ROS

<ROS Melodic 설치>

Link: http://wiki.ros.org/melodic/Installation/Ubuntu

먼저, Alt+Tab 을 눌러 Terminal을 실행시킨다.

1. **Setup your computer to accept software from packages.ros.org.**

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

2. Set up your keys

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

3. Debian package update

```
sudo apt update
```

4. ROS-Melodic FULL INSTALL (ROS, rqt, RViz, robot-generic libraries, etc)

```
sudo apt install ros-melodic-desktop-full
```

5. Environment Setup: Add ROS environment variable to bash session

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

6. Dependencies for building packages(Recommended)

```
sudo apt install python-rosdep python-rosinstall python-rosinstall-generator python-wstool build-essential
```

```
sudo apt install python-rosdep
```

```
sudo rosdep init
```

```
rosdep update
```

<catkin_ws 만들기>

1. **catkin_ws 폴더 생성 및 하위 폴더 src 생성**

```
mkdir -p catkin_ws/src
```

2. catkin_ws 폴더 내로 directory 설정

```
cd catkin_ws
```

3. catkin work space로 build

```
catkin_make
```

ROS-melodic 의 경우는 default python version이 2.7입니다. 따라서 python 3.x 버전을 사용하기 위해서는 python 환경을 설정 해줘야 합니다.

<Python Environment Manager Installation Guide> 는 pyenv를 활용한 환경 설정 가이드입니다.

<Python Environment Manager Installation Guide>

가상환경과는 별개로 python의 버전을 변경할 수 있는 툴로는 `pyenv`와 `conda` 가 있습니다. Version management와 virtual environmanagement는 두 가지 툴 모두 지원하지만 `pyenv`가 시스템 환경에 덜 침습적이고 minimal한 성격을 가지므로 여기서 `pyenv` 를 기준으로 합니다. (맥에서는 homebrew를 통한 설치도 가능합니다)

1. **기본적인 Python development tool을 설치**

```
sudo apt install python3-dev python3-pip
```

2. 설치에 필요한 패키지를 설치해줍니다.

```
sudo apt-get install -y build-essential libssl-dev zlib1g-dev libbz2-dev \
libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev libncursesw5-dev \
xz-utils tk-dev libffi-dev liblzma-dev python-openssl git
```

3. 우선 pyenv installer를 설치합니다.

```
curl https://pyenv.run | bash
```

4. `~/.bashrc` 의 마지막에 다음 내용을 추가합니다.

1) bashrc 열기

```
gedit ~/.bashrc
```

2) **밑에 그냥 복붙하지 마세요!** 사용자의 user name이 있습니다 설치 후 console에 나온 메시지를 복붙하셔야 합니다.

(아래의 커맨드는 pyenv를 path에 등록해주고 auto completion을 사용할 수 있게 해줍니다)

```
export PATH="/home/[YOUR USER NAME]/.pyenv/bin:$PATH"
eval "$(pyenv init -)"
eval "$(pyenv virtualenv-init -)"
```

