

```

1 1. Python 설치 전 작업
2 $ sudo apt update
3 $ sudo apt upgrade
4 $ sudo apt dist-upgrade
5 $ 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
6
7
8
9 2. Python 3.11.x Installation
10 -refer to : https://computingforgeeks.com/how-to-install-python-on-ubuntu-linux/
11 -https://www.python.org/downloads/release/python-3111/
12 1)Method 1 – Install Python 3.11 on Ubuntu from deadsnakes PPA
13 -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!
14 ① Install the required dependency packages:
15 $ sudo apt install software-properties-common -y
16 ② Add the deadsnakes PPA to the APT package manager sources list.
17 $ sudo add-apt-repository ppa:deadsnakes/ppa
18 ③ Press Enter to continue. Once the PPA has been added, you can install Python 3.11 on Ubuntu using the commands:
19 $ sudo apt install python3.11
20 ④Verify the installation:
21 $ python3.11 --version
22 Python 3.11.1
23
24 2)Method 2 – Install Python 3.11 on Ubuntu from source.
25 -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.
26 ①Begin by installing the packages required to build Python 3.11 from the source.
27 $ sudo apt update && 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 -y
28
29 ②Once the packages have been installed, download the latest available Python 3.11 gzipped tarball from the Python official
  release page.
30 ③You can still download the tarball using wget:
31 $ wget https://www.python.org/ftp/python/3.11.1/Python-3.11.1.tgz
32
33 ④Extract the downloaded archive:
34 $ tar -xf Python-3.11.*.tgz
35
36 ⑤Navigate into the directory:
37 $ cd Python-3.11.*/
38
39 ⑥We will check if the required dependencies are met and optimize the binary with the command:
40 $ ./configure --enable-optimizations
41
42 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:
43 $ make -j $(nproc)
44 Sample Output:
45 ...
46 ...
47 make[1]: Leaving directory '/home/ubuntu/Python-3.11.1'
48
49 -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.
50
51 ⑧Once the build process is complete, install Python 3.11 with the command:
52 $ sudo make altinstall
53 -We have used altinstall is used instead of install to keep the default Python binary path in /usr/bin/python.
54 Sample Output :
55 ...
56 ...
57 Successfully installed pip-22.3.1 setuptools-65.5.0
58 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
59
60 ⑨Once complete, check the version:
61 $ python3.11 --version
62 Python 3.11.1
63
64
65
66 3. Change from default to alternative python version
67 1)Find python3.11
68 $ which python3.11
69 /usr/local/bin/python3.11
70
71 2)Method1
72 -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:
73 $ sudo nano ~/.bashrc
74 ...

```

```

75     ...
76     alias python='/usr/local/bin/python3.11'
77 -Save Ctrl + O and Exit Ctrl + X
78 -Once you make the above change, re-login or source your .bashrc file:
79 $ . ~/.bashrc
80
81 3)Method2
82 $ sudo update-alternatives --install /usr/bin/python python /usr/local/bin/python3.11 1
83
84 Sample Output :
85 update-alternatives: using /usr/local/bin/python3.11 to provide /usr/bin/python (python) in auto mode
86
87 4)Check your default python version:
88 $ python --version
89 Python 3.11.1
90
91
92
93 4. Install Python Extensions on Ubuntu 22.04 LTS
94 -Python modules are important since they add functionality to Python. These modules can be installed using the Python
Package manager (PIP).
95 1)Install PIP on Ubuntu 22.04|20.04|18.04:
96 $ sudo apt install python3-pip
97
98 2)Now use PIP to install any module. The command to use has the syntax:
99 $ sudo pip install module-name
100
101 3)You can list all the installed modules using the command;
102 $ pip list
103
104 4)pip update
105 $ pip install --upgrade pip
106
107
108 5. Installing supporting packages for Python 3.11.
109 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.
110 $ sudo apt install python3.11-dev python3.11-venv python3.11-distutils python3.11-gdbm python3.11-tk
python3.11-lib2to3
111
112 2)Here's an explanation of what they are and why you need them as explained in deadsnakes PPA description.
113 -python3.11-dev: includes development headers for building C extensions.
114 -python3.11-venv: provides the standard library 'venv' module.
115 -python3.11-distutils: provides the standard library distutils module.
116 -python3.11-lib2to3: provides the 2to3-3.11 utility and the standard library lib2to3 module.
117 -python3.11-gdbm: provides the standard library dbm.gnu module.
118 -python3.11-tk: provides the standard library tkinter module.
119
120
121
122 6. Python IDLE Installation <---Command Line에서는 불가
123 1)Terminal에서
124 $ sudo apt-get install idle-python3.11
125
126 2)설치확인, Terminal에서
127 $ idle-python3.11.1
128
129
130
131 7. virtualenv Installation
132 1)가상환경 설치하기
133 $ sudo pip3 install virtualenv
134
135 2)가상환경 만들기
136 $ mkdir PythonHome
137 $ cd PythonHome
138 $ virtualenv --python=python3 myenv
139
140 3)가상환경 들어가기
141 $ source myenv/bin/activate
142 (myenv) ....
143
144 4)가상환경 나오기
145 (myenv) .... ~/PythonHome$ deactivate
146
147
148
149 8. Visual Studio Code for Ubuntu <--- 설치 안함
150 1)https://code.visualstudio.com/Download
151 2)Select .deb 64 bit
152 3)Download 후, Downloads directory로 이동한 후 설치하기
153 $ sudo dpkg -i code*.deb
154
155 4)테스트할 PythonHome으로 이동

