

# CV Practice Class

## 1. Introduction

2017-06-21

FIRA 인공지능 에이전트 과정  
SNUVL Lab

# Contents

1. Installations of
  - a. Linux packages
  - b. Python packages
  - c. FFmpeg
  - d. OpenCV
2. Tutorials on
  - a. Linux
  - b. Python and Numpy
  - c. Matplotlib
  - d. Jupyter Notebook

# Installations

Linux packages

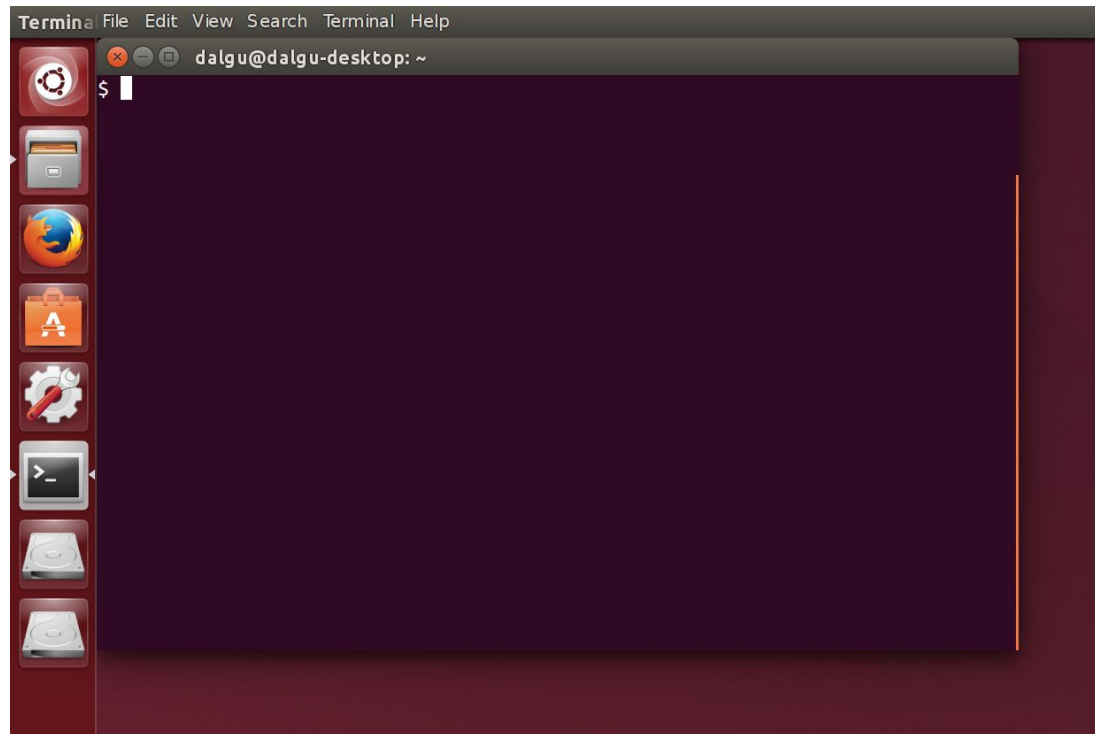
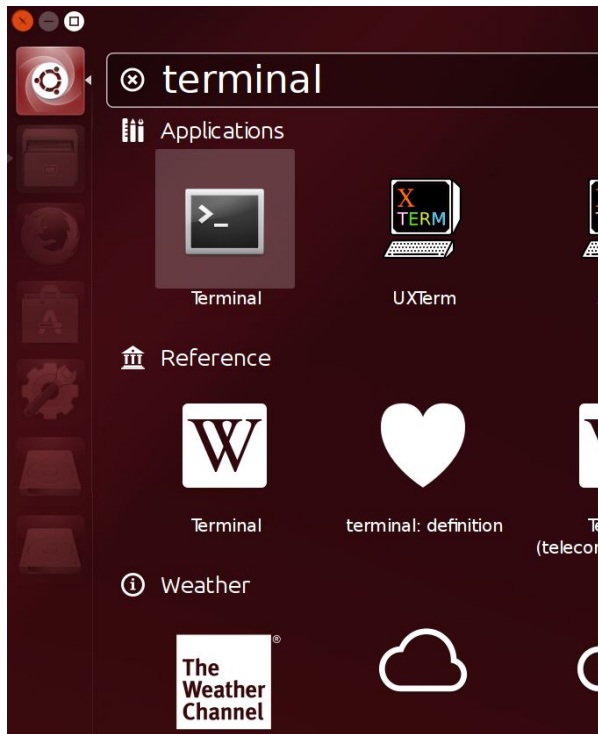
Python packages

FFmpeg

OpenCV

# Open Terminal Window

1. Find 'terminal' at the Dash window(Win-key)
2. Shortcut [Ctrl+Alt+T]



# Installations - Linux packages

- apt-get으로 패키지 관리
  - apt-get(Advanced Packaging Tool): Ubuntu와 같은 데비안(Debian)계열 Linux의 패키지 관리 명령어 도구
  - 패키지 설치/삭제

```
$ sudo apt-get install PACKAGE1 PACKAGE2 ...  
$ sudo apt-get remove PACKAGE1 PACKAGE2 ...
```

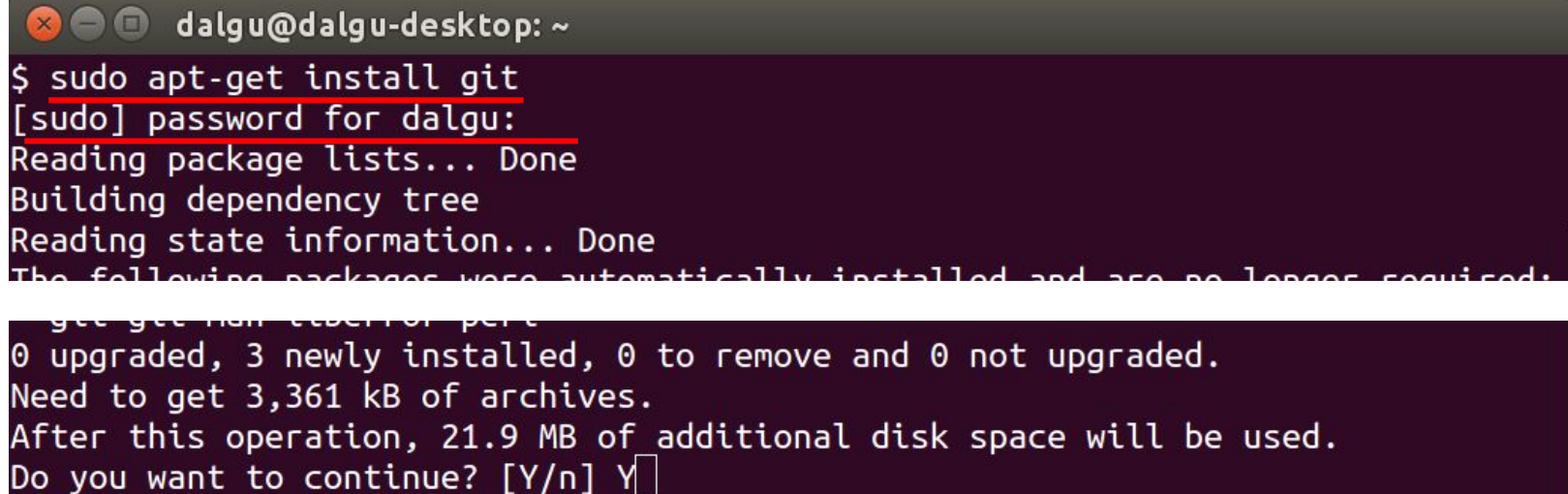
- sudo:root 권한으로 실행
- 사용 가능한 패키지들과 그 버전들의 리스트를 업데이트

```
$ sudo apt-get update
```

# Installations - Linux packages

- Git - 버전 관리 시스템(Version Control System; VCS)의 일종
  - 수업 자료를 git으로 관리
- Git 설치

```
$ sudo apt-get update  
$ sudo apt-get install git
```



A terminal window titled 'dalgu@dalgu-desktop: ~' showing the command 'sudo apt-get install git' being executed. The terminal output includes the password prompt, package list updates, dependency tree building, and a confirmation to continue the installation. The user has entered 'Y' to confirm.

```
dalgu@dalgu-desktop: ~  
$ sudo apt-get install git  
[sudo] password for dalgu:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following packages were automatically installed and are no longer required:  
git-gui git-man liberror-perl  
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.  
Need to get 3,361 kB of archives.  
After this operation, 21.9 MB of additional disk space will be used.  
Do you want to continue? [Y/n] Y
```

# Installations - Linux packages

- Git 확인

```
dalgu@dalgu-desktop: ~  
$ git  
usage: git [--version] [--help] [-C <path>] [-c name=value]  
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]  
        [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]  
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]  
        <command> [<args>]  
  
The most commonly used git commands are:  
add      Add file contents to the index  
bisect   Find by binary search the change that introduced a bug  
branch   List, create, or delete branches  
checkout Checkout a branch or paths to the working tree  
clone    Clone a repository into a new directory  
commit   Record changes to the repository  
diff     Show changes between commits, commit and working tree, etc  
fetch    Download objects and refs from another repository  
grep     Print lines matching a pattern  
init     Create an empty Git repository or reinitialize an existing one  
log      Show commit logs  
merge    Join two or more development histories together  
mv       Move or rename a file, a directory, or a symlink  
pull     Fetch from and integrate with another repository or a local branch  
push     Update remote refs along with associated objects  
rebase   Forward-port local commits to the updated upstream head
```

# Installations - Linux packages

- CMake
  - 멀티플랫폼에서 프로젝트/Makefile을 생성(configure)
  - CMake 설치

```
$ sudo apt-get install cmake
```

- GTK+ library
  - Ubuntu에서 이미지, 창 표시를 위한 GIMP toolkit library
  - GTK+ library 설치

```
$ sudo apt-get install libgtk2.0-dev  
$ sudo pkg-config libgtk2.0-dev
```

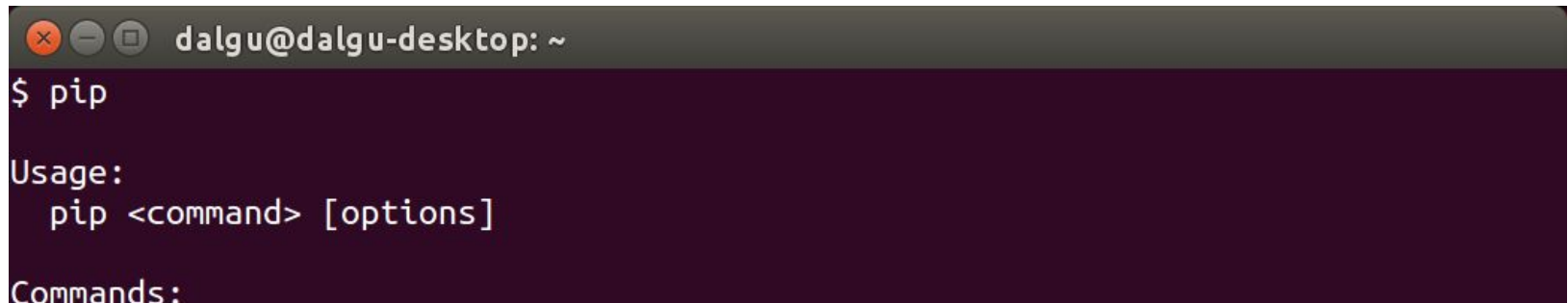


# Installations - Python packages

- pip - Python 패키지 관리 툴
  - pip 및 관련 패키지 설치

```
$ sudo apt-get install python-dev curl
$ curl -k -O https://bootstrap.pypa.io/get-pip.py
$ sudo python get-pip.py
```

