



코디세이 초심자 팁

🌟 고정하기	<input type="checkbox"/>
☰ 대분류	

개요

안녕하세요! Codyssey 튜터 김형주, 이석규입니다.

첫 날 많은 분들이 프로젝트 시작에 어려움을 느끼시는 것 같아 자그마한 팁들을 준비했습니다. 아래와 같은 순서로 시작하신다면 초심자의 벽을 빠르게 넘으실 수 있을 겁니다. 그럼 화이팅!

✅ 과제로 학습하는 방법을 읽는다.

Codyssey, 그리고 더 나아가 프로그래밍의 기본은 **모르는 내용을 어떻게 학습할 것인가**입니다. 예전에는 구글 검색밖에 답이 없었지만 최근에는 Chat GPT와 같은 LLM의 등장으로 모든 내용을 쉽게 배울 수 있습니다.

구글검색, Chat GPT, 동료와의 토의/질의 단계를 거친다면 향후 튜터가 없어도 지속되는 즐거운 Codyssey 활동, 그리고 스스로 어떤 문제든 풀 수 있다는 자신감을 가지게 될 것입니다.

✅ 개발환경을 준비한다.

개발환경을 준비하는 과정도 일종의 과제입니다. “세부 개념 설명” 챕터를 참고해서 위에서 말한 “과제로 학습하는 방법” 단계에 따라서 진행해보세요.

먼저 아래의 용어들을 이해하고 설치해보세요.

- IDE(VS Code 추천), git, 사용할 언어(여기서는 python)

그리고 위의 내용을 잘 사용하기 위해서는 아래와 같은 개념을 이해하는 것이 중요합니다. 깊이 이해하지 않아도 좋습니다. 어렵듯이 개념만 알고 나중에 깊게 파고들어도 좋아요.

- shell(맥 - zsh, 윈도우 - powershell, wsl2 - bash), 환경 변수, github

| 💡 윈도우에서는 WSL2 를 사용하는 것을 강력히 추천해요!

WSL은 윈도우에서 맥의 명령어처럼 사용할 수 있게 도와주는 도구입니다. 개발환경을 통일하면 서로의 환경에서 코드 리뷰하기 편할거예요! 대부분의 개발자들이 맥/리눅스 환경에서 진행하기 때문에 윈도우에서 해당 환경으로 맞춰주는 것이 앞으로 개발할 때 도움이 많이 될 겁니다.

- Bonus) linux, wsl, cli 명령어

| 💡 shell에서는 아래와 같은 것들을 할 수 있어요

폴더, 파일 만들기, 프로그램 실행, 폴더 위치 이동, 이름 바꾸기...

IDE, git, github, terminal을 왜 사용하는지 찾아보세요.

✅ "과제로 학습하는 방법"을 상기하며 과제를 푼다.

이제 행복한 Codysey 생활을 시작하면 됩니다!

✅ 튜터의 사용법

"과제로 학습하는 방법" 의 절차를 다 따랐는데도 해결이 되지 않고 어렵다면 튜터를 불러주세요. 물론 처음부터 막혔을 때 저희를 불러주셔도 좋습니다 :) 그렇지만 혼자, 동료들과 치열하게 고민한 시간이 있을 때 훨씬 기억이 오래가고 의미있는 공부가 될 거예요!

과제로 학습하는 방법

과제 학습하실 때 아래 5단계로 나눠서 진행하신다면 짧은 시간 안에 많은 내용을 배우실 수 있을 겁니다.

1단계: 과제에서 핵심 키워드와 요구사항 뽑아내기

2단계: LLM의 도움을 받아 깊이 있게 학습하기(feat. 공식문서)

3단계: 배운 내용을 실제 코드로 만들어보기

4단계: 막히는 부분이 있다면 동료들과 함께 해결하기

이제 첫번째 과제를 예시로 해서 어떻게 학습을 할 수 있을지 알아보겠습니다!



