



cpp:: 코어\_개발환경\_설정

readable\_code::

# readable\_code :: cpp :: 코어\_개발환경\_설정

1. 강의 개요●
2. 개발환경●
3. 개발환경 필수 프로그램●
4. Version Control System(VCS)●
5. 코딩폰트●
6. Integrated Development Environment(IDE)●
7. 개발환경 설정 자동화 스크립트●

# 강의 개요

강의 설명

목차

# 강의 설명

## 강의 설명

- 무엇을 배울까요?◦

- OS에 상관 없는 C++ 개발환경 설정 방법(MacOS/Ubuntu)◦
- Version Control System(VSC, Github) 설정 방법◦
- 개발 효율을 높이기 위한 코딩폰트 설정 방법◦
- C++ 개발을 위한 Integrated Development Environment(IDE, Visual Studio Code) 설정 방법◦
- 스크립트를 통한 개발환경 설정 자동화◦

- 누구에게 필요할까요?◦

- 개발환경이 무엇인지 알지 못하는 분◦
- C++ 개발을 처음 시작하는 분◦
- PC를 포맷 할 때마다 개발환경을 다시 설정하는 것이 너무 힘들었던 분◦
- 보다 효율적으로 개발환경을 관리 하기를 원하시는 분◦
- 여러 플랫폼에서 일관적인 개발환경을 구성 하기를 원하시는 분◦
- C++ 개발 프로세스를 최적화 하고 싶은 분◦

## 강의 설명

- 무엇을 얻을 수 있을까요?

- 다중 OS(MacOS/Ubuntu)에 대한 C++ 필수 개발환경 설정 방법
- C++ 개발에 최적화 된 IDE 설정과 확장 기능
- 코드의 가독성을 높일 수 있는 코딩폰트 적용 방법
- C++ 필수 개발환경을 실행 한번으로 설치할 수 있는 자동화 스크립트(MacOS/Ubuntu)

- Keywords

- Development Environment
- Compiler
- Build System
- Version Control System
- IDE
- Coding Font
- Automation Script

# 목차

# 목차

## 1. 개발환경

- 1. 개발환경 개요
- 2. 개발환경 정의

## 2. 개발환경 필수 프로그램

- 1. C++ compiler 개요
- 2. Build system 개요
- 3. 필수 프로그램 설치 on MacOS
- 4. 필수 프로그램 설치 on Ubuntu
- 5. 예제 코드 빌드 on terminal

## 3. Version Control System(VCS)

- 1. VCS 개요
- 2. Git & Github CLI 설치 on MacOS
- 3. Git & Github CLI 설치 on Ubuntu

## 4. 코딩폰트

- 1. 코딩폰트 개요
- 2. 코딩폰트 설치 on MacOS
- 3. 코딩폰트 설치 on Ubuntu

## 5. Integrated Development Environment(IDE)

- 1. IDE 개요
- 2. VSCode 설치 on MacOS
- 3. VSCode 설치 on Ubuntu
- 4. VSCode 기본 설정
- 5. C++ 개발을 위한 VSCode 확장기능 설정
- 6. VSCode 권장 확장 기능
- 7. VSCode 테마 설정
- 8. 예제 코드 빌드 using VSCode
- 9. VSCode로 Git 사용하기

## 6. 개발환경 설정 자동화

- 1. 자동화 스크립트 개요
- 2. 개발환경 설정 스크립트 for MacOS
- 3. 개발환경 설정 스크립트 for Ubuntu

# 개발환경

개발환경 개요▣

개발환경 정의▣

# 개발환경 개요

## 개발환경 개요

What is the development environment?

# 개발 환경?

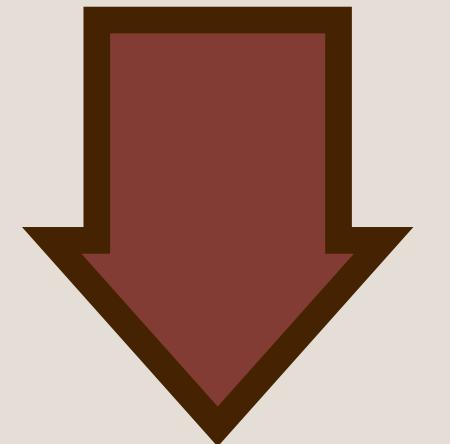
## 개발환경 개요

What is the development environment?

# SW 개발을 하기 위해 필요한 도구들의 모음

프로그래밍 언어

OS



버전 관리

Native

vs

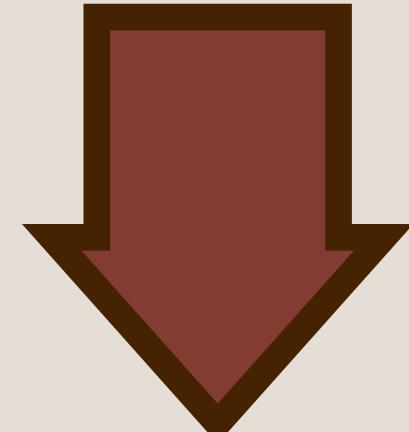
Virtual

등등등...

## 개발환경 개요

What is the development environment?

# SW 개발을 하기 위해 필요한 도구들의 모음



프로그래밍 언어  
OS  
개발 도구  
VirtualBox  
IDE  
tive  
vs  
Virtual

환경 관리  
환경  
등등...

## 개발환경 개요

Components required for a C++ development environment

Optional

Compiler  
or  
Interpreter

Build System

Version Control  
System

Language

Virtual OS

## 개발환경 개요

Components required for a C++ development environment

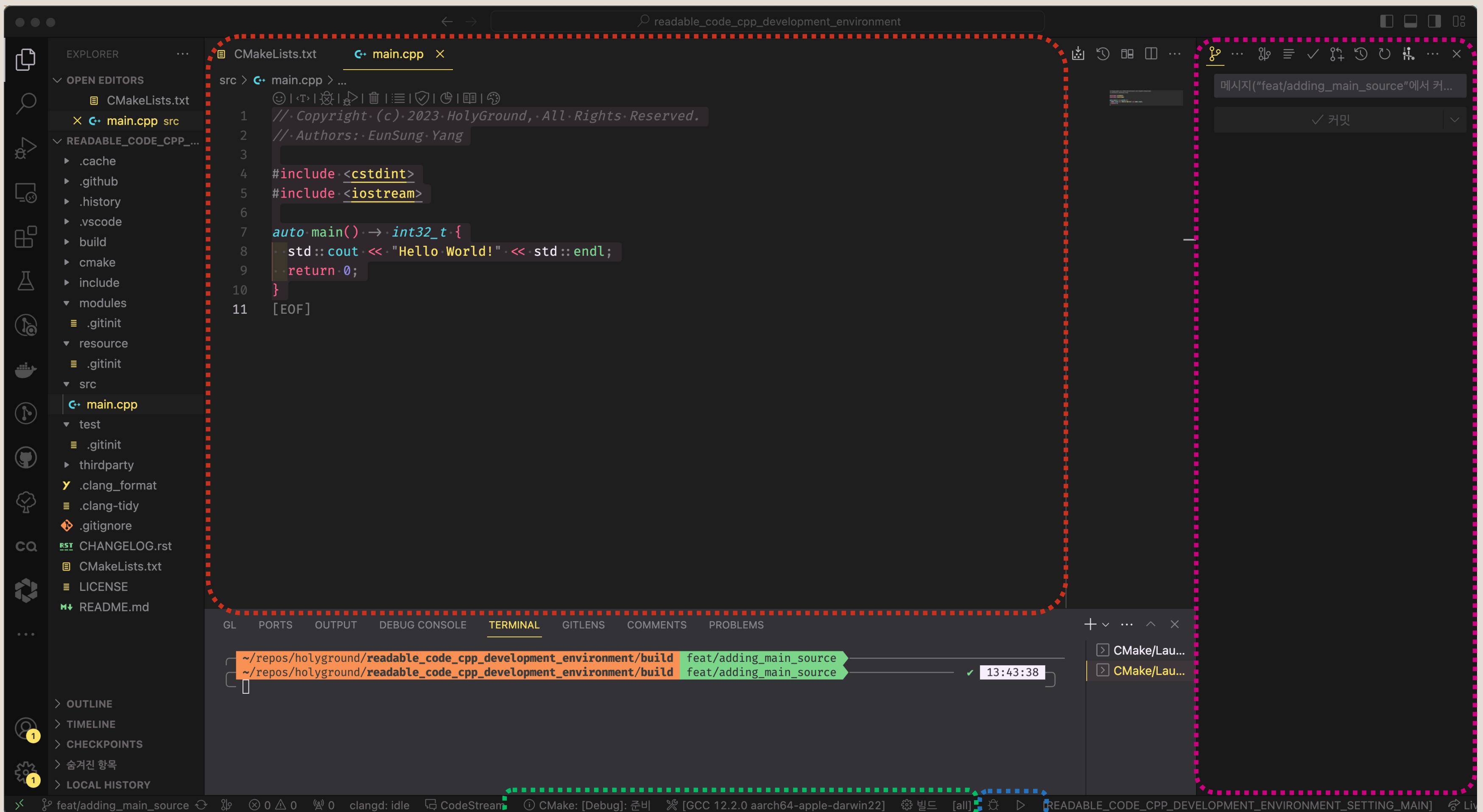
Optional

# Integrated Development Environment

## 개발환경 개요

코드 편집기 또는 통합 개발 환경(Integrated Development Environment, IDE)

- 소스 코드를 작성하고 편집하는 도구
- IDE는 코드 작성, 디버깅, 테스팅 및 프로젝트 관리 기능을 통합하는 종합적인 환경을 제공

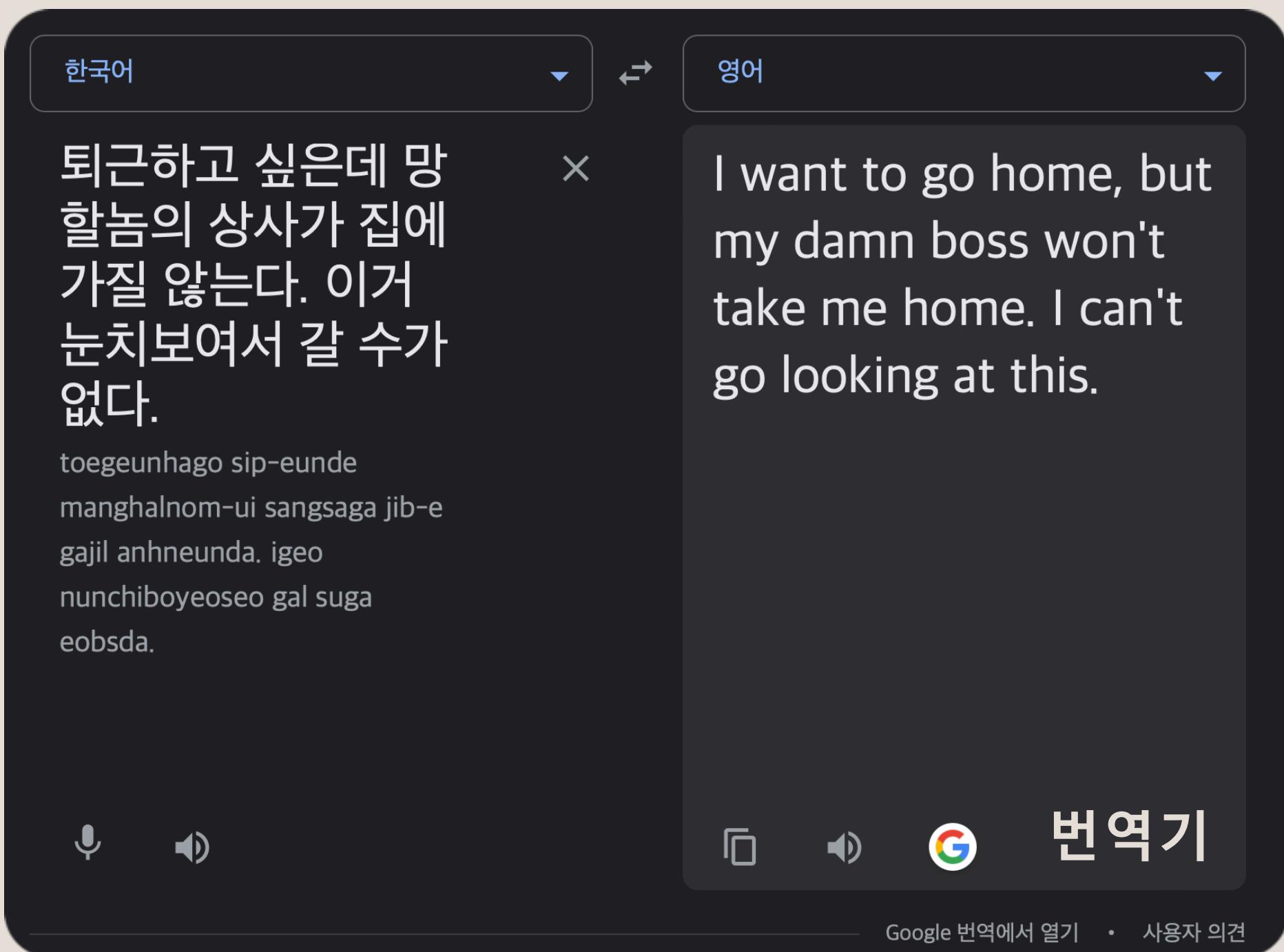


## 개발환경 개요

### 컴파일러/인터프리터

- 컴파일 언어의 경우 컴파일러는 소스 코드를 기계 코드 또는 중간 코드로 변환●
- 인터프리터 언어의 경우 인터프리터가 코드를 직접 실행●

### Compiled Language



### Interpreted Language

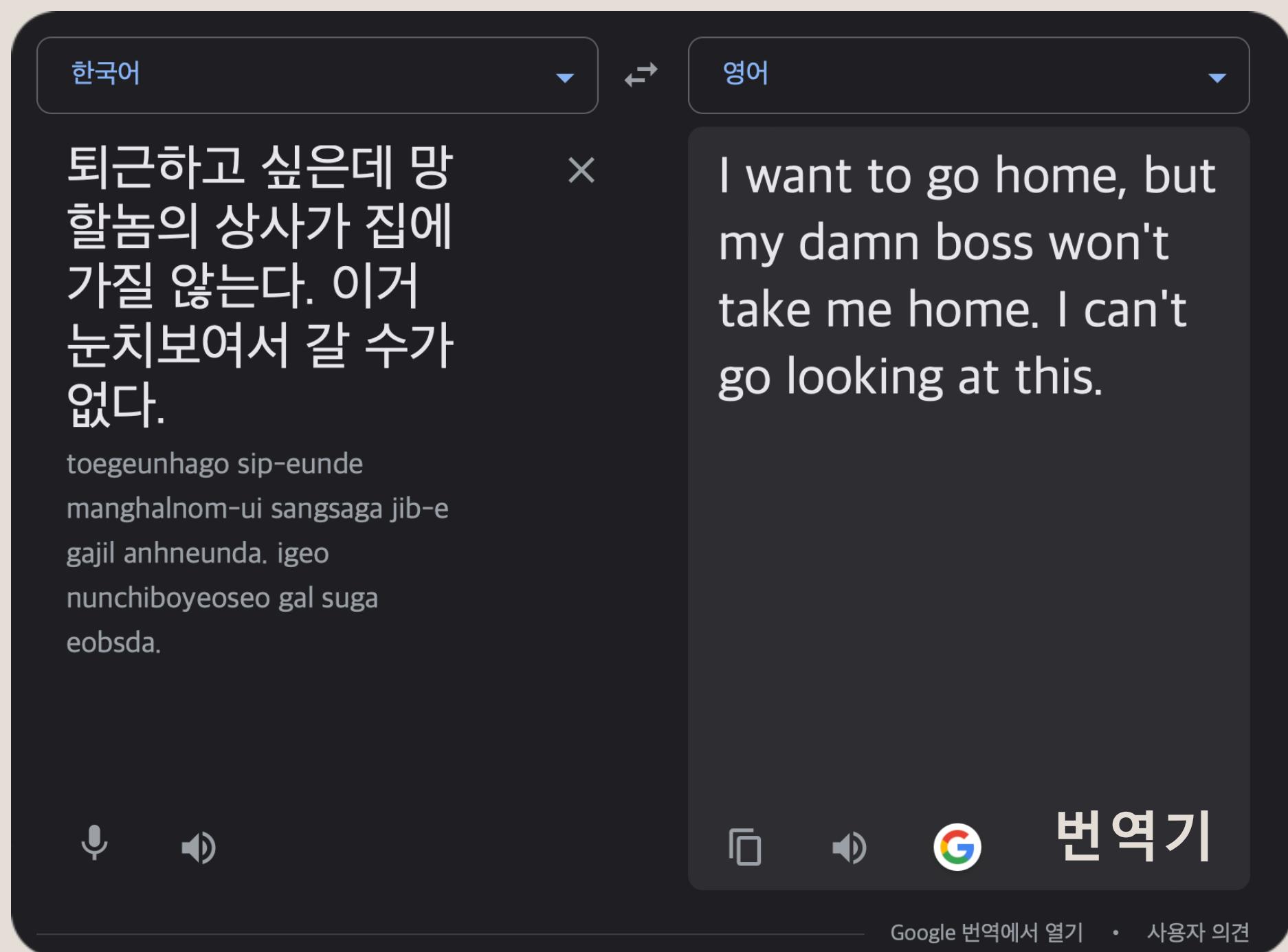


## 개발환경 개요

### 컴파일러/인터프리터

- 컴파일 언어의 경우 컴파일러는 소스 코드를 기계 코드 또는 중간 코드로 변환●
- 인터프리터 언어의 경우 인터프리터가 코드를 직접 실행●

### Compiled Language



### Interpreted Language



## 개발환경 개요

### 빌드 도구(Build System)

- 소스 코드를 컴파일하고 실행 가능한 애플리케이션으로 빌드하는 도구◦
- 빌드 도구는 프로젝트 구조 및 의존성 관리를 담당◦

```
set(${PACKAGE_NAME}_RESOURCE_PATH "${${PACKAGE_NAME}_PATH}/resource")
set(${PACKAGE_NAME}_MODULE_PATH "${${PACKAGE_NAME}_PATH}/module")
set(${PACKAGE_NAME}_THIRDPARTY_PATH "${${PACKAGE_NAME}_PATH}/thirdparty")

message(STATUS "PACKAGE_NAME: ${PACKAGE_NAME}")
message(STATUS "PACKAGE_NAME: ${PACKAGE_VERSION}")
message(STATUS "${PACKAGE_NAME}_PATH: ${${PACKAGE_NAME}_PATH}")
message(STATUS "${PACKAGE_NAME}_SOURCE_PATH: ${${PACKAGE_NAME}_SOURCE_PATH}")
message(STATUS "${PACKAGE_NAME}_INCLUDE_PATH: ${${PACKAGE_NAME}_INCLUDE_PATH}")
message(STATUS "${PACKAGE_NAME}_TEST_PATH: ${${PACKAGE_NAME}_TEST_PATH}")
message(STATUS "${PACKAGE_NAME}_RESOURCE_PATH: ${${PACKAGE_NAME}_RESOURCE_PATH}")
message(STATUS "${PACKAGE_NAME}_MODULE_PATH: ${${PACKAGE_NAME}_MODULE_PATH}")
message(STATUS "${PACKAGE_NAME}_THIRDPARTY_PATH: ${${PACKAGE_NAME}_THIRDPARTY_PATH}\n")

add_subdirectory(${${PACKAGE_NAME}_THIRDPARTY_PATH})
add_subdirectory(${${PACKAGE_NAME}_MODULE_PATH})

add_executable(MAIN src/main.cpp)
target_link_libraries(MAIN
    CMAKE_GROUND_EXAMPLE
)
target_compile_options(MAIN PRIVATE
    -Wall -Wextra -Wpedantic -Werror
)
[EOF]
```

## 개발환경 개요

### 버전 관리 시스템(Version Control System, VCS)

- 소스 코드 변경을 추적하고 협업을 지원하기 위한 도구
- 코드의 이력 관리와 협업을 향상

The screenshot shows a GitHub commit history for a repository. The first commit, 'feat/Added main function printing str' by 'movingChurch' (committed 1 hour ago), adds a main function to print 'Hello World!'. The commit message is 'feat/Added main function printing str'. The commit hash is 92f46da.

Showing 3 changed files with 14 additions and 4 deletions.

**CMakeLists.txt**

```
@@ -34,7 +34,7 @@ message(STATUS "${PACKAGE_NAME}_THIRDPARTY_PATH: ${${PACKAGE_NAME}_THIRDPARTY_PA
34 # add_subdirectory(${${PACKAGE_NAME}_THIRDPARTY_PATH})
35 # add_subdirectory(${${PACKAGE_NAME}_MODULE_PATH})
36
37 - # add_executable(MAIN src/main.cpp)
38 - # target_compile_options(MAIN PRIVATE
39 - #   -Wall -Wextra -Wpedantic -Werror
40 - # )
```

**.gitinit**

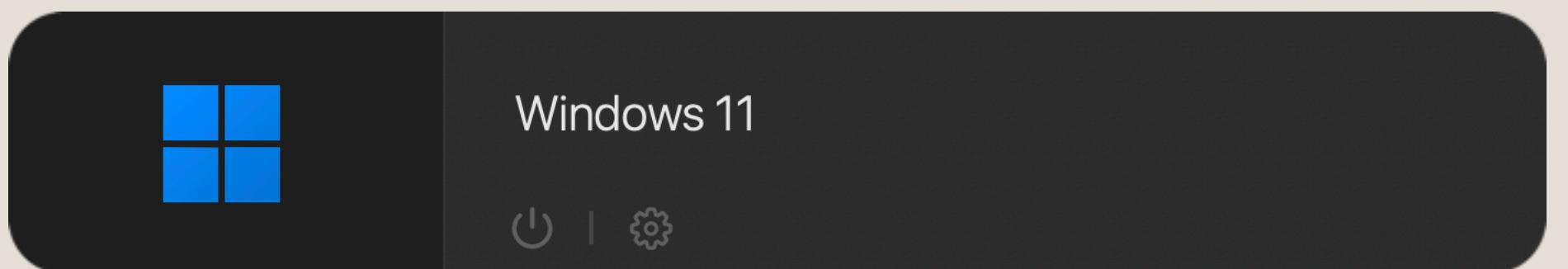
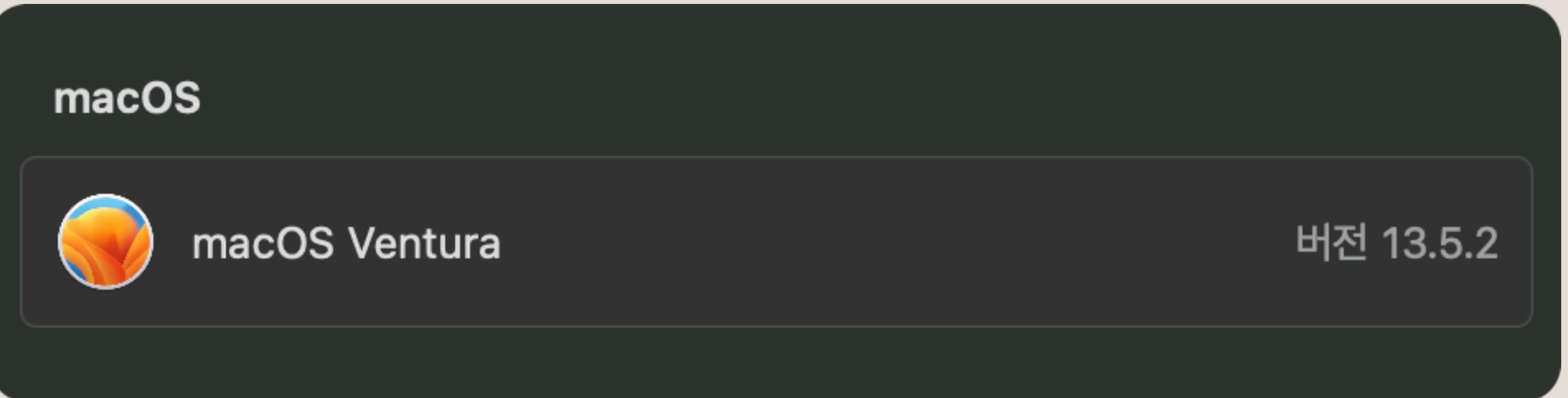
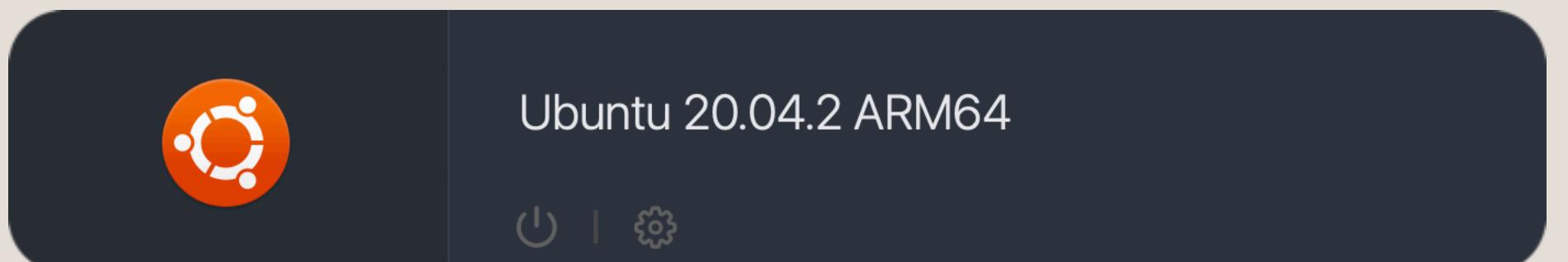
```
Empty file.
```

**src/main.cpp**@@ -0,0 +1,10 @@
1 + // Copyright (c) 2023 HolyGround, All Rights Reserved.
2 + // Authors: EunSung Yang
3 +
4 + #include <cstdint>
5 + #include <iostream>
6 +
7 + auto main() -> int32\_t {
8 + std::cout << "Hello World!" << std::endl;
9 + return 0;
10 + }

## 개발환경 개요

### 운영 체제(Operating System, OS)

- 개발 환경이 실행되는 기본 운영 체제
- Windows, MacOS, Linux 등이 흔히 사용됨
- 개발환경을 docker, anaconda, WSL과 같은 가상환경에서 구성하는 경우도 있음



# 개발환경 정의

## 개발환경 정의

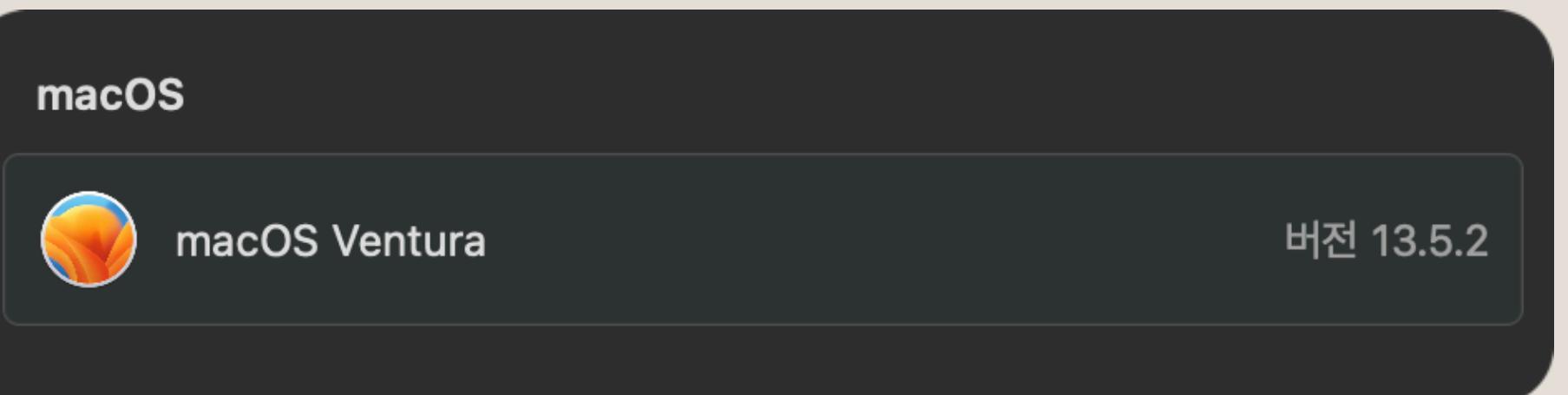
Definition of the development environment for the lecture

- **Hardware**

- Macbook Pro 16
  - Architecture : M1 Max(ARM64)
  - CPU : 10cores
  - RAM : 64GB

- **Software**

- Native OS
  - MacOS 13



## 개발환경 정의

Definition of the development environment for the lecture

### Coding Font

GCC/Clang

CMake

Git/Github CLI

C++

MacOS 13

Ubuntu 22.04

## 개발환경 정의

Definition of the development environment for the lecture

Container

Integrated

Development

Environment

MacOS 10.8 / Ubuntu 12.04

# 개발환경 정의

## Definition of the development environment for the lecture

# Visual Studio Code with Clang

## 개발환경 정의

Definition of the development environment for the lecture

- OS
  - Native OS
    - MacOS 13
  - Virtual OS
    - Ubuntu 22.04 LTS
- Language
  - C++ STD 17
- Compiler
  - Clang latest-release
  - GCC latest-release
- Build System
  - CMake latest-release
- VCS
  - Git latest-release
  - Github CLI latest-release
- IDE
  - VSCode latest-release

# 개발환경 필수 프로그램

C++ compiler 개요◦

Build system 개요◦

필수 프로그램 설치 on MacOS◦

필수 프로그램 설치 on Ubuntu◦

예제 코드 빌드◦

# C++ compiler 개요

## C++ compiler 개요

Programming?



자연어

기계어



컴퓨타

닝겐

## C++ compiler 개요

Programming?



닝겐

자연어

- 사람의 언어
- 컴퓨터는 이해하기 어려운 언어

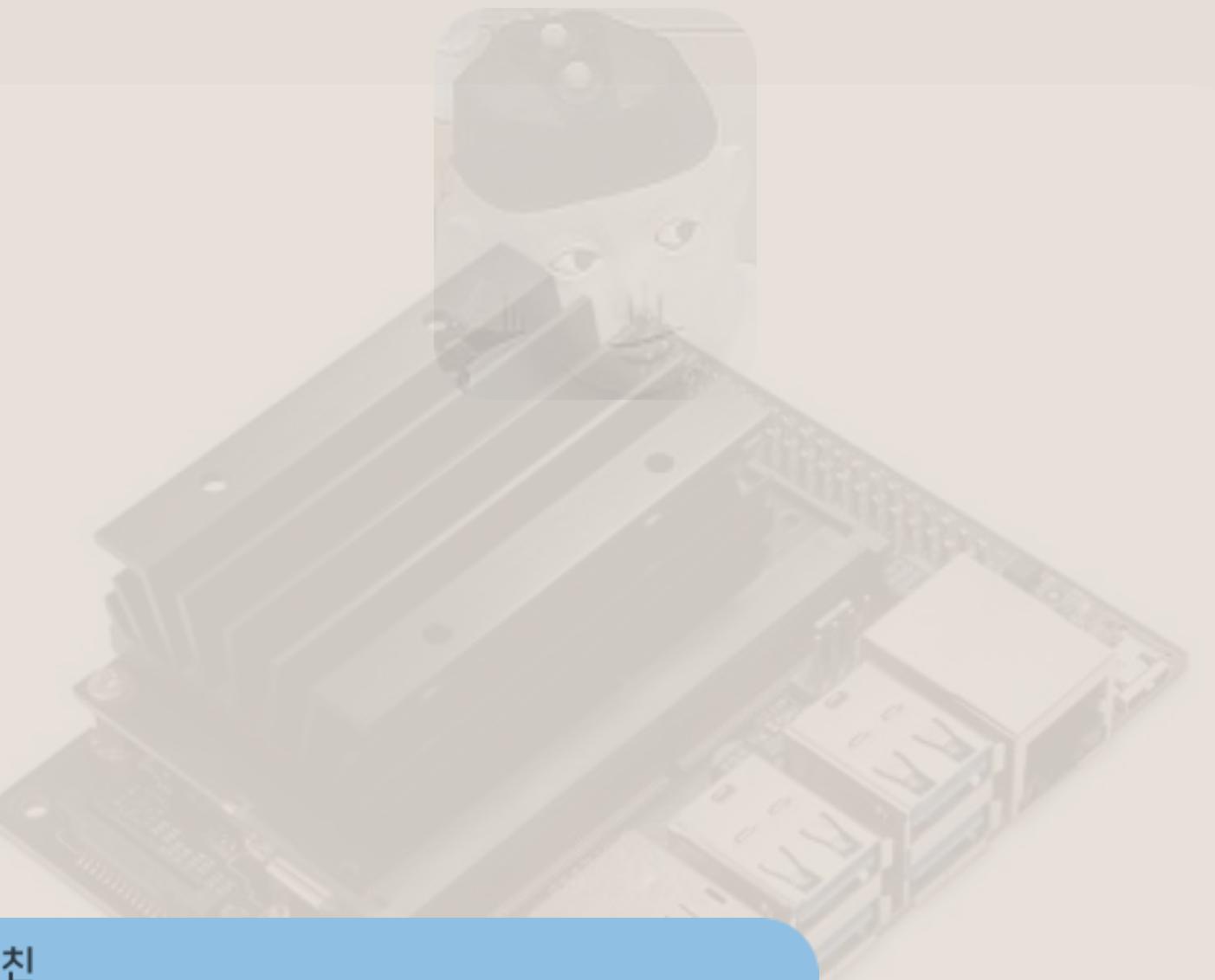
기계어

구남친

구남친  
자니?  
오전 2:14  
자나보네..  
오전 2:46  
잘자.  
오전 3:04

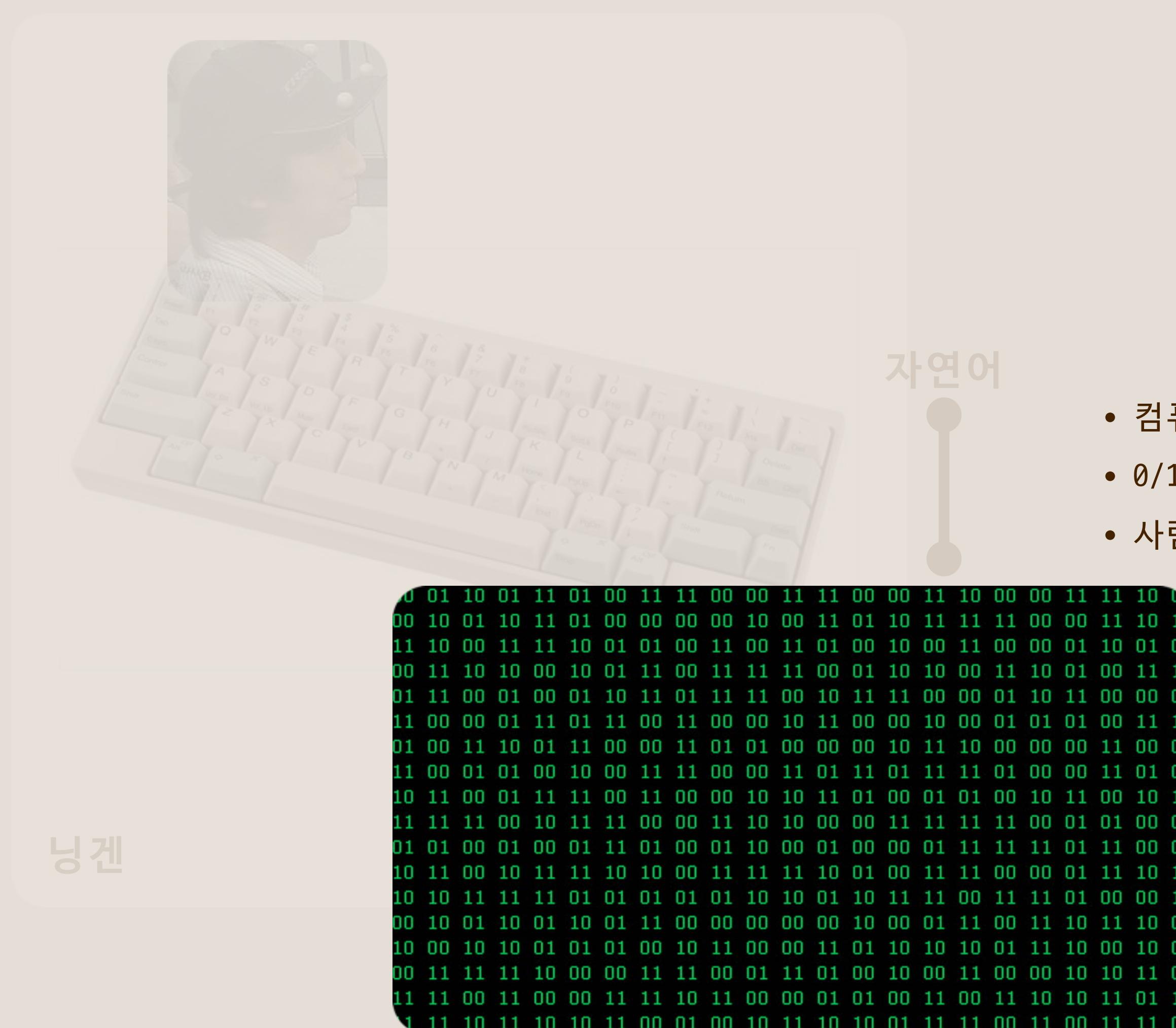
자니?

콤퓨타



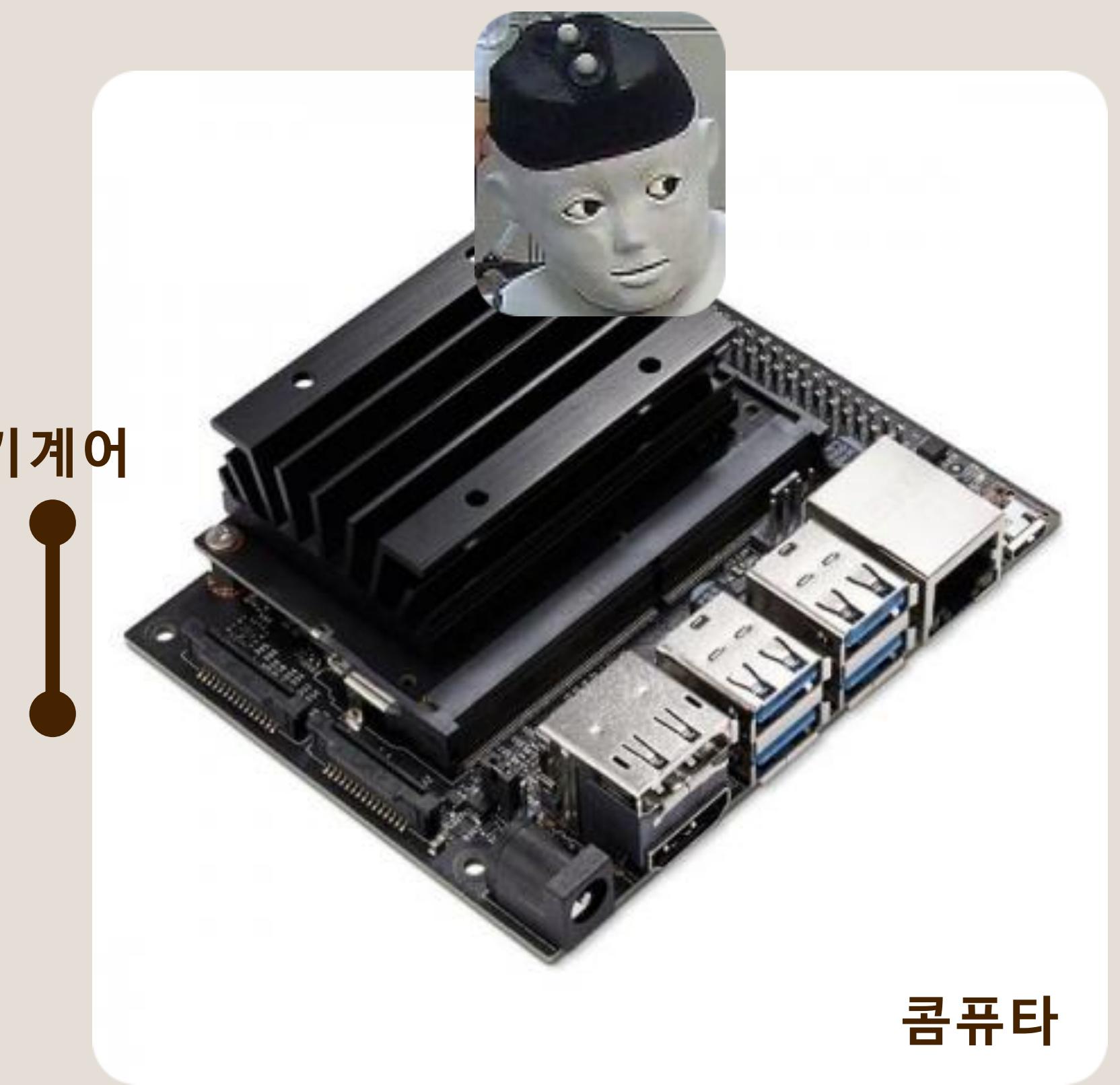
## C++ compiler 개요

Programming?



