

# Tutorials

Anaconda / Jupyter notebook / Tensorflow

# Contents

- Anaconda
  - Install virtual environment (Tensorflow, Jupyter notebook, matplotlib)
- Jupyter notebook tutorial
  - .ipynb files and python
  - Anaconda virtual environment and kernel
- Tensorflow tutorials
  - Model and Session

# Anaconda

- Anaconda 란?
  - 통계관련 프로그램인 Python 및 R 을 보다 편리하게 사용할 수 있도록 도와주는 프로그램
  - Conda 를 통해 다양한 언어들의 package, dependency, environment 를 관리할 수 있다.
- 수업에서의 사용 방향:
  - 가상환경 만들기
  - 수업에 필요한 프로그램을 명령어만으로 설치 (jupyter notebook, tensorflow, matplotlib 등)
  - Jupyternotebook 을 실행하여 실습 수업 진행

# Anaconda

The screenshot shows a web browser window with the URL <https://www.anaconda.com/distribution/#download-section>. The page is titled "Anaconda 2019.03 for Linux Installer" and features two main download options:

- Python 3.7 version**: Includes a green "Download" button and links for "64-Bit (x86) Installer (654 MB)" and "64-Bit (Power8 and Power9) Installer (315 MB)".
- Python 2.7 version**: Includes a green "Download" button and links for "64-Bit (x86) Installer (630 MB)" and "64-Bit (Power8 and Power9) Installer (291 MB)".

Below the download section, there is a "Get Started with Anaconda Distribution" section with five links:

- Documentation**: Installation and user guide for Anaconda Distribution 5
- Anaconda Blog**: News, software releases, and developer best practices
- Community Support**: Solutions and knowledge from the community
- Anaconda Webinars**: Industry trends and tutorials from Anaconda
- Anaconda Training**: Learn Python for Data Science with DataCamp

# Anaconda

- 설치 후에...
- Ubuntu: terminal 실행
- Window: Anaconda prompt 실행
- 가상환경 설치 명령어 입력
  - `conda create -n [가상환경이름] [설치할 패키지들]`
  - `conda create -n tensorflow tensorflow jupyter notebook matplotlib`
- 가상환경 삭제 명령어
  - `conda remove -n [가상환경이름] --all`

# Anaconda

- 설치된 가상환경 확인 명령어
  - `conda env list`
- 설치된 가상환경 실행 명령어
  - `conda activate [가상환경이름]`
- 쥬피터 노트북 커널 설정 명령어
  - `Conda install notebook ipykernel`
  - `Python -m ipykernel install -user -name tensorflow --display-name "Tensorflow"`