수행과제

로그 분석을 위해 Python으로 소프트웨어를 개발해야 한다. 이를 위해서 먼저 Python을 설치해야 한다. 빠른 개발을 위해 Python 개발 도구들을 알아보고 비교해서 하나의 도구를 선정해서 설치한다. 설치가 잘 되었는지 확인 하기 위해서 'Hello Mars'를 출력해 본다. 본격적으로 로그를 분석하기 위해서 mission_computer_main.log 파일을 열고 전체 내용을 화면에 출력해 본다. 이때 코드는 main.py파일로 저장한다. (로그 데이터는 별도 제공) 파일을 처리 할 때에 발생할 수 있는 예외를 처리한다. mission_computer_main.log의 내용을 통해서 사고의 원인을 분석하고 정리해서 보고서(log_analysis.md)를 Markdown 형태로작성해 놓는다.

Python 버전은 3.x 버전으로 한다. Python에서 기본 제공되는 명령어만 사용해야 하며 별도의 라이브러리나 패키지를 사용해서는 안된다. Python의 coding style guide를 확인하고 가이드를 준수해서 코딩한다. (PEP 8 – 파이썬 코드 스타일 가이드 | peps.python.org) 문자열을 표현 할 때에는 ' '을 기본으로 사용한다. 다만 문자열 내에서 '을 사용할 경우와 같이 부득이한 경우에는 " "를 사용한다. foo = (0,) 와 같이 대입문의 = 앞 뒤로는 공백을 준다. 들여 쓰기는 공백을 기본으로 사용합니다.

제약사항

보고서는 UTF8 형태의 encoding을 사용해서 저장한다. 수행 과제에 지시된 파일 이름을 준수한다.

보너스 과제

출력 결과를 시간의 역순으로 정렬해서 출력한다. 출력 결과 중 문제가 되는 부분만 따로 파일로 저장한다.

이제 이 과제를 함께 단계별로 해결해보면서 학습 방법을 익혀보겠습니다!

과제 해결을 위한 4단계 학습법

1단계: 과제에서 핵심 키워드와 요구사항 뽑아내기

첫 번째로 해야 할 일은 과제를 꼼꼼히 읽으면서 무엇을 해야 하는지 정리하는 거예요. 마치 요리를 할 때 레시피를 먼저 읽어보는 것처럼요!

 우리 과제에서 중요한 부분들을 찾아보니:

해야 할 일들:

- Python 설치하고 개발 도구 선택하기
- 'Hello Mars' 출력해보기 (설치 확인용)
- mission_computer_main.log 파일 읽어서 화면에 출력하기
- 파일 처리할 때 예외 상황 대비하기
- 분석 결과를 log_analysis.md 파일로 정리하기

기술적으로 알아야 할 것들:

- Python 파일 입출력
- 예외 처리 (try-except)
- PEP 8 코딩 스타일
- Markdown 문법

꼭 지켜야 할 규칙들:

- Python 기본 기능만 사용 (외부 라이브러리 금지)
- 문자열은 작은따옴표(') 사용
- 코딩 스타일 가이드 준수

이렇게 정리해두니까 뭘 공부해야 할지 훨씬 명확해지죠?

2단계: LLM의 도움을 받아 깊이 있게 학습하기

이제 모르는 부분들을 AI에게 물어보면서 차근차근 배워보겠습니다. 마치 친절한 선생님께 질문하는 것처럼요!

추천 도구:

- **ChatGPT:** <https://chatgpt.com/>
- **Google AI Studio:** <https://aistudio.google.com/>

실제로 질문해볼 내용:

안녕하세요! Python 초보자인데요, 파일을 안전하게 읽고 쓰는 방법을 알려주세요. 특히 파일이 없거나 읽을 수 없을 때는 어떻게 처리해야 하나요?

간단한 예시 코드와 함께 설명해주시면 감사하겠습니다.

