

파이썬 에디터 비교



파이썬 에디터별 특징, 추후 연결되는 작업과 연계해서

파이썬을 처음 배울 때 어떤 에디터(IDE)를 선택하느냐는 이후 작업 효율에 큰 영향을 줍니다. 여기서는 주요 파이썬 에디터들의 특징과 함께, 각 에디터가 어떤 작업에 적합한지, 그리고 향후 연결되는 개발 흐름을 정리해드릴게요.

✓ 주요 파이썬 에디터 비교표

에디터 이름	주요 특징	추천 사용자	향후 연계 작업
IDLE (기본 제공)	- 설치 시 자동 포함 - 매우 가볍고 단순	완전 초보자	- 문법 테스트 - 간단한 계산
VS Code	- 가볍고 빠름 - 다양한 확장 플러그인 - Git, HTML, JS 연동 가능	초보~고급	- 웹 개발 (Flask, Django) - AI, 데이터 분석 (Jupyter 연동 가능) - 자동화 스크립트
PyCharm	- 강력한 코드 완성 - 디버깅, 프로젝트 관리 탁월 - Community(무료) / Professional(유료)	중급 이상	- 대규모 프로젝트 - Django 웹 개발 - 객체지향 개발 및 테스트
Jupyter Notebook	- 셀 단위 실행 - 시각화/설명 혼합 - 데이터 분석 최적	데이터 분석가, AI 학습자	- Pandas, Matplotlib 시각화 - 머신러닝 모델 실험 - 보고서 작성
Thonny	- 매우 쉬운 인터페이스 - 변수 시각화 지원 - 교육용으로 최적	완전 초보자	- 문법 익히기 - 간단한 함수 실습
Spyder	- MATLAB 스타일 - 변수 확인, 콘솔 연동 - 과학/수치 계산용	과학자, 엔지니어	- 수학, 물리 모델링 - Scipy, Numpy 실험
Replit / Google Colab	- 웹 기반 실행 - 설치 필요 없음 - 클라우드 실행	입문자, 노트북 사용 어려운 환경	- 빠른 테스트 - 팀 협업 - GPU 기반 AI 실습

📌 에디터 선택과 연계 가능한 작업 흐름 예시

① 입문자 단계 (기초 문법 연습)

- 추천 에디터: IDLE, Thonny
- 향후 연결 작업: VS Code 또는 Jupyter로 확장

② 자동화 작업 or 스크립트 작성

- 추천 에디터: VS Code
- 연계 작업: 크롤링, 엑셀 자동화, 블로그 자동 게시 등

③ 데이터 분석/AI

- 추천 에디터: Jupyter Notebook, Google Colab
- 연계 작업: Pandas, Matplotlib, Scikit-learn, TensorFlow

④ 웹개발

- 추천 에디터: VS Code 또는 PyCharm
- 연계 작업: Flask, Django 프로젝트 / HTML, JS 연동

⑤ 협업 및 클라우드 실습

- 추천 에디터: Replit, Google Colab
- 연계 작업: 팀 프로젝트, Git 연동, 발표 자료 제작

에디터 선택 꿀팁

- 처음엔 Thonny 또는 VS Code로 시작, 익숙해지면 Jupyter나 PyCharm으로 확장
- 데이터 분석이면 Jupyter, 개발자처럼 프로젝트 관리하고 싶으면 VS Code 또는 PyCharm
- 설치 어려우면 Google Colab이나 Replit에서 바로 시작 가능

원하시면 사용 목적에 맞는 **에디터 설치부터 첫 코드 실행까지 실습 가이드**도 단계별로 제공해드릴 수 있어요. 어떤 작업부터 하고 싶은지 알려주시면, 맞춤 추천 드릴게요.



IDLE (기본 제공) 사용법, 차근히 설명해줘

좋아요! 지금부터 ****IDLE (아이들)****을 처음 사용하는 사람도 이해할 수 있도록 **차근차근 설명**해드릴게요.

