

Python 설치 및 사용환경

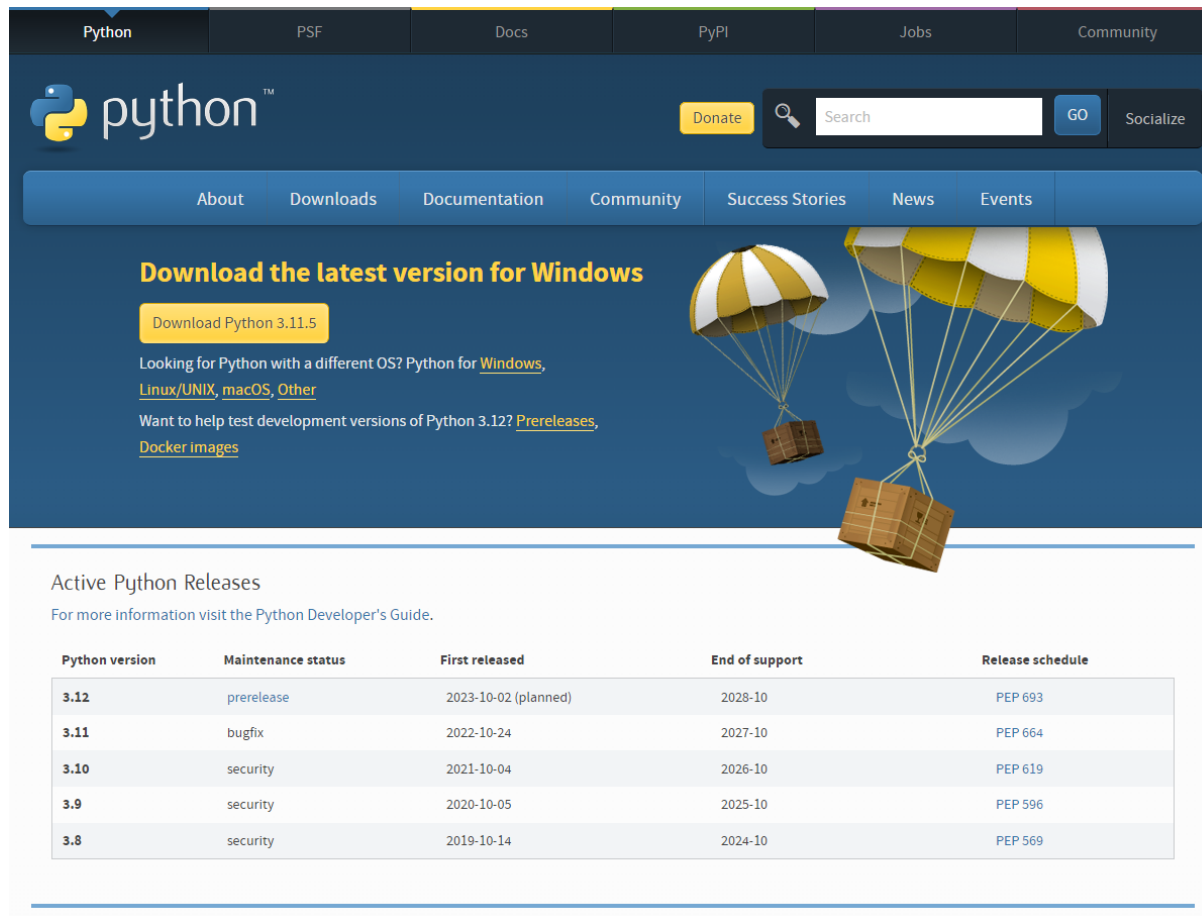


- Python 설치
- VS code 설치
- VS code에서 가상환경 생성
- fast API 간단 예제
- DOS prompt에서 가상환경 생성

Python 설치

- OS(Operating System)
 - Linux : pre-installed
 - Mac, Windows
- 개발환경
 - console + editor
 - prompts ">>>"
 - quit(), exit(), ctrl-Z
 - Visual Studio Code
 - pycharm
 - Jupyter Notebook (**Julia** + **python** + **R**)
 - google colab (cloud)
 - Anaconda, Mini-conda (Anaconda cloud)

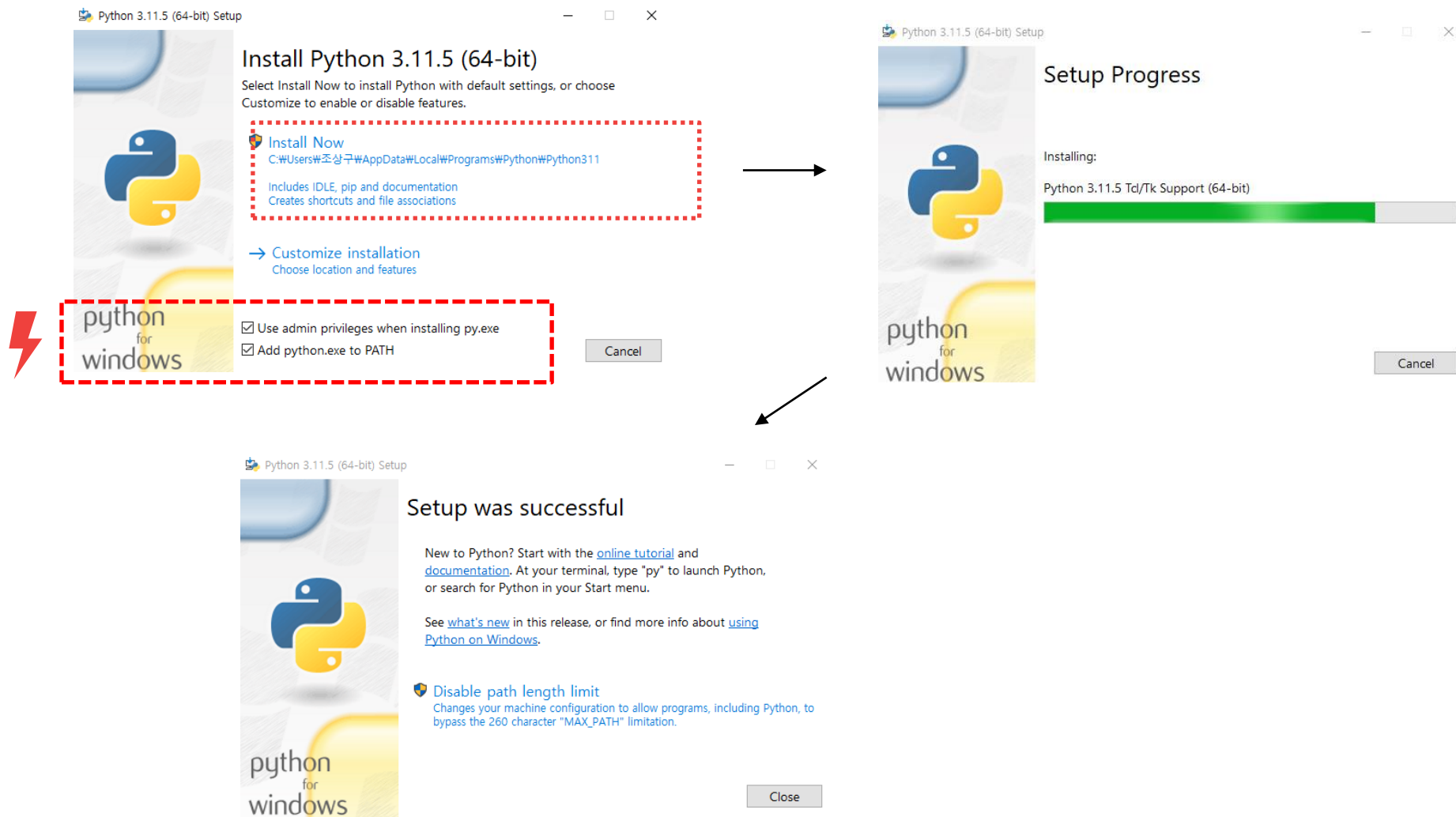
Python 다운로드 및 설치



The screenshot shows the Python.org website. The top navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. Below this is a search bar with a 'GO' button and a 'Socialize' link. The main content area features a large banner with the text 'Download the latest version for Windows' and a button to 'Download Python 3.11.5'. Below the banner, there are links for 'Linux/UNIX, macOS, Other' and 'Prereleases, Docker images'. The bottom section is titled 'Active Python Releases' and contains a table with the following data:

Python version	Maintenance status	First released	End of support	Release schedule
3.12	prerelease	2023-10-02 (planned)	2028-10	PEP 693
3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569

Python 다운로드 및 설치



Python 설치 – 패키지(외부라이브러리) 설치

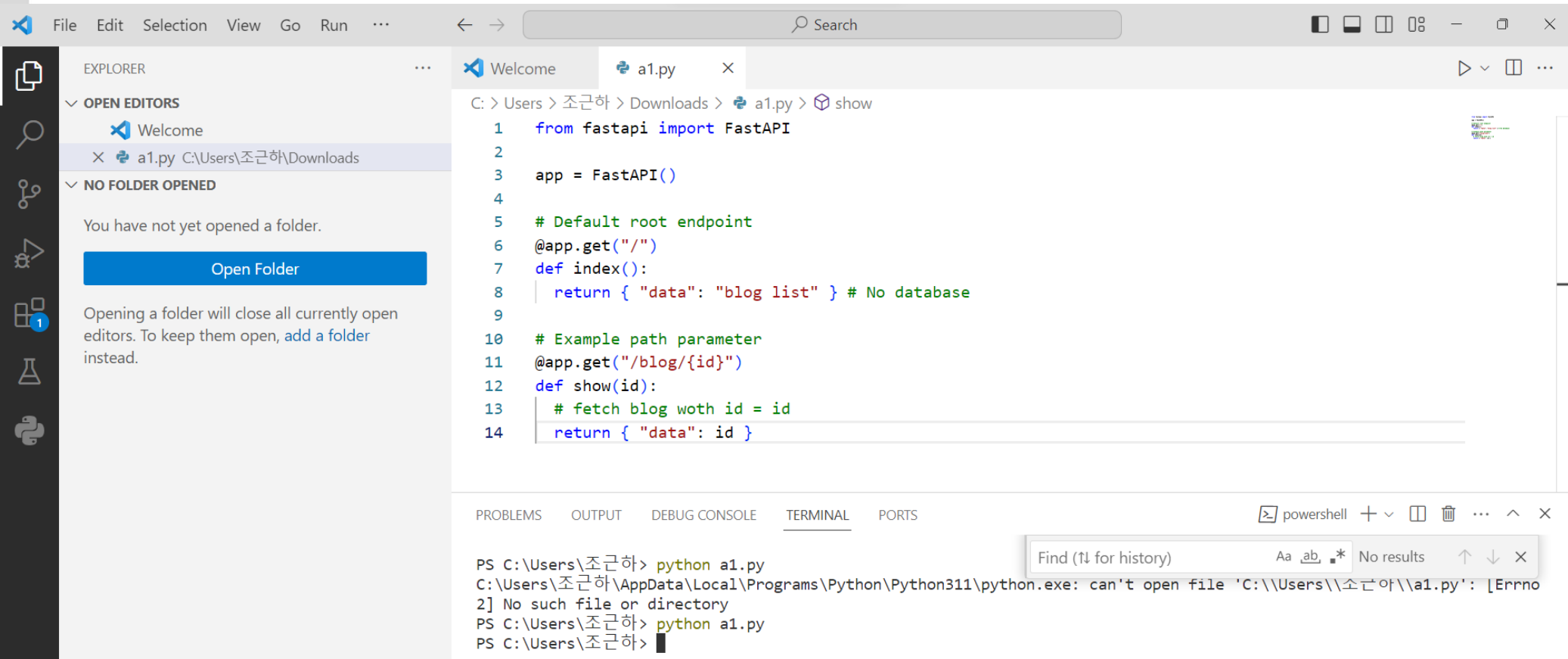
- pip은 Python용 패키지 설치 프로그램
 - Python 패키지는 Python에 기능을 추가하는 코드, 모듈 및 리소스 모음
 - 'pip'을 사용하면 Python 커뮤니티에서 개발한 타사 라이브러리, 프레임워크 및 도구를 설치할 수 있으므로 Python 프로그램의 기능을 훨씬 간단하게 확장
 - 'pip'을 사용하여 설치된 패키지에는 NumPy 및 Pandas와 같은 데이터 분석 라이브러리부터 Flask 및 Django와 같은 웹 프레임워크에 이르기까지 다양한 유형의 소프트웨어가 포함될 수 있습니다.
- 기본적인 패키지 설치 명령어
 - `pip install package-name`
 - `pip install --upgrade package-name`: 이미 설치된 패키지를 최신 버전으로 업그레이드
 - `pip uninstall package-name`: 패키지를 제거
 - `pip list`: 설치된 모든 패키지를 나열
 - `pip install pandas`, `pip install jupyter`,.....

Python 설치 및 사용환경



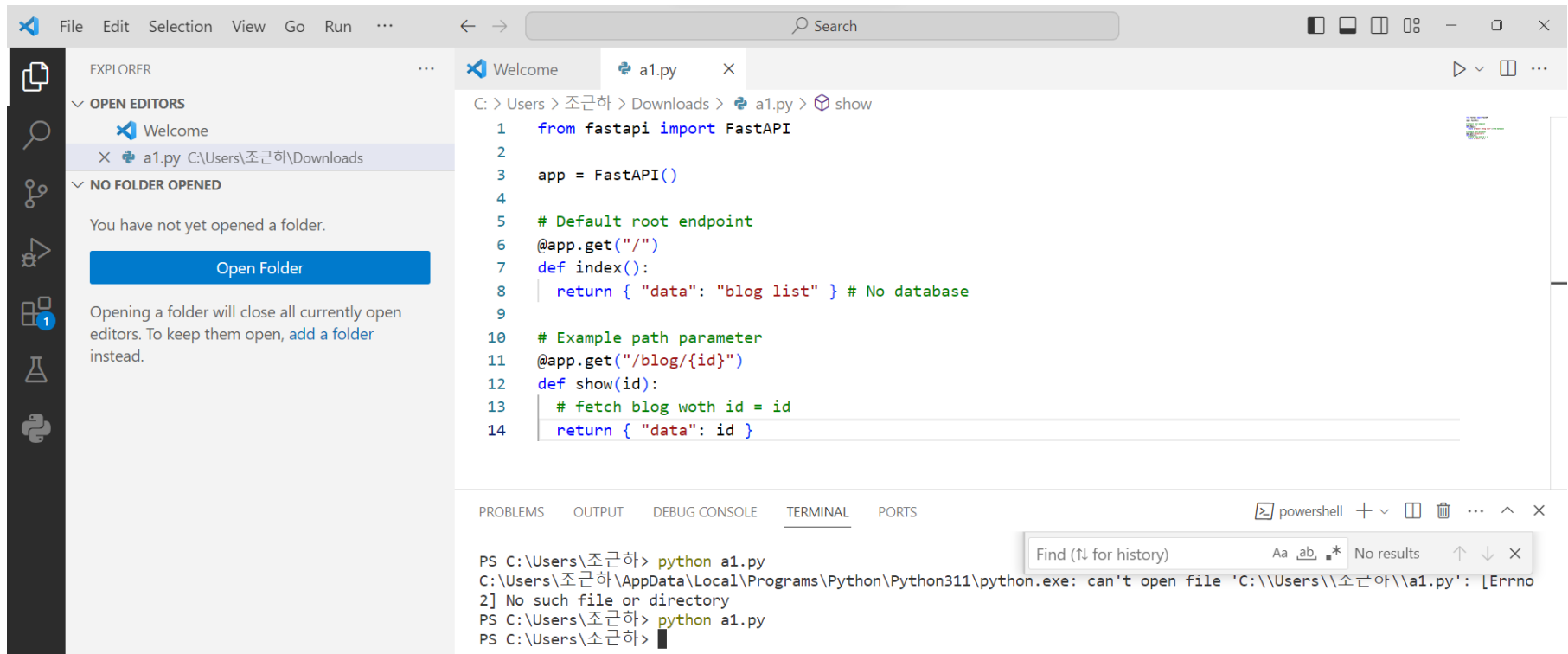
- Python 설치
- VS code 설치
- VS code에서 가상환경 생성
- fast API 간단 예제
- DOS prompt에서 가상환경 생성