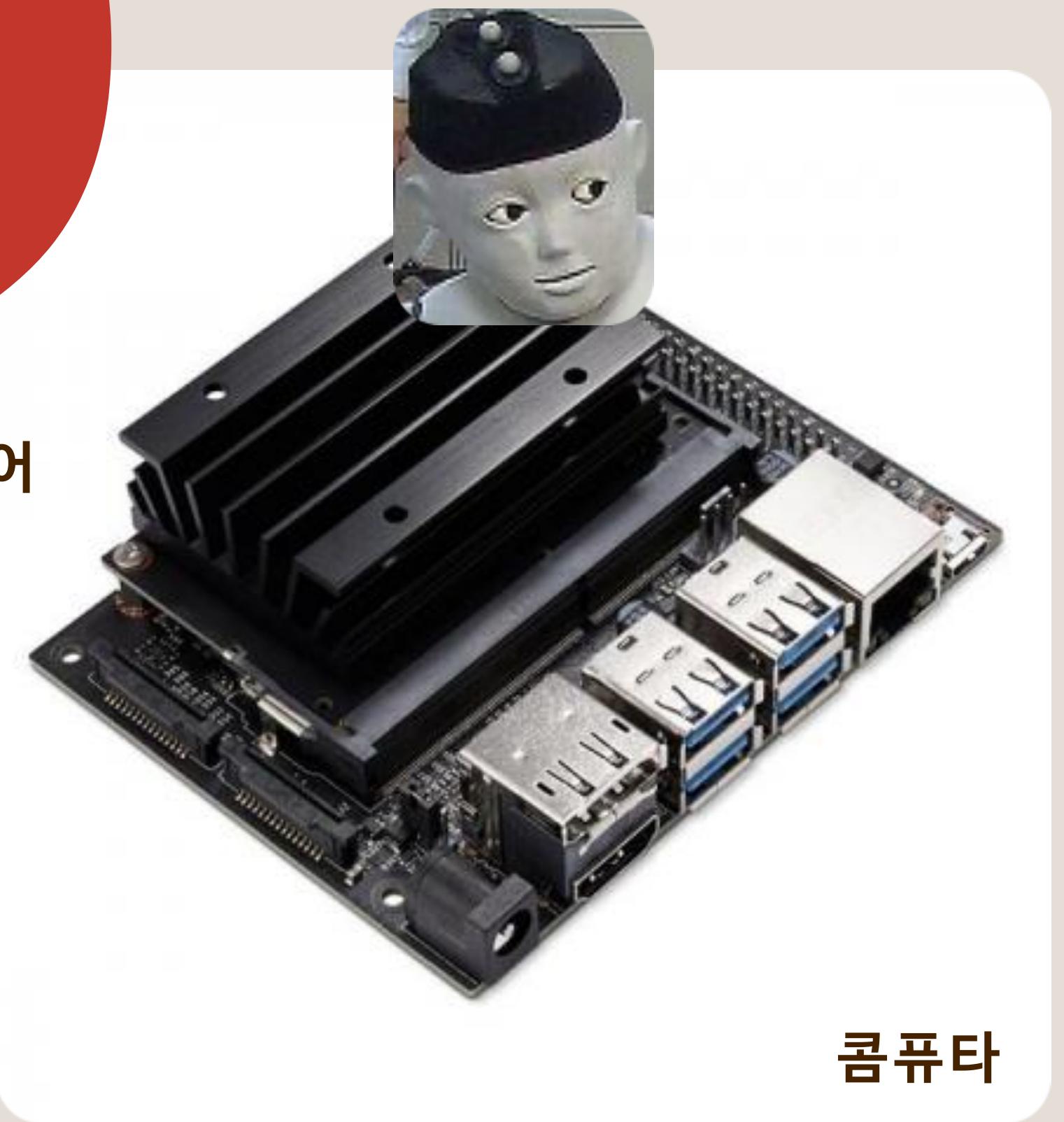
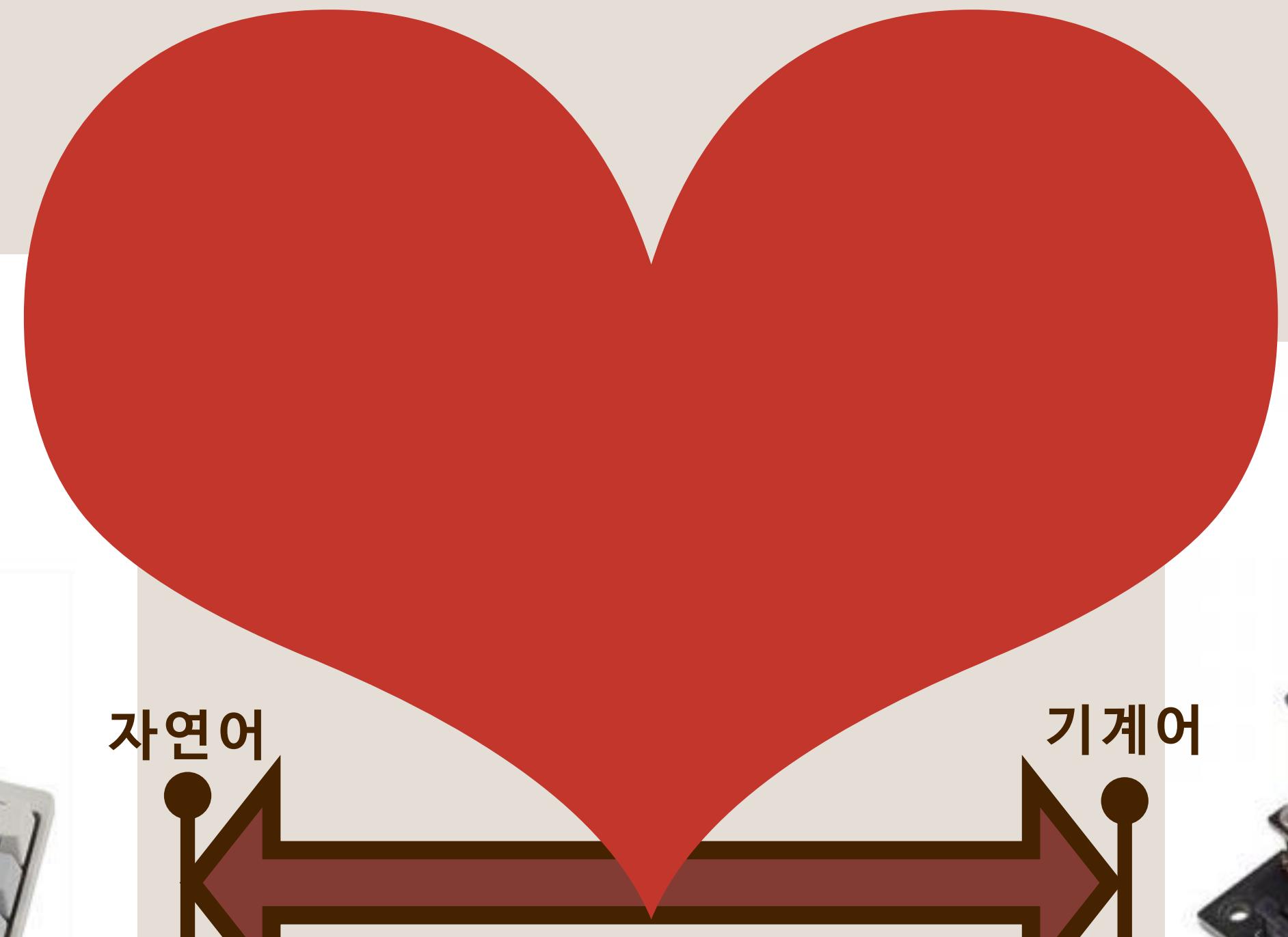
- 컴퓨터의 언어
- 0/1, on/off -> 2진수로 표현
- 사람은 이해하기 어려운 언어

프로그래밍



## C++ compiler 개요

Programming?

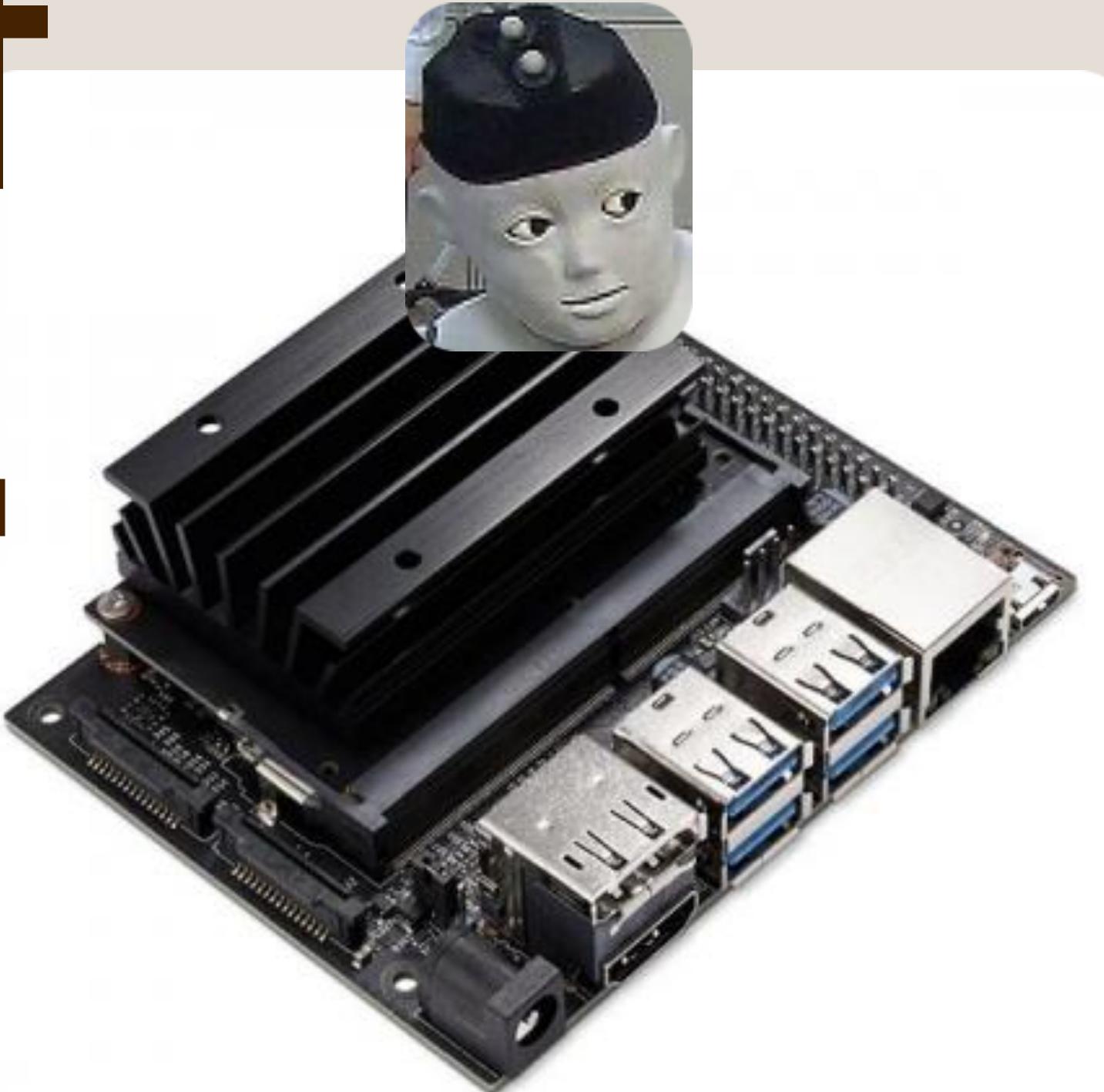
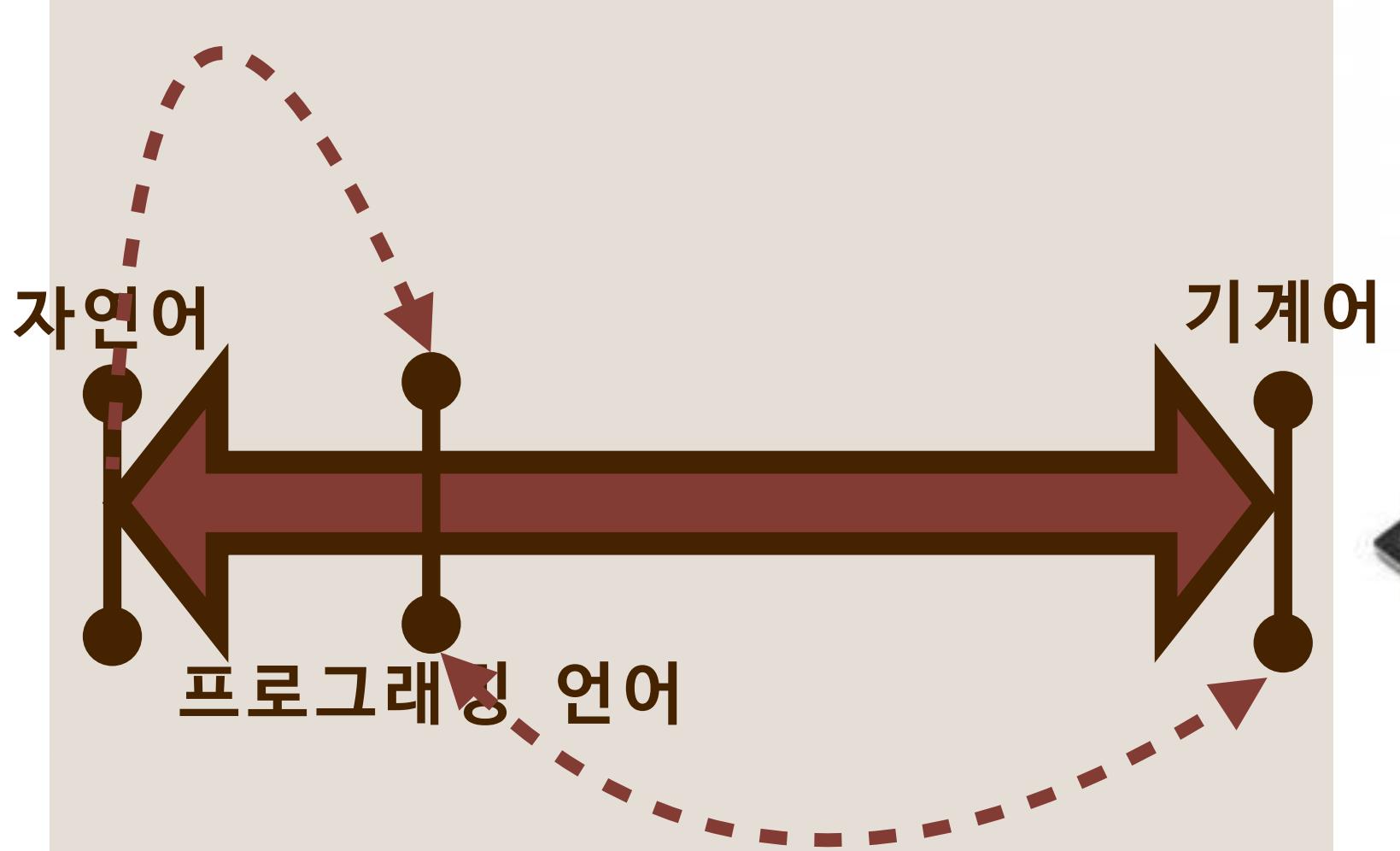


닝겐

## C++ compiler 개요

Programming?

# 닝겐이 간다

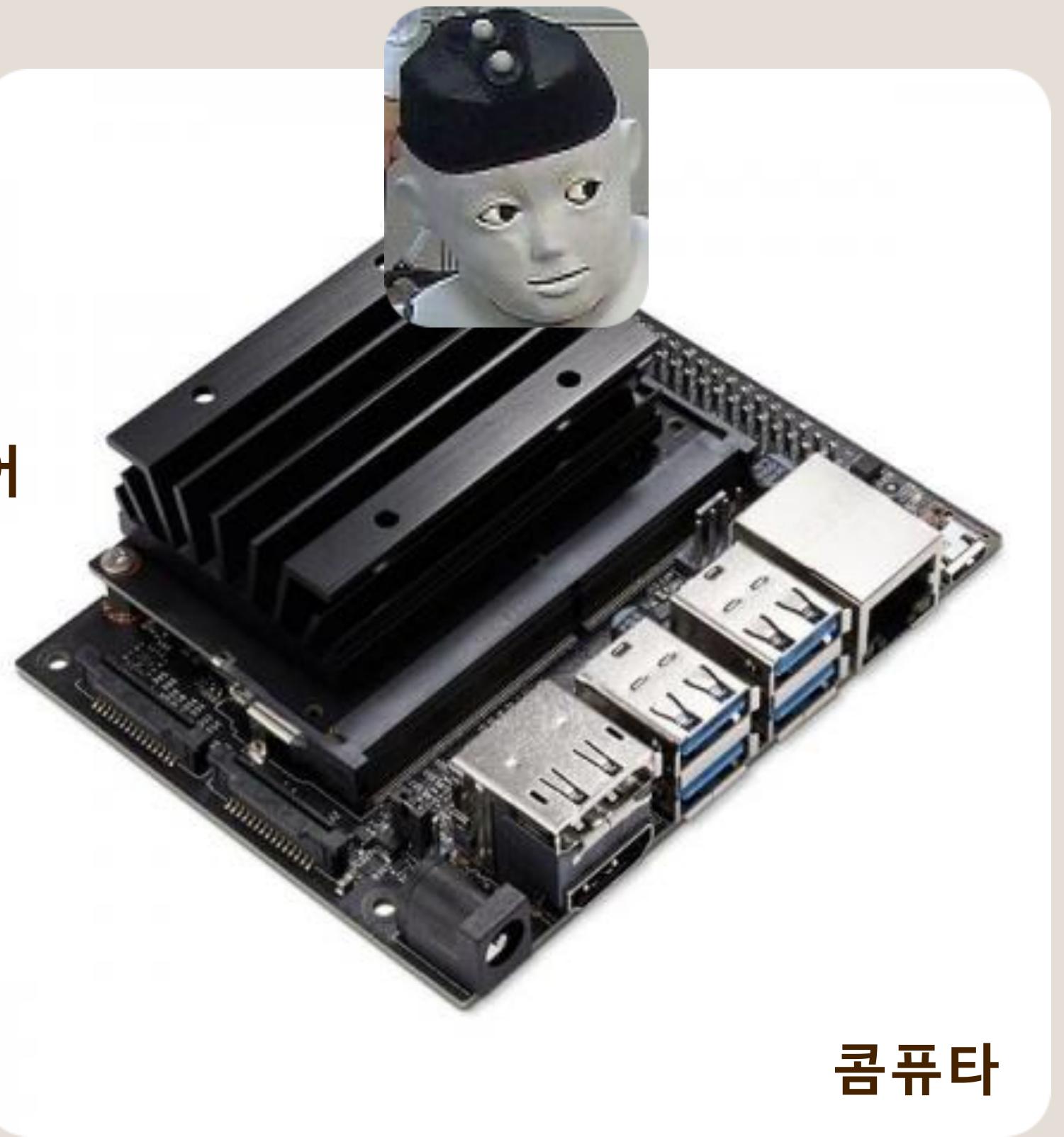
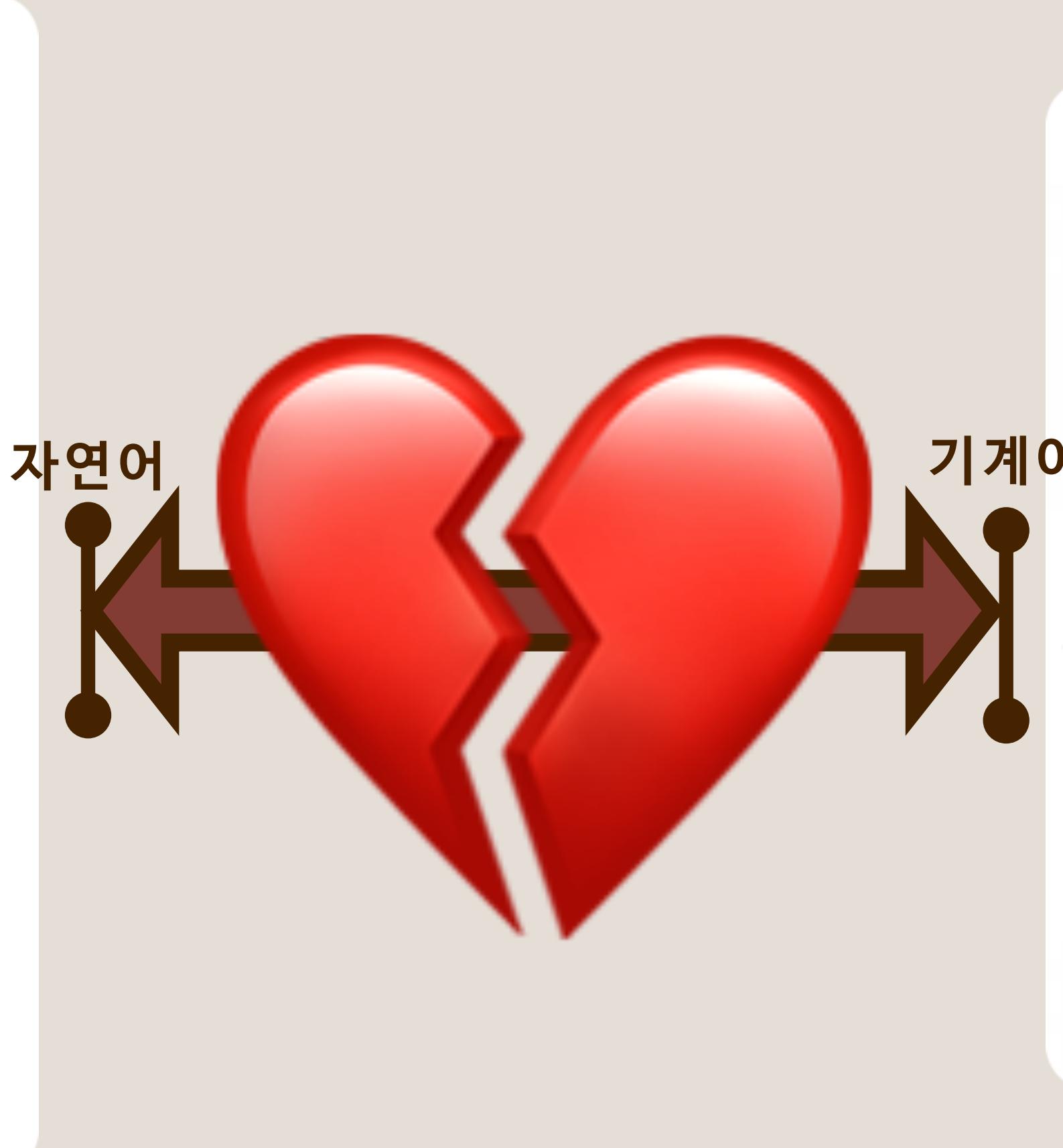


컴퓨터

닝겐

## C++ compiler 개요

Programming?



닝겐

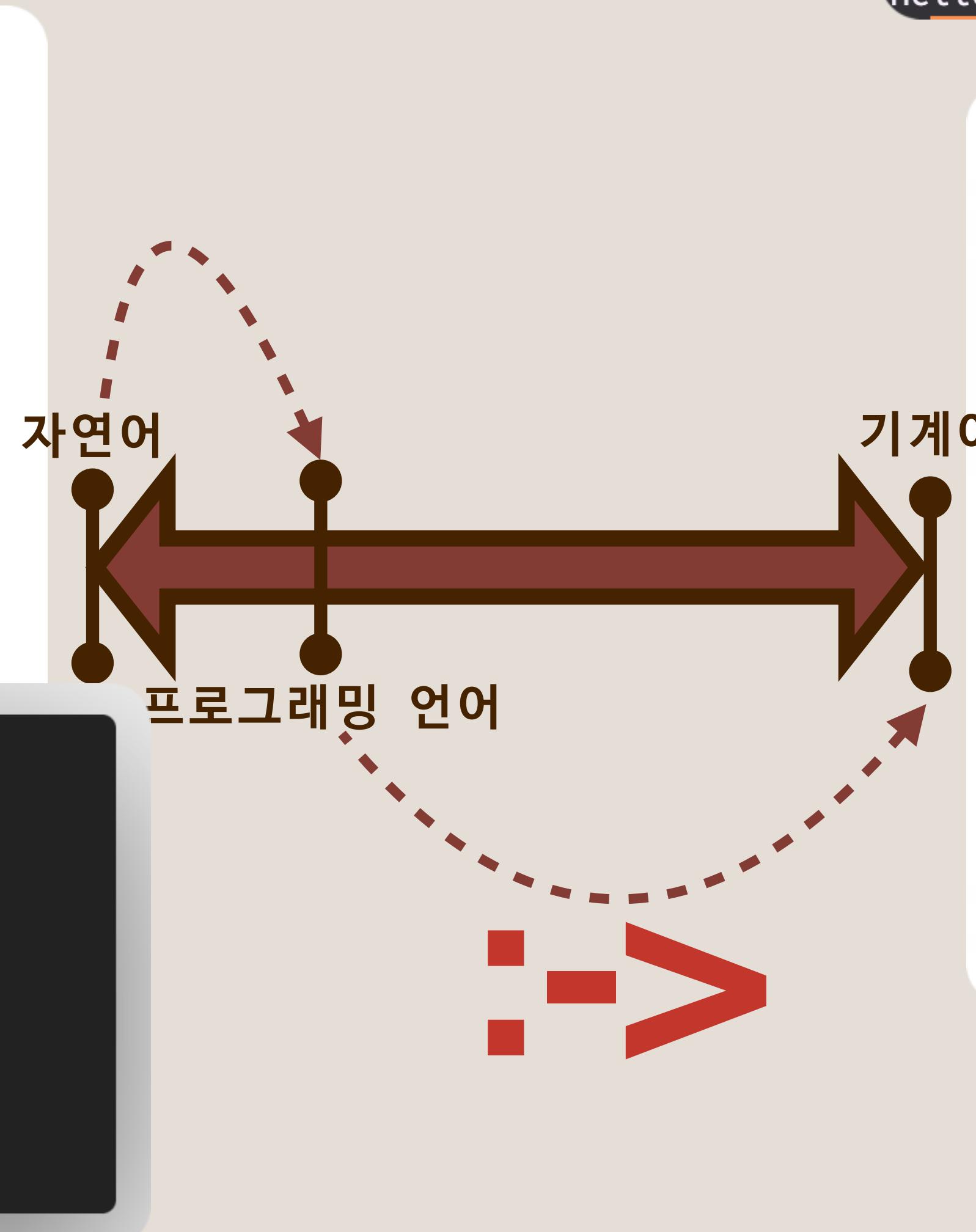
## C++ compiler 개요

Programming?

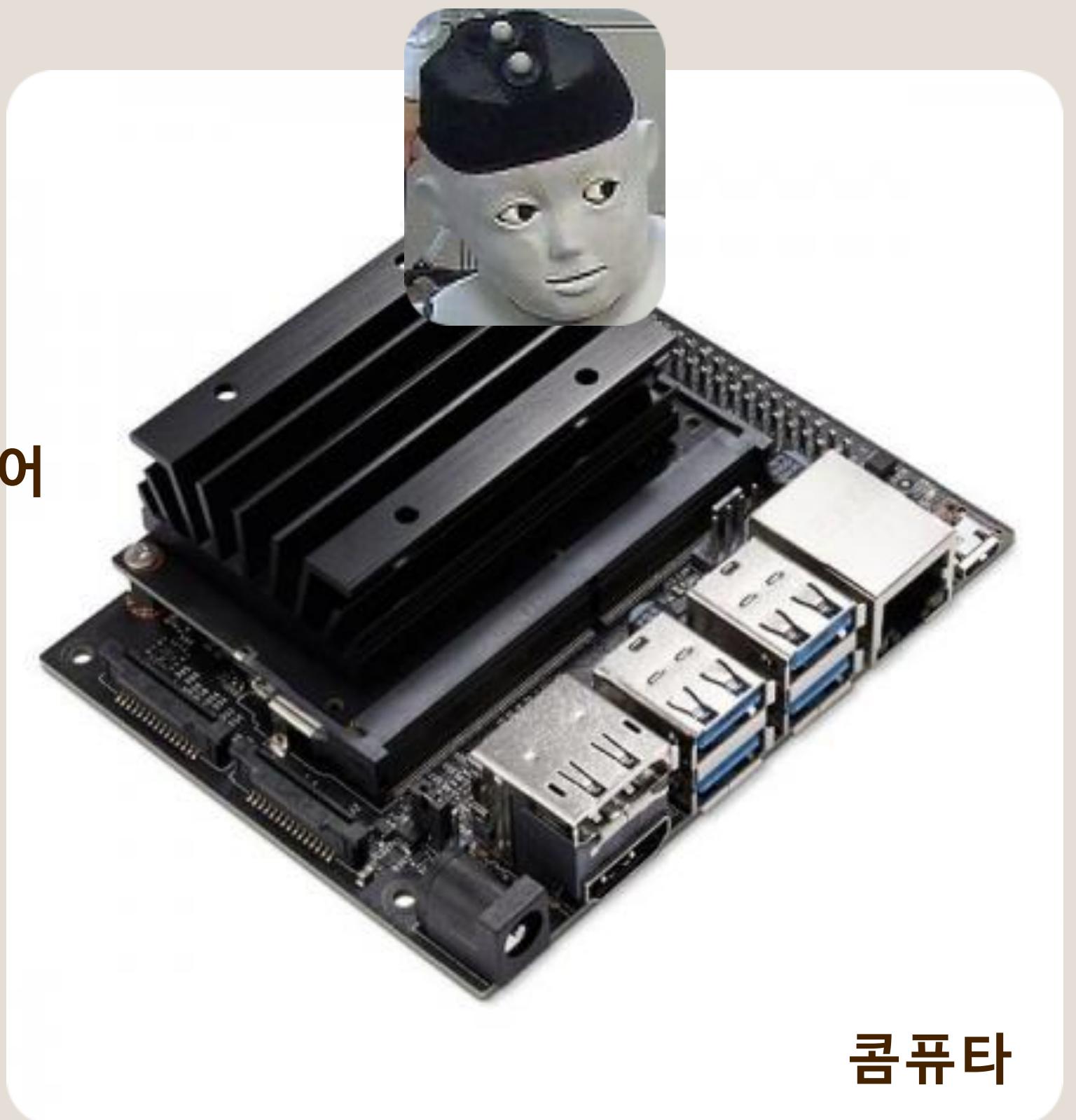


```
#include <cstdint>
#include <iostream>

auto main() -> int32_t {
    std::cout << "Hello World!" << std::endl;
    return 0;
}
```



```
~/repos/holyground/readable_code_cpp_development_e
~/Users/undo/repos/holyground/readable_code_cpp_deve
Hello World!
```



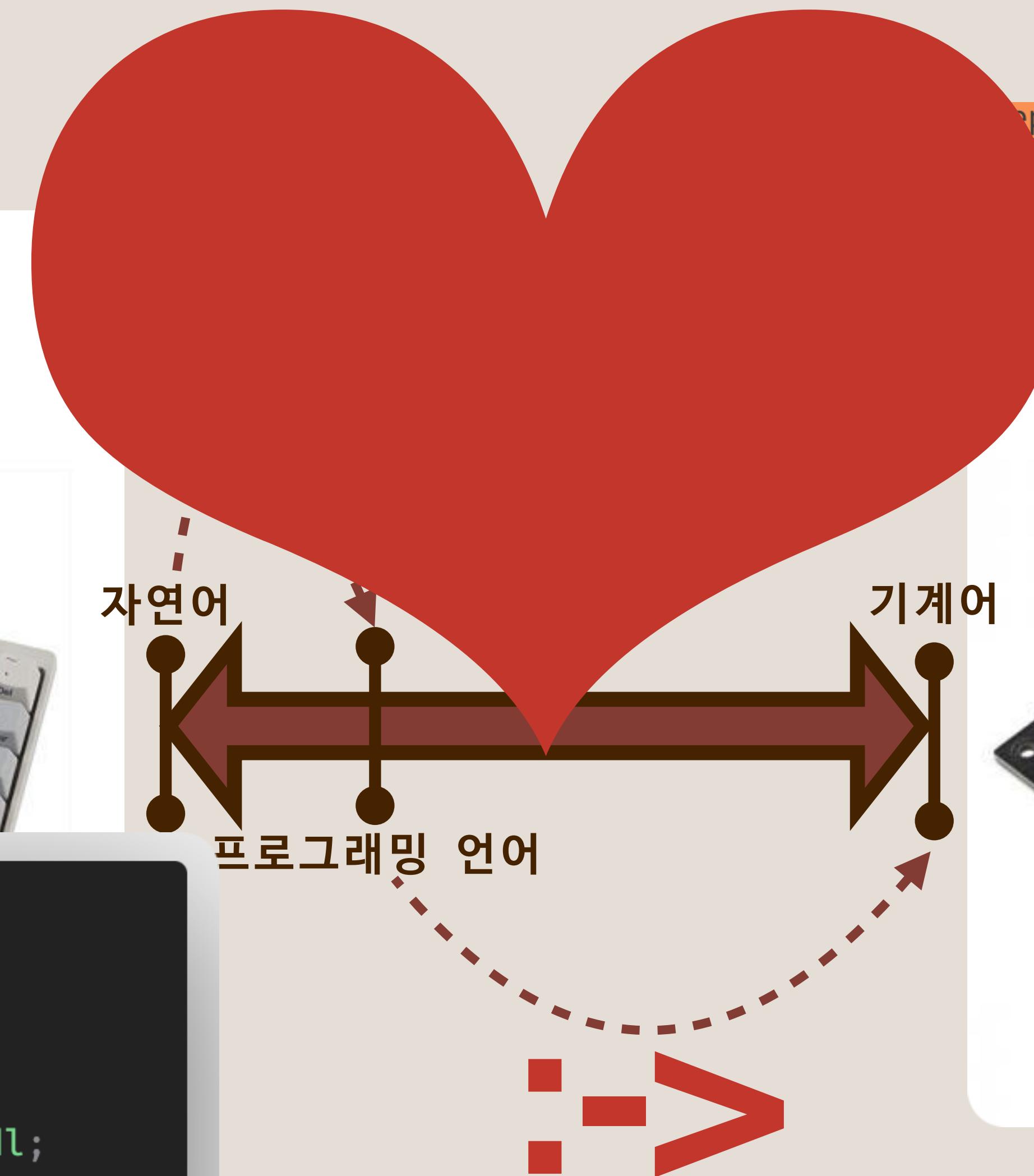
## C++ compiler 개요

Programming?

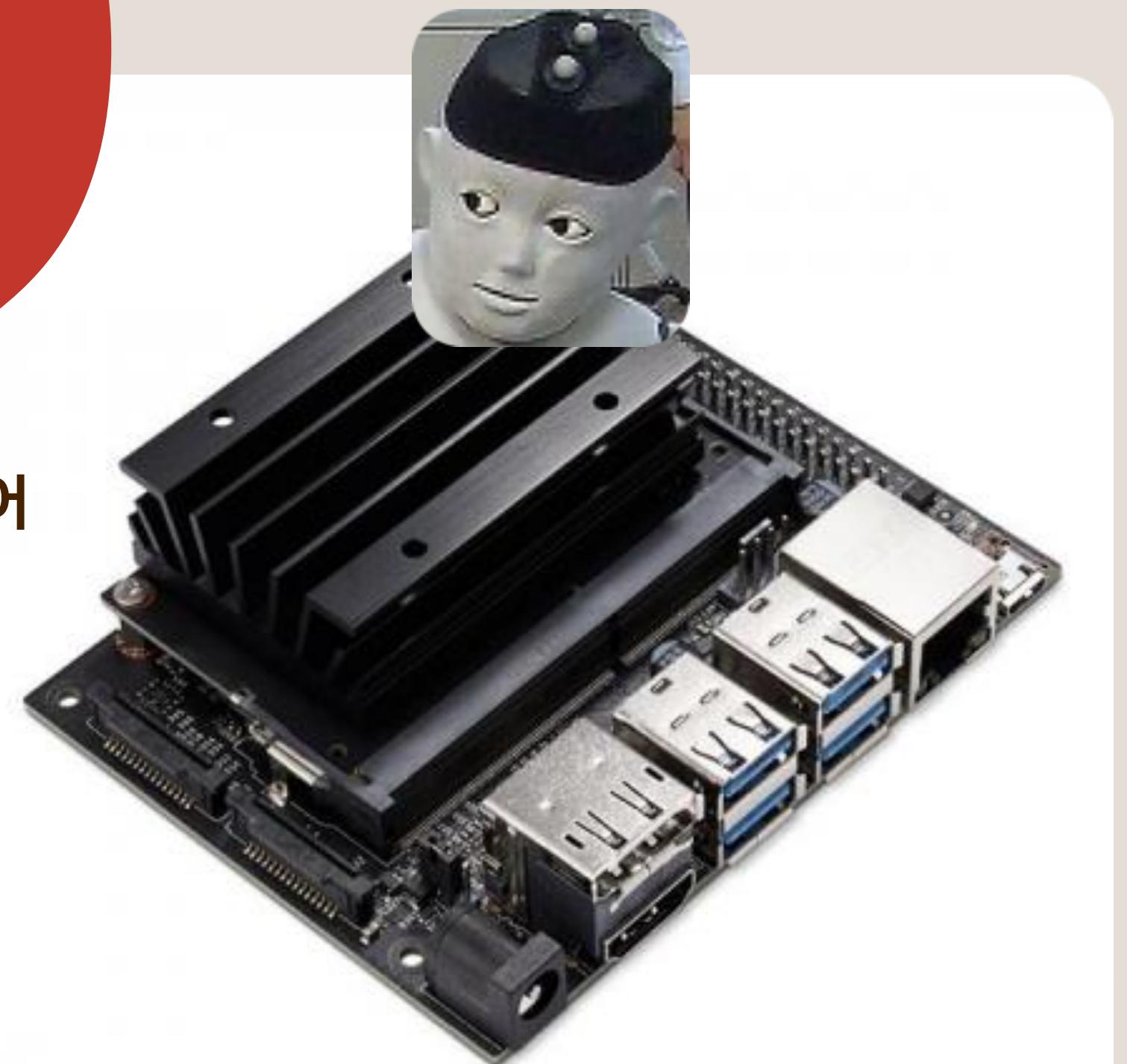


```
#include <cstdint>
#include <iostream>

auto main() -> int32_t {
    std::cout << "Hello World!" << std::endl;
    return 0;
}
```



repos/holyground/readable\_code\_cpp\_development\_environment/undo/repos/holyground/readable\_code\_cpp\_development\_environment/HelloWorld!



컴퓨터

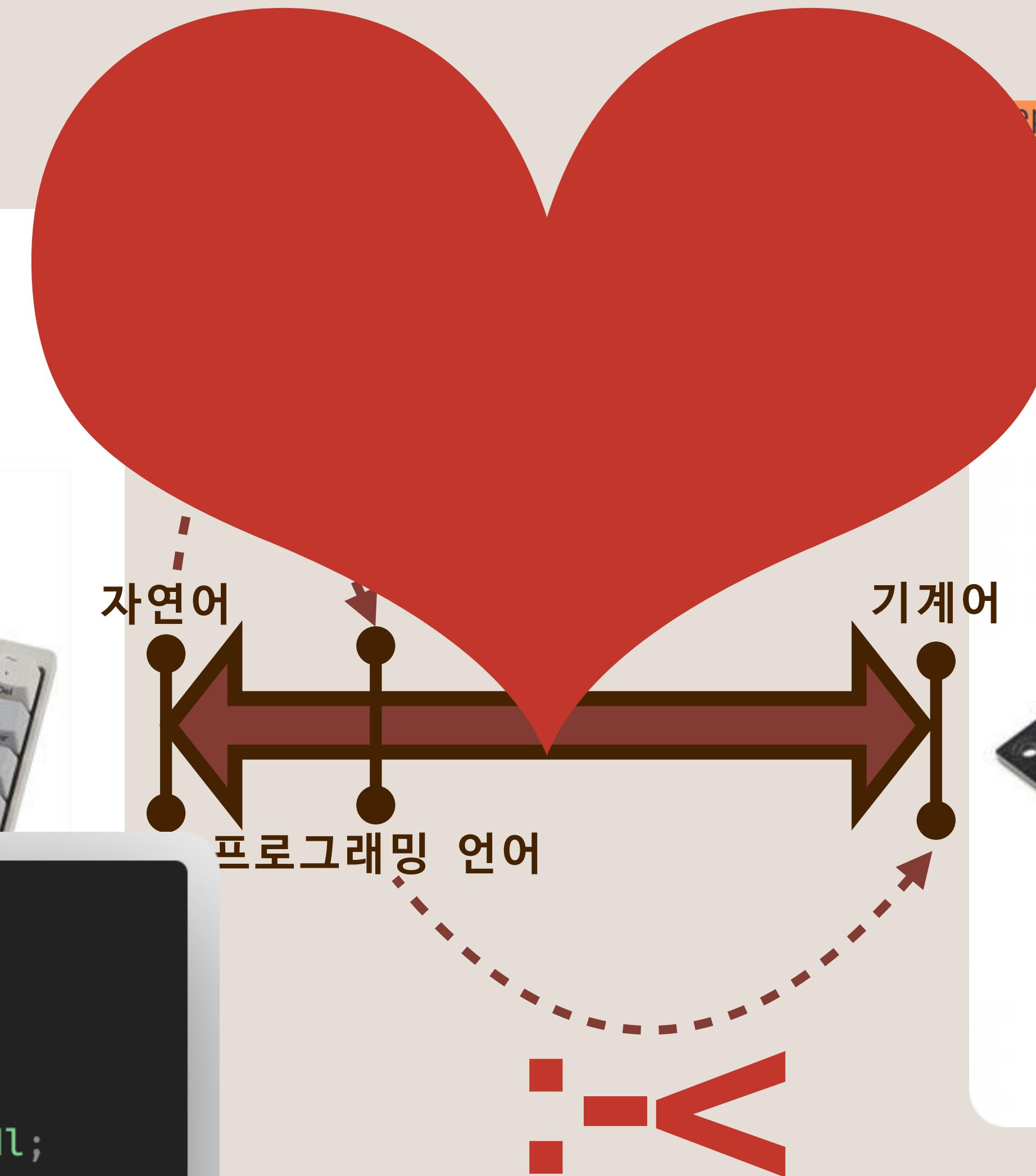
# C++ compiler 개요

# Programming?



```
#include <cstdint>
#include <iostream>

auto main() → int32_t {
    std::cout << "Hello World!" << std::endl;
    return 0;
}
```



# 컴퓨터

## C++ compiler 개요

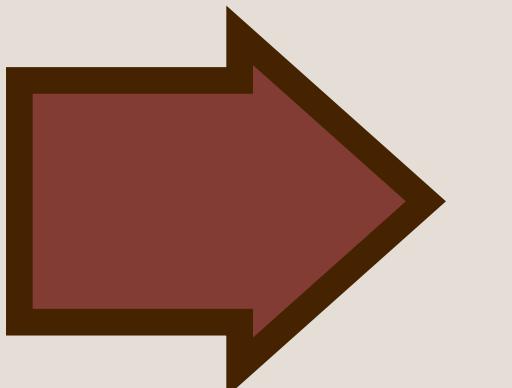
What is compiler

### 사람의 언어

```
#include <cstdint>
#include <iostream>

auto main() → int32_t {
    std::cout << "Hello World!" << std::endl;
    return 0;
}
```

바꾼다

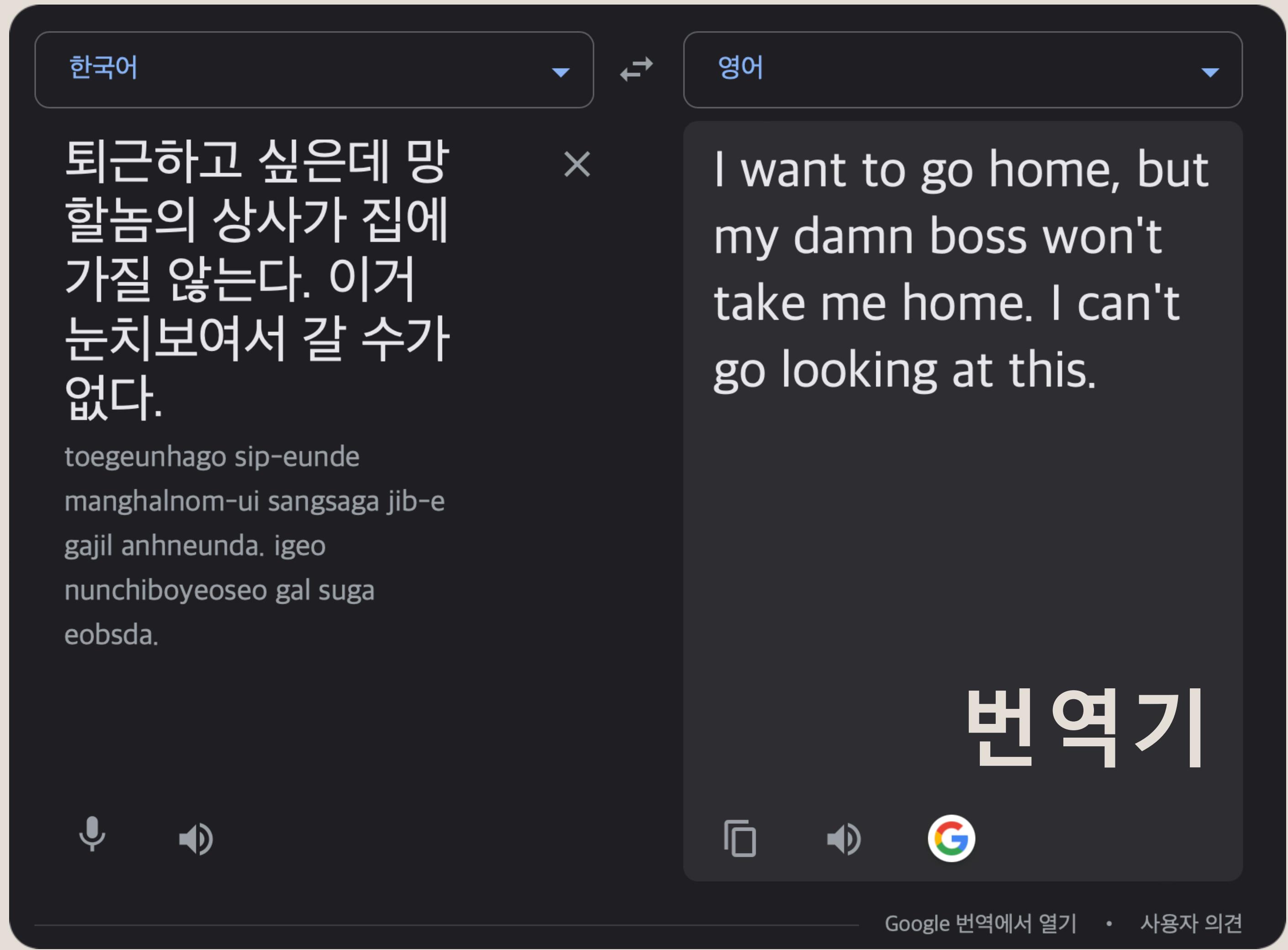


### 기계의 언어

```
~/repos/holyground/readable_code_cpp_development_e
/Users/undo/repos/holyground/readable_code_cpp_deve
Hello World!
```

## C++ compiler 개요

What is compiler



The image shows a screenshot of the Google Translate mobile application. At the top, there are two dropdown menus for language selection: '한국어' (Korean) on the left and '영어' (English) on the right. A double-headed arrow icon is positioned between them. Below these, the Korean text is displayed in a large, bold font:

퇴근하고 싶은데 망  
할놈의 상사가 집에  
가질 않는다. 이거  
눈치보여서 갈 수가  
없다.