IDLE이란?

****IDLE(Integrated Development and Learning Environment)****은 파이썬을 설치하면 함께 제공되는 **기본 파이썬 에디터**입니다. 설치 없이 바로 쓸 수 있고, 초보자에게 매우 좋은 도구예요.

IDLE 실행 방법

1단계. 파이썬이 설치되어 있어야 해요

- <https://python.org> 에서 공식 파이썬 다운로드
- 설치 시 아래 2가지를 꼭 확인하세요:
 - ✓ Add Python to PATH 체크
 - ✓ Install pip 체크

2단계. IDLE 실행하기

윈도우 기준

1. 시작 메뉴 > "IDLE" 검색 > 클릭
2. 아래와 같은 창이 뜨면 성공입니다!

```
python-repl

Python 3.x.x Shell
>>>
```

IDLE 화면 구성

구성	설명
Shell 창 (>>>)	바로 코드를 실행해볼 수 있는 대화창이에요. 계산기처럼 사용 가능
File > New File	새 창(스크립트 창)을 열어 .py 파일을 작성할 수 있어요
Run > Run Module (F5)	코드를 실행하는 메뉴예요 (스크립트 창에서 사용)

IDLE 사용 실습

1. Shell 창에서 계산해보기

```
python

>>> 1 + 2
3
>>> print("안녕하세요!")
안녕하세요!
```

➡ 계산기처럼 바로 결과가 나와요!

2. 새 파일에서 코드 작성 후 실행

- ① 메뉴: File > New File
- ② 아래 코드 작성:

```
python

# hello.py
print("파이썬 공부 시작!")
```

③ 저장 (Ctrl+S)

④ Run > Run Module (또는 F5)

→ 결과가 Shell 창에 나타납니다:

파이썬 공부 시작!

3. 간단한 입력 프로그램 만들어 보기

```
python
```

```
name = input("이름이 뭐예요? ")
print("반가워요,", name)
```

- IDLE에서 .py 파일로 저장하고 실행하면
입력 창이 나오고, 입력값에 따라 결과가 바뀌어요!

자주 하는 실수 & 팁

실수	해결 방법
SyntaxError: unexpected EOF while parsing	괄호나 따옴표가 닫히지 않았어요
Shell에서 여러 줄 코드 입력이 어렵다	New File 에서 작성하고 F5로 실행하세요
코드가 한글 깨짐	저장 시 UTF-8 인코딩 유지, 또는 IDLE 자체가 한글 지원됨

다음 단계 추천

학습 목표	방법
기초 문법 연습	IDLE Shell에서 연습
간단한 프로그램 만들기	.py 파일로 작성하고 실행
데이터 저장/불러오기	open() 함수 연습해보기
시각화/AI	VS Code 또는 Jupyter로 넘어가서 연습

원하시면 IDLE로 연습할 수 있는 **기초 미션 5개**도 만들어 드릴 수 있어요.

혹시 지금 배우고 싶은 주제가 있을까요?

예: 조건문, 반복문, 입력받기, 계산기 만들기 등.



재밌고 간단한 프로그램 예제 부탁

좋아요! 파이썬을 IDLE에서 배우는 분들을 위해, **재밋고 간단한 프로그램 예제 3가지**를 소개할게요.

👉 모두 IDLE에서 직접 실행해볼 수 있어요!

