

Lab3. Installation Jupyter Notebook on Ubuntu

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
```

④ Verify 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.

① Begin by installing the packages required to build Python 3.11 from the source.

```
$ sudo apt update
```

② Once the packages have been installed, download the latest available Python 3.11 gzipped tarball from the Python official release page.

③ You can still download the tarball using wget:

```
$ wget https://www.python.org/ftp/python/3.11.2/Python-3.11.2.tgz
```

④ Extract the downloaded archive:

```
$ tar -xf Python-3.11.*.tgz
```

⑤ Navigate into the directory:

```
$ cd Python-3.11.*/
```

⑥ We will check if the required dependencies are met and optimize the binary with the command:

```
$ ./configure --enable-optimizations
```

⑦ Once 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.

⑧ Once 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>

⑨ Once 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) Method 1

-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
```

```

76     ...
77     ...
78     alias python='/usr/local/bin/python3.11'
79 -Save Ctrl + O and Exit Ctrl + X
80 -Once you make the above change, re-login or source your .bashrc file:
81 $ . ~/.bashrc
82
83 3)Method2
84 $ sudo update-alternatives --install /usr/bin/python python /usr/local/bin/python3.11 1
85
86 Sample Output :
87 update-alternatives: using /usr/local/bin/python3.11 to provide /usr/bin/python (python) in auto mode
88
89 4)Check your default python version:
90 $ python --version
91 Python 3.11.2
92
93
94
95 4. Install Python Extensions on Ubuntu 22.04 LTS
96 -Python modules are important since they add functionality to Python. These modules can be installed using the Python
97 Package manager (PIP).
98 1)Install PIP on Ubuntu 22.04|20.04|18.04:
99 $ sudo apt install python3-pip
100
101 2)Now use PIP to install any module. The command to use has the syntax:
102 $ sudo pip install module-name
103
104 3)You can list all the installed modules using the command;
105 $ pip list
106
107 4)pip update
108 $ pip install --upgrade pip
109
110 5. Installing supporting packages for Python 3.11.
111 1)Python projects often need a bunch of other packages to work correctly. These supporting packages include python-dev,
112 virtualenv, distutils. You must install them with your new upgrade of Python. The following command will install them.
113 $ sudo apt install python3.11-dev python3.11-venv python3-distutils python3-gdbm python3-tk python3-lib2to3
114
115 2)Here's an explanation of what they are and why you need them as explained in deadsnakes PPA description.
116 -python3.11-dev: includes development headers for building C extensions.
117 -python3.11-venv: provides the standard library 'venv' module.
118 -python3-distutils: provides the standard library distutils module.
119 -python3-lib2to3: provides the 2to3-3.11 utility and the standard library lib2to3 module.
120 -python3-gdbm: provides the standard library dbm.gnu module.
121 -python3-tk: provides the standard library tkinter module.
122
123
124 6. Python IDLE Installation <---Command Line에서는 불가
125 1)Terminal에서
126 $ sudo apt-get install idle-python3.11
127
128 2)설치확인, Terminal에서
129 $ idle-python3.11.1
130
131
132
133 7. virtualenv Installation
134 1)가상환경 설치하기
135 $ sudo pip3 install virtualenv
136
137 2)가상환경 만들기
138 -반드시 로그인계정 홈디렉토리에서 실행할 것 ex)/home/ubuntu
139 $ virtualenv --python=python3 DevOps
140
141 3)가상환경 들어가기
142 $ source DevOps/bin/activate
143 (DevOps) ....
144
145 4)가상환경 나오기
146 (DevOps) .... ~$ deactivate
147
148
149
150 8. Visual Studio Code for Ubuntu <--- 설치 안함
151 1)https://code.visualstudio.com/Download
152 2)Select .deb 64 bit
153 3)Download 후, Downloads directory로 이동한 후 설치하기
154 $ sudo dpkg -i code*.deb
155
156 4)테스트할 DevOps으로 이동
157 $ source DevOps/bin/activate