3) `bashrc` 를 반영해주기 위해 shell을 다시 실행해 줍니다.

```
exec "$SHELL"
```

이제 `pyenv` 를 입력했을때 다음과 같은 command 안내가 나오면 정상적으로 설치된 것입니다.

```
pyenv
pyenv 1.2.24
Usage: pyenv <command> [<args>]

Some useful pyenv commands are:
   --version   Display the version of pyenv
   activate    Activate virtual environment
   commands    List all available pyenv commands
   deactivate   Deactivate virtual environment
   doctor      Verify pyenv installation and development tools to build pythons.
   exec        Run an executable with the selected Python version
   global      Set or show the global Python version(s)
   help        Display help for a command
   hooks       List hook scripts for a given pyenv command
   init        Configure the shell environment for pyenv
   install     Install a Python version using python-build
   local       Set or show the local application-specific Python version(s)
   prefix      Display prefix for a Python version
   rehash      Rehash pyenv shims (run this after installing executables)
   root        Display the root directory where versions and shims are kept
   shell       Set or show the shell-specific Python version
   shims       List existing pyenv shims
   uninstall   Uninstall a specific Python version
   version     Show the current Python version(s) and its origin
   version-file   Detect the file that sets the current pyenv version
   version-name   Show the current Python version
   version-origin   Explain how the current Python version is set
   versions    List all Python versions available to pyenv
   virtualenv   Create a Python virtualenv using the pyenv-virtualenv plugin
   virtualenv-delete   Uninstall a specific Python virtualenv
   virtualenv-init   Configure the shell environment for pyenv-virtualenv
   virtualenv-prefix   Display real_prefix for a Python virtualenv version
   virtualenvs   List all Python virtualenvs found in `$PYENV_ROOT/versions/*'.
   whence      List all Python versions that contain the given executable
   which       Display the full path to an executable

See `pyenv help <command>' for information on a specific command.
For full documentation, see: https://github.com/pyenv/pyenv#readme
```

5. 원하는 Python version을 설치해 봅시다.

Pyenv는 시스템 python과 별개로 python의 version을 관리해 줍니다. 설치할 수 있는 python의 version을 보려면 다음을 사용하면 됩니다.

```
pyenv install --list
```

예를 들어 Python 3.8을 설치한다면 `pyenv install 3.8` 까지만 치고 Tab을 하면 가능한 모든 `3.8.x` 버전을 모두 보여줍니다. `3.8.8` 을 설치하는 상황이라면 다음과 같이 입력합니다.

```
pyenv install 3.8.8
```

삭제는 `uninstall` 커맨드를 사용하면 됩니다. Virtual environment도 동일하게 삭제할 수 있습니다.

```
pyenv uninstall 3.8.8
```

설치된 모든 version은 `pyenv versions` 를 통해 볼 수 있습니다.

```
pyenv versions
* system (set by /home/jihoon/.pyenv/version)
  3.8.8
```

6. 이제 virtual environment를 만들어 봅시다. 목표는 virtual environment를 만들고 프로젝트 경로에 들어가기만 하면 해당 프로젝트의 환경을 자동으로 활성화 하는 것입니다.

우선 프로젝트 경로에 들어갑니다. 보통 `git clone` 한 뒤의 프로젝트 폴더에 들어간 상태에서 시작하면 됩니다.

`3.8.8` 버전으로 가상환경의 이름을 `lab` 으로 만든다면 다음과 같이 입력하면 됩니다.

```
pyenv virtualenv 3.8.8 lab
```

이제 pyenv가 관리하는 `3.8.8` 버전의 Python에서 `lab` 이라는 virtual environment가 만들어 졌습니다.