예제 1: 가위바위보 게임

```
python

import random

print("가위바위보 게임을 시작합니다!")
user = input("당신의 선택은? (가위/바위/보): ")
computer = random.choice(["가위", "바위", "보"])

print("컴퓨터의 선택:", computer)

if user == computer:
    print("비겼습니다!")
elif (user == "가위" and computer == "보") or \
      (user == "바위" and computer == "가위") or \
      (user == "보" and computer == "바위"):
    print("이겼습니다!")
else:
    print("졌습니다...")
```

● 포인트:

- `random.choice()`로 컴퓨터가 무작위 선택
- `input()`으로 사용자 입력
- `if` 문으로 결과 판단

예제 2: 오늘의 운세 뽑기

```
python

import random

name = input("이름을 입력하세요: ")
luck = ["대박! 좋은 일이 생길 거예요!",
        "오늘은 조금 조심하세요.",
        "친구와 웃을 일이 많을 거예요!",
        "예상치 못한 선물을 받을 수도!",
        "좋은 아이디어가 떠오를 거예요."]

print(name + "님의 오늘의 운세는...")
print(random.choice(luck))
```

● 포인트:

- `random.choice()`로 운세 리스트에서 1개 추출
- 이름을 입력받아 친근하게 출력



예제 3: 숫자 맞추기 게임

python

```

import random

answer = random.randint(1, 10)
tries = 0

print("1부터 10 사이의 숫자를 맞춰보세요!")

while True:
    guess = int(input("당신의 추측: "))
    tries += 1

    if guess == answer:
        print(f"정답입니다! {tries}번 만에 맞췄어요!")
        break
    elif guess < answer:
        print("더 큰 숫자입니다.")
    else:
        print("더 작은 숫자입니다.")

```

● 포인트:

- while True 반복문
- 정답을 맞출 때까지 계속 입력
- 힌트 제공 (더 크다/작다)

✨ 어떻게 실행하나요?

1. IDLE 실행
2. File > New File
3. 코드 붙여넣기
4. 저장 (Ctrl + S)
5. Run > Run Module (F5)

📌 원하시면...

- 초등학생용 동화 스타일 게임 예제
- 퀴즈 문제 만들기 프로그램
- 입력에 따라 캐릭터 성격 알려주는 심리 테스트

도 만들어 드릴 수 있어요!

"이런 스타일로 만들어줘" 라고 알려주시면 딱 맞게 제작해 드릴게요 😊





IDLE에서 안되는 작업은

아주 좋은 질문입니다!

IDLE은 파이썬 입문자에게 매우 좋은 환경이지만, **한계가 분명한 에디터**이기도 합니다.

다음은 **IDLE에서 잘 안되거나 불편한 작업들**을 정리한 목록입니다.

IDLE에서 어렵거나 불편한 작업

 작업 항목	 이유 / 문제점
웹 개발 (Flask, Django)	터미널이나 명령어 작업이 불편하고, 구조화된 프로젝트 관리가 어려움
파일 여러 개를 다루는 대형 프로젝트	파일 탭 전환이 불편하고, 디렉터리 관리 기능이 약함
깃(Git) 연동, 버전관리	기본적으로 Git 연동 기능 없음. 별도 툴 필요
자동완성, 오류 탐지, 리팩토링	VS Code나 PyCharm에 비해 기능이 매우 제한적
가상환경(venv) 설정 및 관리	복잡한 설정이 필요하고, IDLE에선 비직관적
웹 API 호출 및 디버깅	요청 결과 확인, JSON 시각화 등은 외부 도구 필요
데이터 시각화 (그래프 등)	그래프가 팝업으로는 보이지만, 노트북처럼 직관적으로 출력되지 않음
Jupyter처럼 셀 단위 실행	셀 단위 실행, 설명과 코드 혼합 사용이 불가능함
다국어(예: 한글 경로 문제)	일부 윈도우 환경에서 한글 경로, 한글 인코딩 오류 발생 가능

정리: 언제 IDLE을 벗어나야 할까?

필요 상황	추천 대안
웹개발, 프로젝트 구조 관리	VS Code, PyCharm
AI/데이터 분석, 시각화	Jupyter Notebook, Google Colab
터미널 기반 툴 실행, 서버 배포	VS Code + 터미널, 또는 Linux 환경
협업, Git 버전관리	VS Code + Git, 또는 GitHub Desktop

그럼 IDLE은 언제까지 사용해도 되나요?

