

파이썬 프로그램 자동화

김효관 |



파이썬 프로그램 자동화

자동화 방법

- 1 소스코드 생성
- 2 코드 테스트
- 3 실행파일 생성 및 테스트
- 4 자동화 실행 (스케줄러 등록)
- 5 자동화 실행 (스케줄러 등록)

파이썬 프로그램 자동화

자동화 방법

1 코드 정상구동여부 확인

2 패키지 버전 묶음 생성 (requirements.txt)

3 실행가능 개발코드 생성 (테스트 포함 가능)

4 리눅스 서버 실행 (클라우드 또는 On-premise 환경)

5 리눅스 관리 프로그램 설치 (터미널 접속 / 파일전송)

6 개발코드 이관

7 파이썬 및 가상환경 설치 (개발환경과 동일 버전)

8 패키지 설치 (requirements.txt) 활용

9 기타 필요 프로그램 설치 (스킵 가능) *크롬 브라우저 등

10 파이썬 코드 실행 테스트

11 선택 1 - 자동화 (Crontab)

11 선택 2 - 서비스형 설치 (모델 서빙 등)

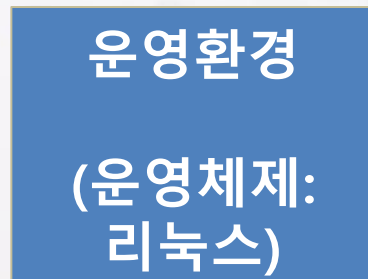
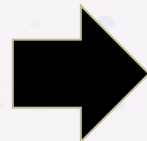
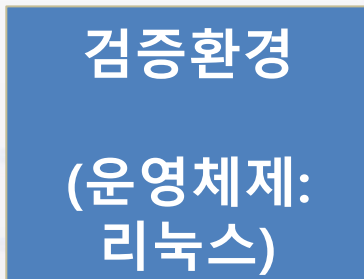
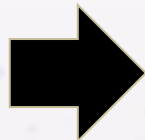
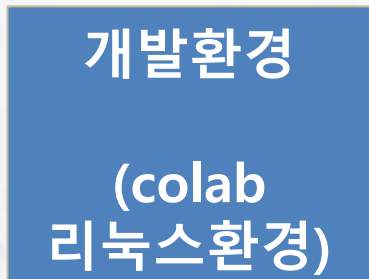
파이썬 프로그램 자동화

