

# Deep Learning / Machine Learning Seminar



- Installation / env setting -

박성현

# TensorFlow ?

- <https://www.tensorflow.org>

The screenshot shows the TensorFlow website homepage. The header features the TensorFlow logo and navigation links for Install, Develop, API r1.2, Deploy, Extend, Community, Versions, and TFRC. A search bar and a GitHub link are also present. The main banner has a yellow-to-orange gradient background with a wireframe mountain graphic and the text "An open-source software library for Machine Intelligence". A "GET STARTED" button is visible. Below the banner are three large icons: a yellow arrow pointing down, the TensorFlow logo inside an orange hexagon, and the TensorFlow logo inside a neural network diagram. The main content area includes sections for "TensorFlow 1.2 has arrived!", "Introducing TensorFlow Research Cloud", and "The 2017 TensorFlow Dev Summit". Each section contains a brief description and a "UPGRADE NOW", "LEARN MORE", or "WATCH VIDEOS" button. At the bottom, there's an "About TensorFlow" section with a detailed description of what TensorFlow is and a video thumbnail.

An open-source software library  
for Machine Intelligence

GET STARTED

TensorFlow 1.2 has arrived!

We're excited to announce the release of TensorFlow 1.2! Check out the release notes for all the latest.

UPGRADE NOW

Introducing TensorFlow Research Cloud

We're making 1,000 Cloud TPUs available for free to accelerate open machine learning research.