IDLE은 다음과 같은 상황에 매우 적합합니다:

- 파이썬 기초 문법 배우기
- 간단한 입출력 프로그램 실습
- `input`, `print`, 조건문, 반복문, 함수 이해하기
- 파일 저장, 열기 등 단일 `.py` 파일 실습

추천: 다음 단계로 넘어가고 싶다면?

원하는 작업	추천 툴
입문자이지만 자동완성, 플러그인 쓰고 싶다	VS Code + Python 확장
데이터 분석 위주 공부	Jupyter Notebook
전문 개발자처럼 프로젝트 관리하고 싶다	PyCharm (Community 버전)

원하시면 IDLE에서 **VS Code로 넘어가는 실습 가이드**도 단계별로 정리해드릴 수 있어요!
어떤 쪽으로 확장하고 싶은지 말씀해 주세요 😊



vs코드를 가상환경에서 작업하고자해. 가상환경의 개념과 설치법 알려줘

좋아요!

지금부터 **VS Code에서 가상환경(venv)을 사용해 파이썬 프로젝트를 설정하는 방법**을

개념 → 설치 → 실습 순서로 차근차근 설명드릴게요.

👉 파이썬 초보자도 따라할 수 있게 쉽게 구성했습니다.

🧠 1. 가상환경(venv)이란?

✅ 개념 한 줄 요약

가상환경은 프로젝트마다 독립된 파이썬 설치 공간을 만드는 것

✅ 왜 필요할까?

- 프로젝트마다 필요한 **패키지 버전**이 다를 수 있음
- 가상환경을 쓰면 서로 **간섭 없이** 관리 가능
- 예:
 - 프로젝트 A: pandas 1.3.5
 - 프로젝트 B: pandas 2.2.2
 → 하나의 컴퓨터에서도 **충돌 없이** **동시 사용** 가능!

🛠️ 2. 가상환경 설치 전 준비물

- Python이 설치되어 있어야 해요
→ `python --version`으로 확인
- VS Code 설치 (확장 프로그램: Python 설치 확인)

🔧 3. 가상환경 만들기 (venv)

📌 Step 1: 원하는 폴더에 VS Code 열기

1. 작업 폴더 생성 → 예: **myproject**

2. VS Code에서 폴더 열기

VS Code > File > Open Folder > **myproject**

📌 Step 2: 터미널 열기

메뉴 > Terminal > New Terminal

📌 Step 3: 가상환경 만들기

윈도우:

```
bash

python -m venv venv
```

맥/Linux:

```
bash

python3 -m venv venv
```

➔ **venv**라는 폴더가 생기고, 이 안에 독립된 파이썬 환경이 설치돼요!

▶ 4. 가상환경 활성화하기

윈도우:

```
bash

venv\Scripts\activate
```

맥/Linux:

```
bash

source venv/bin/activate
```

➔ 아래처럼 **(venv)** 표시가 생기면 성공!

```
scss

(venv) C:\Users\so\myproject>
```

📦 5. 패키지 설치하는 가상환경 안에서!

예:

```
bash

pip install pandas
```

→ 이걸 **myproject** 안에만 적용됨 (시스템 전체에는 영향 없음)

✓ 6. VS Code에서 가상환경을 연결하는 법

VS Code에서 파이썬 가상환경을 인식하지 못하면:

1. **Ctrl + Shift + P** → 명령어 창 열기
2. **Python: Select Interpreter** 검색 → 선택
3. **./venv/Scripts/python.exe** 또는 **./venv/bin/python** 선택

🔧 7. 가상환경 확인하기

파이썬 파일(**app.py**) 작성 후 아래 코드 입력:

```
python

import sys
print("파이썬 실행 위치:", sys.executable)
```

→ 실행 시, **venv** 폴더 안의 파이썬 경로가 나오면 성공!

