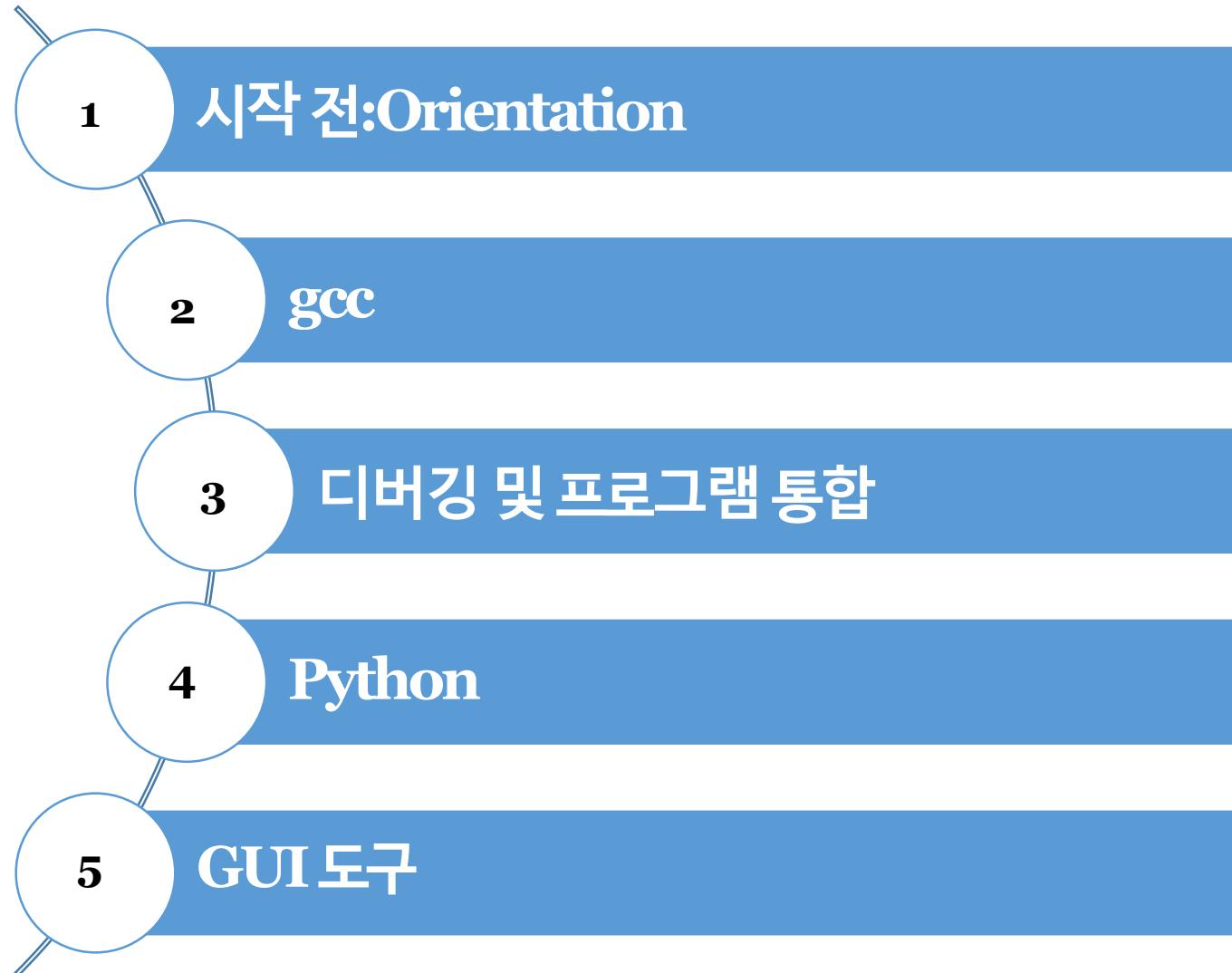


Pre-Requisites

- ◆ 훈련과정명 : OS 기본
- ◆ 훈련기간 : 2023.05.30 ~ 2023.06.02

목차





Syllabus

학습목표

- 이 워크샵에서는 Pre-Requisites에 대해 알 수 있습니다.

눈높이 체크

- 직무와 능력단위를 알고 계신가요?



Pre-Requisites

임베디드 소프트웨어(이하 임베디드 SW)의 개념

- **임베디드 SW의 정의**

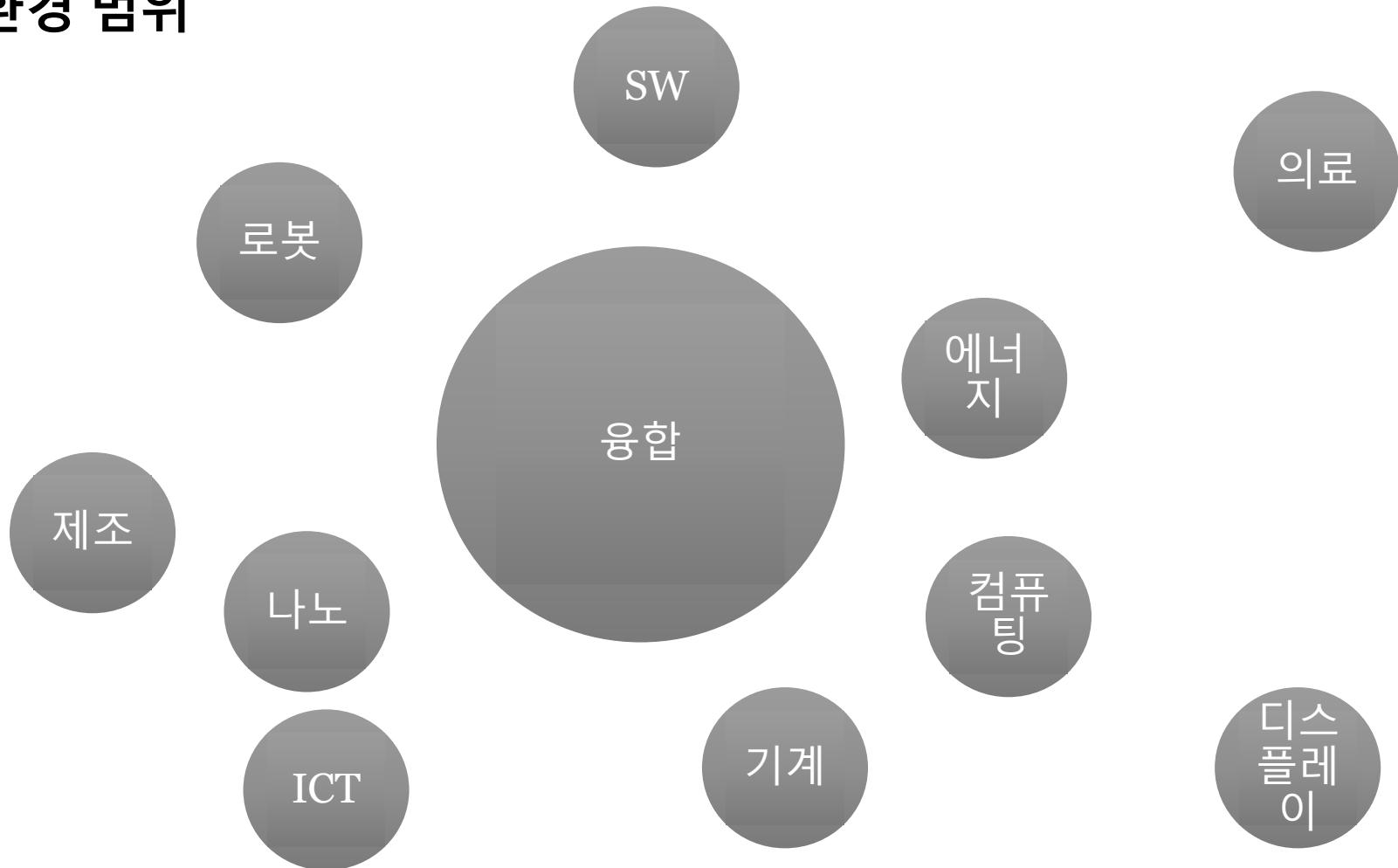
- 임베디드 SW는 전체 시스템에 탑재되어 주변 하드웨어 또는 환경을 제어하는 내장형 프로그램이다.
- 곧, 임베디드 소프트웨어란 IT 차세대 성장동력 분야를 비롯한 정보가전 및 정보통신, 항공기, 차량, 로봇, 산업기기, 의료기기 등 다양한 산업 분야의 디지털 제품에 내장되어 하드웨어의 제어, 통신, 멀티미디어, 인터넷, 게임, 인공지능, 유비쿼터스 컴퓨팅 등 기본 기능 및 다양한 부가기능을 제공함으로써 제품의 경쟁력과 부가가치를 높여주는 소프트웨어를 말한다.
- 임베디드 SW는 스마트폰에서부터 TV, 냉장고, 에어컨 등의 정보 가전뿐만 아니라, 자동차, 항공기, 원자력발전소 등의 안전 필수 시스템(safety-critical system)까지 광범위하게 사용되고 있다.
- 요즘은 클라우드, 인공지능, 빅데이터 등의 다양한 IT 융합 분야에서 융합 SW로 불리며 급속도로 적용 분야가 확대되고 있는 추세이다.



Pre-Requisites

산업 환경

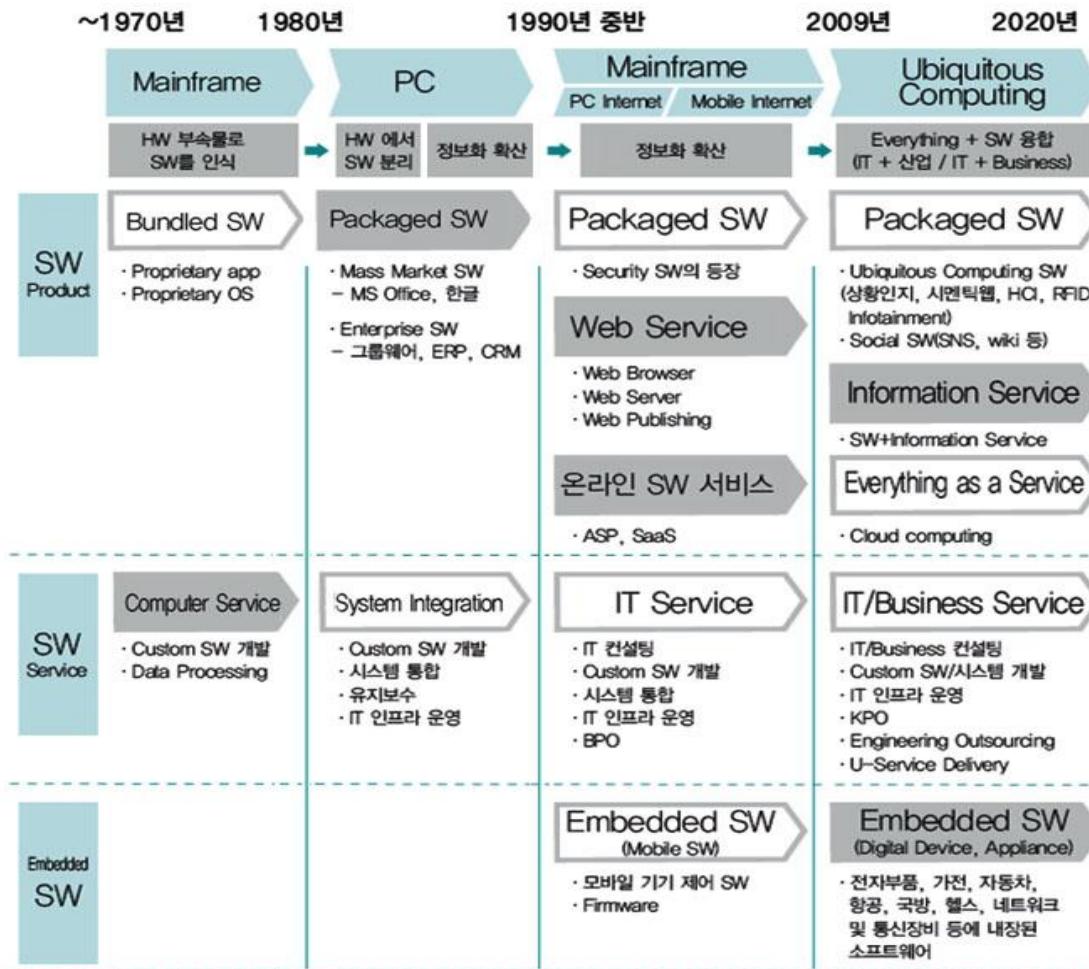
- 산업 환경 범위



Pre-Requisites

산업 환경

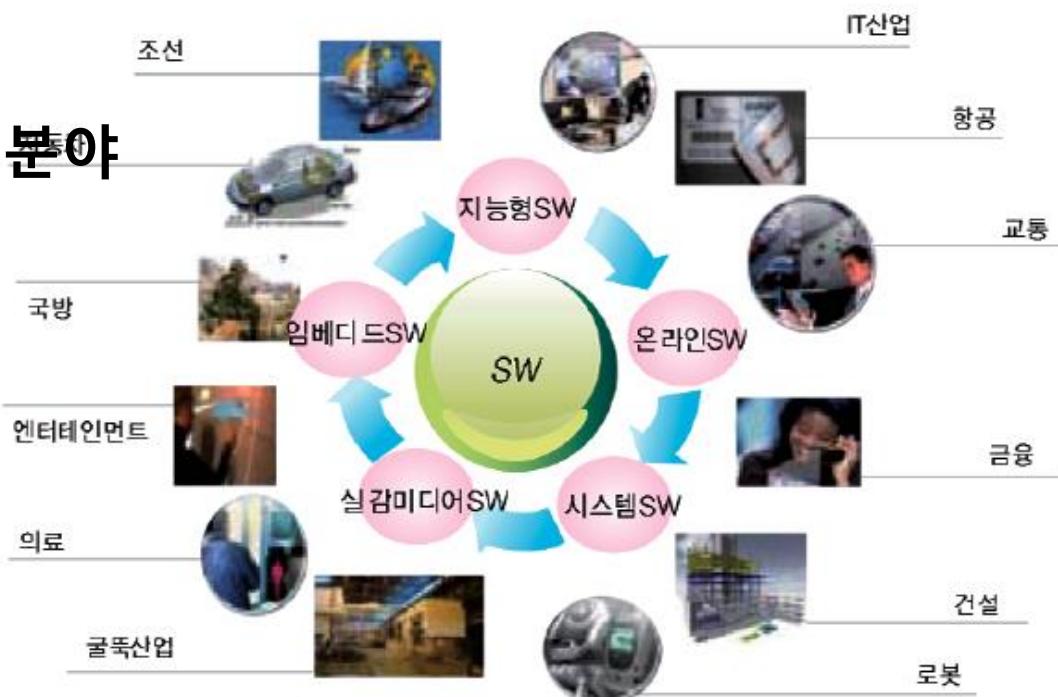
● 기술혁신 단계에 따른 소프트웨어 산업의 외연



Pre-Requisites

산업 환경

● 소프트웨어 기술 및 적용 분야



구분

설명

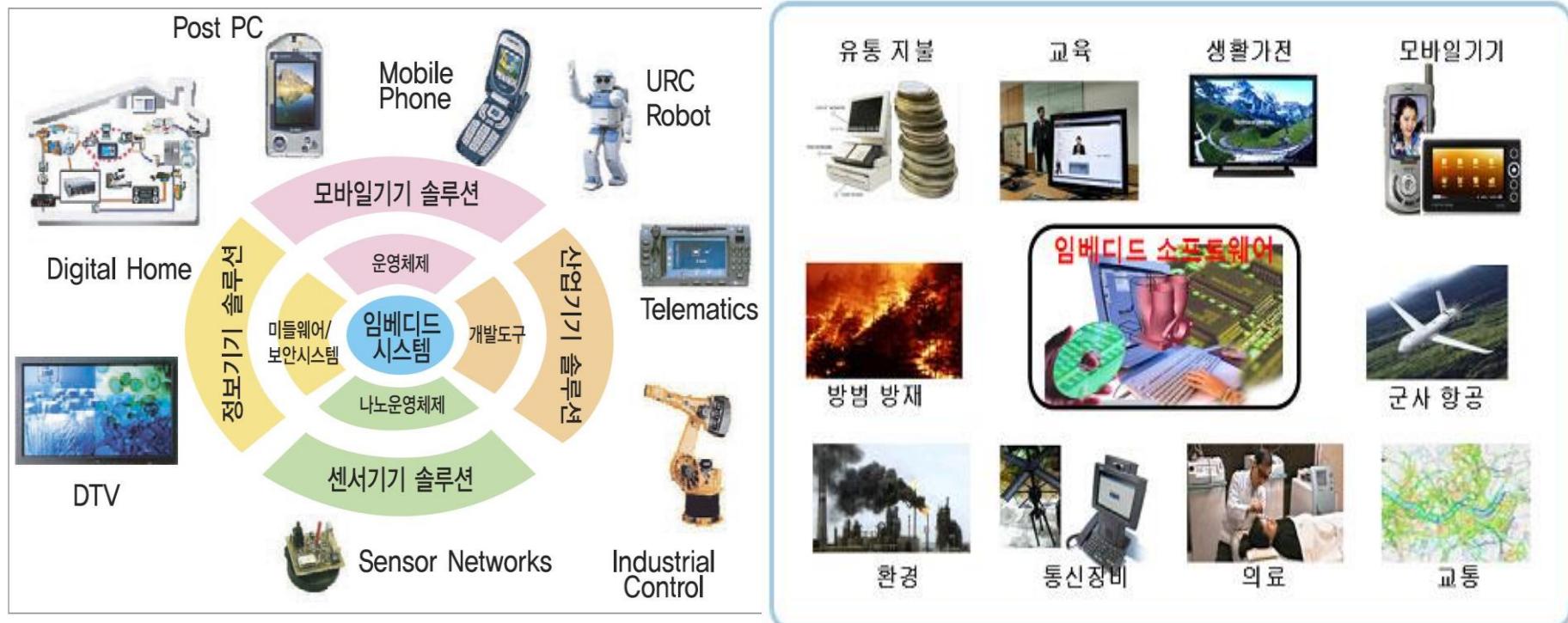
시스템 SW	<ul style="list-style-type: none">시스템가상화SW, 운영체제, 저장시스템 SW, 데이터관리 SW, 병렬분산처리 SW, 프로그램 언어, 클라우드 컴퓨팅
공개 SW	<ul style="list-style-type: none">그린 SW 플랫폼, SaaS 플랫폼, 웹기반 운영체제, 분산 컴퓨팅 SW
지능형 인터페이스	<ul style="list-style-type: none">언어처리 SW, 음성처리 SW, 다국어처리 SW, 인공지능 SW, 지식검색 SW
차세대 웹	<ul style="list-style-type: none">유비쿼터스 웹 SW, 비디오 웹 SW, 모바일 웹 SW, 리치 웹 SW, 시맨틱 웹 SW, 지능형 정보검색 SW
실감미디어 SW	<ul style="list-style-type: none">영상처리 SW, 3D 영상 저작 SW, 실감형 인터랙션 SW, 가상시뮬레이션 SW, 실세계 오감 연동형 SW
임베디드 SW	<ul style="list-style-type: none">임베디드 운영체제, 임베디드 미들웨어 및 프레임워크, 임베디드가상화 SW, 임베디드 SW 개발 환경, 임베디드 SW, 사회안전시스템 SW, 공통응용 및 서비스