Visual Code Studio 설치



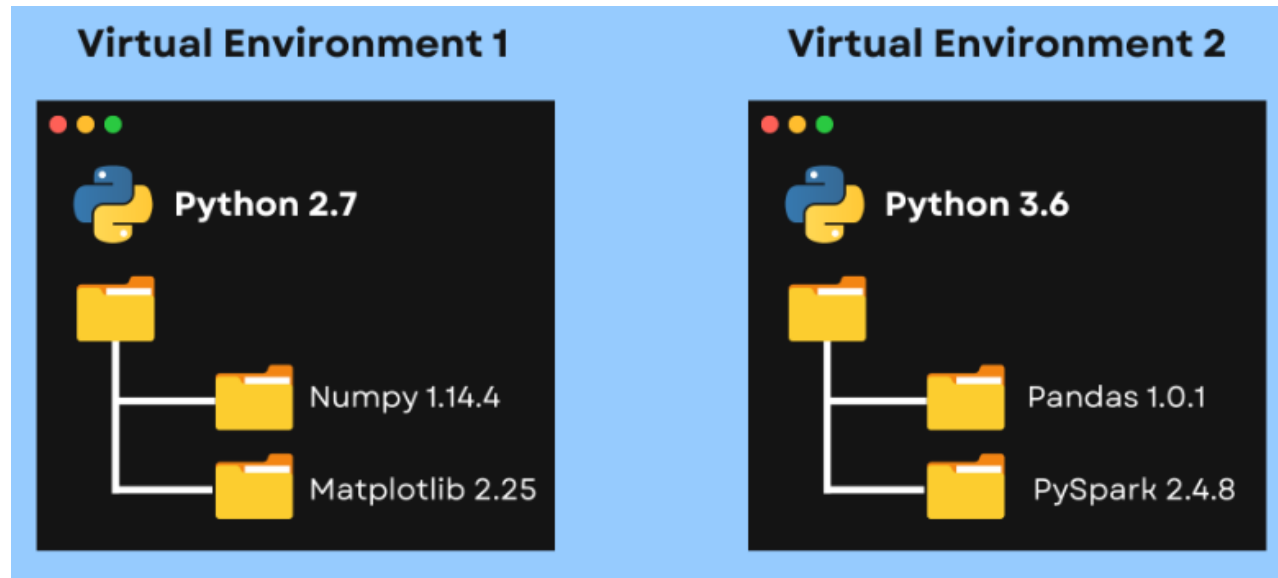
Visual Code Studio Terminal 모드

VS code에서 Terminal 모드로 콘솔 명령어 수행 가능



Python 가상환경

- 특정한 파이썬 인터프리터와 그에 따른 라이브러리, 패키지, 종속성을 포함하는 독립된 디렉토리 환경
- 가상 환경은 서로 다른 파이썬 프로젝트를 서로 격리(Isolation)시켜주며, 패키지 버전 간의 충돌을 방지하고 개발을 위한 통제된 환경을 만드는 데 사용(Dependency Management, Version Compatibility)



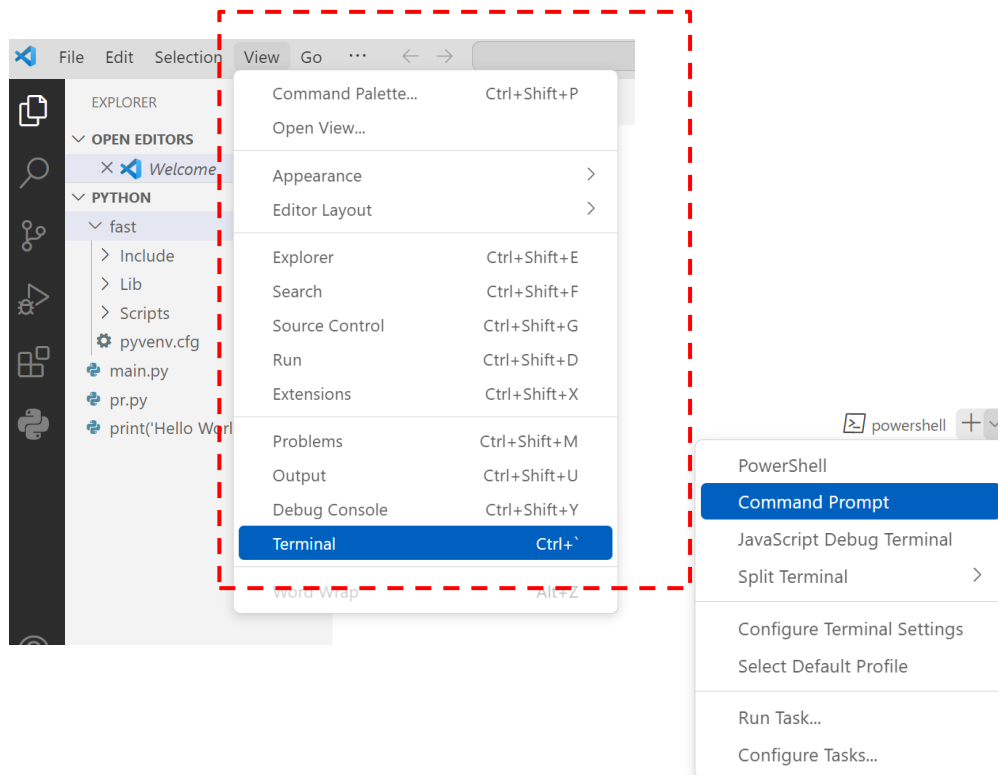
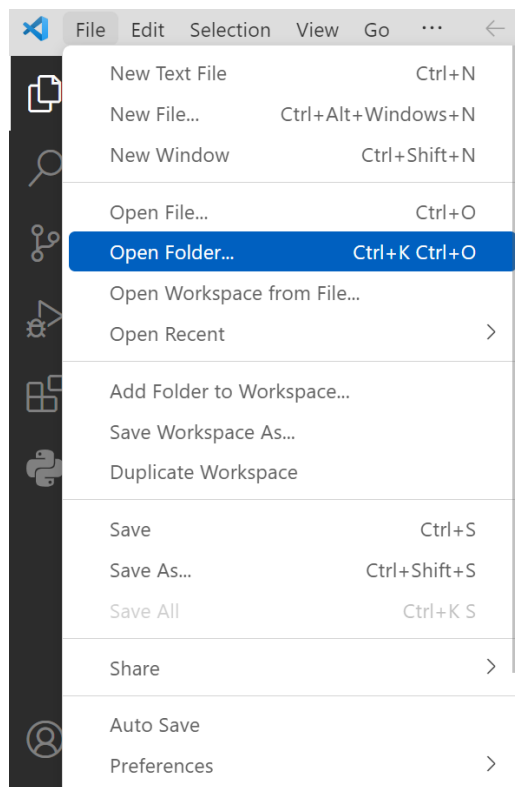
Python 설치 및 사용환경



