1. Python 설치 전 작업
   $ sudo apt update
   $ sudo apt upgrade
   $ sudo apt dist-upgrade
   $ sudo apt install build-essential zlib1g-dev libncurses5-dev libgdbm-dev libnss3-dev libssl-dev libreadline-dev libffi-dev libsqlite3-dev wget libbz2-dev pkg-config

2. Python 3.11.x Installation
   -refer to : https://computingforgeeks.com/how-to-install-python-on-ubuntu-linux/
   -https://www.python.org/downloads/release/python-3111/
   1)Method 1 – Install Python 3.11 on Ubuntu from deadsnakes PPA
      -The deadsnakes PPA provides the simplest method to install Python 3.11 on Ubuntu 22.04|20.04|18.04. It also enables users to receive continued updates, bug fixes, and security updates. A big thumbs up to this custom PPA!
      ① Install the required dependency packages:
         $ sudo apt install software-properties-common -y
      ② Add the deadsnakes PPA to the APT package manager sources list.
         $ sudo add-apt-repository ppa:deadsnakes/ppa
      ③ Press Enter to continue. Once the PPA has been added, you can install Python 3.11 on Ubuntu using the commands:
         $ sudo apt install python3.11
      ④Verify the installation:
         $ python3.11 --version
         Python 3.11.1

   2)Method 2 – Install Python 3.11 on Ubuntu from source.
      -This is an alternative method of installing Python 3.11 on Ubuntu Linux system. With this method, you are guaranteed the latest Python Version. The only problem with this method is that you will not be able to receive continued updates, bug fixes, and security updates through the APT package manager.
      ①Begin by installing the packages required to build Python 3.11 from the source.
         $ sudo apt update

      ②Once the packages have been installed, download the latest available Python 3.11 gzipped tarball from the Python official release page.
      ③You can still download the tarball using wget:
         $ wget https://www.python.org/ftp/python/3.11.2/Python-3.11.2.tgz

      ④Extract the downloaded archive:
         $ tar -xf Python-3.11.*.tgz

      ⑤Navigate into the directory:
         $ cd Python-3.11.*/

      ⑥We will check if the required dependencies are met and optimize the binary with the command:
         $ ./configure --enable-optimizations

      ⑦Once checked, start the build process with the command below. We have used -j to provide the number of cores available on the system, this makes the build process faster:
         $ make -j $(nproc)
         Sample Output:
         ...
         ...
         make[1]: Leaving directory '/home/ubuntu/Python-3.11.2'

         -Building Python from source can take a long time, depending on your system. You can adjust the number passed to the -j option in the make command to specify the number of parallel build jobs to run, which can speed up the build process.

      ⑧Once the build process is complete, install Python 3.11 with the command:
         $ sudo make altinstall
         -We have used altinstall is used instead of install to keep the default Python binary path in /usr/bin/python.
         Sample Output :
         ...
         ...
         Successfully installed pip-22.3.1 setuptools-65.5.0
         WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv

      ⑨Once complete, check the version:
         $ python3.11 --version
         Python 3.11.2

3. Change from default to alternative python version
   1)Find python3.11
      $ which python3.11
      /usr/local/bin/python3.11

   2)Method1
      -To change a python version on per user basis, simply create an alias within user's home directory. Open ~/.bashrc file and add new alias to change your default python executable:
         $ sudo nano ~/.bashrc
            ...
            ...

alias python='/usr/local/bin/python3.11'
-Save Ctrl + O and Exit Ctrl + X
-Once you make the above change, re-login or source your .bashrc file:
$. ~/.bashrc

3)Method2
$ sudo update-alternatives --install /usr/bin/python python /usr/local/bin/python3.11 1

Sample Output :
update-alternatives: using /usr/local/bin/python3.11 to provide /usr/bin/python (python) in auto mode

4)Check your default python version:
$ python --version
Python 3.11.2

4. Install Python Extensions on Ubuntu 22.04 LTS
-Python modules are important since they add functionality to Python. These modules can be installed using the Python Package manager (PIP).
1)Install PIP on Ubuntu 22.04|20.04|18.04:
$ sudo apt install python3-pip

2)Now use PIP to install any module. The command to use has the syntax:
$ sudo pip install module-name

3)You can list all the installed modules using the command;
$ pip list

4)pip update
$ pip install --upgrade pip

5. Installing supporting packages for Python 3.11.
1)Python projects often need a bunch of other packages to work correctly. These supporting packages include python-dev, virtualenv, distutils. You must install them with your new upgrade of Python. The following command will install them.
$ sudo apt install python3.11-dev python3.11-venv python3.11-distutils python3.11-gdbm python3.11-tk python3.11-lib2to3

2)Here's an explanation of what they are and why you need them as explained in deadsnakes PPA description.
-python3.11-dev: includes development headers for building C extensions.
-python3.11-venv: provides the standard library 'venv' module.
-python3.11-distutils: provides the standard library distutils module.
-python3.11-lib2to3: provides the 2to3-3.11 utility and the standard library lib2to3 module.
-python3.11-gdbm: provides the standard library dbm.gnu module.
-python3.11-tk: provides the standard library tkinter module.

6. Python IDLE Installation <---Command Line에서는 불가
1)Terminal에서
$ sudo apt-get install idle-python3.11

2)설치확인, Terminal에서
$ idle-python3.11.1

7. virtualenv Installation
1)가상환경 설치하기
$ sudo pip3 install virtualenv

2)가상환경 만들기
-반드시 로그인계정 홈디렉토리에서 실행할 것 ex)/home/ubuntu
$ mkdir PythonHome
$ cd PythonHome
$ virtualenv --python=python3 myenv

3)가상환경 들어가기
$ source myenv/bin/activate
(myenv) ....