# Jupyter notebook tutorial

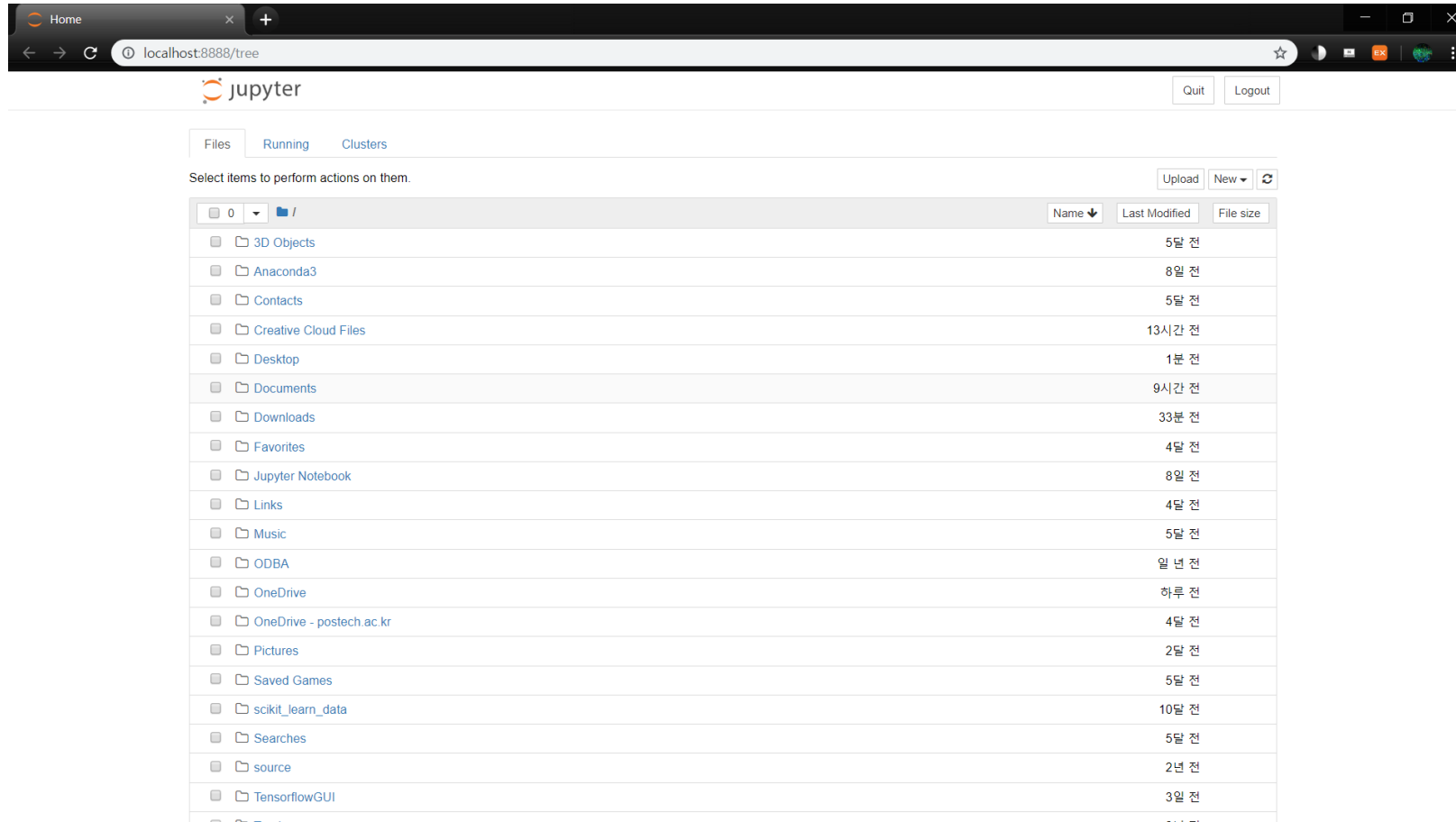
- Jupyter notebook 이란?
  - 대화형 프로그래밍을 지원하는 웹 어플리케이션
  - Jupyter 를 통해 웹에서 프로그래밍을 실시간으로 할 수 있다.
- 수업에서의 사용 방향:
  - 웹(인터넷) 브라우저에서 대화형 프로그래밍을 진행
    - 중간중간 결과 확인 및 검색을 쉽게 진행 가능

# Jupyter notebook tutorial

- Anaconda 에서 Jupyter notebook 을 설치하였다면...  
( jupyter notebook 이 설치된 가상환경을 실행하였다면)
- Jupyter notebook 실행 명령어
  - Jupyter notebook
  - 인터넷 창이 켜지며, *명령어를 입력했던 폴더*의 내용을 보여줌



# Jupyter notebook tutorial



# Jupyter notebook tutorial

- .ipynb
  - 우리가 실습할 코드의 확장자 (ipython notebook 의 약자)
  - python 코드를 실시간으로 실행시켜가며 확인할 수 있게 해준다.
- Kernel 설정
  - Anaconda environment 설정을 jupyter notebook 내부에서도 설정 가능
  - 가끔 실행이 안될 시, jupyter notebook 을 적절한 conda environment 에서 실행한 것이 아니므로, 재시작 혹은 kernel 설정 필요

# Tensorflow tutorials

- What is Tensorflow?
  - Open source software library for numerical computation using data-flow graphs.
- How can we write a Tensorflow program?
  - Building the computational graph
  - Running the computational graph

# Tensorflow tutorials

- Computational graph?
  - A series of Tensorflow operations arranged into a graph. The graph is composed of two types of objects.
- Tensors?
  - The edges in the graph, represent the values that will flow through the graph.
- Operations?
  - The nodes of the graph, describe calculations that consume and produce tensors.

# Components in Tensorflow

- Constant
  - A non-mutable variable.
- Placeholder
  - A tensor taking external inputs(input data).
- Variable
  - A mutable variable(parameter).

# Building computational graph(Constant)

```
import tensorflow as tf

a = tf.constant(3.0, dtype=tf.float32)
b = tf.constant(4.0) # also tf.float32 implicitly
total = a + b
print(a)
print(b)
print(total)
```

```
Tensor("Const:0", shape=(), dtype=float32)
Tensor("Const_1:0", shape=(), dtype=float32)
Tensor("add:0", shape=(), dtype=float32)
```

```
import tensorflow as tf

a = tf.constant([3.0, 4.0], dtype=tf.float32)
print(a)
```

```
Tensor("Const:0", shape=(2, ), dtype=float32)
```