Below the Korean text, its phonetic transcription is shown in smaller text:

toegeunhago sip-eunde  
manghalnom-ui sangsaga jib-e  
gajil anhneunda. igeo  
nunchiboyeoseo gal suga  
eobsda.

To the right of the Korean text, the English translation is provided:

I want to go home, but  
my damn boss won't  
take me home. I can't  
go looking at this.

In the bottom right corner of the main translation area, the word "번역기" (Translator) is written in large, white, sans-serif letters.

At the very bottom of the screen, there are several small icons: a microphone icon, a speaker icon, a square icon, another speaker icon, and the Google logo. Below these icons, the text "Google 번역에서 열기" (Open in Google Translate) and "• 사용자 의견" (User feedback) are visible.

## C++ compiler 개요

### Types of C++ compilers

- GCC(GNU Compiler Collection)◦
- Clang(a C language family frontend for LLVM)◦
- MSVC(Microsoft Visual C++)◦
- Intel C++ Compiler(ICC)

## C++ compiler 개요

### Types of C++ compilers

- GCC(GNU Compiler Collection)◦
- Clang(a C language family frontend for LLVM)◦
- MSVC(Microsoft Visual C++) → for Windows◦
- Intel C++ Compiler(ICC) → for Intel

## C++ compiler 개요

### Types of C++ compilers

- GCC(GNU Compiler Collection)
- Clang(a C language family frontend for LLVM)
- MSVC(Microsoft Visual C++) → for Windows
- Intel C++ Compiler(ICC) → for Intel

## C++ compiler 개요

### What is GCC

- "GNU Compiler Collection"▶
- 여러 프로그래밍 언어에 대한 컴파일러 모음▶
- 원래는 "GNU C Compiler"로 C 언어만을 위한 컴파일러로 시작됨▶
- 시간이 지나면서 다양한 프로그래밍 언어를 지원하게 되면서 이름이 "Compiler Collection"으로 변경

## C++ compiler 개요

### Features of GCC

- 다양한 언어 지원
  - C, C++, Ada, Fortran, Go 등과 같은 여러 프로그래밍 언어를 컴파일 가능
- 다양한 플랫폼 지원
  - 다양한 운영 체제와 아키텍처에서 GCC를 사용 가능
- 오픈소스
  - GNU General Public License(GPL) 하에 배포되는 자유 소프트웨어
  - 소스 코드를 자유롭게 확인, 수정, 재배포 가능
- 최적화 기능
  - 다양한 최적화 옵션을 제공하여 코드의 성능과 크기를 개선 가능
- 플러그인 아키텍처
  - GCC 4.5 버전부터는 플러그인 아키텍처를 지원하기 시작
  - 사용자가 컴파일 과정에 추가적인 기능을 확장 가능
- 표준 준수
  - 여러 프로그래밍 언어 표준을 준수
    - C 언어의 경우 ANSI C, C99, C11 등의 표준
    - C++의 경우 C++98, C++11, C++14 등의 표준

## C++ compiler 개요

### What is Clang

- C, C++, Objective-C, 그리고 Objective-C++ 프로그래밍 언어를 위한 컴파일러 프론트엔드◦
- LLVM 프로젝트의 일부로서, LLVM 백엔드와 함께 사용되어 소스 코드를 기계 코드로 변환◦
- 성능, 메모리 사용 효율성, 그리고 진단 메시지의 명확성등의 이점을 가짐◦
- GCC의 GPL3 라이센스 문제로 인해 기업들이 LLVM(Clang)으로 이주(Apple, Google, Sony 등)

## C++ compiler 개요

### Features of Clang

- 성능 및 메모리 효율성
  - 빠른 컴파일 시간과 적은 메모리 소모로 설계됨
- 명확하고 유용한 진단 메시지
  - 오류나 경고가 발생할 때, 문제의 원인을 명확히 알려주는 메시지를 제공
  - 때로는 문제를 해결할 수 있는 제안도 함께 제시
- 모듈 지원
  - C++ 모듈을 지원하여 컴파일 시간을 줄이고 코드 재사용성을 향상
- 고도의 호환성
  - GCC와 높은 호환성을 가지며, 대부분의 GCC 플래그와 옵션을 지원
- 라이브러리 기반 아키텍처
  - 다양한 구성 요소(구문 분석, AST 생성)는 재사용 가능한 라이브러리로 제공
  - 이 라이브러리들을 사용하여 코드 분석, 리팩토링 및 다른 도구를 개발 가능
- 통합 정적 분석
  - 정적 코드 분석 도구가 내장되어 있음(Clang Tidy)
  - 버그와 코드 품질 문제를 자동으로 감지 가능
- 다양한 플랫폼 및 아키텍처 지원
  - 다양한 운영 체제 및 하드웨어 아키텍처에서 사용 가능

## C++ compiler 개요

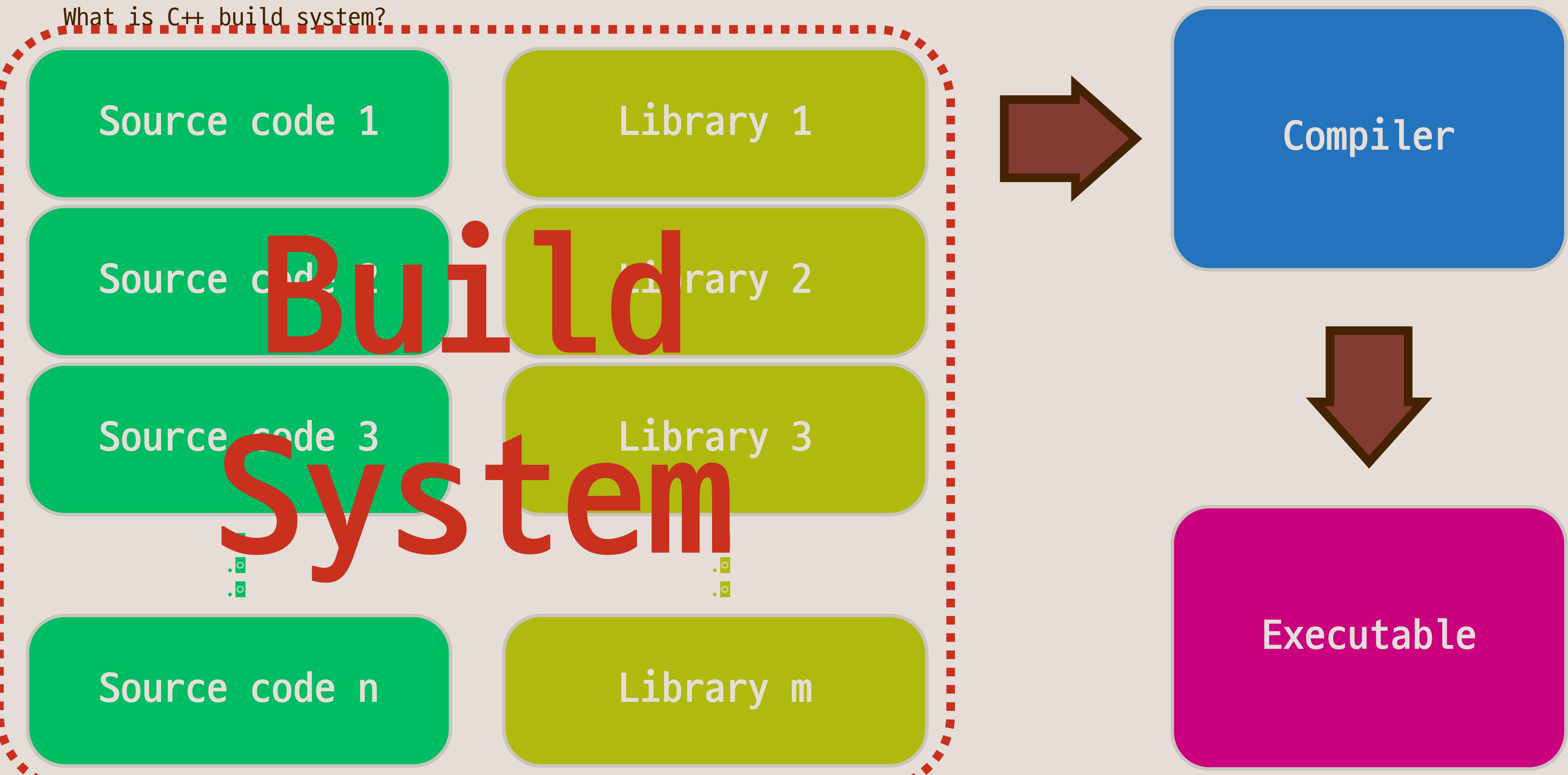
Why do we use Clang/GCC

- 두 컴파일러 모두 OS/Architecture에 상관없이 사용 가능, 크로스 컴파일 가능▪
- 동시에 두 가지 이상의 컴파일러를 사용해서 보다 완전 무결한 코드를 작성 가능▪
- Clang의 경우 GCC보다 최적화가 잘 되어 있고 라이센스 이슈로 부터 자유로움▪
- Clang-tidy, clang-format은 매우 강력한 정적 분석도구이자 formatter임▪
- 만약 둘 중 하나의 컴파일러만 사용하는 경우에는 GCC보다 Clang을 권장

# Build system 개요

## Build system 개요

What is C++ build system?



## Build system 개요

### Features of C++ build system

- 코드 컴파일 및 빌드◦
  - 소스 코드 파일을 컴파일러와 링커를 사용하여 실행 가능한 프로그램으로 변환◦
  - 소스 코드의 구문 검사, 의미 분석, 최적화, 목적 코드 생성 등을 포함◦
- 의존성 관리◦
  - 프로젝트 내의 파일 및 모듈 간의 의존성을 추적하고 관리◦
  - 특정 파일이 변경되었을 때 어떤 파일을 다시 컴파일해야 하는지 결정하는 데 도움◦
- 자동화◦
  - 빌드 프로세스를 자동화하여 반복적이고 복잡한 작업을 간소화하며 인간의 실수를 줄임◦
  - 개발자가 더 많은 시간을 코드 작성 및 디버깅에 집중할 수 있도록 도움◦
- 환경 관리◦
  - 다양한 플랫폼 또는 환경에서 소프트웨어를 빌드하고 실행할 수 있도록 도움◦
  - 다른 운영 체제 또는 아키텍처에서 동일한 코드를 재사용하기 위해 중요◦
- 테스트 및 배포◦
  - 프로젝트의 테스트 단계를 포함하여 소프트웨어의 품질을 관리하고 배포 가능한 최종 결과물을 생성

## Build system 개요

### List of C++ build systems

- Make(GNU Make)
  - 가장 오래되고 널리 사용되는 빌드 도구 중 하나
  - Makefile이라는 스크립트를 작성하여 빌드 과정을 정의
  - 다양한 플랫폼에서 사용 가능하며, 많은 프로젝트에서 기본 빌드 도구로 선택

- CMake
  - 플랫폼 및 컴파일러 독립적인 빌드 시스템
  - CMakeLists.txt 파일을 통해 빌드 규칙을 정의
  - CMake는 다양한 빌드 시스템의 프로젝트 파일(예: Makefile, Visual Studio 프로젝트 파일)을 생성 가능

- Bazel
  - 대규모 프로젝트에 적합한 Google에서 개발한 빌드 시스템
  - 플랫폼 독립적이며, 빌드 및 테스트를 위한 고성능 도구를 제공

- Meson
  - 현대적인 빌드 시스템으로, 빠른 빌드 속도와 사용자 친화적인 구성을 목표
  - Ninja 빌드 시스템과 함께 사용

- SCons
  - Python 스크립트를 사용하여 빌드 규칙을 정의
  - 플랫폼 독립적이며, 다양한 컴파일러와의 호환성을 제공

## Build system 개요

### List of C++ build systems

- Make(GNU Make)
  - 가장 오래되고 널리 사용되는 빌드 도구 중 하나◦
  - Makefile이라는 스크립트를 작성하여 빌드 과정을 정의◦
  - 다양한 플랫폼에서 사용 가능하며, 많은 프로젝트에서 기본 빌드 도구로 선택◦

- CMake
  - 플랫폼 및 컴파일러 독립적인 빌드 시스템◦
  - CMakeLists.txt 파일을 통해 빌드 규칙을 정의◦
  - CMake는 다양한 빌드 시스템의 프로젝트 파일(예: Makefile, Visual Studio 프로젝트 파일)을 생성 가능◦

- Bazel
  - 대규모 프로젝트에 적합한 Google에서 개발한 빌드 시스템◦
  - 플랫폼 독립적이며, 빌드 및 테스트를 위한 고성능 도구를 제공◦

- Meson
  - 현대적인 빌드 시스템으로, 빠른 빌드 속도와 사용자 친화적인 구성을 목표◦
  - Ninja 빌드 시스템과 함께 사용◦

- SCons
  - Python 스크립트를 사용하여 빌드 규칙을 정의◦
  - 플랫폼 독립적이며, 다양한 컴파일러와의 호환성을 제공◦

## Build system 개요

Why do we use CMake?

- 플랫폼 독립성
  - 여러 운영 체제에서 동일한 빌드 프로세스를 사용 가능
  - 크로스 플랫폼 컴파일 가능
- 유연한 빌드 옵션
  - 다양한 빌드 옵션을 지원하며, 개발자가 필요한 컴파일러, 라이브러리, 플래그 등을 설정 가능
  - 프로젝트의 요구 사항에 맞게 빌드 환경을 쉽게 구성 가능
- 다양한 IDE(Integrated Development Environment) 지원
  - 다양한 통합 개발 환경(Visual Studio, Xcode, CLion, Code::Blocks, ...)을 지원
  - 개발자는 자신이 선호하는 IDE에서 C++ 프로젝트를 개발 가능 등 다양한 IDE와 통합이 가능

## Build system 개요

Why do we use CMake?

- 자동 의존성 관리
  - 프로젝트의 의존성 관리를 자동화함
  - 다른 라이브러리나 모듈을 프로젝트에 추가하면 CMake는 해당 의존성을 찾고 빌드 프로세스에 통합함
- 모듈화와 재사용성
  - 프로젝트를 모듈화하고 라이브러리 형태로 작성하여 코드 재사용성을 증진 가능
  - 다른 프로젝트에서 동일한 라이브러리를 사용할 수 있으며, CMake는 해당 라이브러리를 찾아 자동으로 연결
- 간결한 CMakeLists.txt 파일
  - 프로젝트를 구성하는 데 사용되는 CMakeLists.txt 파일은 비교적 간결하며 가독성이 높음
  - 프로젝트의 구조를 쉽게 파악하고 유지보수 가능
- 활발한 커뮤니티와 지원
  - 유저수가 가장 많은 build system
  - 활발한 오픈 소스 커뮤니티에 의해 개발되고 지원
  - 문제가 발생할 때 지원을 받을 수 있으며, 새로운 기능과 업데이트가 지속적으로 제공

# 필수 프로그램 설치 on MacOs

## 필수 프로그램 설치 on MacOS

### Terminal 실행

- Lunchpad → 기타 -> 터미널



## 필수 프로그램 설치 on MacOS

### Homebrew Installation

- 라이브러리 설치를 위한 Homebrew 설치◦
- 하기 커맨드를 순차적으로 입력◦
  - /bin/bash -c “\$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)”◦
  - (echo; echo 'eval "\$(/opt/homebrew/bin/brew shellenv)"') >> /Users/\$(id -un)/.zprofile◦
  - eval "\$((/opt/homebrew/bin/brew shellenv)"

## 필수 프로그램 설치 on MacOS

### GCC Installation

- 하기 커맨드를 순차적으로 입력▶
  - brew install gcc

## 필수 프로그램 설치 on MacOS

### Checking GCC Installation

- 하기 커맨드를 순차적으로 입력▶

- gcc --version▶

- 아래와 같은 결과가 나오면 설치 성공

```
~/repos/holyground/readable_code_cpp_development_environment/scripts/macos feat/adding_codes_for_lecture ?1
└── gcc --version
Apple clang version 14.0.3 (clang-1403.0.22.14.1)
Target: arm64-apple-darwin22.6.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
```

## 필수 프로그램 설치 on MacOS

### Clang& Clangd Installation

- 하기 커맨드를 순차적으로 입력▶
  - brew install llvm

## 필수 프로그램 설치 on MacOS

### Checking Clang& Clangd Installation

- 하기 커맨드를 순차적으로 입력◦

- clang --version◦
  - clangd --version◦

- 아래와 같은 결과가 나오면 설치 성공

```
└ ~/repos/holyground/readable_code_cpp_development_environment/scripts/macos feat/adding_codes_for_lecture ?1
  └ clang --version
Apple clang version 14.0.3 (clang-1403.0.22.14.1)
Target: arm64-apple-darwin22.6.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
└ ~/repos/holyground/readable_code_cpp_development_environment/scripts/macos feat/adding_codes_for_lecture ?1
  └ clangd --version
Apple clangd version 14.0.3 (clang-1403.0.22.14.1)
Features: mac+xpc
Platform: x86_64-apple-darwin22.6.0; target=arm64-apple-darwin22.6.0
```

## 필수 프로그램 설치 on MacOS

### CMake Installation

- 하기 커맨드를 순차적으로 입력▶
  - brew install cmake

## 필수 프로그램 설치 on MacOS

### Checking CMake Installation

- 하기 커맨드를 순차적으로 입력▶

- cmake --version▶

- 아래와 같은 결과가 나오면 설치 성공

```
~/repos/holyground/readable_code_cpp_development_environment/scripts/macosfeat/adding_codes_for_lecture ?1
└── cmake --version
cmake version 3.27.5

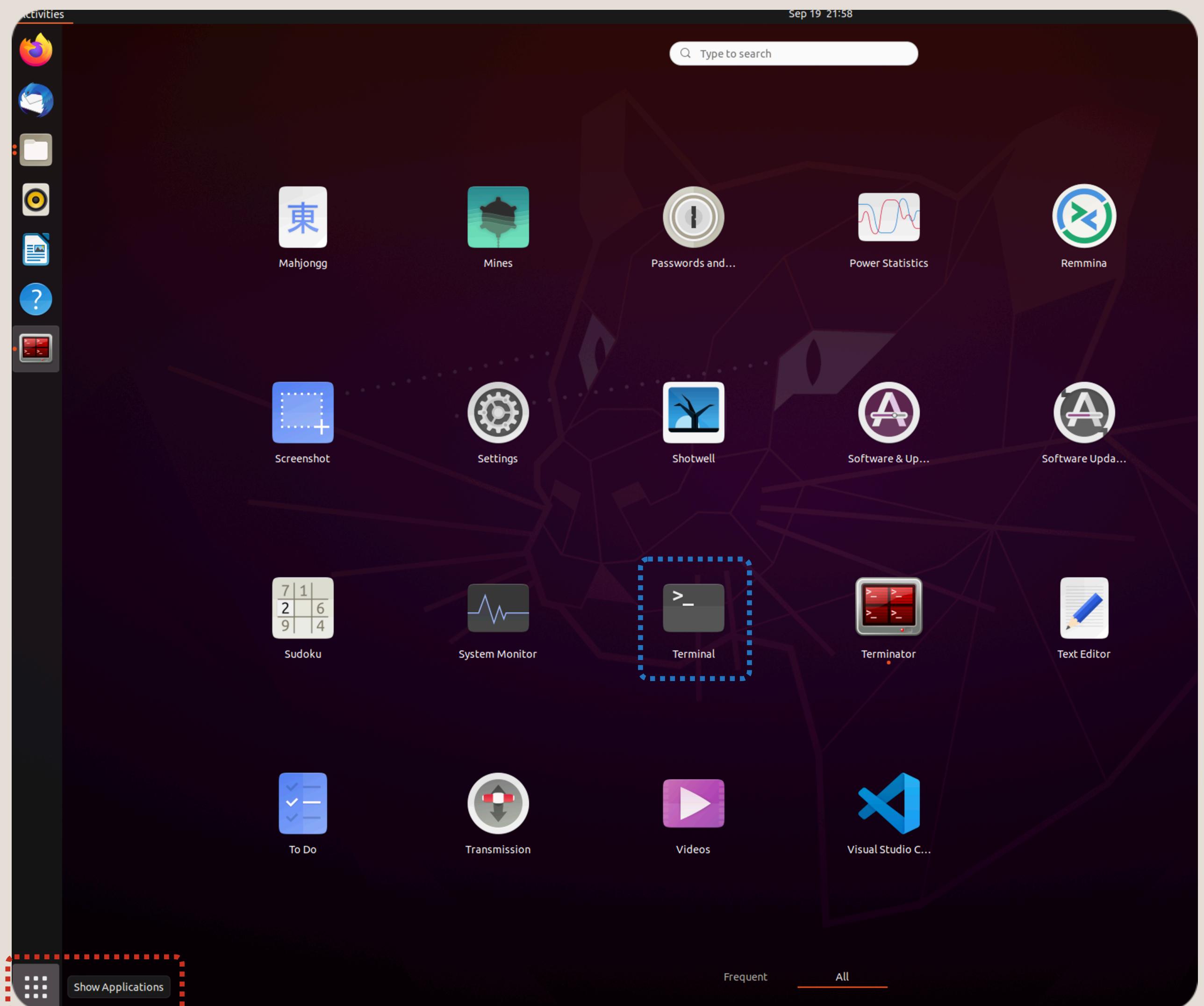
CMake suite maintained and supported by Kitware (kitware.com/cmake).
```

# 필수 프로그램 설치 on Ubuntu

## 필수 프로그램 설치 on Ubuntu

### Terminal 실행

- Show Application → Terminal



## 필수 프로그램 설치 on Ubuntu

### 필요한 종속성 Installation

- 하기 커맨드를 순차적으로 입력▶
  - sudo apt update -y▶
  - sudo apt upgrade -y▶
  - sudo apt install -y build-essential

## 필수 프로그램 설치 on Ubuntu

### GCC Installation

- 하기 커맨드를 순차적으로 입력▶
  - sudo apt install -y gcc

## 필수 프로그램 설치 on Ubuntu

### Checking GCC Installation

- 하기 커맨드를 순차적으로 입력▶
  - gcc --version▶
- 아래와 같은 결과가 나오면 설치 성공

```
parallels@ubuntu-linux-20-04-desktop:~/Desktop$ gcc --version
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

## 필수 프로그램 설치 on Ubuntu

### Clang& Clangd Installation

- 하기 커맨드를 순차적으로 입력◦
  - wget <https://apt.llvm.org/llvm.sh>◦
  - chmod +x llvm.sh◦
  - sudo ./llvm.sh 14◦
  - sudo apt install -y clang clangd

## 필수 프로그램 설치 on Ubuntu

### CMake Installation

- 하기 커맨드를 순차적으로 입력▣
  - sudo apt install -y cmake

## 필수 프로그램 설치 on Ubuntu

### Checking CMake Installation

- 하기 커맨드를 순차적으로 입력◦
  - cmake -version◦
- 아래와 같은 결과가 나오면 설치 성공

```
parallels@ubuntu-linux-20-04-desktop:~$ cmake --version
cmake version 3.16.3

CMake suite maintained and supported by Kitware (kitware.com/cmake).
```

# 예제 코드 빌드 on terminal

## 예제 코드 빌드 on terminal

### 예제 프로젝트 작성

- 하기와 같은 폴더 구조로 예제 프로젝트 작성(폴더, 파일)◦

```
└── cmake_example_project◦  
    ├── CMakeLists.txt◦  
    └── build◦  
        └── main.cpp
```

## 예제 코드 빌드 on terminal

예제 프로젝트 작성 - CMakeLists.txt

```
cmake_minimum_required(VERSION 3.11)

set(PACKAGE_NAME CMAKE_TEST_PROJECT)
set(PACKAGE_VERSION 1.1.0)

project(${PACKAGE_NAME} VERSION ${PACKAGE_VERSION} LANGUAGES CXX)

set(CMAKE_CXX_STANDARD 20)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
set(CMAKE_CXX_EXTENSIONS OFF)

set(CPACK_PROJECT_NAME ${PROJECT_NAME})
set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})
include(CPack)

message(STATUS "PACKAGE_NAME: ${PACKAGE_NAME}")
message(STATUS "PACKAGE_VERSION: ${PACKAGE_VERSION}")

add_executable(${PACKAGE_NAME}_MAIN main.cpp)
target_compile_options(${PACKAGE_NAME}_MAIN PRIVATE
    -Wall -Wextra -Wpedantic -Werror
)
```

## 예제 코드 빌드 on terminal

예제 프로젝트 작성 - main.cpp

```
#include <cstdint>
#include <iostream>

auto main() → int32_t {
    std::cout << "Hello World!" << std::endl;
    return 0;
}
```

# 예제 코드 빌드 on terminal

# 예제 프로젝트 빌드 및 실행

```
~/repos/holyground
└── cmake_example_project
    ├── CMakeLists.txt
    └── src
        └── main.cpp
~/repos/holyground/cmake_example_project
└── build
    ├── CMakeCache.txt
    ├── CMakeFiles
    │   ├── CMakeFiles.cmake
    │   ├── CPackConfig.cmake
    │   ├── CPackSourceConfig.cmake
    │   └── Makefile
    └── cmake_install.cmake
~/repos/holyground/cmake_example_project/build
└── ./CMAKE_TEST_PROJECT_MAIN
    Hello World!
```

# Version Control System(VCS)

VCS 개요

Git& Github CLI 설치 on MacOs

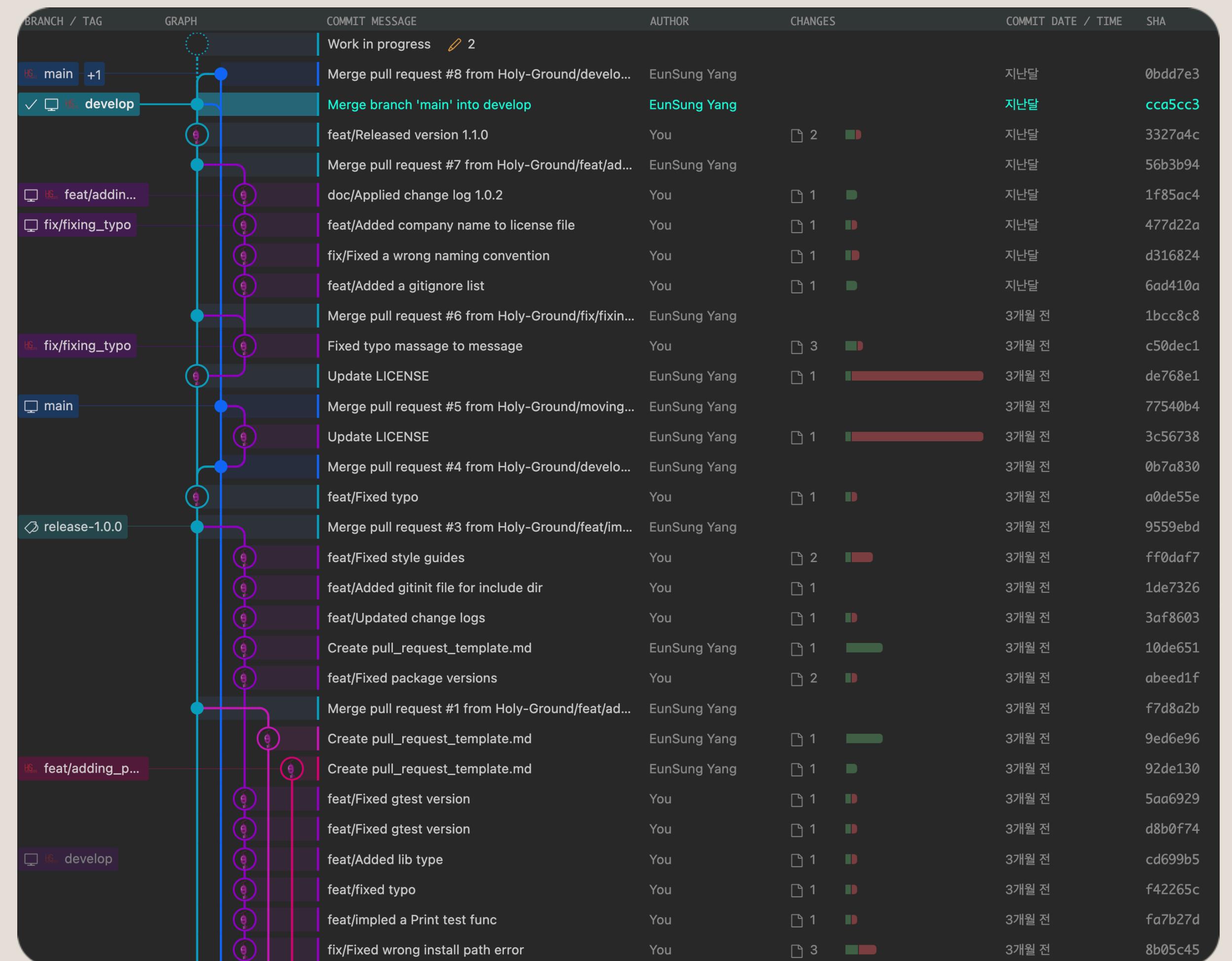
Git& Github CLI 설치 on Ubuntu

# VCS 개요

## VCS 개요

What is VCS?

- 프로젝트의 소프트웨어 버전을 관리하기 위해 사용 되는 도구
- 소스 코드의 저장소 역할을 함
- 작업 흐름 관리를 통해 협업에 대한 전반적인 프로세스를 관리



## VCS 개요

### Features of VCS

- 버전 관리◦
  - 프로젝트의 모든 버전 및 변경 내역을 저장하고 관리◦
  - 이러한 기록은 각 버전의 스냅샷으로 구성되며, 프로젝트의 이전 및 현재 상태를 쉽게 복구 가능◦
- 변경 추적◦
  - 소스 코드 파일의 변경 내용을 추적◦
  - 어떤 파일이 언제 수정되었는지, 누가 수정했는지 등을 알려주므로 변경 이력을 파악 가능◦
- 병합(Merge) 및 분기(Branch)◦
  - 여러 개발자가 동시에 작업하는 경우, 변경 사항을 통합하고 충돌을 처리하는 기능을 제공◦
  - 프로젝트의 다른 버전을 만들어 실험하거나 안정성을 확인하는 데 사용되는 분기를 지원◦
- 백업 및 회복◦
  - 프로젝트 파일을 안전하게 보관하고 필요한 경우 이전 버전으로 되돌릴 수 있는 기능을 제공◦
  - 코드 손상이나 실수로 인한 문제를 방지하고 회복하는 데 도움을 줌

## VCS 개요

### Features of VCS

- 협업
  - 여러 개발자가 동시에 작업하는 경우, 변경 내용을 효율적으로 공유하고 조율하는 데 도움을 줌
  - 다른 개발자의 작업을 병합하거나 수정 사항을 검토하는 등의 작업을 수행 가능

- 보안
  - 코드의 보안을 강화 가능
  - 접근 제어 및 권한 관리를 통해 민감한 정보를 안전하게 보호하고 무단 수정을 방지 가능

- 작업 흐름 관리
  - 작업 흐름을 관리하고 프로젝트의 다양한 버전을 관리 가능
  - 소프트웨어 개발 생명주기(Lifecycle)를 관리하는 데 도움을 줌

## VCS 개요

### List of VCS

- Git
- Subversion(SVN)
- Mercurial
- Perforce(Helix Core)

## VCS 개요

### List of VCS

- Git
- Subversion(SVN)
- Mercurial
- Perforce(Helix Core)

## VCS 개요

### Why do we use Git

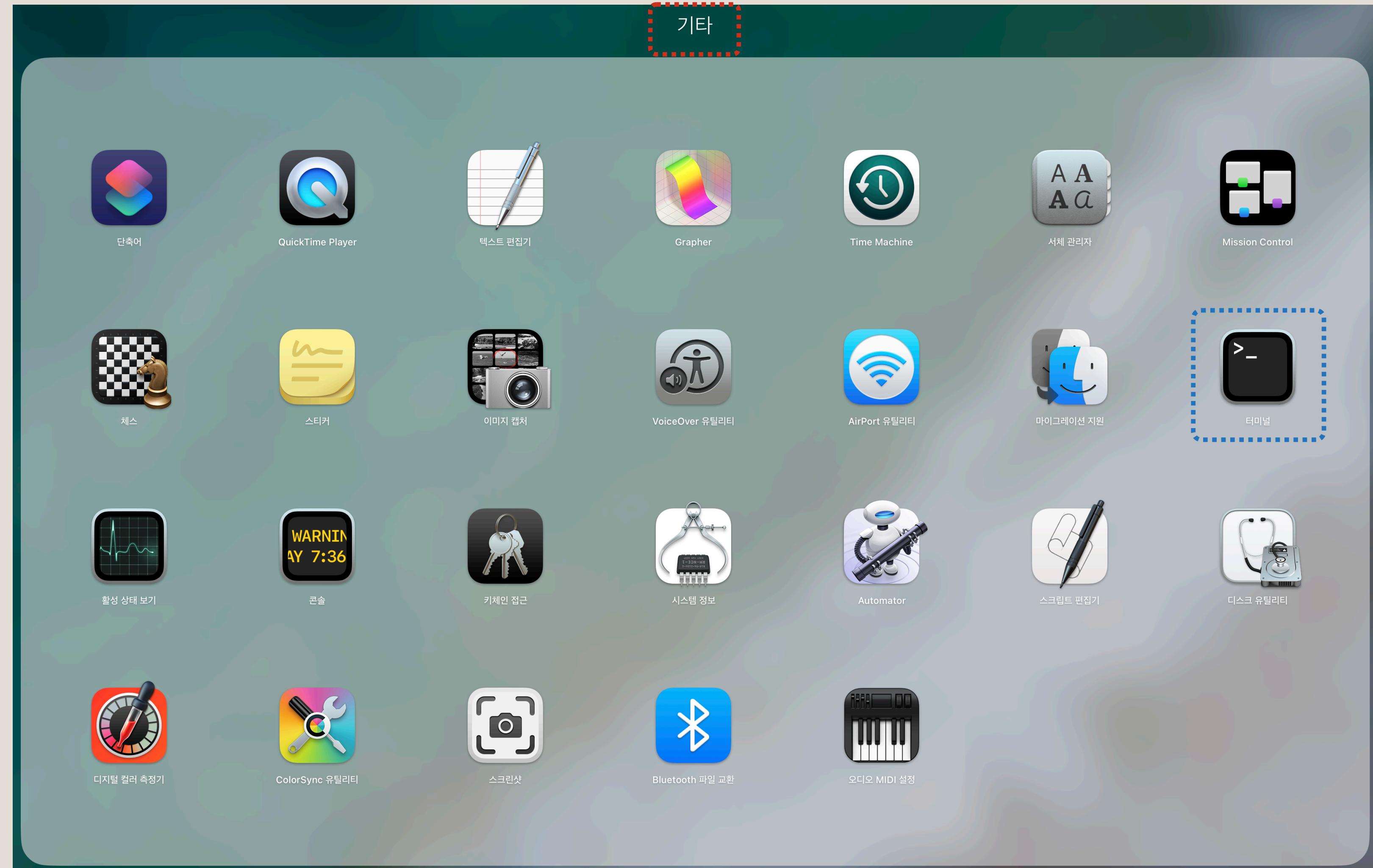
- 크로스 플랫폼 사용 가능■
- 무료 오픈소스, 유료 플랜으로 상용 서비스도 사용 가능■
- 가장 많은 유저가 사용하고 있어서 커뮤니티를 통한 지원이 용의■
- 가장 많은 오픈소스 보유(Github, Gitlab, Bitbucket, etc...)

# Git& Github CLI 설치 on MacOS

## Git & Github CLI 설치 on MacOS

### Terminal 실행

- Launchpad → 기타 -> 터미널



## Git & Github CLI 설치 on MacOS

### Git Installation

- 하기 커맨드를 순차적으로 입력▶
  - brew install git

## Git & Github CLI 설치 on MacOS

### Checking Git Installation

- 하기 커맨드를 순차적으로 입력▶
  - git --version▶
- 아래와 같은 결과가 나오면 설치 성공

```
~/repos/holyground/cmake_example_project/build
└─ git --version
git version 2.42.0
```

## Git& Github CLI 설치 on MacOS

### Github CLI Installation

- 하기 커맨드를 순차적으로 입력▶
  - brew install gh

## Git & Github CLI 설치 on MacOS

### Checking Github CLI Installation

- 하기 커맨드를 순차적으로 입력◦
  - gh --version◦
- 아래와 같은 결과가 나오면 설치 성공

```
~/repos/holyground/cmake_example_project/build
gh --version
gh version 2.35.0 (2023-09-19)
https://github.com/cli/cli/releases/tag/v2.35.0
```

## Git & Github CLI 설치 on MacOS

Login Github using CLI

- Github에 login 하기 위해 하기 커맨드를 입력▣

- gh auth login▣

- 입력 후 하기와 같이 프로세스 진행

```
~/repos/holyground/readable_code_cpp_development_environment/scripts/macos
└─ gh auth login
? What account do you want to log into? GitHub.com
? You're already logged into github.com. Do you want to re-authenticate? Yes
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

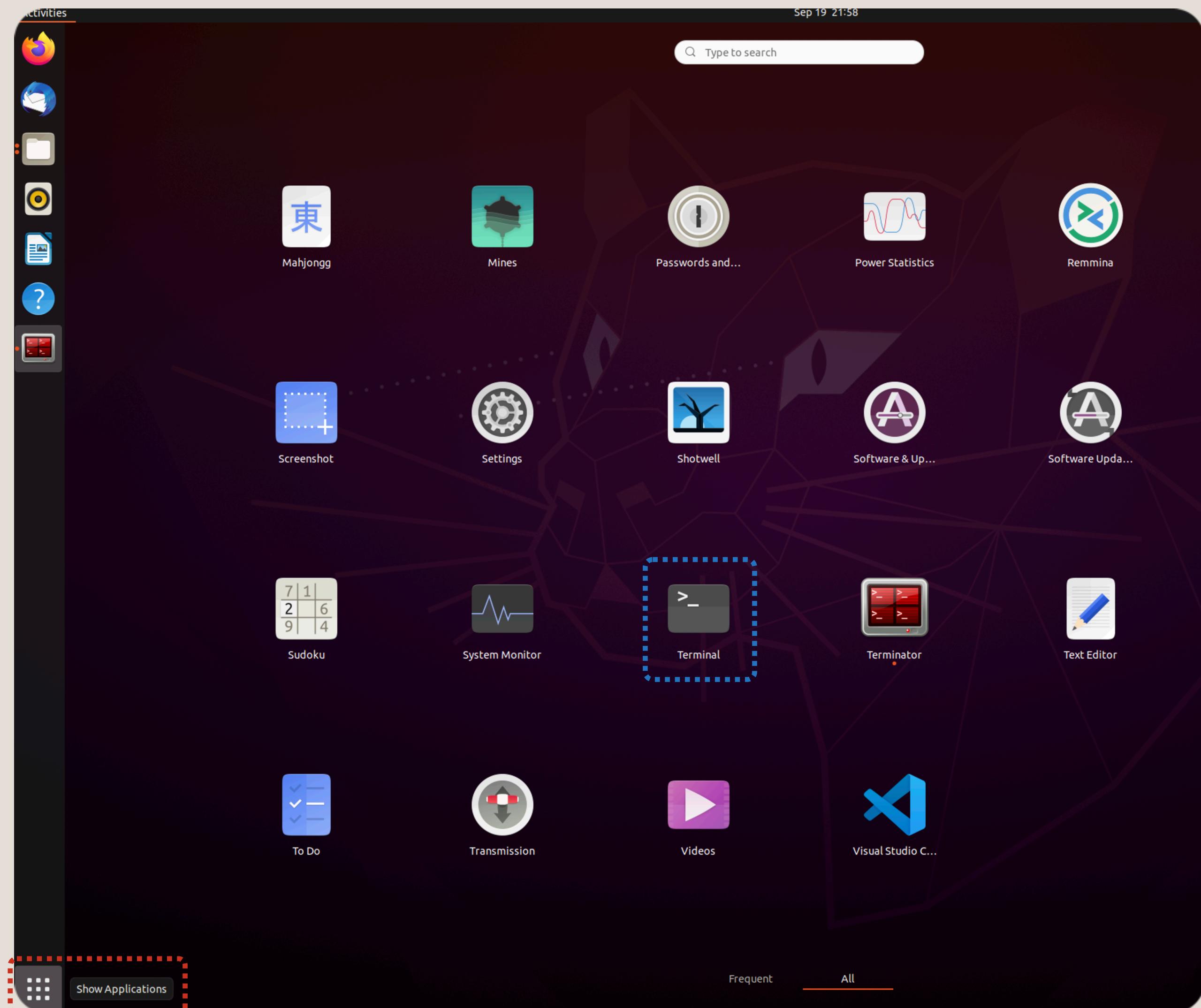
! First copy your one-time code: 612D-C586
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as movingChurch
```

# Git& Github CLI 설치 on Ubuntu

# Git & Github CLI 설치 on Ubuntu

## Terminal 실행

- Show Application → Terminal



## Git& Github CLI 설치 on Ubuntu

### 필요한 종속성 Installation

- 하기 커맨드를 순차적으로 입력▶
  - sudo apt update -y▶
  - sudo apt upgrade -y▶
  - sudo apt install -y curl

## Git & Github CLI 설치 on Ubuntu

### Git Installation

- 하기 커맨드를 순차적으로 입력▶
  - sudo apt install git

## Git & Github CLI 설치 on Ubuntu

### Checking Git Installation

- 하기 커맨드를 순차적으로 입력◦
  - git -version◦
- 아래와 같은 결과가 나오면 설치 성공