🛑 8. 가상환경 비활성화

작업 끝나면 아래 명령으로 종료:

```
bash

deactivate
```

📁 추가 팁: .gitignore에 venv 제외

```
text

venv/
```

→ 협업 시, 가상환경 폴더는 공유하지 않도록 설정 (보통 **requirements.txt**로 대체)

← END 요약 정리

단계	설명
개념	프로젝트별 파이썬 공간 만들기
만들기	<code>python -m venv venv</code>
활성화	윈도우: <code>venv\Scripts\activate</code>
VS Code 연동	Python: Select Interpreter
설치	<code>pip install</code> 로 라이브러리 설치
종료	<code>deactivate</code>

다음으로는

- ✓ `requirements.txt` 만들기
 - ✓ `pip freeze`로 환경 공유하는 방법도 알려드릴 수 있어요!
- 필요하면 바로 도와드릴게요.



너무 어려워. 초보자도 쉽게 할수 있는 가상환경 세팅을 알려줘.

물론이에요!

파이썬 **완전 초보자**도 쉽게 따라할 수 있는 **가상환경 세팅 방법**을

"진짜 하나씩" 설명드릴게요.

- ✓ 그림 없이도 **윈도우 + VS Code** 기준으로 **편하게 따라할 수 있게** 정리할게요.



목표

프로젝트 폴더 안에 가상환경 만들고, VS Code에서 코드 실행까지!



준비물 (설치되어 있어야 해요)

항목	확인 방법
Python	시작 > cmd → <code>python --version</code> 입력
VS Code	바탕화면에 'Visual Studio Code' 아이콘 확인
VS Code 확장: Python	VS Code 좌측 "확장" 아이콘(블록같은 모양) → Python 검색 후 설치

□ 1단계: 내 작업 폴더 만들기

1. 바탕화면에 폴더 만들기 → 이름 예: `myproject`
2. VS Code 실행 → 파일 > 폴더 열기 → `myproject` 선택

□ 2단계: VS Code에서 터미널 열기

- 상단 메뉴에서 터미널 > 새 터미널 클릭
아래쪽에 까만 창(터미널)이 열릴 거예요.

□ 3단계: 가상환경 만들기

터미널에 다음을 입력:

```
bash

python -m venv venv
```

👉 **venv**라는 이름의 폴더가 자동으로 만들어집니다.

□ 4단계: 가상환경 켜기 (활성화)

터미널에 다음을 입력:

```
bash

venv\Scripts\activate
```

입력하고 나면, 터미널 앞에 **(venv)** 표시가 생겨요!

예시:

```
scss

(venv) C:\Users\사용자\myproject>
```

□ 5단계: 파이썬 파일 만들기

1. VS Code 왼쪽 상단 → 새 파일 아이콘 클릭
2. **hello.py** 입력하고 Enter
3. 아래 코드 입력:

```
python

print("가상환경에서 실행 중입니다!")
```

□ 6단계: 실행하기

- **hello.py** 파일 안에서 오른쪽 마우스 클릭
- **Python** 파일 실행 클릭
➡ 터미널에 결과가 출력됩니다!

가상환경에서 실행 중입니다!

□ 7단계: 라이브러리 설치 (예: pandas)

```
bash

pip install pandas
```

설치 후 `import pandas` 해보면 에러 없이 작동해요.
(이건 **venv** 안에만 설치된 것이라 다른 프로젝트와 겹치지 않아요!)

□ 8단계: 작업 끝나면 가상환경 끄기

터미널에 아래처럼 입력하면 꺼져요:

```
bash

deactivate
```

🧠 가상환경 관련 정리

작업	명령어
가상환경 만들기	<code>python -m venv venv</code>
가상환경 켜기	<code>venv\Scripts\activate</code>
가상환경 끄기	<code>deactivate</code>
라이브러리 설치	<code>pip install 라이브러리명</code>

💡 진짜 쉽게 기억하는 요령

“내 프로젝트 폴더에 내 파이썬 공간을 따로 만든다!”
그래서 **겹치지 않고**, **지워도 안전하게** 유지돼요.