- pip 설치 확인

A terminal window with a dark purple background. The title bar shows 'dalgu@dalgu-desktop: ~'. The prompt is '\$ pip'. The output shows 'Usage:' followed by 'pip <command> [options]' and 'Commands:' on the next line.

```
dalgu@dalgu-desktop: ~
$ pip
Usage:
  pip <command> [options]

Commands:
```

- pip로 python 패키지 설치

```
$ sudo pip install PACKAGE1 PACKAGE2 ...
```

# Installations - Python packages

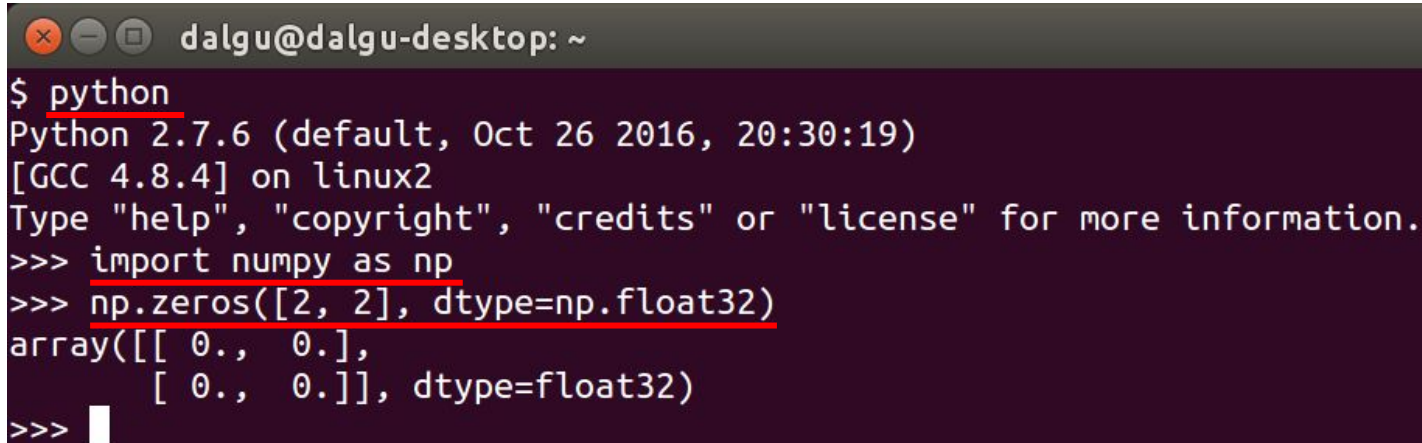
- Numpy 설치

- 다차원 배열 객체와 이와 관련한 기본적인 도구를 제공하는 Python 패키지
- 계산/과학분야의 Python 패키지들은 Numpy와 Numpy array를 사용한다
- pip으로 설치

```
$ sudo pip install numpy
```

- (시간소요; c++ 컴파일 필요)

- Numpy 설치 확인

A terminal window titled 'dalgu@dalgu-desktop: ~' showing the verification of Numpy installation. The user runs '\$ python', which starts a Python 2.7.6 shell. The user then runs '>>> import numpy as np' and '>>> np.zeros([2, 2], dtype=np.float32)', which outputs 'array([[ 0., 0.], [ 0., 0.]])'.

```
dalgu@dalgu-desktop: ~  
$ python  
Python 2.7.6 (default, Oct 26 2016, 20:30:19)  
[GCC 4.8.4] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import numpy as np  
>>> np.zeros([2, 2], dtype=np.float32)  
array([[ 0., 0.],  
       [ 0., 0.]])  
>>>
```

# Installations - Python packages

- 필요한 python 패키지 설치
  - Matplotlib, Jupyter notebook 과 필요 Linux 패키지

```
$ sudo apt-get install libpng-dev libfreetype6-dev  
$ sudo pip install matplotlib  
$ sudo pip install jupyter
```

- Matplotlib: Python 시각화 라이브러리
- Jupyter Notebook: 인터랙티브 python 컴퓨팅 웹 서비스

# Installations - FFmpeg

- FFmpeg
  - 동영상/이미지의 인코딩/디코딩을 위한 오픈소스 소프트웨어
- FFmpeg 설치
  - root 권한 shell 열기

```
$ sudo -i
```

- 필요 패키지 설치

```
# apt-get install -y yasm nasm libx264-dev
```

- FFmpeg 소스 파일 다운로드

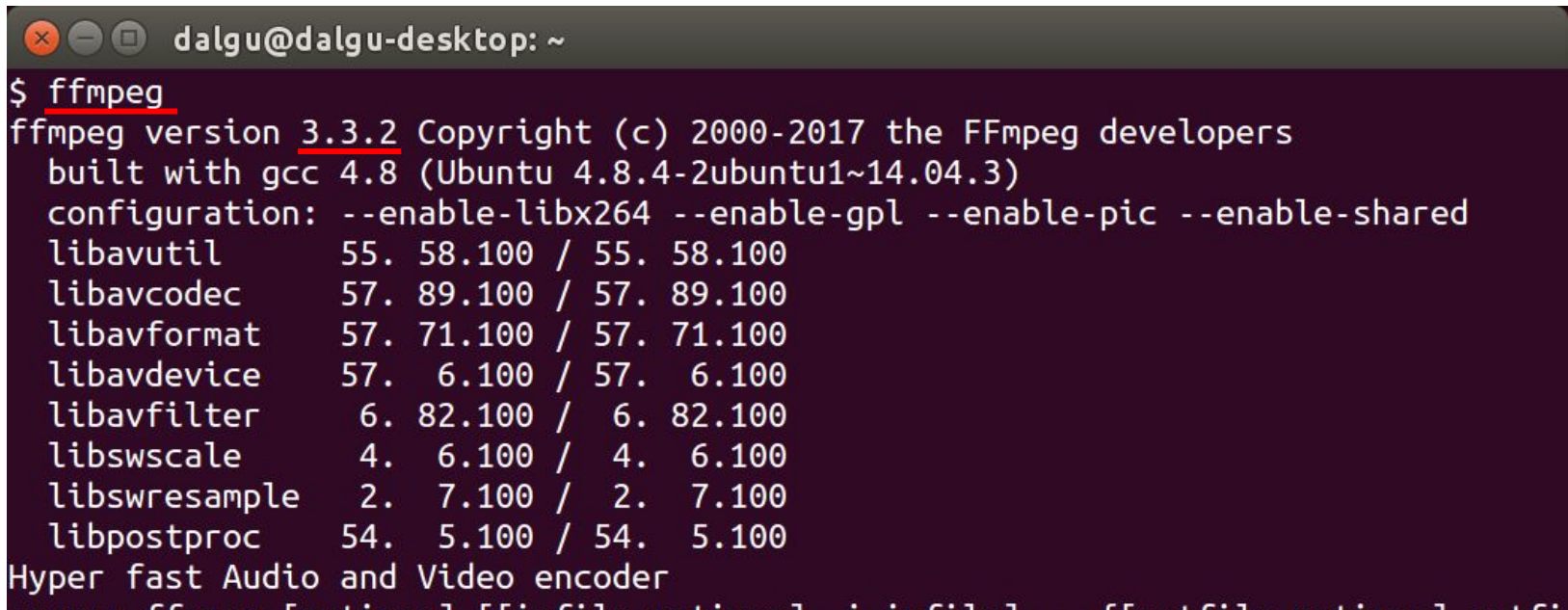
```
# cd /usr/src  
# wget https://ffmpeg.org/releases/ffmpeg-3.3.2.tar.gz  
# tar -xvzf ffmpeg-3.3.2.tar.gz  
# cd ffmpeg-3.3.2
```

# Installations - FFmpeg

- FFmpeg 설치
  - Configure, 컴파일, 설치

```
# ./configure --enable-libx264 --enable-gpl --enable-pic --enable-shared
# make -j8 && make install
# ldconfig
# exit
```

- FFmpeg 설치 확인



```
dalgu@dalgu-desktop: ~
$ ffmpeg
ffmpeg version 3.3.2 Copyright (c) 2000-2017 the FFmpeg developers
  built with gcc 4.8 (Ubuntu 4.8.4-2ubuntu1~14.04.3)
  configuration: --enable-libx264 --enable-gpl --enable-pic --enable-shared
  libavutil      55. 58.100 / 55. 58.100
  libavcodec     57. 89.100 / 57. 89.100
  libavformat    57. 71.100 / 57. 71.100
  libavdevice    57.  6.100 / 57.  6.100
  libavfilter     6. 82.100 /  6. 82.100
  libswscale     4.  6.100 /  4.  6.100
  libswresample  2.  7.100 /  2.  7.100
  libpostproc   54.  5.100 / 54.  5.100
Hyper fast Audio and Video encoder
```

# Installations - OpenCV

- OpenCV
  - 오픈소스 컴퓨터비전 라이브러리
  - C++로 작성, 실습에서는 Python wrapper(cv2) 사용
- OpenCV 설치
  - OpenCV 소스 코드, opencv\_contrib 소스 코드 다운로드 (opencv\_contrib: OpenCV's extra modules)



```
$ mkdir ~/opencv && cd ~/opencv
$ wget https://github.com/opencv/opencv/archive/3.2.0.zip -O
opencv-3.2.0.zip
$ wget https://github.com/opencv/opencv_contrib/archive/3.2.0.zip -O
opencv_contrib-3.2.0.zip
$ unzip opencv-3.2.0.zip && unzip opencv_contrib-3.2.0.zip
$ cd opencv-3.2.0
```

# Installations - OpenCV

- OpenCV 설치
  - Configure with CMake(flag 유의)

```
$ mkdir build && cd build  
$ cmake -D CMAKE_BUILD_TYPE=RELEASE -D WITH_CUDA=ON -D  
BUILD_NEW_PYTHON_SUPPORT=ON -D BUILD_EXAMPLES=ON -D WITH_FFMPEG=ON -D  
OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-3.2.0/modules ..
```