```
parallels@ubuntu-linux-20-04-desktop:~$ git --version
git version 2.25.1
```

## Git & Github CLI 설치 on Ubuntu

### Github CLI Installation

- 하기 커맨드를 순차적으로 입력

```
• type -p curl >/dev/null || (sudo apt update & sudo apt install curl -y)  
  
• curl -fsSL https://cli.github.com/packages/githubcli-archive-keyring.gpg | sudo dd of=/usr/share/keyrings/githubcli-archive-keyring.gpg \ & sudo chmod go+r /usr/share/keyrings/githubcli-archive-keyring.gpg \ & echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/githubcli-archive-keyring.gpg] https://cli.github.com/packages stable main" | sudo tee /etc/apt/sources.list.d/github-cli.list > /dev/null \ & sudo apt update \ & sudo apt install gh -y
```

## Git & Github CLI 설치 on Ubuntu

### Checking Github CLI Installation

- 하기 커맨드를 순차적으로 입력▶
  - gh -version▶
- 아래와 같은 결과가 나오면 설치 성공

```
parallels@ubuntu-linux-20-04-desktop:~$ gh --version
gh version 2.35.0 (2023-09-19)
https://github.com/cli/cli/releases/tag/v2.35.0
```

## Git & Github CLI 설치 on Ubuntu

Login Github using CLI

- Github에 login 하기 위해 하기 커맨드를 입력▣
  - gh auth login▣
- 입력 후 하기와 같이 프로세스 진행

```
parallels@ubuntu-linux-20-04-desktop:~$ gh auth login
? What account do you want to log into? GitHub.com
? You're already logged into github.com. Do you want to re-authenticate? Yes
? What is your preferred protocol for Git operations? HTTPS
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 9127-5C7D
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as movingChurch
```

# 코딩폰트

코딩폰트 개요

코딩폰트 설치 on MacOS

코딩폰트 설치 on Ubuntu

# 코딩폰트 개요

## 코딩폰트 개요

What is coding font?

- 프로그래밍을 할 때 읽기 쉽고 코드의 구조를 명확하게 파악할 수 있도록 디자인된 폰트들이 존재◦
- 일반적으로 “coding font” 또는 “programming font”라고 불림◦
- 코딩에 적합한 폰트를 사용하면, 코드의 가독성이 향상되어 실수를 줄이고 더 효율적으로 작업 가능◦

## 코딩폰트 개요

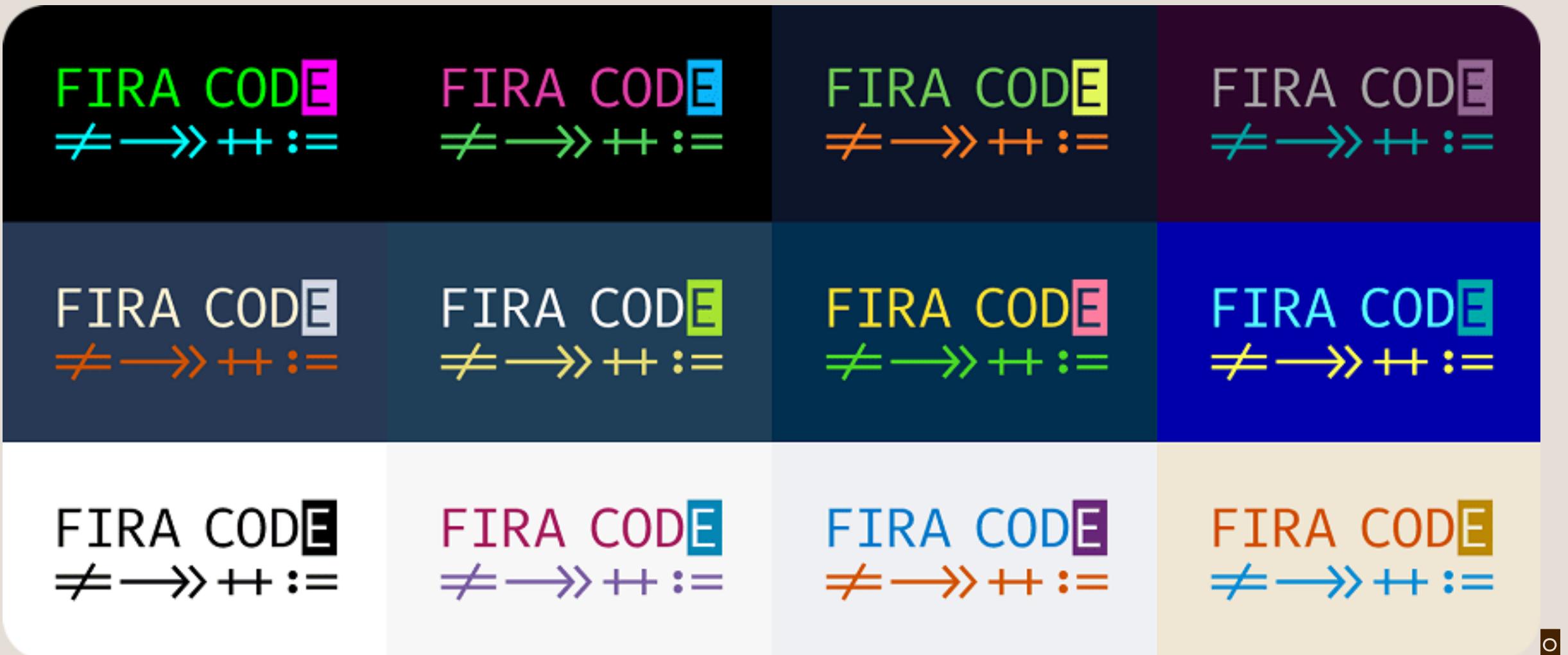
Features of coding font

- 고정폭(Monospaced)
  - 각 문자와 기호가 동일한 너비를 가지는 폰트
  - 이 특징은 코드의 정렬과 들여쓰기에 도움을 주어 가독성을 높임
- 구분력 있는 디자인
  - 일부 문자들은 코드에서 혼란을 야기함(예: 숫자 '1'과 소문자 'l', 대문자 'O'와 숫자 '0')
  - 코딩 폰트는 이러한 문자들이 명확하게 구분되도록 설계되어 있음
- 리간처리(Ligatures)
  - 몇몇 코딩 폰트는 특수한 리간처리 기능을 제공하여 여러 문자나 기호들을 하나의 특별한 기호로 표시하는 것을 지원
  - 예를 들면, '==' 또는 '!='와 같은 기호가 하나의 그래픽으로 표현될 수 있음
- 크기와 높이
  - 코딩 폰트는 일반적으로 코드의 가독성을 최적화하기 위해 적절한 크기와 문자 높이를 가짐

## 코딩폰트 개요

List of coding font

- Fira Code



- Proggy Fonts