AI가 답변해준 내용을 보고 이해가 안 되는 부분이 있다면 계속 질문하세요:

- def, open이 뭐예요?
- () 를 붙이는건 왜 그런거예요?
- with open() 구문이 왜 좋은 방법인지 더 자세히 설명해주세요.
- 그리고 PEP 8 스타일 가이드에서 말하는 코딩 규칙들도 구체적인 예시와 함께 알려주세요.

질문할 때 팁:

- 한 번에 너무 많은 걸 묻지 말고 하나씩 차근차근
- 이해 안 되는 부분은 "더 쉽게 설명해주세요" 라고 요청하기
- 실제 코드 예시를 꼭 요청하기

공식문서를 사용하는 것도 좋은 방법입니다. 구글에 python tutorial, git tutorial 등으로 검색해보세요! 공식 문서는 대부분 영어로 되어있기 때문에 공식문서를 복사해서 Chat GPT와 함께 공부하는 것도 추천드려요.

3단계: 배운 내용을 실제 코드로 만들어보기

이제 배운 걸 직접 손으로 코딩합니다. 처음에는 간단하게 시작해서 점점 기능을 추가해나가는 방식으로 해보세요.

단계별로 코드 만들어보기:

첫 번째: 기본 틀 만들기

두 번째: 파일 읽기 기능 추가

테스트해보기:

- 로그 파일이 있을 때와 없을 때 각각 실행해보세요
- 에러 처리를 적절하게 했는지 검토해보세요
- PEP 8 스타일에 맞게 작성했는지 다시 한번 체크해보세요

4단계: 막히는 부분이 있다면 동료들과 함께 해결하기

혼자서 3단계까지 해봤는데도 잘 안 되는 부분이 있다면, 이제 주변 동료들에게 도움을 요청할 차례예요. 서로 도우면서 함께 성장하는 게 바로 동료평가 방식의 핵심이거든요!

이렇게 도움을 요청해보세요:

"안녕하세요! 저는 로그 파일 분석 과제를 하고 있는데요, 시간 역순 정렬 부분에서 막혔어요. 로그 파일에서 시간 정보를 어떻게 추출해야 할지 모르겠더라고요. 혹시 어떤 방법을 사용하셨나요?"

함께 나눌 수 있는 이야기들:

코드 리뷰 요청하기:

- "제 코드 좀 봐주실 수 있나요? PEP 8 스타일을 제대로 지켰는지 확신이 안 서요."
- "예외 처리 부분이 이렇게 해도 괜찮을까요?"

해결 방법 공유하기:

- "저는 이런 식으로 해결했는데, 다른 방법도 있을까요?"
- "마크다운 보고서는 어떤 구조로 작성하셨어요?"

어려웠던 점 나누기:


- "처음에 이 부분에서 계속 오류가 났는데, 이렇게 해결했어요!"
- "공식 문서를 찾아보니 이런 유용한 정보가 있더라고요."

동료평가 때를 위한 준비:

- 자신의 코드를 다른 사람에게 쉽게 설명할 수 있도록 연습해보세요
- 다른 사람 코드를 볼 때 어떤 질문을 할지 미리 생각해보세요
- 서로 다른 접근 방식을 이해하려고 노력해보세요

강의/책 추천

지옥에서 온 Git

 https://www.youtube.com/playlist?list=PLuHgQVnccGM_A8iwZwrGyNXCGy2LAAstXk



점프 투 파이썬

이 책은 파이썬이란 언어를 처음 접해보는 독자들과 프로그래밍을 한번도 해 본적이 없는 사람들을 대상으로 한다. 프로그래밍을 할 때 사용되는 전문적인 용어들을 알기 쉽게 풀어서 ...

 <https://wikidocs.net/book/1>



세부 개념 설명

VSCode

1. VSCode란?

- 무료·오픈소스 코드 편집기(IDE)로, Microsoft가 개발·배포합니다.
- Windows·macOS·Linux 모두 지원합니다.
- 풍부한 **확장 프로그램(Extensions)**, **Git 통합**, **디버깅** 기능을 기본 제공해 초보자·전문가 모두에게 사랑받습니다.