- 컴파일

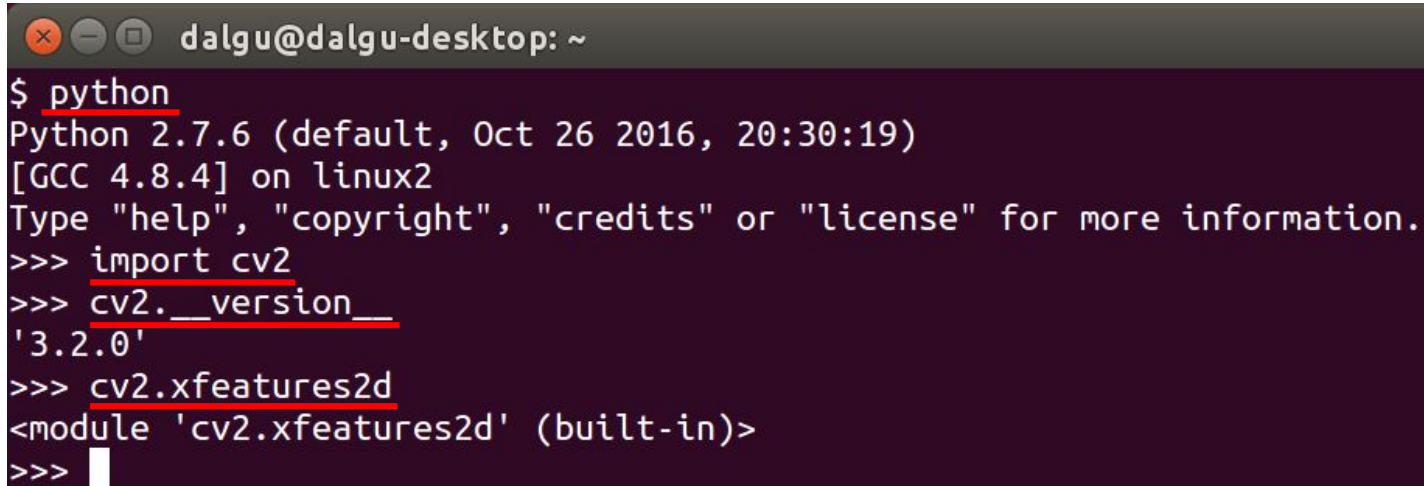
```
$ make -j8
```

- 설치

```
$ sudo make install  
$ sudo ldconfig
```

# Installations - OpenCV

- OpenCV 설치
  - 설치 확인

A terminal window with a dark purple background and white text. The window title bar shows 'dalgu@dalgu-desktop: ~'. The terminal output shows the execution of 'python', which starts the Python 2.7.6 interpreter. Inside the interpreter, the commands 'import cv2', 'cv2.\_\_version\_\_', and 'cv2.xfeatures2d' are executed, resulting in the version '3.2.0' and the module object '<module 'cv2.xfeatures2d' (built-in)>'.

```
$ python
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
[GCC 4.8.4] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.2.0'
>>> cv2.xfeatures2d
<module 'cv2.xfeatures2d' (built-in)>
>>> █
```



# Tutorials

Linux

Python and Numpy

Matplotlib

Jupyter Notebook

# Tutorial on Linux, especially on Terminal

- Linux
  - Unix 기반의 오픈소스 운영체제, 또는 그 커널
  - Ubuntu는 Linux의 여러 배포판 중 가장 많이 쓰이는 운영체제
- Some Facts
  - Linux는 기본적으로 CLI(Command Line Interface). GUI 에서 할 수 있는 것들은 대부분 터미널에서 할 수 있다.
  - 경로는 /(root)로 시작할 경우 절대경로, 그 외에는 상대경로이다.
  - 터미널의 시작 위치는 user의 home directory(~)이다.
  - 현재 위치는 . 부모 폴더는 ..
  - 파일, 폴더에는 user, group, other에 대한 권한이 있다.
  - 명령어, 폴더, 파일은 모두 case-sensitive
  - 모든 명령어의 도움말은 `man command_name`, `command_name -h`, `command_name --help` 를 사용해서 볼 수 있다.
  - 부분적으로 입력한 명령어, 파일, 폴더는 tab으로 자동 완성
  - 여러 flag는 한번에 입력 가능하다(`ls -alh`)



# Commands of Linux Terminal

- **cd**: 디렉토리 변경
  - `cd foldername`: 현재 폴더 하위의 `foldername`으로 이동
  - `cd:home` 폴더로 이동
  - `cd ..`: 상위 폴더로 이동
  - `cd /some/absolute/path`: 절대 경로로 이동
- **ls**: 파일 목록 불러오기
  - `ls -l`: 상세 보기, `ls -a`: 숨김파일 포함, `ls -h`: 파일 크기를 K,M,G로 표시

```
$ ls -l
drwxr-xr-x    4 cliff  user      1024 Jun 18 09:40 WAITRON_EARNIN
-rw-r--r--    1 cliff  user    767392 Jun  6 14:28 scanlib.tar.gz
^  ^  ^  ^      ^  ^      ^
|  |  |  |      |  |      |
|  |  |  |      | owner  group    size  date  time  name
|  |  |  |      number of links to file or directory contents
|  |  |  |      permissions for world
|  |  |  |      permissions for members of group
|  |  |  |      permissions for owner of file: r = read, w = write, x = execute
type of file: - = normal file, d=directory, l = symbolic link, ...
```

# Commands of Linux Terminal

- cp: 파일 복사

```
$ cp src_files dst_f_or_d  
$ cp -r src_dirs dst_f_or_d  
$ cp *.txt dst_dir  
$ cp prefix* dst_dir
```

- mv: 파일 이동(파일명 변경)

```
$ mv source_files dest_f_or_d
```

- rm: 파일 삭제

```
$ rm filenames  
$ rm -r dirnames  
$ rm -r files_or_dirs
```

- mkdir: 폴더 생성

```
$ mkdir folder_name  
$ mkdir -p /path/to/newfolder1/newfolder2
```

# Commands of Linux Terminal

- 권한 설정

\$ chmod 755 FILE	Changes the permissions of FILE to be rwx for the owner, and rx for the group and the world. (7 = rwx = 111 binary. 5 = r-x = 101 binary)
\$ chgrp GROUP FILE	Makes FILE belong to the group USER.
\$ chown USER FILE	Makes USER the owner of FILE.
\$ chown -R USER DIR	Makes USER the owner of DIR and everything in its directory tree.

- 파일 편집

\$ cat filename	Dump a file to the screen in ascii.
\$ more filename	Progressively dump a file to the screen: ENTER = One line down, SPACEBAR = page down, q=quit
\$ vi filename	Edit a file using the vi editor. All UNIX systems will have vi in some form.
\$ head -n filename	Show the first n lines of a file.
\$ tail -n filename	Show the last n lines of a file.

# Commands of Linux Terminal

- 기타 명령어

```
$ pwd          Show current directory
$ du -cksh dir_name  Show the size of folder dir_name
$ df -h         Calculate the disk space
$ ps aux        List current processes
$ top           Monitor processes
$ find . -name 'aaa.txt' Find files named 'aaa.txt' under the current
                  directory
```

- Pipe(|)와 redirection(>, >>)

- Pipe: 명령어의 결과를 다음 명령어의 인수로 사용
- Redirection: 명령어의 결과를 파일로 출력

```
$ ls -al | more      List long directory list progressively
$ du -sc * | sort -n  List all files and directory sorted by sizes
$ ls -al > newfile    Write(overwrite) directory list to the file
                     newfile
$ ls -al >> existingfile Append directory list at the end of the file
                     existingfile
```

# Tutorial on Python



- Python
  - 동적 프로그래밍 언어, 스크립트 언어
  - 고급 프로그래밍 언어(다중 패러다임; 객체지향 지원)
  - 짧고 뛰어난 가독성이 높은 코드
  - 실용적인 언어, 주요한 모든 플랫폼에서 사용 가능
- Python 버전
  - 현재 python 2.7과 python 3.6의 두 가지 버전이 있다.
  - 많은 계산/과학분야, 그리고 수업에서는 python 2.7 사용
- Python 스크립트 실행
  - .py 확장자

```
$ python SCRIPT.py
```

# Tutorial on Python

- Example python code
  - Quicksort 알고리즘
  - 높은 가독성, pseudo code 수준의 짧은 코드

```
def quicksort(arr):  
    if len(arr) <= 1:  
        return arr  
    pivot = arr[len(arr) / 2]  
    left = [x for x in arr if x < pivot]  
    middle = [x for x in arr if x == pivot]  
    right = [x for x in arr if x > pivot]  
    return quicksort(left) + middle + quicksort(right)  
  
print quicksort([3,6,8,10,1,2,1])  
# 출력 "[1, 1, 2, 3, 6, 8, 10]"
```



# Tutorial on Python

- Python 프롬프트 사용
  - 기본 python 프롬프트: `$ python`
  - IPython 프롬프트(가독성, 편의성): `$ ipython`
  - Jupyter Notebook(웹, 셀 단위 인터페이스, 시각화): `$ jupyter notebook`

```
$ python
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
[GCC 4.8.4] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> x = 3
>>> print type(x)
<type 'int'>
>>> print "Hello" + " World!"
Hello World!
>>> █
```

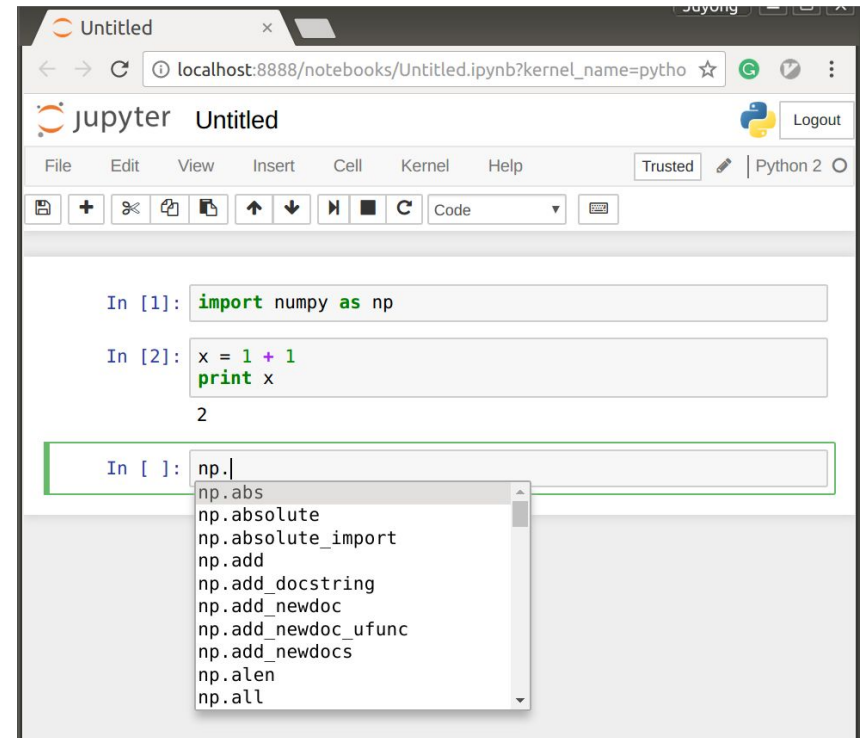
```
$ ipython
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
Type "copyright", "credits" or "license" for more information.

IPython 5.4.1 -- An enhanced Interactive Python.
?      -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help    -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
```

```
In [1]: import os
```

```
In [2]: os.get█
```

os.getcwd	os.geteuid	os.getlogin	os.getppid	os.getuid
os.getcwdu	os.getgid	os.getpgid	os.getresgid	
os.getegid	os.getgroups	os.getpgrp	os.getresuid	
os.getenv	os.getloadavg	os.getpid	os.getsid	