LEARN MORE

The 2017 TensorFlow Dev Summit

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

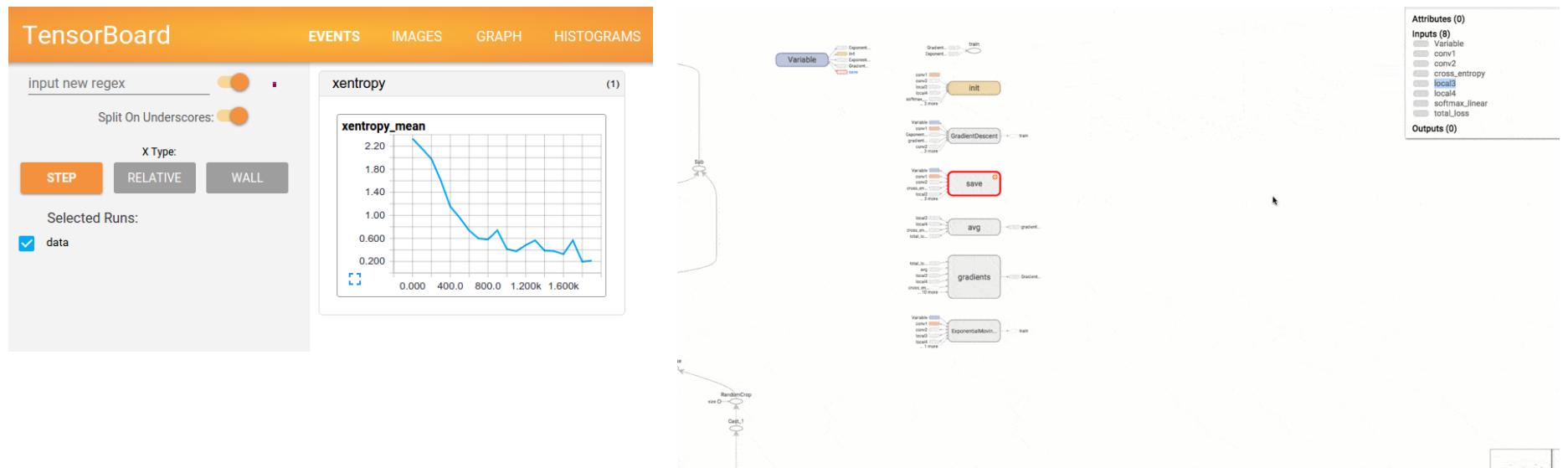
WATCH VIDEOS

About TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a

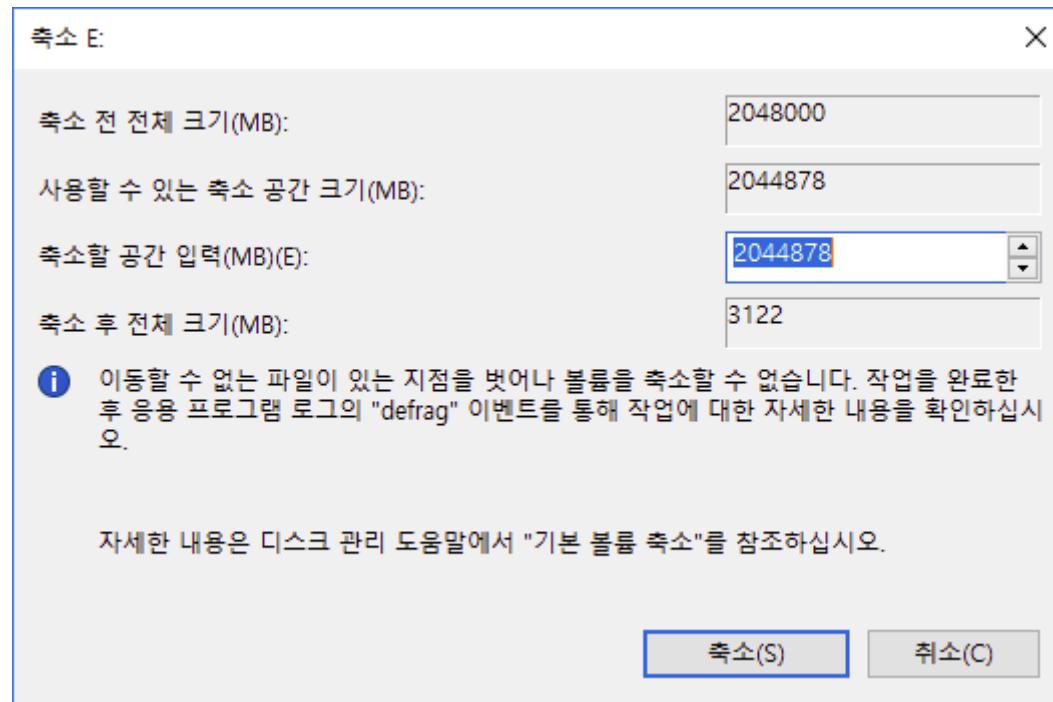
# TensorFlow ?

- 그냥 라이브러리 종류
- “편하게” 코드를 짤 수 있도록 도와줌
- Tensorboard, graph-node 등을 보여줌으로써 디버깅 기능을 가능하게 해줌