2. 설치 방법

먼저 자신의 운영체제를 확인하세요. 아래 단계 중 해당 OS 부분만 따라 하시면 됩니다.

2.1 Windows

1. VSCode 다운로드 페이지에서 **User Installer (x64)** 버전을 클릭해 설치 파일을 받습니다.
2. 설치 마법사 실행 → **"Add to PATH"** 와 **"Register as default editor for supported file types"** 체크박스에 체크된 상태로 **Next**.
3. 설치 완료 후 **시작 메뉴**에서 "Visual Studio Code" 실행.

4. **code** 명령 확인: *Windows Terminal/PowerShell* 에서 `code -v` 입력 → 버전이 표시되면 패스 설정 성공.

2.2 macOS

1. **Homebrew**가 설치되어 있다면 터미널에서:

```
brew install --cask visual-studio-code
```

Homebrew가 없다면 [다운로드 페이지](#)에서 **.zip** 파일을 내려받아 **Applications** 폴더로 드래그합니다.

2. `code` 명령 등록: VSCode 실행 후 **Command Palette**(`⌘⇧P`) → "Shell Command: Install 'code' command in PATH" 선택.
3. 터미널에서 `code -v` 로 버전이 보이면 완료.

2.3 Linux (Ubuntu/Debian 기준)

```
sudo apt update && sudo apt install wget gpg
wget -qO- <https://packages.microsoft.com/keys/microsoft.asc> | gpg --d
earmor > microsoft.gpg
sudo install -o root -g root -m 644 microsoft.gpg /usr/share/keyrings/
sudo sh -c 'echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/s
hare/keyrings/microsoft.gpg] <https://packages.microsoft.com/repos/vsco
de> stable main" > /etc/apt/sources.list.d/vscode.list'
sudo apt update && sudo apt install code
```

(다른 배포판은 [공식 문서](#) 참고)

3. 첫 실행 & 필수 초기 설정

1. **한국어 메뉴 사용**: 좌측 Activity Bar → **Extensions**(네모+아이콘) → `Korean Language Pack` 검색·설치 → **Restart**.
2. **테마 변경**: `Ctrl+⌘+K` → `Ctrl+⌘+T` → 원하는 Light/Dark 테마 선택.
3. **폰트 크기**: `Ctrl+⌘+,` (설정) → `Editor: Font Size` 검색 후 숫자 조정.
4. **Settings Sync(선택)**: 여러 PC 사용 시 **Manage(톱니)** → **Turn on Settings Sync....**

4. VSCode 기본 화면 한눈에 보기

영역	설명
Activity Bar	화면 좌측 아이콘 모음; 주요 기능 진입점
Explorer	파일·폴더 구조(⌘/Ctrl + Shift + E)
Search	전체 프로젝트 문자열 검색(⌘/Ctrl + Shift + F)
Source Control	Git 상태보기·커밋·푸시(⌘/Ctrl + Shift + G)
Run and Debug	코드 빌드·디버그(⌘/Ctrl + Shift + D)
Extensions	확장 설치(⌘/Ctrl + Shift + X)

▶ **Command Palette:** 모든 명령을 불러오는 범용 검색창. ⌘ ⇧ P / Ctrl Shift P 단축키.

5. 가장 많이 쓰는 단축키 10선

- ⌘/Ctrl + P : 파일 빠른 열기 (파일명 검색)
- ⌘/Ctrl + O : 파일 열기
- ⌘/Ctrl + S : 저장 / ⌘/Ctrl + K S : 전체 저장
- `⌘/Ctrl + `` : 통합 터미널 열기·닫기
- ⌃/Alt + ↑/↓ : 선택 줄 위·아래 이동
- ⇧/Shift + ⌃/Alt + F : 코드 자동 포맷
- ⌘/Ctrl + F : 현재 파일 내 검색
- ⌘/Ctrl + H : 현재 파일 내 치환
- ⌘/Ctrl + Shift + L : 동일 단어 다중 커서
- F2 : 변수·함수 이름 일괄 변경(리팩터링)

Tip: ⌘/Ctrl + K ⌘/Ctrl + S 단축키로 전체 단축키를 검색·수정할 수 있습니다.