기본 컨셉

- 개발 / 검증 / 운영 환경의 이해

소스코드 이관

소스코드 이관



버전 통일

버전 통일

순번	모듈	버전
1	python	3.12
2	pandas	2.2.2
...
30	requests	2.32.3

순번	모듈	버전
1	python	3.12
2	pandas	2.2.2
...
30	requests	2.32.3

순번	모듈	버전
1	python	3.12
2	pandas	2.2.2
...
30	requests	2.32.3

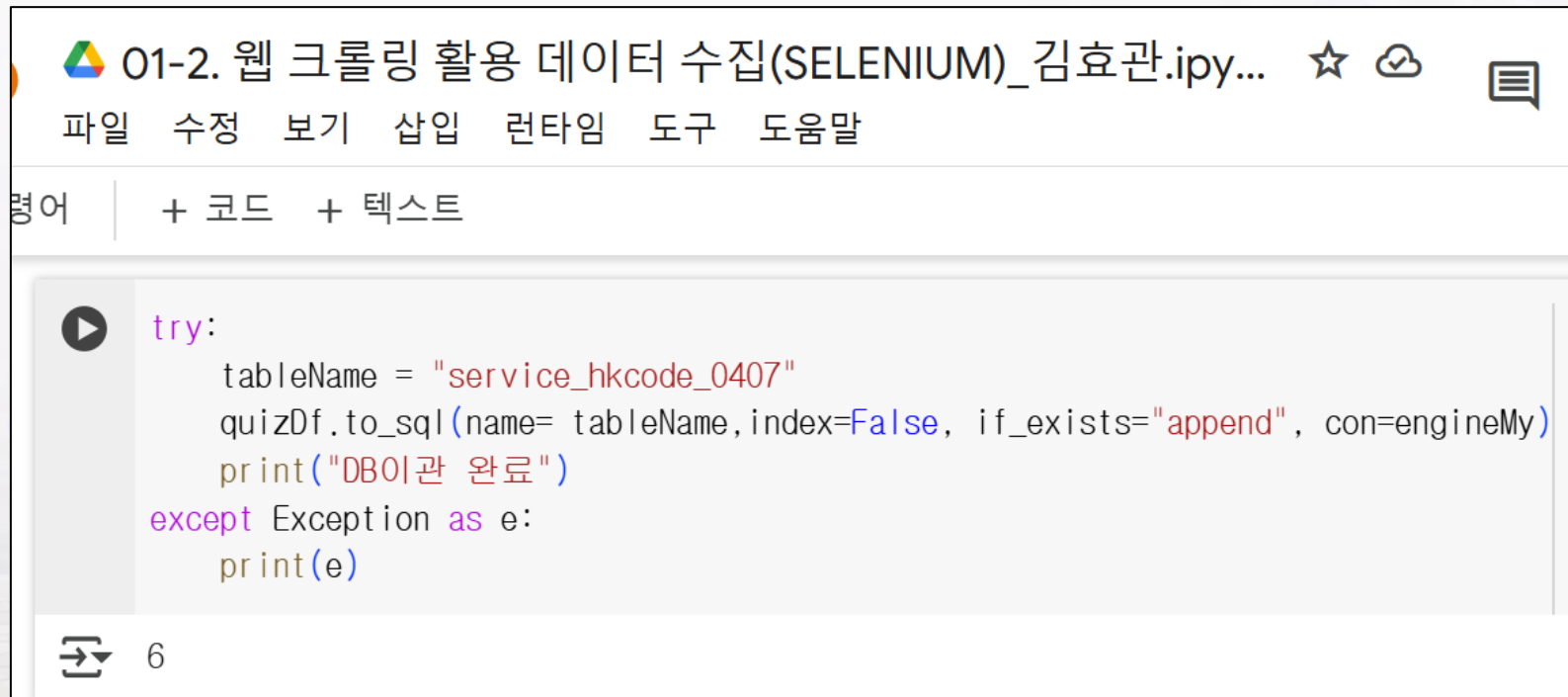
파이썬 프로그램 자동화

파이썬 코드 어떻게 사용할지?



파이썬 프로그램 자동화

1. 코드 정상구동여부 확인



The screenshot shows a Jupyter Notebook window with the title "01-2. 웹 크롤링 활용 데이터 수집(SELENIUM)_김효관.ipyn...". The interface includes a top bar with icons for file operations, a menu bar with options like "파일", "수정", "보기", "삽입", "런타임", "도구", and "도움말", and a toolbar with "+ 코드" and "+ 텍스트" buttons. The main area contains a code cell with a play button icon on the left. The code is a Python snippet that attempts to insert data into a database table. The code is as follows:

```
try:
    tableName = "service_hkcode_0407"
    quizDf.to_sql(name= tableName, index=False, if_exists="append", con=engineMy)
    print("DB이관 완료")
except Exception as e:
    print(e)
```

At the bottom of the code cell, there is a status bar showing a refresh icon and the number "6".

파이썬 프로그램 자동화

2. 패키지 묶음 생성

- 주요 패키지만 별도 추출 하는 함수 또는 `pip freeze >> requirements.txt`

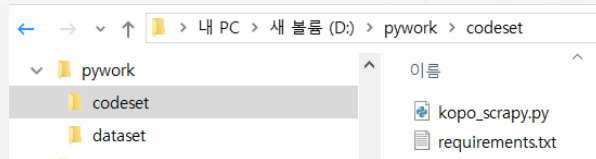
```
import sys, pkg_resources
# 현재 메모리에 로드된 모듈 이름 목록을 리스트로 저장
# sys.modules는 실행 중에 동적으로 import된 모듈까지 포함
mods = list(sys.modules)
# import된 패키지들의 (이름, 버전) 튜플을 저장할 리스트
pkgs = []
# 설치된 모든 패키지 정보 가져오기
dists = list(pkg_resources.working_set)

# 설치된 패키지 중 현재 import된 것만 골라서 pkgs에 추가
for i in range(0, len(dists)):
    name = dists[i].project_name
    key = dists[i].key
    if name in mods or key in mods:
        pkgs.append((name, dists[i].version))

# requirements.txt 파일로 저장
with open('requirements.txt', 'w') as f:
    for i in range(0, len(pkgs)):
        f.write(f"{pkgs[i][0]}=={pkgs[i][1]}\n")
```

requirements.txt

```
1 Bottleneck==1.4.2
2 PyMySQL==1.1.1
3 backcall==0.2.0
4 certifi==2025.1.31
5 chardet==5.2.0
6 cloudpickle==3.1.1
7 cryptography==43.0.3
8 cyclert==0.12.1
9 debugpy==1.8.0
10 decorator==4.4.2
11 defusedxml==0.7.1
12 google==2.0.3
13 greenlet==3.1.1
14 html5lib==1.1
15 httplib2==0.22.0
16 idna==3.10
17 ipykernel==6.17.1
18 kiwisolver==1.4.8
19 lxml==5.3.1
20 matplotlib==3.10.0
21 numexpr==2.10.2
```



파이썬 프로그램 자동화

2. 패키지 묶음 생성

- 주요 패키지만 별도 추출 하는 함수 또는 `pip freeze >> requirements.txt`

requirements.txt X

```
1 Bottleneck==1.4.2
2 PyMySQL==1.1.1
3 backcall==0.2.0
4 certifi==2025.1.31
5 chardet==5.2.0
6 cloudpickle==3.1.1
7 cryptography==43.0.3
8 cyclr==0.12.1
9 debugpy==1.8.0
10 decorator==4.4.2
11 defusedxml==0.7.1
12 google==2.0.3
13 greenlet==3.1.1
14 html5lib==1.1
15 httplib2==0.22.0
16 idna==3.10
17 ipykernel==6.17.1
18 kiwisolver==1.4.8
19 lxml==5.3.1
20 matplotlib==3.10.0
21 numexpr==2.10.2
```