# Tutorial on Python

- 불리언(bool)
  - 모든 연산자를 기호, 단어로 지원(ex. and = &&)

```
t = True
f = False
print type(t) # 출력 "<type 'bool'>"
print t and f # 논리 AND; 출력 "False"
print t or f  # 논리 OR; 출력 "True"
print not t   # 논리 NOT; 출력 "False"
print t ^ f   # 논리 XOR; 출력 "True"
```

- 문자열(string)
  - 기본 자료형으로 string 지원. 다양한 기능을 지원

```
hello = 'hello'    # String 문자열을 표현할 땐 따옴표나
world = "world"    # 쌍따옴표가 사용된다; 어떤 걸 써도 상관없다.
print hello        # 출력 "hello"
print len(hello)   # 문자열 길이; 출력 "5"
print hello + ' ' + world # 문자열 연결; 출력 "hello world"
```

# Tutorial on Python

- Basic types
  - 숫자, 문자, 불리언, 문자열 등의 기본 자료형이 존재. 다른 언어와 유사
- 숫자
  - 정수형(integer)와 실수형(floats) (long, complex 타입도 존재)

```
x = 3
print type(x) # 출력 "<type 'int'>"
print x      # 출력 "3"
print x + 1   # 덧셈; 출력 "4"
print x * 2   # 곱셈; 출력 "6"
print x ** 2  # 제곱; 출력 "9"
x += 1       # ++, -- 연산자는 없음
print x      # 출력 "4"
x *= 2
print x      # 출력 "8"
y = 2.5
print type(y) # 출력 "<type 'float'>"
print y, y + 1, y * 2, y ** 2 # 출력 "2.5 3.5 5.0 6.25"
```

# Tutorial on Python

- 문자열(string)
  - string 객체에는 기본적으로 유용한 기능들을 제공
  - 모든 문자열 메소드들은 [python official document](#)에서 확인 가능

```
hw12 = '%s %s %d' % ("hello", "world", 12) # sprintf 방식 문자열 서식
print hw12 # 출력 "hello world 12"

s = "hello"
print s.capitalize() # 문자열을 대문자로 시작하게 함; 출력 "Hello"
print s.upper()      # 모든 문자를 대문자로 바꿈; 출력 "HELLO"
print s.rjust(7)      # 문자열 오른쪽 정렬, 빈공간은 여백; 출력 "  hello"
print s.center(7)     # 문자열 가운데 정렬, 빈공간은 여백; 출력 " hello "
print s.replace('l', '(e11)') # 첫 번째 인자를 두 번째 인자로 치환;
                                # 출력 "he(e11)(e11)o"
print '  world '.strip() # 문자열 앞뒤 공백 제거; 출력 "world"
```

# Tutorial on Python

- 리스트(list)
  - 슬라이싱(slicing): 리스트의 일부분에 접근하는 간결한 문법. MATLAB의 indexing과 유사

```
nums = range(5)      # range는 파이썬에 구현되어 있는 함수이며 정수들로
                      # 구성된 리스트를 만듭니다
print nums           # 출력 "[0, 1, 2, 3, 4]"
print nums[2:4]      # 인덱스 2에서 4(제외)까지 슬라이싱; 출력 "[2, 3]"
print nums[2:]        # 인덱스 2에서 끝까지 슬라이싱; 출력 "[2, 3, 4]"
print nums[:2]        # 처음부터 인덱스 2(제외)까지 슬라이싱; 출력 "[0, 1]"
print nums[:]         # 전체 리스트 슬라이싱; 출력 "[0, 1, 2, 3, 4]"
print nums[:-1]       # 슬라이싱 인덱스는 음수도 가능; 출력 "[0, 1, 2, 3]"
nums[2:4] = [8, 9]    # 슬라이스된 리스트에 새로운 리스트 할당
print nums            # 출력 "[0, 1, 8, 9, 4]"
```

# Tutorial on Python

- 컨테이너
  - 다른 변수들을 담을 수 있는 자료형
  - 기본적으로 리스트(list), 딕셔너리(dict), 집합(set), 튜플(tuple) 지원
- 리스트(list)
  - Python의 배열. 크기 변경이 가능하며 다른 자료형을 함께 저장
  - 배열의 index는 0부터 시작

```
xs = [3, 1, 2]      # 리스트 생성
print xs, xs[2]    # 출력 "[3, 1, 2] 2"
print xs[-1]       # 음수 인덱스, 리스트의 끝에서부터 세어짐; 출력 "2"
xs[2] = 'foo'      # 리스트는 자료형이 다른 요소들을 저장할 수 있다
print xs           # 출력 "[3, 1, 'foo']"
xs.append('bar')    # 리스트의 끝에 새 요소 추가
print xs           # 출력 "[3, 1, 'foo', 'bar']"
x = xs.pop()       # 리스트의 마지막 요소 삭제하고 반환
print x, xs        # 출력 "bar [3, 1, 'foo']"
```

# Tutorial on Python

- 리스트(list)
  - 반복분(for): 순차적으로 배열의 요소에 접근

```
animals = ['cat', 'dog', 'monkey']  
for animal in animals:  
    print animal  
# 출력 "cat", "dog", "monkey", 한 줄에 하나씩 출력.
```

- 요소의 인덱스와 함께 접근하고 싶다면 enumerate(built-in) 사용

```
animals = ['cat', 'dog', 'monkey']  
for idx, animal in enumerate(animals):  
    print '#%d: %s' % (idx + 1, animal)  
# 출력 "#1: cat", "#2: dog", "#3: monkey", 한 줄에 하나씩 출력.
```

# Tutorial on Python

- 리스트(list)
  - List comprehension: 위 코드 블록은 아래의 코드 블록과 동일하게 리스트 생성

```
nums = [0, 1, 2, 3, 4]
squares = []
for x in nums:
    squares.append(x ** 2)
print squares    # 출력 [0, 1, 4, 9, 16]
```

```
nums = [0, 1, 2, 3, 4]
squares = [x ** 2 for x in nums]
print squares    # 출력 [0, 1, 4, 9, 16]
```

- List comprehension에 조건을 추가하는 것도 가능하다

```
nums = [0, 1, 2, 3, 4]
even_squares = [x ** 2 for x in nums if x % 2 == 0]
print even_squares    # 출력 "[0, 4, 16]"
```



# Tutorial on Python

- 딕셔너리(dict)
  - C++(STL), 자바의 map, 자바스크립트의 object와 유사
  - Key-value 쌍의 저장. Key로 value를 저장, 접근

```
d = {'cat': 'cute', 'dog': 'furry'} # 새로운 딕셔너리를 만듭니다
print d['cat'] # 딕셔너리의 값을 받음; 출력 "cute"
print 'cat' in d # 딕셔너리가 주어진 키를 가졌는지 확인; 출력 "True"
d['fish'] = 'wet' # 딕셔너리의 값을 지정
print d['fish'] # 출력 "wet"
# print d['monkey'] # KeyError: 'monkey' not a key of d
print d.get('fish', 'N/A') # 딕셔너리의 값을 받음. 존재하지 않는다면
'N/A'; 출력 "wet"
print d.get('monkey', 'N/A') # 딕셔너리의 값을 받음. 존재하지 않는다면
'N/A'; 출력 "N/A"
del d['fish'] # 딕셔너리에 저장된 요소 삭제
print d.get('fish', 'N/A') # "fish"는 더 이상 열쇠가 아님; 출력 "N/A"
```

# Tutorial on Python

- 딕셔너리(dict)
  - 반복문: 리스트와 유사. Key, value에 동시에 접근하기 위해서 `iteritems` 사용

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal in d:
    legs = d[animal]
    print 'A %s has %d legs' % (animal, legs)

for animal, legs in d.iteritems():
    print 'A %s has %d legs' % (animal, legs)
# 둘 다 "A person has 2 legs", "A spider has 8 legs", "A cat has 4
legs", 한 줄에 하나씩 출력.
```

- 딕셔너리 comprehension: 리스트와 유사하게 딕셔너리 생성

```
nums = [0, 1, 2, 3, 4]
even_num_to_square = {x: x ** 2 for x in nums if x % 2 == 0}
print even_num_to_square # 출력 "{0: 0, 2: 4, 4: 16}"
```

# Tutorial on Python

- 집합(set)
  - 순서 구분이 없는 컨테이너.

```
animals = {'cat', 'dog'}
print 'cat' in animals    # 요소가 집합에 있는지 확인; 출력 "True"
print 'fish' in animals   # 출력 "False"
animals.add('fish')        # 요소를 집합에 추가
print 'fish' in animals    # 출력 "True"
print len(animals)         # 집합에 포함된 요소의 수; 출력 "3"
animals.add('cat')         # 이미 포함되어있는 요소를 추가할 경우 아무 변화
없음
animals.remove('cat')      # 집합에 포함된 요소를 제거
print len(animals)        # 출력 "2"
```

```
animals = {'cat', 'dog', 'fish'}
for idx, animal in enumerate(animals):
    print '#%d: %s' % (idx + 1, animal)
# 출력 "#1: fish", "#2: dog", "#3: cat", 한 줄에 하나씩 출력.
```

# Tutorial on Python

- 집합(set)
  - Set comprehension: 리스트, 딕셔너리와 마찬가지로의 방법으로 집합 생성

```
from math import sqrt
nums = {int(sqrt(x)) for x in range(30)}
print nums # 출력 "set([0, 1, 2, 3, 4, 5])"
```

- 튜플(tuple)
  - 순서가 있으며 값, 크기가 변하지 않는 리스트
  - 튜플은 딕셔너리의 key, 집합의 요소가 될 수 있지만 리스트는 될 수 없다

```
d = {(x, x + 1): x for x in range(10)} # 튜플을 열쇠로 하는 딕셔너리
# 생성
t = (5, 6) # 튜플 생성
print type(t) # 출력 "<type 'tuple'>"
print d[t] # 출력 "5"
print d[(1, 2)] # 출력 "1"
```

# Tutorial on Python

- 함수
  - Python의 함수는 `def` 키워드를 통해 정의
  - Python은 indent가 scope를 대신한다

```
def sign(x):  
    if x > 0:  
        return 'positive'  
    elif x < 0:  
        return 'negative'  
    else:  
        return 'zero'  
  
for x in [-1, 0, 1]:  
    print sign(x)  
# 출력 "negative", "zero", "positive", 한 줄에 하나씩 출력.
```

# Tutorial on Python

- 함수
  - 선택적으로 받는 인자가 가능(오버라이딩을 적게 할 수 있다)

```
def hello(name, loud=False):  
    if loud:  
        print 'HELLO, %s!' % name.upper()  
    else:  
        print 'Hello, %s' % name  
  
hello('Bob') # 출력 "Hello, Bob"  
hello('Fred', loud=True) # 출력 "HELLO, FRED!"
```

# Tutorial on Python

- 클래스
  - `class` 키워드로 선언. 기본적으로 `object` 클래스를 상속
  - 인스턴스에 `self`로 접근
  - 클래스에 대한 더 많은 정보는 [python official document](#)를 참조