6. 추천 확장 프로그램 (Extensions)

Extension	용도
Korean Language Pack	VSCode 한글 메뉴

GitLens — Git supercharged	Git 히스토리 시각화·분석
Python / Jupyter	파이썬 개발·노트북 지원

7. 자주 하는 질문(FAQ)

- **VSCode를 한글로 바꿨는데 일부 메뉴가 영어예요!** → 언어팩 설치 후 *Window Reload* 또는 VSCode 재시작 필요.
- **터미널 기본 셸을 바꾸고 싶어요** → Command Palette → "Terminal: Select Default Profile".
- **'code .' 명령이 안 돼요** → PATH에 code가 등록되지 않은 경우. macOS는 Shell Command 설치, Windows는 "Add to PATH" 옵션 확인.

Git & Github

1. Git & GitHub란?

구분	설명
Git	로컬에서 소스 코드의 변경 이력을 기록·관리하는 분산 버전 관리 시스템
GitHub	Git 저장소를 **클라우드(원격)** 에 호스팅하고, 이슈 관리·협업 기능을 제공하는 플랫폼

요약: Git은 도구, GitHub은 Git 저장소를 보관하고 협업할 수 있는 장소입니다.

2. Git 설치

2.1 Windows

1. [Git for Windows](#) 사이트 → **Download** 클릭 (64-bit).
2. 설치 마법사 기본값 **그대로** 진행 → "**Git Bash Here**" 옵션이 있으면 체크.
3. **Git Bash** 실행 후 버전 확인:

```
git --version # ex) git version 2.45.1.windows.1
```

2.2 macOS

- **Homebrew**로 설치가 가장 쉽습니다:

```
brew install git
git --version
```

2.3 Linux (Ubuntu/Debian)

```
sudo apt update && sudo apt install git
```

(다른 배포판은 패키지 매니저로 `git` 설치)

3. 첫 사용자 정보 설정 (한 번만)

```
git config --global user.name "이름" # 예: "Gildong Hong"
git config --global user.email "you@example.com"
```

- `global` 옵션은 **PC 전체**에 적용. 특정 저장소만 다르게 하려면 폴더에서 `global` 을 빼고 실행.

줄바꿈 설정(Windows 권장)

```
git config --global core.autocrlf true # macOS/Linux는 false 또는 input
```

4. 새 프로젝트 시작: `git init`

1. 프로젝트 폴더 생성·이동:

```
mkdir my-project && cd my-project
```

2. Git 저장소 초기화:

```
git init # .git 폴더가 생성되며 로컬 저장소 활성화
```

3. **.gitignore** 파일로 추적 제외할 파일 지정 (로그·빌드 파일 등).

예시: `node_modules/` , `*.log` , `.env`

5. 기존 저장소 복제: `git clone`

```
git clone <https://github.com/사용자명/저장소.git>
# 또는 SSH 사용 시
git clone git@github.com:사용자명/저장소.git
```

- URL은 GitHub 페이지 **Code** → **HTTPS/SSH 주소 복사**.

6. Git의 3단계 구조 이해

작업 디렉터리 (Working Directory) --git add→ 스테이징 영역 (Staging Area)
--git commit→ 로컬 저장소 (Local Repository)

- **git push** → 원격 저장소(Remote)에 반영

7. 변경 사항 추가: `git add`

명령	설명
<code>git add <파일></code>	특정 파일만 스테이지
<code>git add .</code>	현재 폴더 이하 모든 변경 스테이지
<code>git add -p</code>	변경 덩어리(hunk)별 선택적 스테이지

Tip: VSCode Source Control 패널에서 + 아이콘으로도 스테이지 가능.