프로젝트 위치에 들어오면 자동으로 해당 환경을 활성화 하기위해 다음과 같이 설정해 줍니다.

```
pyenv local lab
```

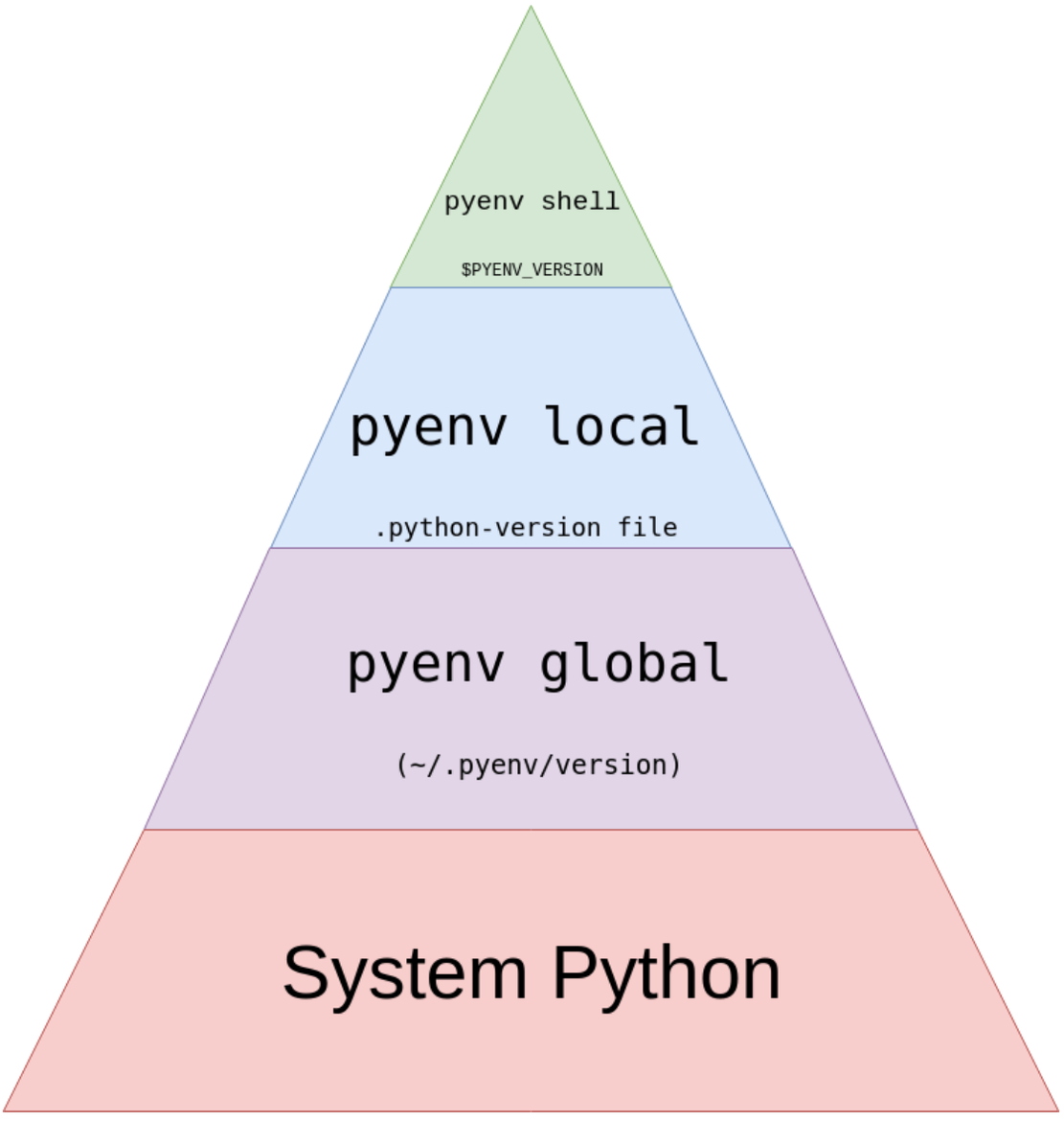
직접 `pyenv activate lab` 을 해주어도 좋지만 위와 같이 자동으로 활성화되게 해주어 패키지가 꼬이는 상황을 예방해 봅시다.

Appendix

Pyenv의 장점은 환경을 원하는 대로 자유롭게 구성할 수 있는 것이지만 처음에는 관리 구조가 헷갈릴 수 있습니다. 이 때 아래 그림을 보시면 이해하기가 수월합니다.

System Python은 Ubuntu에서 기본적으로 제공하는 Python Version입니다. 가급적이면 System Python은 시스템 전반에 반영할 목적이 아니라면 직접 사용하는 것은 지양하는 것이 좋습니다.

`pyenv global` 을 사용하면 virtualenv를 사용하지 않는 상황에서 적용되는 Python 환경입니다. System Python이 기본적으로 `pyenv global` 가 적용됩니다. 모든 python 환경에 동일하게 적용하지 않는 한 `pyenv local` 로 환경을 분명하게 구분해주는 것이 좋습니다.


pyenv shell
$PYENV_VERSION
pyenv local
.python-version file
pyenv global
(~/.pyenv/version)
System Python