- Python 설치
- VS code 설치
- VS code에서 가상환경 생성
- fast API 간단 예제
- DOS prompt에서 가상환경 생성

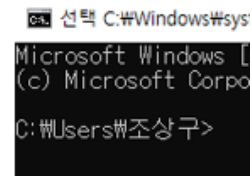
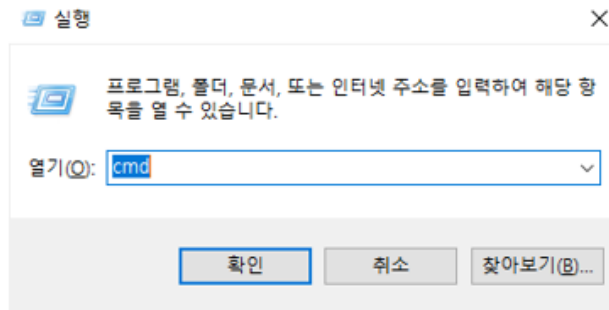
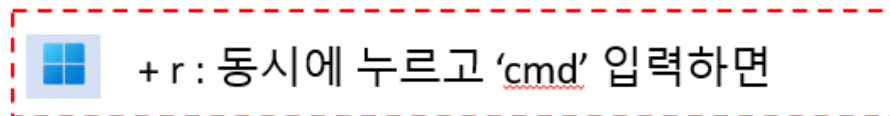
Python 가상환경

File > Open Folder 명령어로 미리 만든 python 디렉토리로 이동하고 **View > Terminal > Command Prompt**를 클릭



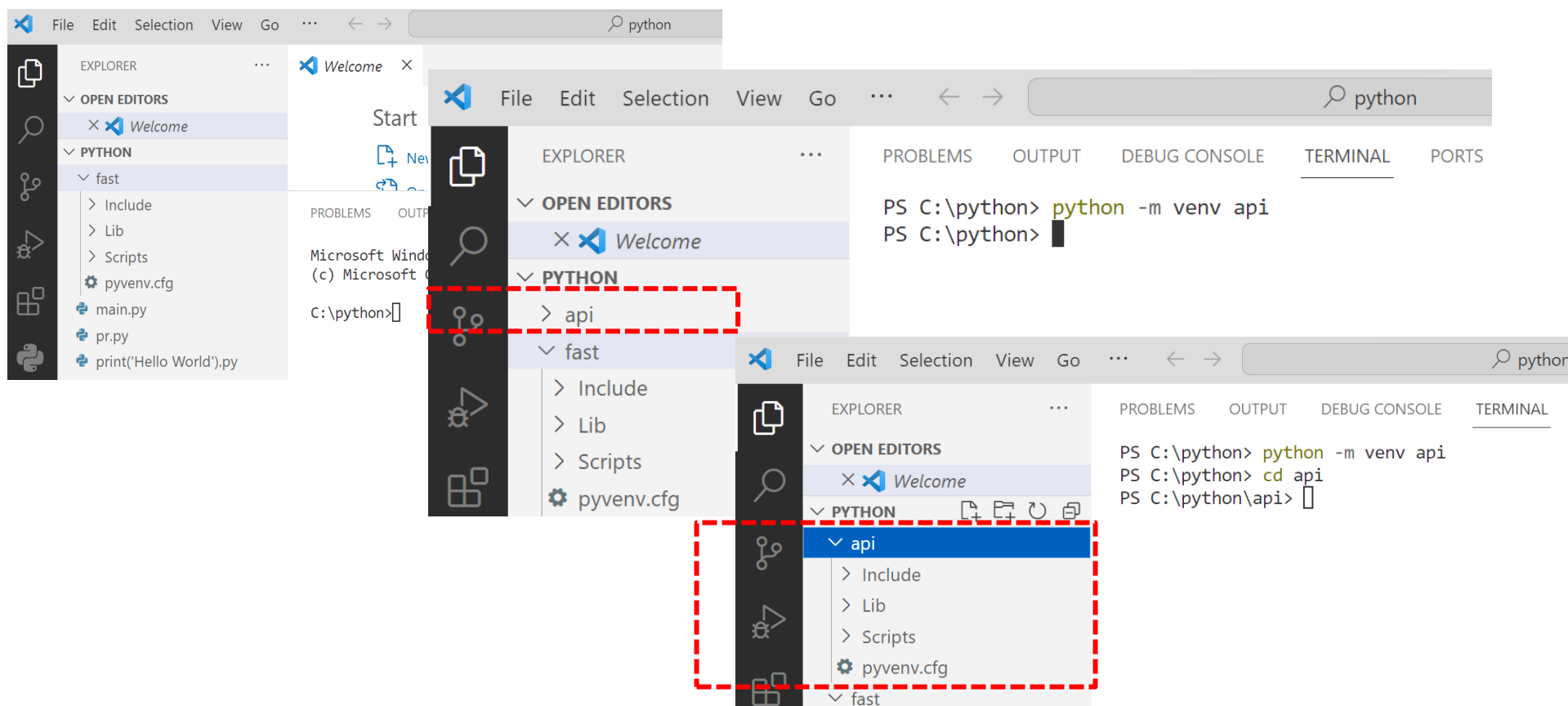
Python 가상환경

View > Terminal > Command Prompt를 클릭 하는 것은 아래 Dos shell 명령어와 동일함



Python 가상환경

이제 python 디렉토리에서 'python -m venv api'를 입력하면 'api' 가상환경 디렉토리가 만들어지고 하위 디렉토리와 파일이 자동으로 만들어짐



Python 가상환경

이제 가상환경 **Scripts** 디렉토리로 들어가서 **'activate'**를 입력하면 Dos 환경이
(api) C:\python\api\Scripts> 로 변하면서 가상환경이 만들어진다.(해제는 **'deactivate'**)

```
File Edit Selection View Go ... python
```

EXPLORER

OPEN EDITORS

- × Welcome

PYTHON

- api
 - Include
 - Lib
 - Scripts
 - pyenv.cfg
 - fast

```
PS C:\python> python -m venv api
PS C:\python> cd api
PS C:\python\api>
```

```
File Edit Selection View Go ... python
```

EXPLORER

OPEN EDITORS

- × Welcome

PYTHON

- api
- fast

```
PS C:\python> python -m venv api
PS C:\python> cd api
PS C:\python\api> cd Scripts
PS C:\python\api\Scripts> activate
```

(api) C:\python\api\Scripts>

Python 설치 및 사용환경



- Python 설치
- VS code 설치
- VS code에서 가상환경 생성
- fast API 간단 예제
- DOS prompt에서 가상환경 생성