8. 버전 기록: `git commit`

```
git commit -m "의미 있는 커밋 메시지"
```

- 커밋 메시지는 **현재 시제**로 간결하게: `Add login form`, `Fix typo in README` 등.

9. 원격 저장소 연결 & 동기화

1. 처음 한 번만 원격 설정:

```
git remote add origin <https://github.com/사용자명/저장소.git>
```

2. 최초 푸시(브랜치 생성 + 추적 설정):

```
git push -u origin main # GitHub 기본 브랜치가 main이라면
```

3. 이후에는 단순 `git push` 만으로 main → origin/main 동기화.

변경 가져오기: `git pull` vs `git fetch`

명령	동작	사용 시점
<code>git pull</code>	fetch + merge 한 번에	빠르게 최신 커밋 가져와 합칠 때
<code>git fetch</code>	원격 커밋만 로컬로 다운로드(병합 없음)	충돌 여부 확인 후 수동으로 병합하고 싶을 때

```
git pull origin main # 원격 변경 즉시 병합
```

```
git fetch origin # 변경 다운로드만
```

```
git merge origin/main # 필요 시 병합
```

10. 브랜치 기본(고급)

```
git branch # 브랜치 목록 표시
```

```
git checkout -b feature/ui # 새 브랜치 생성 후 전환
```

```
git switch main # 최신 버전 Git에서는 switch 권장
```

- 브랜치 병합: `git merge feature/ui`
- 원격 브랜치 푸시: `git push -u origin feature/ui`

11. 필수 명령어 1페이지 요약

단계	명령어 & 예시	기억 포인트
저장소 준비	<code>git init</code> / <code>git clone URL</code>	새로 만들기 / 복제
스테이지	<code>git add .</code> , <code>git add <파일></code>	커밋 후보 선정
커밋	<code>git commit -m "msg"</code>	이력 저장

원격	<code>git remote add origin URL</code> , <code>git push -u origin main</code>	GitHub 연결 및 첫 푸시
업데이트	<code>git pull</code> , <code>git fetch</code>	최신 내용 가져오기
상태	<code>git status</code> , <code>git log --oneline --graph</code>	현황·이력 확인

12. 자주 하는 질문(FAQ)

- 커밋 메시지를 수정하고 싶어요!

→ `git commit --amend` 후 새 메시지 입력 (이미 푸시한 커밋은 강제 푸시 필요 주의).

- 깃허브에 2FA(2단계 인증) 때문에 푸시가 안 돼요

→ **Personal Access Token**을 비밀번호 대신 입력 or SSH 키 사용.

- 충돌(conflict)이 나면 어떻게 하나요?

→ 파일 내 `<<<<<<` , `=====` , `>>>>>>` 구역을 수동 편집해 정리 → `git add` 후 `git commit` .

- 브랜치를 삭제하고 싶어요?

→ 로컬: `git branch -d feature/ui` , 원격: `git push origin --delete feature/ui` .

Windows 환경 변수 설정 가이드

1. 환경 변수 & PATH란?

용어	설명
환경 변수	운영체제·프로세스가 참조하는 문자열 값. 예) <code>PATH</code> , <code>JAVA_HOME</code>
PATH	실행 파일(예: git.exe)을 찾을 때 탐색할 디렉터리 목록 을 <code>;</code> 로 구분해 저장한 변수

요약 : git 명령이 동작하려면, git.exe가 들어 있는 폴더가 PATH에 포함되어야 합니다.

2. Git 설치 경로 확인

- 기본 설치 위치

◦ 64-bit Windows: `C:\Program Files\Git\cmd`

- 32-bit Windows: `C:\Program Files (x86)\Git\cmd`
- `git.exe` 가 실제로 존재하는지 탐색기로 확인해 둡니다.