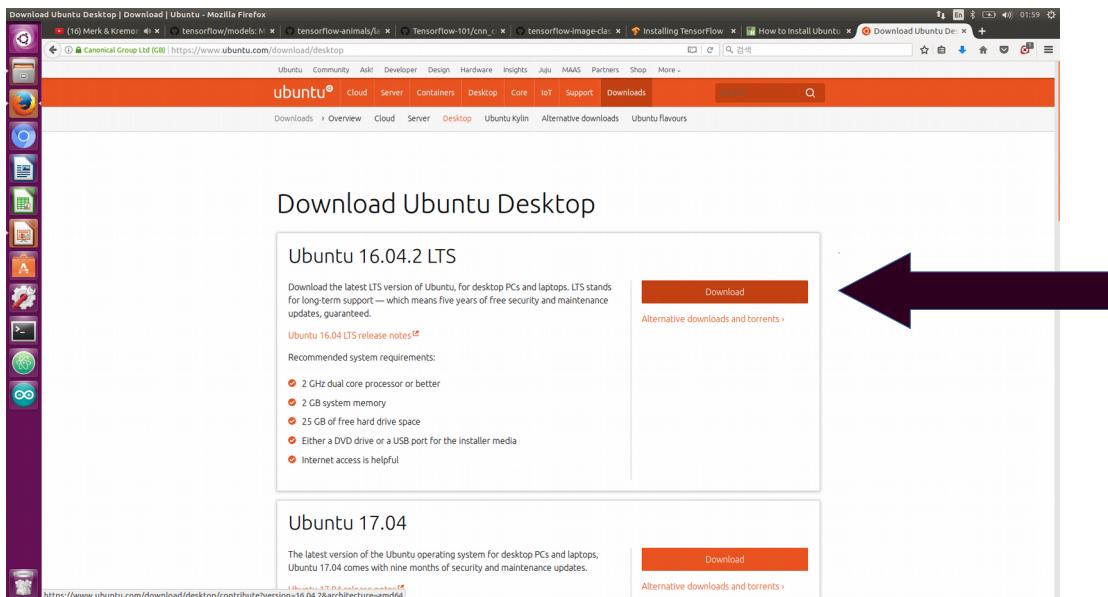
# Installation – Ubuntu 16.04

- 파티션 사이즈 재설정 – ubuntu 가 설치될 자리를 마련하는 것
- 내컴퓨터에서 디스크 파티션 사이즈를 축소 / 나누기 등이 가능



# Installation – Ubuntu 16.04

- <https://www.ubuntu.com/download/desktop>



- 64-bit로 설치할 것.
- [오래된 컴퓨터 / 미니 컴퓨터의 경우, 32-bit일 수 있으니 설치 전에 자기 컴퓨터 확인]

# Installation – Ubuntu 16.04

- 준비해 둔 usb 등에 ubuntu .iso 파일을 구워야 함
- 참고 <http://sergeswin.com/1178>
- 현재 os 기준으로 굽는법:
  - windows : <https://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/>
  - ubuntu : Gparted 등을 이용