Python 가상환경

이제 가상환경에만 필요한 모듈, fastapi, uvicorn을 설치하면 된다.

```
(api) C:\python\api\Scripts>pip install fastapi
```

```
Collecting fastapi
```

```
Obtaining dependency information for fastapi from https://files.pythonhosted.org/packages/4d/d2/3ad038a2365fefbac19d9a046cab7ce45f4c7bfa81d877cbece9707de9ce/fastapi-0.103.2-py3-none-any.whl.metadata
```

```
Using cached fastapi-0.103.2-py3-none-any.whl.metadata (24 kB)
```

```
Collecting anyio<4.0.0,>=3.7.1 (from fastapi)
```

```
Obtaining dependency information for anyio<4.0.0,>=3.7.1 from https://files.pythonhosted.org/packages/19/24/44299477fe7dcc9cb58d0a57d5a7588d6af2ff403fdd2d47a246c91a3246/anyio-3.7.1-py3-none-any.whl.metadata
```

```
(api) C:\python\api\Scripts>pip install uvicorn
```

```
Collecting uvicorn
```

```
Obtaining dependency information for uvicorn from https://files.pythonhosted.org/packages/79/96/b0882a1c3f7ef3dd86879e041212ae5b62b4bd352320889231cc735a8e8f/uvicorn-0.23.2-py3-none-any.whl.metadata
```

```
Using cached uvicorn-0.23.2-py3-none-any.whl.metadata (6.2 kB)
```

```
Collecting click>=7.0 (from uvicorn)
```

```
Obtaining dependency information for click>=7.0 from https://files.pythonhosted.org/packages/00/2e/d53fa4befbf2cfa713304affc7ca780ce4fc1fd8710527771b58311a3229/click-8.1.7-py3-none-any.whl.metadata
```

```
Using cached click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
```

```
Collecting h11>=0.8 (from uvicorn)
```


Python 가상환경

main.py 파일을 아래와 같이 만든다.

```
Welcome  main.py  X
api > Scripts > main.py > show
1  from fastapi import FastAPI
2
3  app = FastAPI()
4
5  # Default root endpoint
6  @app.get("/")
7  def index():
8      return { "data": "blog list" } # No database
9
10 # Example path parameter
11 @app.get("/blog/{id}")
12 def show(id):
13     # fetch blog with id = id
14     return { "data": id }
```

1. `from fastapi import FastAPI`: FastAPI 모듈을 가져옵니다.
2. `app = FastAPI()`: FastAPI 애플리케이션을 생성합니다.
3. `@app.get("/")`: HTTP GET 요청을 처리하는 데코레이터입니다. 이 데코레이터는 `/`` 경로에 대한 요청을 처리하는 함수인 `index`를 지정합니다.
4. `def index()`: `/`` 경로에 대한 요청을 처리하는 함수입니다. 이 함수는 단순히 JSON 형식의 응답을 반환합니다. 응답은 `{ "data": "blog list" }` 형식으로 되어 있으며, 이 예제에서는 데이터베이스에 접근하지 않고 단순한 텍스트 응답을 반환합니다.
5. `@app.get("/blog/{id}")`: `/blog/{id}`` 경로에 대한 요청을 처리하는 데코레이터입니다. `{id}``는 경로 매개변수로 사용됩니다.
6. `def show(id)`: `/blog/{id}`` 경로에 대한 요청을 처리하는 함수입니다. 이 함수는 `id`` 매개변수를 받아서 해당 ID에 대한 블로그를 검색하고, 그 결과를 JSON 형식으로 반환합니다. 이 예제에서는 실제로 데이터베이스에 접근하지 않고, 단순히 `id`` 값을 반환하는 예시입니다.

이러한 코드를 실행하면 FastAPI 애플리케이션이 실행되고, `/`` 경로로의 GET 요청은 `{ "data": "blog list" }`를 반환하고, `/blog/{id}`` 경로로의 GET 요청은 `{ "data": id }` 값을 반환합니다. 이는 FastAPI를 사용하여 간단한 API를 구축하는 기본 예제입니다.

Python 가상환경

‘uvicorn main:app --reload’를 실행

```
(api) C:\python\api\Scripts>uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['C:\\python\\api\\Scripts']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [24372] using StatReload
INFO: Started server process [3836]
INFO: Waiting for application startup.
INFO: Application startup complete.
```



이 명령은 Uvicorn이라는 ASGI 서버를 실행하며, **main:app**은 FastAPI 애플리케이션의 위치를 지정하고 **--reload** 플래그는 코드가 변경될 때 애플리케이션을 다시 로드하도록 지시

1. **uvicorn**: Uvicorn은 ASGI(Asynchronous Server Gateway Interface) 서버로, FastAPI와 함께 사용됩니다. ASGI 서버는 비동기 웹 애플리케이션을 처리하는 데 특화
2. **main:app**: FastAPI 애플리케이션의 진입점을 지정합니다. **main**은 Python 파일의 이름이며, **app**은 FastAPI 애플리케이션 객체의 변수명입니다. 이 명령은 **main.py** 파일에 정의된 FastAPI 애플리케이션을 실행
3. **--reload**: 이 플래그를 사용하면 코드 변경을 감지하고 변경 사항이 발생할 때마다 애플리케이션을 자동으로 다시 로드합니다. 이것은 개발 중에 코드를 수정하고 즉시 테스트하고 디버그하는 데 매우 편리합니다. 코드를 수정하고 저장할 때마다 서버가 다시 시작됩니다.
4. 따라서 **uvicorn main:app --reload** 명령을 실행하면 FastAPI 애플리케이션을 실행하고 개발 중에 코드를 수정하면 서버가 자동으로 다시 시작하여 변경 사항을 즉시 반영할 수 있습니다.

실행결과 확인하기

```
1 // 20231004200831
2 // http://127.0.0.1:8000/
3
4 {
5   "data": "blog list"
6 }
```

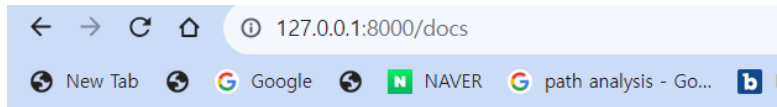


```
1 // 20231004200947
2 // http://localhost:8000/
3
4 {
5   "data": "blog list"
6 }
```

```
1 // 20231004201114
2 // http://127.0.0.1:8000/blog/1
3
4 {
5   "data": "1"
6 }
```

```
1 // 20231004201138
2 // http://127.0.0.1:8000/blog/100
3
4 {
5   "data": "100"
6 }
```