필요하시면 실제로 쓸 수 있는 예제(`pandas`, `matplotlib`, `flask`)도 함께 드릴게요.
어떤 걸 하고 싶으세요? 😊

- 데이터 분석용 연습
- 웹 만들기
- 간단한 게임 만들기
- 자동화 프로그램 만들기



콘다로 작업하면 어떨까?

아주 좋은 질문입니다!

방금까지는 **venv**(파이썬 기본 가상환경)를 사용했는데,
이번에는 ****Conda(콘다)****로 작업할 경우 어떤 점이 더 좋고, 어떻게 하면 되는지를
초보자 눈높이에서 쉽게 비교해드릴게요.

🧠 콘다(Conda)란?

Conda는 **가상환경 + 패키지 관리**를 모두 할 수 있는 도구입니다.

- **Anaconda** 또는 **Miniconda**를 설치하면 함께 제공돼요.
- **venv**보다 강력하고, 특히 **데이터분석/AI**에 많이 쓰입니다.

🔍 venv vs Conda 간단 비교

항목	venv	Conda
설치 방식	파이썬 기본 포함	따로 설치 (Miniconda 권장)
패키지 관리	pip install	conda install 또는 pip install
과학/AI 패키지 설치	느리고 실패 많음	빠르고 충돌 적음
가상환경 이름 지정	폴더명 기준	이름 직접 지정 가능
플랫폼 의존성	약간 있음	윈도우/리눅스 간 호환 좋음
VS Code 연동	수동 선택 필요	자동으로 인식 잘 됨

✅ 결론:

- 가볍고 간단한 프로젝트는 **venv**
- 데이터 분석, 인공지능, 복잡한 라이브러리 설치에는 **conda**가 더 편리!

🚀 초보자를 위한 Conda 설치 & 사용법 (Miniconda 기준)

1단계: Miniconda 설치

- 사이트: <https://docs.conda.io/en/latest/miniconda.html>
- 본인 운영체제에 맞는 설치파일 다운로드 후 설치
✅ 설치 시: **"Add Miniconda to PATH"** 체크!

2단계: 새 가상환경 만들기

터미널(cmd 또는 VS Code에서)

```
bash

conda create -n myenv python=3.11
```

- **myenv**: 내가 원하는 환경 이름
- **python=버전**: 원하는 버전 지정 가능

3단계: 가상환경 활성화

```
bash

conda activate myenv
```

✅ 터미널에 **(myenv)** 표시됨

4단계: 패키지 설치

```
bash

conda install pandas matplotlib
```

➔ conda는 **pandas, numpy, scikit-learn, tensorflow** 설치가 pip보다 안정적이에요!

5단계: VS Code와 연결

1. VS Code에서 **Ctrl + Shift + P**
2. **Python: Select Interpreter** 입력
3. **Conda (myenv)** 선택

이제 해당 가상환경에서 코드 실행 가능!

6단계: 가상환경 종료

```
bash

conda deactivate
```

추가 팁: pip도 같이 사용 가능!

```
bash

pip install openai
```

➔ conda 환경에서도 pip 사용 가능! 하지만 **conda 설치** 우선이 더 안전함.