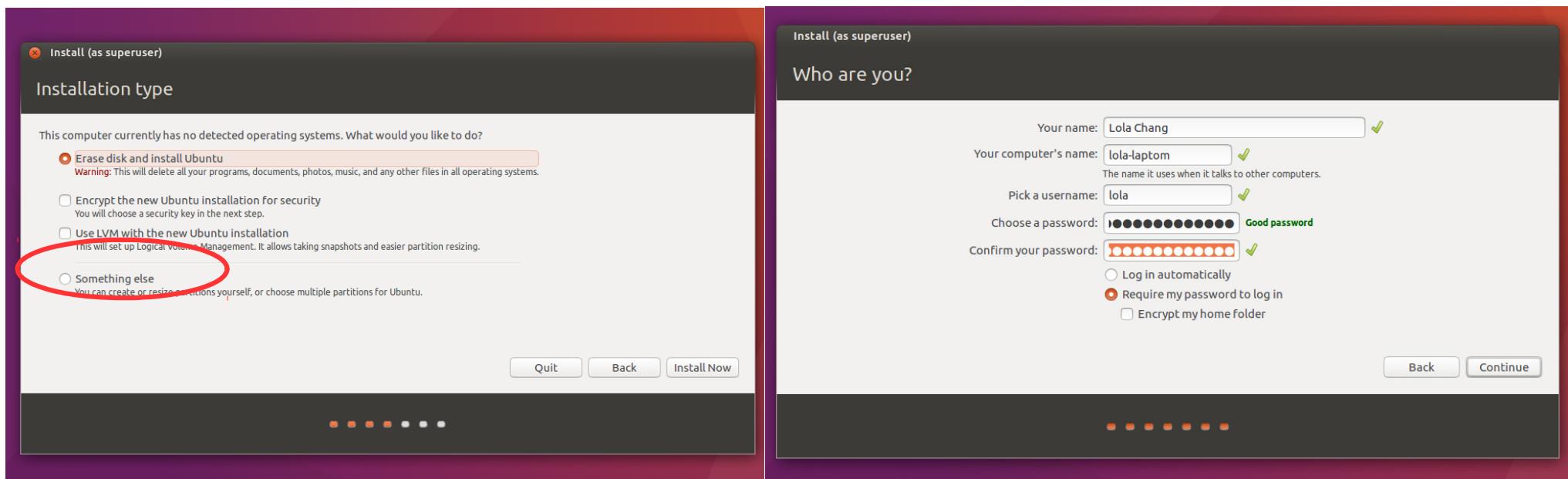
**주의 사항: 기존 os를 미리 백업하고 우분투 설치할 것 !!!**

**윈도우 파티션을 절대로 건들지 말 것 !!! ← 복구 불가능함**

**넉넉 잡아서 2~30기가 정도로 설치할 것**

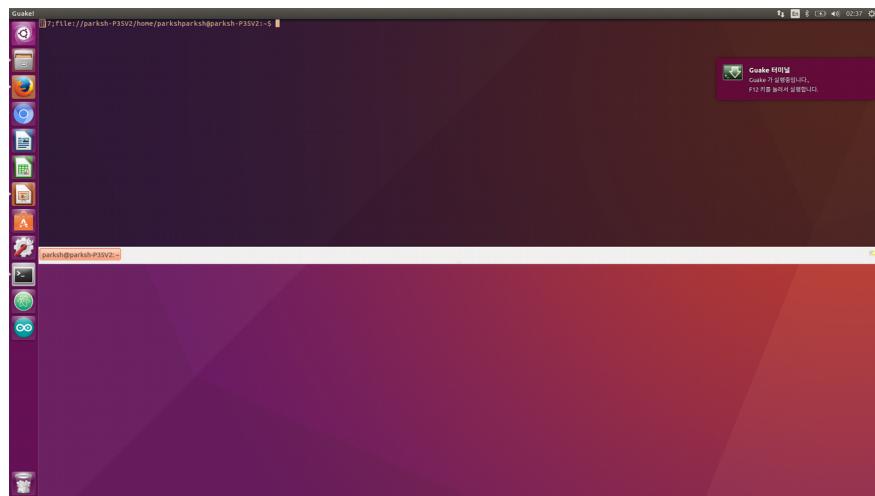
# Installation – Ubuntu 16.04

- 전원을 껼다가 키 뒤에 Bios에서 우분투가 설치된 usb로 부팅 시작
- Install Ubuntu 선택해서 부트 시작
- 무조건 “something else”로 설치해야 한다



# Installation – Ubuntu 16.04

- 지역 / 언어 [영어 추천] 설정 뒤에 설치 기다리면 끝
- 절대로 사용자 이름 등을 한국어나 다른 언어로 설정하지 말 것!!!



- 추가로 설치하면 좋은 것들 [1] : guake terminal // 간지 + 터미널 키기 편함
- 추가로 설치하면 좋은 것들 [2] : gnome theme // 이쁨

# Installation - TensorFlow

- 우선 python 부터 설치
- [다른 플러그인들과의 호환성을 위해 2.7 설치 요망]
- <https://www.python.org/downloads/source/>

```
localhost:~# tar xfz Python-2.3.tgz
localhost:~# cd Python-2.3
localhost:~/Python-2.3# ./configure
checking MACHDEP... linux2
checking EXTRAPLATDIR...
checking for --without-gcc... no
...
localhost:~/Python-2.3# make
gcc -pthread -c -fno-strict-aliasing -DNDEBUG -g -O3 -Wall -Wstrict-prototypes
-I. -I./Include -DPy_BUILD_CORE -o Modules/python.o Modules/python.c
gcc -pthread -c -fno-strict-aliasing -DNDEBUG -g -O3 -Wall -Wstrict-prototypes
-I. -I./Include -DPy_BUILD_CORE -o Parser/acceler.o Parser/acceler.c
gcc -pthread -c -fno-strict-aliasing -DNDEBUG -g -O3 -Wall -Wstrict-prototypes
-I. -I./Include -DPy_BUILD_CORE -o Parser/grammar1.o Parser/grammar1.c
...
localhost:~/Python-2.3# make install
/usr/bin/install -c python /usr/local/bin/python2.3
...
localhost:~/Python-2.3# exit
logout
localhost:~$ which python
/usr/local/bin/python
localhost:~$ python
Python 2.3.1 (#2, Sep 24 2003, 11:39:14)
[GCC 3.3.2 20030908 (Debian prerelease)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> [press Ctrl+D to get back to the command prompt]
-----
```

# Installation - TensorFlow

- Python virtualenv 설정
- 왜 사용하는가 - 기존의 파이썬 환경 설정과의 충돌을 방지하기 위해
- 참고 :  
<https://www.youtube.com/watch?v=N5vscPTWKOk&ytbChannel=Corey%20Schafer>
- 추가로 설치할 때 : 현재 사용중인 파이썬을 virtualenv python으로 변경 뒤 pip install 하면 됨