```

```

158 (DevOps) .... ~$ code .
159
160 5)test.py 파일 생성
161 6)python Extension 설치
162   -Extension 검색창에서 python으로 검색
163   -목록에서 다음의 extension 설치
164     --Python
165     --Python for VSCode
166     --Python Extension Pack
167
168 7)test.py에서
169   -Linter pylint is not installed
170     --[Install] click
171   -IntelliCode Python support requires you to use the Microsoft Python...
172     --[Enable it and Reload Window] Click
173
174
175
176 9. Anaconda Installation <---설치 안함.
177 1)https://www.anaconda.com/products/distribution
178 2)Click [Linux]
179 3)Anaconda3 2022.10 for Linux Installer
180   -Python 3.9 version
181   -Click '64-Bit(x86) Installer (737 MB)'
182
183 $ cd Downloads
184 $ bash Ana*.sh
185   -마지막 질문에 no로 할 것
186
187 4)~/.bashrc 수정
188   $ gedit ~/.bashrc
189   -제일 아래로 이동하여 아래의 코드를 삽입후, 저장한 다음 창을 닫는다.
190     export PATH="$PATH:/home/username/anaconda3/bin"
191
192   $ source ~/.bashrc
193   $ conda -V
194   -----
195   conda 4.7.10
196
197 $ conda config --set auto_activate_base False
198
199
200
201 10. Jupyter Notebook Installation
202 1)Anaconda 설치했다면
203   -Terminal에서
204     $ conda install jupyter
205
206 2)Anaconda 설치하지 않았다면
207   $ cd PythonHome
208   $ source myenv/bin/activate
209   (myenv) ....PythonRoom $ pip install jupyter
210   (myenv) ....PythonRoom $ jupyter notebook
211   ...
212   ...
213   Ctrl + C
214   Shutdown this notebook server (y/[n])? y
215
216
217
218 11. Jupyter notebook 원격 접속 서버 설정하기
219 1)Ubuntu에서 포트 방화벽 해제하기
220   -Terminal에서
221     $ sudo ufw allow 8888
222
223   or AWS에서 EC2를 생성했다면 보안 그룹 Inbound Rule 설정
224     --유형 : 사용자 지정 TCP
225     --프로토콜 : TCP
226     --포트범위 : 8888
227     --소스 : 0.0.0.0/0
228
229 2)config 파일 만들기
230   -Terminal에서
231     (myenv) ....PythonRoom$ jupyter notebook --generate-config
232   -위 코드를 실행하면 /home/ubuntu/.jupyter directory에 jupyter_notebook_config.py 파일이 생성된다.
233
234 3)Server 비밀번호 생성
235   (myenv) ....PythonRoom$ ipython
236
237   -위 코드를 실행하면 열리는 Ipython prompt 환경에서 아래 코드를 순서대로 실행한다.
238   -주의할점은 "Enter password:" 에 사용할 비밀번호를 입력할 때 타이핑을 해도 화면에는 커서가 바뀐다던지, 입력한 비밀번호가 터미널에 표시된다던지 하는
   반응이 전혀 없다.
239
240   In [1]: from notebook.auth import passwd

```

```

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

246
247
248 -위의 암호화된 비밀번호 복사한다. ipython 환경을 빠져나온다.
249 In [3] : exit()
250
251
252 4)Jupyter Server 환경설정하기
253 -/home/ubuntu/.jupyter directory에 가서 jupyter_notebook_config.py 파일을 연다.
254
255 (myenv) ..../PythonRoom$ nano -c /home/ubuntu/.jupyter/jupyter_notebook_config.py
256
257 -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.
258 -아래코드처럼 수정하는 부분은 앞에 #을 빼서 주석이 아닌 일반 코드로 시킨다.
259 -jupyter_notebook_config.py 를 열고 제일 위에 아래 코드를 넣는다.
260 c = get_config()
261
262 -외부 접속 허용하기
263 c.NotebookApp.allow_origin = '*' # 136line
264
265 -작업경로 설정(ex:/home/ubuntu/PythonHome)
266 c.NotebookApp.notebook_dir = '원하는/작업경로를/입력' # 450line
267
268 -비밀번호 설정(위에서 복사해둔 암호화된 비밀번호 여기에 입력) # 469line
269 c.NotebookApp.password =
u'argon2:$argon2id$v=19$m=10240,t=10,p=8$0JmxWY2OaukX8YSoBF+Drg$ExRWFk+U5MgTiiwIM3r61krJOpy89HTGW
12wNp9GbI'

270
271 -시작시 브라우저 실행여부 # 458line
272 c.NotebookApp.open_browser = False # 서버로 실행될때 서버PC에서 주피터 노트북 창이 새로 열릴 필요가 없다.
273
274 -수정이 완료됐으면 jupyter_notebook_config.py 저장.
275 -Text Editor를 닫는다.
276 -Terminal을 닫는다.
277
278
279 5)Jupyter Server 시작하기
280 -Terminal에서
281 (myenv) ..../PythonRoom$ jupyter notebook -ip=0.0.0.0 -port=8888 --allow-root
282
283 -서버가 실행되었다.
284 -이제 브라우저에서 주소창에
285 http://localhost:8888
286
287 -패스워드를 넣고 원격으로 jupyter notebook으로 로그인한다.

```