Pre-Requisites

임베디드 소프트웨어(이하 임베디드 SW)의 개념

임베디드 시스템

임베디드 소프트웨어

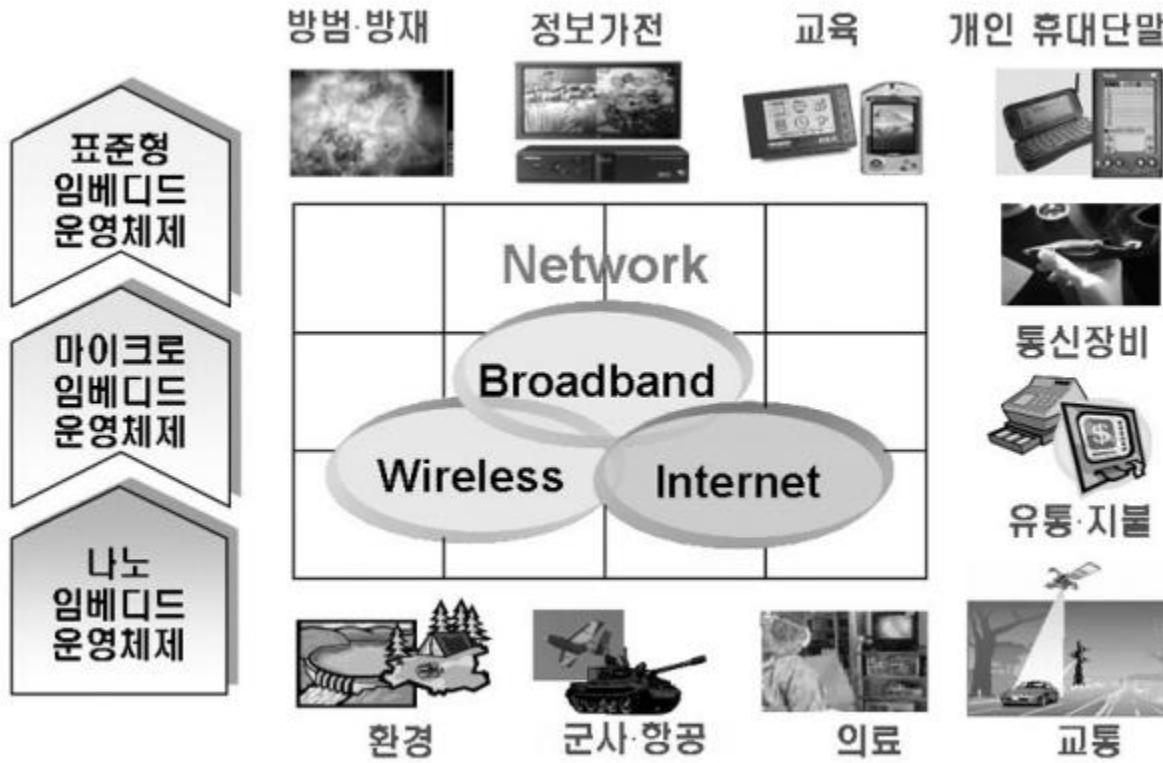




Pre-Requisites

임베디드 소프트웨어(이하 임베디드 SW)의 개념

- 임베디드 S/W 기반의 응용서비스





Pre-Requisites

임베디드 소프트웨어(이하 임베디드 SW)의 개념

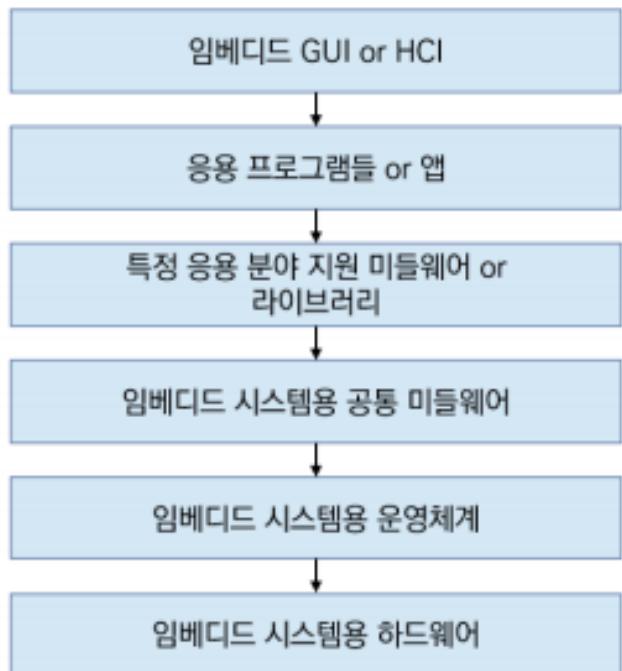
- **임베디드 SW의 특징**

- 임베디드 소프트웨어는 임베디드 시스템을 구성하는 소프트웨어로서 운영체제, 미들웨어, 응용 소프트웨어 등으로 이루어져 있다. 여기에 임베디드 시스템을 개발하기 위하여 운영체제 개발도구인 임베디드 시스템 개발 도구와 응용 소프트웨어 개발을 효율적으로 하도록 지원해 주는 그래픽 사용자 인터페이스 기반의 통합개발 환경도구가 하나의 패키지로 제공되고 있다.
- 임베디드 소프트웨어는 다양한 특성을 가진 임베디드 시스템에 내장될 수 있도록 저렴한 가격, 소형화, 저전력 소비, 고신뢰성, 소프트웨어의 기능 및 성능의 최적화, 하드웨어에 대한 효율적 자원관리 등이 임베디드 소프트웨어가 지녀야 할 기본적인 특성이다.

Pre-Requisites

임베디드 소프트웨어(이하 임베디드 SW)의 개념

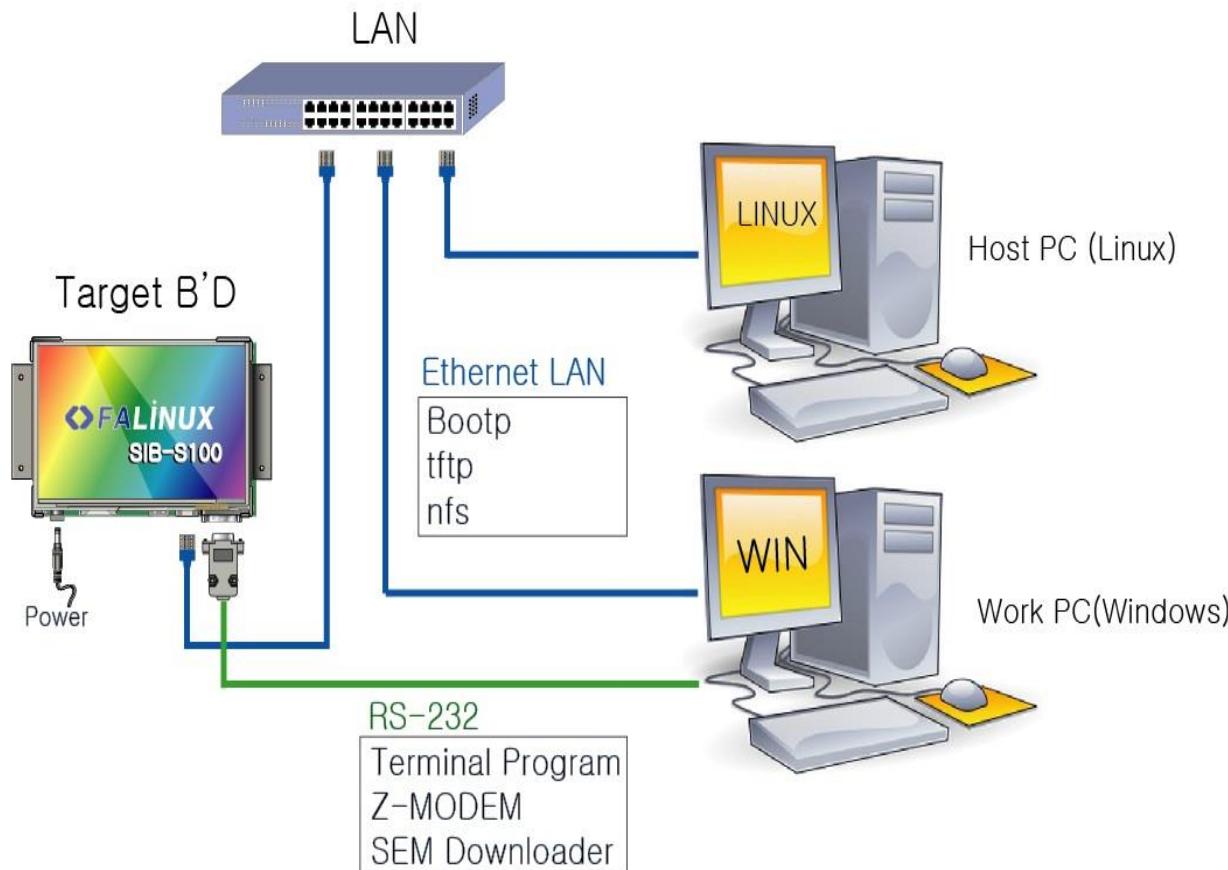
● 임베디드 SW 계층 구조



Pre-Requisites

임베디드 소프트웨어(이하 임베디드 SW)의 개념

- 임베디드 툴체인 예





Pre-Requisites

임베디드 소프트웨어(이하 임베디드 SW)의 개념

● 임베디드 SW의 연구 동향

영역	연구 개발 동향
개인	단말 장치와 연동하여 스마트 워치, 바이오 기기, 스마트 뱅킹 등 다양한 융복합 형태로의 진화를 보이고 있다.
산업	스마트 팩토리, 무인 자동차, UAM 등 기간 제조 산업과 함께 최종단의 정보 수집 장치로 경량화 유/무선 NW 기반으로 센서 간 연동을 통하여 임베디드 S/W 기술을 활용하고 있다.
교육	스마트 온라인 교육 및 스마트 스쿨 기반의 인프라를 구성하기 위하여 아두이노, 스케치 SW를 활용하고, 하드웨어로 아두이노, 라즈베리 파이 등을 통하여 코딩 교육의 생태계를 구축하고 있는 상황이다.
의료	사람과 직결된 안전성 확보를 위하여 다양한 표준 기반의 S/W를 검증하고 있으며 보안 강화, 비용 절감, 시스템 안전성 효과를 높이기 위해 품질 검사 기준을 마련하고 있다.
공통	임베디드 시스템 S/W의 안정성을 위하여 자동차 분야 IEEE26262, 항공 분야 DO-178, 전기전자 분야 IEEE-61508 기반의 표준이 마련되어 있어 S/W를 평가하고 보안사고 대응을 위하여 임베디드 설계 원칙 등에 대하여 연구하고 있다.



Pre-Requisites

임베디드 개발자가 일하는 분야

- **현대자동차**

- 전자제어 S/W : 자동차 각 부품 제어 및 RTOS 시스템 개발
- 지능형차 S/W : 인공지능 시스템 / ADAS / 음성인식 시스템 개발
- 차량IT S/W : 차량 부품 / 동작 모니터링 및 제어 시스템 개발

- **한화디펜스**

- 발사체계 S/W : 발사대 제어, 벌사대 성능시험 SW 개발
- 국방로봇 S/W : 자율주행 / 물체탐지 / 추적 SW 개발
- 에너지 S/W : 배터리 및 에너지 관리 SW 개발
- 제어장치 S/W : 전투장치 통제장치 / 양산장비 사격통제 장치 SW 개발



Pre-Requisites

임베디드 개발자가 일하는 분야

- **삼성**

- **Firmware 개발** : 메모리 Firmware 개발 / 검증 툴 개발 / OS System 개발
- **Smart Factory 개발** : 공장 제어 및 모니터링 SW 개발
- **HW 제어 Firmware 개발** : 다양한 가전기기내의 SW 개발
- **프레임 워크 및 OS 개발** : 스마트 기기내의 프레임워크 및 Operating System (Tizen/Android) 개발

- **한국항공우주산업**

- **비행체 시스템 SW 개발** / 항공전자 시스템 개발 / 항공 HW 제어 SW 개발



임베디드 개발자가 일하는 분야

- **LIG 넥스원**

- 레이다 임베디드 SW 개발 : 레이다 신호처리, 표적 추적 SW 개발
- 레이다 신호처리 / 통제제어 SW 개발 : 신호처리, 제어 관련 SW 개발

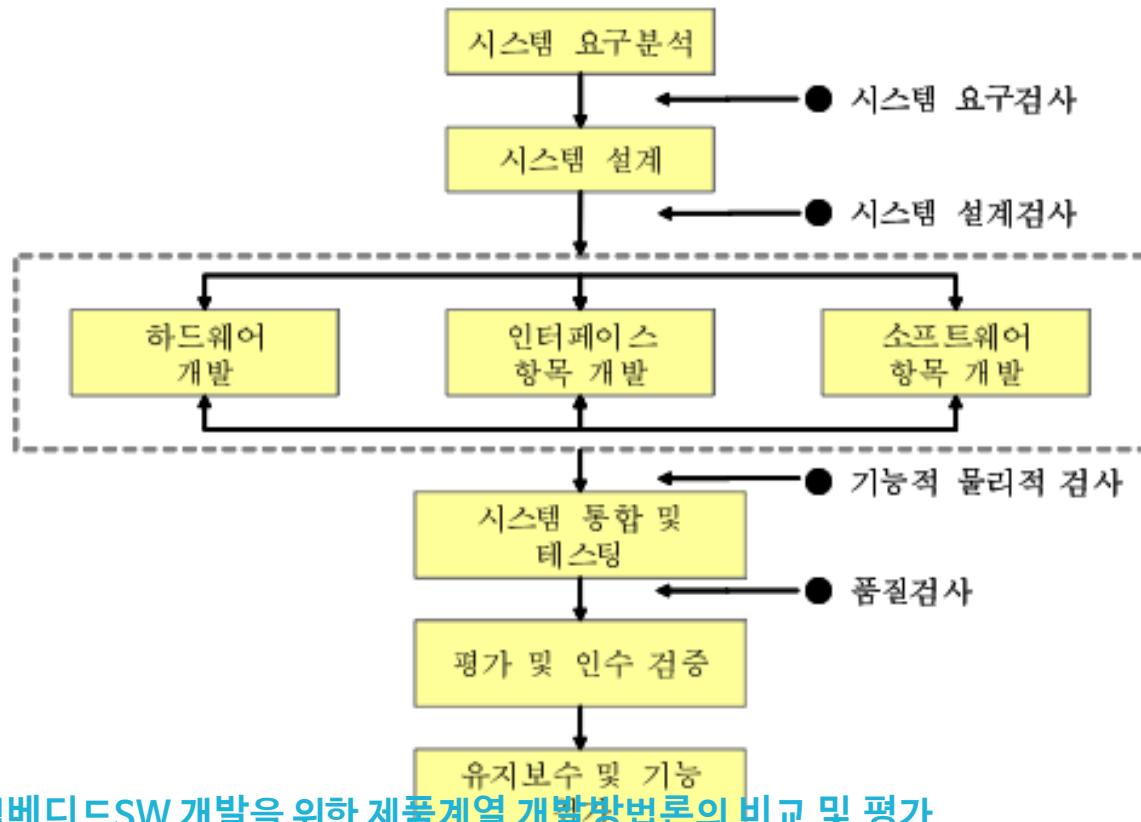
- **한화테크원**

- 임베디드 리눅스 개발 : 리눅스 환경 BSP 개발, 리눅스 Cloud 서비스 연동, 형상/빌드 관리
- 영상저장장치 SW 개발 : 대용량 미디어 저장관리

Pre-Requisites

임베디드 소프트웨어 개발 절차

- **임베디드 시스템 개발 프로젝트 수명주기**
- 임베디드 시스템 개발 프로젝트의 수명주기는 그림에서 보는 바와 같이 크게 6단계로 나눌 수 있으며 각 단계에 대응하는 검증 절차들을 가진다.

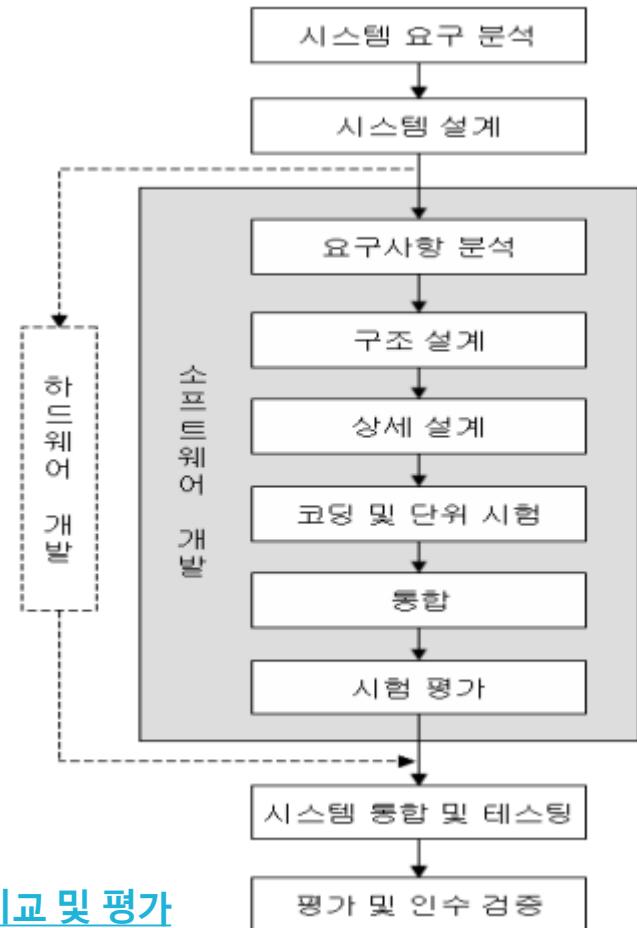


Pre-Requisites

임베디드 소프트웨어 개발절차

- **임베디드SW 개발절차**

- 임베디드 시스템의 수명주기를 기반으로 임베디드SW의 개발
- 임베디드SW 개발절차를 보면 시스템 요구사항 분석 및 설계 이후 HW 개발과 SW 개발로 나누어 동시 진행하여 개발한 이후 시스템 통합 및 테스팅과 평가 및 인수 검증을 수행 한다.