실행결과 확인하기



FastAPI 0.1.0 OAS 3.1
[/openapi.json](#)

default

GET / Index

GET /blog/{id} Show

Schemas

HTTPValidationError > Expand all object

ValidationError > Expand all object

Python 설치 및 사용환경

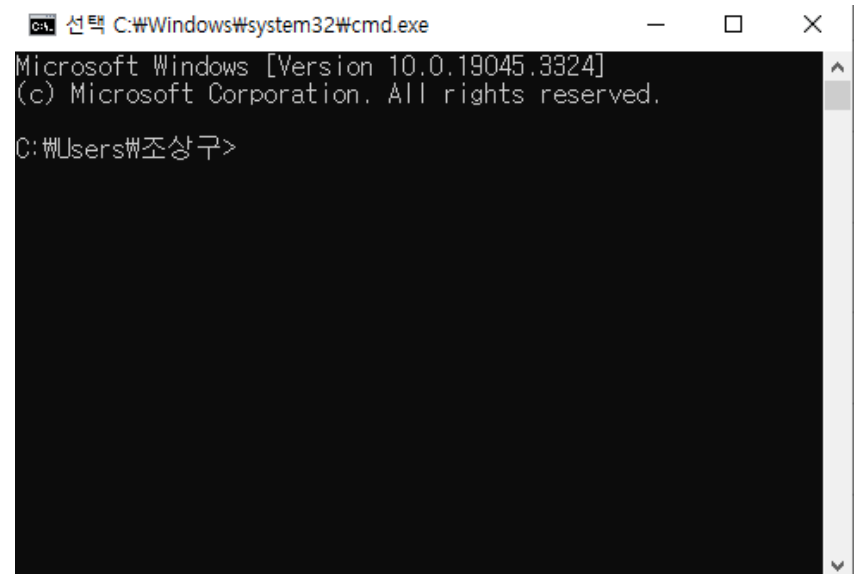
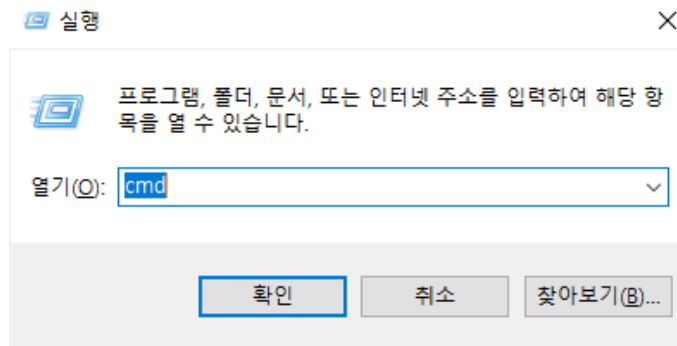


- Python 설치
- VS code 설치
- VS code에서 가상환경 생성
- fast API 간단 예제
- DOS prompt에서 가상환경 생성

python 가상환경 만들기



+ r : 동시에 누르고 'cmd' 입력하면



python 가상환경 만들기

1. 작업할 디렉토리 'my_project'를 만들고(Optional) 해당 디렉토리로 이동

C:\Windows\system32\cmd.exe

Microsoft Windows [Version 10.0.19045.3324]
(c) Microsoft Corporation. All rights reserved.

C:\Users\조상구>cd..

C:\Users>cd..

C:\>mkdir my_project

C:\>cd my_project

C:\my_project>

- ✓ 상위 디렉토리로 이동(change directory)
- ✓ 한 번 더 (상대 주소 like excel)
- ✓ 'my_project'라는 디렉토리 만들기(make directory)
- ✓ 'my_project'로 이동

python 가상환경 만들기

2. 작업할 디렉토리 'my_project'에서 가상환경 디렉토리 만들기

Virtual environments

<https://flask.palletsprojects.com/en/2.3.x/installation/#virtual-environments>

Use a virtual environment to manage the dependencies for your project, both in development and in production.

What problem does a virtual environment solve? The more Python projects you have, the more likely it is that you need to work with different versions of Python libraries, or even Python itself. Newer versions of libraries for one project can break compatibility in another project.

Virtual environments are independent groups of Python libraries, one for each project. Packages installed for one project will not affect other projects or the operating system's packages.

Python comes bundled with the **venv** module to create virtual environments.

Create an environment

Create a project folder and a **.venv** folder within:

macOS/Linux

Windows

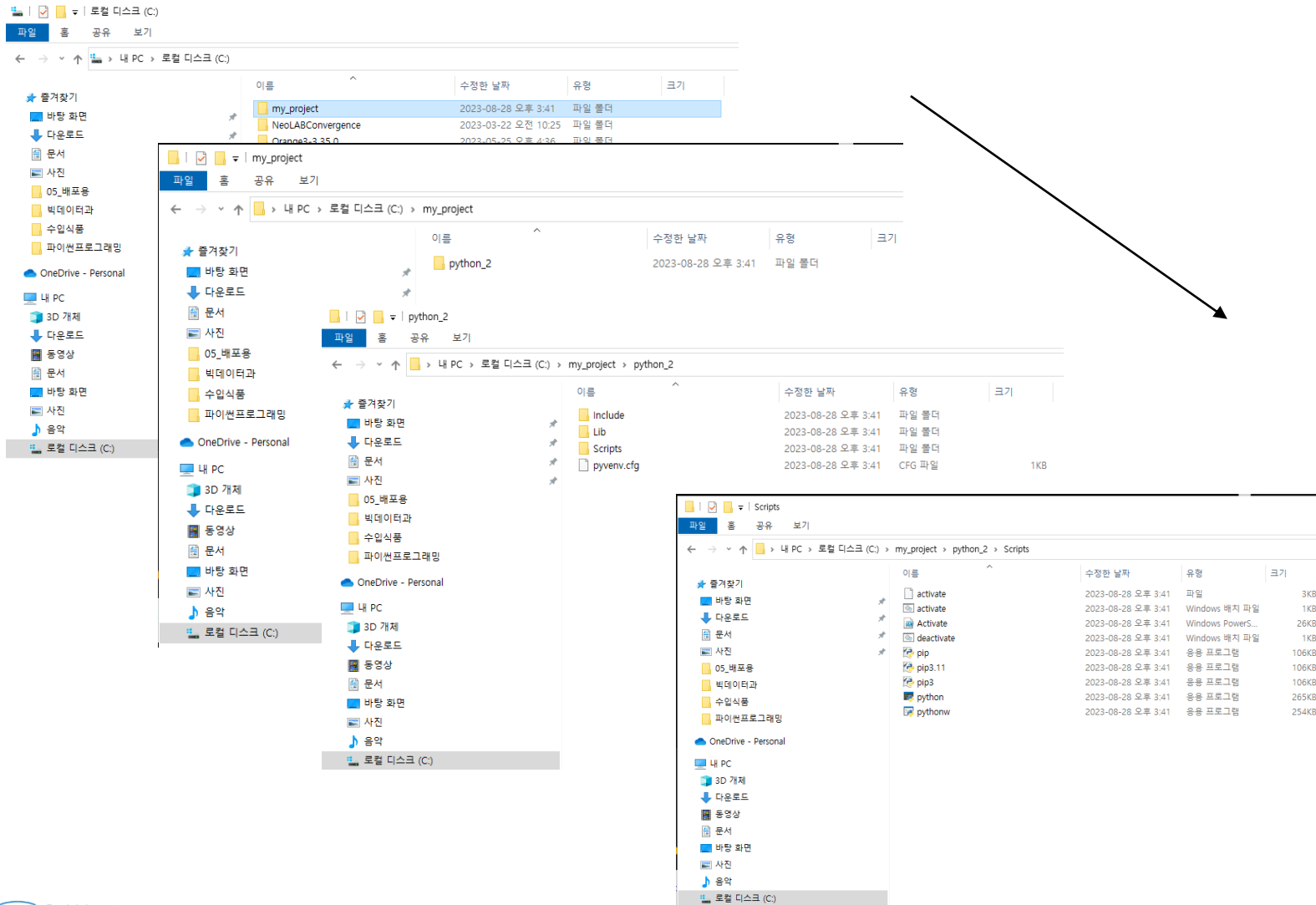
```
> mkdir myproject
> cd myproject
> py -3 -m venv .venv
```

- **python -m venv <가상환경 dir 이름>**
 - 'python_2'라는 가상환경 디렉토리 자동생성

C:\Windows\system32\cmd.exe

```
C:\my_project>python -m venv python_2
C:\my_project>
```


python 가상환경 만들기



python 가상환경 만들기

3. 가상환경 디렉토리('python_2')로 이동하여 가상환경 생성(activate 입력)

❖ 가상환경 생성 이후에는 아무것도 없기 때문에 먼저 가상 환경을 활성화(Scripts 디렉토리에서) 하고 python 설치해야한다.