```
class Greeter(object):  
    # 생성자  
    def __init__(self, name):  
        self.name = name # 인스턴스 변수 선언  
    # 인스턴스 메소드  
    def greet(self, loud=False):  
        if loud:  
            print 'HELLO, %s!' % self.name.upper()  
        else:  
            print 'Hello, %s' % self.name  
  
g = Greeter('Fred') # Greeter 클래스의 인스턴스 생성  
g.greet()           # 인스턴스 메소드 호출; 출력 "Hello, Fred"  
g.greet(loud=True)  # 인스턴스 메소드 호출; 출력 "HELLO, FRED!"
```

# Tutorial on Numpy



- Numpy
  - Python의 계산/과학분야에서 핵심적인 하는 라이브러리
  - 고성능의 다차원 배열 객체와 이와 관련한 메소드 제공
- 배열(Numpy array)
  - 동일한 자료형을 가지는 다차원의 배열
  - rank: 배열의 차원, shape: 배열의 크기(각 차원의 크기가 모인 튜플)

```
import numpy as np # Numpy 패키지 import

a = np.array([1, 2, 3]) # rank가 1인 배열 생성
print type(a)           # 출력 "<type 'numpy.ndarray'>"
print a.shape           # 출력 "(3,)"
print a[0], a[1], a[2]  # 출력 "1 2 3"
a[0] = 5                 # 요소를 변경
print a                  # 출력 "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]]) # rank가 2인 배열 생성
print b.shape            # 출력 "(2, 3)"
print b[0, 0], b[0, 1], b[1, 0] # 출력 "1 2 4"
```



# Tutorial on Numpy

- 배열(Numpy array)
  - 기본적인 형태의 배열을 생성하기 위한 다양한 함수 제공

```
a = np.zeros((2,2)) # 모든 값이 0인 배열 생성
print a             # 출력 "[[ 0.  0.]
                    #      [ 0.  0.]]"
b = np.ones((1,2))  # 모든 값이 1인 배열 생성
print b             # 출력 "[[ 1.  1.]]"
c = np.full((1,3), 7) # 모든 값이 특정 상수인 배열 생성
print c             # 출력 "[[ 7.  7.  7.]]"
d = np.eye(2)        # 2x2 단위행렬 생성
print d             # 출력 "[[ 1.  0.]
                    #      [ 0.  1.]]"
e = np.random.random((2,2)) # 임의의 값으로 채워진 배열 생성
print e             # 임의의 값 출력 "[[ 0.9194  0.0814]
                    #      [ 0.6874  0.8723]]"
```

# Tutorial on Numpy

- 배열(Numpy array)
  - 배열 슬라이싱(slicing): Python 리스트와 유사. 각 차원에 대해 인덱스를 표시

```
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# 슬라이싱을 이용하여 첫 두 행과 1열, 2열로 이루어진 부분배열;
# b는 shape가 (2,2)인 배열이 됩니다:
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]

# 슬라이싱된 배열은 원본 배열과 같은 데이터를 참조한다,
# 즉 슬라이싱된 배열을 수정하면 원본 배열 역시 수정된다.
print a[0, 1]    # 출력 "2"
b[0, 0] = 77     # b[0, 0]은 a[0, 1]과 같은 데이터
print a[0, 1]    # 출력 "77"
```

# Tutorial on Numpy

- 배열(Numpy array)
  - 배열 슬라이싱(slicing): 슬라이싱은 인덱싱과 다르게 rank를 보존한다.

```
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# 배열의 중간 행에 접근하는 두 가지 방법이 있습니다.
# 정수 인덱싱과 슬라이싱을 혼합해서 사용하면 낮은 rank의 배열이 생성되지만,
# 슬라이싱만 사용하면 원본 배열과 동일한 rank의 배열이 생성됩니다.
row_r1 = a[1, :]      # 배열a의 두 번째 행을 rank가 1인 배열로
row_r2 = a[1:2, :]    # 배열a의 두 번째 행을 rank가 2인 배열로
print row_r1, row_r1.shape # 출력 "[5 6 7 8] (4,)"
print row_r2, row_r2.shape # 출력 "[[5 6 7 8]] (1, 4)"
```

# Tutorial on Numpy

- 배열(Numpy array)
  - 정수 배열 인덱싱: sparse하게 배열의 요소에 접근하는 방법

```
a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
print a # 출력 "array([[ 1,  2,  3],
#               [ 4,  5,  6],
#               [ 7,  8,  9],
#               [10, 11, 12]])"

# 정수 배열 인덱싱으로 요소에 접근
print a[[0, 1, 2, 3], [0, 2, 0, 1]] # 출력 "[ 1  6  7 11]"

# 정수 배열 인덱싱으로 요소 변경
a[[0, 1, 2, 3], [0, 2, 0, 1]] += 10

print a # 출력 "array([[11,  2,  3],
#               [ 4,  5, 16],
#               [17,  8,  9],
#               [10, 21, 12]])"
```

# Tutorial on Numpy

- 배열(Numpy array)
  - 불리언 배열 인덱싱: 조건으로 생성. 원배열과 크기가 같은 불리언 배열

```
a = np.array([[1,2], [3, 4], [5, 6]])

bool_idx = (a > 2) # 2보다 큰 a의 요소를 찾는다;
                  # bool_idx의 각 요소는 동일한 위치에 있는 a의
                  # 요소가 2보다 큰지를 말해준다.

print bool_idx      # 출력 "[[False False]
                   #      [ True  True]
                   #      [ True  True]]"

# 불리언 배열 인덱싱을 통해 bool_idx에서 True인
# rank 1인 배열을 구성
print a[bool_idx]   # 출력 "[3 4 5 6]"

# 위에서 한 모든것을 한 문장으로 할 수 있다:
print a[a > 2]      # 출력 "[3 4 5 6]"
```

# Tutorial on Numpy

- 배열(Numpy array)
  - 자료형:Numpy의 배열은 동일한 자료형의 격자. 다양한 자료형을 제공
  - 배열의 생성시 자료형을 지정 가능. 하지 않을 경우 스스로 추측한다

```
import numpy as np

x = np.array([1, 2]) # Numpy가 자료형을 추측해서 선택
print x.dtype       # 출력 "int64"

x = np.array([1.0, 2.0]) # Numpy가 자료형을 추측해서 선택
print x.dtype          # 출력 "float64"

x = np.array([1, 2], dtype=np.int64) # 특정 자료형을 명시적으로 지정
print x.dtype                       # 출력 "int64"
```

# Tutorial on Numpy

- 배열(Numpy array)
  - 기본적인 배열 간의 연산은 요소별(element-wise) 연산(내적은 dot 메소드)

```
x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)
print x + y
print np.add(x, y)
# [[ 6.0  8.0]
#  [10.0 12.0]]
print np.multiply(x, y) # x * y
# [[ 5.0 12.0]
#  [21.0 32.0]]
print np.divide(x, y) # x / y
# [[ 0.2          0.33333333]
#  [ 0.42857143  0.5         ]]
print np.sqrt(x)
# [[ 1.          1.41421356]
#  [ 1.73205081  2.         ]]
```

# Tutorial on Numpy

- 배열(Numpy array)

- Numpy는 배열을 다루는 다양한 함수를 제공한다. 많이 사용하는 함수는 `sum`

```
x = np.array([[1,2],[3,4]])
```

```
print np.sum(x) # 모든 요소를 합한 값을 연산; 출력 "10"
```

```
print np.sum(x, axis=0) # 각 열에 대한 합을 연산; 출력 "[4 6]"
```

```
print np.sum(x, axis=1) # 각 행에 대한 합을 연산; 출력 "[3 7]"
```

- 다차원 행렬의 전치(transpose)는 `np.transpose()` 또는 `.T` property 사용

```
x = np.array([[1,2], [3,4]])
```

```
print x # 출력 "[[1 2]  
# [3 4]]"
```

```
print x.T # 출력 "[[1 3]  
# [2 4]]"
```

```
# rank 1인 배열을 전치할 경우 아무 일도 일어나지 않는다:
```

```
v = np.array([1,2,3])
```

```
print v # 출력 "[1 2 3]"
```

```
print v.T # 출력 "[1 2 3]"
```



# Tutorial on Numpy

- 배열(Numpy array)
  - 브로드캐스팅(broadcasting): shape가 다른 배열 간의 연산을 가능하게 한다
  - rank가 다를 경우 낮은 쪽의 rank를 높이고 그 차원의 크기를 늘린다

```
# 벡터 v를 행렬 x의 각 행에 더한 뒤, 그 결과를 행렬 y에 저장한다
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
```

```
# 명시적 반복문을 통해 행렬 x의 각 행에 벡터 v를 더하는 방법
```

```
for i in range(4):
    y[i, :] = x[i, :] + v
```

```
# 브로드캐스팅을 이용하여 v를 x의 각 행에 더하기
```

```
y = x + v
```

```
print y # 출력 "[[ 2  2  4]
#       [ 5  5  7]
#       [ 8  8 10]
#       [11 11 13]]"
```

# Tutorial on Matplotlib

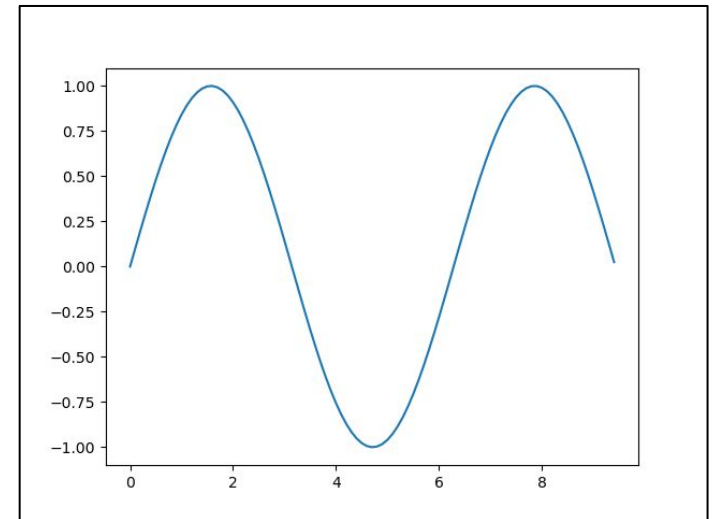


- Matplotlib
  - Python의 plotting 라이브러리.
  - MATLAB과 유사한 `matplotlib.pyplot` 모듈에 대해 소개
- Plotting
  - `plt.plot()` 함수 사용