『1과목』 프로그래밍 언어

Gcc

(GNU Compiler Collection)





학습목표

- 개발 도구와 프로그래밍 언어(들)를 사용하여 단위, 공통 모듈의 프로그램을 구현할 수 있다.
- 요구사항이 반영된 구현 코드를 컴파일하여 모든 오류와 경고를 소스를 수정하여 제거할 수 있다.

눈높이 체크

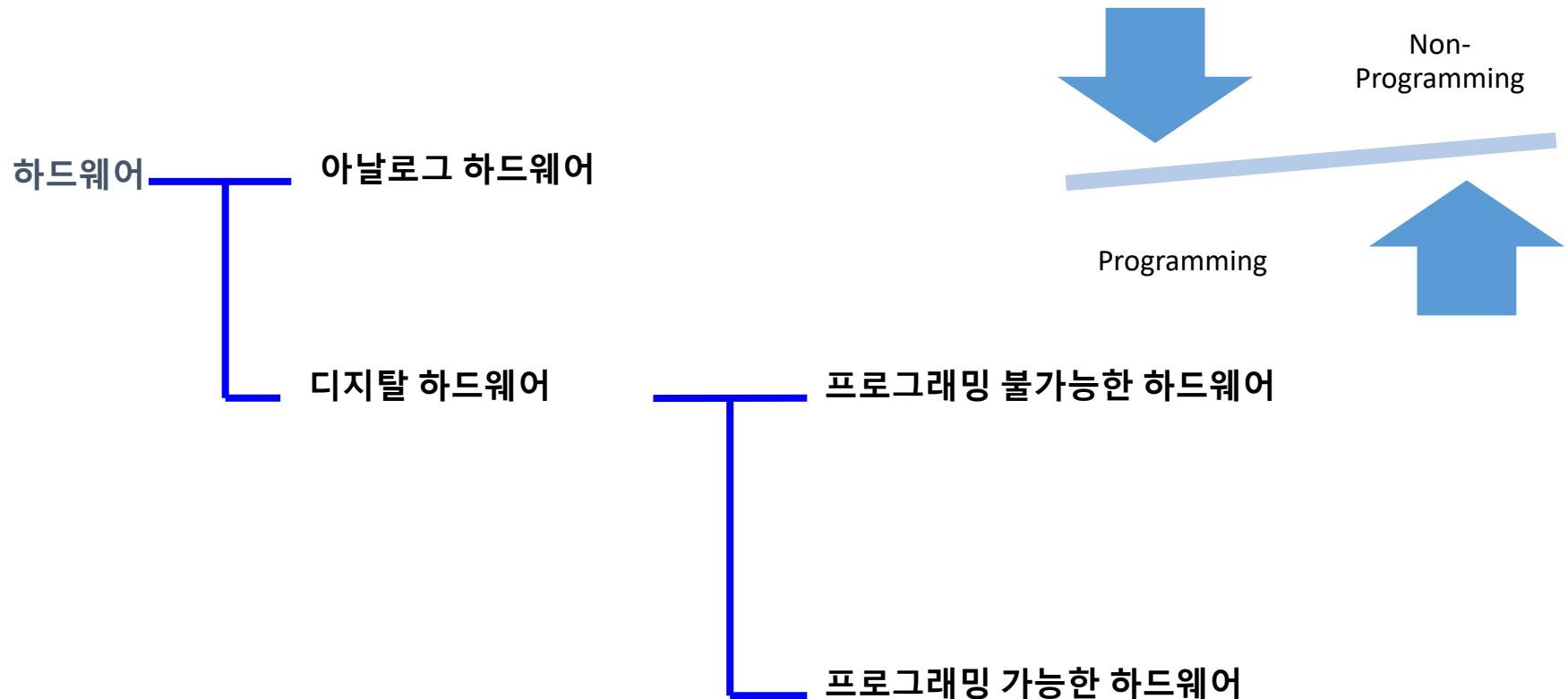
- 개발 도구와 프로그래밍 언어(들)를 알고 계신가요?



Syllabus

하드웨어와 프로그래밍의 관계

- 하드웨어와 소프트웨어의 구성





하드웨어와 프로그래밍의 관계

- **프로그래밍 불가능한 하드웨어**

- 대부분의 기능이 ASIC(주문형 반도체)으로 구현되어 있는 경우
- 하드웨어만으로 구성되어 있거나, ASIC을 사용하는 경우에는 개발자가 프로그래밍을 할 수 없는 구조로 되어 있다.

- **프로그래밍 가능한 하드웨어**

- CPU, RAM, ROM 등으로 구성이 되어 있는 경우
- 이와 같은 경우에는 개발자가 프로그래밍을 해서 하드웨어에 **upload** 한 후 그 소프트웨어가 직접 하드웨어를 제어할 수 있다.



임베디드에서 사용하는 운영체제

- 운영체제가 없는 경우

- 작은 규모의 단순한 임베디드 시스템의 경우는 운영체제가 없는 시스템도 많다.

- 상용화된 실시간 운영체제를 사용하는 경우

- 매우 수준높은 안정성이나 실시간성을 요구하는 경우에는 VxWorks나 pSOS, NuCleus등의 상용 실시간 운영체제를 필요로 한다

- 임베디드 리눅스를 사용하는 경우

- 근래에 가장 빠르게 성장하고 있는 임베디드 운영체제로, 기존의 상용 임베디드 운영체제 시장을 빠르게 잠식하고 있다.
- ex) uLinux등이 대표적인 임베디드 리눅스이다

Non-Operating System

Operating System



임베디드 개발자가 사용하는 프로그래밍 언어는?

- C언어

- 가장 강력한 컴퓨터 프로그래밍 언어 중 하나인 C는 임베디드 시스템 프로그래밍을 위한 사실상의 선택입니다. 이것은 이 언어가 제공하는 낮은 수준의 액세스로 인해 전혀 놀라운 사실이 아닙니다. 컴파일된 C 프로그램의 낮은 메모리 사용량은 C가 최고의 임베디드 시스템 프로그래밍 언어로 인정받는 요인이기도 합니다. 또한 이 정적으로 유형이 지정된 언어는 다른 프로그래밍 언어와 비교할 수 없는 속도를 제공합니다.

- C++

- C가 초보자를 위한 임베디드 시스템 프로그래밍을 위한 최고의 솔루션이라면, C++ 확실히 더 노련한 개발자에게 적합합니다. C와 같이 놀랍도록 빠른 이 언어는 저수준 시스템 리소스에 대한 액세스를 제공하고 메모리 스택에 불필요한 시스템 호출이 없는지 확인합니다. 객체 지향 프로그래밍 패러다임의 균일한 제공 덕분에 베테랑 개발자는 이전 버전보다 C++로 훨씬 더 많은 것을 달성할 수 있습니다.



임베디드 개발자가 사용하는 프로그래밍 언어는?

● Java

- Java 는 기존 데스크탑 소프트웨어를 위한 실행 가능한 솔루션이며 임베디드 시스템 프로그래밍을 위한 훌륭한 후보입니다. 안정적인 특성과 수많은 개발 도구의 가용성 덕분에 Java 프로그램은 일반적으로 임베디드 시스템 개발자에게 높은 평가를 받고 있습니다. 게다가, 일단 당신이 자바로 임베디드 시스템 프로그램을 작성하면, 당신은 그것을 다른 곳으로 쉽게 이식할 수 있습니다. IoT 플랫폼, 모두 JVM의 혁신적인 구현 덕분입니다.

● Python

- Web 및 Enterprise 세계의 주요 업체로 자리 잡은 후, 파이썬 최근 몇 년 동안 핵심 임베디드 시스템 프로그래밍 언어로 추진력을 얻었습니다. 마이크로컨트롤러가 날로 강력해짐에 따라 오늘날 임베디드 장치 내에서 Python 프로그램을 실행하는 것은 매우 쉽습니다. 간결하고 읽기 쉬운 코딩 스타일 덕분에 Python은 새로운 프로그래머를 위한 임베디드 시스템을 구축하기 위한 최고의 솔루션이 되었습니다. Python은 개발자에게 임베디드 시스템을 위한 빠른 프로그램을 작성할 수 있는 기능을 제공합니다.



Syllabus

임베디드 개발자가 사용하는 프로그래밍 언어는?

- 이외에도 Rust, Ada, JavaScript, Go, Lua, NS#, Assembly, C# 등 있다.



1. gcc(GNU Compiler Collection)

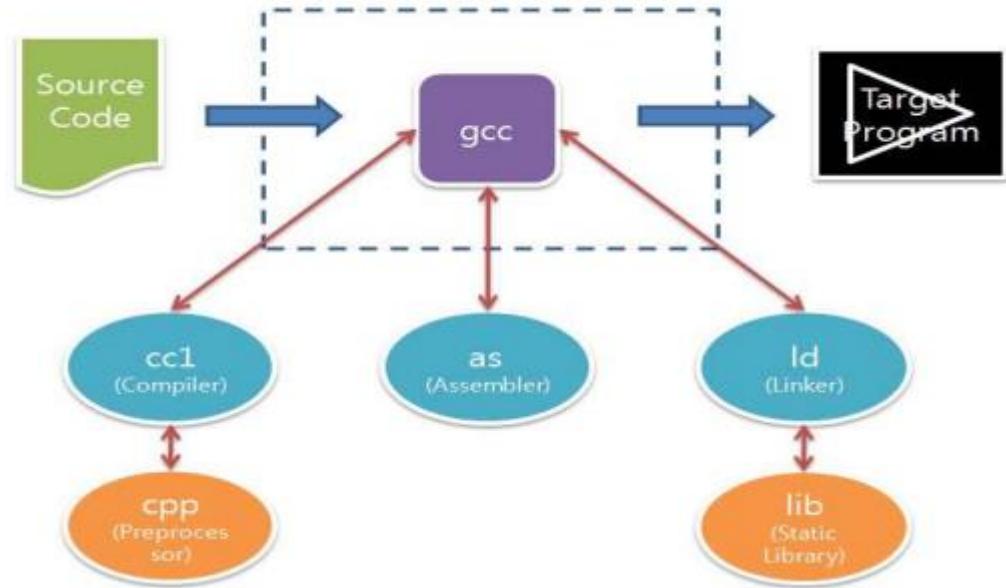
gcc의 개요

- gcc는 GNU 프로젝트의 일환으로 리처드 스톤만에 의해 개발되었다. 초기에는 C만을 지원하였고 이후 JAVA, FORTRAN, C++, ADA, Objective C 등을 컴파일한다.
- GNU 컴파일러 모음(GCC: GNU Compiler Collection)은 GNU 프로젝트의 한 축으로 개발되어 널리 쓰이는 컴파일러이다. 자유 소프트웨어 중에 가장 많이 알려진 것들 중 하나인 GCC는 처음 C만을 지원하였던 컴파일러로 명칭도 GNU C 컴파일러이다.
- GCC는 GNU 컴파일러 모음의 파트인 GNU C 컴파일러(GNU C Compiler)의 약자로 쓰인다. 이후에 C++, 자바, 포트란, 에이다 등 다른 언어를 컴파일할 수 있게 되면서, 본명칭으로 변경되었다.
- 지원 프로그래밍 언어는 C, 오브젝티브 C++, 자바, 에이다, 고언어를 위한 프런트엔드가 포함되어 있다. 이용은 가능하지만 표준에는 미포함되는 것으로 Modula-2, Modula-3, PL/I, VHDL, 파스칼, 머큐리 등이 있다.

1. gcc(GNU Compiler Collection)

gcc의 동작 원리

- gcc의 동작 개념도



- gcc는 소스 파일로부터 실행 파일을 만들 때까지 상기 과정을 진행한다. 즉, gcc는 소스 코드를 cpp를 이용하여 전처리하고, cc1을 이용하여 컴파일, as로 어셈블링, ld로 링크 등을 수행하기 위하여 cpp, cc1, as, ld 등을 호출한다. 이후 NFS 등을 활용하여 생성된 실행 파일을 타깃 시스템으로 전송한다.

1. gcc(GNU Compiler Collection)

Windows 환경에서 GCC 사용해 보기

1. MinGW-w64 다운로드

- **MinGW-w64 - for 32 and 64 bit Windows**

<https://sourceforge.net/projects/mingw-w64/files/>

The screenshot shows a web browser displaying the SourceForge project page for MinGW-w64. The URL in the address bar is <https://sourceforge.net/projects/mingw-w64/files/>. The page features a navigation menu at the top with options like 'Open Source Software', 'Business Software', 'Resources', 'Menu', and a search bar. The main content area lists various project categories and their details, such as 'Junk' (2021-07-01), 'Multilib Toolchains(Targetting Win32 and Win64)' (2016-05-19), 'mingw-w64' (2015-07-25), 'Cygwin Snapshots' (2013-07-07), '3rd party development tools' (2012-08-07), 'External binary packages (Win64 hosted)' (2011-11-15), 'Toolchains targetting Win64' (2011-05-06), 'Toolchains targetting Win32' (2011-05-06), 'Toolchain sources' (2011-05-06), 'Toolchains targetting NonWin' (2010-12-17), and 'README.rst' (2018-05-24). A red callout box with the text 'Click!' is positioned over the 'mingw-w64' link in the sidebar. To the right of the main content, there are two advertisements: one for 'Click Here for More Information' and another for 'Recommended Projects' featuring 'MinGW - Minimalist GNU for Windows' and 'MinGW-builds'. At the bottom left, there's a download progress bar for 'x86_64-8.1.0-release...7z'.

1. gcc(GNU Compiler Collection)

Windows 환경에서 GCC 사용해 보기

1. MinGW-w64 다운로드

- MinGW-w64 - for 32 and 64 bit Windows

<https://sourceforge.net/projects/mingw-w64/files/>

- 위의 링크에서 x86_64-posix-seh 파일(.7z)을 다운받습니다. installer로 설치할 경우 제대로 설치가 되지 않는 문제가 발생

MinGW-W64 Online Installer

- [MinGW-W64-install.exe](#)

MinGW-W64 GCC-8.1.0

- [x86_64-posix-sjlj](#)
- [x86_64-posix-seh](#)
- [x86_64-win32-sjlj](#)
- [x86_64-win32-seh](#)
- [i686-posix-sjlj](#)
- [i686-posix-dwarf](#)
- [i686-win32-sjlj](#)
- [i686-win32-dwarf](#)

Click!

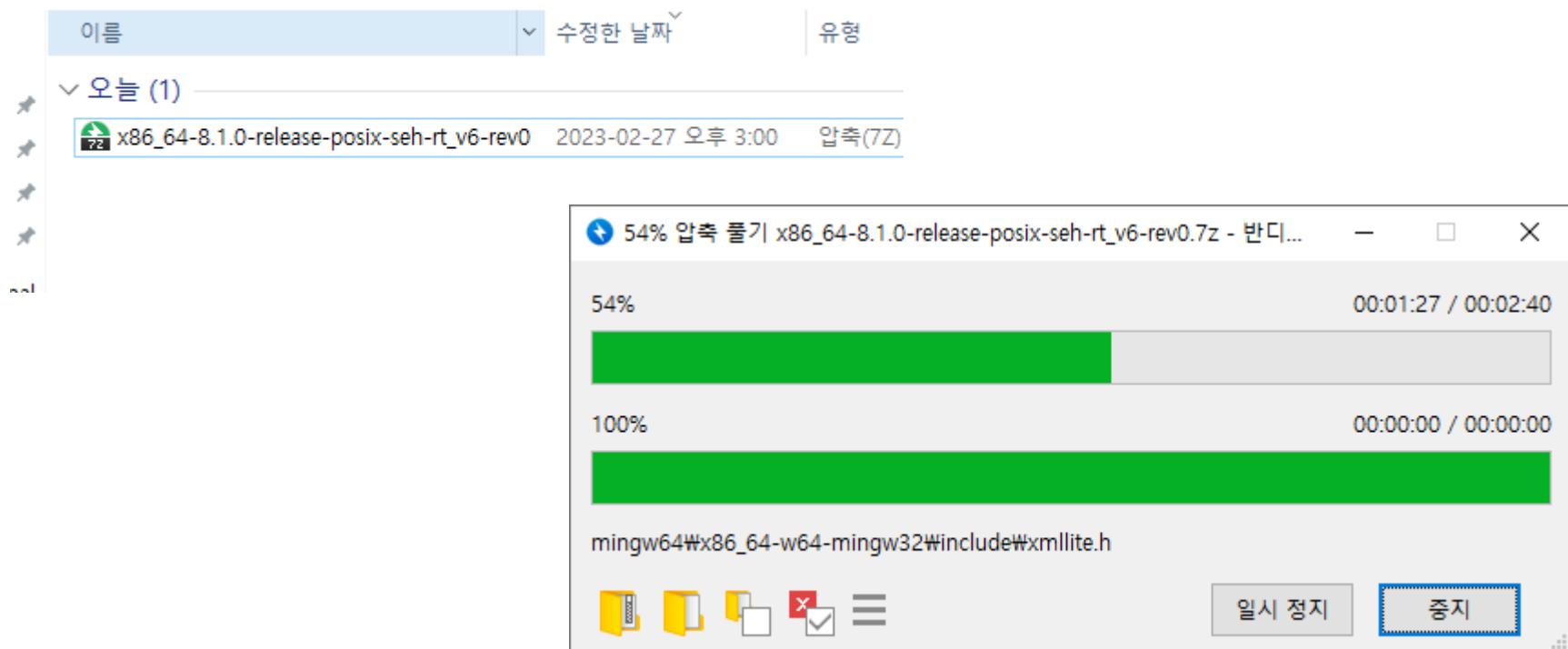
1. gcc(GNU Compiler Collection)

Windows 환경에서 GCC

2. 압축 해제

- 다운받은 .7z 파일의 압축을 해제하고 원하는 위치에 저장해줍니다.

내 PC > 로컬 디스크 (C:) > 사용자 > k8s > 다운로드



1. gcc(GNU Compiler Collection)



Windows 환경에서 GCC

2. 앱축 해제

- C:\DEV\mingw64 이 위치에 저장했습니다.