라이브러리 선언

```
[20] # !pip install pandas==2.2.2
```

```
[6] import pandas as pd
    from datetime import datetime
    from sqlalchemy import create_engine, inspect
    # Selenium과 webdriver-manager를 사용한 Chrome 브라우저 실행 코드
    from selenium import webdriver
    from selenium.webdriver.chrome.service import Service
    from webdriver_manager.chrome import ChromeDriverManager
    from selenium.webdriver.common.by import By
    from selenium.webdriver.common.keys import Keys
    import bs4
```

requirements.txt - Windows 메모장

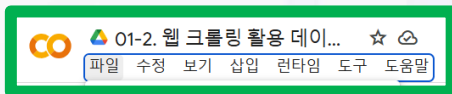
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
PyMySQL==1.1.1
numpy==2.0.2
pandas==2.2.2
requests==2.32.3
selenium==4.31.0
sqlalchemy==2.0.40
webdriver_manager==4.0.2
bs4==0.0.2
```


파이썬 프로그램 자동화

3. 실행가능 개발코드 생성 (테스트 포함 가능) ** Colab환경

• py 파일로 변환 및 코드 폴더 생성

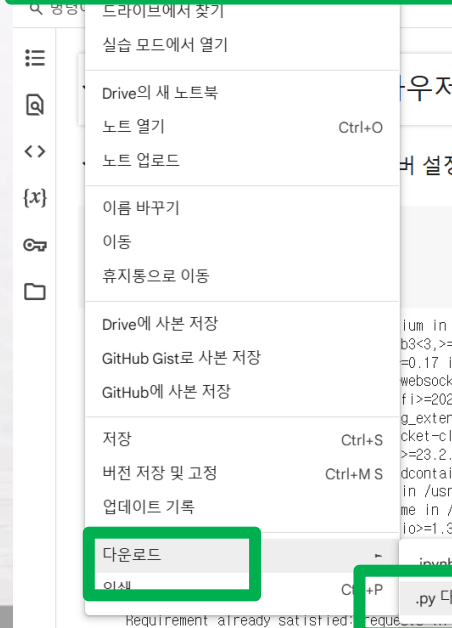


[코드에서 주석처리] *내/외부망 확인
1. 패키지 설치
2. 프로그램 설치
3. 매직명령어 %

```
[ ] # Colab에 필요한 패키지 설치
!pip install selenium
!pip install webdriver-manager
!pip install pymysql
```

```
[ ] # 2. Chrome 브라우저 설치
!apt-get update # 패키지 목록 업데이트
!apt-get install -y wget unzip # wget과 unzip 패키지 설치
!wget -q -O - https://dl-ssl.google.com/linux/linux_signing_key.pub | apt-key add -
!sh -c 'echo "deb [arch=amd64] https://dl.google.com/linux/chrome/deb/ stable main"
!apt-get update # 패키지 목록 다시 업데이트
!apt-get install -y google-chrome-stable # 최신 안정 버전의 Chrome 브라우저 설치
!apt-get install -y fonts-nanum # 한글폰트 설치
```

