6.0, shape=(), dtype=float32)
z1 = calc_z1(2., 2.)
print(z1) # tf.Tensor(4.0, shape=(), dtype=float32)
```


식 실행과 변수값 할당

Tensorflow ver1.x

```
import tensorflow.compat.v1 as tf # ver 1.x
tf.disable_v2_behavior() # ver 2.x 사용안함

x_data = tf.random_normal((1, 10))
x = tf.placeholder(tf.float32)
W = tf.Variable(tf.glorot_uniform_initializer()( (10, 10)))
b = tf.Variable(tf.zeros(10))
y_pred = tf.matmul(x, W) + b

c = tf.Variable(0.5)
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init) # 변수 초기화
    feed_data = {x : sess.run(x_data)} # data 공급
    print(sess.run(y_pred, feed_dict = feed_data))
    # 변수 값 할당
    sess.run(c.assign_add(10)) # 0.5+10 -> 10.5
    print('c=', sess.run(c))
```

Tensorflow ver2.x

```
import tensorflow as tf # ver 2.x import

W=tf.Variable(tf.random_uniform_initializer)((10,10)))
b = tf.Variable(tf.zeros(10))
c = tf.Variable(0.5)

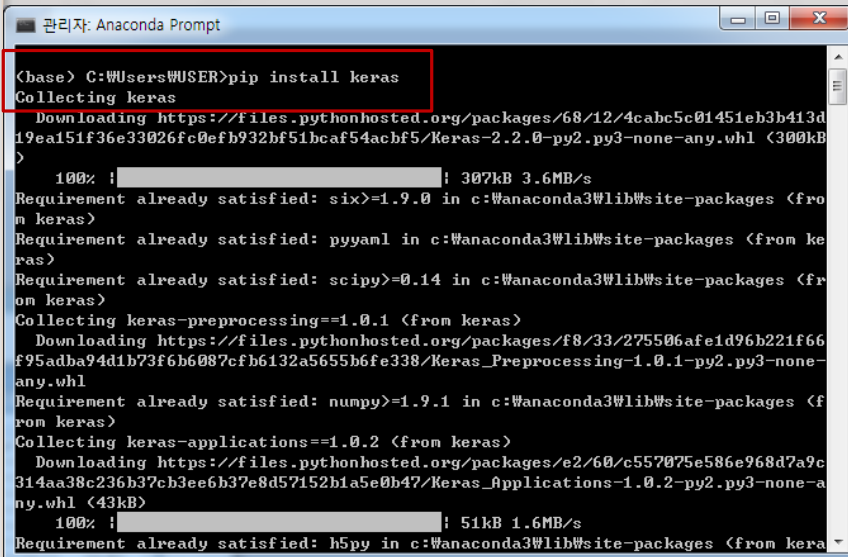
@tf.function
def f(x):
    c.assign_add(10) # 초기값 변경
    y_pred = tf.matmul(x, W) + b # 행렬곱 연산 함수
    return y_pred, c.read_value()

# x변수 생성 + 초기화
x=tf.Variable(tf.random_uniform_initializer)((1, 10)))
# 함수 호출
y_pred, c_val = f(x)
print(y_pred) # Tensorflow 2.x 출력
# 변수값 할당
print('c=',c_val)
```

Keras 호출 -> 통합

Tensorflow ver1.x

```
from keras.datasets import mnist # MNIST dataset
from keras.models import Sequential # model 생성
from keras.layers.core import Dense, Dropout, Activation
from keras.losses import categorical_crossentropy
```



```
관리자: Anaconda Prompt
(base) C:\Users\WUSER>pip install keras
Collecting keras
  Downloading https://files.pythonhosted.org/packages/68/12/4cabc5c01451eb3b413d19ea151f36e33026fc0efb932bf51bc5454achf5/Keras-2.2.0-py2.py3-none-any.whl (300kB)
    100% |#####| 307kB 3.6MB/s
Requirement already satisfied: six>=1.9.0 in c:\w\anaconda3\lib\site-packages (from keras)
Requirement already satisfied: pyyaml in c:\w\anaconda3\lib\site-packages (from keras)
Requirement already satisfied: scipy>=0.14 in c:\w\anaconda3\lib\site-packages (from keras)
Collecting keras-preprocessing==1.0.1 (from keras)
  Downloading https://files.pythonhosted.org/packages/f8/33/275506afe1d96b221f66f95adba94d1b73f6b6087cfb6132a5655b6fe338/Keras_Preprocessing-1.0.1-py2.py3-none-any.whl
Requirement already satisfied: numpy>=1.9.1 in c:\w\anaconda3\lib\site-packages (from keras)
Collecting keras-applications==1.0.2 (from keras)
  Downloading https://files.pythonhosted.org/packages/e2/60/c557075e586e968d7a9c314aa38c236b37cb3ee6b37e8d57152b1a5e0b47/Keras_Applications-1.0.2-py2.py3-none-any.whl (43kB)
    100% |#####| 51kB 1.6MB/s
Requirement already satisfied: h5py in c:\w\anaconda3\lib\site-packages (from keras)
```

Tensorflow ver2.x

```
from tensorflow.keras.datasets.mnist import load_data
from tensorflow.keras import Sequential # model object
from tensorflow.keras.layers import Dense # DNN layer
```

```
(x_train, y_train), (x_test, y_test) = load_data()
```