> 내 PC > 로컬 디스크 (C:) > DEV > mingw64

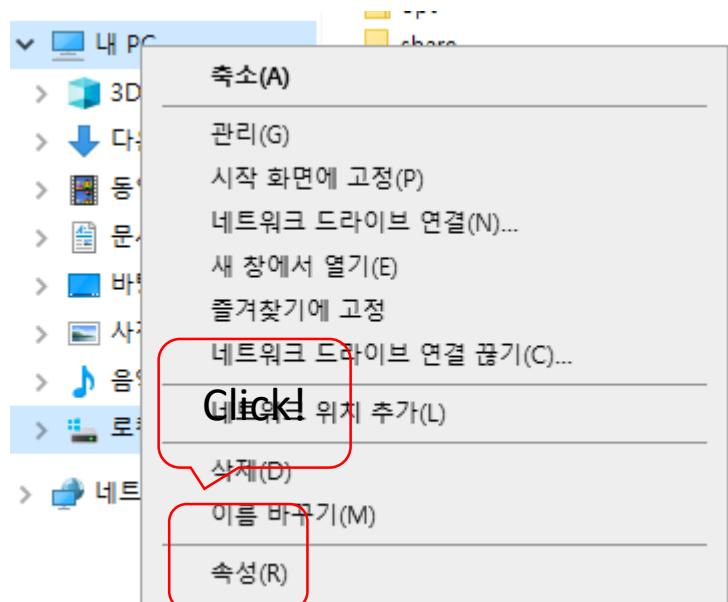
이름	수정한 날짜	유형
bin	2018-05-12 오후 4:29	파일 폴더
etc	2018-05-12 오후 4:02	파일 폴더
include	2018-05-12 오후 4:29	파일 폴더
lib	2018-05-12 오후 4:29	파일 폴더
libexec	2018-05-12 오후 3:10	파일 폴더
licenses	2018-05-12 오후 4:29	파일 폴더
opt	2018-05-12 오후 3:40	파일 폴더
share	2018-05-12 오후 4:29	파일 폴더
x86_64-w64-mingw32	2018-05-12 오후 12:07	파일 폴더
build-info	2018-05-12 오후 4:29	텍스트 문서

1. gcc(GNU Compiler Collection)

Windows 환경에서 GCC

3. 환경 변수 설정

- 명령 프롬프트의 어느 위치에서나 GCC를 실행시킬 수 있도록 환경 변수 (Path)를 설정해줘야 합니다.



1. gcc(GNU Compiler Collection)

Windows 환경에서 GCC

3. 환경 변수 설정

- 명령 프롬프트의 어느 위치에서나 GCC를 실행시킬 수 있도록 환경 변수 (Path)를 설정해줘야 합니다.

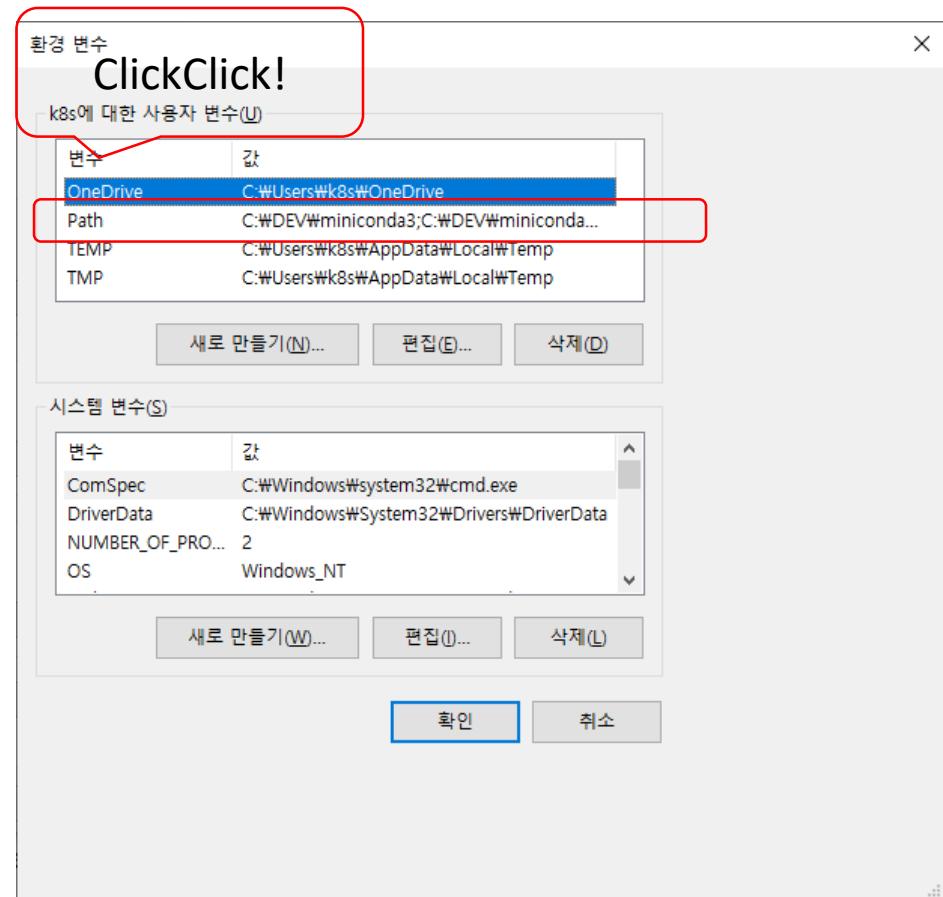


1. gcc(GNU Compiler Collection)

Windows 환경에서 GCC

3. 환경 변수 설정

- 명령 프롬프트의 어느 위치에서나 GCC를 실행시킬 수 있도록 환경 변수 (Path)를 설정해줘야 합니다.

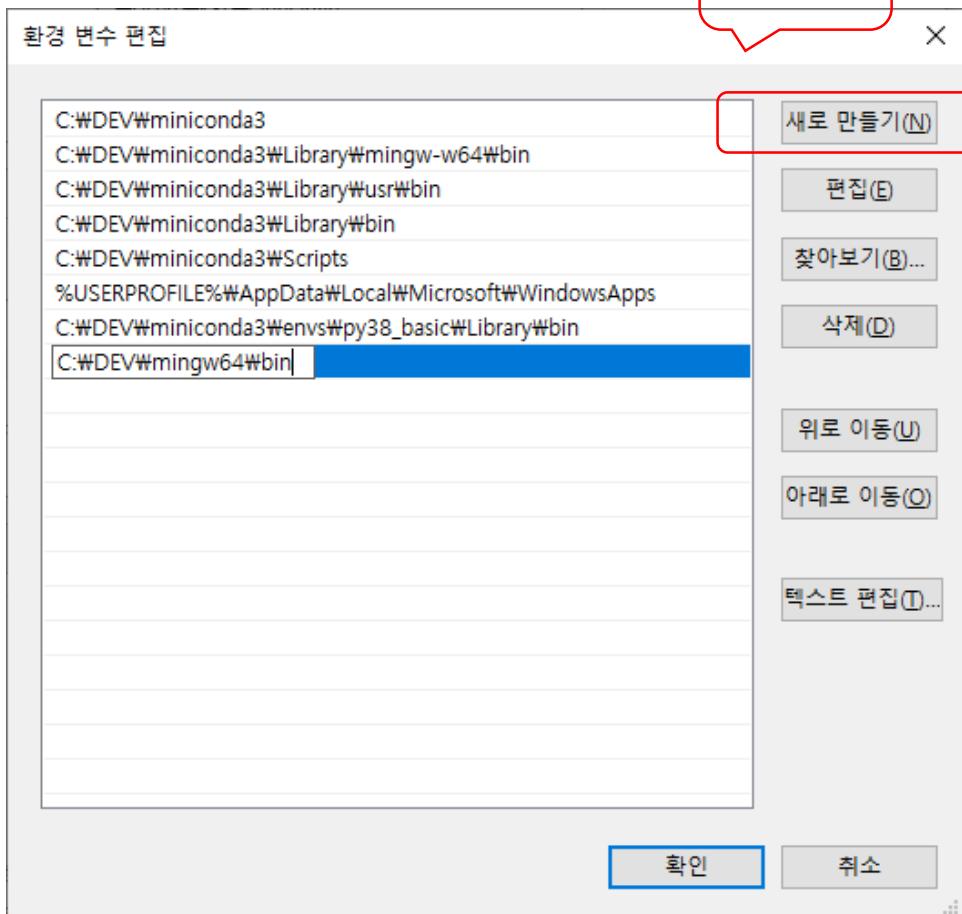


1. gcc(GNU Compiler Collection)

Windows 환경에서 GCC

3. 환경 변수 설정

- 사용자 변수의 Path에 C:\DEV\mingw64\bin 경로 추가

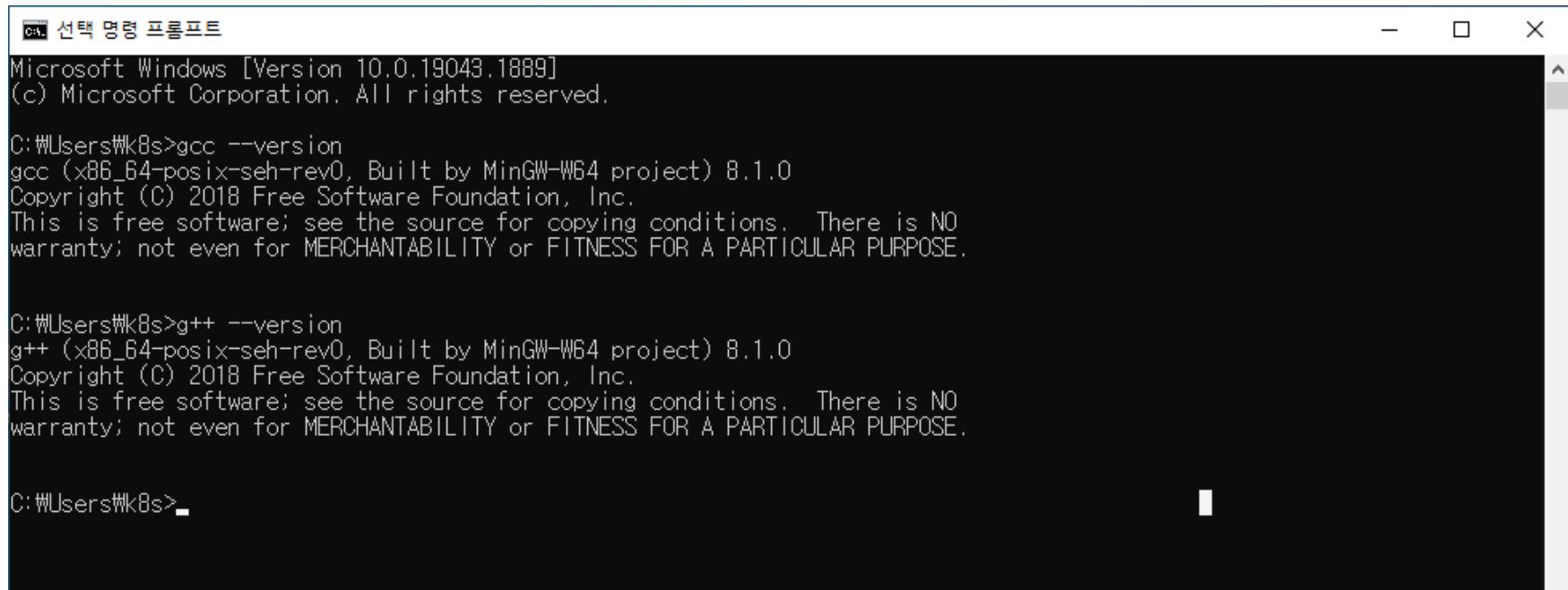


1. gcc(GNU Compiler Collection)

Windows 환경에서 GCC

4. 확인

- 명령 프롬프트를 실행해서 `gcc --version`을 입력하여 gcc가 제대로 설치 되었는지 확인합니다.



```
선택 명령 프롬프트
Microsoft Windows [Version 10.0.19043.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\k8s>gcc --version
gcc (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 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.

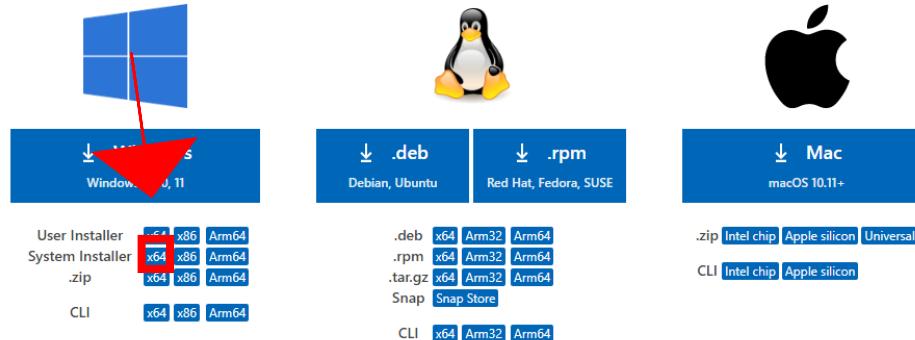
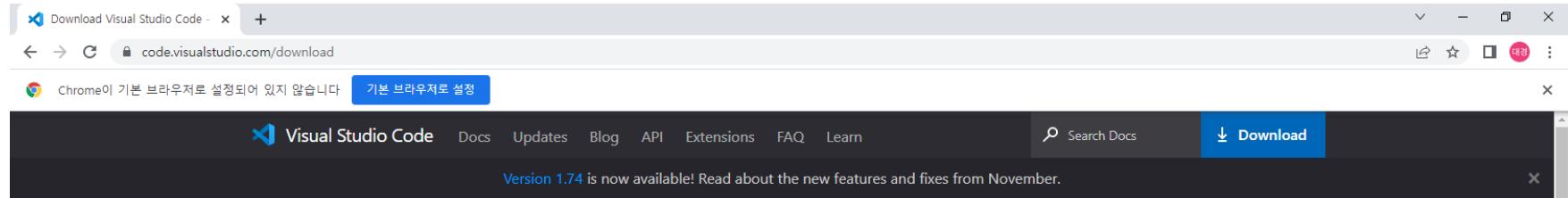
C:\Users\k8s>g++ --version
g++ (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

C:\Users\k8s>
```

2. Visual Studio Code

Visual Studio Code 설치

- <https://code.visualstudio.com/download>





2. Visual Studio Code

Visual Studio Code 설치

- **User Installer, System Installer, zip** 압축파일이 배포되고 있습니다.
- User Installer의 경우 다음 위치에 설치되며 유저 인터페이스의 디폴트 언어가 영어가 됩니다.

C:\Users\사용자이름\AppData\Local\Programs\Microsoft VS Code

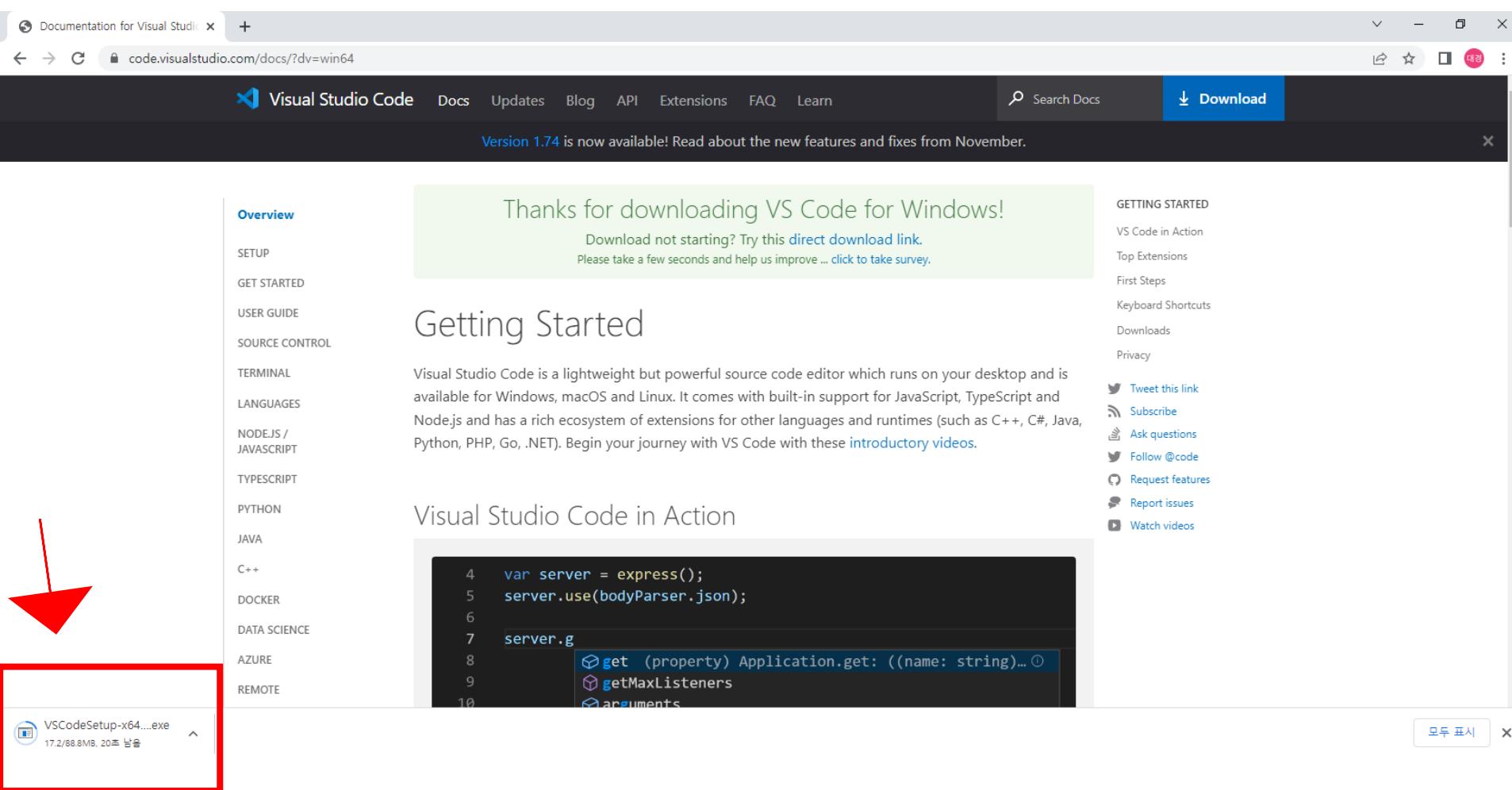
- System Installer의 경우에는 다음 위치에 설치되며 유저 인터페이스의 디폴트 언어가 영어가 됩니다.

C:\Program Files\Microsoft VS Code

2. Visual Studio Code

Visual Studio Code 설치

● System Installer x64를 설치



The screenshot shows the official Visual Studio Code documentation website at code.visualstudio.com/docs/?dv=win64. The page header includes a search bar, a 'Download' button, and a message about Version 1.74. The main content area features a 'Getting Started' section with a code editor preview showing JavaScript code and a 'VS Code in Action' section. A red arrow points from the bottom left towards the 'Download' button, and a red box highlights the download link in the footer.