```
%matplotlib inline
%cd
```

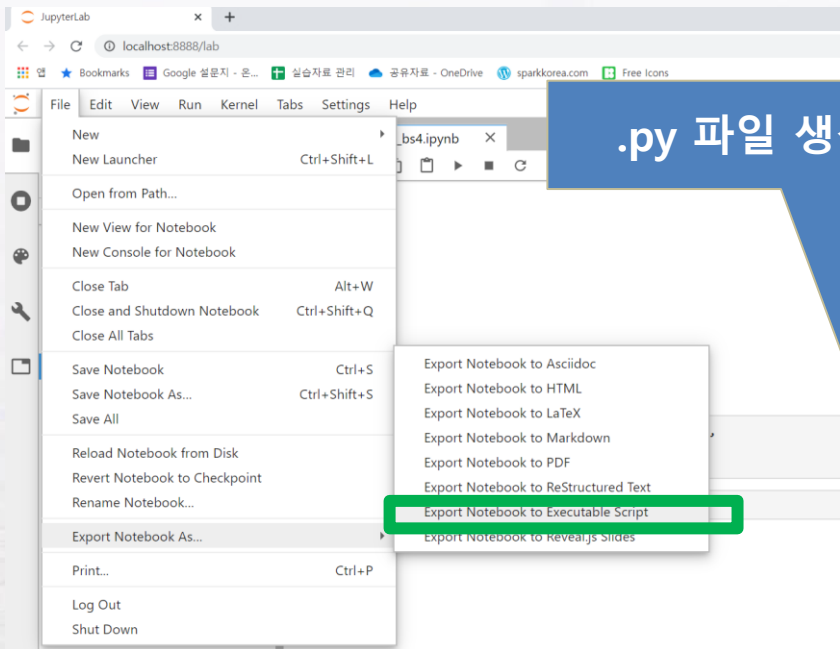


```
um in /usr/local/lib/python3.11/dist-packages (4.30.0)
b3<3,>=1.26 in /usr/local/lib/python3.11/dist-packages (from urllib3[socks]<3,>=1.26->selenium) (2.3.0)
=0.17 in /usr/local/lib/python3.11/dist-packages (from selenium) (0.29.0)
websocket~=0.9 in /usr/local/lib/python3.11/dist-packages (from selenium) (0.12.2)
fi>=2021.10.8 in /usr/local/lib/python3.11/dist-packages (from selenium) (2025.1.31)
g_extensions~=4.9 in /usr/local/lib/python3.11/dist-packages (from selenium) (4.13.0)
cket-client~=1.8 in /usr/local/lib/python3.11/dist-packages (from selenium) (1.8.0)
>=23.2.0 in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (25.3.0)
dcontainers in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (2.4.0)
in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (3.10)
me in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (1.3.0.post0)
io>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (1.3.1)
.11/dist-packages (from trio-websocket~=0.9->selenium) (1.2.0)
b/111/python3.11/dist-packages (from urllib3[socks]<3,>=1.26->selenium) (1.7.1)
.11/dt-packages (from wsproto>=0.14->trio-websocket~=0.9->selenium) (0.14.0)
hon3.11/dist-packages (4.0.2)
dist-packages (from webdriver-manager) (2.32.3)