```
import numpy as np
import matplotlib.pyplot as plt

# 사인 곡선의 x,y 좌표를 계산
x = np.arange(0, 3 * np.pi, 0.1)
y = np.sin(x)

# matplotlib를 이용해 점들을 그리기
plt.plot(x, y)
# 그래프 윈도우를 나타나게 하기 위해 호출
plt.show()
```

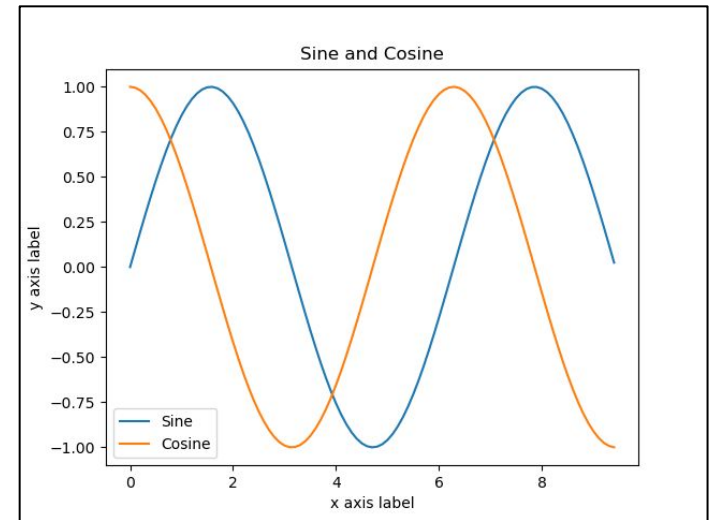


# Tutorial on Matplotlib

- Plotting
  - 여러 그래프를 한번에 표현
  - 그래프의 제목, 범주, 축 이름을 표시

```
# 사인과 코사인 곡선의 x,y 좌표를 계산
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# matplotlib를 이용해 점들을 그리기
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
plt.show()
```



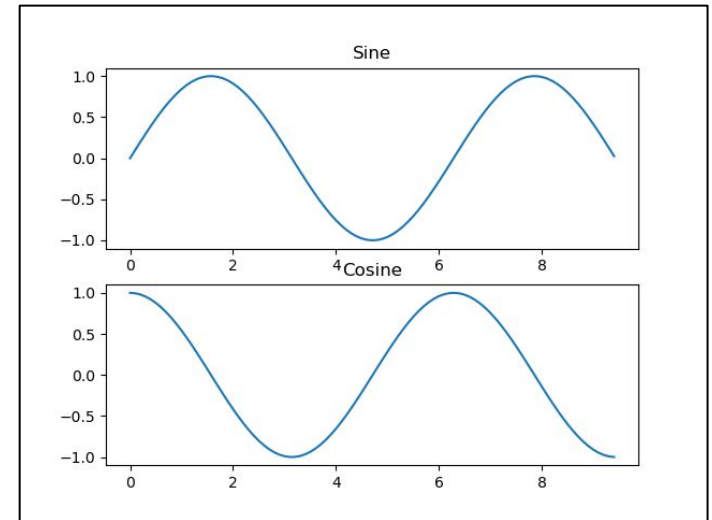
# Tutorial on Matplotlib

- Subplots

- `plt.subplot()`으로 구획을 나누어 활성화하고 여러 그래프를 격자로 표시

```
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# 높이 2, 너비 1인 subplot 구획을 설정,
# 첫 번째 구획을 활성화 하고 그리기
plt.subplot(2, 1, 1)
plt.plot(x, y_sin)
plt.title('Sine')
# 두 번째 subplot 구획을 활성화 하고 그리기
plt.subplot(2, 1, 2)
plt.plot(x, y_cos)
plt.title('Cosine')
# 그림 보이기.
plt.show()
```



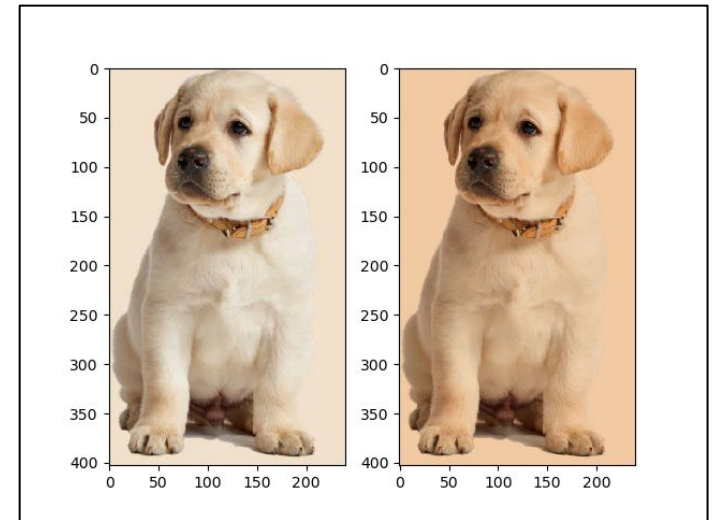
# Tutorial on Matplotlib

- 이미지 표시
  - `plt.imshow()` 함수를 사용해 이미지 표시

```
import cv2

img = cv2.cvtColor(cv2.imread('dog.jpg'),
cv2.COLOR_BGR2RGB)
img_tinted = img * [1, 0.95, 0.9]

# 원본 이미지 나타내기
plt.subplot(1, 2, 1)
plt.imshow(img)
# 색변화된 이미지 나타내기
plt.subplot(1, 2, 2)
# 명시적으로 자료형을 uint8로 형변환 해준다.
plt.imshow(np.uint8(img_tinted))
plt.show()
```

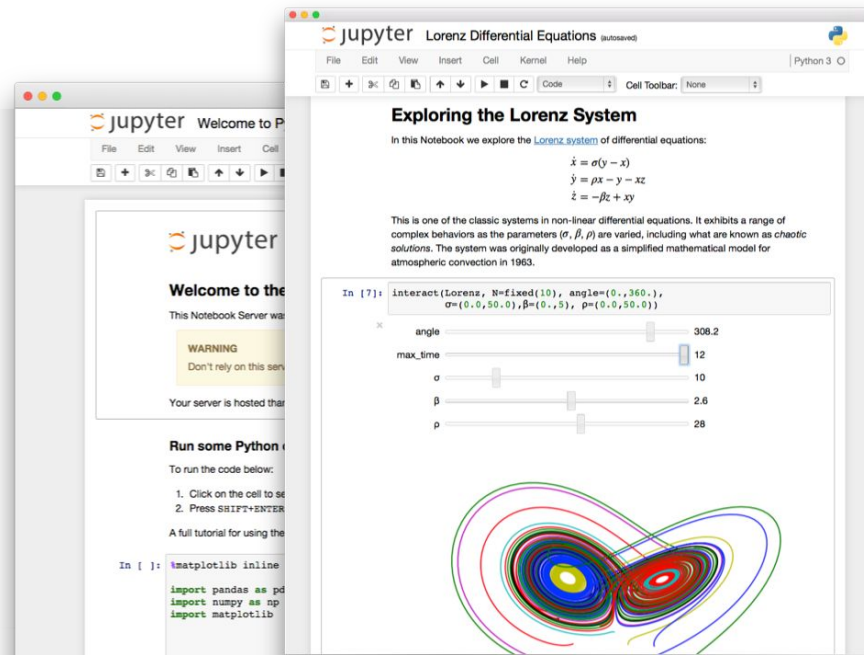


# Tutorial on Jupyter Notebook

- Jupyter Notebook
  - 웹에서 인터랙티브한 코딩을 할 수 있게 해주는 오픈소스 소프트웨어
  - 40+개 언어 지원, Markdown 표시 지원
- Jupyter Notebook 실행