```

```

156 $ cd ~/PythonHome
157 $ code .
158
159 5)test.py 파일 생성
160 6)python Extension 설치
161   -Extension 검색창에서 python으로 검색
162   -목록에서 다음의 extension 설치
163     --Python
164     --Python for VSCode
165     --Python Extension Pack
166
167 7)test.py에서
168   -Linter pylint is not installed
169     --[Install] click
170   -IntelliCode Python support requires you to use the Microsoft Python...
171     --[Enable it and Reload Window] Click
172
173
174
175 9. Anaconda Installation <---설치 안함.
176 1)https://www.anaconda.com/products/distribution
177 2)Click [Linux]
178 3)Anaconda3 2022.10 for Linux Installer
179   -Python 3.9 version
180   -Click '64-Bit(x86) Installer (737 MB)'
181
182 $ cd Downloads
183 $ bash Ana*.sh
184   -마지막 질문에 no로 할 것
185
186 4)~/.bashrc 수정
187 $ gedit ~/.bashrc
188   -제일 아래로 이동하여 아래의 코드를 삽입후, 저장한 다음 창을 닫는다.
189     export PATH="$PATH:/home/username/anaconda3/bin"
190
191 $ source ~/.bashrc
192 $ conda -V
193 -----
194 conda 4.7.10
195
196 $ conda config --set auto_activate_base False
197
198
199
200 10. Jupyter Notebook Installation
201 1)Anaconda 설치했다면
202   -Terminal에서
203     $ conda install jupyter
204
205 2)Anaconda 설치하지 않았다면
206 $ cd PythonHome
207 $ source myenv/bin/activate
208 (myenv) ....PythonRoom $ pip install jupyter
209 (myenv) ....PythonRoom $ jupyter notebook
210 ...
211 ...
212 Ctrl + C
213 Shutdown this notebook server (y/[n])? y
214
215
216
217 11. Jupyter notebook 원격 접속 서버 설정하기
218 1)Ubuntu에서 포트 방화벽 해제하기
219   -Terminal에서
220     $ sudo ufw allow 8888
221
222   or AWS에서 EC2를 생성했다면 보안 그룹 Inbound Rule 설정
223     --유형 : 사용자 지정 TCP
224     --프로토콜 : TCP
225     --포트범위 : 8888
226     --소스 : 0.0.0.0/0
227
228 2)config 파일 만들기
229   -Terminal에서
230     (myenv) ....PythonRoom$ jupyter notebook --generate-config
231   -위 코드를 실행하면 /home/ubuntu/.jupyter directory에 jupyter_notebook_config.py 파일이 생성된다.
232
233 3)Server 비밀번호 생성
234 (myenv) ....PythonRoom$ ipython
235
236   -위 코드를 실행하면 열리는 Ipython prompt 환경에서 아래 코드를 순서대로 실행한다.
237   -주의할점은 "Enter password:" 에 사용할 비밀번호를 입력할 때 타이핑을 해도 화면에는 커서가 바뀐다던지, 입력한 비밀번호가 터미널에 표시된다던지 하는
   반응이 전혀 없다.
238

```

```

239 In [1]: from notebook.auth import passwd
240 In [2]: passwd()
241 Enter password:
242 Verify password:
243
244 Out[2]:
'argon2:$argon2id$v=19$m=10240,t=10,p=8$xydT0+sORmMAM3II4/tIw$LGAYnvTWYUjD8UxaPgBPbxU6TTXczUMnx1f
dcnAGW5I'
# 비밀번호를 암호화 하여 반환. 당연히 위와 다름.

245
246
247 -위의 암호화된 비밀번호 복사한다. ipython 환경을 빠져나온다.
248 In [3] : exit()
249
250

```

4)Jupyter Server 환경설정하기

-/home/ubuntu/.jupyter directory에 가서 jupyter_notebook_config.py 파일을 연다.

```
(myenv) ..../PythonRoom$ nano -linenumbers /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+U5MgTiiwIM3r61krJOpY89HTGW
12wNp9GbI'
```

-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.