✓ 추천 시나리오

상황	추천 도구
파이썬 처음 배우는 중	venv 또는 IDLE
Pandas, Matplotlib, AI 공부	conda 추천
대용량 데이터 처리 또는 GPU 사용	무조건 conda (or conda + Colab 연동)

🧠 요약

Conda의 장점
✓ 가상환경 만들기 쉽다 (conda create)
✓ 복잡한 과학/AI 패키지 설치가 안정적
✓ VS Code와 잘 연동됨
✓ 환경 이름 지정 가능 & 관리 편함
✓ pip와 함께 사용 가능

원하시면 **Conda로 Pandas + 시각화 예제**,
또는 **Conda로 Flask 웹서버 만들기도** 단계별로 만들어드릴게요.
"어떤 작업을 하고 싶은지" 말씀해주시면 거기에 딱 맞춰 도와드릴게요 😊



콘다에서의 가상환경 관련 명령어 다 알려줘, 차근히

물론입니다!
지금부터 ****Conda(콘다)****에서 사용하는 **가상환경 관련 명령어 전체**를
초보자 기준으로 하나씩 설명드릴게요.

🎯 목표: Conda 환경 만들기, 사용, 삭제까지 전체 흐름을 이해하기

🧩 1. Conda 가상환경 확인하기

◆ 설치된 가상환경 목록 보기

```
bash

conda env list
```

또는


```
bash

conda info --envs
```

✓ 결과:

```
csharp

# conda environments:
#
base                * C:\Users\so\miniconda3
myenv               C:\Users\so\miniconda3\envs\myenv
```

- * 표시는 현재 활성화된 환경

□ 2. 가상환경 만들기

◆ 기본 사용법

```
bash

conda create -n 환경이름 python=버전
```

✓ 예시:

```
bash

conda create -n pytest python=3.11
```

→ **pytest**라는 이름으로 Python 3.11 환경이 만들어짐

→ **envs/pytest** 폴더에 저장됨

● 3. 가상환경 활성화 (켜기)

```
bash

conda activate 환경이름
```

✓ 예시:

```
bash

conda activate pytest
```

→ 터미널에 **(pytest)** 표시되면 성공!

● 4. 가상환경 비활성화 (끄기)

```
bash

conda deactivate
```

→ (pytest) 사라지면 정상 종료

5. 가상환경에서 패키지 설치

```
bash

conda install 패키지이름
```

✓ 예시:

```
bash

conda install pandas matplotlib
```

6. 가상환경 삭제

◆ 가상환경 완전히 삭제

```
bash

conda remove -n 환경이름 --all
```

✓ 예시:

```
bash

conda remove -n pytest --all
```

→ **pytest** 환경과 관련된 모든 것이 삭제됨

7. 가상환경 복제

◆ 기존 환경을 새 이름으로 복사

```
bash

conda create --name 새이름 --clone 기존이름
```

✓ 예시:

```
bash

conda create --name pyclone --clone pytest
```

→ **pytest** 환경을 그대로 복사해서 **pyclone**으로 생성

8. 환경 저장 & 불러오기 (공유용)

◆ 현재 가상환경을 YAML 파일로 저장

```
bash

conda env export > environment.yml
```

→ **environment.yml** 파일이 생성됨 (패키지 목록 포함)

◆ 다른 사람/PC에서 불러오기

```
bash

conda env create -f environment.yml
```

9. 환경 내 패키지 목록 보기 / 삭제

패키지 목록 보기

```
bash

conda list
```

패키지 삭제

```
bash

conda remove 패키지이름
```

✓ 예시:

```
bash

conda remove pandas
```

전체 흐름 요약

목적	명령어
환경 목록 보기	<code>conda env list</code>
환경 만들기	<code>conda create -n myenv python=3.11</code>
환경 켜기	<code>conda activate myenv</code>
환경 끄기	<code>conda deactivate</code>
환경 삭제	<code>conda remove -n myenv --all</code>
패키지 설치	<code>conda install numpy</code>
환경 복제	<code>conda create --name newenv --clone oldenv</code>
환경 저장	<code>conda env export > environment.yml</code>
환경 불러오기	<code>conda env create -f environment.yml</code>

필요하시면 위 명령어들을 **VS Code**에서 직접 따라하는 실습 시트도 제공해드릴게요.

또는 실제로 **conda** 환경으로 데이터분석이나 AI 실습 예제를 원하시면 지금 알려주세요! 😊