The screenshot shows a Windows command prompt window with the following commands and output:

```
C:\Windows\system32\cmd.exe

(python_2) C:\my_project\python_2\Scripts>deactivate
C:\my_project\python_2\Scripts>cd .

C:\my_project\python_2>activate
'activate'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.

C:\my_project\python_2>cd Scripts
C:\my_project\python_2\Scripts>activate
```

Below the command prompt, a File Explorer window shows the contents of the `Scripts` directory. The files listed are:

이름	수정된 날짜	유형	크기
activate	2023-08-28 오후 3:41	파일	3KB
activate	2023-08-28 오후 3:41	Windows 배치 파일	1KB
Activate	2023-08-28 오후 3:41	Windows Powershell 스크립트	26KB
deactivate	2023-08-28 오후 3:41	Windows 배치 파일	1KB
pip	2023-08-28 오후 3:41	응용 프로그램	106KB
pip3.11	2023-08-28 오후 3:41	응용 프로그램	106KB
pip3	2023-08-28 오후 3:41	응용 프로그램	106KB
python	2023-08-28 오후 3:41	응용 프로그램	265KB
pythonw	2023-08-28 오후 3:41	응용 프로그램	254KB

python 가상환경 만들기

4. 가상환경에서 python 버전 확인

With the virtual environment activated, you don't need to install Python separately. The virtual environment already contains its own Python interpreter.

C:\Windows\system32\cmd.exe

```
(python_2) C:\my_project\python_2\Scripts>python --version  
Python 3.11.5
```

```
(python_2) C:\my_project\python_2\Scripts>
```

선택 C:\Windows\system32\cmd.exe - python -m notebook

```
(python_2) C:\my_project\python_2\Scripts>python -m notebook
```

```
[I 2023-08-28 16:11:39.004 ServerApp] Package notebook took 0.0000s to import  
[I 2023-08-28 16:11:39.027 ServerApp] Package jupyter_lsp took 0.0229s to import  
[W 2023-08-28 16:11:39.027 ServerApp] A `_jupyter_server_extension_points` function was not found in jupyter_lsp. Instead, a `_jupyter_server_extension_paths` function was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.  
[I 2023-08-28 16:11:39.042 ServerApp] Package jupyter_server_terminals took 0.0136s to import  
[I 2023-08-28 16:11:39.042 ServerApp] Package jupyterlab took 0.0000s to import  
[I 2023-08-28 16:11:39.093 ServerApp] Package notebook_shim took 0.0000s to import  
[W 2023-08-28 16:11:39.093 ServerApp] A `_jupyter_server_extension_points` function was not found in notebook_shim. Instead, a `_jupyter_server_extension_paths` function was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.  
[I 2023-08-28 16:11:39.094 ServerApp] jupyter_lsp | extension was successfully linked.  
[I 2023-08-28 16:11:39.103 ServerApp] jupyter_server_terminals | extension was successfully linked.  
[I 2023-08-28 16:11:39.114 ServerApp] jupyterlab | extension was successfully linked.  
[I 2023-08-28 16:11:39.124 ServerApp] notebook | extension was successfully linked.  
[I 2023-08-28 16:11:39.132 ServerApp] Writing Jupyter server cookie secret to C:\Users\조상구\AppData\Roaming\jupyter\runtime\jupyter_cookie_secret  
[I 2023-08-28 16:11:39.683 ServerApp] notebook_shim | extension was successfully linked.
```

python 가상환경_jupyter 패키지 설치하기

5. 가상환경에서 python 버전 확인 후 'python -m notebook' 명령어 입력

With the virtual environment activated, you don't need to install Python separately. The virtual environment already contains its own Python interpreter.

```
C:\Windows\system32\cmd.exe

(python_2) C:\my_project\python_2\Scripts>python --version
Python 3.11.5

(python_2) C:\my_project\python_2\Scripts>

선택 C:\Windows\system32\cmd.exe - python -m notebook

(python_2) C:\my_project\python_2\Scripts>python -m notebook
[I 2023-08-28 16:11:39.004 ServerApp] Package notebook took 0.0000s to import
[I 2023-08-28 16:11:39.027 ServerApp] Package jupyter_lsp took 0.0229s to import
[W 2023-08-28 16:11:39.027 ServerApp] A `_jupyter_server_extension_points` function was not found in jupyter_lsp. Instead,
a `jupyter_server_extension_points` function should be defined.
To access the server, open this file in a browser:
  file:///C:/Users/EC%A1%B0%EC%83%B1%EA%B5%AC/AppData/Roaming/jupyter/runtime/jpse
Or copy and paste one of these URLs:
  http://localhost:8888/tree?token=9c44aa869e1b0e5cad2d2018f04942f2c94d4b8f9b27ff41
  http://127.0.0.1:8888/tree?token=9c44aa869e1b0e5cad2d2018f04942f2c94d4b8f9b27ff41
[I 2023-08-28 16:11:39.882 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-
nodejs, javascript-typescript-langserver, jedi-language-server, julia-language-server, pyright, python-language-server,
python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-language-server, unified-language-server, v
scode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yaml-language-server
0.00s - Debugger warning: It seems that frozen modules are being used, which may
0.00s - make the debugger miss breakpoints. Please pass -Xfrozen_modules=off
0.00s - to python to disable frozen modules.
0.00s - Note: Debugging will proceed. Set PYDEVD_DISABLE_FILE_VALIDATION=1 to disable this validation.
```

url 복사하고 브라우저 실행

Jupyter notebook

The screenshot displays the Jupyter Notebook web interface in a browser. The address bar shows the URL `127.0.0.1:8888/tree`. The browser's tab bar includes several open tabs, such as "Google", "NAVER", "path analysis - Go...", "How to Calculate...", "[Windows Server]...", "한양대학교 백남학...", "한양대학교 포털", "Google", "Beautiful Soup Do...", "Boosting Algorithm...", and "대한민국 임시 의정...".


The Jupyter interface features a top menu bar with "File", "View", "Settings", and "Help". The "File" menu is currently open, revealing options like "New", "Open from Path...", "New Console for Activity", "Save", "Save As...", "Save All", "Reload from Disk", "Revert to Checkpoint...", "Download", "Save and Export Notebook As...", "Trust Notebook", "Close and Shut Down Notebook", "Log Out", and "Shut Down". A sub-menu for "New" is also visible, containing "Console", "Notebook", "Terminal", "Text File", "Markdown File", and "Python File".

Below the menu, a file browser pane shows a list of files with columns for "Last Modified" and "File Size". The files listed are:

Last Modified	File Size
4 minutes ago	72 B
52 seconds ago	72 B
42 minutes ago	2 KB

Overlaid on the bottom right of the Jupyter interface is a "Select Kernel" dialog box. It prompts the user to "Select kernel for: 'Untitled1.ipynb'" and shows "Python 3 (ipykernel)" as the selected option. At the bottom of the dialog, there are three buttons: "Always start the preferred kernel" (which is checked), "No Kernel", and "Select".