# Running computational graph(Constant)

```
import tensorflow as tf
```

```
a = tf.constant(3.0, dtype=tf.float32)
b = tf.constant(4.0) # also tf.float32 implicitly
total = a + b
```

```
sess = tf.Session()
print(sess.run(total))
print(sess.run([a, b]))
sess.close()
```

```
7.0
[3.0 4.0]
```

```
import tensorflow as tf
```

```
a = tf.constant([3.0, 4.0], dtype=tf.float32)
sess = tf.Session()
print(sess.run(a))
sess.close()
```

```
[3.0 4.0]
```

# Building computational graph(Placeholder)

```
import tensorflow as tf
```

```
x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)
z = x + y
print(x)
print(y)
print(z)
```

```
Tensor("Placeholder:0", dtype=float32)
Tensor("Placeholder_1:0", dtype=float32)
Tensor("add:0", dtype=float32)
```

```
import tensorflow as tf
```

```
x1 = tf.placeholder(tf.float32, shape=[2])
y1 = tf.placeholder(tf.float32, shape=[2])
z1 = x1 + y1
print(x)
print(y)
print(z)
```

```
Tensor("Placeholder:0", shape=(2,), dtype=float32)
Tensor("Placeholder_1:0", shape=(2,), dtype=float32)
Tensor("add:0", shape=(2,), dtype=float32)
```



# Running Computational graph(Placeholder)

```
import tensorflow as tf
```

```
x = tf.placeholder(tf.float32)  
y = tf.placeholder(tf.float32)  
z = x + y
```

```
sess = tf.Session()  
print(sess.run(z, feed_dict={x: 3, y: 4.5}))  
sess.close()
```

7.5

```
import tensorflow as tf
```

```
x1 = tf.placeholder(tf.float32, shape=[2])  
y1 = tf.placeholder(tf.float32, shape=[2])  
z1 = x1 + y1
```

```
sess = tf.Session()  
print(sess.run(z1, feed_dict={x1: [1, 3], y1: [2, 4]}))  
sess.close()
```

[3. 7.]

# Building computational graph(Variable)

```
import tensorflow as tf

my_variable = tf.Variable("my_variable", [1, 2, 3])
print(my_variable)
```

```
<tf.Variable 'Variable:0' shape=() dtype=string_ref>
```

# Building computational graph(Variable)

```
import tensorflow as tf

my_variable = tf.get_variable("my_variable", [1, 2, 3])
my_int_variable = tf.get_variable("my_int_variable", [1, 2, 3], dtype=tf.int32,
    initializer=tf.zeros_initializer)
print(my_variable)
print(my_int_variable)
```

```
<tf.Variable 'my_variable:0' shape=(1, 2, 3) dtype=float32_ref>
<tf.Variable 'my_int_variable:0' shape=(1, 2, 3) dtype=int32_ref>
```

# Building Computational graph(Variable)

```
import tensorflow as tf
```

```
my_variable = tf.get_variable("my_variable", [1])  
dup = tf.get_variable("my_variable", [1])
```

**ValueError:** Variable my\_variable already exists, disallowed.  
Did you mean to set reuse=True or reuse=tf.AUTO\_REUSE in VarScope?

# Running Computational graph(Variable)

```
import tensorflow as tf  
  
v = tf.get_variable("v", [1, 2, 3], dtype=tf.float32)  
  
sess = tf.Session() sess.run(tf.global_  
variables_initializer()) print(sess.run(v))
```

```
[[[-0.61995244  0.6097648 -0.23456109]  
  [-0.7480066  0.2547065 -0.39186072]]]
```

# Running Computational graph(Variable)

```
import tensorflow as tf

v1 = tf.get_variable("v1", [1, 2, 3], dtype=tf.int32,
                    initializer=tf.zeros_initializer)
v2 = tf.get_variable("v2", [1, 2, 3], dtype=tf.float32,
                    initializer=tf.constant_initializer(1.0))
sess = tf.Session() sess.run(tf.global_variables_initializer())
o1, o2 = sess.run([v1, v2])

print(o1)
print(o2)
```

```
[[[0. 0. 0.]
  [0. 0. 0.]]]
[[[1. 1. 1.]
  [1. 1. 1.]]]
```

# Running Computational graph(Variable)

```
import tensorflow as tf
```

```
v3 = tf.get_variable("v3", [1, 2, 3], dtype=tf.float32,  
                    initializer=tf.random_normal_initializer())  
v4 = tf.get_variable("v4", [1, 2, 3], dtype=tf.float32, ini  
                    tializer=tf.random_uniform_initializer())  
sess = tf.Session() sess.run(tf.global_variables_init  
initializer())      o3, o4 = sess.ru  
n([v3, v4])
```

```
print(o3)
```

```
print(o4)
```

```
[[[-0.42939642  0.6930299  1.1907268 ]  
 [ 1.5753491   0.7039551  -0.33378837]]]  
[[[0.08607972 0.8290528  0.74907434]  
 [0.9640163  0.650851   0.33157504]]]
```