```
bool tMath::tExtractRotationEulerXYZ
(
    tVec3& sol1, tVec3& sol2, const tMat4& rot,
    float gimbalZ, tIntervalBias bias
)
{
    // Extraction based on http://staff.city.ac.uk/~sbbh653/publications/euler.pdf
    float gimbalEpsilon = 0.000001f;
    bool gimbalNeg = tApproxEqual(rot.a31, -1.0f, gimbalEpsilon);
    bool gimbalPos = tApproxEqual(rot.a31, 1.0f, gimbalEpsilon);
    if (!gimbalNeg && !gimbalPos)
    {
        sol1.y = -tArcSin(rot.a31);                      // [-Pi/2, Pi/2]
        sol2.y = tGetNormalizedAngle( Pi - sol1.y, bias );
    }
}
```

## 코딩폰트 개요

List of coding font

- DejaVu Sans Mono

DejaVu Sans Mono

Aa Ee Qq

Aa Ee Qq

a

Open source

- Source Code Pro

Regular 400

Almost before we knew it, we had left the ground.

Regular 400 italic

Almost before we knew it, we had left the ground.

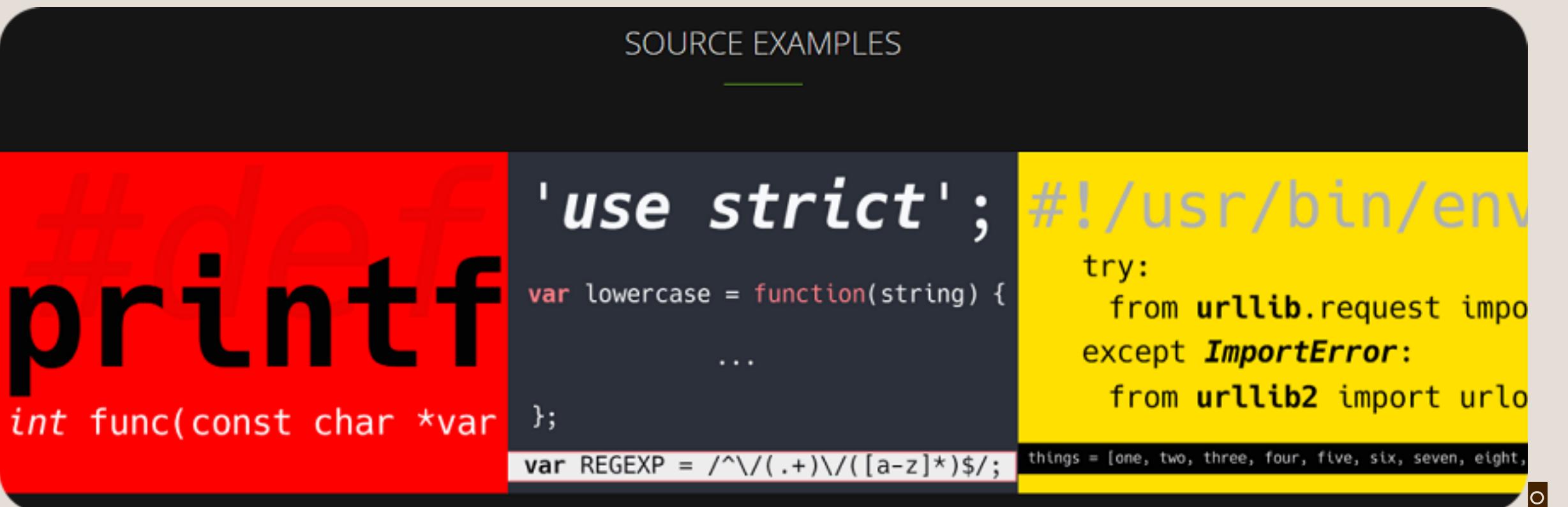
Medium 500

Almost before we knew it, we had left the ground.

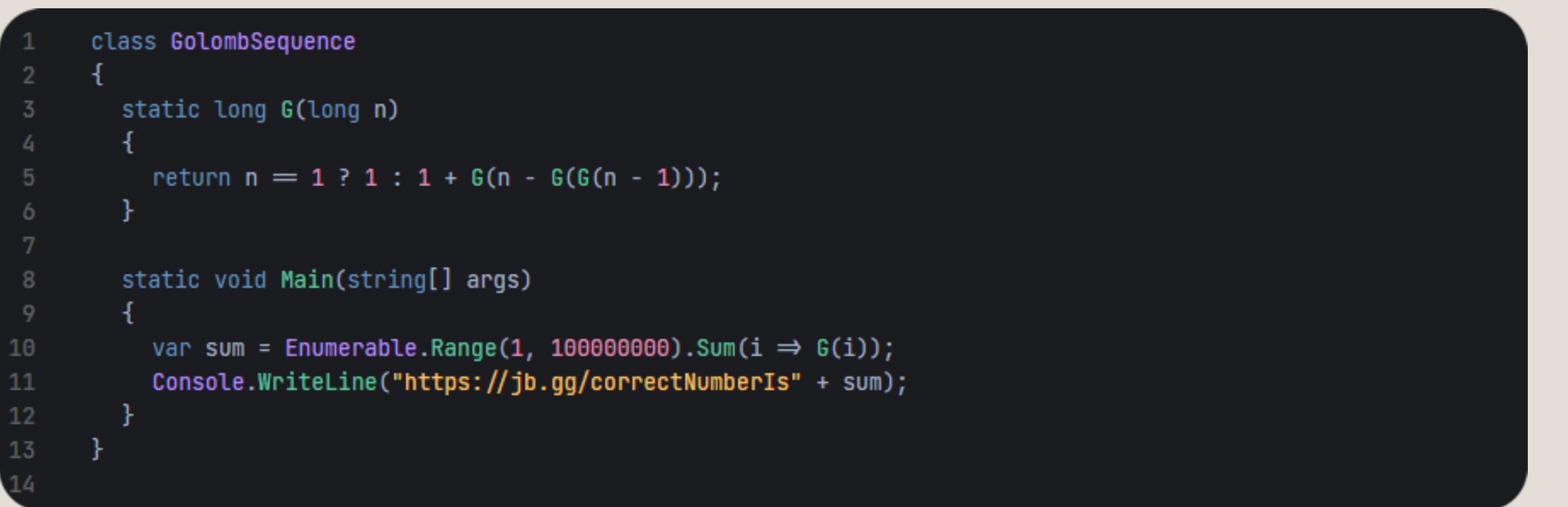
## 코딩폰트 개요

List of coding font

- Hack 



- JetBrains Mono 



# 코딩폰트 설치 on MacOs

## 코딩폰트 설치 on MacOS

Download Fira Code coding font

- 사이트(<https://github.com/tonsky/FiraCode>) 접속
- Readme의 Download & Install 항목에서 **Download icon** 클릭

### Solution ↗

Fira Code is a free monospaced font containing ligatures for common programming multi-character combinations. This is just a font rendering feature: underlying code remains ASCII-compatible. This helps to read and understand code faster. For some frequent sequences like `..` or `//`, ligatures allow us to correct spacing.

### Download & Install ↗



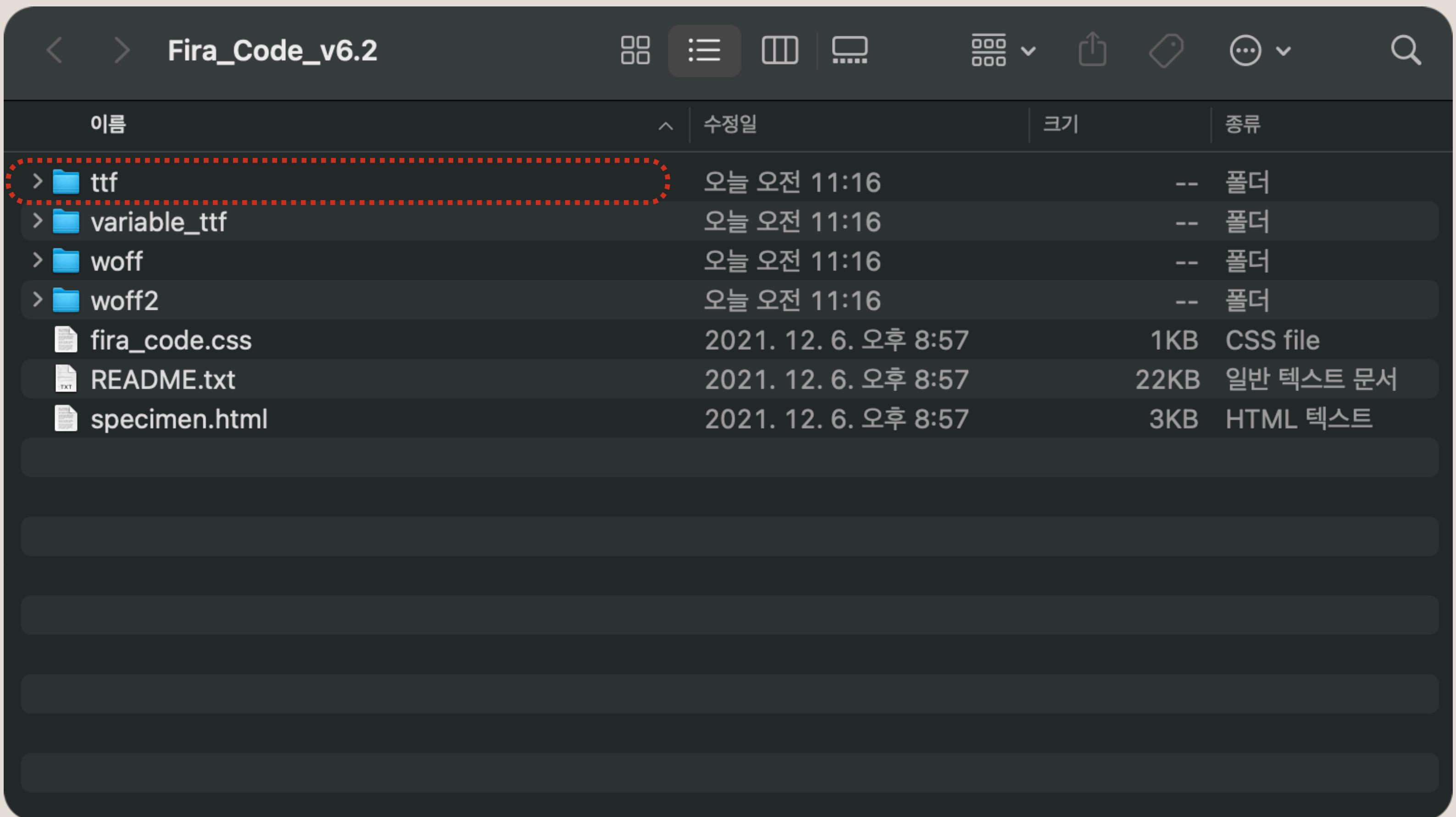
Then:

- [How to Install](#)
- [Troubleshooting](#)
- [News & Updates](#)

## 코딩폰트 설치 on MacOS

Install Fira Code coding font

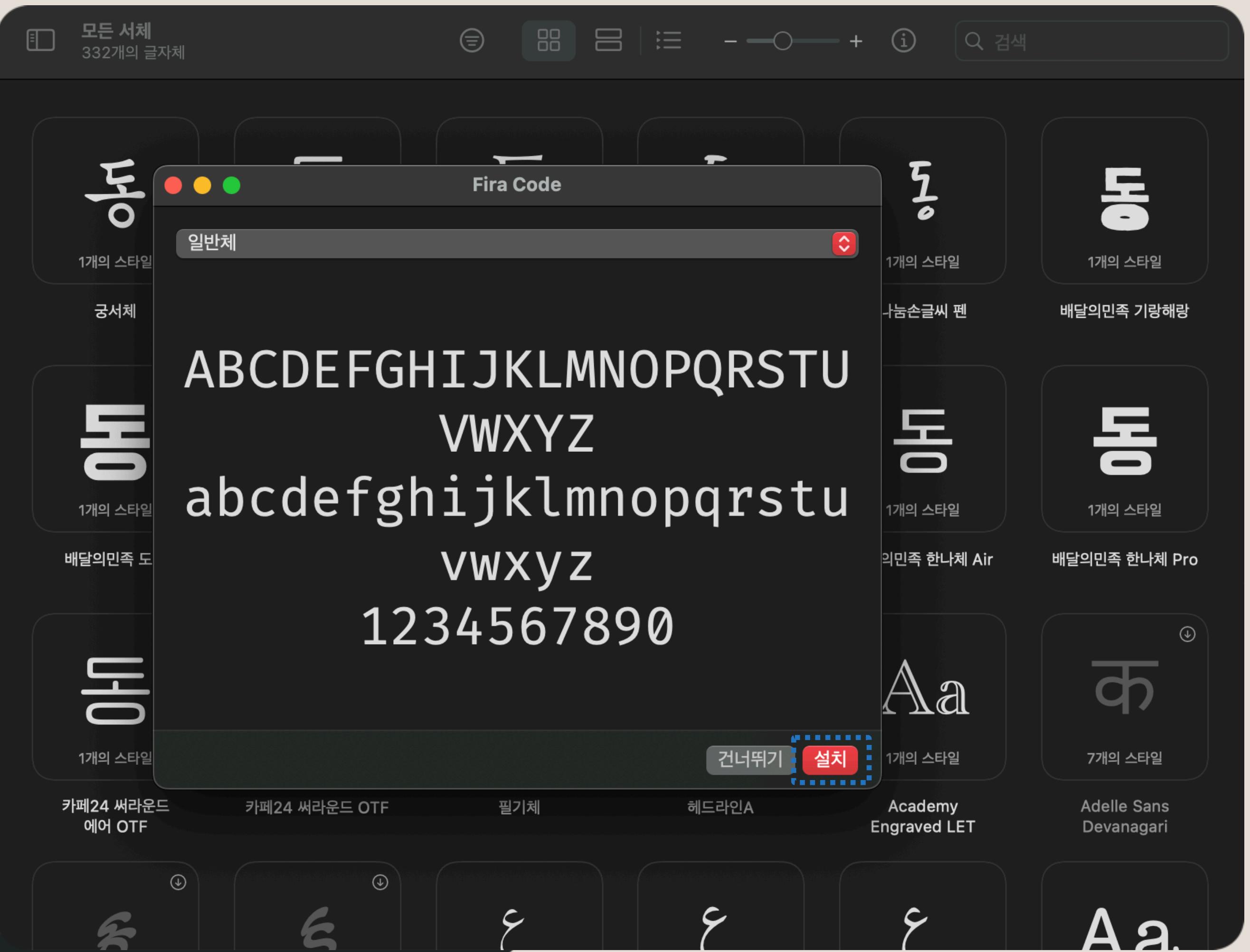
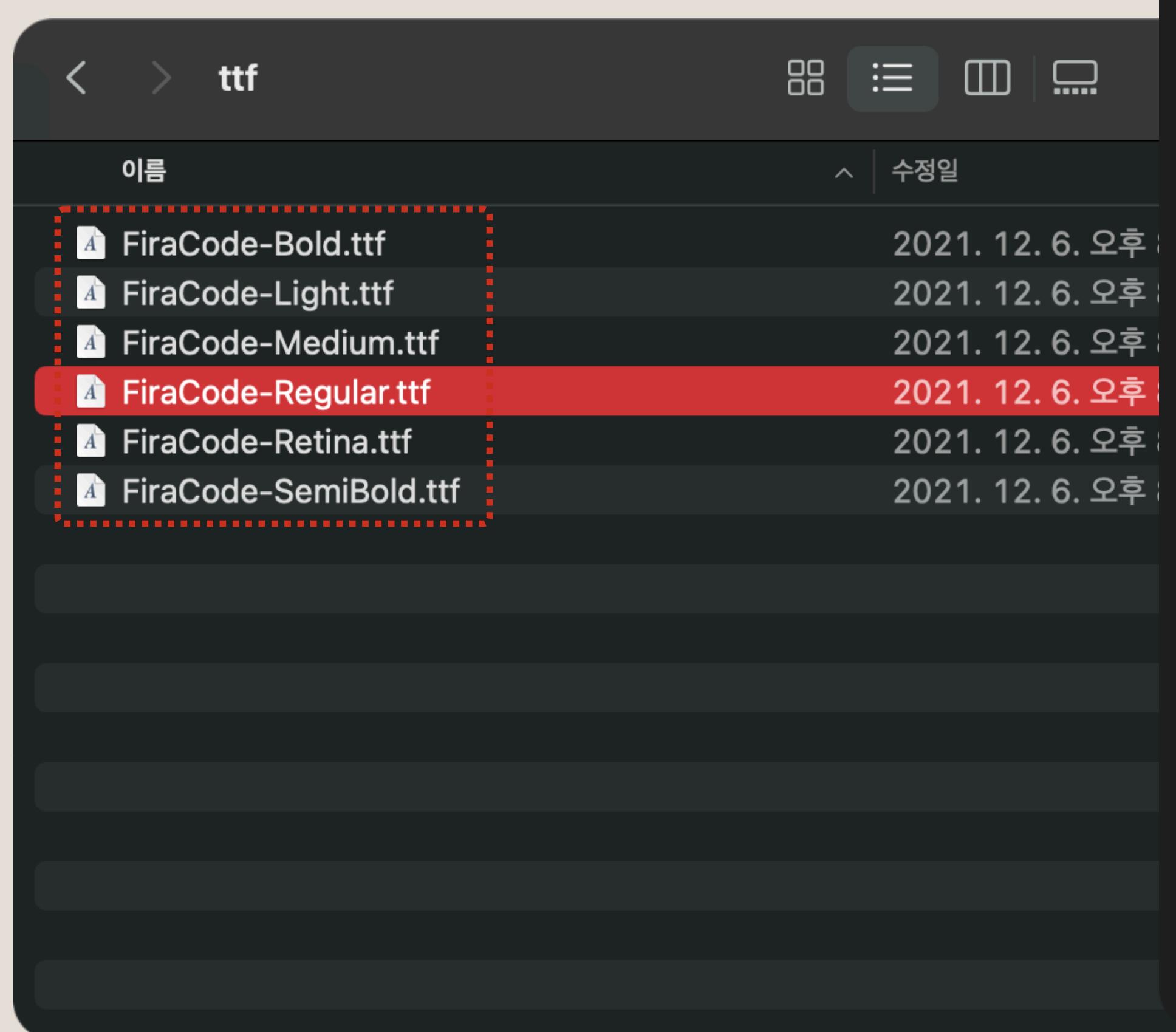
- 다운로드 된 파일 압축 해제
- 압축을 폰 폴더로 이동
- ttf 폴더로 이동



# 코딩폰트 설치 on MacOS

# Install Fira Code coding font

- 모든 파일에 대해 파일을 하나씩 실행
  - 설치 진행



# 코딩폰트 설치 on MacOS

# Test Fira Code coding font

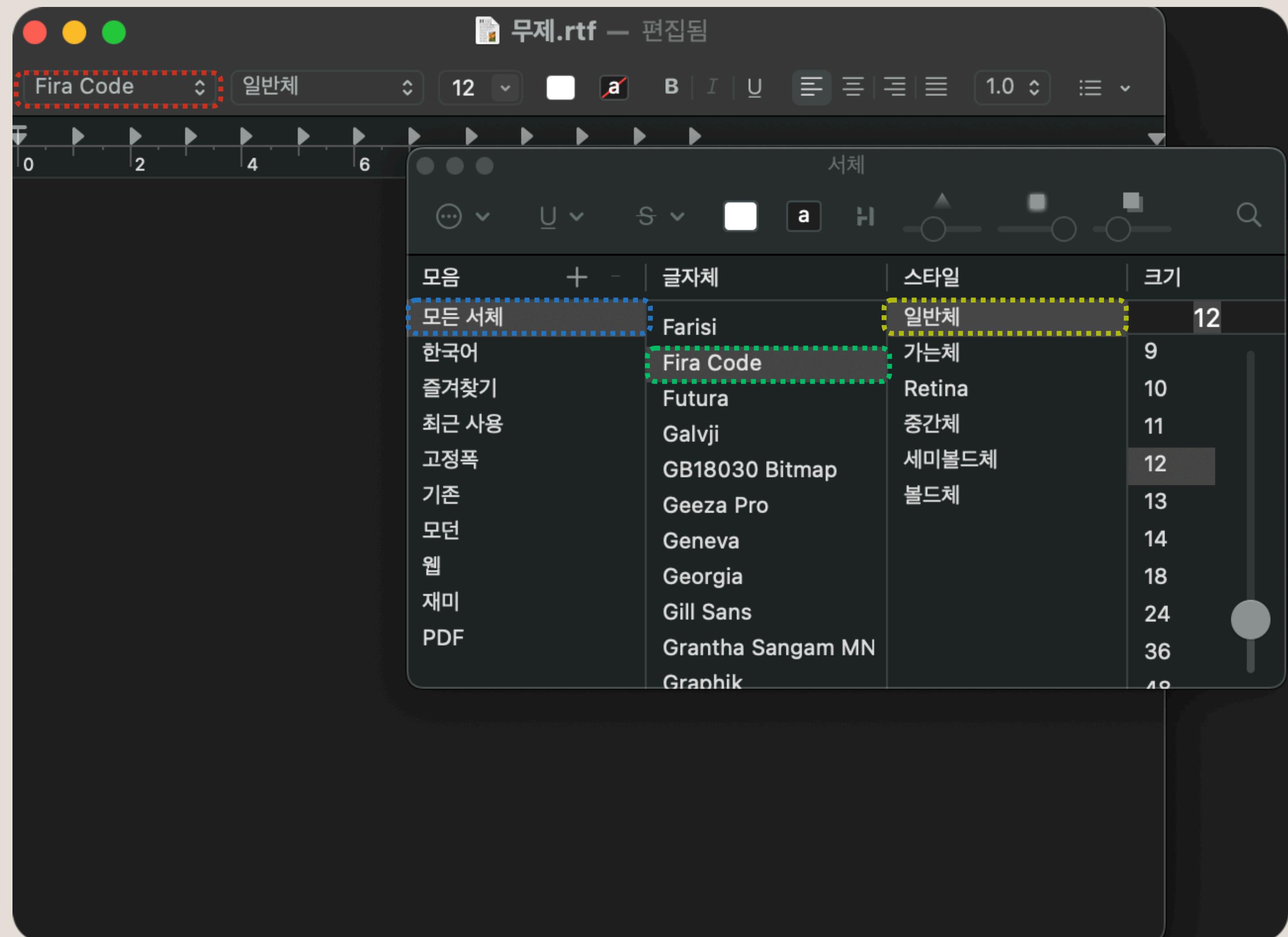
- Lunchpad → 기타 -> 텍스트 편집기
  - 새로운 문서



## 코딩폰트 설치 on MacOS

Test Fira Code coding font

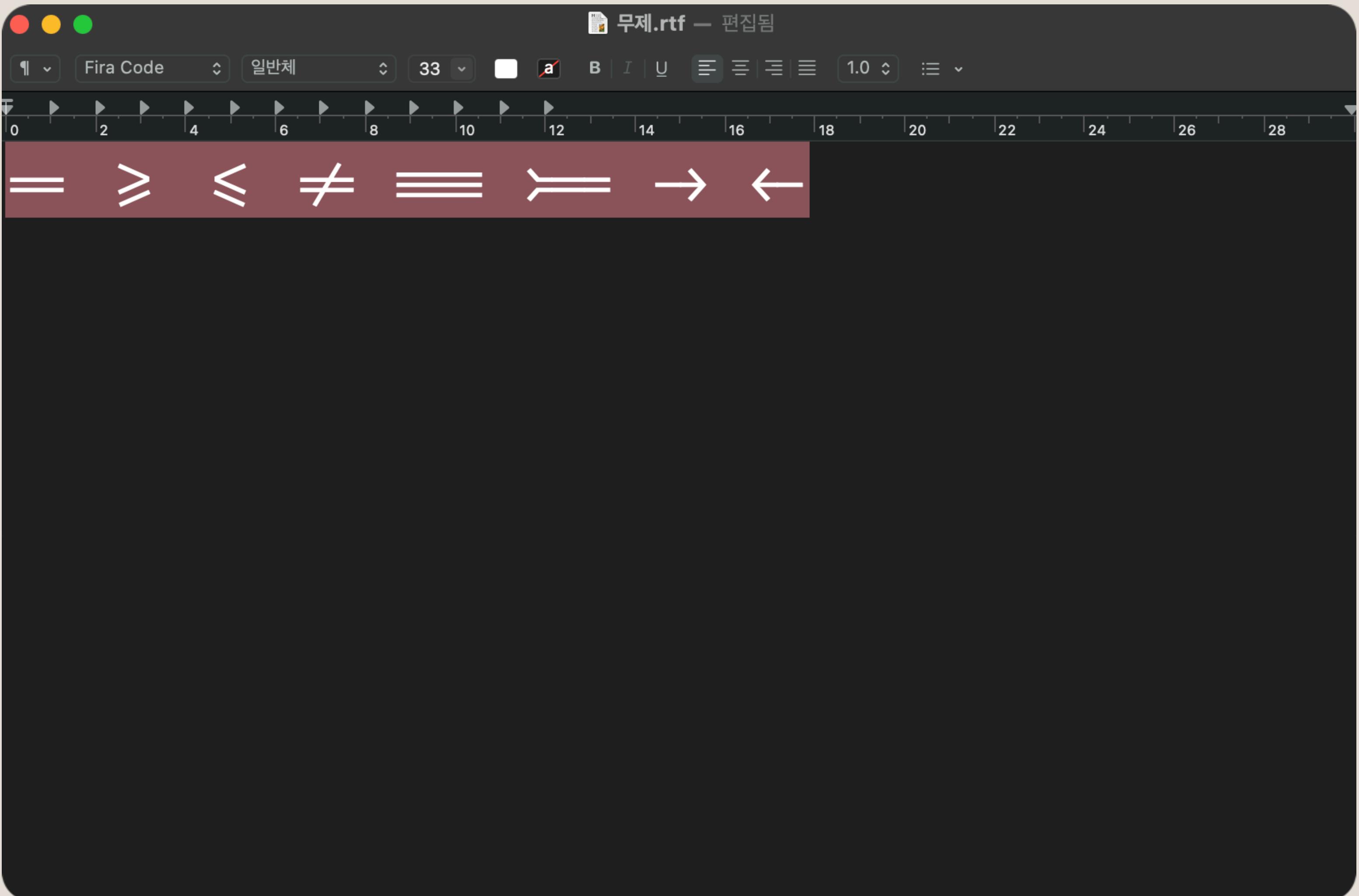
- 글꼴 선택 - 서체 보기... 실행
- 모든 서체 - Fira Code - 일반체 선택



## 코딩폰트 설치 on MacOS

Test Fira Code coding font

- 텍스트 편집기를 사용하여 Ligatures 검증



# 코딩폰트 설치 on Ubuntu

## 코딩폰트 설치 on Ubuntu

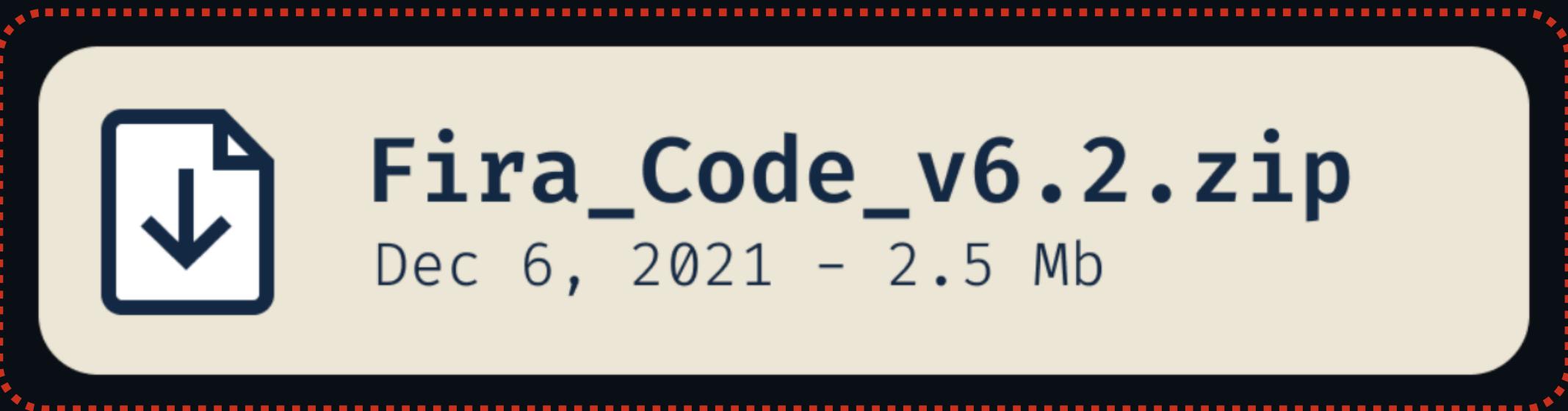
Download Fira Code coding font

- 사이트(<https://github.com/tonsky/FiraCode>) 접속
- Readme의 Download & Install 항목에서 **Download icon** 클릭

### Solution ↗

Fira Code is a free monospaced font containing ligatures for common programming multi-character combinations. This is just a font rendering feature: underlying code remains ASCII-compatible. This helps to read and understand code faster. For some frequent sequences like `..` or `//`, ligatures allow us to correct spacing.

### Download & Install ↗



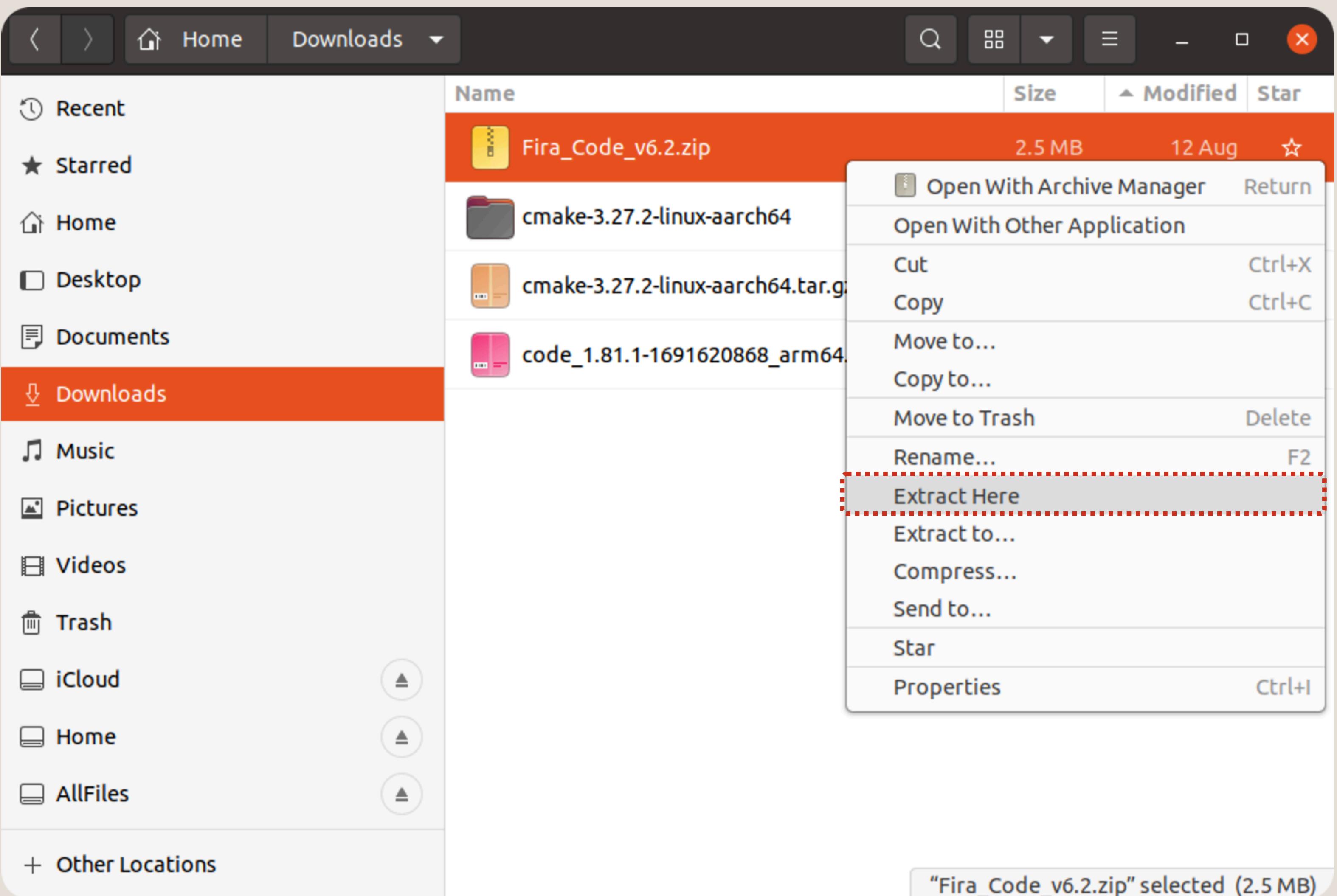
Then:

- [How to Install](#)
- [Troubleshooting](#)
- [News & Updates](#)

## 코딩폰트 설치 on Ubuntu

Install Fira Code coding font

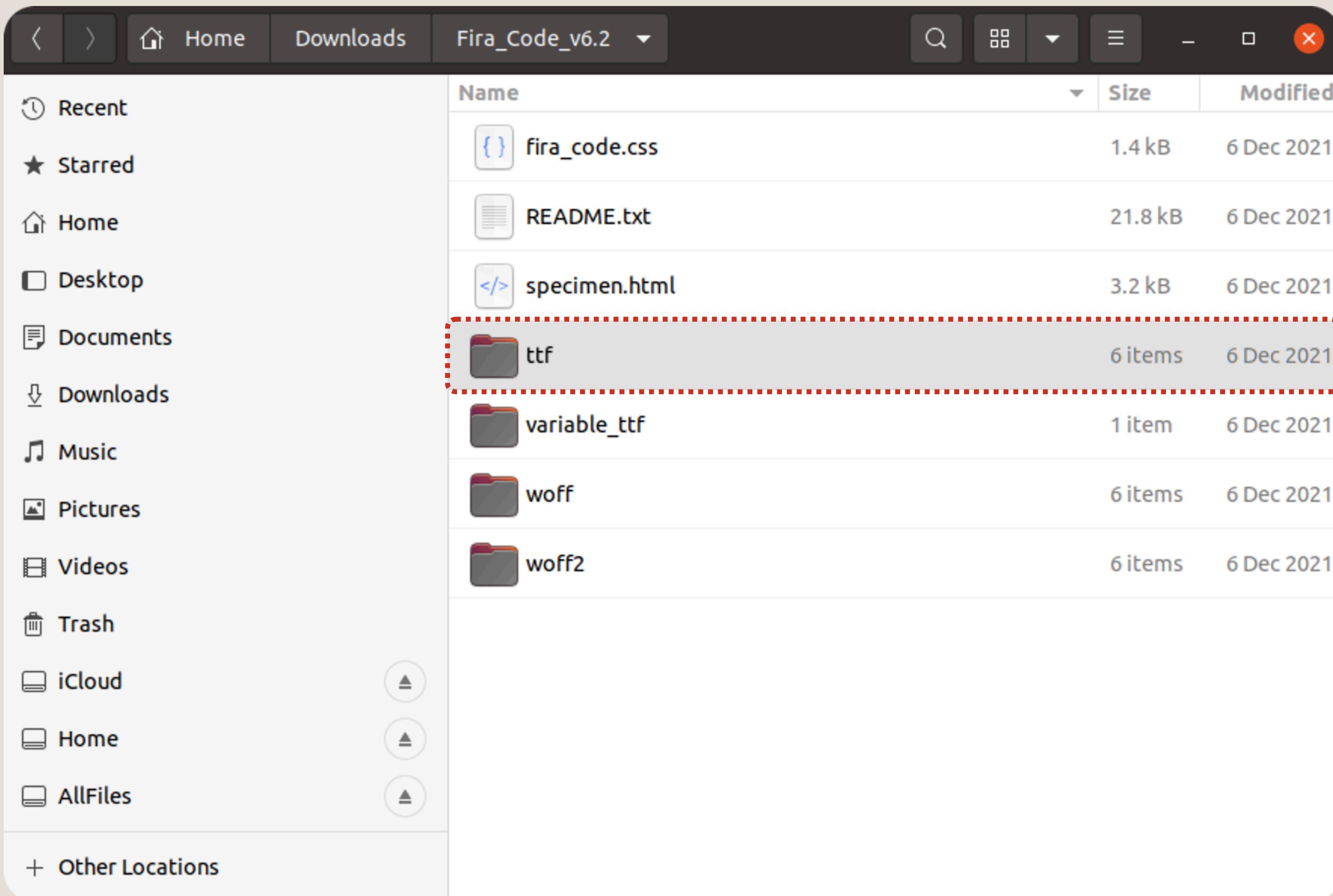
- 압축 풀기 진행



## 코딩폰트 설치 on Ubuntu

Install Fira Code coding font

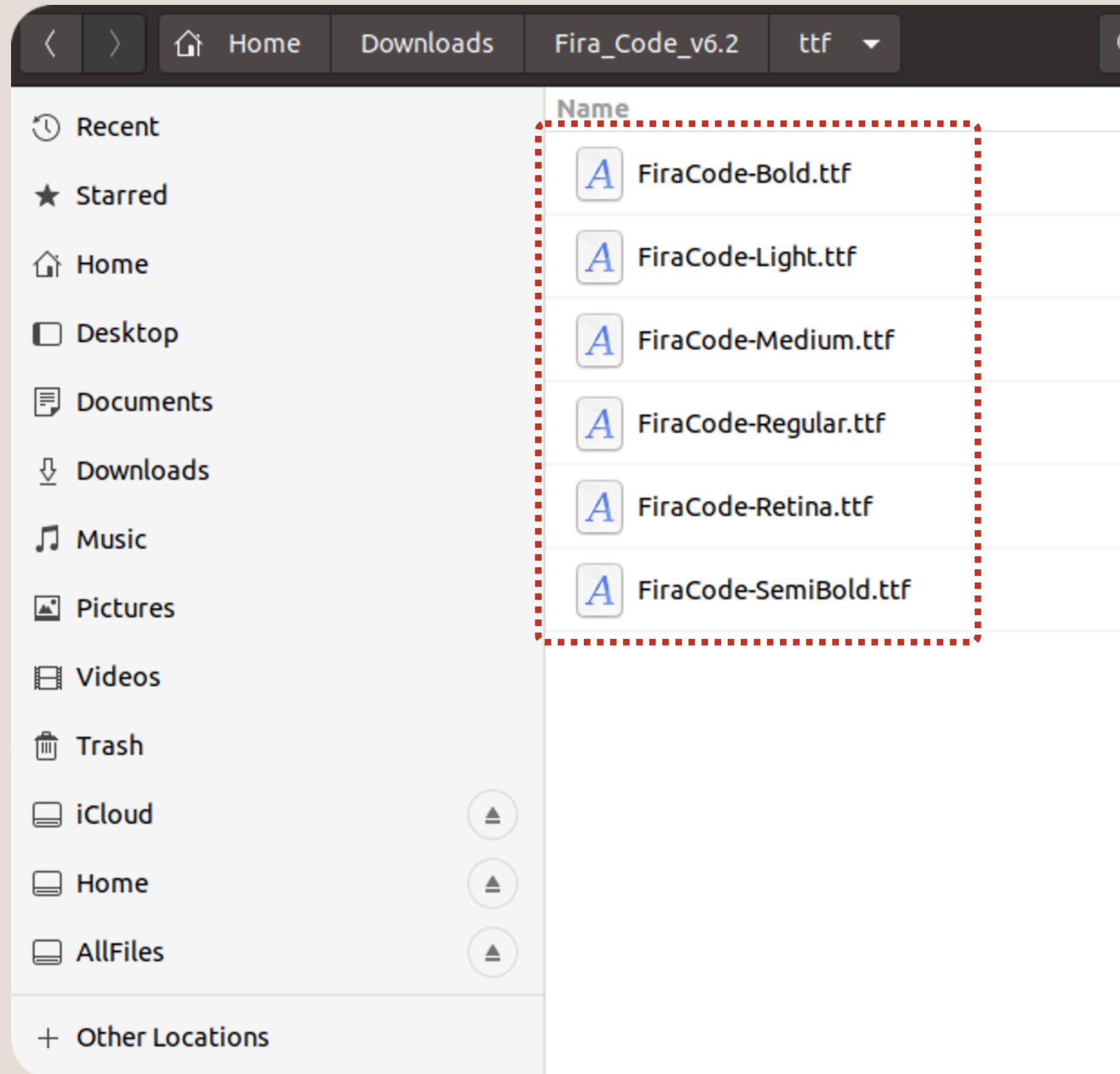
- 압축을 푼 폴더로 이동
- ttf 폴더로 이동



## 코딩폰트 설치 on Ubuntu

Install Fira Code coding font

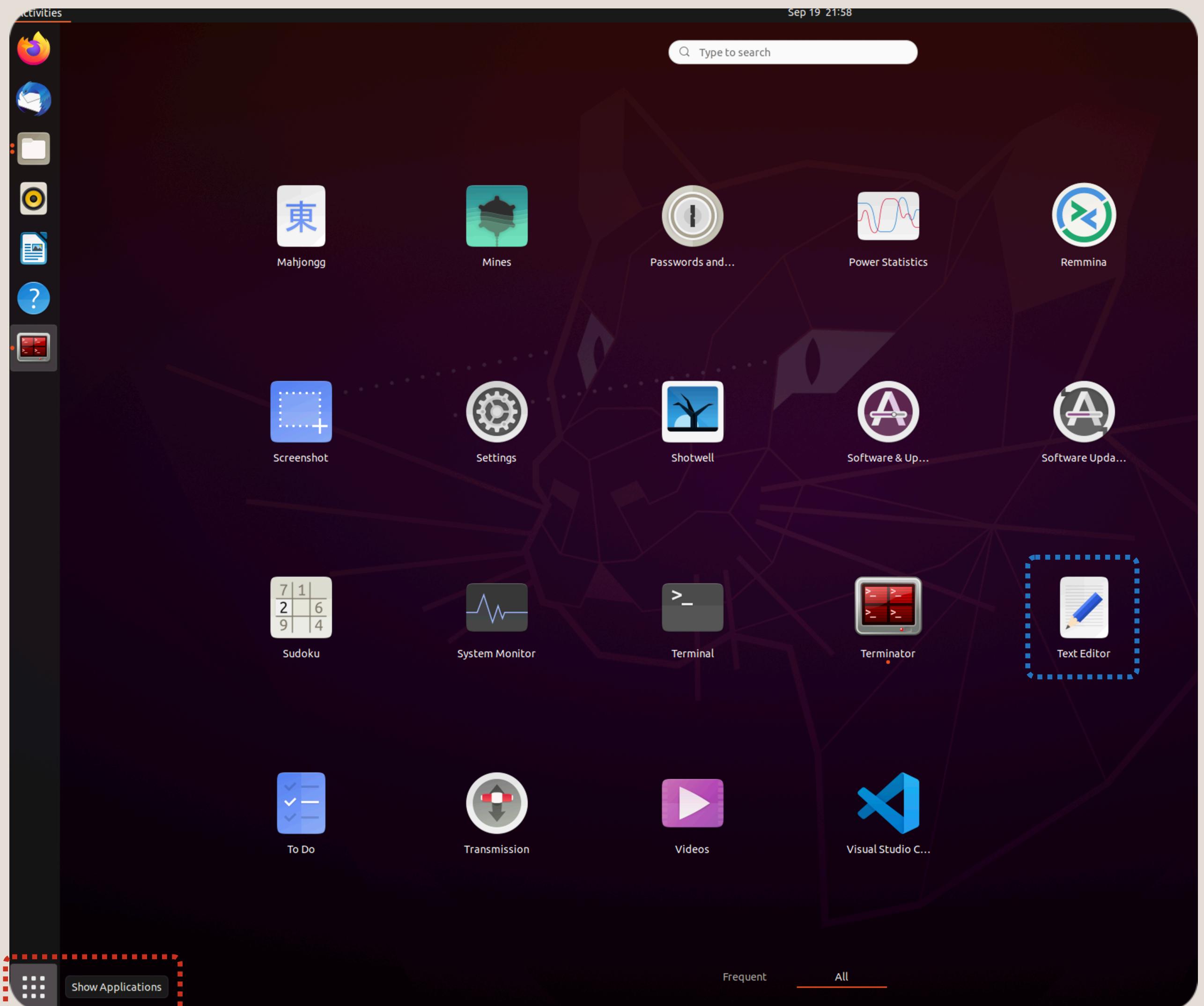
- 모든 파일에 대해 파일을 하나씩 실행
- 설치 진행



## 코딩폰트 설치 on Ubuntu

Test Fira Code coding font

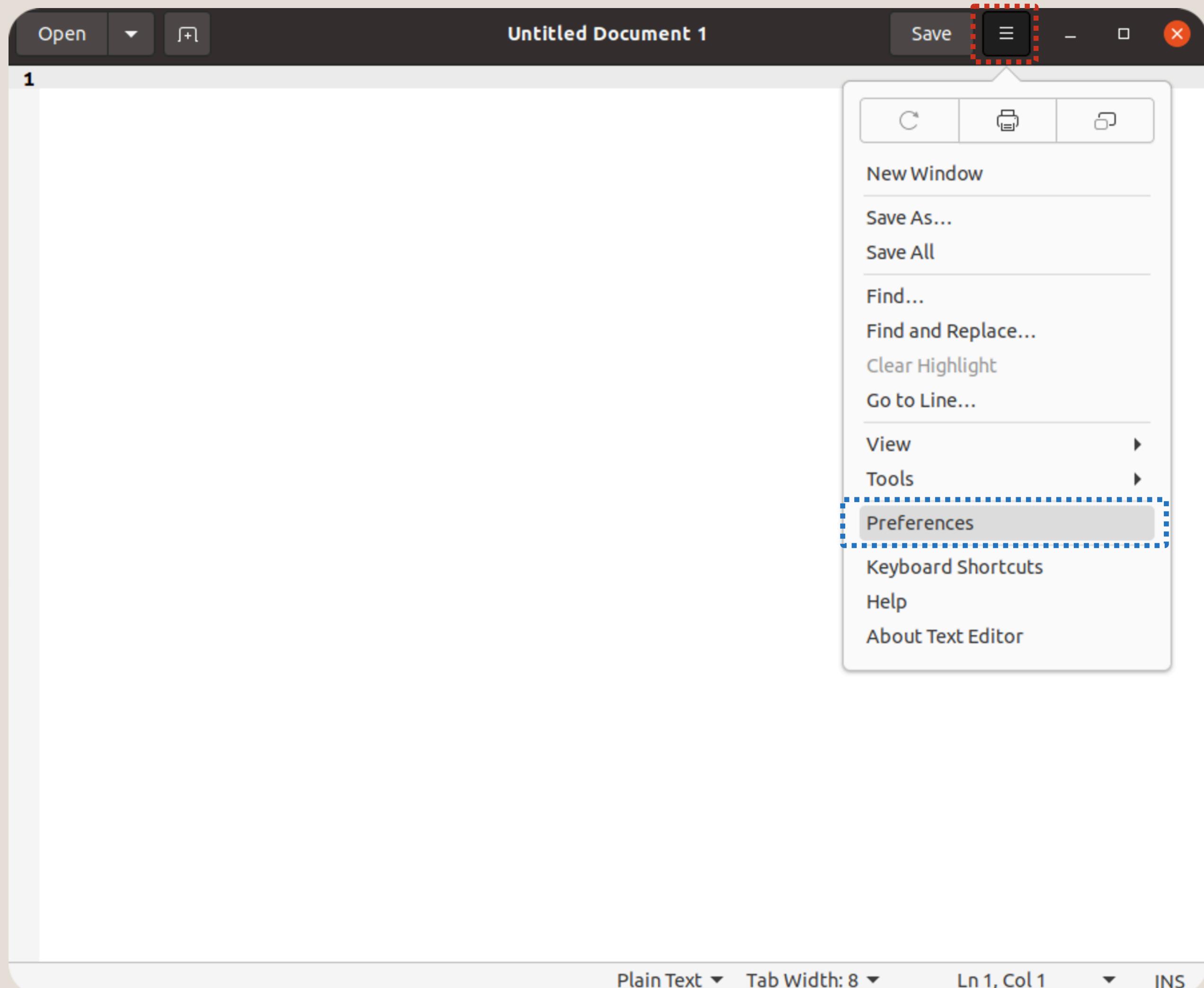
- Show Applications - Text Editor 실행



## 코딩폰트 설치 on Ubuntu

Test Fira Code coding font

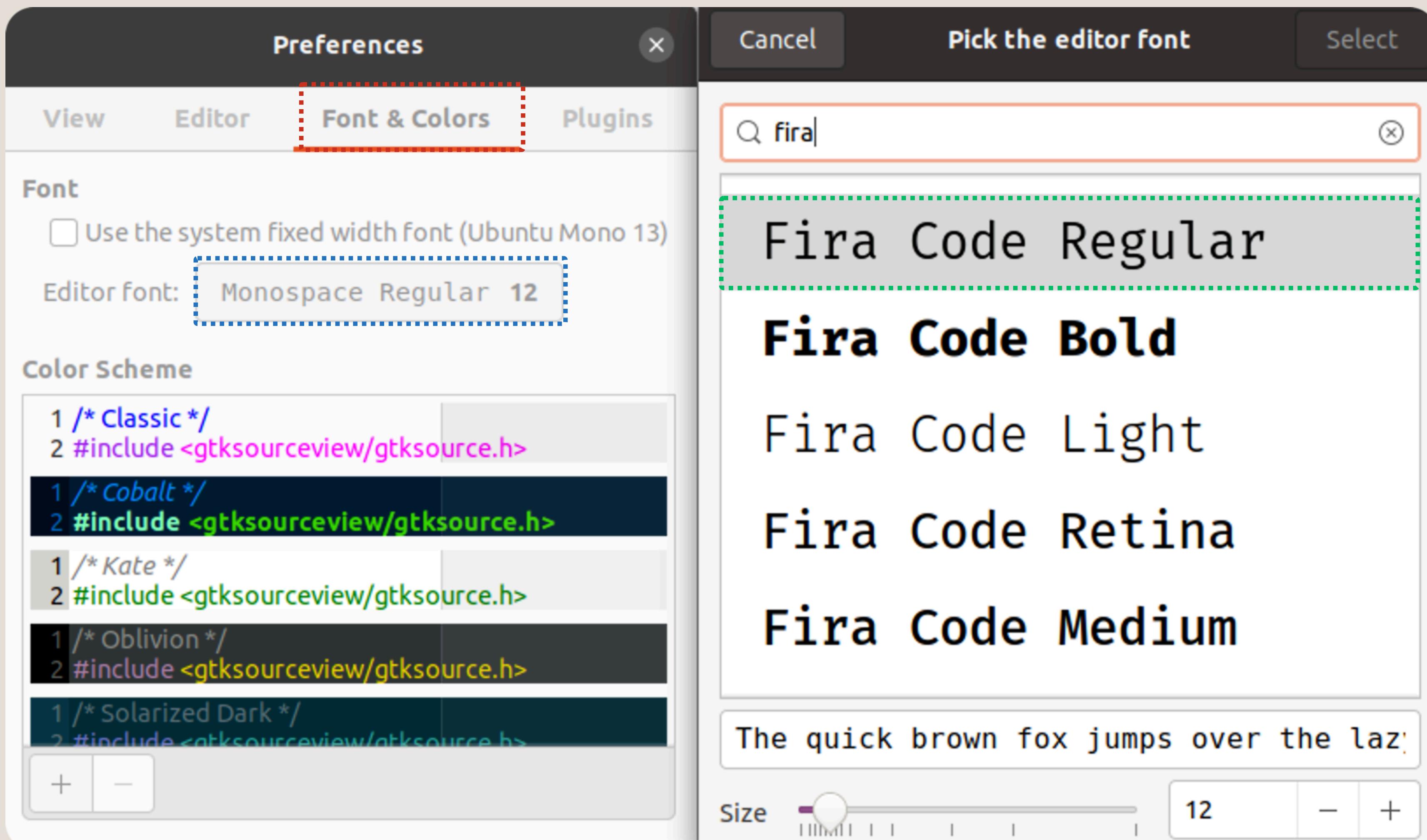
- 설정 - Preferences 실행



## 코딩폰트 설치 on Ubuntu

Test Fira Code coding font

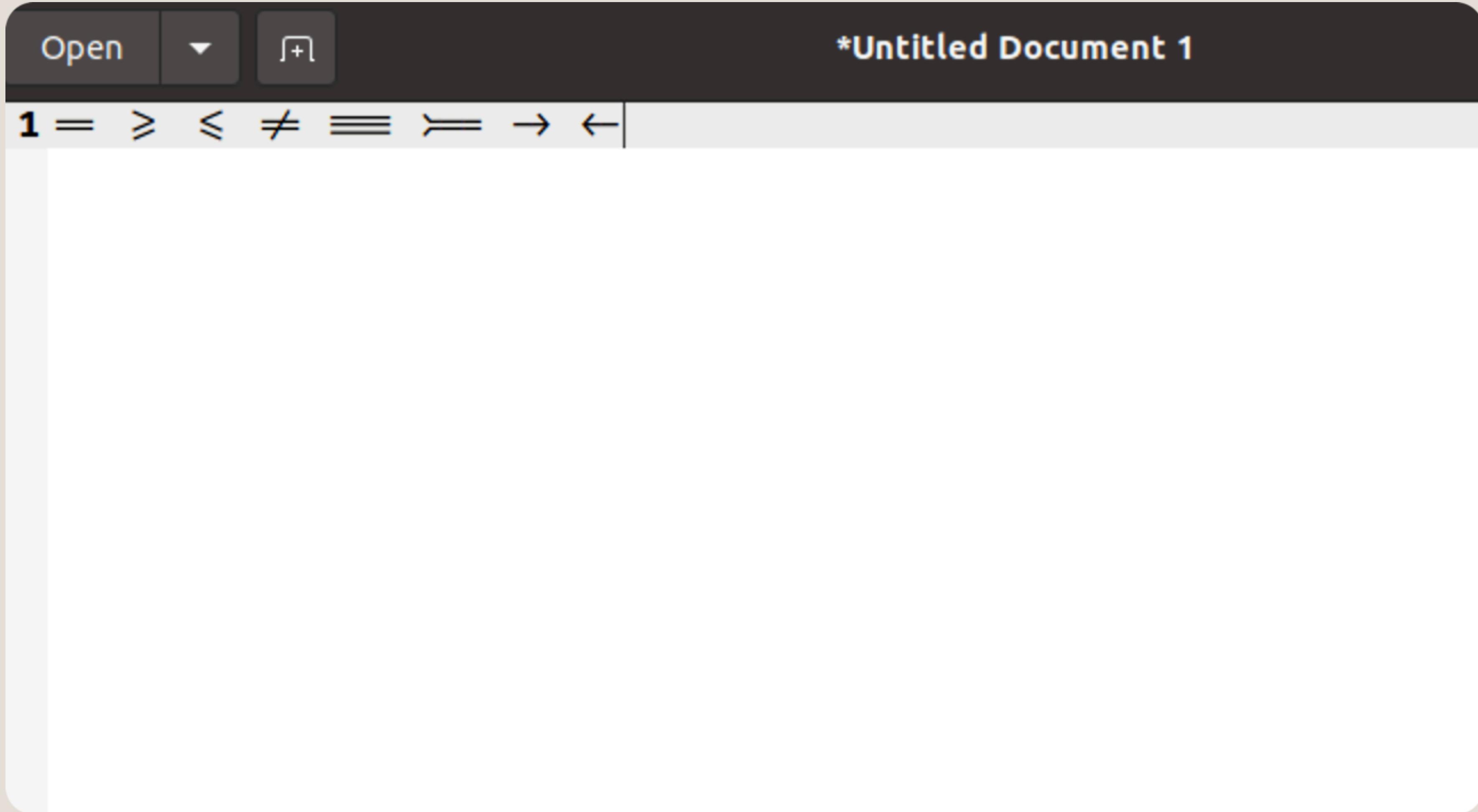
- Font & Colors - Editor font - Fira Code Regular 적용



## 코딩폰트 설치 on Ubuntu

Test Fira Code coding font

- Text Editor를 사용하여 Ligatures 검증■



# Integrated Development Environment(IDE)

IDE 개요◦

VSCode 설치 on MacOS◦

VSCode 설치 on Ubuntu◦

VSCode 기본 설정◦

C++ 개발을 위한 VSCode 확장기능 설정◦

VSCode 권장 확장 기능◦

VSCode 테마 설정◦

예제 코드 빌드 using VSCode◦

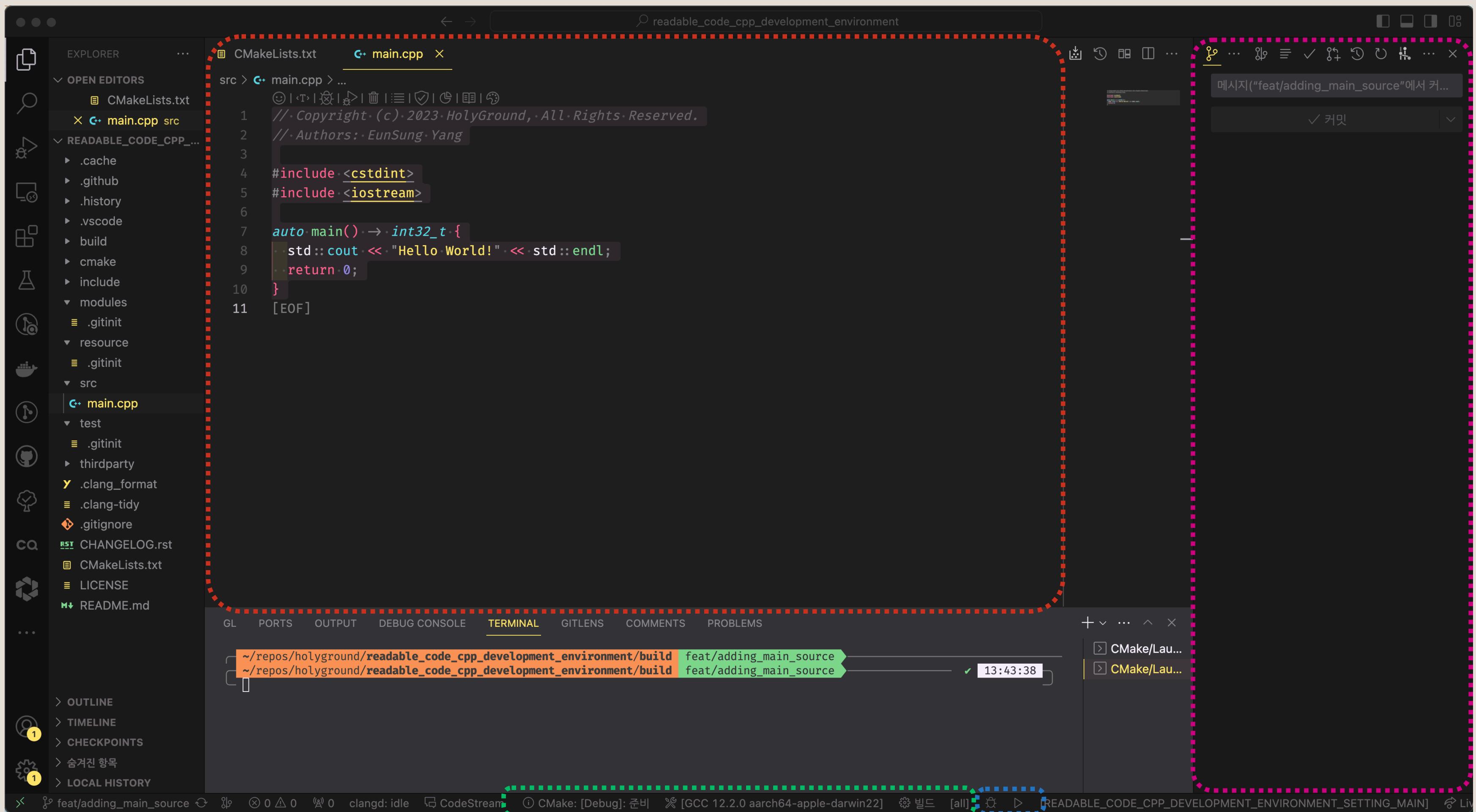
VSCode로 Git 사용하기

# IDE 개요

# IDE 개요

# What is IDE

- 통합 개발 환경(Integrated Development Environment)◦
  - 소프트웨어 개발을 위한 프로그래밍 환경을 한곳에서 제공하는 애플리케이션◦
  - 코드 작성, 디버깅, 테스팅 및 프로젝트 관리 기능을 통합하는 종합적인 환경을 제공◦



## IDE 개요

### Features of IDE

- 코드 편집기
  - 소스 코드를 작성하고 편집하는 도구
  - 문법 강조(syntax highlighting), 자동 완성(auto-completion), 인덴트 자동 조절(auto-indentation) 등의 기능을 제공

- 컴파일러/인터프리터
  - 소스 코드를 바이너리 코드로 변환하거나 직접 실행시켜주는 도구
  - 어떤 언어를 사용하는지에 따라 컴파일러나 인터프리터가 내장되어 있음

- 디버거
  - 프로그램의 실행을 모니터링하고, 오류를 찾아내고 수정하는 도구
  - 중단점(breakpoint) 설정, 단계별 실행(step-through), 변수 감시(variable watch) 등의 기능을 제공

- 빌드 자동화 도구
  - 소스 코드에서 실행 가능한 애플리케이션을 만들어주는 도구
  - 라이브러리 관리, 빌드 스크립트 실행, 패키징 등의 작업을 자동화

- 테스트 도구
  - 단위 테스트(unit test)나 통합 테스트(integration test) 등을 수행하는 도구
  - 자동 테스트 실행, 테스트 결과 리포트 생성 등의 기능을 제공

- 버전 관리 시스템
  - 코드의 이전 버전을 추적
  - 여러 사람이 동시에 작업할 때 충돌을 관리하는 도구
  - Git 등의 버전 관리 시스템과 연동되어 사용됨

## IDE 개요

### List of IDE

- Eclipse

- 주 지원 언어: Java
- 라이센스: 무료 (Eclipse Public License)
- 기타 지원 언어: C/C++, PHP, Python, Perl, Ruby 등 다양한 언어 지원
- 특장점: 확장성이 좋아 다양한 플러그인으로 기능을 확장할 수 있음

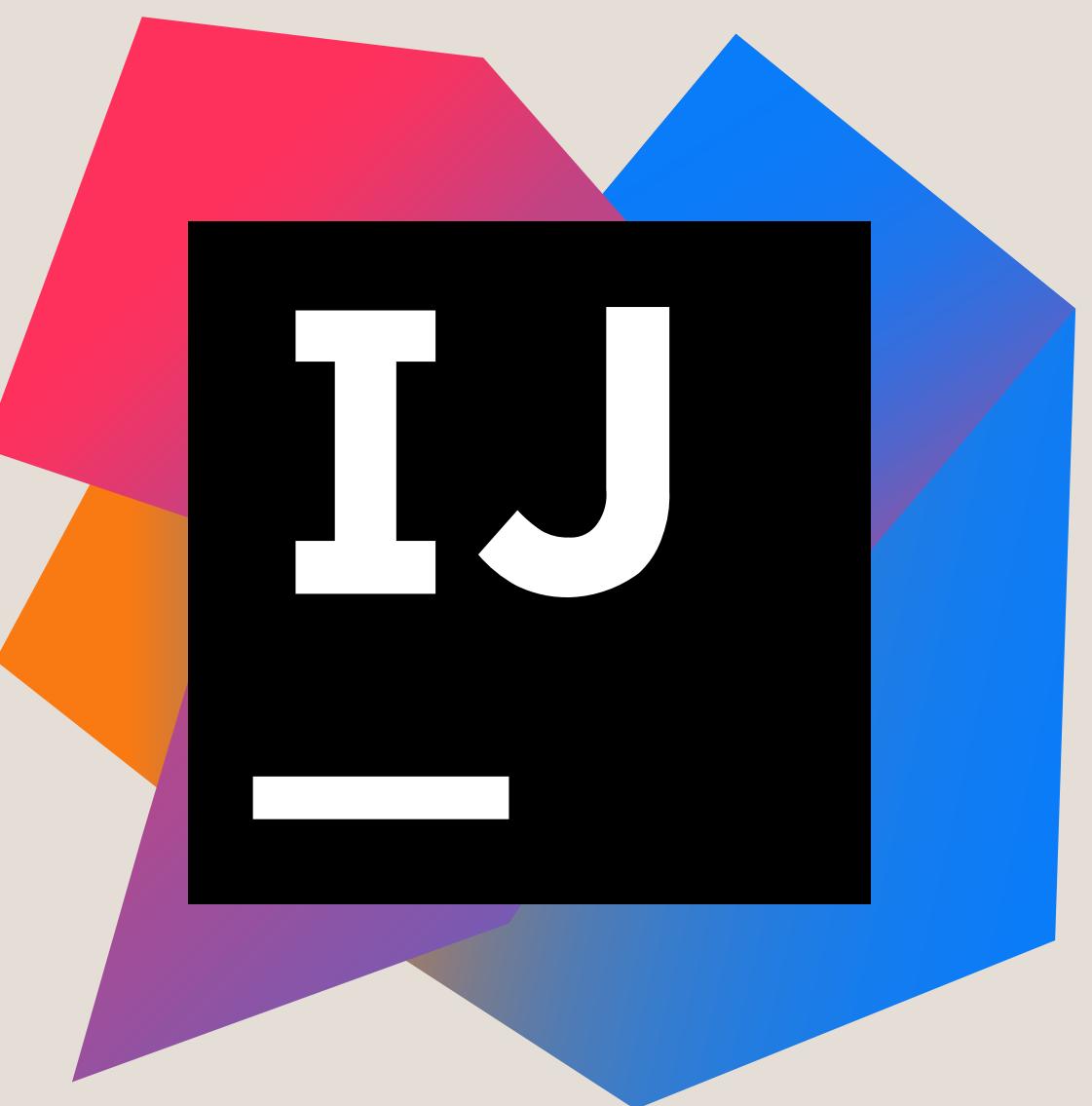


## IDE 개요

### List of IDE

- IntelliJ IDEA

- 주 지원 언어: Java
- 라이센스: 무료/유료 (Community Edition은 무료, Ultimate Edition은 유료)
- 기타 지원 언어: Kotlin, Groovy, Scala 등
- 특장점: 강력한 자동 완성, 리팩토링 도구, 효율적인 네비게이션 기능



## IDE 개요

### List of IDE

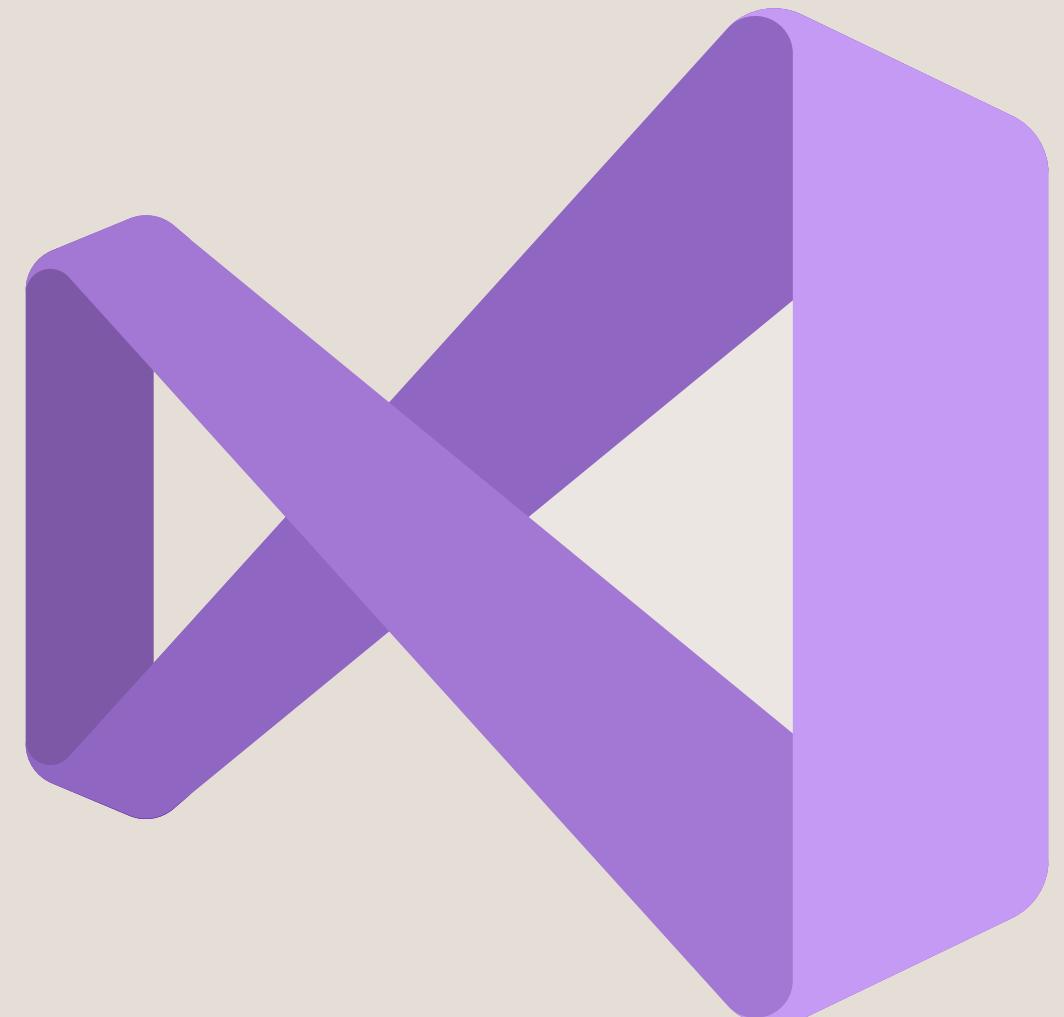
- PyCharm•
  - 주 지원 언어: Python•
  - 라이센스: 무료/유료 (Community Edition은 무료, Professional Edition은 유료)•
  - 기타 지원 언어: HTML, JS, SQL 등 웹 개발 관련 언어•
  - 특장점: 파이썬 개발에 최적화된 강력한 기능 (코드 분석, 자동 완성, 웹 개발 도구 등)



## IDE 개요

### List of IDE

- Visual Studio(VS)
  - 주 지원 언어: C#, VB.NET, F#
  - 라이센스: 무료/유료 (Community Edition은 무료, Professional 및 Enterprise Edition은 유료)
  - 기타 지원 언어: C++, JavaScript, Python 등 다양한 언어 지원
  - 특장점: Microsoft .NET 플랫폼과의 강력한 통합, 훌륭한 디버깅 도구

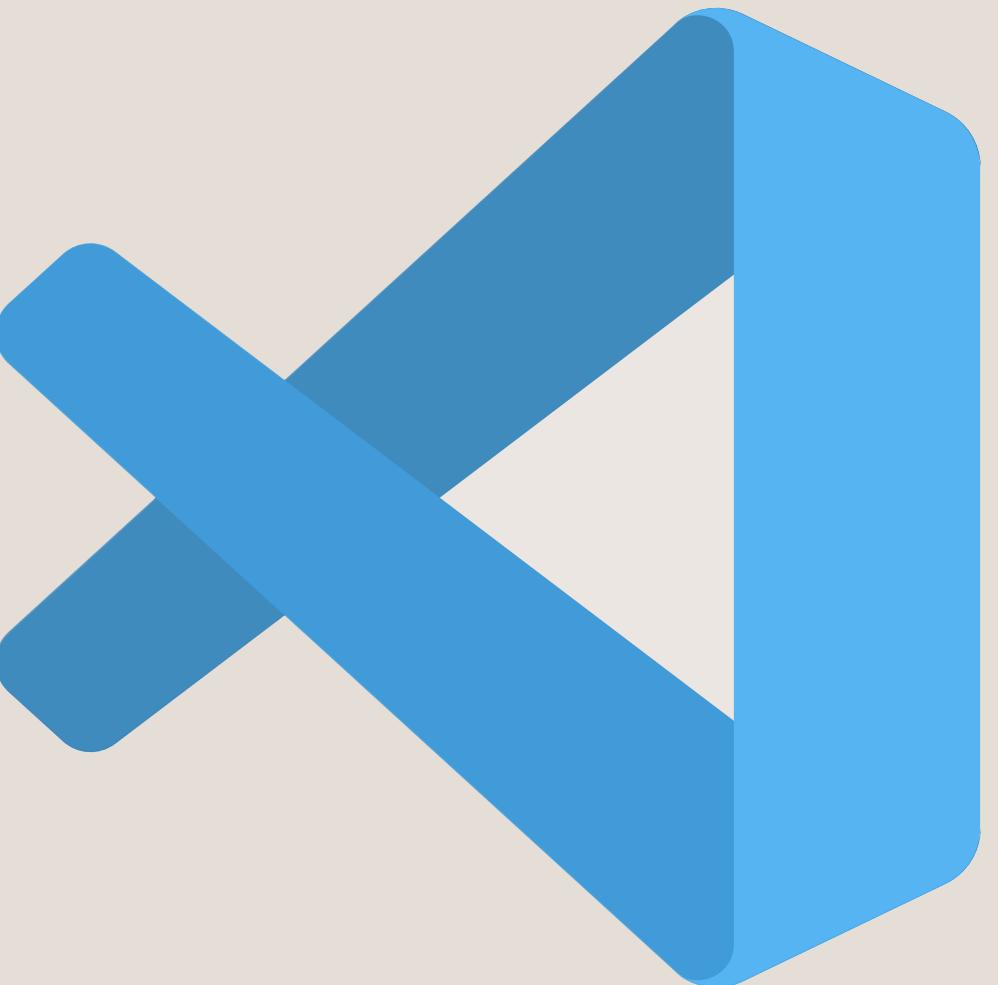


## IDE 개요

### List of IDE

- Visual Studio Code(VSCode)■

- 주 지원 언어: 다양한 언어 지원■
- 라이센스: 무료 (MIT License)■
- 서브 지원 언어: Java, JavaScript, Python, C#, C++, PHP 등 다양한 언어 지원■
- 특장점: 가볍고 빠른 실행, 다양한 언어 지원, 강력한 확장 기능



## IDE 개요

### List of IDE

- Xcode

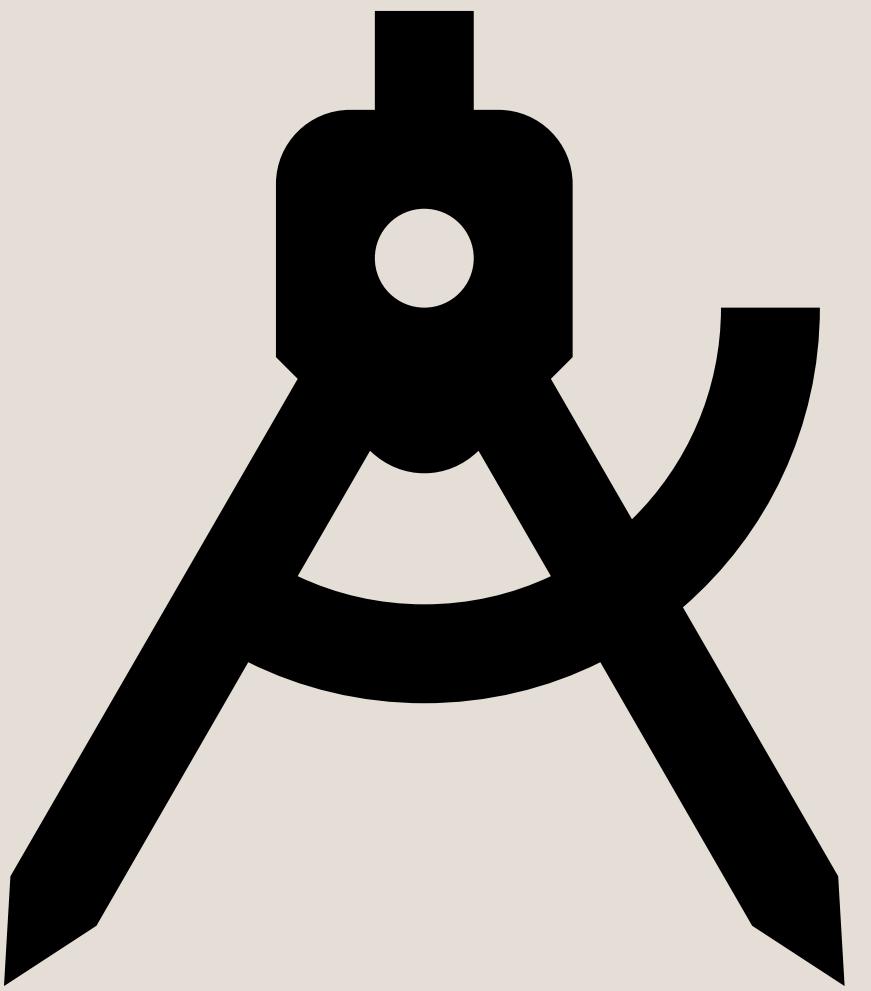
- 주 지원 언어: Swift, Objective-C
- 라이센스: 무료
- 기타 지원 언어: C, C++
- 특장점: iOS 및 macOS 개발에 필요한 모든 도구 제공, Apple 플랫폼과의 강력한 통합



## IDE 개요

### List of IDE

- **Android Studio**
  - 주 지원 언어: Java, Kotlin
  - 라이센스: 무료 (Apache License 2.0)
  - 서브 지원 언어: C/C++
  - 특장점: Android 애플리케이션 개발에 최적화, 강력한 에뮬레이터와 디버깅 도구

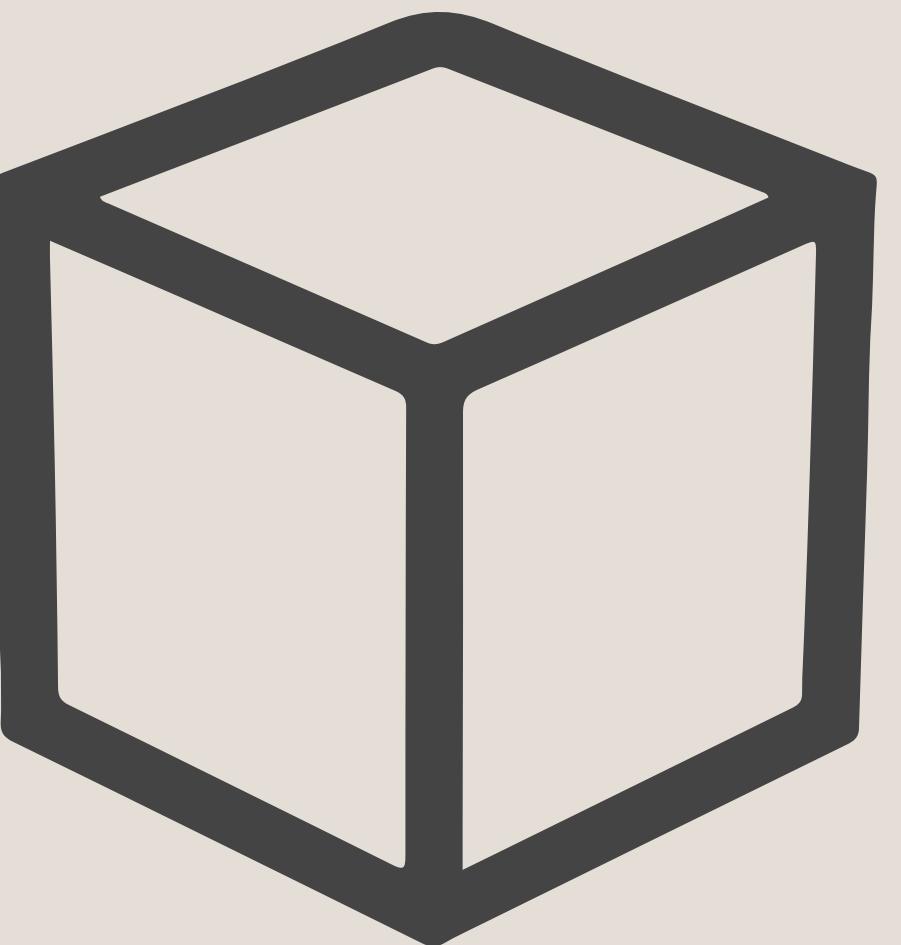


## IDE 개요

### List of IDE

- NetBeans

- 주 지원 언어: Java
- 라이센스: 무료 (Apache License 2.0)
- 기타 지원 언어: C/C++, PHP, JavaScript 등 다양한 언어 지원
- 특장점: 사용하기 쉬운 GUI, 다양한 언어 지원, 강력한 코드 편집과 리팩토링 도구

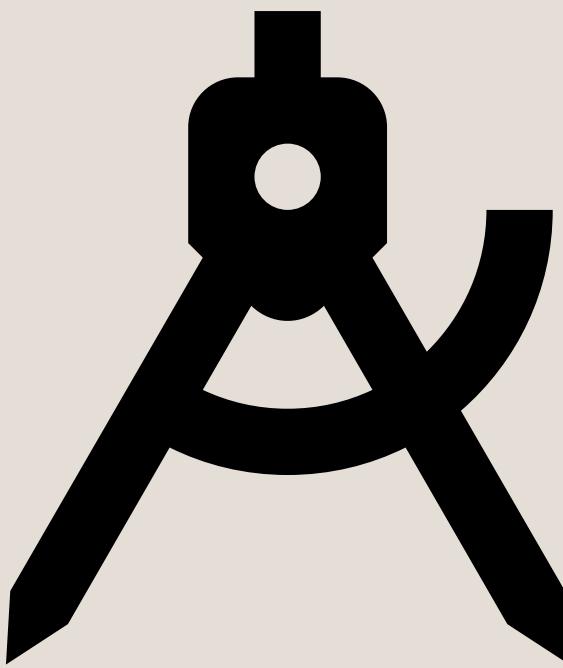
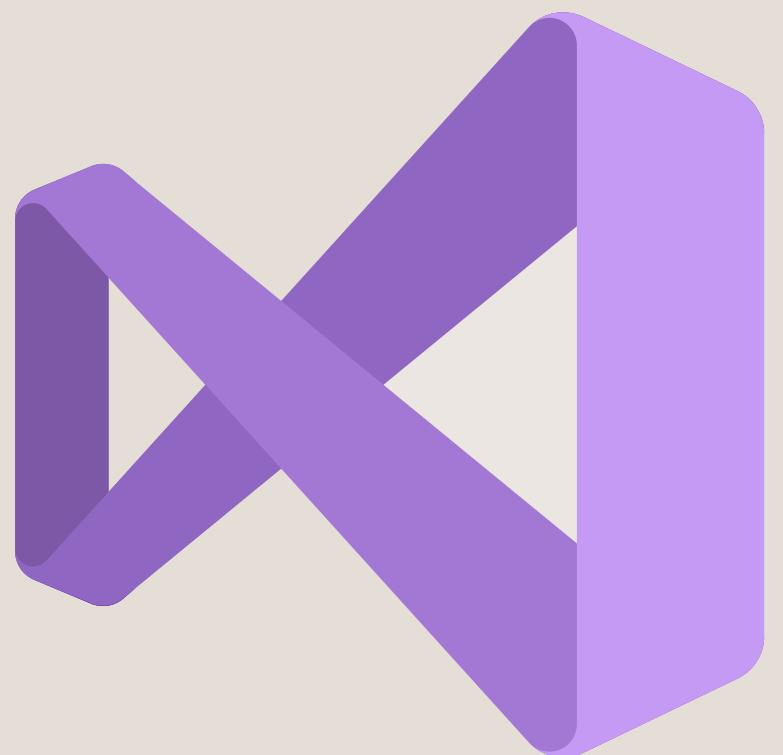
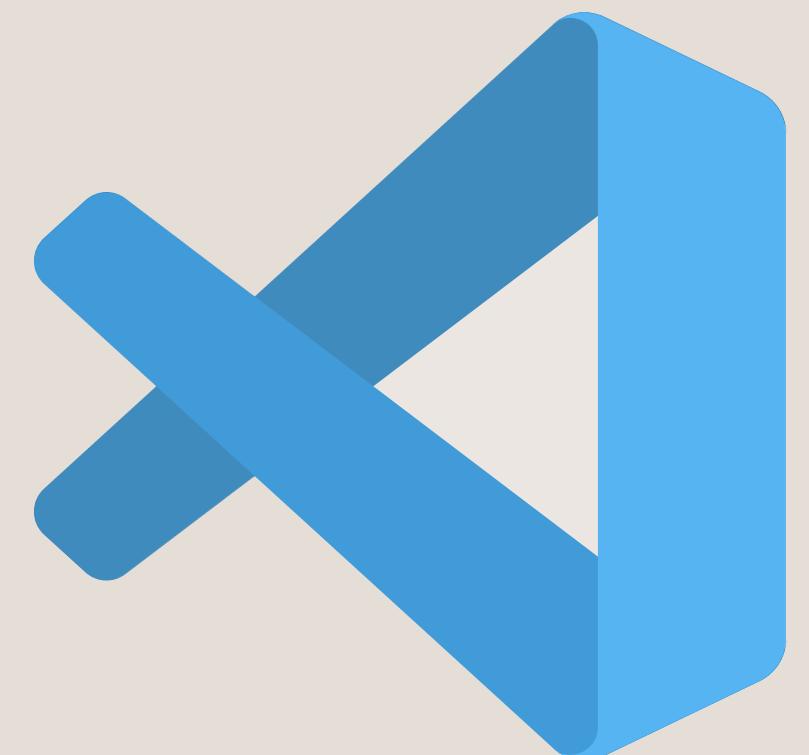
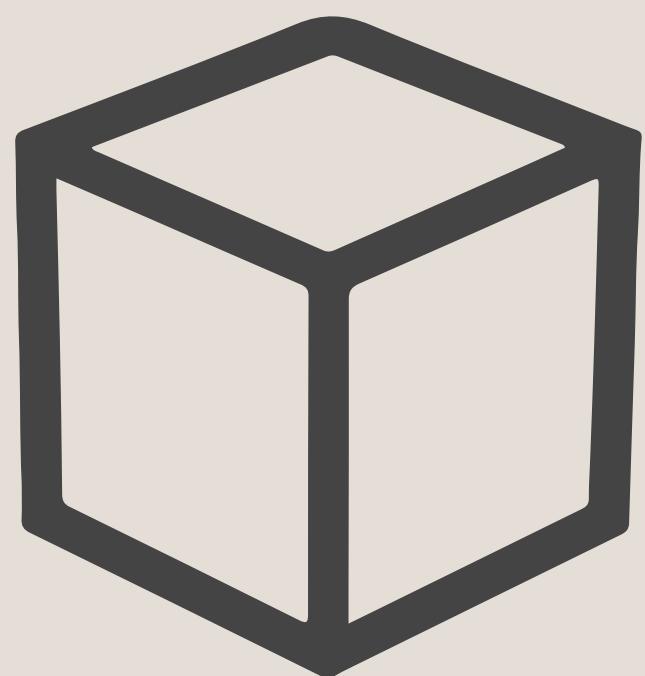
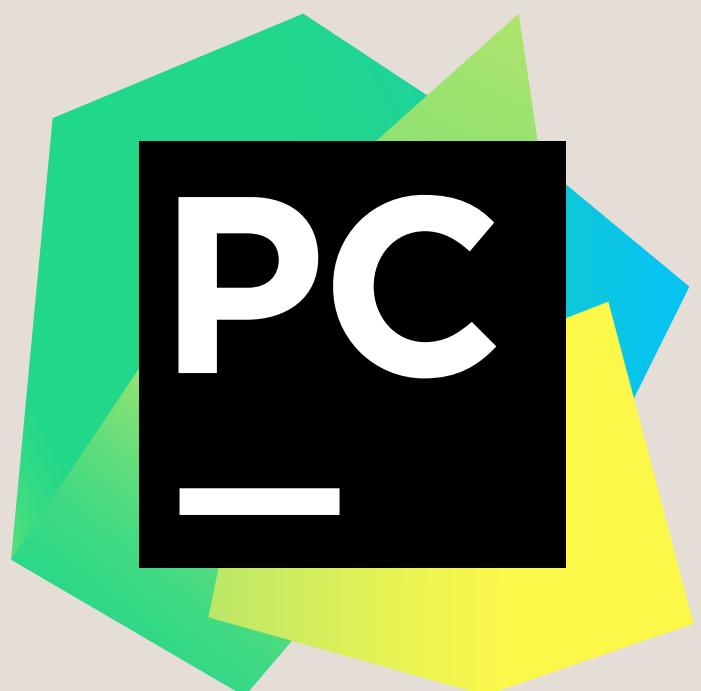
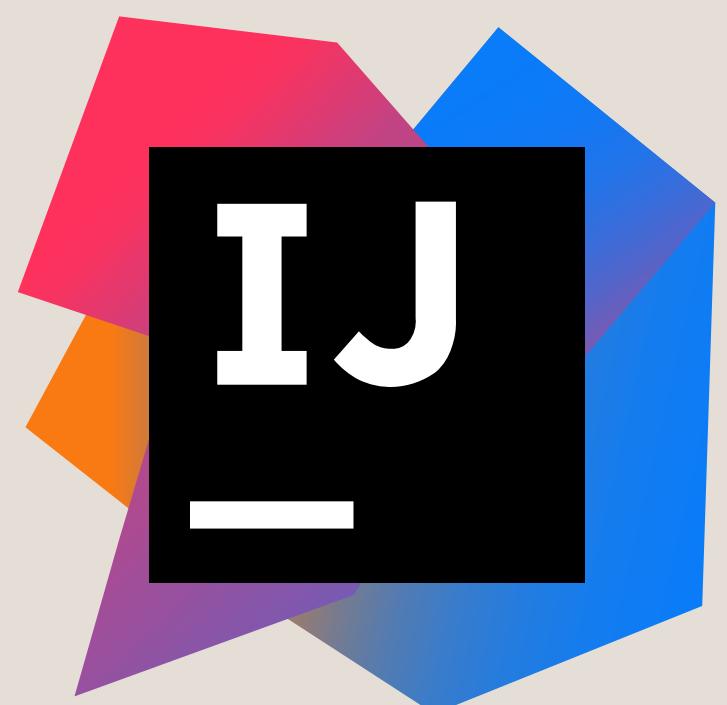


## IDE 개요

IDE selection



eclipse



## IDE 개요

Why do we use VSCode

- 다양한 언어 지원

- Python, Java, JavaScript, TypeScript, C#, C++, PHP 등과 같은 다양한 언어를 지원
- 이는 언어별로 특화된 확장 기능들을 통해 가능

- 확장성

- 가장 큰 강점 중 하나는 확장 기능들이 많다는 점
- 이들을 통해 원하는 기능을 추가하거나, 지원하는 언어를 확장 가능
- 테마, 코드 포매팅, 린터, 디버거, 버전 관리 도구 등 다양한 확장 기능이 존재

- 경량화와 성능

- 다른 IDE보다 가벼우며, 빠르게 실행됨
- 적은 시스템 리소스를 사용하면서도 강력한 기능을 제공

## IDE 개요

Why do we use VSCode

- **Git 통합**
  - Git과 같은 버전 관리 시스템과 잘 통합되어 있음
  - 이를 통해 변경 사항을 추적하고 커밋하며, 푸시 등의 작업을 IDE 내부에서 직접 수행할 수 있음

- **디버깅 도구**
  - 풍부한 디버깅 도구가 내장되어 있음
  - 코드를 실행하며 중단점을 설정하고 변수를 검사하며 스택을 탐색하는 등의 작업을 수행할 수 있음

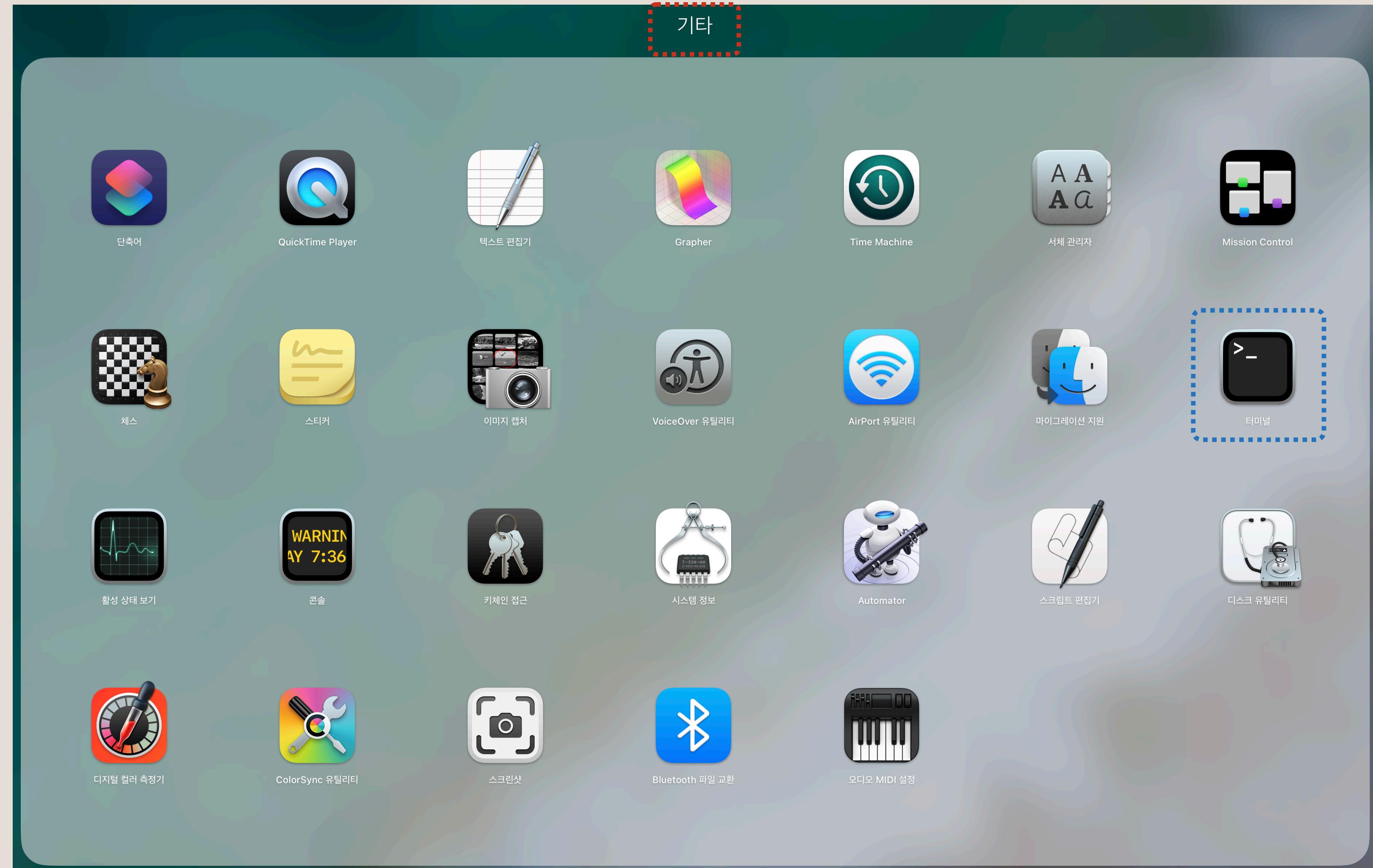
- **다중 플랫폼 지원**
  - Windows, macOS, Linux 등의 주요 운영체제에서 모두 실행 가능
  - 이는 다양한 개발 환경에서 일관된 경험을 제공한다는 의미

# VSCode 설치 on MacOS

## VSCode 설치 on MacOS

### Terminal 실행

- Launchpad → 기타 -> 터미널



## VSCode 설치 on MacOS

### VSCode 설치

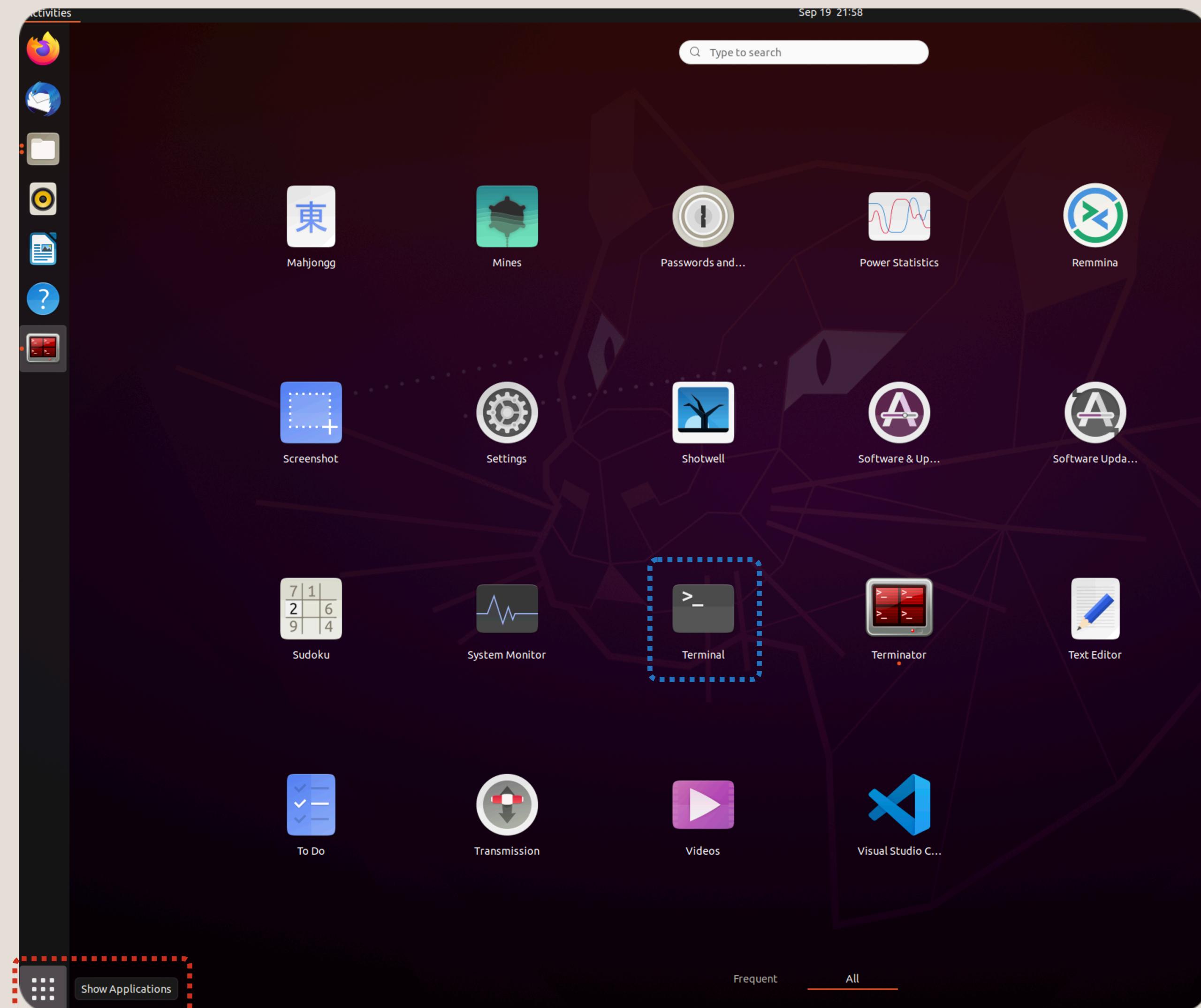
- 하기 커맨드를 순차적으로 입력▶
  - brew install --cask visual-studio-code

# VSCode 설치 on Ubuntu

## VSCode 설치 on Ubuntu

### Terminal 실행

- Show Application → Terminal



## VSCode 설치 on Ubuntu

### VSCode 설치

- 하기 커맨드를 순차적으로 입력◦