Documentation for Visual Studio

code.visualstudio.com/docs/?dv=win64

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn Search Docs Download

Version 1.74 is now available! Read about the new features and fixes from November.

Overview

SETUP

GET STARTED

USER GUIDE

SOURCE CONTROL

TERMINAL

LANGUAGES

NODEJS / JAVASCRIPT

TYPESCRIPT

PYTHON

JAVA

C++

DOCKER

DATA SCIENCE

AZURE

REMOTE

Thanks for downloading VS Code for Windows!

Download not starting? Try this [direct download link](#).
Please take a few seconds and help us improve ... [click to take survey](#).

Getting Started

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET). Begin your journey with VS Code with these [introductory videos](#).

Visual Studio Code in Action

```
4 var server = express();
5 server.use(bodyParser.json());
6
7 server.g
8   ⚡ get (property) Application.get: ((name: string)... ⓘ
9   ⚡ getMaxListeners
10  ⚡ arguments
```

GETTING STARTED

VS Code in Action

Top Extensions

First Steps

Keyboard Shortcuts

Downloads

Privacy

[Tweet this link](#)

[Subscribe](#)

[Ask questions](#)

[Follow @code](#)

[Request features](#)

[Report issues](#)

[Watch videos](#)

VSCodeSetup-x64...exe

17.2/88.8MB, 20초 남음

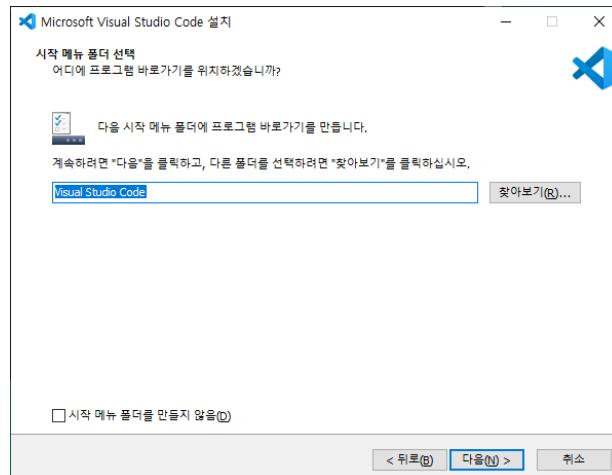
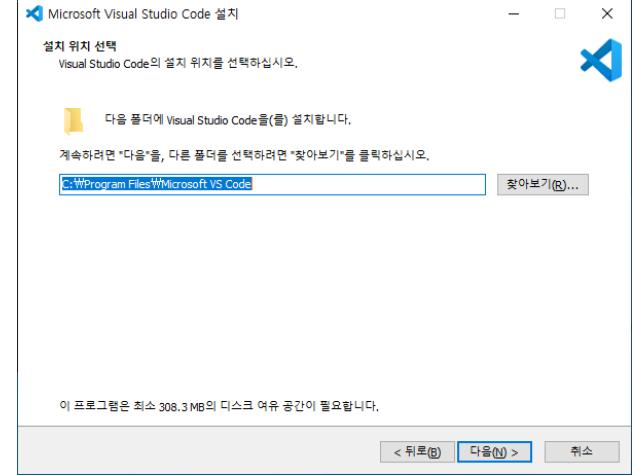
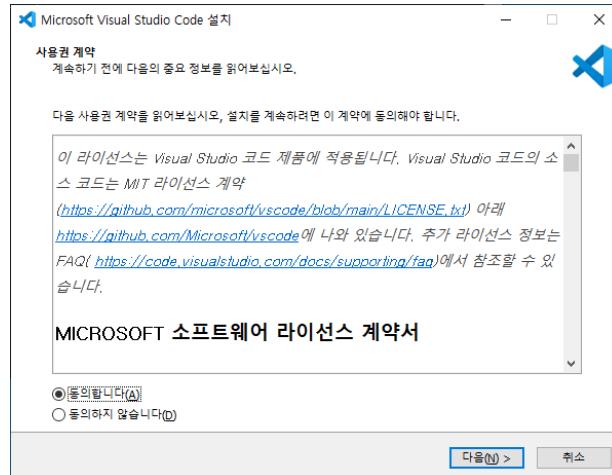
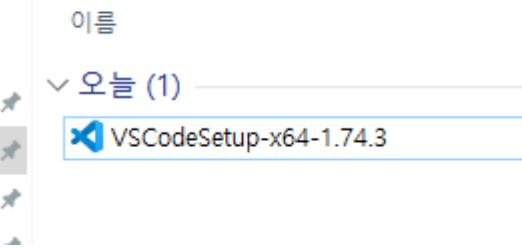
모두 표시

2. Visual Studio Code

Visual Studio Code 설치

● Visual Studio Code 설치

내 PC > 다운로드

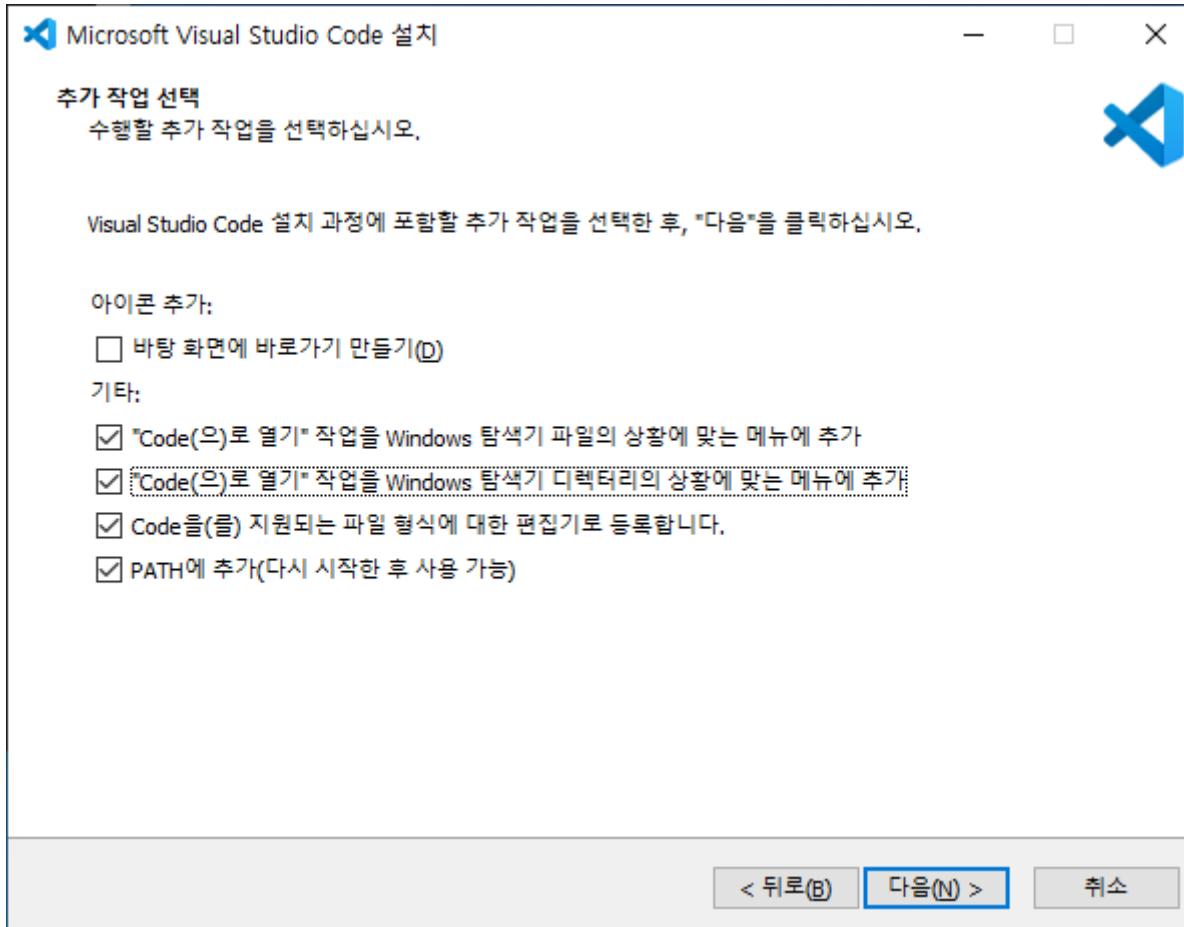




2. Visual Studio Code

Visual Studio Code 설치

● Visual Studio Code 설치

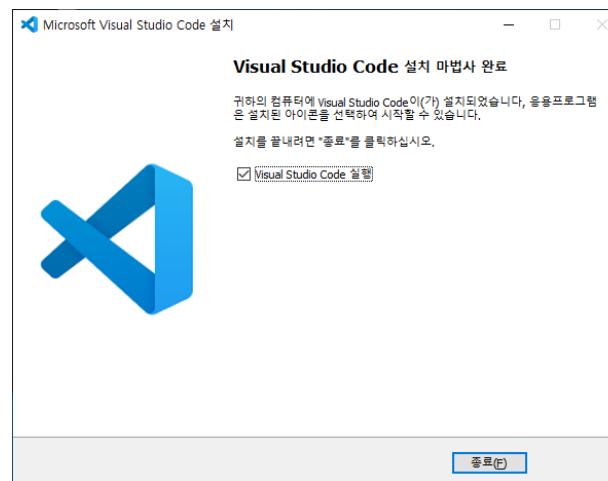
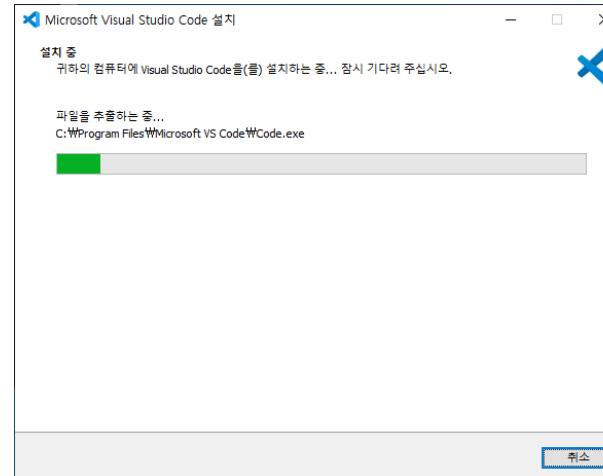
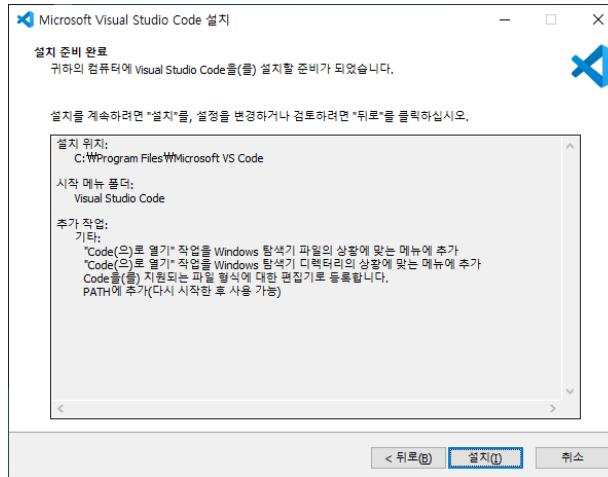




2. Visual Studio Code

Visual Studio Code 설치

● Visual Studio Code 설치

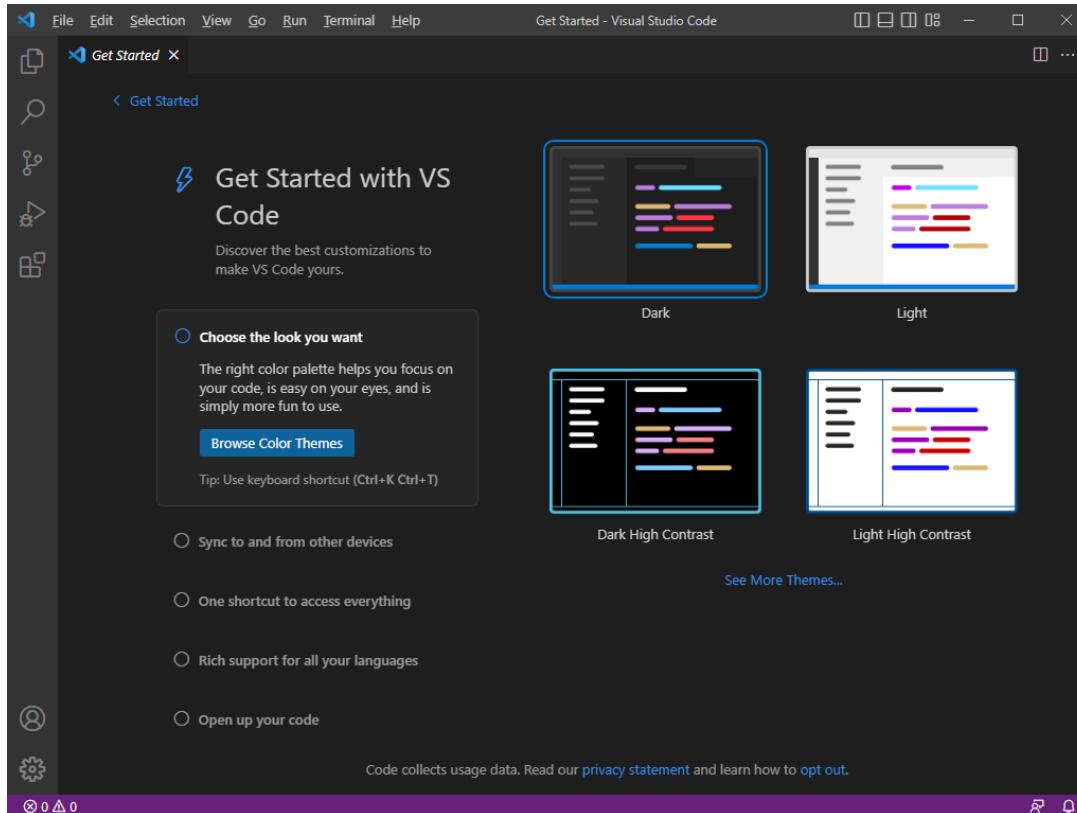




2. Visual Studio Code

Visual Studio Code 설치

● Visual Studio Code 설치

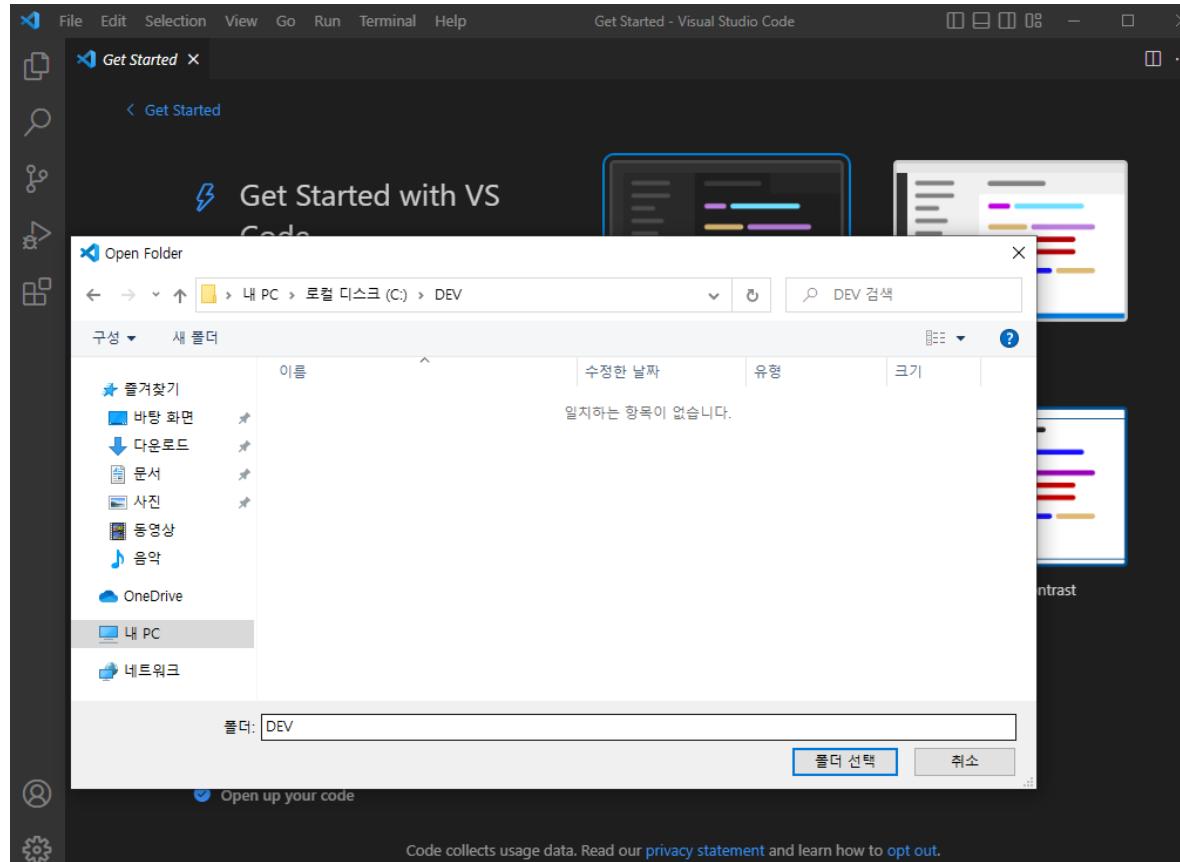




2. Visual Studio Code

Visual Studio Code 설치

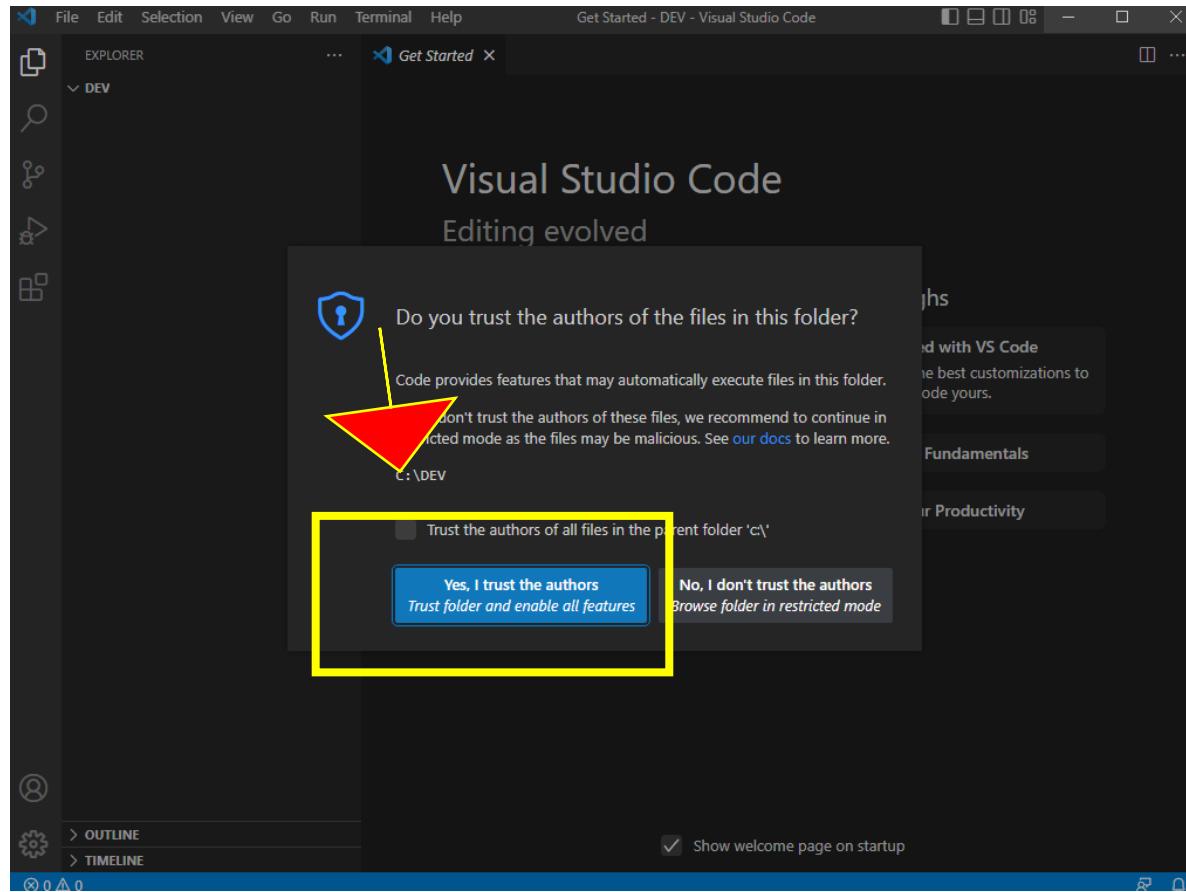
- Visual Studio Code를 실행하여 메뉴에서 File > Open Folder를 선택