# Installation - TensorFlow

- 예시

```
parksh@parksh-P35V2: ~/Documents/example_of_virtualenv
parksh@parksh-P35V2:~$ pip install virtualenv
Requirement already satisfied: virtualenv in /usr/local/lib/python2.7/dist-packages
parksh@parksh-P35V2:~$ cd ~/Documents/
parksh@parksh-P35V2:~/Documents$ mkdir example_of_virtualenv
parksh@parksh-P35V2:~/Documents$ ls
ARM_project dl-workshop example_of_virtualenv ml_seminar python_project
parksh@parksh-P35V2:~/Documents$ cd example_of_virtualenv/
parksh@parksh-P35V2:~/Documents/example_of_virtualenv$ virtualenv example_env
New python executable in /home/parksh/Documents/example_of_virtualenv/example_env/bin/python
Installing setuptools, pip, wheel...done.
parksh@parksh-P35V2:~/Documents/example_of_virtualenv$ ls
example_env
parksh@parksh-P35V2:~/Documents/example_of_virtualenv$ 버추얼 파이썬 쓰는방법
버추얼: 명령을 찾을 수 없습니다
parksh@parksh-P35V2:~/Documents/example_of_virtualenv$ source example_env/bin/activate
(example_env) parksh@parksh-P35V2:~/Documents/example_of_virtualenv$ which python
/home/parksh/Documents/example_of_virtualenv/example_env/bin/python
(example_env) parksh@parksh-P35V2:~/Documents/example_of_virtualenv$ pip list
DEPRECATION: The default format will switch to columns in the future. You can use --format=(legacy|columns) (or define a format=(legacy|columns) in your pip.conf under the [list] section) to disable this warning.
actionlib (1.11.9)
angles (1.9.11)
bondpy (1.7.19)
camera-calibration (1.12.20)
camera-calibration-parsers (1.11.12)
catkin (0.7.6)
controller-manager (0.11.5)
controller-manager-msgs (0.11.5)
cv-bridge (1.12.4)
diagnostic-analysis (1.9.0)
diagnostic-common-diagnostics (1.9.0)
diagnostic-updater (1.9.0)
dynamic-reconfigure (1.5.48)
gazebo-plugins (2.5.13)
gazebo-ros (2.5.13)
genCPP (0.5.5)
geneus (2.2.6)
genlisp (0.4.16)
genmsg (0.5.8)
genodejs (2.0.1)
genpy (0.6.5)
image-geometry (1.12.4)
interactive-markers (1.11.3)
laser-geometry (1.6.4)
message-filters (1.12.7)
pip (9.0.1)
pluginlib (1.10.5)
python-qt-binding (0.3.2)
qt-dotgraph (0.3.4)
qt-gui (0.3.4)
qt-gui-cpp (0.3.4)
qt-gui-py-common (0.3.4)
resource-retriever (1.12.3)
rosbag (1.12.7)
rosboost-cfg (1.13.5)
rosclean (1.13.5)
roscreate (1.13.5)
rosgraph (1.12.7)
roslaunch (1.12.7)
roslib (1.13.5)
roslint (0.11.0)
roslz4 (1.12.7)
```

# Installation - TensorFlow

- TensorFlow 설치 방법:
- [1] pip install 을 통해 설치하는 방법
- [https://www.tensorflow.org/install/install\\_linux](https://www.tensorflow.org/install/install_linux)
- [2] build from source
- [https://www.tensorflow.org/install/install\\_sources](https://www.tensorflow.org/install/install_sources)
- 확인방법 : pip list –format columns 에서 tensorflow 유무 확인