```

파이썬 프로그램 자동화

3. 실행가능 개발코드 생성 (테스트 포함 가능) ** Jupyter 환경

- py 파일로 변환 및 코드 폴더 생성



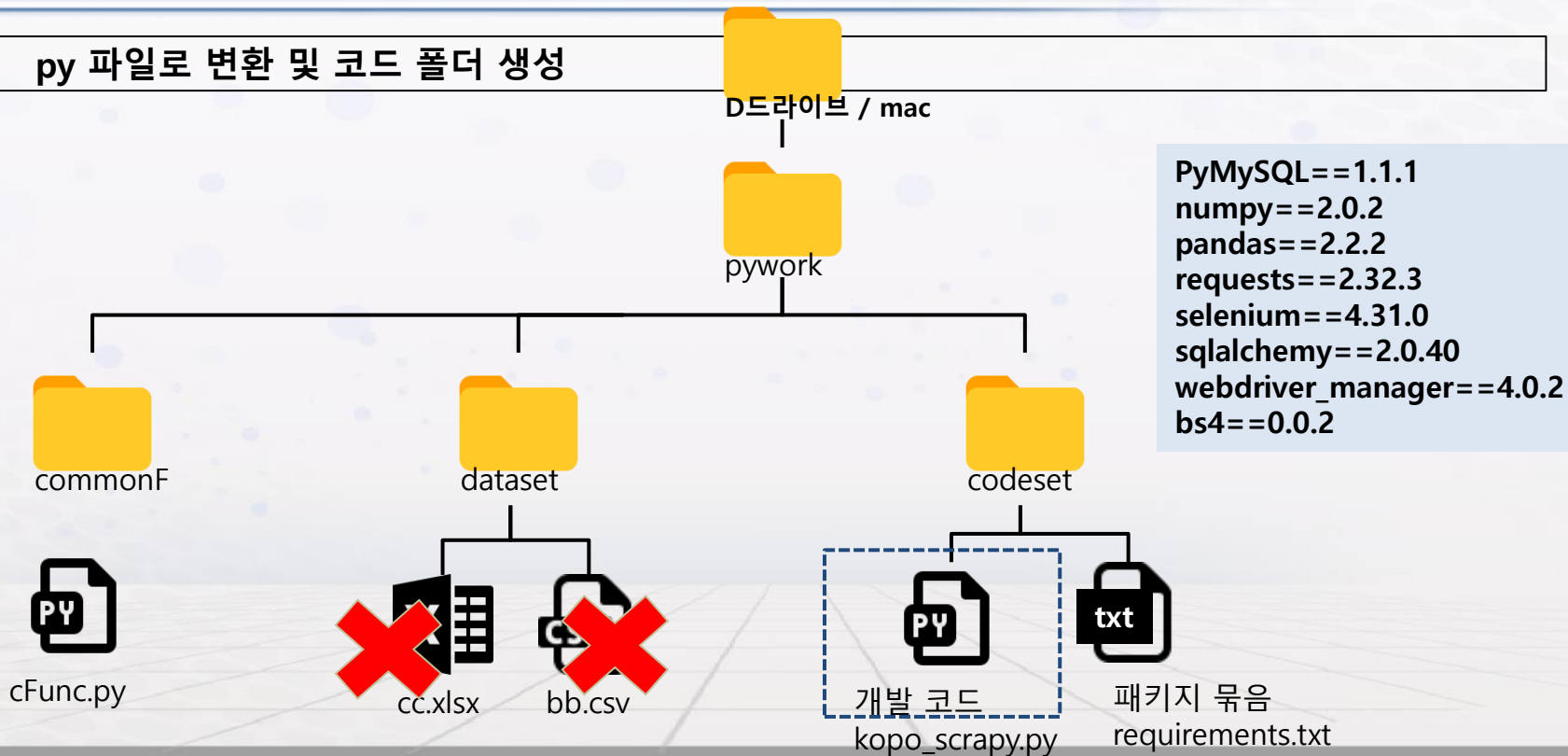
.py 파일 생성

Export Notebook to AsciiDoc
Export Notebook to HTML
Export Notebook to LaTeX
Export Notebook to Markdown
Export Notebook to PDF
Export Notebook to ReStructured Text
Export Notebook to Executable Script
Export Notebook to Reveal.js Slides

파이썬 프로그램 자동화

3. 실행가능 개발코드 생성 (테스트 포함 가능) ** Jupyter 환경

- py 파일로 변환 및 코드 폴더 생성



파이썬 프로그램 자동화

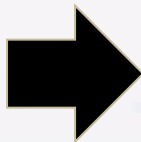
4. 리눅스 서버 실행 (클라우드 또는 On-premise 환경)

- 별도 파일 참고 (★AWS프리티어가입_EC2실행_터미널_파일전송_개발환경(DB_파이썬)_스케줄러_웹)

소스코드 이관

검증환경

(운영체제:
리눅스)



운영환경

(운영체제:
리눅스)

버전 통일

순번	모듈	버전
1	python	3.12
2	pandas	2.2.2
...
30	requests	2.32.3

순번	모듈	버전
1	python	3.12
2	pandas	2.2.2
...
30	requests	2.32.3



파이썬 프로그램 자동화

5. 리눅스 관리 프로그램 설치 (터미널 접속 / 파일 전송)

- 별도 파일 참고 (★AWS프리티어가입_EC2실행_터미널_파일전송_개발환경(DB_파이썬)_스케줄러_웹)

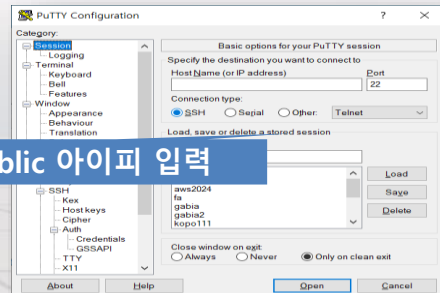
명령어
실행
(터미널 접속)



파일전송
(FTP)



public 아이피 입력

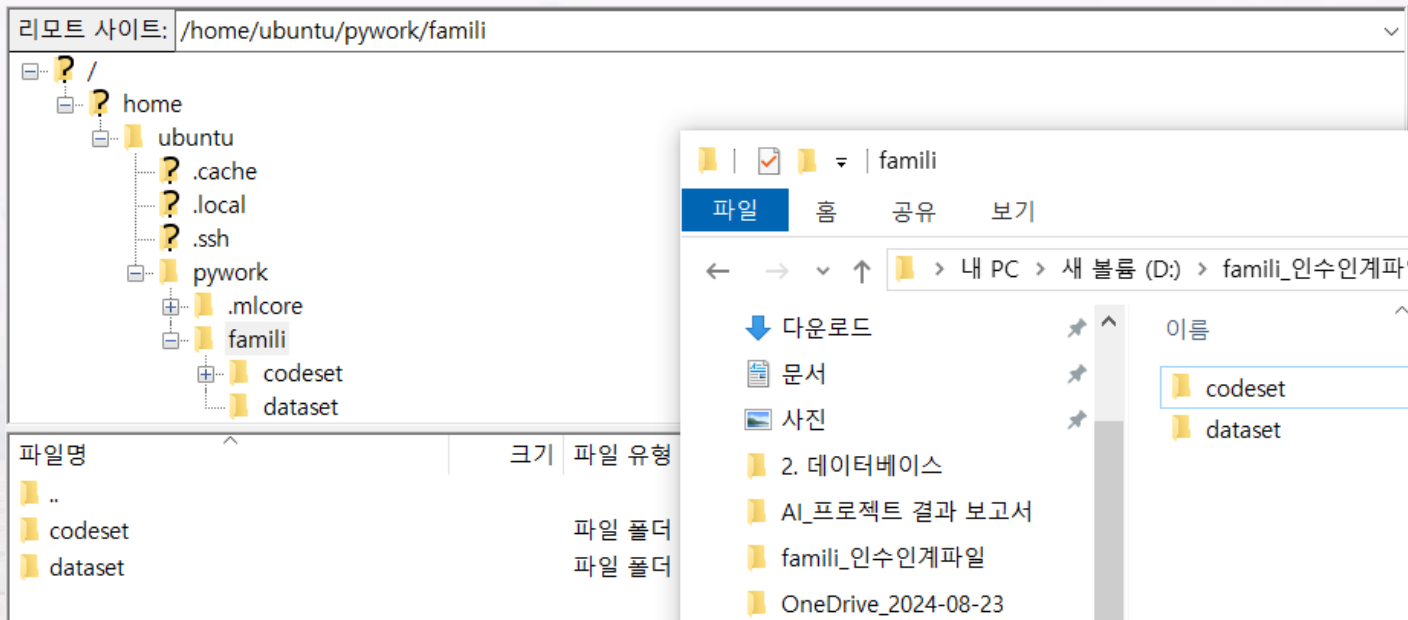


파이썬 프로그램 자동화

6. 개발코드 이관

- **별도 파일 참고 (★AWS프리티어가입_EC2실행_터미널_파일전송_개발환경(DB_파이썬)_스케줄러_웹)**

/home/ubuntu/pywork/ 하단에 파일 이관



파이썬 프로그램 자동화

7. 파이썬 가상환경 설치

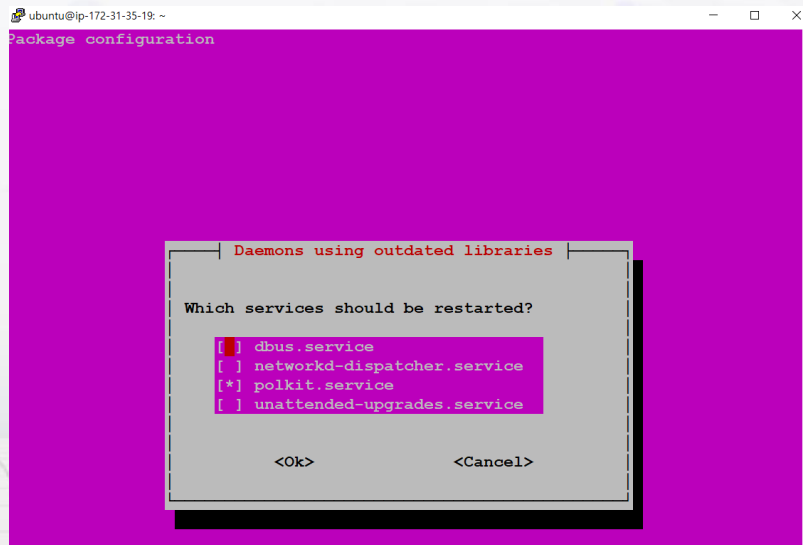
- **별도 파일 참고 (★AWS프리티어가입_EC2실행_터미널_파일전송_개발환경(DB_파이썬)_스케줄러_웹)**