2. Visual Studio Code

Visual Studio Code 설치

- Visual Studio Code를 실행하여 메뉴에서 File > Open Folder 를 선택

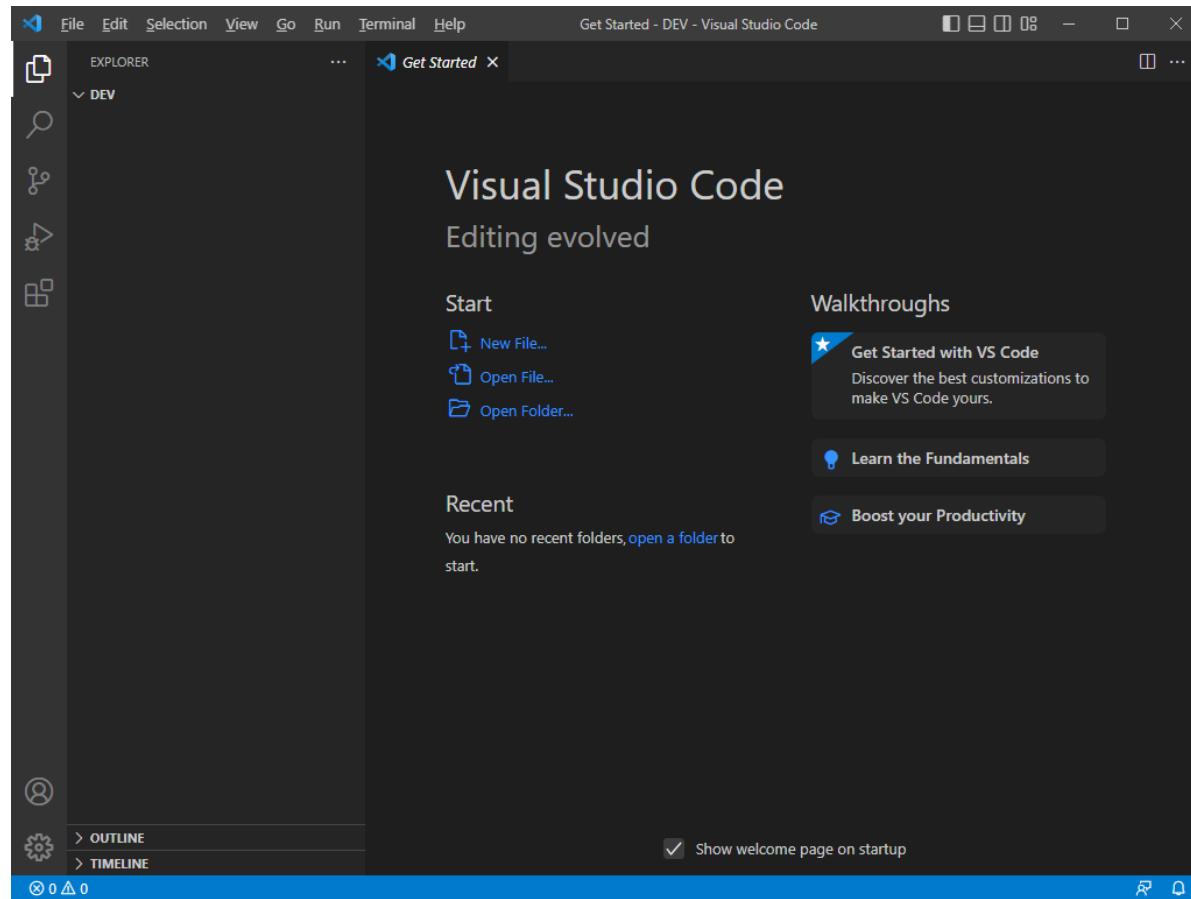




2. Visual Studio Code

Visual Studio Code 설치

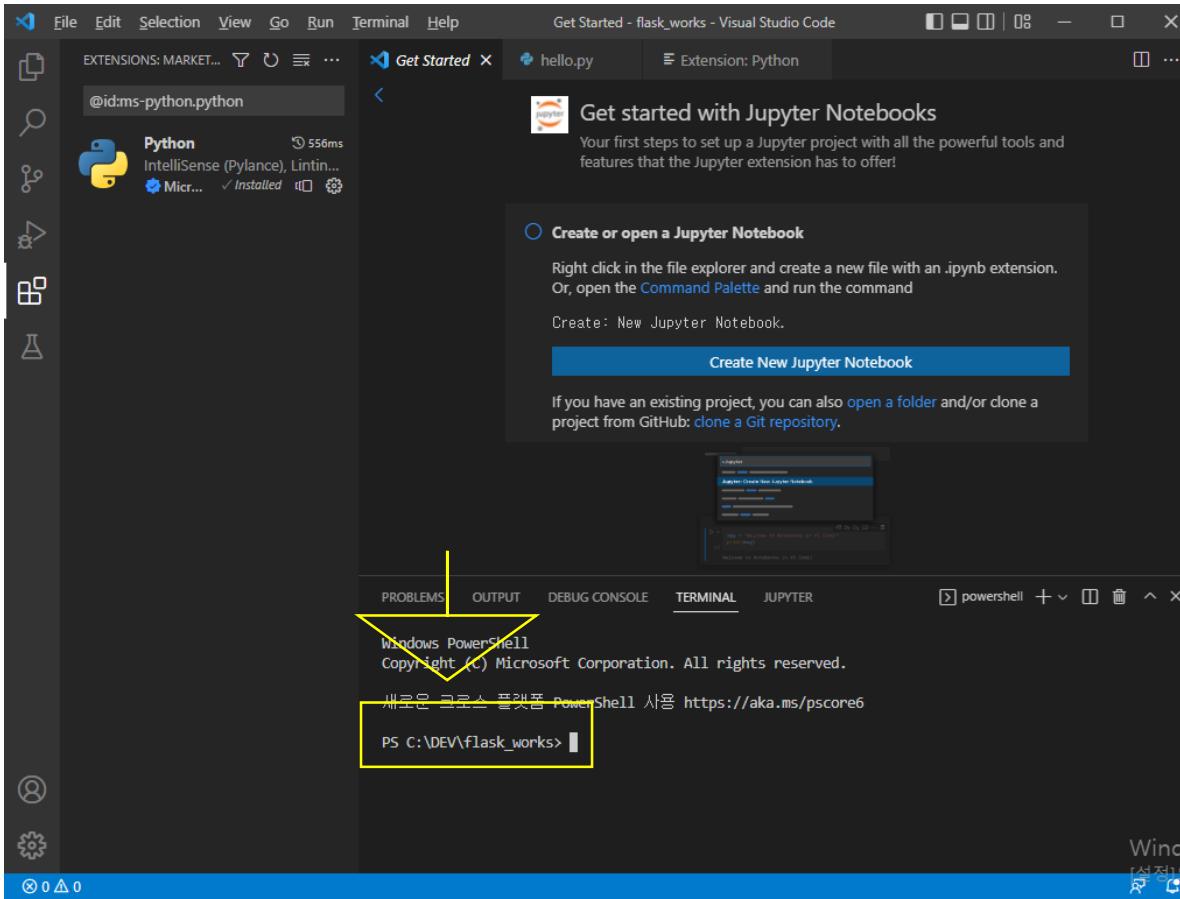
- 해당 폴더가 Visual Studio Code 왼쪽에 보이게 됩니다.



2. Visual Studio Code

Terminal 변경

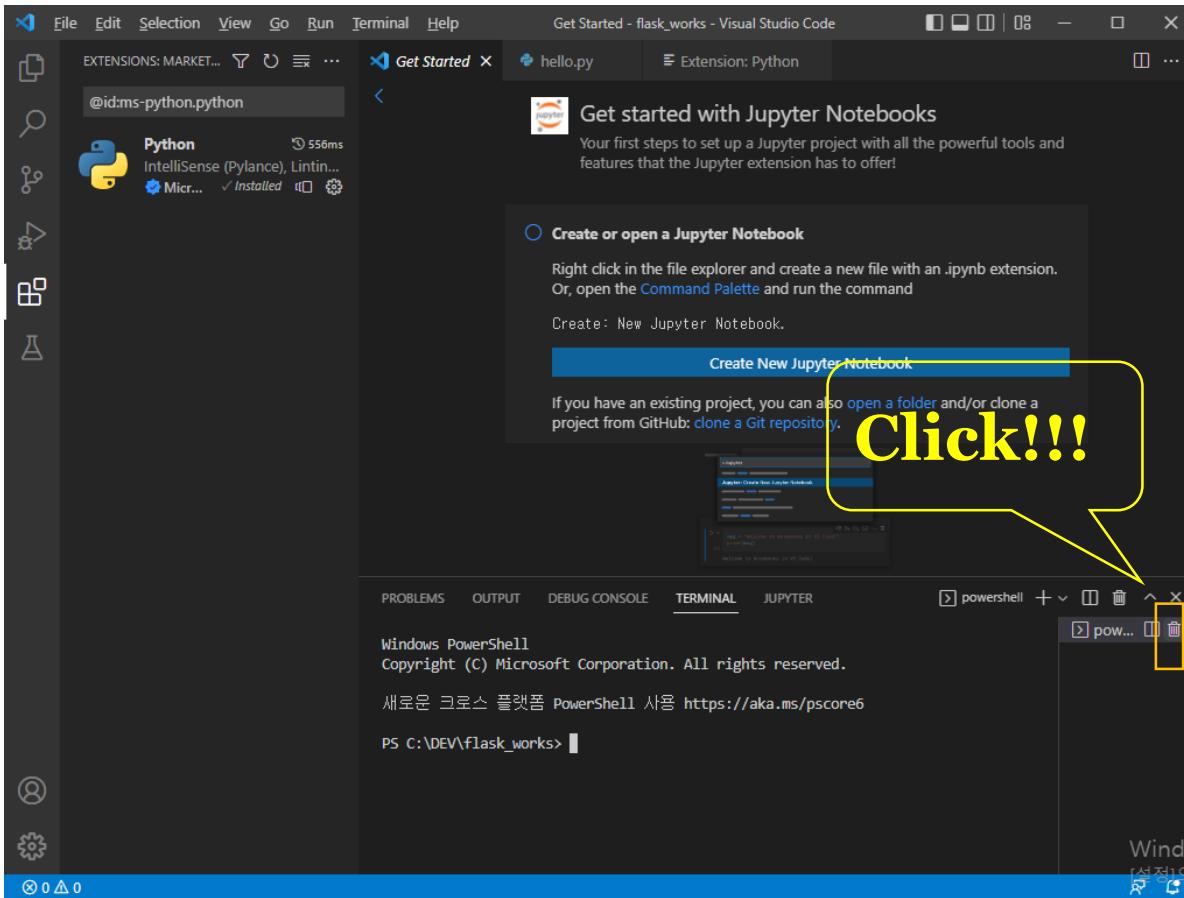
- 메뉴에서 View > Terminal을 선택합니다. 명령 프롬프트 (cmd.exe)를 사용하도록 변경해줘야 합니다.



2. Visual Studio Code

Terminal 변경

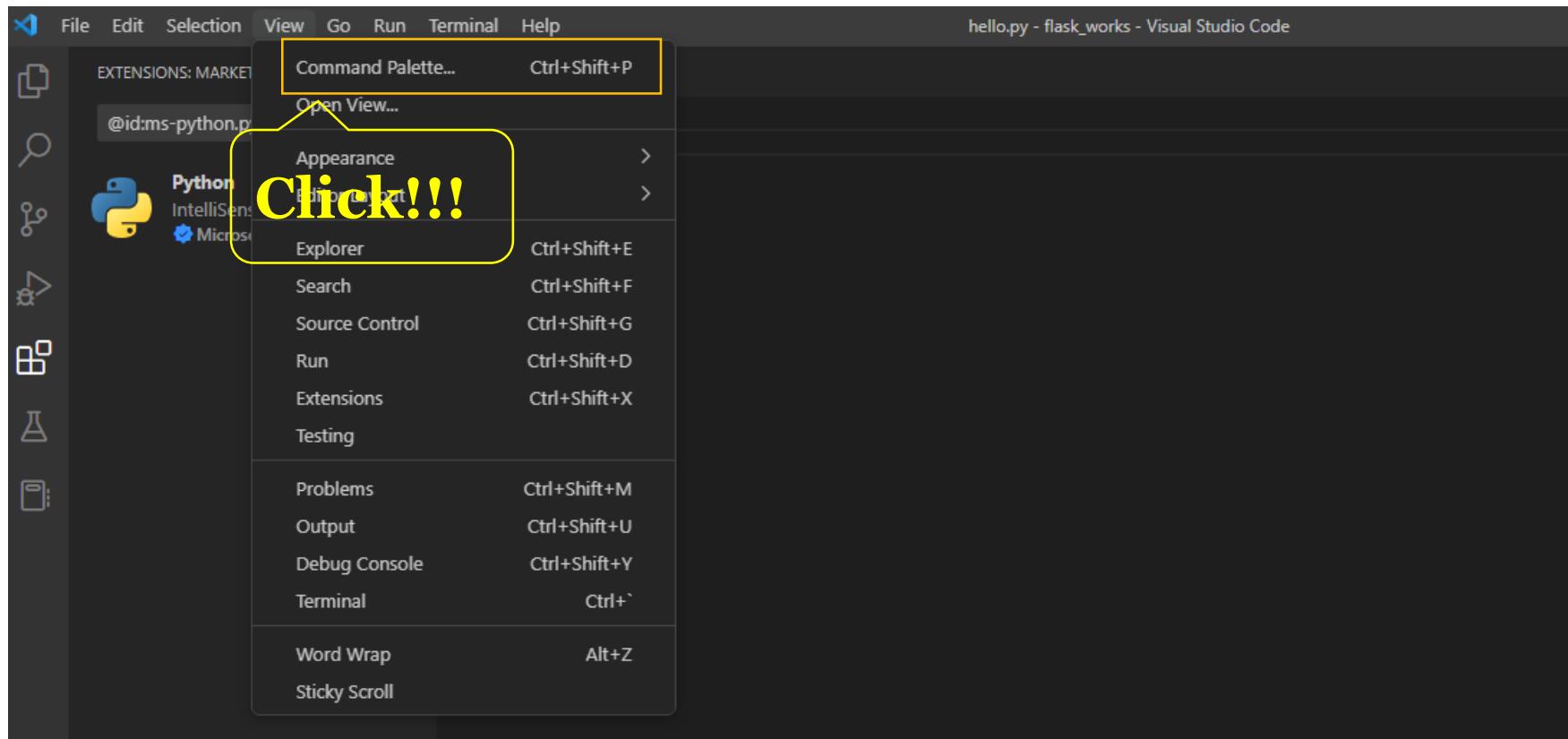
- 오른쪽에 보이는 쓰레기통 아이콘을 클릭



2. Visual Studio Code

Terminal 변경

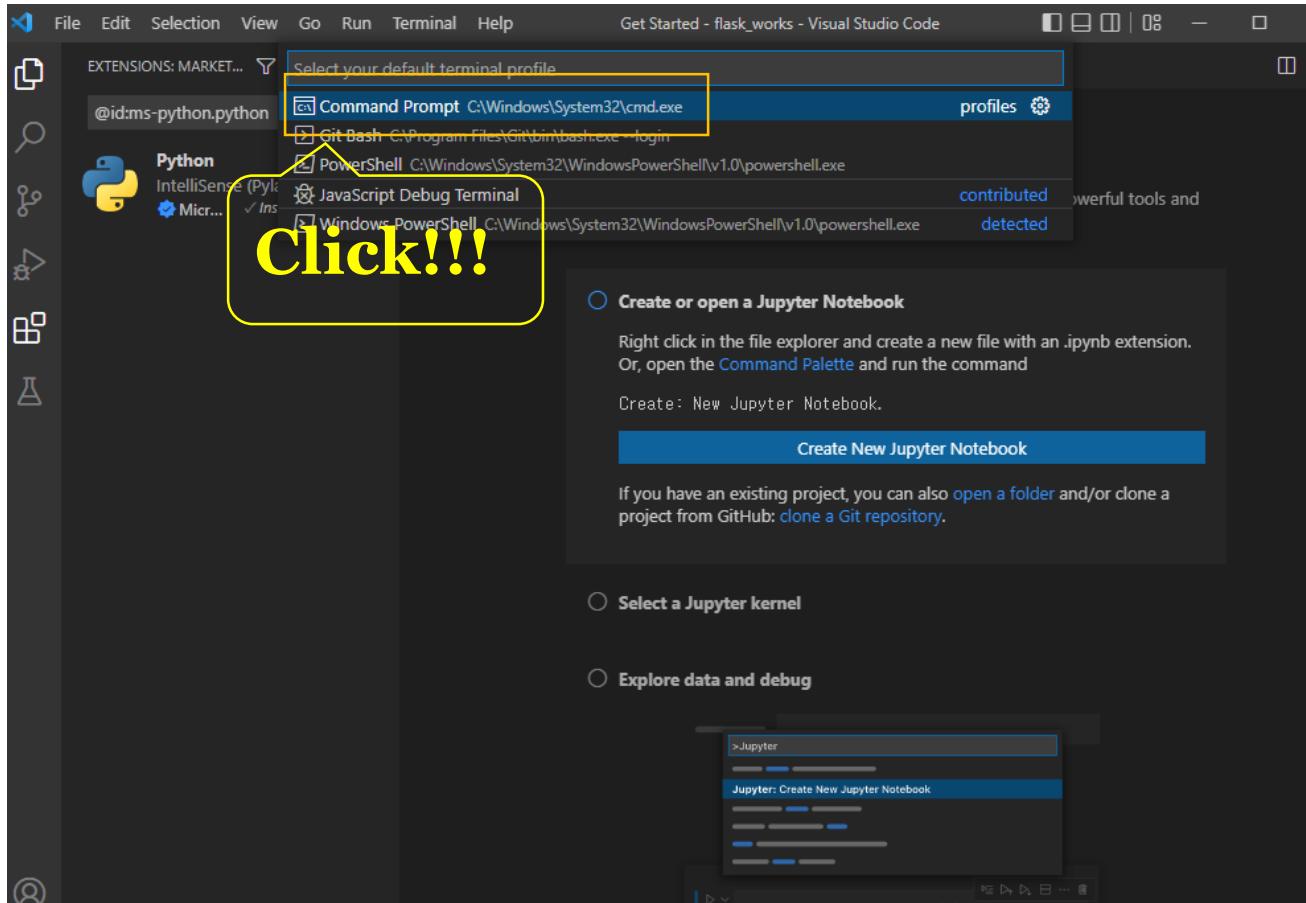
- Ctrl + Shift + P를 입력한 후, default profile을 입력하여 검색 되는 항목을 선택