# Running Computational graph(Variable)

```
import tensorflow as tf
```

```
my_variable = tf.get_variable("my_variable", [1, 2, 3])  
my_int_variable = tf.get_variable("my_int_variable", [1, 2, 3], dtype=tf.int32,  
    initializer=tf.zeros_initializer)
```

```
sess = tf.Session()  
sess.run(tf.global_variables_initializer())  
v, w = sess.run([my_variable, my_int_variable])
```

```
print(v)
```

```
print(w)
```

```
sess.close()
```

```
[[[ 0.7382598 -0.4009663 -0.20311093]  
  [-0.666464 -0.4459092  0.7911737 ]]]  
[[[0 0 0]  
  [0 0 0]]]
```



# Running Computational graph(Variable)

```
import tensorflow as tf
```

```
with tf.variable_scope("foo"):  
    my_variable = tf.get_variable("my_variable", [1])  
with tf.variable_scope("foo", reuse=True):  
    dup = tf.get_variable("my_variable", [1])
```

```
print(my_variable)      <tf.Variable 'foo/my_variable:0' shape=(1,) dtype=float32_ref>  
print(dup)              <tf.Variable 'foo/my_variable:0' shape=(1,) dtype=float32_ref>
```

```
with tf.Session() as sess: sess.run(tf.global_variables_initializer())  
    print(sess.run(my_variable))  
    print(sess.run(dup))
```

```
[1.260512]  
[1.260512]
```

# Running Computational graph(Variable)

```
import tensorflow as tf
```

```
with tf.variable_scope("foo"):  
    my_variable = tf.get_variable("my_variable", [1])  
with tf.variable_scope("bar"):  
    dup = tf.get_variable("my_variable", [1])
```

```
print(my_variable)      <tf.Variable 'foo/my_variable:0' shape=(1,) dtype=float32_ref>  
print(dup)              <tf.Variable 'bar/my_variable:0' shape=(1,) dtype=float32_ref>
```

```
with tf.Session() as sess: sess.run(tf.global_variables_initializer())  
    print(sess.run(my_variable))  
    print(sess.run(dup))
```

```
[0.8239702]  
[0.43597305]
```

# Examples

```
import tensorflow as tf
```

```
x = tf.placeholder(tf.float32)
w = tf.get_variable("w",
                    initializer=tf.constant([2.]))
b = tf.constant(4.0)
y = x*w + b
```

```
print(x)
print(w)
print(b)
print(y)
```

```
sess = tf.Session() sess.run(tf.global_
variables_initializer()) print(sess.run(y,
feed_dict={x: 1}))
```

```
Tensor("Placeholder:0", dtype=float32)
<tf.Variable 'w:0' shape=(1,) dtype=float32_ref>
Tensor("Const_1:0", shape=(), dtype=float32)
Tensor("add:0", dtype=float32)
```

```
[6.0]
```

# Examples

```
import tensorflow as tf
```

```
x = tf.placeholder(tf.float32, shape=[1])  
w = tf.get_variable("w", shape=[1],  
                    initializer=tf.zeros_initializer)  
b = tf.constant([4.0])  
y = x*w + b
```

```
print(x)  
print(w)  
print(b)  
print(y)
```

```
sess = tf.Session() sess.run(tf.global_  
variables_initializer()) print(sess.run(y,  
feed_dict={x: [1]}))
```

```
Tensor("Placeholder:0", shape=(1,), dtype=float32)  
<tf.Variable 'w:0' shape=(1,) dtype=float32_ref>  
Tensor("Const:0", shape=(1,), dtype=float32) T  
ensor("add:0", shape=(1,), dtype=float32
```

```
[4.0]
```

# Examples

```
import numpy as np
import tensorflow as tf
```

```
x = tf.placeholder(tf.float32, shape=[None, 2])
w = tf.get_variable("w", shape=[2],
                    initializer=tf.random_normal_initializer())
y = x*w
```

```
print(x)
print(w)
print(y)
```

```
Tensor("Placeholder:0", shape=(?, 2), dtype=float32)
<tf.Variable 'w:0' shape=(2) dtype=float32_ref>
Tensor("add:0", shape=(?, 2), dtype=float32)
```

```
sess = tf.Session() sess.run(tf.global_variables_initializer())
print(sess.run(y, feed_dict={x: np.random.rand(3, 2)}))
```

```
[[0.47765136 0.8566245 ]
 [0.5623522  0.26835   ]
 [0.39336765 0.13821977]]
```

# References

- <https://www.anaconda.com/>
- <https://jupyter.org/>
- <https://www.tensorflow.org/guide/>
- <https://github.com/golbin/TensorFlow-Tutorials>