```
sudo apt install python3.12-venv  
python3 -m venv .mlcore  
source .mlcore/bin/activate
```

* pandas 는 그냥 설치 후

```
ubuntu@ip-172-31-35-19:~$ sudo apt install python3-pip  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  build-essential bzip2 cpp cpp-11 dpkg-dev fakeroot fontconfig-config  
  fonts-dejavu-core g++ g++-11 gcc gcc-11 gcc-11-base javascript-common  
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl  
  libasan6 libatomic1 libc-dev-bin libc-devtools libc6-dev libcc1-0  
  libcrypt-dev libdeflate0 libdpkg-perl libexpat1 libexpat1-dev libfakeroot  
  libfile-fcntllock-perl libfontconfig1 libgcc-11-dev libgd3 libgomp1 libisl23  
  libitm1 libjbig0 libjpeg-turbo8 libjpeg8 libjs-jquery libjs-sphinxdoc  
  libjs-underscore liblsan0 libmpc3 libnsl-dev libpython3-dev
```

```
sudo apt remove nodejs
```



파이썬 프로그램 자동화

8. 패키지 설치 (requirements.txt 활용)

pip install -r requirements.txt

또는

pip install uvicorn==0.30.1

pip install fastapi==0.111.0

pip install pandas==2.0.3

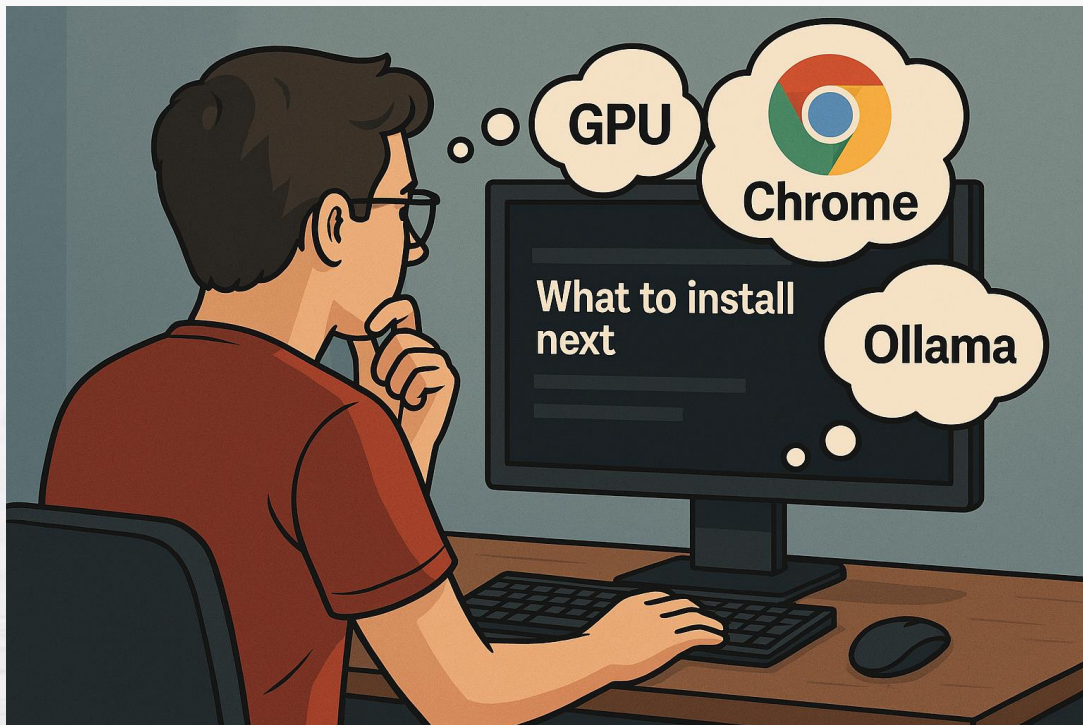
pip install scikit-learn==1.2.2

pip show matplotlib==3.7.1

```
ubuntu@ip-172-31-3-32: ~/pywork
Downloading kiwisolver-1.4.5-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (1.6 MB)
----- 1.6/1.6 MB 62.6 MB/s eta 0:00:00
Collecting cycler>=0.10
  Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Collecting fonttools>=4.22.0
  Downloading fonttools-4.53.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.6 MB)
----- 4.6/4.6 MB 66.5 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.20 in ./mlcore/lib/python3.10/site-packages (from matplotlib==3.7.1) (1.26.4)
Collecting pyparsing>=2.3.1
  Downloading pyparsing-3.1.2-py3-none-any.whl (103 kB)
----- 103.2/103.2 KB 14.9 MB/s eta 0:00:00
Collecting packaging>=20.0
  Downloading packaging-24.0-py3-none-any.whl (53 kB)
----- 53.5/53.5 KB 8.4 MB/s eta 0:00:00
Requirement already satisfied: six>=1.5 in ./mlcore/lib/python3.10/site-packages (from python-dateutil>=2.7->matplotlib==3.7.1) (1.16.0)
Installing collected packages: pyparsing, pillow, packaging, kiwisolver, fonttools, cycler, contourpy, matplotlib
Successfully installed contourpy-1.2.1 cycler-0.12.1 fonttools-4.53.0 kiwisolver-1.4.5 matplotlib-3.7.1 packaging-24.0 pillow-10.3.0 pyparsing-3.1.2
(.mlcore) ubuntu@ip-172-31-3-32:~/pywork$
```


파이썬 프로그램 자동화

9. 기타 프로그램 설치 (스킵 가능)