2. Visual Studio Code

Terminal 변경

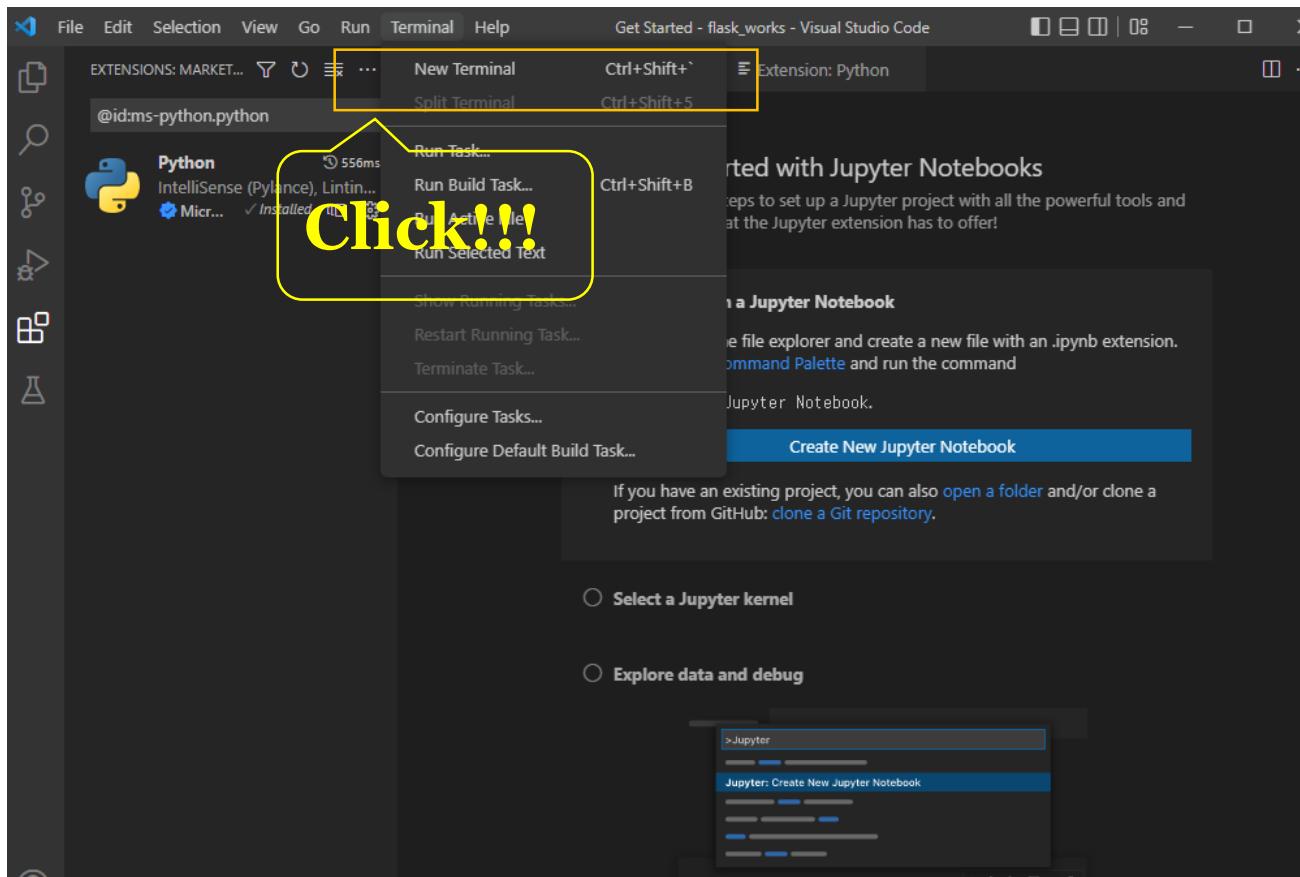
● Command Prompt를 선택



2. Visual Studio Code

Terminal 변경

- 메뉴에서 Terminal>New Terminal을 선택하면 명령 프롬프트가 실행



2. Visual Studio Code

Terminal 변경

- 명령 프롬프트가 실행

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Extensions Bar:** EXTENSIONS: MARKET..., Python (IntelliSense (Pylance), Linting... by Microsoft).
- Code Editor:** A file named "hello.py" is open, showing the number "1".
- Terminal Tab:** The "TERMINAL" tab is selected, showing the Windows command prompt output:

```
Microsoft Windows [Version 10.0.19043.1889]
(c) Microsoft Corporation. All rights reserved.

C:\DEV\flask_works>C:/DEV/miniconda3/Scripts/activate.bat

(base) C:\DEV\flask_works>
```
- Terminal Buttons:** cmd, +, - (minimize/maximize), X (close).



2. Visual Studio Code

Visual Studio Code에서 gcc 사용

- GCC를 설치했다면 VS Code의 터미널(**ctrl + j**)에서 명령어를 입력하여 gcc를 실행시킬 수 있습니다. 예)

C:\DEV>gcc --version

```
gcc (x86_64-posix-seh-revo, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 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.
```

C:\DEV>g++ --version

```
g++ (x86_64-posix-seh-revo, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 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.
```



3. gcc의 사용 gcc의 사용

gcc의 사용 방법

- gcc를 이용하여 소스 코드를 컴파일하기 위한 기본 구문은 다음과 같다.

gcc [소스 코드명]

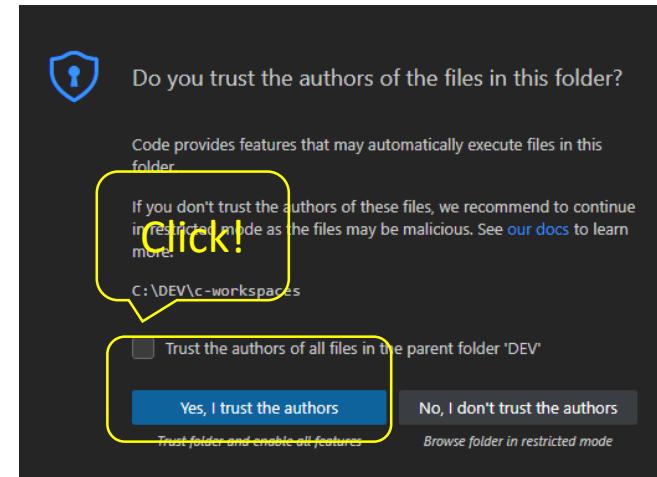
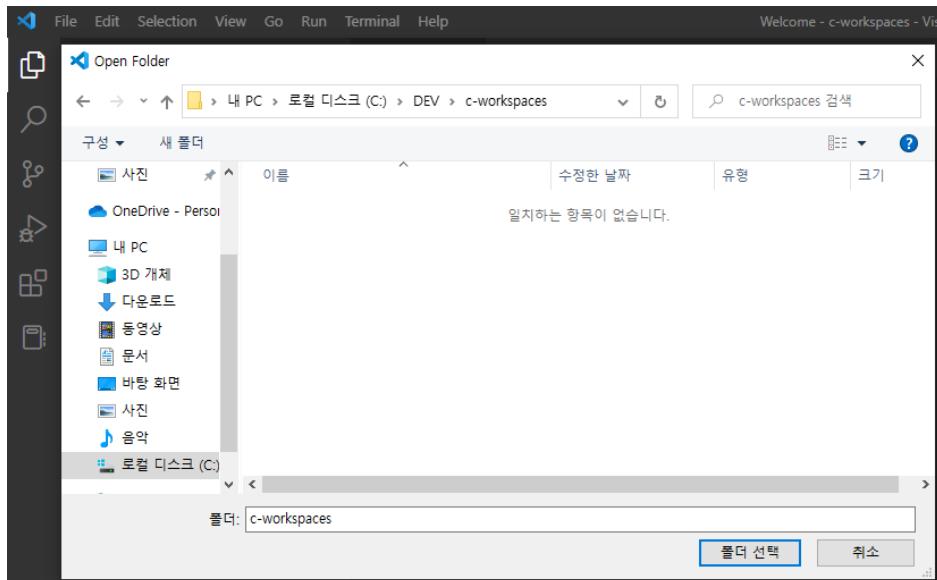
- 이 외에 최적화, 라이브러리 관련 설정, ANSI 호환 등과 관련된 다양한 옵션을 제공하고 있다.

옵션	세부 설명
-V	컴파일 각 단계의 상황을 시현
-E	전처리 단계까지만 수행하고 컴파일을 중단
-c	소스 파일을 컴파일만 하고 링크를 중단, 즉 오브젝트 파일만 생성
-o	타깃 프로그램의 이름을 지정, 지정하지 않았을 경우에는 디폴트값으로 a.out이 생성
-I	헤더 파일의 PATH 지정
-L	라이브러리 파일의 PATH 지정
-l	라이브러리 파일을 링크
-O	컴파일의 최적화 옵션
-static	정적 라이브러리 링크

3. gcc의 사용 gcc의 사용

Visual Studio Code에서 gcc 사용

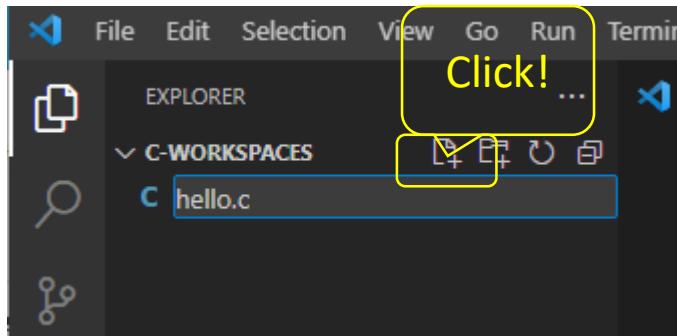
- 폴더 선택
- C:\DEV\c-workspaces



3. gcc의 사용 gcc의 사용

Visual Studio Code에서 gcc 사용

- 새 파일 추가
- hello.c 입력하고 엔터





3. gcc의 사용 gcc의 사용

Visual Studio Code에서 gcc 사용

● 리눅스

```
#include <stdio.h>
int main( void ){
    printf("HELLO WORLD !! \n");
    return 0;
}
```

[컴파일] gcc hello.c

[실행] ./a.out

● 윈도

```
#include <stdio.h>
int WinMain(){
    printf("HELLO WORLD !! \n");
    return 0;
}
```

[컴파일] C:\DEV\c-workspaces>gcc hello.c

C:\DEV\c-workspaces>dir/w/p

C 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: BEDo-C858

C:\DEV\c-workspaces 디렉터리

```
[.] [..] a.exe  hello.c
2개 파일      54,781 바이트
2개 디렉터리 16,860,844,032 바이트 남음
```

[실행] C:\DEV\c-workspaces>a
HELLO WORLD !!

```
... Welcome C hello.c ...
C hello.c
1 #include <stdio.h>
2 int WinMain(){
3     printf("HELLO WORLD !! \n");
4     return 0;
5 }
```



3. gcc의 사용 gcc의 사용

Visual Studio Code에서 gcc 사용

● 명시적 실행 파일이름

```
C:\DEV\c-workspaces>gcc -o hello hello.c
```

```
C:\DEV\c-workspaces>dir/w/p
```

C 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: BEDo-C858

C:\DEV\c-workspaces 디렉터리

```
[.]      [...]    a.exe    hello.c    hello.exe  
3개 파일          109,476 바이트  
2개 디렉터리  16,860,426,240 바이트 남음
```

```
C:\DEV\c-workspaces>hello
```

HELLO WORLD !!

```
C:\DEV\c-workspaces>
```



3. gcc의 사용 gcc의 사용

여러 파일 컴파일하기 방법-1

```
C:\DEV\c-workspaces>gcc -c fun1.c  
C:\DEV\c-workspaces>gcc -c fun2.c  
C:\DEV\c-workspaces>gcc -c main.c  
C:\DEV\c-workspaces>gcc -o main main.o fun1.o fun2.o  
C:\DEV\c-workspaces>main  
fun1 !!  
fun2 !!  
C:\DEV\c-workspaces>
```

Sample.h

```
#include <stdio.h>  
void fun1();  
void fun2();
```

main.c

```
#include "Sample.h"  
int WinMain(){  
    fun1();  
    fun2();  
    return 0;  
}
```

fun1.c

```
#include <stdio.h>  
void fun1(){  
    printf("fun1 !! \n");  
}
```

fun2.c

```
#include <stdio.h>  
void fun2(){  
    printf("fun2 !! \n");  
}
```



3. gcc의 사용 gcc의 사용

여러 파일 컴파일하기 방법-2

- 먼저, 실행파일과 오브젝트 파일을 모두 삭제한다.

C:\DEV\c-workspaces>dir/w/p

C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: BEDo-C858

C:\DEV\c-workspaces 디렉터리

```
[.]      [..]      a.exe      fun1.c      fun1.o      fun2.c      fun2.o      hello.c      hello.exe      main.c      main.exe      main.o      Sample.h  
11개 파일          168,032 바이트  
2개 디렉터리 16,845,598,720 바이트 남음
```

C:\DEV\c-workspaces>del *.exe *.o

C:\DEV\c-workspaces>dir/w/p

C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: BEDo-C858

C:\DEV\c-workspaces 디렉터리

```
[.]      [..]      fun1.c      fun2.c      hello.c      main.c      Sample.h  
5개 파일          335 바이트  
2개 디렉터리 16,849,702,912 바이트 남음
```

C:\DEV\c-workspaces>



3. gcc의 사용 gcc의 사용

여러 파일 컴파일하기 방법-2

```
C:\DEV\c-workspaces>gcc -o main main.c fun1.c fun2.c
```

```
C:\DEV\c-workspaces>dir/w/p
```

C 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: BED0-C858

```
C:\DEV\c-workspaces 디렉터리
```

```
[.]    [...] fun1.c   fun2.c   hello.c  main.c  main.exe Sample.h  
      6개 파일        56,154 바이트  
      2개 디렉터리 16,849,620,992 바이트 남음
```

```
C:\DEV\c-workspaces>main
```

fun1 !!

fun2 !!

```
C:\DEV\c-workspaces>
```



3. gcc의 사용 gcc의 사용

여러 파일 컴파일하기 방법-2

- 먼저, 실행파일과 오브젝트 파일을 모두 삭제한다.

```
C:\DEV\c-workspaces>dir/w/p
```

C 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: BED0-C858

```
C:\DEV\c-workspaces 디렉터리
```

```
[.] [...] fun1.c fun2.c hello.c main.c main.exe Sample.h  
      6개 파일      56,154 바이트  
      2개 디렉터리 16,849,334,272 바이트 남음
```

```
C:\DEV\c-workspaces>del *.exe *.o
```

```
C:\DEV\c-workspaces>dir/w/p
```

C 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: BED0-C858

```
C:\DEV\c-workspaces 디렉터리
```

```
[.] [...] fun1.c fun2.c hello.c main.c Sample.h  
      5개 파일      335 바이트  
      2개 디렉터리 16,849,391,616 바이트 남음
```

```
C:\DEV\c-workspaces>
```



4. make

1. make의 개요

- 컴파일을 자동화해주는 명령어 생성 프로그램
 - make는 하나 이상의 소스 코드 파일로 이루어진 프로그램을 컴파일하고 재컴파일 시 효율을 높이기 위한 프로그램으로서, make 파일이라는 애플리케이션 구성 방법이 기록된 텍스트 파일을 사용한다.
 - make는 make 파일상의 조건으로부터 타깃 프로그램을 만들어 낼 때 변경된 파일의 유무와 대상을 판별하고 재컴파일 필요성에 따라 타깃 프로그램을 생성한다.
 - 최종 목표 파일(target file)의 빌드(Build) 과정을 정의해두면 필요한 작업만 수행하여 목표 파일을 생성
 - 종속된 파일들의 상호 의존 관계를 명시한 목록 파일(description file)을 사용하여 목표 파일을 생성
 - 원시 파일의 일부가 수정되면 목표 파일을 새롭게 생성하기 위해 다시 컴파일할 파일을 자동으로 판단, 필요한 명령어만 이용해 다시 컴파일하는 지능적인 유틸리티
 - 다중 모듈 프로그램의 재사용성을 유지하고 프로그램 수정에 대한 재번역을 최소화



4.make

1. make의 개요

- Make 유ти리티의 사용

```
make [-f <목록 파일>] [<목표 파일>]
```

- hello를 생성하기 위한 Makefile
- gedit Makefile or gedit Makefile_Test
예) make 혹은 make -f Makefile_Test

hello: hello.c

gcc -o hello hello.c



4.make

1. make의 개요

- 목록 파일이 **makefile** 혹은 **Makefile**라면 **-f** 옵션을 사용해 목록 파일을 명시할 필요 없음

옵션	세부 설명
-f	make 파일을 지정, 지정하지 않았을 경우에는 디폴트명인 Makefile 등으로 지정됨
-n	make가 실행하는 명령을 출력, 실제 실행되지는 않음
-s	make의 실행 명령을 출력하지 않고 실행 -n 옵션의 반대
-d	make 실행 시 디버깅 정보도 출력



4.make

1. make의 개요

- 매크로와 레이블
 - 매크로: 특정한 코드를 단순하게 표현하는 기법
 - 레이블: 부품 파일이 없는 의존행의 목표 파일 이름을 의미하며, 명령행의 성격을 대표하는 이름

TARGET := hello

OBJECT := hello.o

SOURCE := hello.c

\$(TARGET): \$(SOURCE)

gcc -o \$(TARGET) \$(SOURCE)

clean:

rm -f \$(TARGET) \$(OBJECT)



4.make

2. make 사용해 컴파일

● Windows 10에서 make 사용하기

- Windows에서 GNU make utility를 사용하여 설치하는 방법은 아래와 같다.