```
$ jupyter notebook
```

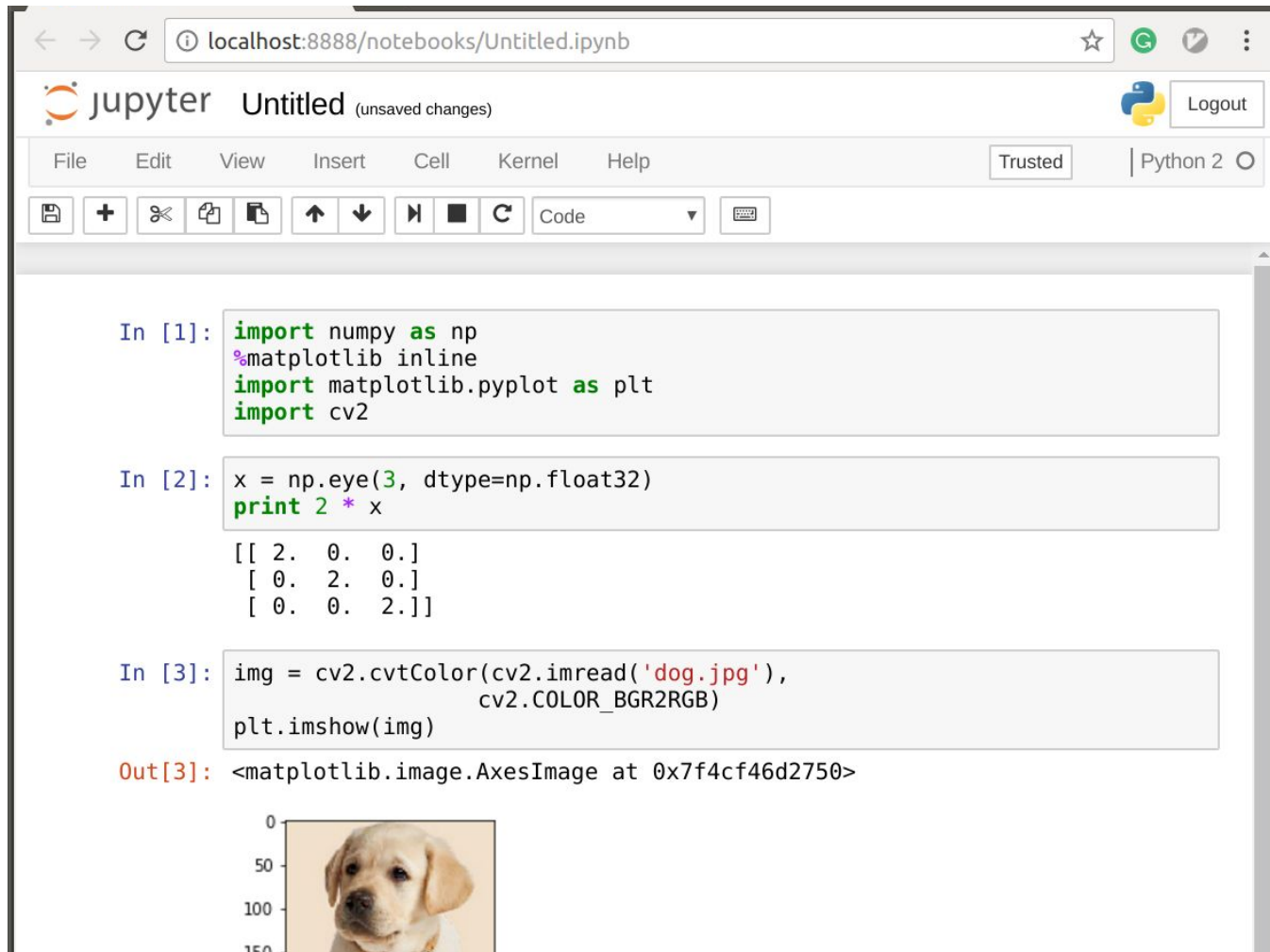
- <http://localhost:8888/>로 접속(실행 시 기본적으로 home 열림)



# Tutorial on Jupyter Notebook

- Jupyter Notebook

- 노트북 파일로 저장. 한 커널이 하나의 노트북을 실행. 코드는 셀 단위로 실행
- 한 커널 내에서 변수 공유. 페이지를 달아도 커널은 유지(종료는 shutdown)



# Reference

- Detailed instructions on installing OpenCV on Linux
  - [Official documentation](#) / [A blog](#)
- Python and Numpy tutorials
  - A part of [Stanford CS231n](#): [Original](#) / [한글 번역본](#)



# Q&A


Any Question?

# (Appendix) Ubuntu 16.04 Installation

- About
  - 본 슬라이드는 윈도우가 설치된 머신에서 우분투 16.04를 설치하는 방법에 대한 간략한 안내를 담고 있습니다.
- 설치방법
  - 듀얼부팅(가상머신 X): 가상 머신으로 사용하는 경우 성능 저하의 문제
- 준비물
  - USB 메모리(4GB 이상)
- 윈도우/우분투 듀얼부팅 시 유의사항
  - 설치 순서는 선 윈도우 후 우분투 설치가 편하다. 만약 우분투가 먼저 깔려있는 경우, 다시 윈도우 설치 후 우분투 재설치가 쉽다.
  - 그냥 우분투를 테스트 해보고 싶다면 **USB 없이**, 파티션 수정 없이 설치하는 방법(wubi)도 있다. (16.04를 공식지원 하지 않지만 가능, 성능 저하)

# (Appendix) Ubuntu 16.04 Installation







- 준비 - Ubuntu 16.04 설치 이미지 다운로드
  - Download a file named **ubuntu-16.04.2-desktop-amd64.iso** at <http://releases.ubuntu.com/16.04.2/>
  - desktop/server? amd64/i386?



## Ubuntu 16.04.2 LTS (Xenial Xerus)

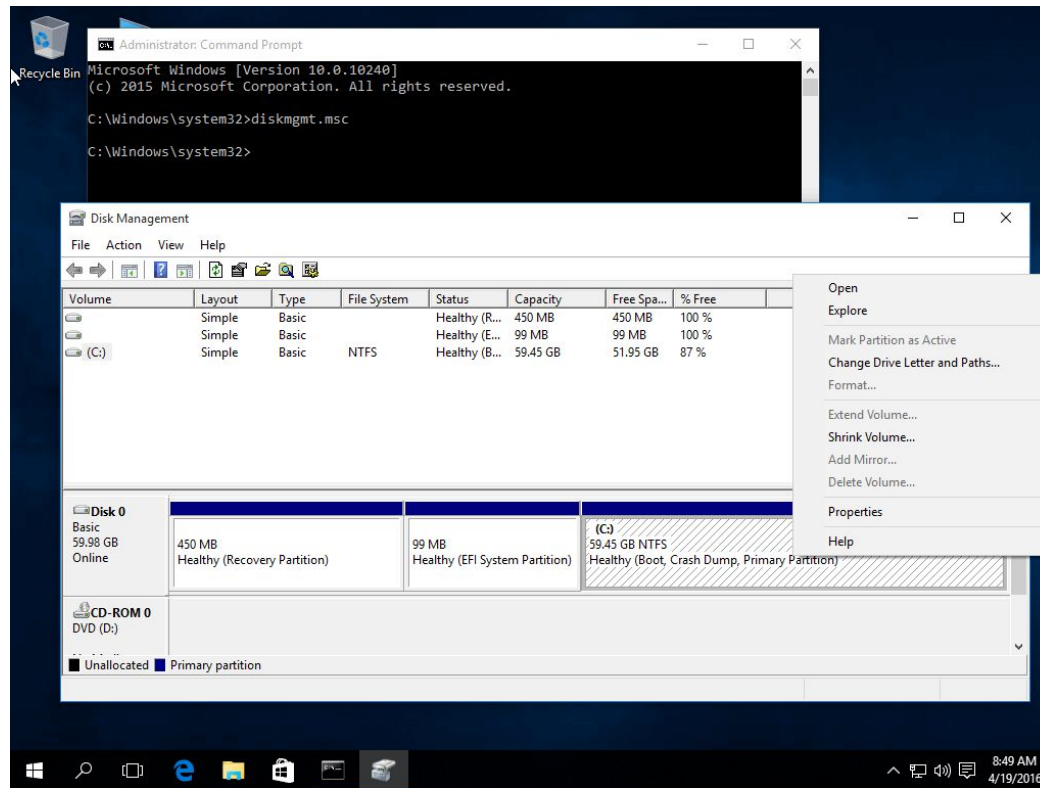
Select an image

...

	SHA256SUMS	2017-02-17 00:04	587	
	SHA256SUMS.gpg	2017-02-17 00:04	933	
	ubuntu-16.04.2-desktop-amd64.iso	2017-02-15 21:44	1.4G	Desktop image for 64-bit PC (AMD64) computers (standard dc
	ubuntu-16.04.2-desktop-amd64.iso.torrent	2017-02-16 23:57	58K	Desktop image for 64-bit PC (AMD64) computers (BitTorrent
	ubuntu-16.04.2-desktop-amd64.iso.zsync	2017-02-16 23:57	2.9M	Desktop image for 64-bit PC (AMD64) computers (zsync metaf
	ubuntu-16.04.2-desktop-amd64.list	2017-02-15 21:44	4.4K	Desktop image for 64-bit PC (AMD64) computers (file listir

# (Appendix) Ubuntu 16.04 Installation

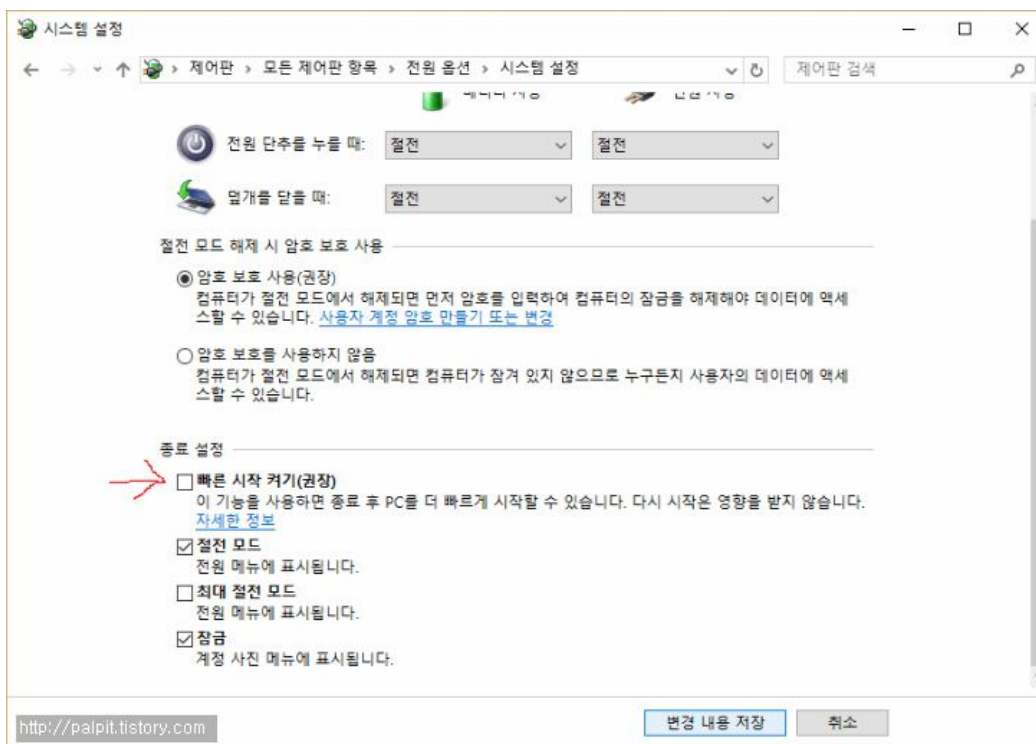
- 준비 - 파티션 볼륨 축소
  - 윈도우와 같은 하드디스크에 설치하나 별도의 용량을 할당해놓지 않은 경우, 윈도우 파티션의 용량을 줄여 확보