```
options.add_argument('--ignore-certificate-errors')
```



파이썬 프로그램 자동화

10. 파이썬 코드 실행 테스트

```
hkcode@ip-172-31-35-19:~/pywork$ python3 app.py
INFO: Will watch for changes in these directories: ['/home/hkcode/pywork']
INFO: Uvicorn running on (Press CTRL+C to quit)
INFO: Started reloader process [7657] using StatReload
INFO: Started server process [7659]
INFO: Waiting for application startup.
INFO: Application startup complete.
```



⚠ 주의 요함 3.35.167.36:9999

★ Bookmarks  NE Books – NE능률...  공유자료_2023년모

pretty print 적용 ☐

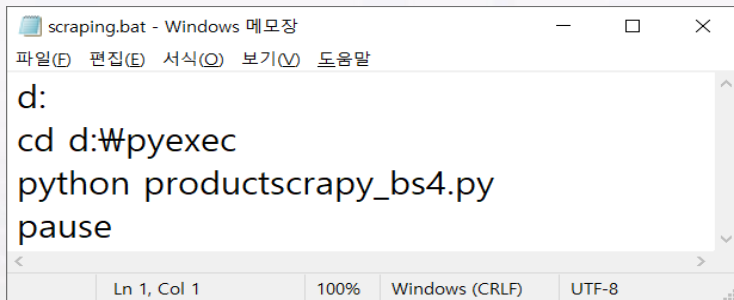
```
{"message": "online"}
```

파이썬 프로그램 자동화

11. 선택1 - 자동화 (crontab 활용)

- 윈도우 : 메모장 열기 → 폴더이동 및 실행코드 작성 → **scraping.bat** 저장
- 리눅스 : 셸스크립트 생성 → 폴더이동 및 실행코드 작성 → **scraping.sh** 저장 이후 권한 설정 **chmod 775**

윈도우 버전



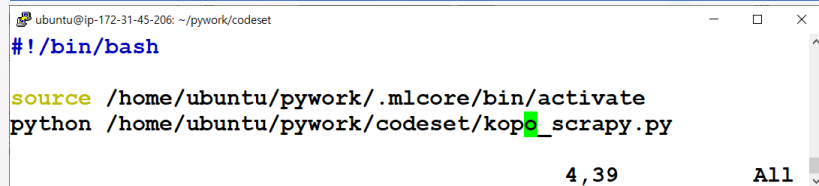
```
d:
cd d:\pyexec
python productscrapy_bs4.py
pause
```

파일 이름(N):

파일 형식(T):

폴더 숨기기

리눅스 버전

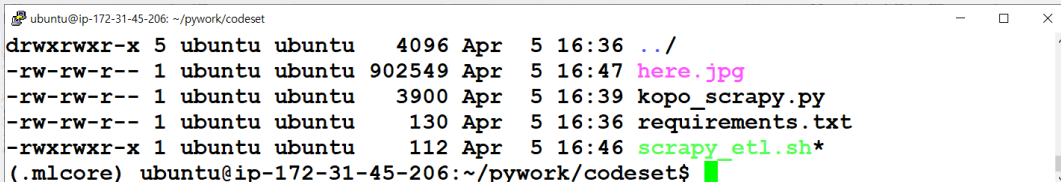


```
#!/bin/bash

source /home/ubuntu/pywork/.mlcore/bin/activate
python /home/ubuntu/pywork/codeset/kopo_scrapy.py
```

#!/bin/bash

```
source /home/ubuntu/pywork/.mlcore/bin/activate
python /home/ubuntu/pywork/codeset/kopo_scrapy.py
```



```
(mlcore) ubuntu@ip-172-31-45-206:~/pywork/codeset$
```

***** /bin/bash -c 'source /home/ubuntu/pywork/.mlcore/bin/activate && python /home/ubuntu/pywork/codeset/kopo_scrapy.py'

파이썬 프로그램 자동화

11. 선택1 - 자동화 (crontab 활용)

- 리눅스: 크론탭 활용

크론탭 편집기 실행 명령어
crontab -e

```
* /3 * * * * /home/ubuntu/pywork/kopo_scrapy.sh
```

```
# |----- minute (0 - 59)
# |----- hour (0 - 23)
# |----- day of the month (1 - 31)
# |----- month (1 - 12)
# |----- day of the week (0 - 6) (Sunday to Saturday;
# |                                     7 is also Sunday on some systems)
# |
# |
# * * * * * command to execute
```

```
* /3 * * * * /home/ubuntu/pywork/codeset/scrapy_etl.sh >> /home/ubuntu/pywork/codeset/scrapy_etl.log 2>&1
```

표준출력과 에러를 로그로 저장 (scrapy_etl.log)

매 3분마다 실행

* [wiki 가이드](https://en.wikipedia.org/wiki/Cron) → <https://en.wikipedia.org/wiki/Cron>

3.2. 특정 시간 실행

```
# 매주 금요일 오전 5시 45분에 test.sh 를 실행
45 5 * * 5 /home/script/test.sh
```

파이썬 프로그램 자동화

11. 선택1 - 자동화 (crontab 활용)