Jupyter notebook



The screenshot displays the JupyterLab interface with a notebook titled 'Untitled1'. The top bar shows the Jupyter logo, the title 'Untitled1', and the last checkpoint time 'Last Checkpoint: 7 minutes ago'. On the right, there is a 'Trusted' status indicator and a Python logo. The main menu includes 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help'. Below the menu is a toolbar with icons for file operations and execution. The notebook content consists of several code cells:

- Cell [7]:

```
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))
```
- Cell [6]:

```
import warnings
warnings.simplefilter("ignore")
```
- Cell [9]:

```
1+1
2*2
```
- Cell [9]:

```
2
```
- Cell [9]:

```
4
```
- Cell [8]:

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```
- Cell [10]:

```
print('Hello Jupyternotebook on virtual environment')
```

Hello Jupyternotebook on virtual environment

Jupyter notebook

```
[12]: np.arange(range(12))
```

```
-----  
NameError                                Traceback (most recent call last)  
Cell In[12], line 1  
----> 1 np.arange(range(12))  
  
NameError: name 'np' is not defined
```

```
[13]: ! pip install numpy
```

```
Collecting numpy  
  Obtaining dependency information for numpy from https://files.pythonhosted.org/packages/72/b2/02770e60c4e2f7e158d923ab0dea4e9f146a2dbf267fec6d8dc61d475689/numpy-1.25.2-cp311-cp311-win\_amd64.whl.metadata  
  Downloading numpy-1.25.2-cp311-cp311-win_amd64.whl.metadata (5.7 kB)  
  Downloading numpy-1.25.2-cp311-cp311-win_amd64.whl (15.5 MB)  
----- 0.0/15.5 MB ? eta -:--:--  
----- 0.0/15.5 MB ? eta -:--:--  
----- 0.1/15.5 MB 2.4 MB/s eta 0:00:07  
----- 1.3/15.5 MB 16.8 MB/s eta 0:00:01  
----- 2.3/15.5 MB 21.3 MB/s eta 0:00:01  
----- 2.8/15.5 MB 20.0 MB/s eta 0:00:01  
----- 3.2/15.5 MB 16.9 MB/s eta 0:00:01  
----- 4.2/15.5 MB 19.2 MB/s eta 0:00:01  
----- 5.3/15.5 MB 19.8 MB/s eta 0:00:01  
----- 6.0/15.5 MB 20.2 MB/s eta 0:00:01  
----- 6.3/15.5 MB 20.2 MB/s eta 0:00:01  
----- 7.3/15.5 MB 20.4 MB/s eta 0:00:01  
----- 8.4/15.5 MB 21.6 MB/s eta 0:00:01  
----- 9.0/15.5 MB 20.5 MB/s eta 0:00:01
```

Jupyter notebook

6. 파일 이름변경하고 저장하고 현재 디렉토리 확인(pwd)

Rename File

File Path

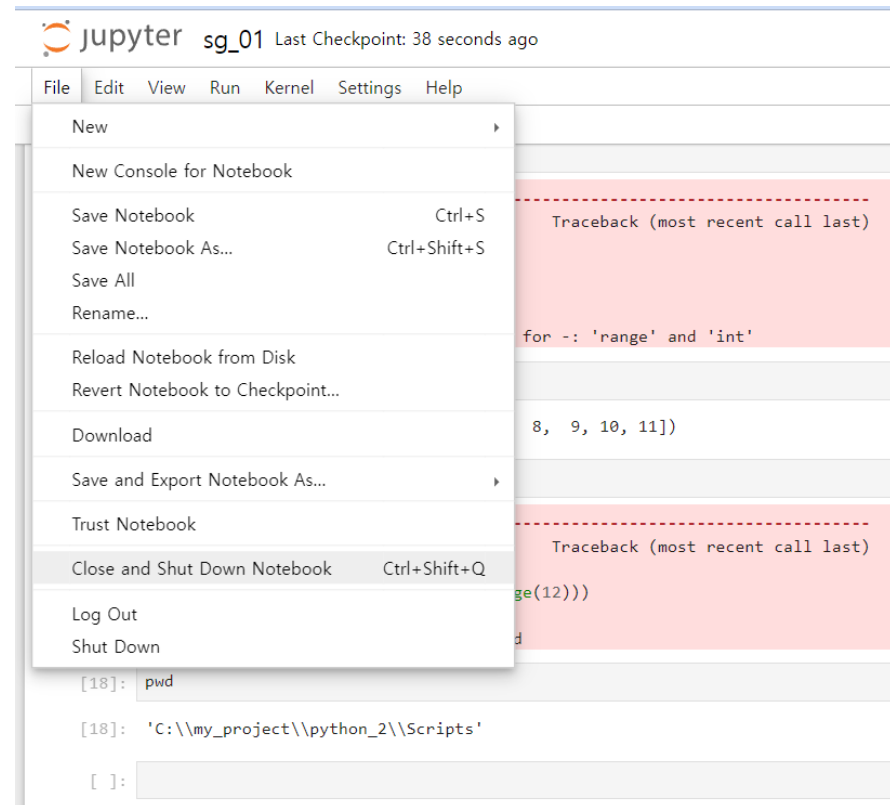
Untitled1.ipynb

New Name

sg_01.ipynb

Cancel

Rename



Jupyter notebook

7. import os 패키지 들여다 보기, 'os.tab'을 누르면 '

```
[19]: import os
```

```
[ ]: os.
```



```
[17]: pandas.DataFrame(np.arange(range(12)))
```

m	abc	module
f	abort	function
f	access	function
f	add_dll_directory	function
i	altsep	instance
f	chdir	function
f	chmod	function
f	close	function
f	closerange	function
f	cpu_count	function

```
[ ]: os.
```

Jupyter notebook

8. import glob 패키지 들여다 보기_디렉토리 상대 주소에 대한 이해

```
[20]: import glob
```

```
[21]: glob.glob('*')
```

```
[21]: ['activate',  
      'activate.bat',  
      'Activate.ps1',  
      'deactivate.bat',  
      'f2py.exe',  
      'ipython.exe',  
      'ipython3.exe',  
      'jlpmpm.exe',  
      'jsonpointer',  
      'jsonschema.exe',  
      'jupyter-consol',  
      'jupyter-dejavu',  
      'jupyter-events',  
      'jupyter-execut',  
      'jupyter-kernel',  
      'jupyter-kernel',  
      'jupyter-lab.ex',  
      'jupyter-labext
```

```
[22]: glob.glob('./**')
```

```
[22]: ['.\\activate',  
      '.\\activate.bat',  
      '.\\Activate.ps1',  
      '.\\deactivate.bat',  
      '.\\f2py.exe',  
      '.\\ipython.exe',  
      '.\\ipython3.exe',  
      '.\\jlpmpm.exe',  
      '.\\jsonpointer',  
      '.\\jsonschema.exe',  
      '.\\jupyter-console.exe',  
      '.\\jupyter-dejavu.exe',  
      '.\\jupyter-events.exe',  
      '.\\jupyter-execute.exe',  
      '.\\jupyter-kernel.exe',  
      '.\\jupyter-kernelspec.exe',  
      '.\\jupyter-lab.exe',  
      '.\\jupyter-labextension.exe',  
      ...]
```

```
glob.glob('../**')
```

```
['..\\etc',  
 '..\\Include',  
 '..\\Lib',  
 '..\\pyenv.cfg',  
 '..\\Scripts',  
 '..\\share']
```

```
[24]: glob.glob('../**/*')
```

```
[24]: ['../..\\python_2']
```

```
[31]: glob.glob('../share/*')
```

```
[31]: ['../share\\applications',  
      '../share\\icons',  
      '../share\\jupyter',  
      '../share\\man']
```