```
• wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add -◦  
• sudo add-apt-repository "deb [arch=$(dpkg --print-architecture)] https://packages.microsoft.com/repos/vscode stable main"◦  
• sudo apt install -y code◦
```

## VSCode 설치 on Windows

### Checking VSCode 설치

- 하기 커맨드를 순차적으로 입력◦
  - code --version◦
- 아래와 같은 결과가 나오면 설치 성공◦

```
parallels@ubuntu-linux-20-04-desktop:~$ code --version
1.82.2
abd2f3db4bdb28f9e95536dfa84d8479f1eb312d
arm64
```

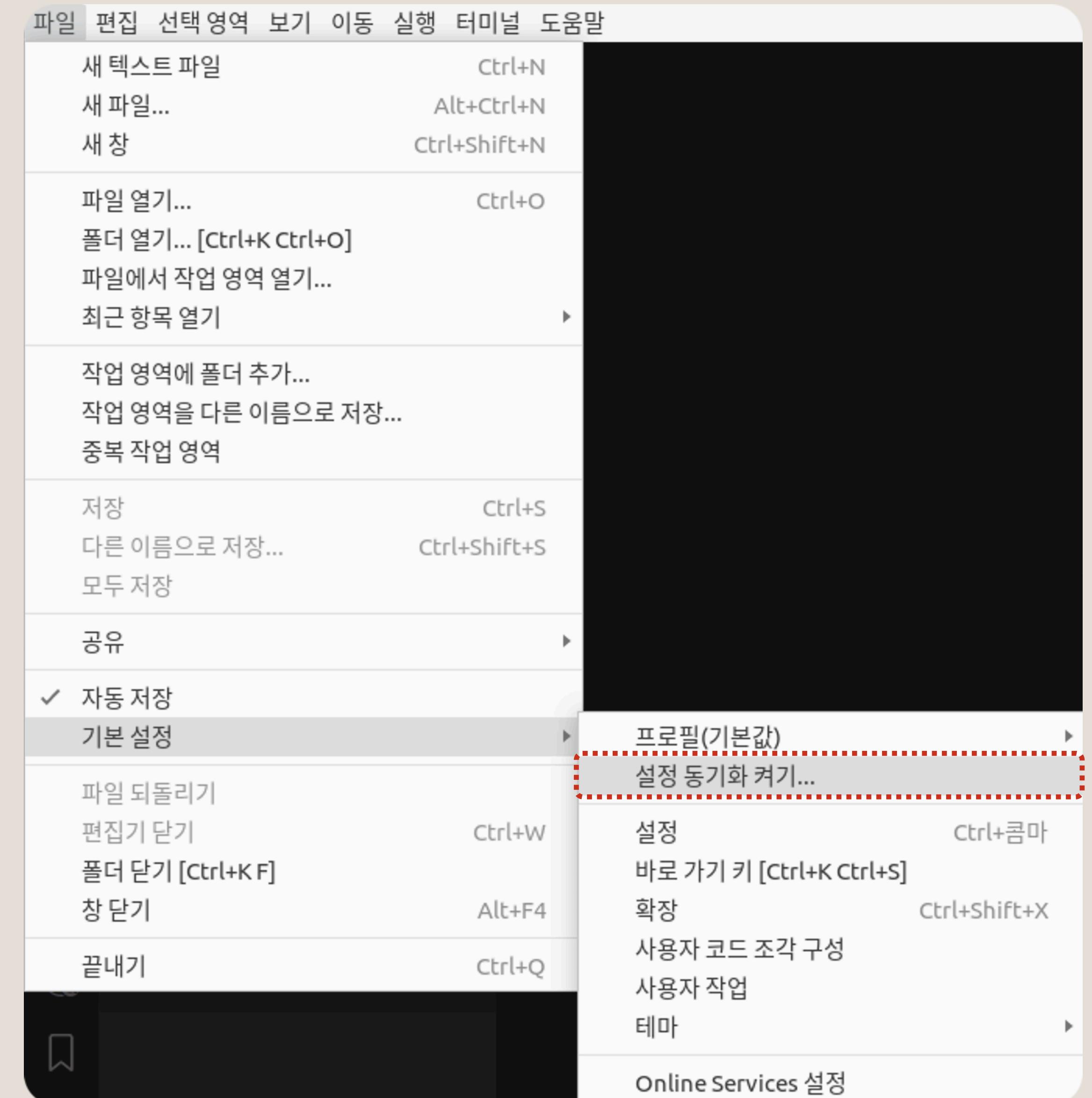
- 하기 커맨드를 입력하여 VSCode 실행◦
  - code

# VSCode 기본 설정

## VSCode 기본 설정

### 설정 동기화 설정

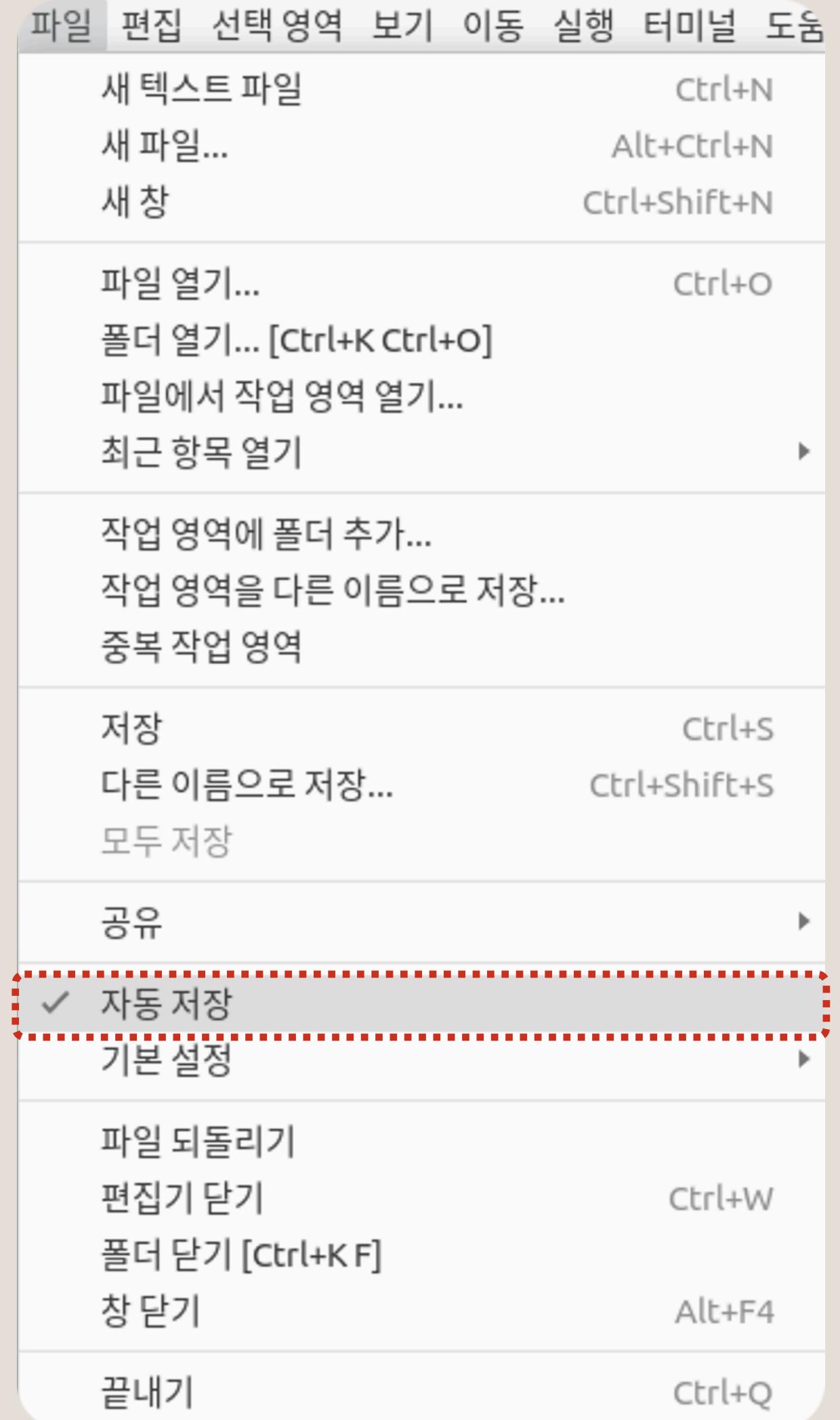
- 파일 -> 기본 설정 -> 설정 동기화 켜기...▶
  - Github 계정을 통해 VSCode의 설정을 저장 가능▶
  - PC, OS 관련 없이 동일한 설정으로 개발 가능▶



## VSCode 기본 설정

### 자동 저장 설정

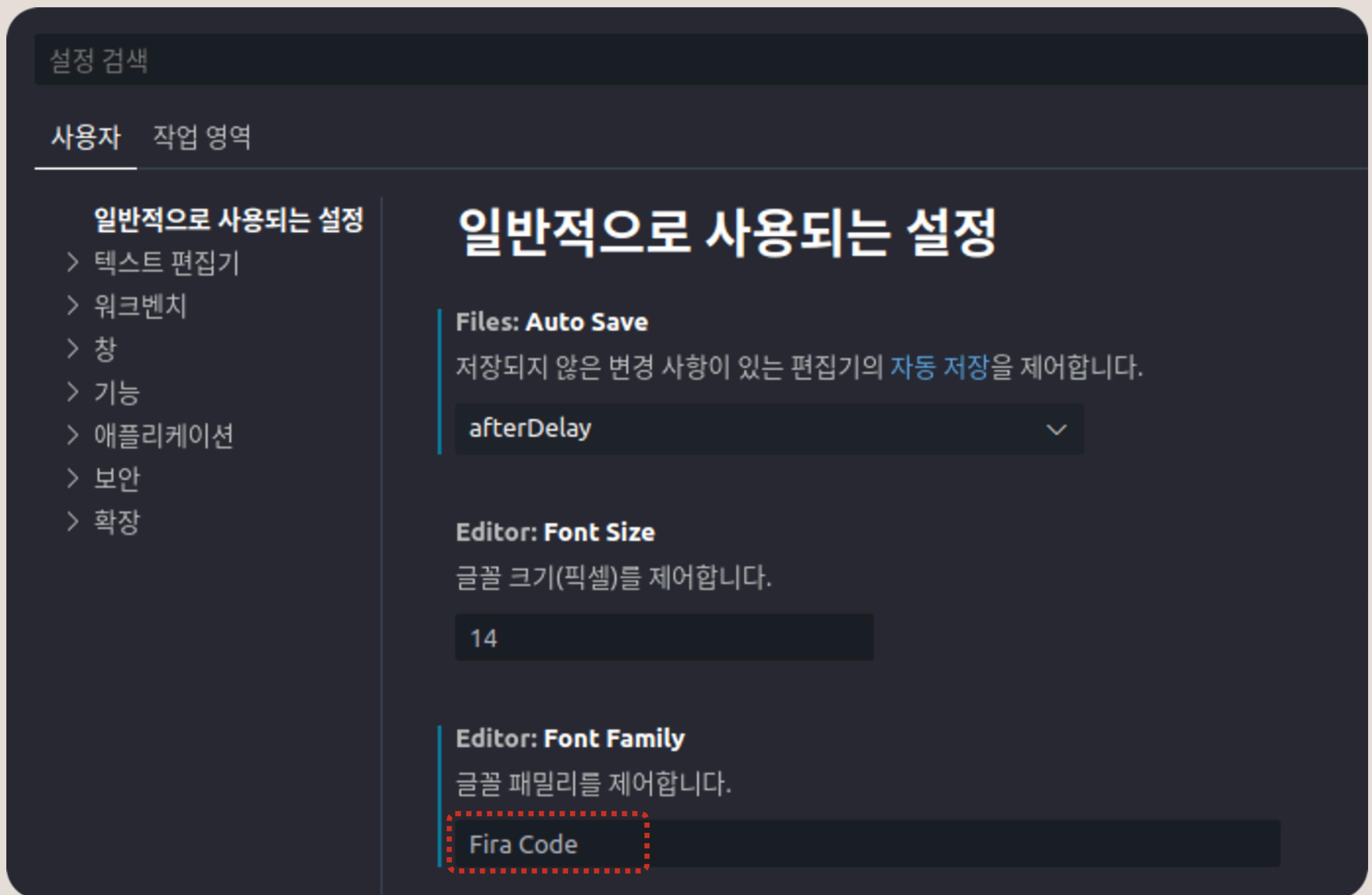
- 파일 -> 자동 저장
  - 수정시 자동 저장됨
  - 저장 문제 가능성을 사전에 차단



## VSCode 기본 설정

### Coding Font 적용

- Font Family → ‘설치한 font 이름’ 입력

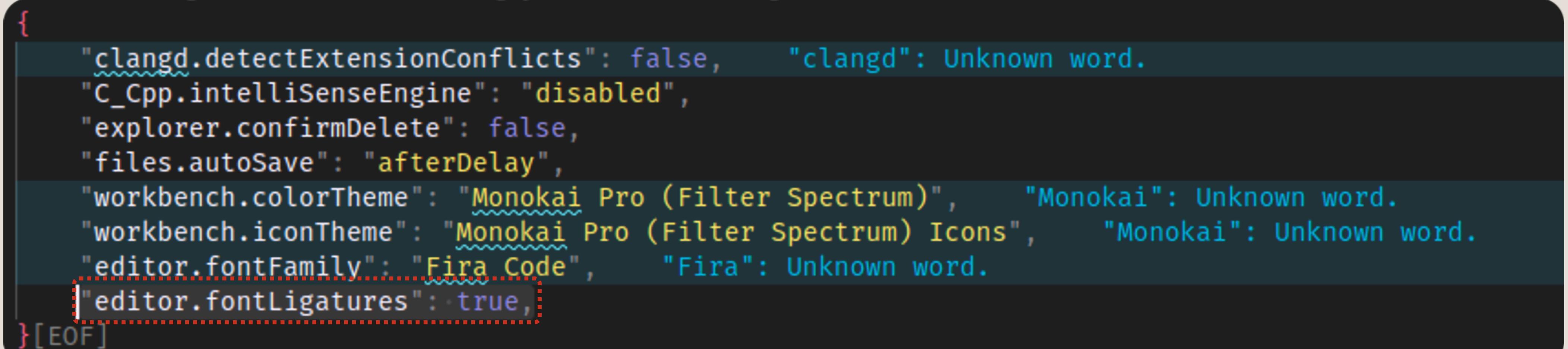


## VSCode 기본 설정

### Coding Font 적용

- control + shift + p → Preferences: Open User Settings (JSON)▫
- 하기 한줄 추가▫

- "editor.fontLigatures": true,

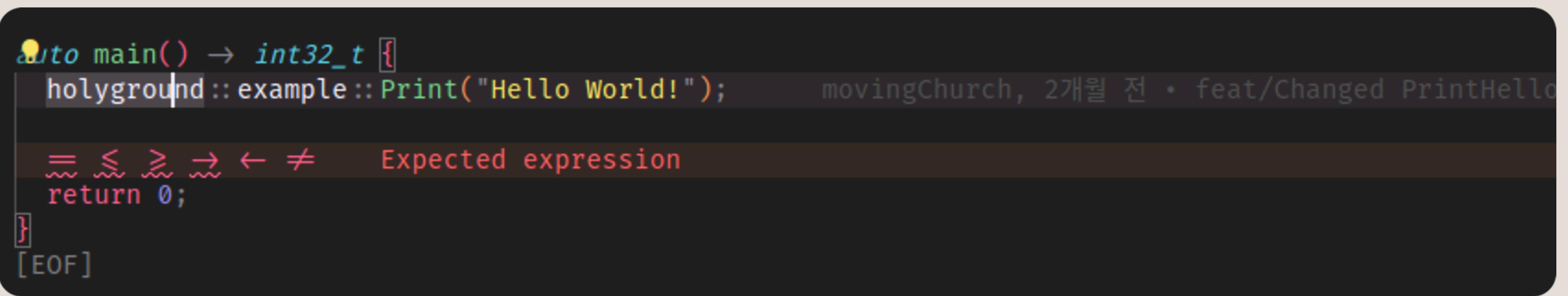


```
{  
    "clangd.detectExtensionConflicts": false,      "clangd": Unknown word.  
    "C_Cpp.intelliSenseEngine": "disabled",  
    "explorer.confirmDelete": false,  
    "files.autoSave": "afterDelay",  
    "workbench.colorTheme": "Monokai Pro (Filter Spectrum)",      "Monokai": Unknown word.  
    "workbench.iconTheme": "Monokai Pro (Filter Spectrum) Icons",      "Monokai": Unknown word.  
    "editor.fontFamily": "Fira Code",      "Fira": Unknown word.  
    "editor.fontLigatures": true,  
}[EOF]
```

A screenshot of the VSCode settings.json file. The file contains JSON configuration options. A red dashed box highlights the line "editor.fontLigatures": true, which is being added to the configuration. The rest of the file includes settings for clangd, C\_Cpp, explorer, files, workbench, and editor font family.

## VSCode 기본 설정

Coding Font 적용



A screenshot of the Visual Studio Code (VSCode) interface. The code editor shows a C++ file with the following content:

```
auto main() -> int32_t {
    holyground :: example :: Print("Hello World!");
    return 0;
}
[EOF]
```

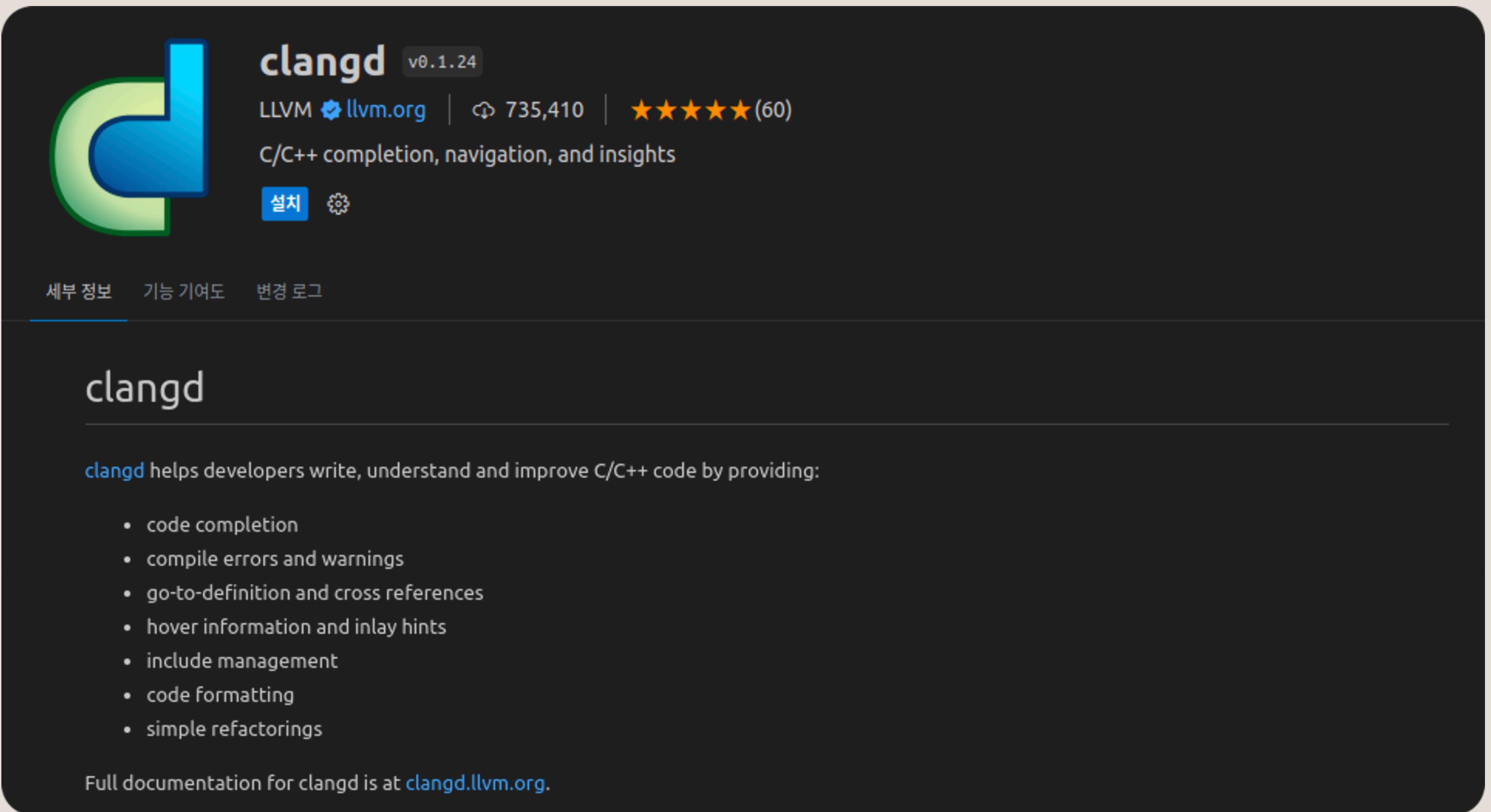
The word "holyground" is highlighted in blue, indicating it is a user-defined variable or function name. A yellow circular icon with a question mark is positioned next to the opening brace of the main function. A red error message "Expected expression" is displayed above the closing brace of the main function. The status bar at the bottom of the screen shows the file path "movingChurch, 2개월 전 · feat/Changed PrintHello" and the commit hash "feat/Changed PrintHello".

# C++ 개발을 위한 VSCode 확장기능 설정

## C++ 개발을 위한 VSCode 확장기능 설정

CMake를 사용한 C++ 개발환경 설정

- clangd
  - CMake 추가 설정 없이 바로 사용 가능
  - 기본 C/C++ 확장보다 더욱 다양한 기능 제공



The screenshot shows the clangd extension page on the Visual Studio Marketplace. The page features a large logo with a stylized 'cd' monogram. To the right of the logo, the text 'clangd v0.1.24' is displayed, along with the LLVM logo, a download count of 735,410, and a five-star rating of 60. Below this, a subtitle reads 'C/C++ completion, navigation, and insights'. Two buttons are present: a blue '설치' (Install) button and a gear-shaped settings button. At the bottom of the page, there are three tabs: '세부 정보' (Detailed Information), '기능 기여도' (Contribution), and '변경 로그' (Change Log). A large heading 'clangd' is centered below the logo. The main content area describes what clangd does: 'clangd helps developers write, understand and improve C/C++ code by providing:' followed by a bulleted list of features: code completion, compile errors and warnings, go-to-definition and cross references, hover information and inlay hints, include management, code formatting, and simple refactorings. A note at the bottom states: 'Full documentation for clangd is at [clangd.llvm.org](https://clangd.llvm.org)'.

## VSCode 권장 확장 기능

### Extensions - for C++

- C/C++
  - 기본 C++을 위한 확장 기능
  - clangd와 동시에 사용

The screenshot shows the Microsoft C/C++ extension page on the Visual Studio Code marketplace. The extension icon features a purple and white design with the text 'C/C++'. The title 'C/C++ v1.16.3' is displayed, along with the developer name 'Microsoft' and the URL 'microsoft.com'. It shows 50,448,848 installs and a rating of 4.5 stars from 522 reviews. Below the title, it says 'C/C++ IntelliSense, debugging, and code browsing.' There are buttons for '사용 안 함' (Not used), '제거' (Remove), and '시험판 버전으로 전환' (Switch to preview version). A note at the bottom states '이 확장은 전역적으로 사용하도록 설정되었습니다.' (This extension is set to be used globally). At the bottom of the page, there is a section titled 'C/C++ for Visual Studio Code' with links to 'Repository', 'Issues', 'Documentation', and 'Code Samples'. It also indicates that 'Live Share' is enabled. A descriptive text explains that the extension adds language support for C/C++ to VS Code, including editing and debugging features. A 'Pre-requisites' section notes that C++ is a compiled language and requires compilation before execution, stating that the extension does not include a compiler or debugger.

C/C++ v1.16.3

Microsoft [microsoft.com](#) | 50,448,848 | ★★★★☆ (522)

C/C++ IntelliSense, debugging, and code browsing.

사용 안 함 | 제거 | 시험판 버전으로 전환

이 확장은 전역적으로 사용하도록 설정되었습니다.

세부 정보 기능 기여도 변경 로그 런타임 상태

## C/C++ for Visual Studio Code

[Repository](#) | [Issues](#) | [Documentation](#) | [Code Samples](#)

Live Share enabled

The C/C++ extension adds language support for C/C++ to Visual Studio Code, including [editing \(IntelliSense\)](#) and [debugging](#) features.

### Pre-requisites

C++ is a compiled language meaning your program's source code must be translated (compiled) before it can be run on your computer. VS Code is first and foremost an editor, and relies on command-line tools to do much of the development workflow. The C/C++ extension **does not include a C++ compiler or debugger**. You will need to install these tools or use those already installed on your computer.

## C++ 개발을 위한 VSCode 확장기능 설정

CMake를 사용한 C++ 개발환경 설정

- clangd - 설정▣
  - control + shift + p → Preferences: Open User Settings (JSON)▣
  - 하기 두줄 추가▣