```
* /3 * * * * . /home/ubuntu/hkcode/pywork/.mlcore/bin/activate && python  
/home/ubuntu/hkcode/pywork/projectCode/codeset/spark_scrapy.py >>  
/home/ubuntu/hkcode/pywork/projectCode/codeset/spark_scrapy.log 2>&1
```

우분투 기준 /var/log/syslog 파일 로그 확인 가능

파이썬 프로그램 자동화

11. 선택1 - 자동화 (crontab 활용)

리눅스 버전

```
tail -f /var/log/cron
```

단, 주의할 점은 crontab은 사용자 환경변수를 사용하지 않는다.
etc/profile 에 시스템 환경변수 등록 후 source /etc/profile 반영 필요

참고 - 타임존 설정

타임존 설정

리눅스 버전

타임존 변경

```
sudo timedatectl set-timezone Asia/Seoul
```

```
* timedatectl list-timezones
```

타임존 확인

```
timedatectl
```

에러 시

```
sudo apt-get install tzdata
```

참고 - 셀레니움 (크롬브라우저 연동) 관련

크롬 및 드라이버 설치(기본)

방법: <https://suucong.tistory.com/58>

크롬드라이버 다운로드: <https://googlechromelabs.github.io/chrome-for-testing/>

<https://github.com/GoogleChromeLabs/chrome-for-testing/blob/main/data/known-good-versions-with-downloads.json>

리눅스 버전

[크롬 브라우저 설치]

GPG 키 추가

wget -q -O - https://dl.google.com/linux/linux_signing_key.pub | sudo apt-key add -

리포지터리 등록

echo "deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable main" | sudo tee /etc/apt/sources.list.d/google-chrome.list

업데이트 및 설치

sudo apt update

sudo apt install google-chrome-stable

설치 버전 확인

google-chrome --version

삭제 시: sudo apt purge google-chrome-stable

또는

[크롬드라이버 설치]

wget https://storage.googleapis.com/chrome-for-testing-public/127.0.6533.2/linux64/chromedriver-linux64.zip

[압축해제]

unzip chromedriver-linux64.zip

참고 - 셀레니움 (크롬브라우저 연동) 관련

크롬 및 드라이버 설치(No)

방법: <https://suucong.tistory.com/58>

크롬드라이버 다운로드: <https://googlechromelabs.github.io/chrome-for-testing/>

<https://github.com/GoogleChromeLabs/chrome-for-testing/blob/main/data/known-good-versions-with-downloads.json>

리눅스 버전

[크롬 브라우저 설치]

Google Chrome의 최신 .deb 파일 다운로드

```
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
```

다운로드 받은 .deb 파일을 설치

```
sudo apt install ./google-chrome-stable_current_amd64.deb
```

설치 버전 확인

```
google-chrome --version
```

또는

[크롬드라이버 설치]

```
wget https://storage.googleapis.com/chrome-for-testing-public/127.0.6533.2/linux64/chromedriver-linux64.zip
```

[압축해제]

```
sudo apt install unzip
```

```
unzip chromedriver-linux64.zip
```

참고 - 셀레니움 (크롬브라우저 연동) 관련

셀레니움 코드 수정 및 실행 (드라이버 최신으로 설정 시)

리눅스 버전

WebDriver 설치 및 브라우저 실행 설정

```
options = webdriver.ChromeOptions()
```

```
options.add_argument("--headless") # 헤드리스 모드로 실행 (브라우저 창을 표시하지 않음)
```

```
options.add_argument("window-size=1920x1080") # 헤드리스 모드로 실행 (명시적으로 표기)
```

```
options.add_argument("--no-sandbox") # 샌드박스 모드 비활성화 (가상 환경에서 안정적인 실행을 위해)
```

```
options.add_argument("--disable-dev-shm-usage") # /dev/shm 사용 비활성화 (메모리 부족 방지)
```

user-agent 값 설정

```
user_agent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
```

```
Chrome/114.0.0.0 Safari/537.36"
```

```
options.add_argument(f"user-agent={user_agent}")
```

```
service = Service(executable_path="/home/ubuntu/pywork/chromedriver-linux64/chromedriver")
```

```
driver = webdriver.Chrome(service= service, options=options)
```

참고 - 셀레니움 (크롬브라우저 연동) 관련

셀레니움 코드 수정 및 실행 (드라이버 고정 시!)

리눅스 버전

WebDriver 설치 및 브라우저 실행 설정

```
options = webdriver.ChromeOptions()
```

```
options.add_argument("--headless") # 헤드리스 모드로 실행 (브라우저 창을 표시하지 않음)
```

```
options.add_argument("window-size=1920x1080") # 헤드리스 모드로 실행 (명시적으로 표기)
```

```
options.add_argument("--no-sandbox") # 샌드박스 모드 비활성화 (가상 환경에서 안정적인 실행을 위해)
```

```
options.add_argument("--disable-dev-shm-usage") # /dev/shm 사용 비활성화 (메모리 부족 방지)
```

user-agent 값 설정

```
user_agent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
```

```
Chrome/114.0.0.0 Safari/537.36"
```

```
options.add_argument(f"user-agent={user_agent}")
```

```
service = Service(executable_path="/home/ubuntu/pywork/chromedriver-linux64/chromedriver")
```

```
driver = webdriver.Chrome(service= service, options=options)
```

감사합니다.