4)가상환경 나오기
(myenv) .... ~/PythonHome$ deactivate

8. Visual Studio Code for Ubuntu <--- 설치 안함
1)https://code.visualstudio.com/Download
2)Select .deb 64 bit
3)Download 후, Downloads dirctory로 이동한 후 설치하기
$ sudo dpkg -i code*.deb

4)테스트할 PythonHome으로 이동

$ cd ~/PythonHome
$ code .

5)test.py 파일 생성
6)python Extension 설치
-Extension 검색창에서 python으로 검색
-목록에서 다음의 extension 설치
--Python
--Python for VSCode
--Python Extension Pack

7)test.py에서
-Linter pylint is not installed
--[Install] click
-IntelliCode Python support requires you to use the Microsoft Python...
--[Enable it and Reload Window] Click

9. Anaconda Installation <---설치 안함.
1)https://www.anaconda.com/products/distribution
2)Click [Linux]
3)Anaconda3 2022.10 for Linux Installer
-Python 3.9 version
-Click '64-Bit(x86) Installer (737 MB)'

$ cd Downloads
$ bash Ana*.sh
-마지막 질문에 no로 할 것

4)~/.bashrc 수정
$ gedit ~/.bashrc
-제일 아래로 이동하여 아래의 코드를 삽입후, 저장한 다음 창을 닫는다.
export PATH="$PATH:/home/username/anaconda3/bin"

$ source ~/.bashrc
$ conda -V
---
conda 4.7.10

$ conda config --set auto_activate_base False

10. Jupyter Notebook Installation
1)Anaconda 설치했다면
-Terminal에서
$ conda install jupyter

2)Anaconda 설치하지 않았다면
$ cd PythonHome
$ source myenv/bin/activate
(myenv) ..../PythonRoom $ pip install jupyter
(myenv) ..../PythonRoom $ jupyter notebook
...
...
Ctrl + C
Shutdown this notebook server (y/[n])? y

11. Jupyter notebook 원격 접속 서버 설정하기
1)Ubuntu에서 포트 방화벽 해제하기
-Terminal에서
$ sudo ufw allow 8888

or AWS에서 EC2를 생성했다면 보안 그룹 Inbound Rule 설정
--유형 : 사용자 지정 TCP
--프로토콜 : TCP
--포트범위 : 8888
--소스 : 0.0.0.0/0

2)config 파일 만들기
-Terminal에서
(myenv) ..../PythonRoom$ jupyter notebook --generate-config
-위 코드를 실행하면 /home/ubuntu/.jupyter directory에 jupyter_notebook_config.py 파일이 생성된다.

3)Server 비밀번호 생성
(myenv) ..../PythonRoom$ ipython

-위 코드를 실행하면 열리는 Ipython prompt 환경에서 아래 코드를 순서대로 실행한다.
-주의할점은 "Enter password:" 에 사용할 비밀번호를 입력할 때 타이핑을 해도 화면에는 커서가 바뀐다던지, 입력한 비밀번호가 터미널에 표시된다던지 하는 반응이 전혀 없다.

ln [1]: from jupyter_server.auth import passwd
ln [2]: passwd()
Enter password:
Verify password:

Out[2]:
'argon2:$argon2id$v=19$m=10240,t=10,p=8$xkydTO+sORmMAM3II4/tIw$LGAynvTWYUjD8UxaPgBPbxU6TTXczUMnx1fdcnAGW5I'
비밀번호를 암호화 하여 반환. 당연히 위와 다름.

-위의 암호화된 비밀번호 복사한다. ipython 환경을 빠져나온다.
In [3] : exit()

4)Jupyter Server 환경설정하기
-/home/ubuntu/.jupyter directory에 가서 jupyter_notebook_config.py 파일을 연다.

(myenv) ..../PythonRoom$ nano -c /home/ubuntu/.jupyter/jupyter_notebook_config.py

-jupyter_notebook_config.py 파일은 Jupyter Notebook 환경설정이 저장되어 있는 파일인데, 모든 환경설정들이 전부 # c.Notebook.App.ip='' 이런 식으로 앞에 # 이 붙어서 주석처리 되어 있다.
-아래코드처럼 수정하는 부분은 앞에 #을 빼서 주석이 아닌 일반 코드화 시킨다.
-jupyter_notebook_config.py 를 열고 제일 위에 아래 코드를 넣는다.
c = get_config()

-외부 접속 허용하기
c.NotebookApp.allow_origin = '*' # 136line

-작업경로 설정(ex:/home/ubuntu/PythonHome)
c.NotebookApp.notebook_dir = '원하는/작업경로를/입력' # 450line

-비밀번호 설정(위에서 복사해둔 암호화된 비밀번호 여기에 입력) # 469line
c.NotebookApp.password = u'argon2:$argon2id$v=19$m=10240,t=10,p=8$0JmxWY2OaukX8YSoBF+Drg$ExRWFK+U5MgTiiiwIM3r61krJOpY89HTGW12wNp9GbI'

-시작시 브라우저 실행여부 # 458line
c.NotebookApp.open_browser = False # 서버로 실행될때 서버PC에서 주피터 노트북 창이 새로 열릴 필요가 없다.

-수정이 완료됐으면 jupyter_notebook_config.py 저장.
-Text Editor를 닫는다.
-Terminal을 닫는다.

5)Jupyter Server 시작하기
-Terminal에서
(myenv) ..../PythonRoom$ jupyter notebook –ip=0.0.0.0 –port=8888 –-allow-root

-서버가 실행되었다.
-이제 브라우저에서 주소창에
http://localhost:8888

-패스워드를 넣고 원격으로 jupyter notebook으로 로그인한다.