```
"clangd.detectExtensionConflicts": false,  
"C_Cpp.intelliSenseEngine": "Disabled",
```

## C++ 개발을 위한 VSCode 확장기능 설정

### Extensions - for C++

- C++ Algorithm Mnemonics
  - STL 알고리즘 제안 기능

The screenshot shows the extension page for 'C++ Algorithm Mnemonics' on the Visual Studio Marketplace. The extension is version v1.0.3, developed by David Brötje, with 108,042 installs and a 5-star rating. The description states: 'C++ Algorithm Mnemonics as VSCode snippets.' Below the description are buttons for '사용 안 함' (Not Used) and '제거' (Remove). A note in Korean says: '이 확장은 전역적으로 사용하도록 설정되었습니다.' (This extension is set to be used globally.) At the bottom, there are tabs for '세부 정보', '기능 기여도', '변경 로그', and '런타임 상태'.

### C++ Algorithm Mnemonics for Visual Studio Code

This extension is a port of [Algorithm Mnemonics: Increase Productivity with STL Algorithms](#) by [Tommy Bennett](#).

#### Features

[Algorithm Mnemonics: Increase Productivity with STL Algorithms](#) provides code templates for C++ STL algorithms. Each code template is given a name and a 3-letter mnemonic (All available code templates are listed in the file [STLInstructionSet.md](#)).

This extension implements these code templates as VS Code snippets accessible by its 3-letter mnemonic.

#### Usage

## C++ 개발을 위한 VSCode 확장기능 설정

### Extensions - for C++

- C++ TestMate
  - C++ Test를 위한 확장

The screenshot shows the Microsoft Store page for the "C++ TestMate" extension. The extension is version v4.5.1, developed by Mate Pek, with 207,708 installs and a rating of ★★★★★ (16). The description states: "Run GoogleTest, Catch2 and DOCTest tests from VSCode". A note below says: "이 확장은 전역적으로 사용하도록 설정되었습니다." (This extension is set to be used globally). Below the store page, there is a detailed description of the extension's features and a list of its latest changes.

**C++ TestMate** v4.5.1

Mate Pek | 207,708 | ★★★★★ (16)

Run GoogleTest, Catch2 and DOCTest tests from VSCode

사용 안 함 | 제거 | ⚙️

이 확장은 전역적으로 사용하도록 설정되었습니다.

세부 정보 기능 기여도 변경 로그 런타임 상태

## C++ TestMate

A **Catch2**, **GoogleTest**, **doctest** and **GoogleBenchmark** Explorer for VSCode

version v4.5.1 issues 5 open downloads 1.2M

This extension allows you to run your **Catch2**, **Google Test** and **DOCTest** tests using the native testing `vscode-api`. It also have basic support for **Google Benchmark**.

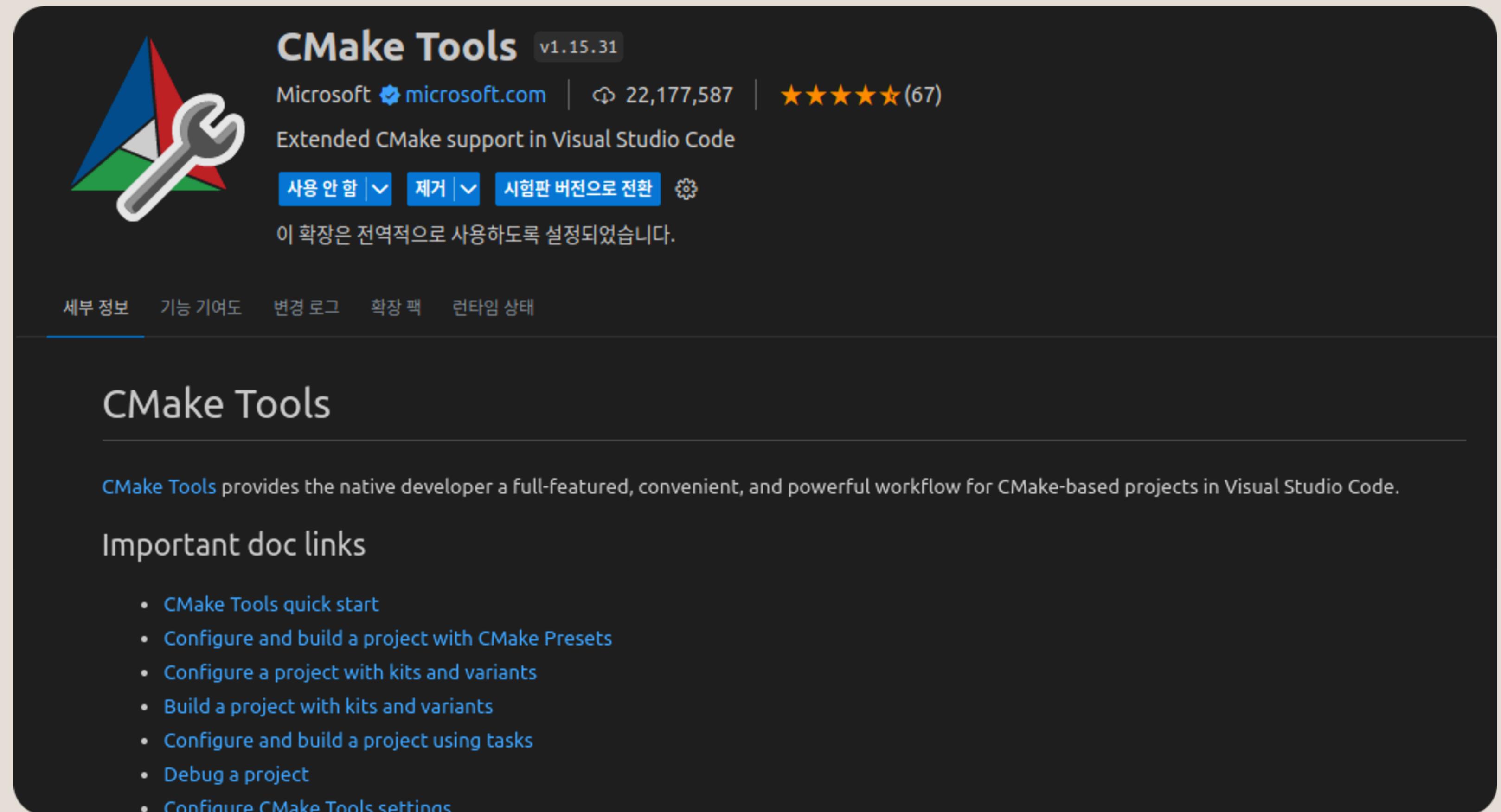
### Features / Show-Off

- New testing API integration has just happened with a tons of improvements.
  - Streaming the test run: Don't have to wait for the result to see the progress (in case you test uses `std::cout`)

## C++ 개발을 위한 VSCode 확장기능 설정

### Extensions - for CMake

- CMake Tools



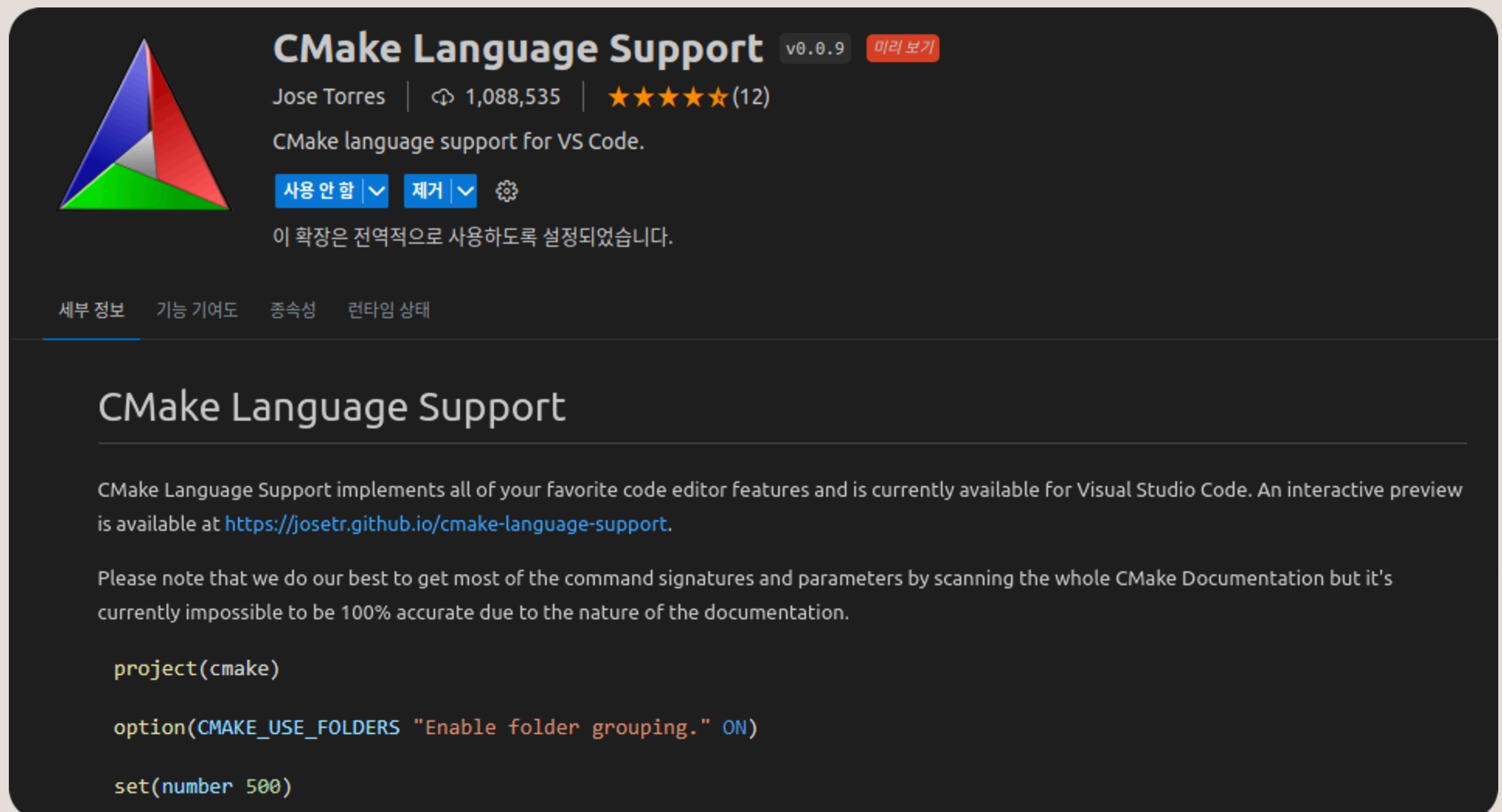
The screenshot shows the Microsoft Store page for the "CMake Tools" extension. The extension is version v1.15.31, developed by Microsoft, with over 22,177,587 installations and a rating of 5 stars from 67 reviews. The description reads "Extended CMake support in Visual Studio Code". Below the description, there are buttons for "사용 안 함" (Not used), "제거" (Remove), "시험판 버전으로 전환" (Switch to beta version), and a gear icon. A message at the bottom states "이 확장은 전역적으로 사용하도록 설정되었습니다." (This extension is set to be used globally). At the bottom of the page, there are tabs for "세부 정보", "기능 기여도", "변경 로그", "확장 팩", and "런타임 상태". The main content area features the title "CMake Tools" and a brief description: "CMake Tools provides the native developer a full-featured, convenient, and powerful workflow for CMake-based projects in Visual Studio Code." Below this, there is a section titled "Important doc links" with a list of links:

- [CMake Tools quick start](#)
- [Configure and build a project with CMake Presets](#)
- [Configure a project with kits and variants](#)
- [Build a project with kits and variants](#)
- [Configure and build a project using tasks](#)
- [Debug a project](#)
- [Configure CMake Tools settings](#)

# C++ 개발을 위한 VSCode 확장기능 설정

## Extensions - for CMake

- CMake Language Support



The screenshot shows the Visual Studio Code Marketplace page for the "CMake Language Support" extension. The extension is version 0.0.9, developed by Jose Torres, with 1,088,535 installs and a 5-star rating from 12 reviews. It is described as providing CMake language support for VS Code. A note at the bottom indicates it is a preview release. The page includes tabs for "세부 정보", "기능 기여도", "종속성", and "런타임 상태". Below the main header, there is a large heading "CMake Language Support" followed by a description of its features and a link to an interactive preview. A code snippet example is shown at the bottom.

**CMake Language Support** v0.0.9 [미리 보기](#)

Jose Torres | 1,088,535 | ★★★★★(12)

CMake language support for VS Code.

사용 안 함 | 제거 | ⚙️

이 확장은 전역적으로 사용하도록 설정되었습니다.

세부 정보 기능 기여도 종속성 런타임 상태

## CMake Language Support

CMake Language Support implements all of your favorite code editor features and is currently available for Visual Studio Code. An interactive preview is available at <https://josetr.github.io/cmake-language-support>.

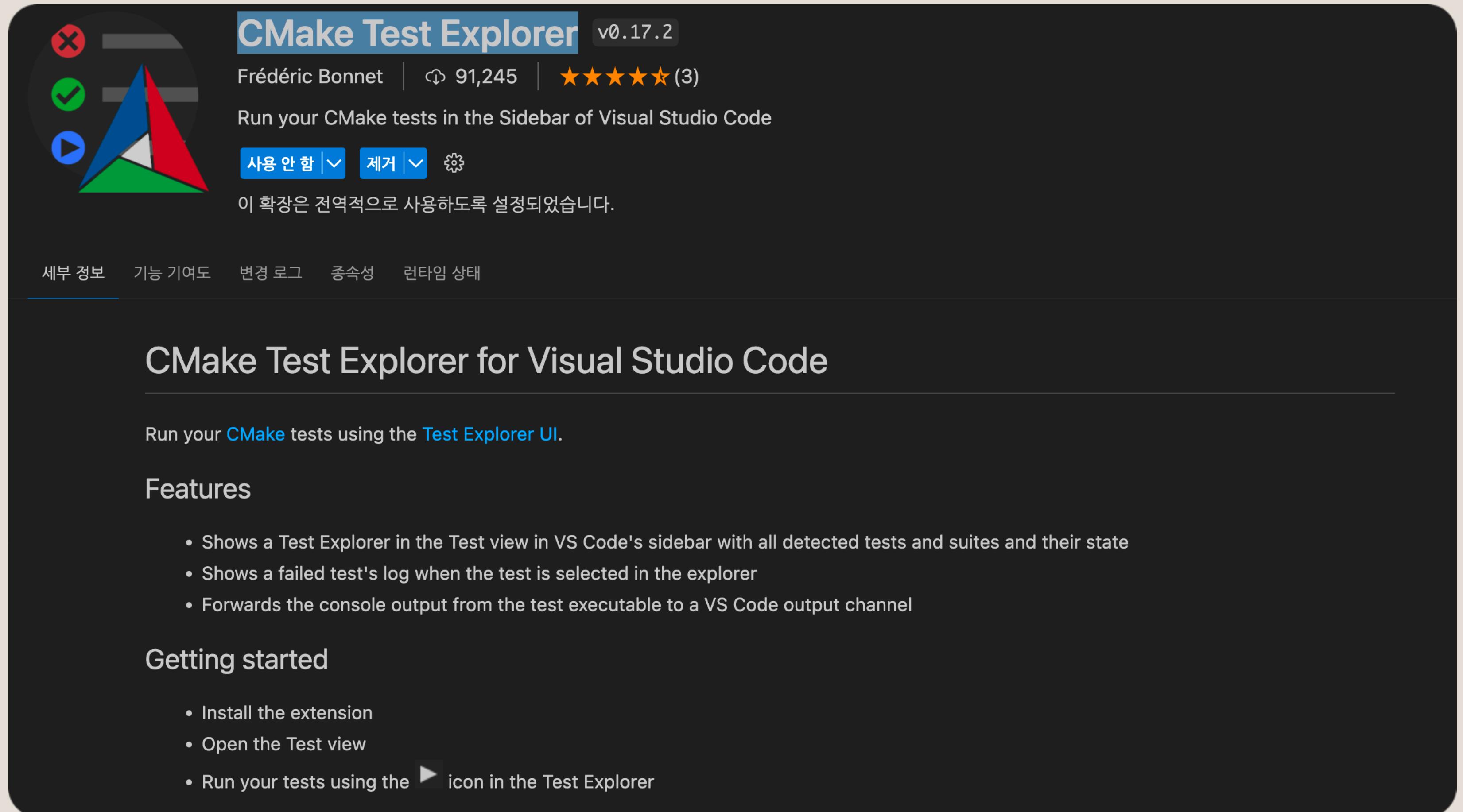
Please note that we do our best to get most of the command signatures and parameters by scanning the whole CMake Documentation but it's currently impossible to be 100% accurate due to the nature of the documentation.

```
project(cmake)
option(CMAKE_USE_FOLDERS "Enable folder grouping." ON)
set(number 500)
```

# C++ 개발을 위한 VSCode 확장기능 설정

## Extensions - for CMake

- CMake Test Explorer



The screenshot shows the extension page for 'CMake Test Explorer' on the Visual Studio Marketplace. The title 'CMake Test Explorer' is at the top, followed by the version 'v0.17.2'. Below it are the developer's name 'Frédéric Bonnet', the download count '91,245', and a 5-star rating '(3)'. A subtitle reads 'Run your CMake tests in the Sidebar of Visual Studio Code'. There are buttons for '사용 안 함' (Not used), '제거' (Remove), and a gear icon. A note in Korean says '이 확장은 전역적으로 사용하도록 설정되었습니다.' (This extension is set to be used globally). At the bottom, there are tabs for '세부 정보', '기능 기여도', '변경 로그', '종속성', and '런타임 상태'. The main content area is titled 'CMake Test Explorer for Visual Studio Code' and describes running CMake tests using the Test Explorer UI. It lists features such as showing a Test Explorer in the sidebar, displaying failed test logs, and forwarding console output. It also provides a 'Getting started' section with steps: install the extension, open the Test view, and run tests using the play icon in the Test Explorer.

**CMake Test Explorer** v0.17.2

Frédéric Bonnet | 91,245 | ★★★★★ (3)

Run your CMake tests in the Sidebar of Visual Studio Code

사용 안 함 | 제거 | ☰

이 확장은 전역적으로 사용하도록 설정되었습니다.

세부 정보 기능 기여도 변경 로그 종속성 런타임 상태

## CMake Test Explorer for Visual Studio Code

Run your [CMake](#) tests using the [Test Explorer UI](#).

### Features

- Shows a Test Explorer in the Test view in VS Code's sidebar with all detected tests and suites and their state
- Shows a failed test's log when the test is selected in the explorer
- Forwards the console output from the test executable to a VS Code output channel

### Getting started

- Install the extension
- Open the Test view
- Run your tests using the ► icon in the Test Explorer

# VSCode 권장 확장 기능

## VSCode 권장 확장 기능

Extensions - for Github

- GitHub Pull Requests and Issues
- gitignore
- GitLens – Git supercharged

## VSCode 권장 확장 기능

### Extensions - for Readability

- Code Spell Checker
- Color Highlight
- colorize
- EOF Mark
- Rainbow CSV
- Trailing Spaces

## VSCode 권장 확장 기능

### Extensions - for Comments

- Doxygen Documentation Generator [🔗](#)
- Better Comments [🔗](#)
- Todo Tree [🔗](#)

## VSCode 권장 확장 기능

Extensions - for Performance

- IntelliCode
- Code Runner
- Error Lens

## VSCode 권장 확장 기능

### Extensions - for Etc

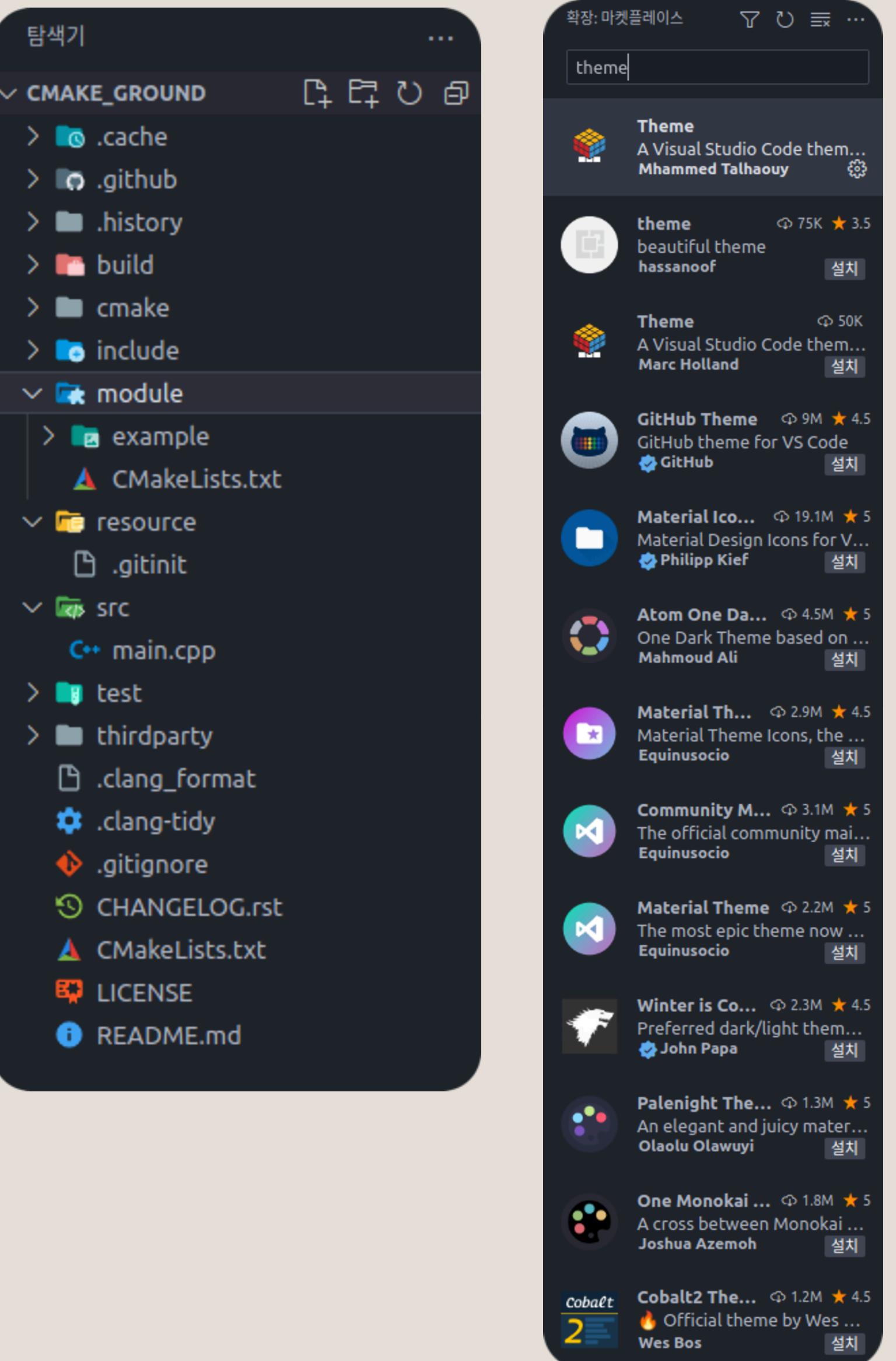
- Bookmarks○
- file-size○
- Polacode○
- project-tree○
- vscode-pdf○
- XML Tools○
- YAML○
- markdownlint○

# VSCode 테마 설정

## VSCode 테마 설정

### Extensions - for Theme

- Icon theme와 color theme가 있음
- 원하는 theme를 찾아서 적용해보도록



# 예제 코드 빌드 using VSCode

## 예제 코드 빌드 using VSCode

### 예제 프로젝트 작성

- 하기와 같은 폴더 구조로 예제 프로젝트 작성(폴더, 파일)◦

```
└── cmake_example_project◦  
    ├── CMakeLists.txt◦  
    └── build◦  
        └── main.cpp
```

## 예제 코드 빌드 using VSCode

예제 프로젝트 작성 - CMakeLists.txt

```
cmake_minimum_required(VERSION 3.11)

set(PACKAGE_NAME CMAKE_TEST_PROJECT)
set(PACKAGE_VERSION 1.1.0)

project(${PACKAGE_NAME} VERSION ${PACKAGE_VERSION} LANGUAGES CXX)

set(CMAKE_CXX_STANDARD 20)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
set(CMAKE_CXX_EXTENSIONS OFF)

set(CPACK_PROJECT_NAME ${PROJECT_NAME})
set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})
include(CPack)

message(STATUS "PACKAGE_NAME: ${PACKAGE_NAME}")
message(STATUS "PACKAGE_VERSION: ${PACKAGE_VERSION}")

add_executable(${PACKAGE_NAME}_MAIN main.cpp)
target_compile_options(${PACKAGE_NAME}_MAIN PRIVATE
    -Wall -Wextra -Wpedantic -Werror
)
```

## 예제 코드 빌드 using VSCode

예제 프로젝트 작성 - main.cpp

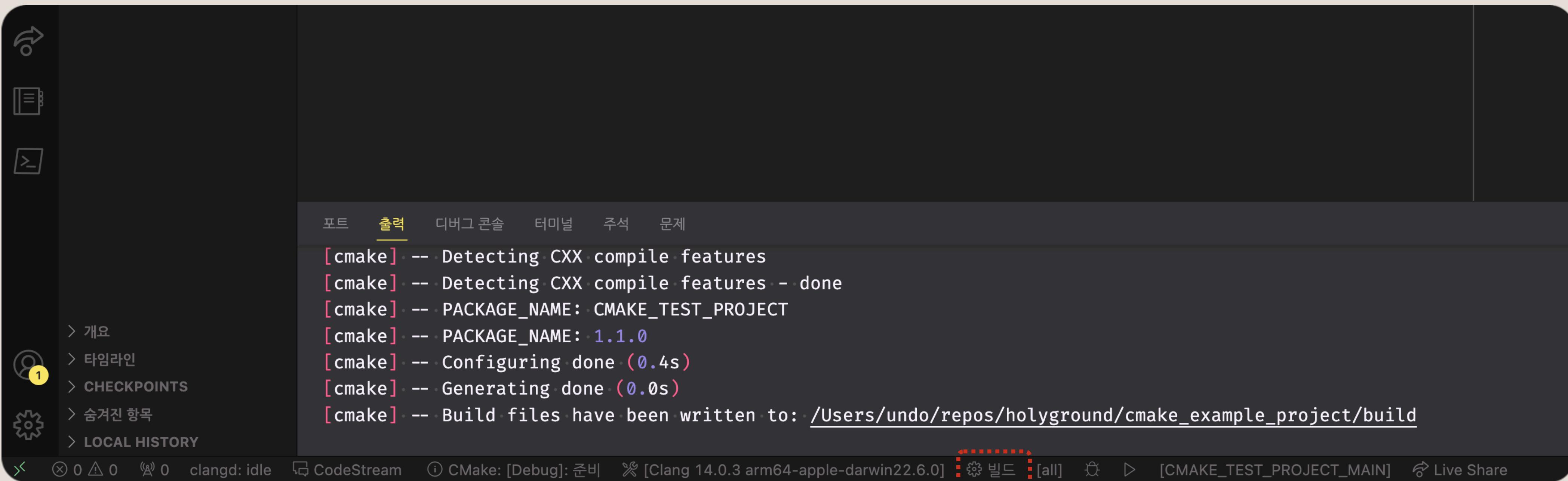
```
#include <cstdint>
#include <iostream>

auto main() → int32_t {
    std::cout << "Hello World!" << std::endl;
    return 0;
}
```

## 예제 코드 빌드 using VSCode

### 예제 프로젝트 빌드

- 좌측 하단 창의 Build(빌드) 클릭



The screenshot shows the VSCode interface with the Output panel open. The panel title is 'Output' and it displays the following CMake build logs:

```
[cmake] -- Detecting CXX compile features
[cmake] -- Detecting CXX compile features - done
[cmake] -- PACKAGE_NAME: CMAKE_TEST_PROJECT
[cmake] -- PACKAGE_NAME: 1.1.0
[cmake] -- Configuring done (0.4s)
[cmake] -- Generating done (0.0s)
[cmake] -- Build files have been written to: /Users/undo/repos/holyground/cmake_example_project/build
```

The bottom status bar shows the build configuration: 'CMake: [Debug]: 준비' (CMake: [Debug]: ready). A red dashed box highlights the 'Build' icon in the status bar.

## 예제 코드 빌드 using VSCode

### 예제 프로젝트 빌드

- 좌측 하단 창의 실행 버튼 클릭
- 실행 결과

```
/Users/undo/repos/holyground/cmake_example_project/build/CMAKE_TEST_PROJECT_MAIN
~/repos/holyground/cmake_example_project/build
/Users/undo/repos/holyground/cmake_example_project/build/CMAKE_TEST_PROJECT_MAIN
Hello World!
~/repos/holyground/cmake_example_project/build
```

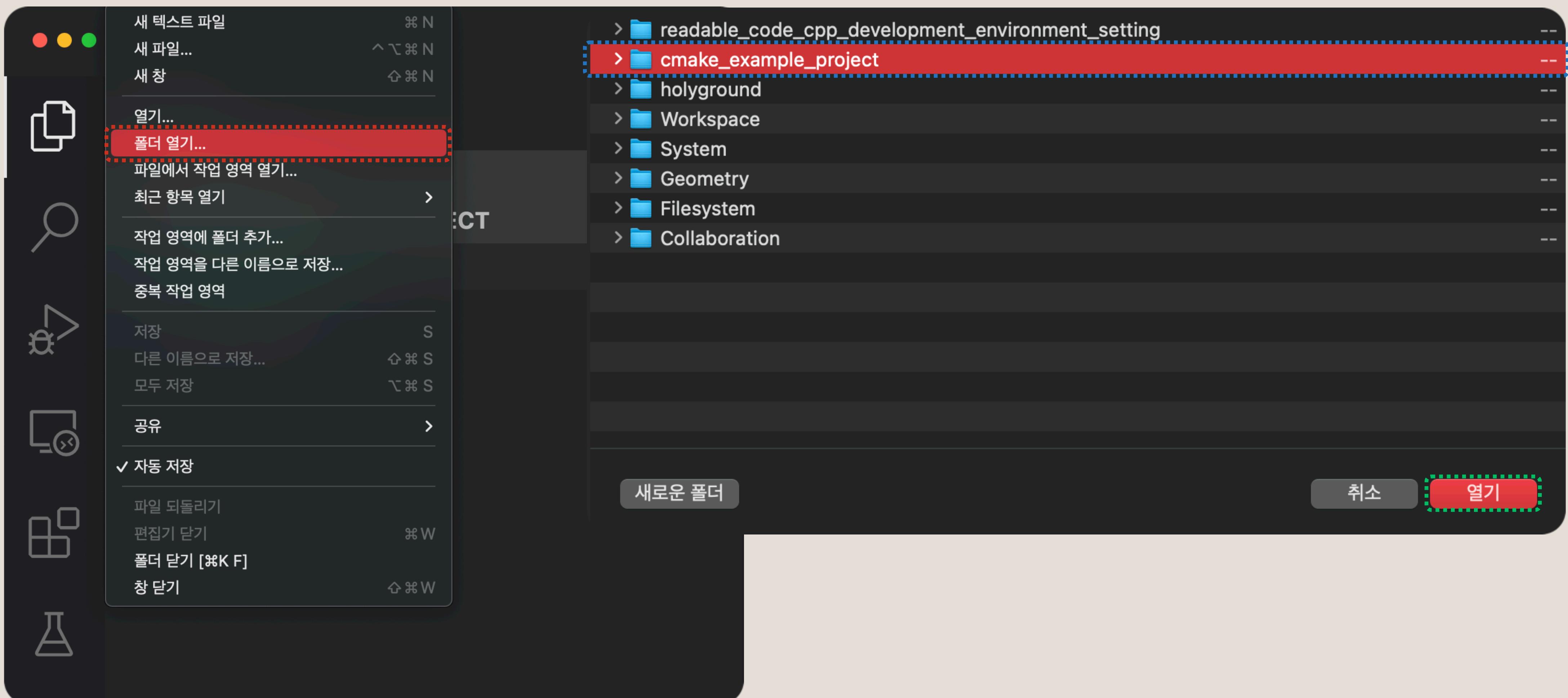
The screenshot shows the VSCode interface with the terminal tab selected. The terminal window displays the command 'cmake --build . --target CMAKE\_TEST\_PROJECT\_MAIN' followed by the output 'Hello World!'. The output text is highlighted with a blue dashed rectangle. The terminal also shows the status bar at the bottom with various icons and text.

# VSCode로 Git 사용하기

## 예제 코드 빌드 using VSCode

VSCode로 Git 사용하기 - 프로젝트 설정

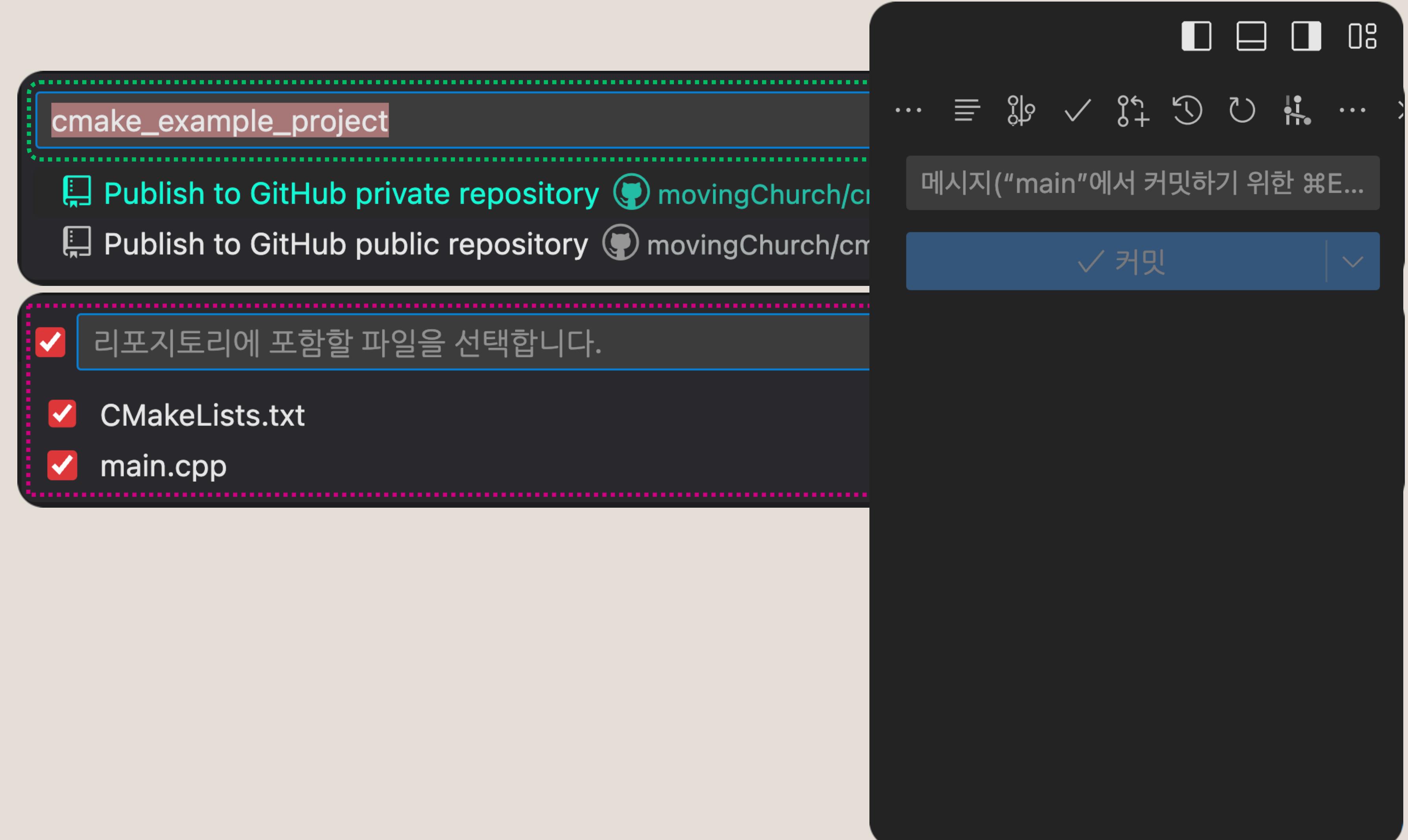
- 기존에 생성한 cmake\_test\_example을 기준으로 작업▶
- 파일 -> 폴더 열기...▶
- cmake\_example\_project → 열기



# 예제 코드 빌드 using VSCode

# VSCode로 Git 사용하기 - Update a new repo

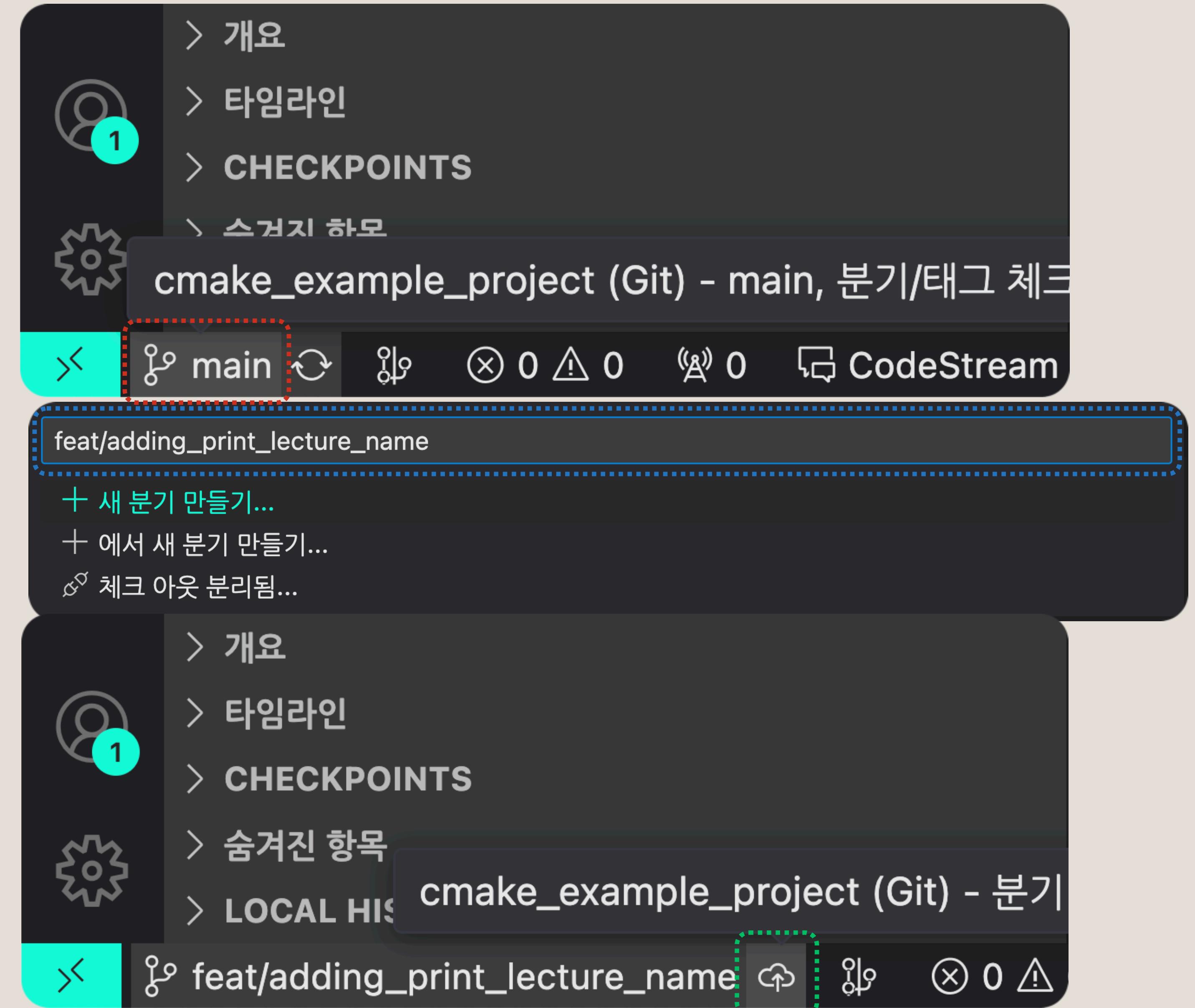
- 좌측 상단의 좌측 탭 활성화
  - GitHub에 게시 클릭
  - Repo 이름 기술 후 Enter
  - 추가할 파일 선택
  - OK 클릭
  - Repo 활성화 완료



## 예제 코드 빌드 using VSCode

VSCode로 Git 사용하기 - Create a new branch

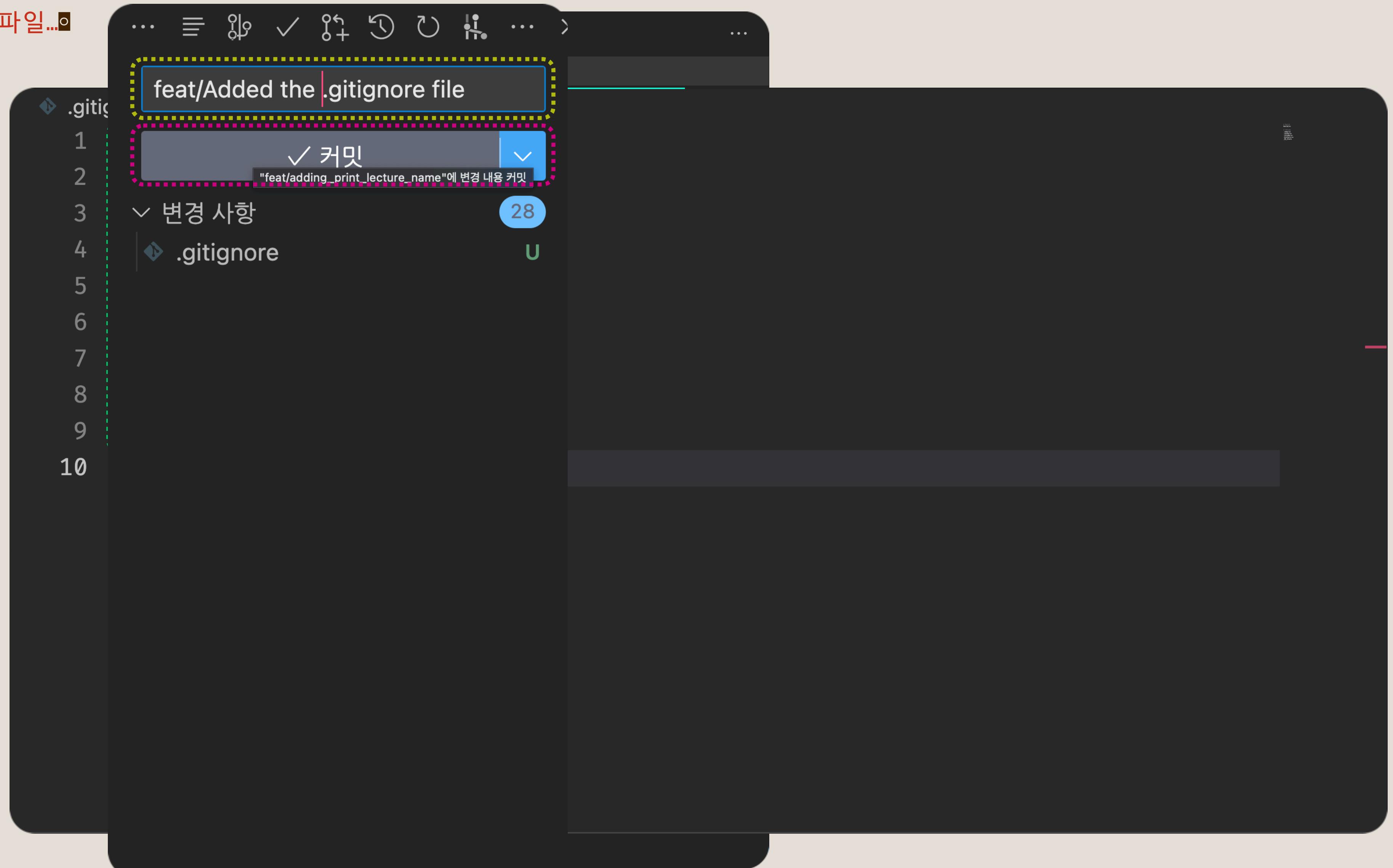
- 작업할 새로운 branch 생성
- Branch 이름 기술 후 enter
- Branch를 remote로 update



## 예제 코드 빌드 using VSCode

VSCode로 Git 사용하기 - Commit/Add the gitignore file

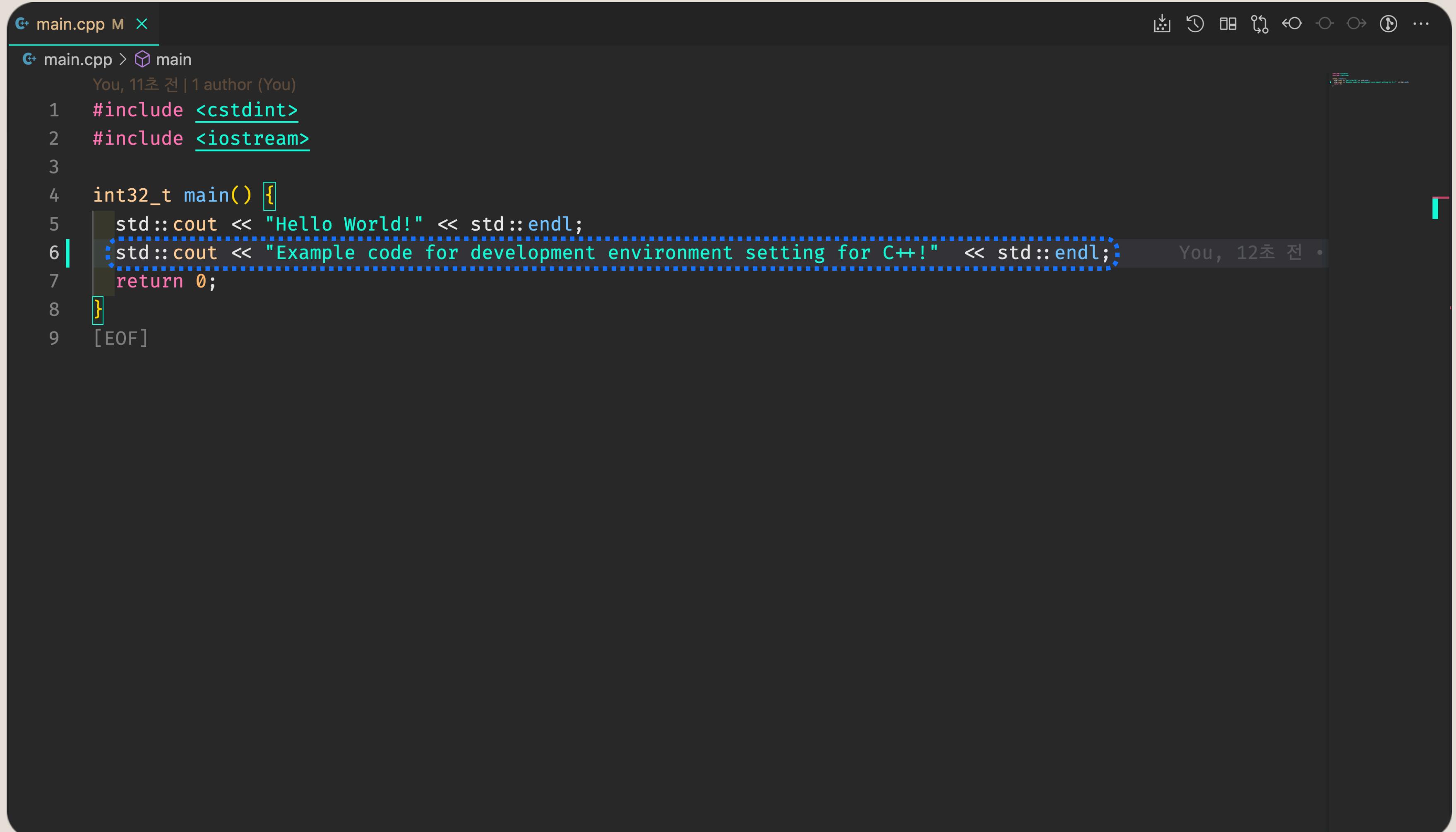
- 탐색기 창에서 좌 클릭 -> 새 파일...
- .gitignore file 추가
- file 내용 작성
- Commit message 입력
- 커밋 버튼 클릭



## 예제 코드 빌드 using VSCode

VSCode로 Git 사용하기 - main.cpp 소스 코드 수정

- 탐색기 창에서 main.cpp 실행
- 추가 코드 작성



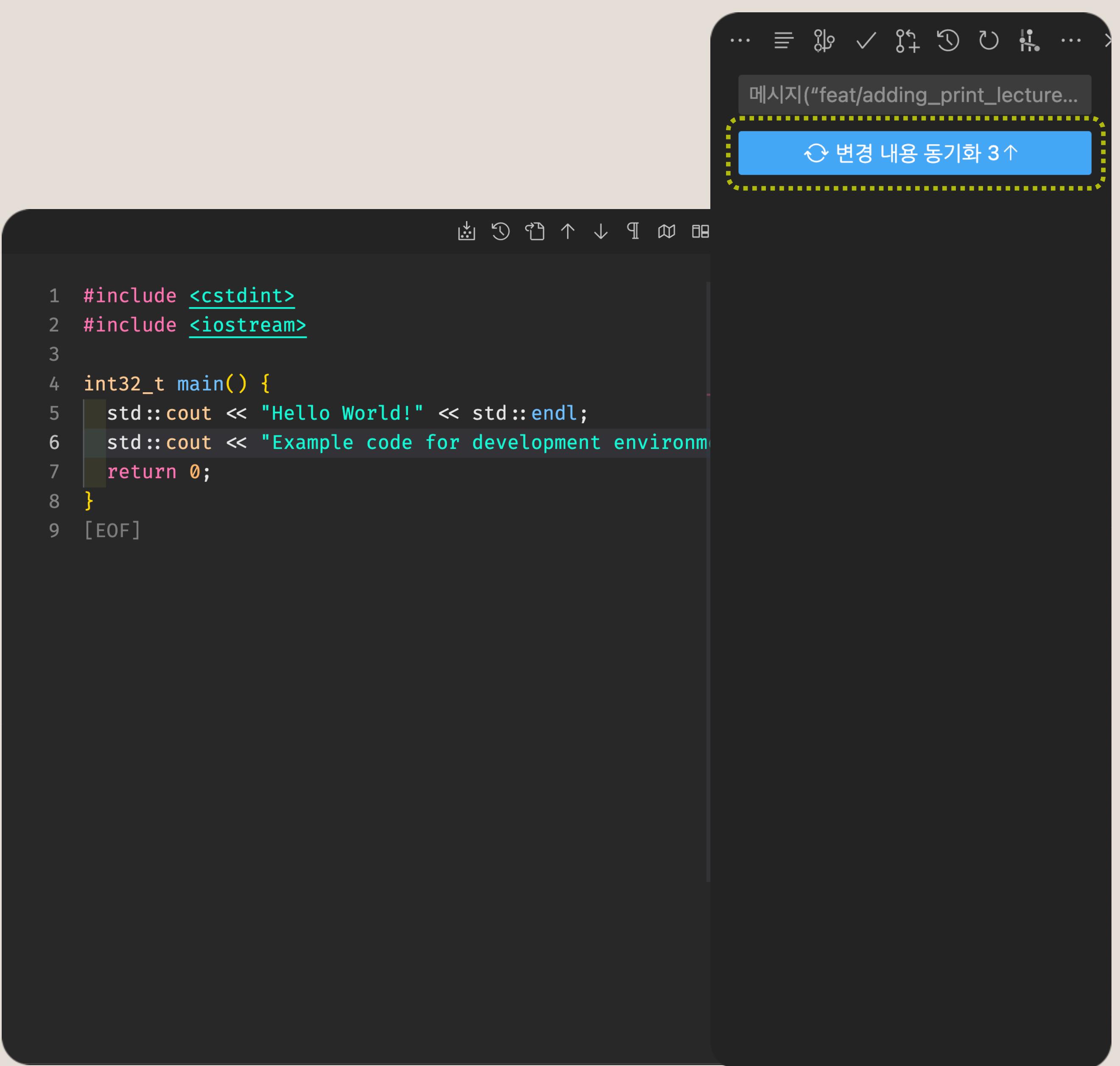
The screenshot shows the VSCode interface with a dark theme. A code editor window is open for 'main.cpp'. The code contains a 'main' function that prints 'Hello World!' and an additional line of text. The last line, 'std::cout << "Example code for development environment setting for C++!" << std::endl;', is highlighted with a blue selection bar. The status bar at the bottom right indicates the file was modified 12 seconds ago.