3. GUI로 PATH 추가하기 (추천)

| 관리자 권한 불필요, Windows 10/11 공통

1. 시작 → “환경 변수” 검색 → “시스템 환경 변수 편집” 실행.
2. 고급 탭 → 환경 변수(N)... 버튼 클릭.
3. 사용자 변수 또는 시스템 변수 중 PATH 선택 → 편집(E)....
 - 사용자 변수: 현재 사용자만 적용.
 - 시스템 변수: 모든 사용자·서비스에 적용(관리자 권한 필요).
4. 새로 만들기(N) 클릭 → Git 설치 경로 `C:\Program Files\Git\cmd` 입력.
5. 확인 → 모든 창 확인/닫기.
6. 이미 열려 있던 cmd/PowerShell/IDE를 전부 종료했다가 다시 열고:

```
git --version
```

버전 정보가 표시되면 완료!

| 팁 : Windows 11에서는 설정 → 시스템 → 정보 → 고급 시스템 설정 경로로도 이동 가능합니다.

5. Python 개발 도구 환경 변수 예시

도구	필수 변수	설정 예시
Python	PATH	<code>C:\Python312\;C:\Python312\Scripts\</code>

| 순서 중요 : Windows는 PATH 항목을 앞에서부터 검색합니다. 충돌 가능성이 있다면 원하는 버전을 상위에 배치하세요.

6. PATH 관련 흔한 문제 & 해결법

증상	원인	해결
git : 인식되지 않는 명령입니다	터미널이 새 PATH를 못 읽음	모든 터미널·IDE 재시작 또는 PC 재부팅
PATH에 동일한 폴더가 두 번	중복 입력	편집 창에서 중복 행 삭제
길이 제한 오류	PATH 길이 2048자를 초과	불필요한 경로 정리 or Win 10 1607↑ LongPathsEnabled 레지스트리 설정

7. 정리

1. Git 설치 경로 ...\Git\cmd 확인.
2. 시스템/사용자의 PATH에 해당 경로 추가.
3. 새 터미널에서 git --version 으로 정상 동작 확인.
4. 다른 개발 도구도 비슷한 방식으로 PATH 또는 전용 변수(JAVA_HOME 등) 설정.

환경 변수는 OS 재시작 없이 적용되지만, 이미 실행 중인 터미널·프로그램은 재시작해야 새 설정을 읽어옵니다.

WSL 사용 가이드

1. WSL이란?

구분	설명
WSL 1	Windows 커널과 POSIX 호환 레이어로 리눅스 시스템 호출을 변환. 속도 빠름, 완전한 커널 기능 일부 제한
WSL 2	경량 VM에 실제 Linux 커널 실행. Docker·inotify 등 100% 리눅스 호환, I/O 속도 개선

TIP : 신규 설치는 WSL 2 사용이 권장됩니다.

2. 설치 전 체크리스트

1. Windows 10 버전 2004 이상 또는 Windows 11 (빌드 22000+)인지 확인.
설정 → 시스템 → 정보 → Windows 사양에서 빌드 확인.

2. BIOS/UEFI에서 **가상화(Virtualization)** 기능이 **Enabled**인지 확인 (대부분 기본 활성화).
3. 관리자 권한 PowerShell 실행 준비 (`Win + X` → `Windows Terminal(관리자)`).

3. WSL 설치 (가장 간단한 방법)

```
wsl --install
```

- Windows가 **WSL 필요한 옵션**(가상 머신 플랫폼·Linux 서브시스템) 활성화 후 재부팅 요청.
- 재부팅 후 **Ubuntu LTS**가 자동 설치·셸 창 열림 → 사용자명/비밀번호 입력해 초기 설정 완료.