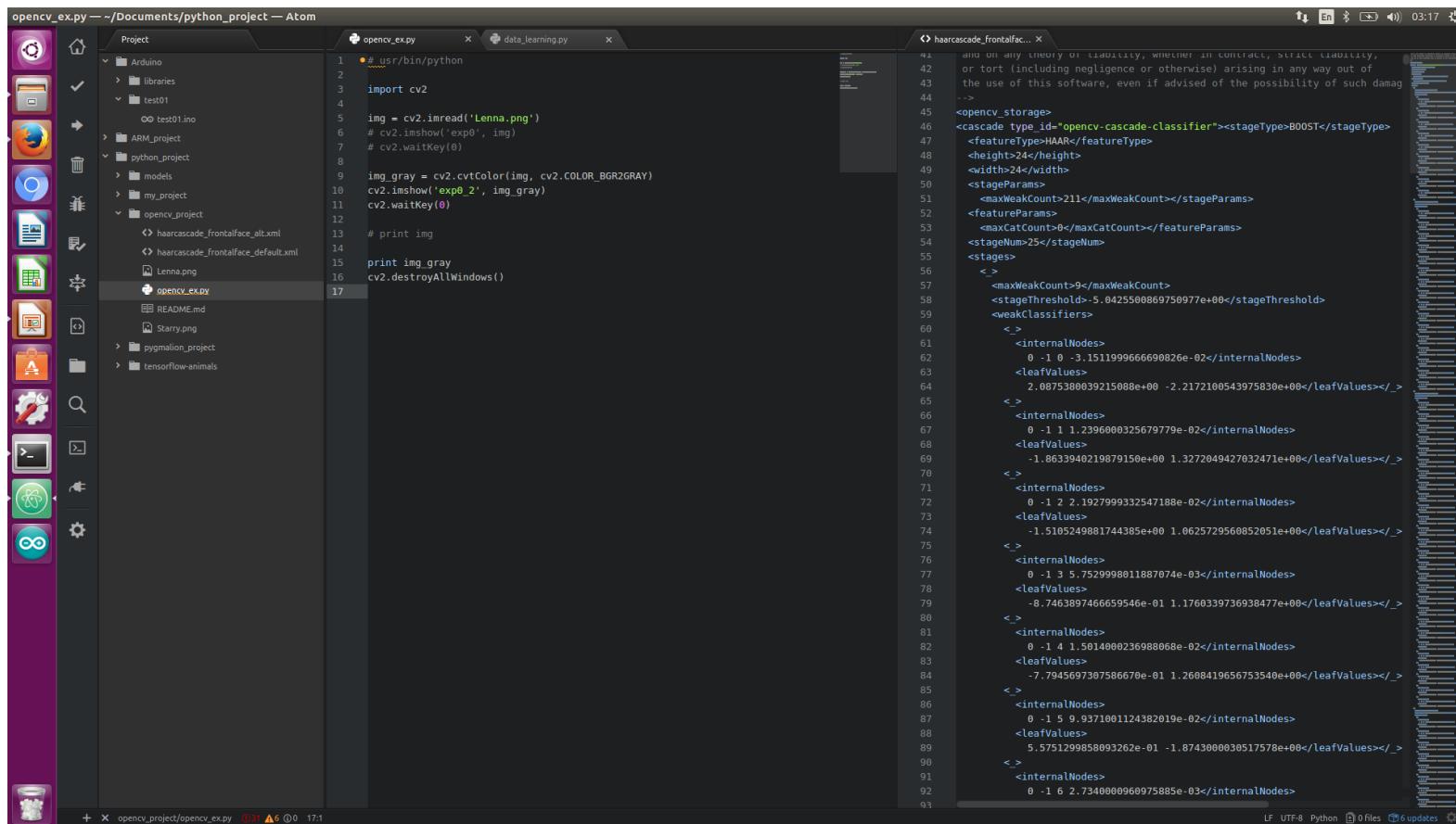
```
rqt-top          0.4.8
rqt-topic        0.4.8
rqt-web          0.4.8
rviz            1.12.10
scikit-image    0.13.0
scikit-learn     0.18.2
scipy           0.19.1
sensor-msgs      1.12.5
setuptools       36.0.1
six              1.10.0
smach            2.0.1
smach-ros        2.0.1
smclib           1.7.19
subprocess32     3.2.7
tensorflow        1.2.1
tf                1.11.8
tf-conversions   1.11.8
tf2-geometry-msgs 0.5.15
tf2-kdl          0.5.15
tf2-py            0.5.15
tf2-ros          0.5.15
Theano           0.9.0
topic-tools      1.12.7
urllib           1.21.1
urllib3          1.21.1
Werkzeug         0.12.2
wheel             0.29.0
xacro            1.11.2
(my_project) parksh@parksh-P35V2:~/Documents/python_project/my_project$
```

추가로 설치해야 할 것들:

- numpy // 연산, 데이터 가공
- matplotlib // 그래프 출력
- scipy // 이미지, 연산, 데이터 가공
- jedi // 에디터에서의 autocomplete
- scikit-image // 이미지 프로세싱