```
main.cpp M X
main.cpp > main
You, 11초 전 | 1 author (You)
1 #include <cstdint>
2 #include <iostream>
3
4 int32_t main() {
5     std::cout << "Hello World!" << std::endl;
6     std::cout << "Example code for development environment setting for C++!" << std::endl;
7     return 0;
8 }
9 [EOF]
```

## 예제 코드 빌드 using VSCode

### VSCode로 Git 사용하기 - Commit& Update

- main.cpp 변경 내용 스테이징에 추가
- Commit message 작성
- 커밋 버튼 클릭
- 변경 내용 동기화 버튼 클릭



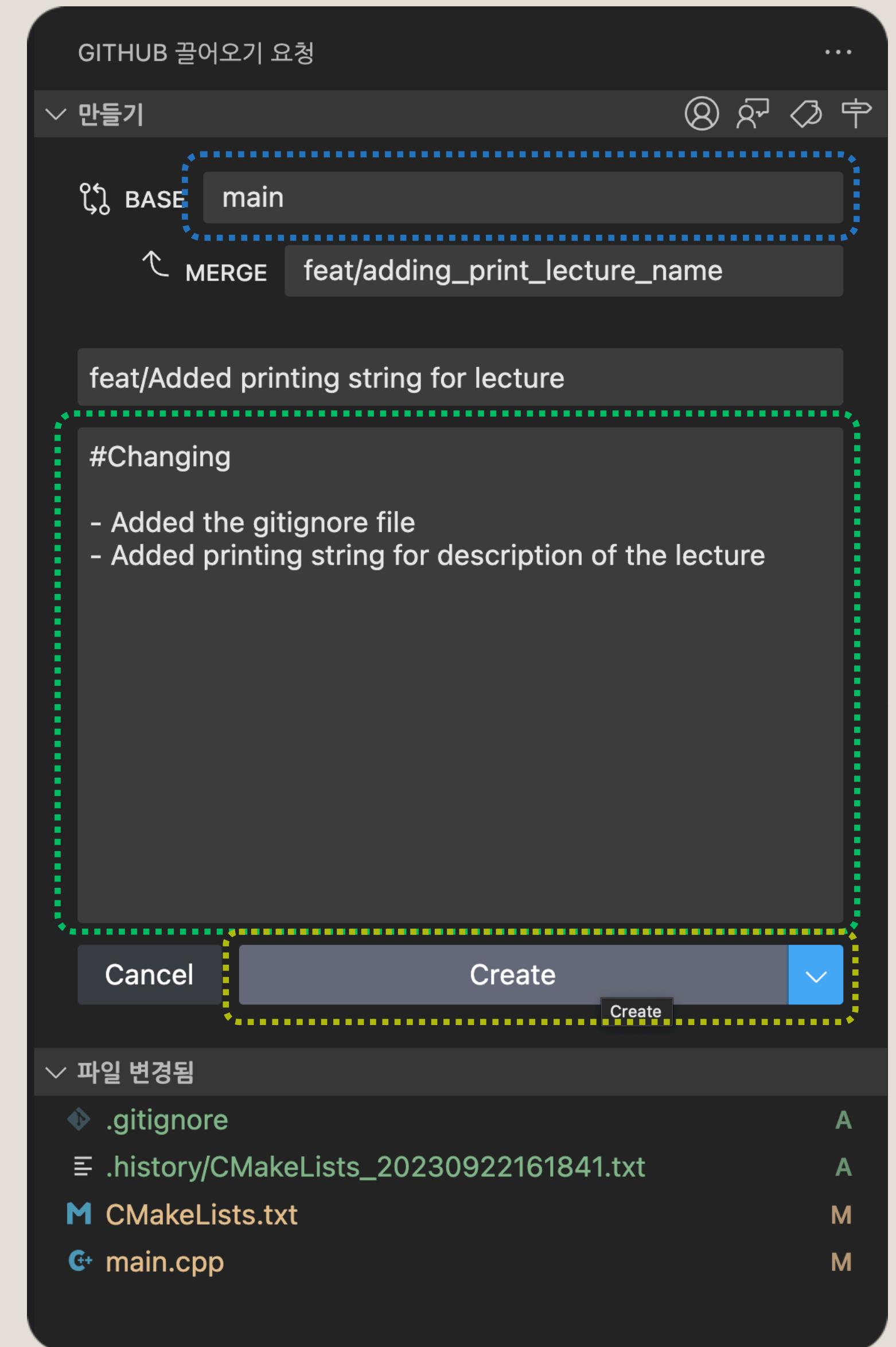
The screenshot shows the VSCode interface with a dark theme. On the right, the Git commit panel is open, displaying a commit message: "feat/adding\_print\_lecture...". A blue button labeled "변경 내용 동기화 3↑" is highlighted with a yellow dashed border. In the center, a code editor window shows the following C++ code:

```
1 #include <cstdint>
2 #include <iostream>
3
4 int32_t main() {
5     std::cout << "Hello World!" << std::endl;
6     std::cout << "Example code for development environment";
7     return 0;
8 }
9 [EOF]
```

## 예제 코드 빌드 using VSCode

VSCode로 Git 사용하기 - Create pull requests

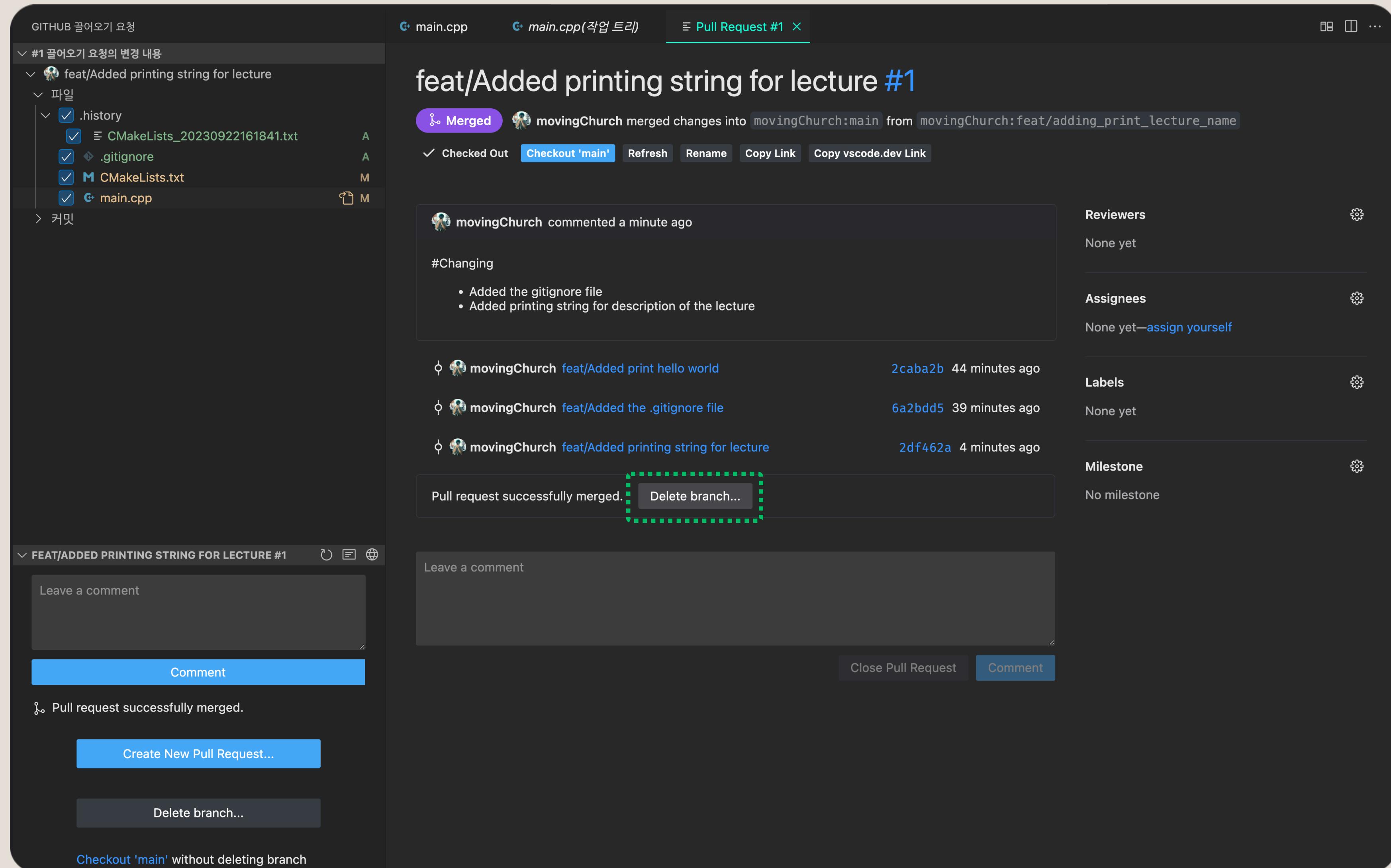
- 끌어오기 요청 만들기 클릭
- Base branch 지정
- PR 내용 입력
- Create 버튼 클릭



## 예제 코드 빌드 using VSCode

VSCode로 Git 사용하기 - Merge pull requests

- Review 진행
- Review 완료 후 Create Merge Commit
- Delete branch...를 통해 기존 branch 삭제



# 개발환경 설정 자동화

자동화 스크립트 개요

개발환경 설정 스트립트 for MacOS

개발환경 설정 스트립트 for Ubuntu

# 자동화 스크립트 개요

## 자동화 스크립트 개요

What is script?

- 일련의 컴퓨터 명령을 포함한 작은 프로그램 또는 명령어 집합을 나타냄▪
- 스크립트는 텍스트 파일로 작성되며, 특정 작업을 자동화하거나 반복적인 작업을 간소화하기 위해 사용▪

## 자동화 스크립트 개요

### Types of scripts

- Bash 스크립트
  - Unix 및 Linux 환경에서 사용
  - 터미널 명령을 스크립트로 작성하기 위해 사용
- PowerShell 스크립트
  - Windows 환경에서 사용
  - Windows 운영 체제와 관련된 작업을 자동화하는 데 사용
- Python 스크립트
  - 범용 스크립트 언어
  - 다양한 운영 체제 및 작업에 사용 가능
- JavaScript 스크립트
  - 웹 개발과 관련된 작업에 사용
  - 브라우저 환경 및 Node.js에서 실행 가능

## 자동화 스크립트 개요

Why do we use the script for development environment setting

- 사람이 반복 작업을 할 필요가 없음■
- 일관적인 개발환경을 구축할 필요가 있음■
- 잘 모듈화 되어 만들어진 스크립트는 재활용 가능■

# 개발환경 설정 스트립트 for MacOS

## 개발환경 설정 스트립트 for MacOS

Components of 개발환경 설정 스트립트

- 하기와 같은 폴더 구조로 예제 프로젝트 작성(폴더, 파일)◦

```
└─ script_for_macos◦  
    ├─ install_clang.sh◦  
    ├─ install_cmake.sh◦  
    ├─ install_gcc.sh◦  
    ├─ install_git.sh◦  
    ├─ install_homebrew.sh◦  
    ├─ install_vscode.sh◦  
    ├─ setting_compilers.sh◦  
    └─ setting_development_environment.sh◦
```

## 개발환경 설정 스트립트 for MacOS

Components of 개발환경 설정 스트립트

- setting\_development\_environment.sh를 실행 시 개발환경이 셋업되는 시스템◦
- 각각 설치 스크립트를 분리하여 모듈화/재활용성을 확보◦

## 개발환경 설정 스트립트 for MacOS

Components of 개발환경 설정 스트립트 - install\_clang.sh

```
#!/bin/bash

# Check if Homebrew is installed and install if not
./install_homebrew.sh

# Install Clang and Clangd
echo "Installing Clang and Clangd..."
brew install llvm

# Verify installed versions
echo "Clang version:"
clang --version
echo "Clangd version:"
clangd --version
```

## 개발환경 설정 스트립트 for MacOS

Components of 개발환경 설정 스트립트 - install\_cmake.sh

```
#!/bin/bash

# Check if Homebrew is installed and install
if not
./install_homebrew.sh

# Install CMake
echo "Installing CMake ... "
brew install cmake

# Verify installed versions
cmake --version
```

## 개발환경 설정 스트립트 for MacOS

Components of 개발환경 설정 스트립트 - install\_gcc.sh

```
#!/bin/bash

# Check if Homebrew is installed and install
if not
./install_homebrew.sh

# Install GCC
echo "Installing GCC..."
brew install gcc

# Verify installed versions
echo "GCC version:"
gcc --version
```

## 개발환경 설정 스트립트 for MacOS

Components of 개발환경 설정 스트립트 - install\_git.sh

```
#!/bin/bash

# Check if Homebrew is installed and install if not
./install_homebrew.sh

# Install Git
echo "Installing Git..."
brew install git

# Install Git CLI
echo "Installing Git CLI..."
brew install gh

# Verify installed versions
echo "Git version:"
git --version
echo "Git CLI version:"
gh --version

# Log in to GitHub
gh auth login
```

## 개발환경 설정 스트립트 for MacOS

Components of 개발환경 설정 스트립트 - install\_homebrew.sh

```
#!/bin/bash

# Check if Homebrew is installed and install if not
if ! command -v brew &> /dev/null; then
    echo "Installing Homebrew ... "
    /bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
    (echo; echo 'eval "$(/opt/homebrew/bin/brew shellenv)"') >> /Users/$(id -un)/.zprofile
    eval "$(/opt/homebrew/bin/brew shellenv)"
fi
```

## 개발환경 설정 스트립트 for MacOS

Components of 개발환경 설정 스트립트 - install\_vscode.sh

```
#!/bin/bash

# Check if Homebrew is installed and install if not
./install_homebrew.sh

# Install VSCode
brew install visual-studio-code --cask

# Verify installed versions
echo "VSCode version:"
code --version
```

## 개발환경 설정 스트립트 for MacOS

Components of 개발환경 설정 스트립트 - setting\_compilers.sh

```
#!/bin/bash

./install_gcc.sh
./install_clang.sh
```

## 개발환경 설정 스트립트 for MacOS

Components of 개발환경 설정 스트립트 - setting\_development\_environment.sh

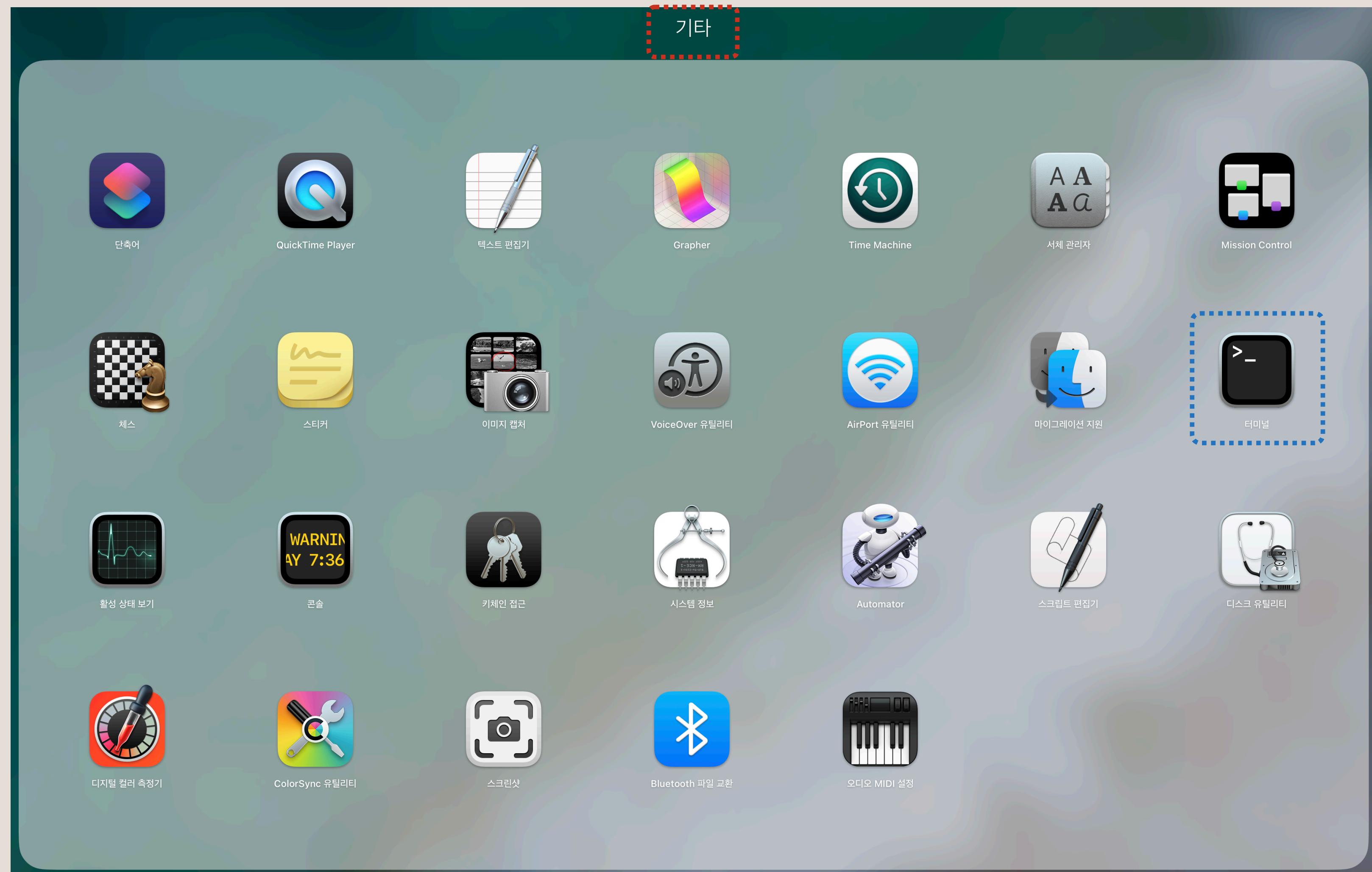
```
#!/bin/bash

./setting_compilers.sh
./install_cmake.sh
./install_git.sh
./install_vscode.sh
```

# 개발환경 설정 스트립트 for MacOS

Setting development environment using scripts

- Launchpad → 기타 -> 터미널



## 개발환경 설정 스크립트 for MacOS

Setting development environment using scripts

- cd 명령어를 사용하여 **스크립트 폴더로 이동**
- ls 명령어를 사용하여 **폴더 내부 파일 목록 확인**
- chmod +x 명령어를 사용하여 **모든 스크립트 파일에 권한 부여**

The terminal session shows the following steps:

```
cd repos/holyground/readable_code_cpp_development_environment/scripts/macos
ls
chmod +x install_clang.sh install_cmake.sh install_gcc.sh install_git.sh install_homebrew.sh install_vscode.sh setting_compilers.sh setting_development_environment.sh
```

The terminal window has three distinct sections highlighted by dashed boxes:

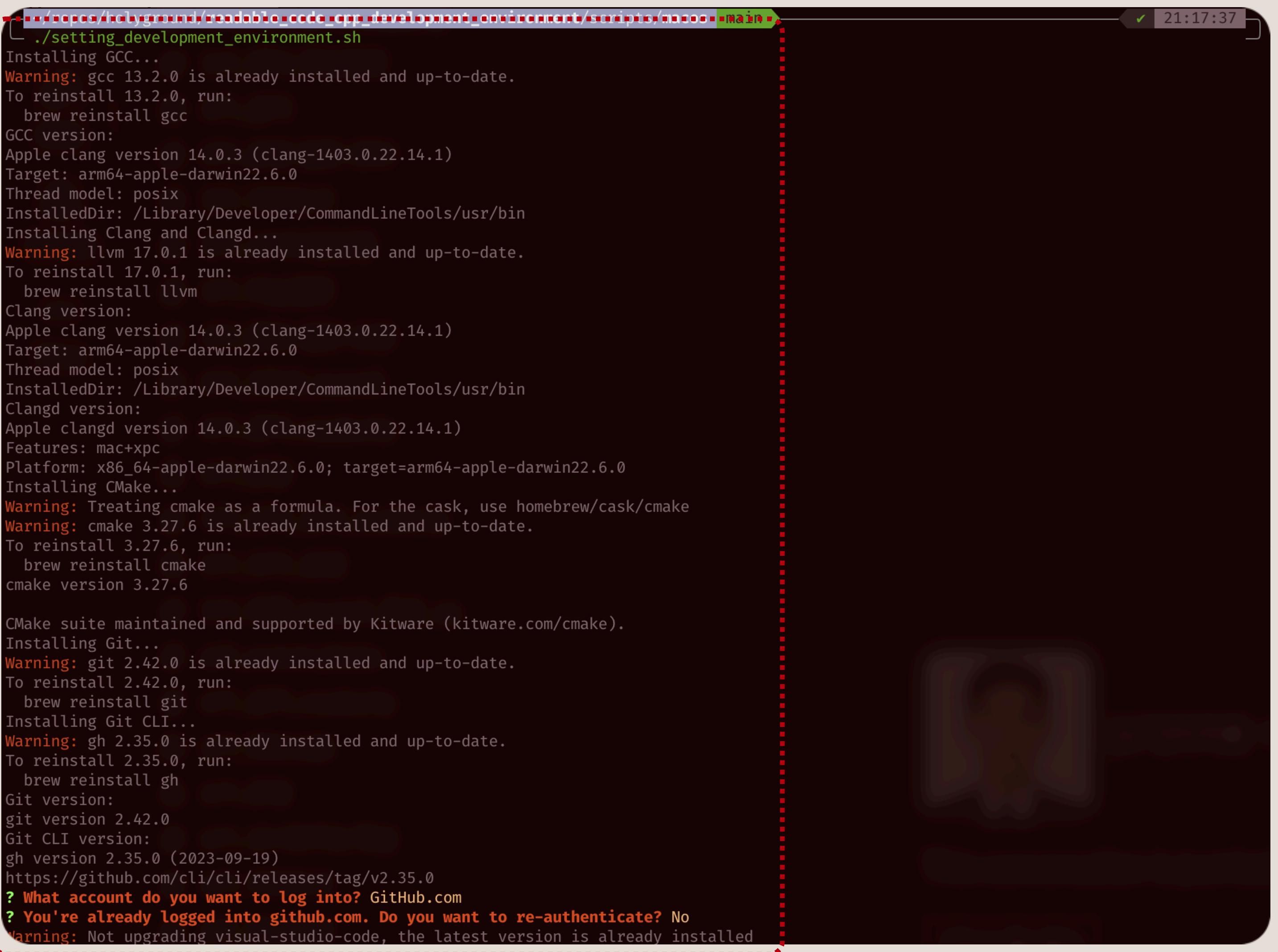
- Red Box (Top):** Contains the command `cd repos/holyground/readable_code_cpp_development_environment/scripts/macos`.
- Blue Box (Middle):** Contains the command `ls` and lists the files: `install_clang.sh`, `install_cmake.sh`, `install_gcc.sh`, `install_git.sh`, `install_homebrew.sh`, `install_vscode.sh`, `setting_compilers.sh`, and `setting_development_environment.sh`.
- Green Box (Bottom):** Contains the command `chmod +x` followed by a space-separated list of all the files listed in the blue box.

Timestamps in the terminal indicate the sequence of operations: 21:17:27, 21:17:29, and 21:17:30.

## 개발환경 설정 스트립트 for MacOS

Setting development environment using scripts

- ./ 명령어를 사용하여 `setting_development_environment.sh` 실행



```
./setting_development_environment.sh
Installing GCC...
Warning: gcc 13.2.0 is already installed and up-to-date.
To reinstall 13.2.0, run:
  brew reinstall gcc
GCC version:
Apple clang version 14.0.3 (clang-1403.0.22.14.1)
Target: arm64-apple-darwin22.6.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
Installing Clang and Clangd...
Warning: llvm 17.0.1 is already installed and up-to-date.
To reinstall 17.0.1, run:
  brew reinstall llvm
Clang version:
Apple clang version 14.0.3 (clang-1403.0.22.14.1)
Target: arm64-apple-darwin22.6.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
Clangd version:
Apple clangd version 14.0.3 (clang-1403.0.22.14.1)
Features: mac+xpc
Platform: x86_64-apple-darwin22.6.0; target=arm64-apple-darwin22.6.0
Installing CMake...
Warning: Treating cmake as a formula. For the cask, use homebrew/cask/cmake
Warning: cmake 3.27.6 is already installed and up-to-date.
To reinstall 3.27.6, run:
  brew reinstall cmake
cmake version 3.27.6

CMake suite maintained and supported by Kitware (kitware.com/cmake).
Installing Git...
Warning: git 2.42.0 is already installed and up-to-date.
To reinstall 2.42.0, run:
  brew reinstall git
Installing Git CLI...
Warning: gh 2.35.0 is already installed and up-to-date.
To reinstall 2.35.0, run:
  brew reinstall gh
Git version:
git version 2.42.0
Git CLI version:
gh version 2.35.0 (2023-09-19)
https://github.com/cli/cli/releases/tag/v2.35.0
? What account do you want to log into? GitHub.com
? You're already logged into github.com. Do you want to re-authenticate? No
Warning: Not upgrading visual-studio-code, the latest version is already installed
```

# 개발환경 설정 스트립트 for Ubuntu

## 개발환경 설정 스트립트 for Ubuntu

Components of 개발환경 설정 스트립트

- 하기와 같은 폴더 구조로 예제 프로젝트 작성(폴더, 파일)◦

```
└─ script_for_ubuntu◦  
    ├─ install_clang.sh◦  
    ├─ install_cmake.sh◦  
    ├─ install_gcc.sh◦  
    ├─ install_git.sh◦  
    ├─ install_vscode.sh◦  
    ├─ update_ubuntu.sh◦  
    ├─ setting_compilers.sh◦  
    └─ setting_development_environment.sh◦
```

## 개발환경 설정 스트립트 for Ubuntu

Components of 개발환경 설정 스트립트

- setting\_development\_environment.sh를 실행 시 개발환경이 셋업되는 시스템
- 각각 설치 스크립트를 분리하여 모듈화/재활용성을 확보

## 개발환경 설정 스트립트 for Ubuntu

Components of 개발환경 설정 스트립트 - install\_clang.sh

```
#!/bin/bash

# Install necessary dependencies
./update_ubuntu.sh
sudo apt install -y build-essential

# Install Clang and Clangd
echo "Installing Clang and Clangd ... "
wget https://apt.llvm.org/llvm.sh
chmod +x llvm.sh
sudo ./llvm.sh 14
sudo apt install -y clang clangd

# Verify installed versions
echo "Clang version:"
clang --version
echo "Clangd version:"
clangd --version
```

## 개발환경 설정 스트립트 for Ubuntu

Components of 개발환경 설정 스트립트 - install\_cmake.sh

```
#!/bin/bash

# Install necessary dependencies
./update_ubuntu.sh
sudo apt install -y build-essential

# Install CMake
echo "Installing CMake ... "
sudo apt install -y cmake

# Verify installed versions
cmake --version
```

## 개발환경 설정 스트립트 for Ubuntu

Components of 개발환경 설정 스트립트 - install\_gcc.sh

```
#!/bin/bash

# Install necessary dependencies
./update_ubuntu.sh
sudo apt install -y build-essential

# Install GCC
echo "Installing GCC ... "
sudo apt install -y gcc

# Verify installed versions
echo "GCC version:"
gcc --version
```

## 개발환경 설정 스트립트 for Ubuntu

Components of 개발환경 설정 스트립트 - install\_git.sh

```
#!/bin/bash

# Install necessary dependencies
./update_ubuntu.sh
sudo apt install -y curl

# Install Git
echo "Installing Git..."
sudo apt install -y git

# Install Git CLI
echo "Installing Git CLI..."
type -p curl >/dev/null || (sudo apt update && sudo apt install curl -y)
curl -fsSL https://cli.github.com/packages/githubcli-archive-keyring.gpg | sudo dd
of=/usr/share/keyrings/githubcli-archive-keyring.gpg \
&& sudo chmod go+r /usr/share/keyrings/githubcli-archive-keyring.gpg \
&& echo "deb [arch=$(dpkg --print-architecture)
signed-by=/usr/share/keyrings/githubcli-archive-keyring.gpg] https://cli.github.com/packages stable main" | sudo
tee /etc/apt/sources.list.d/github-cli.list > /dev/null \
&& sudo apt update \
&& sudo apt install gh -y

# Verify installed versions
echo "Git version:"
git --version
echo "Git CLI version:"
gh --version

# Log in to GitHub
gh auth login
```

## 개발환경 설정 스트립트 for Ubuntu

Components of 개발환경 설정 스트립트 - update\_ubuntu.sh

```
#!/bin/bash

sudo apt update -y
sudo apt upgrade -y
```

## 개발환경 설정 스트립트 for Ubuntu

Components of 개발환경 설정 스트립트 - install\_vscode.sh

```
#!/bin/bash

# Install necessary dependencies
./update_ubuntu.sh
sudo apt -y install software-properties-common apt-transport-https wget

# Install VSCode
echo "Installing VSCode ... "
wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add -
sudo add-apt-repository "deb [arch=$(dpkg --print-architecture)]
] https://packages.microsoft.com/repos/vscode stable main"
sudo apt install -y code

# Verify installed versions
echo "VSCode version:"
code --version
```

## 개발환경 설정 스트립트 for Ubuntu

Components of 개발환경 설정 스트립트 - setting\_compilers.sh

```
#!/bin/bash

./install_gcc.sh
./install_clang.sh
```

## 개발환경 설정 스트립트 for Ubuntu

Components of 개발환경 설정 스트립트 - setting\_development\_environment.sh

```
#!/bin/bash

./setting_compilers.sh
./install_cmake.sh
./install_git.sh
./install_vscode.sh
```

## 개발환경 설정 스크립트 for Ubuntu

Setting development environment using scripts

- cd 명령어를 사용하여 **스크립트 폴더로 이동**
- ls 명령어를 사용하여 **폴더 내부 파일 목록 확인**
- chmod +x 명령어를 사용하여 **모든 스크립트 파일에 권한 부여**

```
parallels@ubuntu-linux-22-04-desktop:~$ cd Desktop/readable_code_cpp_development_environment_setting/scripts/ubuntu/
parallels@ubuntu-linux-22-04-desktop:~/Desktop/readable_code_cpp_development_environment_setting/scripts/ubuntu$ ls
install_clang.sh  install_gcc.sh  install_vscode.sh      setting_development_environment.sh
install_cmake.sh   install_git.sh   setting_compilers.sh
parallels@ubuntu-linux-22-04-desktop:~/Desktop/readable_code_cpp_development_environment_setting/scripts/ubuntu$ chmod +x
install_clang.sh  install_cmake.sh  install_gcc.sh  install_git.sh  install_vscode.sh  setting_compilers.sh  setting_development_environ
ment.sh
```

## 개발환경 설정 스트립트 for Ubuntu

Setting development environment using scripts

- ./ 명령어를 사용하여 setting\_development\_environment.sh 실행

```
parallels@ubuntu-linux-22-04-desktop:~/Desktop/readable_code_cpp_development_environment_setting/scripts/ubuntu$ ./setting_development_environment.sh
[sudo] password for parallels:
Hit:1 http://packages.microsoft.com/repos/code stable InRelease
Hit:2 https://packages.microsoft.com/repos/vscode stable InRelease
Hit:3 https://packages.microsoft.com/ubuntu/22.04/prod jammy InRelease
Hit:4 https://cli.github.com/packages stable InRelease
Hit:5 http://ports.ubuntu.com/ubuntu-ports jammy InRelease
Get:6 http://ports.ubuntu.com/ubuntu-ports jammy-updates InRelease [119 kB]
Get:8 http://ports.ubuntu.com/ubuntu-ports jammy-backports InRelease [109 kB]
Get:9 http://ports.ubuntu.com/ubuntu-ports jammy-security InRelease [110 kB]
Get:10 http://ports.ubuntu.com/ubuntu-ports jammy-updates/main arm64 DEP-11 Metadata [101 kB]
Get:11 http://ports.ubuntu.com/ubuntu-ports jammy-updates/universe arm64 DEP-11 Metadata [285 kB]
Hit:7 https://apt.llvm.org/jammy llvm-toolchain-jammy-17 InRelease
Get:12 http://ports.ubuntu.com/ubuntu-ports jammy-updates/multiverse arm64 DEP-11 Metadata [212 B]
Get:13 http://ports.ubuntu.com/ubuntu-ports jammy-backports/main arm64 DEP-11 Metadata [3488 B]
Get:14 http://ports.ubuntu.com/ubuntu-ports jammy-backports/universe arm64 DEP-11 Metadata [17.3 kB]
Get:15 http://ports.ubuntu.com/ubuntu-ports jammy-security/main arm64 DEP-11 Metadata [43.0 kB]
Get:16 http://ports.ubuntu.com/ubuntu-ports jammy-security/universe arm64 DEP-11 Metadata [39.8 kB]
Fetched 828 kB in 4s (226 kB/s)
Reading package lists... Done
```

# QnA



# Thanks!

소속  
이름  
연락처