구버전 Windows 10은 수동 명령:

```
dism.exe /online /enable-feature /featurename:Microsoft
-Windows-Subsystem-Linux /all /norestart
dism.exe /online /enable-feature /featurename:VirtualMa
chinePlatform /all /norestart
wsl --set-default-version 2
```

4. 배포판 관리 기본

명령	기능
<code>wsl --list --online</code>	설치 가능한 배포판 목록 조회
<code>wsl --install -d <배포판></code>	선택 배포판 설치(예: <code>Ubuntu-24.04</code>)
<code>wsl --list --verbose</code>	설치된 배포판·WSL 버전 확인
<code>wsl --set-version <배포판> 2</code>	WSL 1→2 업그레이드
<code>wsl --set-default <배포판></code>	기본 배포판 지정

5. Windows Terminal & WSL 접속

1. **Microsoft Store**에서 *Windows Terminal* 설치(Win 11 기본 탑재).

2. Terminal 실행 → ▼ 메뉴에서 *Ubuntu* 클릭 → 리눅스 셸 진입.

3. 단축키

- 새 탭: `Ctrl + Shift + T`
- 탭 전환: `Ctrl + Tab` / `Ctrl + Shift + Tab`

파일 경로 이해 :

- Windows → 리눅스: `C:\\Users` → `/mnt/c/Users`
- 리눅스 → Windows: `/home/<사용자>` 는 Windows 탐색기 주소창에 `\\\\ws\\$\\Ubuntu\\home\\<사용자>` 입력

6. VSCode ↔ WSL 연동

6-1 Remote – WSL 확장 설치

1. Windows VSCode 실행(`code`).
2. **Extensions** (`Ctrl + Shift + X`)에서 **"Remote - WSL"** 검색·설치.

6-2 WSL에서 VSCode 열기 (권장)

```
# WSL 셸(ubuntu)에서 프로젝트 폴더로 이동한 뒤  
code .
```

- 처음 실행 시 Server 컴포넌트 자동 설치 후 VSCode 창이 **녹색 WSL 표시**로 열립니다.
- **통합 터미널**(`Ctrl + ```) 이 자동으로 WSL 셸을 사용.

6-3 Windows 적용 설정

- 기본 통합 터미널 프로필을 WSL로:
`Ctrl + ,` → *Features > Terminal > Integrated: Default Profile Windows*
→ Ubuntu 선택.
- **파일 탐색기 우클릭** → **'Open with Code'** 시 Windows 경로로 열리므로, 리눅스에서 작업할 폴더는 VSCode에서 *Remote Explorer > WSL*로 열기 권장.

7. WSL 2 내부 개발 환경 설정 예시

```
# WSL 셸에서
sudo apt update && sudo apt upgrade -y

# Git, build-essential, curl 등 설치
sudo apt install -y git build-essential curl

# Node (nvm) 예시
curl -o- <https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh
> | bash
source ~/.bashrc
nvm install --lts

# Python 예시
sudo apt install -y python3 python3-pip
```

- **SSH 키**를 WSL 안에서 생성하면 GitHub SSH 인증이 Windows와 분리돼 보안 향상.
- 필요 시 Windows VSCode에서 터미널 분할(**Ctrl + Shift + 5**)로 Windows cmd ↔ WSL 셸 동시에 사용.

8. 자주 묻는 질문(FAQ)

질문	답변
WSL 2가 Docker Desktop 없이도 컨테이너 실행되나요?	가능합니다. <i>Docker Engine for Linux</i> 설치 후 VSCode Remote - Containers 사용.
디스크 용량이 부족해요!	<code>wsl --shutdown</code> 후 VHDX 파일 압축(Optimize-VHD) 또는 배포판 이동(<code>export</code> / <code>import</code>).
WSL에서 Windows exe 실행 방법?	<code>notepad.exe</code> , <code>explorer.exe</code> . 처럼 확장자 포함 호출.
리눅스에서 Windows PATH 공유?	<code>/mnt/c/Windows/System32</code> 등이 기본 포함. 추가 경로는 <code>.bashrc</code> 에 export.