  - 1. 디스크 관리자 실행: [시작] → [실행] → 'diskmgmt.msc'
  - 2. 줄이고자 하는 파티션에 마우스 오른쪽 쪽 → [볼륨 축소]
  - 3. 51200~102400MB(50~100GB) 정도 할당한 후 [축소]



# (Appendix) Ubuntu 16.04 Installation

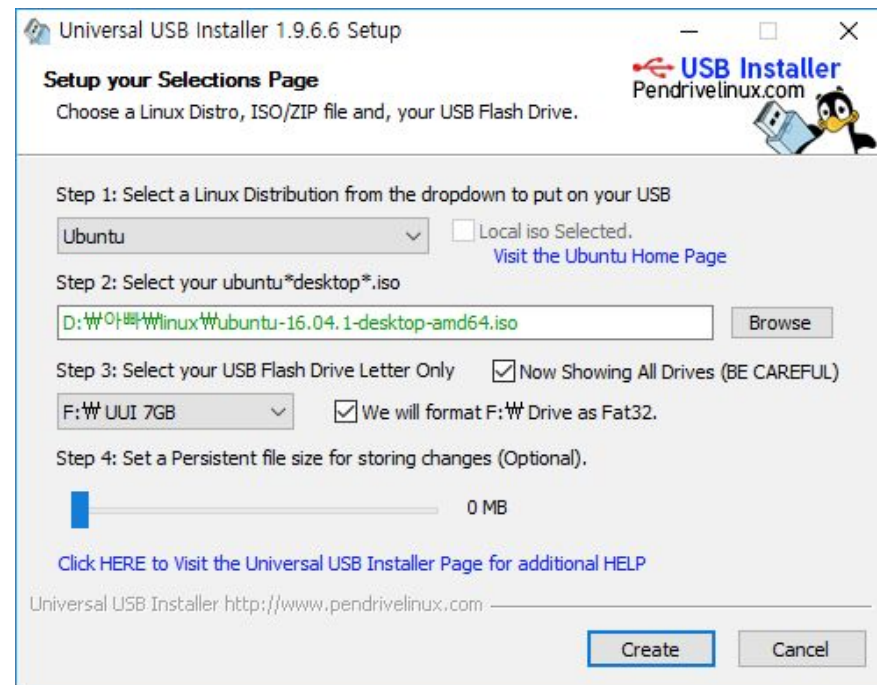
- 준비 - Fast Startup 해제

- 윈도우(>8)에서부터 빠른 부팅을 위해 시스템 종료를 최대 절전모드와 유사하게 수행. CMOS 진입/디스크 사용을 위해 해제 필요
- 1. [제어판] → [전원 옵션] → [전원 단추 작동 설정]
- 2. 현재 사용할 수 없는 설정 변경 활성화
- 3. 빠른 시작 켜기(권장)을 해제



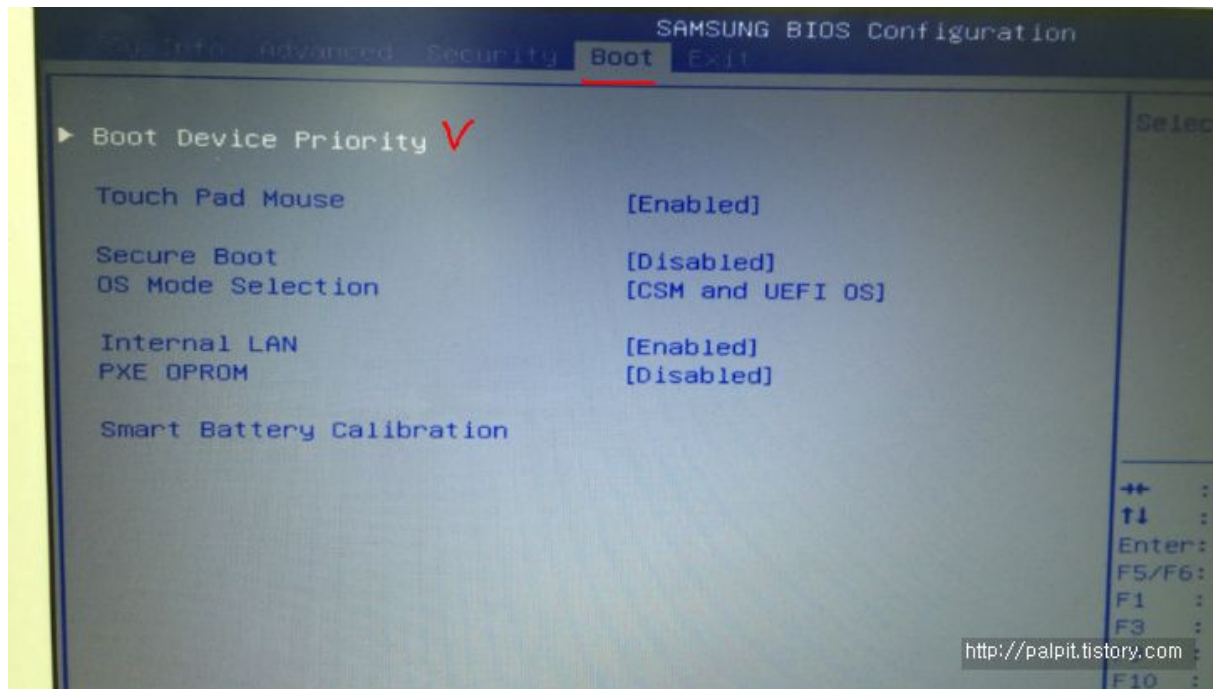
# (Appendix) Ubuntu 16.04 Installation

- 준비 - USB Installer 만들기
  - [Universal USB Installer](#) 다운로드
  - USB 플래시 메모리 삽입 후 프로그램 실행
  - Linux distribution 'Ubuntu' 선택 → .iso file 선택 → USB 드라이브 선택
  - Create! (복사 및 설정 시간 소요)



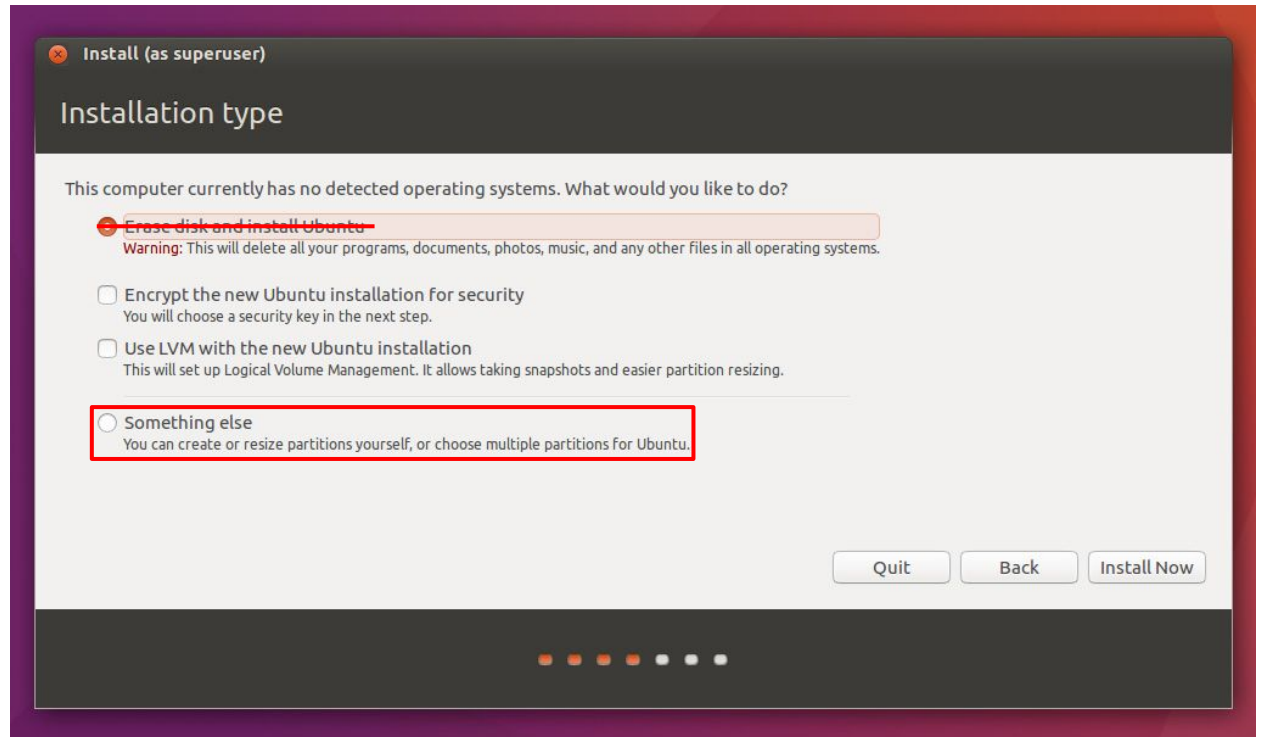
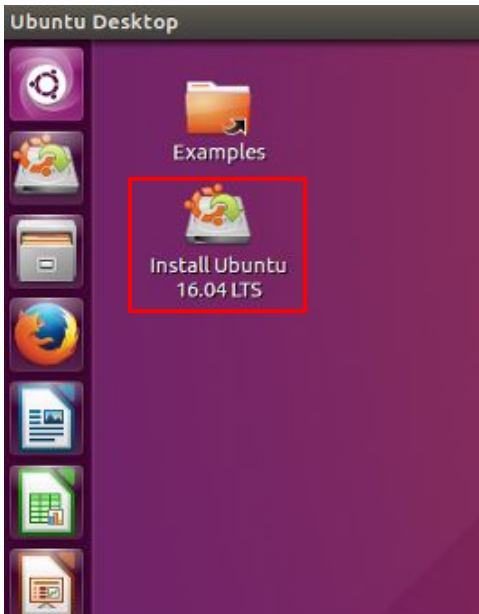
# (Appendix) Ubuntu 16.04 Installation

- Ubuntu 설치 - 0. BIOS 세팅
  - BIOS 메뉴 진입: 재부팅 시 F2키 또는 Delete키 입력
  - Boot 메뉴에 **Secure Boot** 항목이 있고 Enabled 이면 Disabled
  - **Boot Device Priority**에서 삽입된 USB의 우선순위를 1번으로 설정
  - Save Changes and Reset
    - BIOS 인터페이스 및 메뉴 순서는 다를 수 있음



## (Appendix) Ubuntu 16.04 Installation

- Ubuntu 설치 - 1. 설치 초기
  1. USB 부팅 시 뜨는 바탕화면에서 **Install Ubuntu 16.04.2 LTS** 선택
  2. 설치언어는 영어(English) 권장
  3. [Download updates while installing] [Install this third-party software] 선택 해제
  4. Installation Type에서 [Something else] 선택





# (Appendix) Ubuntu 16.04 Installation

- Ubuntu 설치 - 2. 파티션 할당

1. Swap 영역 할당

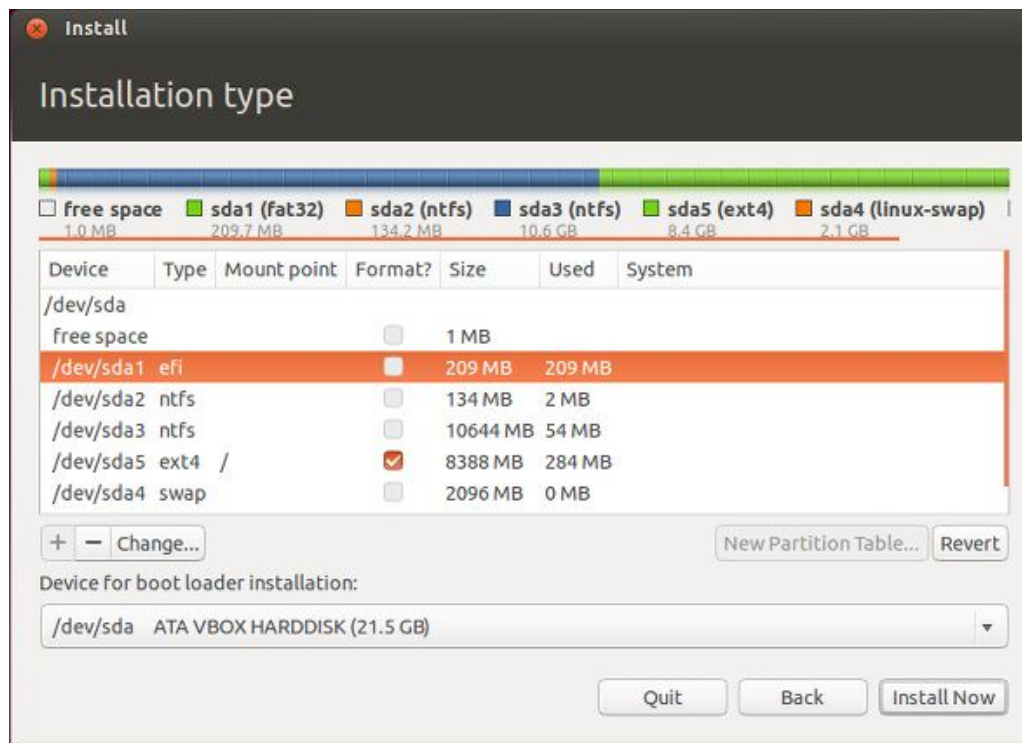
free space 선택 후 (+) 버튼을 눌러 파티션을 다음과 같이 설정 후 OK

Size: 8196MB(=RAM과 같은 크기), Use as: [swap area]

2. Home 영역 할당

Size: 나머지 전체, Use as: [Ext4 journaling file system], Mount point: [/]

3. Boot loader location: /dev/sda (그대로; 함께 설치되는 하드디스크 최상위)



# (Appendix) Ubuntu 16.04 Installation

- Ubuntu 설치 - 3. 기타 설정 및 설치
  1. Location: Seoul
  2. Keyboard layout: Korean - Korean
  3. 사용자/컴퓨터명 할당, 암호 설정
    - 이때 입력한 암호가 sudo의 password
  4. 설치 및 완료: 설치가 끝났다면 재부팅. 곧바로 USB는 제거
    - 부팅 시 부트로더(grub)에서 운영체제(윈도우/우분투) 선택