# Installation - Atom editor

- gedit 으로 코딩하든 vim을 쓰든 emacs를 쓰든 상관 없음
- 그 어떤 에디터를 사용해보지 못한 분들을 위한 추천: atom editor



The screenshot shows the Atom code editor interface. On the left is a sidebar with various icons for file operations like Open, Save, Find, and Settings. The main area has three tabs: 'opencv\_ex.py' (active), 'data\_learning.py', and 'haarcascade\_frontalface.xml'. The 'opencv\_ex.py' tab contains Python code for reading an image and displaying it. The 'haarcascade\_frontalface.xml' tab contains XML code for a Haar cascade classifier. The status bar at the bottom shows the file path (~ /Documents/python\_project), the current file name (opencv\_ex.py), line numbers (0-93), and other status indicators.

```
#_usr/bin/python
import cv2
img = cv2.imread('Lenna.png')
cv2.imshow('exp0', img)
cv2.waitKey(0)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow('exp0 2', img_gray)
cv2.waitKey(0)
print img_gray
cv2.destroyAllWindows()
```

```
<cascade type id="opencv-cascade-classifier"><stageType>BOOST</stageType>
<featureType>HAAR</featureType>
<height>24</height>
<width>24</width>
<stageParams>
<maxWeakCount>211</maxWeakCount>
<featureParams>
<maxCatCount>0</maxCatCount>
<stageNum>25</stageNum>
<stages>
<!-->
<maxWeakCount>9</maxWeakCount>
<stageThreshold>5.0425500869750977e+00</stageThreshold>
<weakClassifiers>
<!-->
<internalNodes>
0 -1 0 -3.1511999666690826e-02</internalNodes>
<leafValues>
0 -1 1.2396000325679779e-02</leafValues>
<!-->
<internalNodes>
0 -1 2 1.1927999332547188e-02</internalNodes>
<leafValues>
-1.8633940219879150e+00 1.327204942703247le+00</leafValues>
<!-->
<internalNodes>
0 -1 2 2.1927999332547188e-02</internalNodes>
<leafValues>
-1.5105249881744385e+00 1.062572956085205le+00</leafValues>
<!-->
<internalNodes>
0 -1 3 5.7529998011887074e-03</internalNodes>
<leafValues>
-8.7463897466659546e-01 1.1760339736938477e+00</leafValues>
<!-->
<internalNodes>
0 -1 4 1.5014000236988068e-02</internalNodes>
<leafValues>
-7.7945697307586670e-01 1.2608419656753540e+00</leafValues>
<!-->
<internalNodes>
0 -1 5 9.9371001124382019e-02</internalNodes>
<leafValues>
5.5751299858093262e-01 -1.8743800030517578e+00</leafValues>
<!-->
<internalNodes>
0 -1 6 2.7340000960975885e-03</internalNodes>
```

# Installation - Atom editor

- 추가 설치 패키지들:
- **autocomplete-python**
- **emmet**
- **file-icons**
- **intentionation**
- **linter-pylama**
- **platformio [시그마에서 활동하다보면 언젠가 100% 도움될 패키지]**

# Start

```
parksh@parksh-P35V2: ~/Documents/python_project
parksh@parksh-P35V2:~/Documents/python_project/my_project$ cd ..
parksh@parksh-P35V2:~/Documents/python_project$ source my_project/bin/activate
(my_project) parksh@parksh-P35V2:~/Documents/python_project$ which python
/home/parksh/Documents/python_project/my_project/bin/python
(my_project) parksh@parksh-P35V2:~/Documents/python_project$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow !')
>>> sess = tf.Session()
2017-07-28 03:23:41.124814: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 instructions, but these are available on your machine and could speed up CPU computations.
2017-07-28 03:23:41.124864: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.2 instructions, but these are available on your machine and could speed up CPU computations.
2017-07-28 03:23:41.124876: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX instructions, but these are available on your machine and could speed up CPU computations.
2017-07-28 03:23:41.124890: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX2 instructions, but these are available on your machine and could speed up CPU computations.
2017-07-28 03:23:41.124939: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use FMA instructions, but these are available on your machine and could speed up CPU computations.
>>> print (sess.run(hello))
Hello, TensorFlow !
>>> 
```