<http://gnuwin32.sourceforge.net/packages/make.htm>

If you download the [Setup program](#) of the package, any requirements for running applications, such as dynamic link libraries (DLL's) from the dependencies as listed below under Requirements, are already included. If you download the package as Zip files, then you must download and install the [dependencies zip file](#) yourself. Developer files (header files and libraries) from other packages are however not included; so if you wish to develop your own applications, you must separately install the required packages.

Description	Download	Size	Last change	Md5sum
• Complete package, except sources	Setup	3384653	25 November 2006	8ae51379d1f3eeff8360d14e674f17d6d
• Sources	Setup	1252948	25 November 2006	b896c02e3d581040ba1ad65024bbf2cd
• Binaries	Zip	495645	25 November 2006	3521948bc27a31d1ade0dcb23be16d49
• Dependencies	Zip	708206	25 November 2006	d370415aa924fa023411c4099ef84563
• Documentation	Zip	2470575	25 November 2006	43a07e449d4bab3eb3f31821640ecab7
• Sources	Zip	2094753	25 November 2006	8bed4cf17c5206f8094f9c96779be663

You can also download the files from the GnuWin32 [files page](#).

You can [monitor](#) new releases of the port of this package.



4.make

2. make 사용해 컴파일

● Windows 10에서 make 사용하기

- 아래와 같은 페이지가 나오며, 잠시 대기하면 다운로드가 된다.

The screenshot shows the SourceForge website interface. At the top, there's a navigation bar with links for "Open Source Software", "Business Software", and "Resources". On the right side of the header are links for "Help", "Create", "Join", and "Login". Below the header, there's a search bar with the placeholder "Search for software or solutions" and a magnifying glass icon. A promotional banner for "BigPicture" tools (Gantt and Agile) is displayed, along with a "DISCOVER" button. The main content area shows the download page for "GnuWin". The page title is "GnuWin" and it describes the software as "Provides native Win32 open source ports and utilities". It credits "Brought to you by: gnuwin32". A message says "Your download will start shortly..." with a green "1" in a circle next to it. Below this are buttons for "Get Updates", "Share This", and "Problems Downloading?". A note at the bottom states "make-3.81.exe | Scanned for malware ✓". To the right, there's a sidebar titled "Mirror Provided by" featuring the logo of JAIST (Japan Advanced Institute of Science and Technology). A link below the logo says "Learn more about Japan Advanced Institute of Science and Technology". At the bottom left, there's a section titled "Other Useful Business Software".

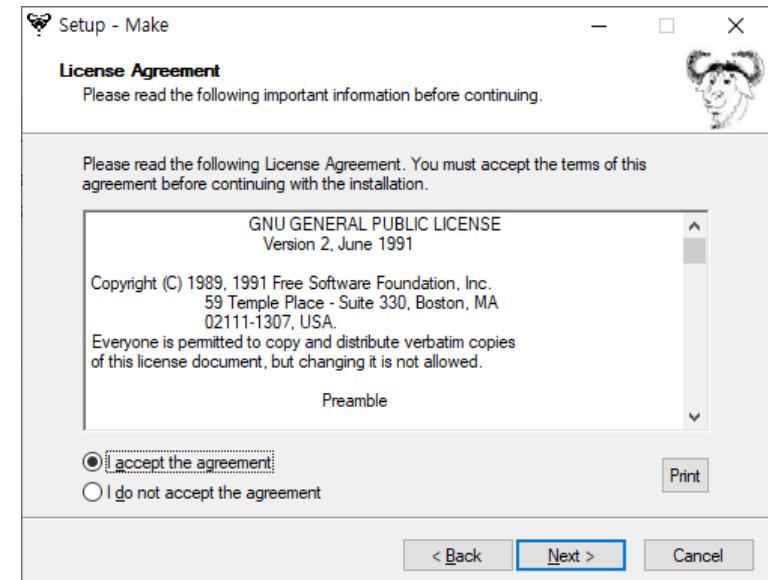
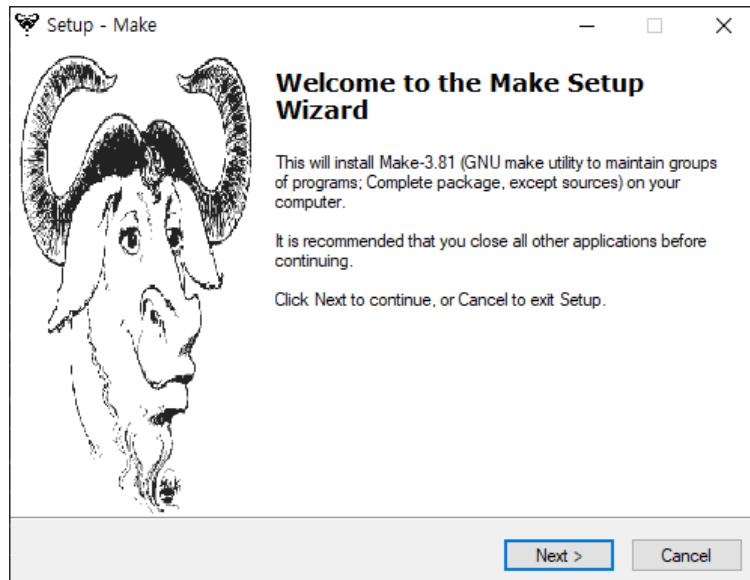


4.make

2. make 사용해 컴파일

- Windows 10에서 make 사용하기
- 다운로드된 실행파일(.exe)를 실행

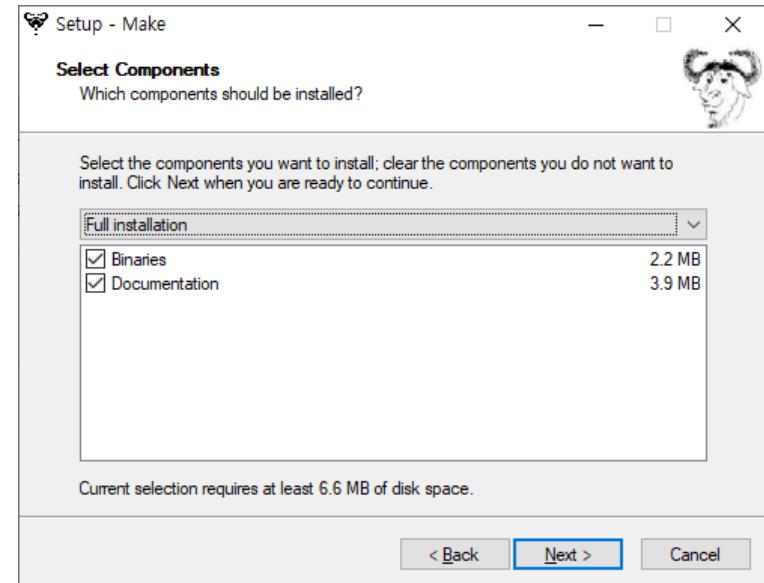
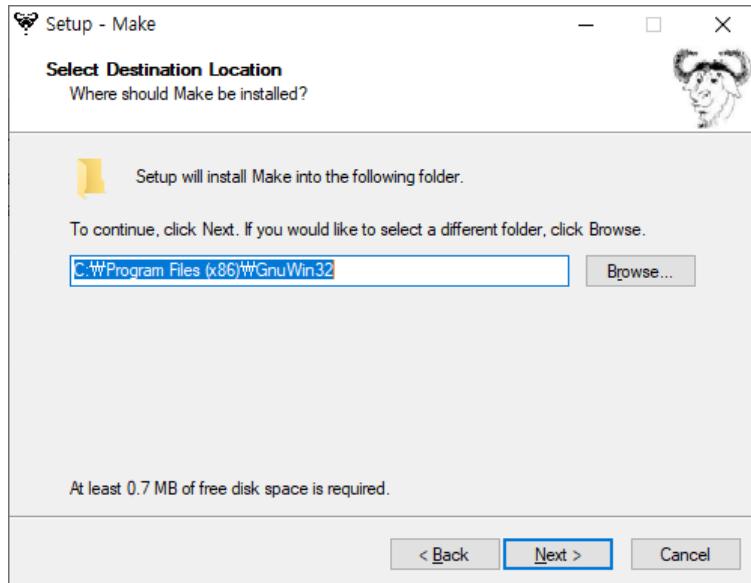
내 PC > 다운로드 >





2. make 사용해 컴파일

- Windows 10에서 make 사용하기
- 다운로드된 실행파일(.exe)를 실행

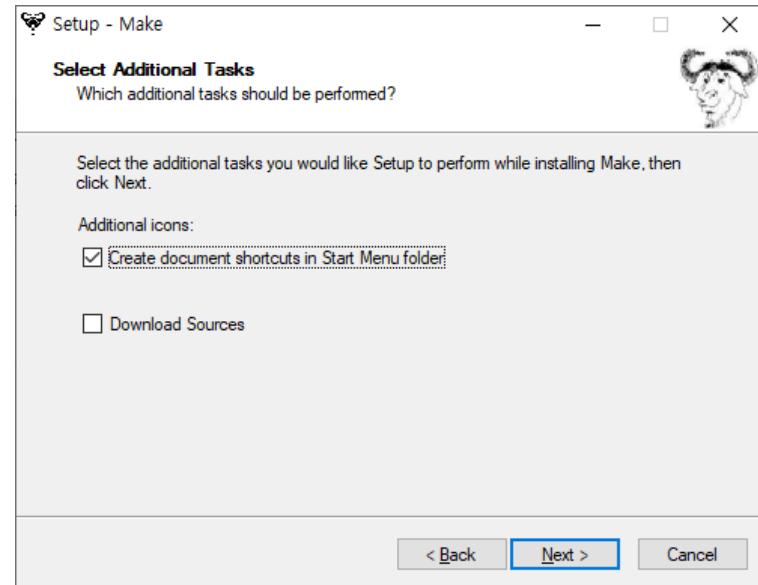
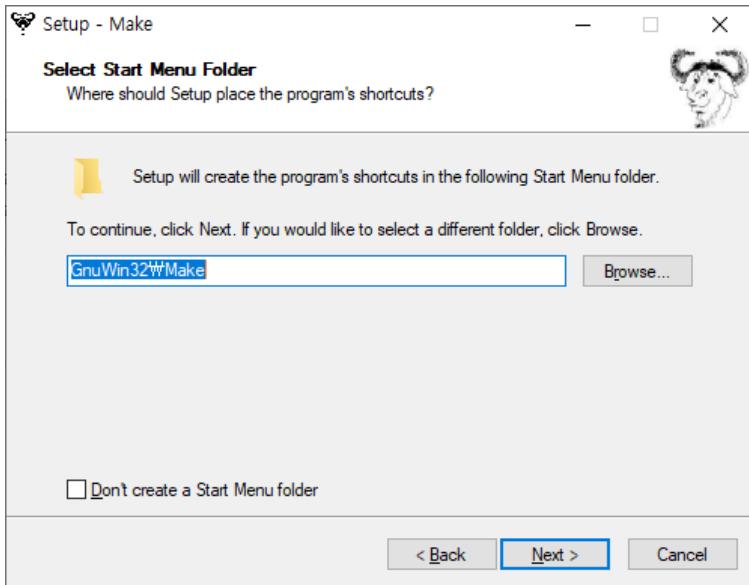




4.make

2. make 사용해 컴파일

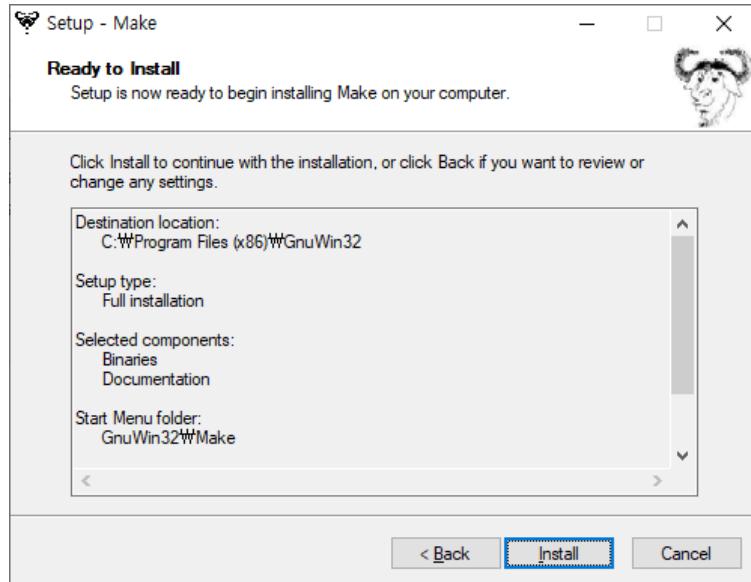
- Windows 10에서 make 사용하기
- 다운로드된 실행파일(.exe)를 실행





2. make 사용해 컴파일

- Windows 10에서 make 사용하기
 - 다운로드된 실행파일(.exe)를 실행

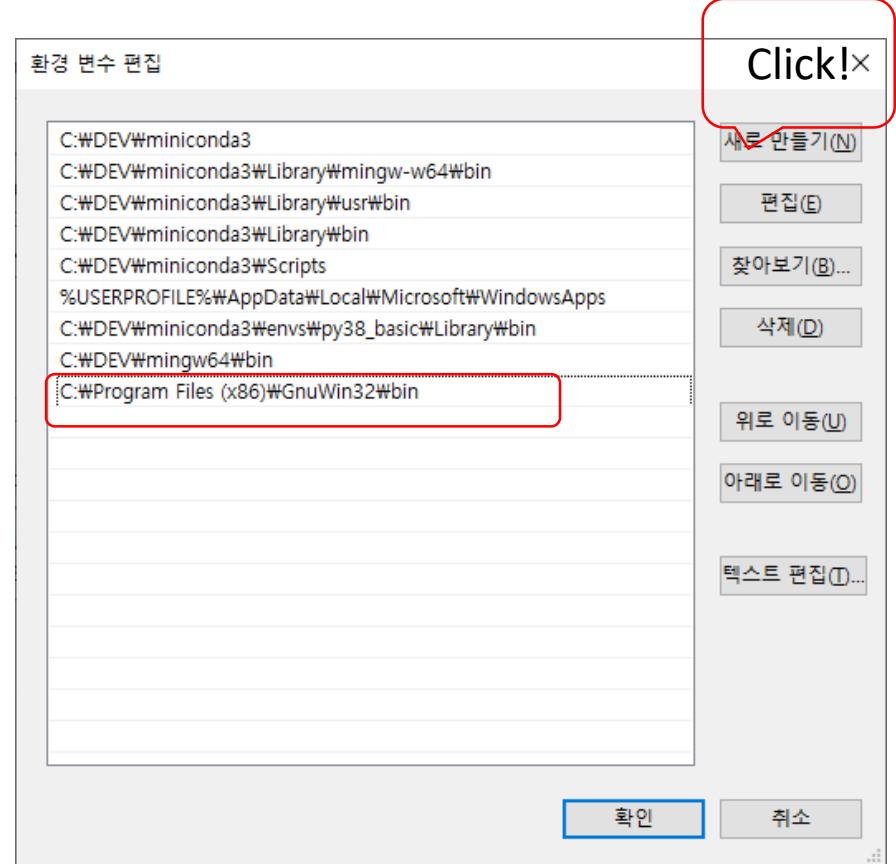
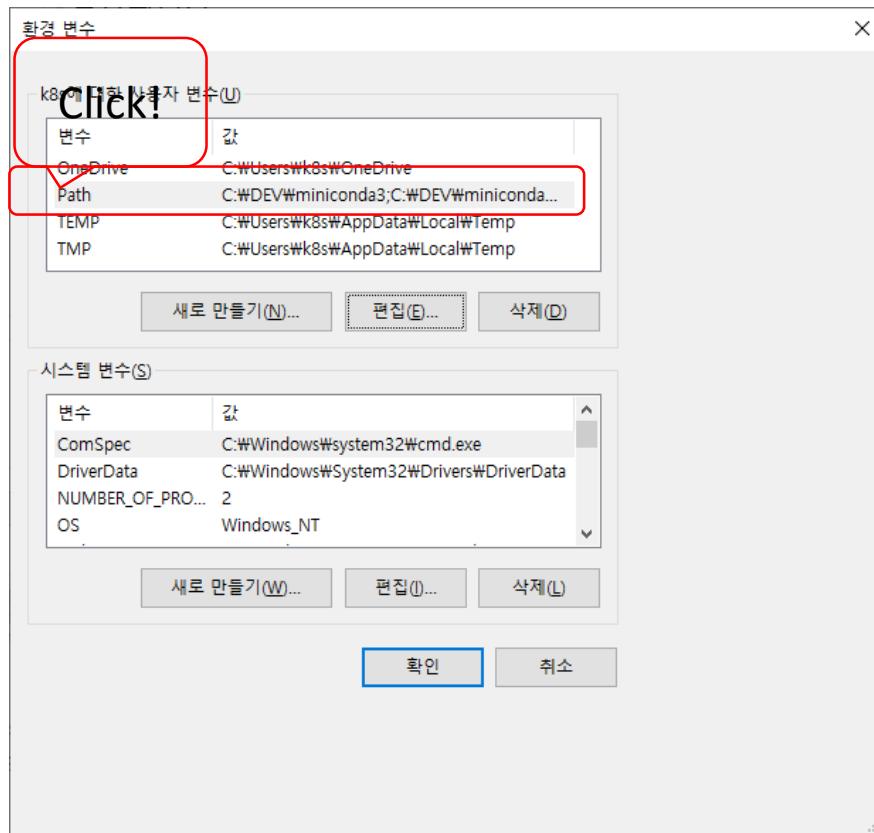


4.make

2. make 사용해 컴파일

● Windows 10에서 make 사용하기

- 사용자 환경 변수 편집 > C:\Program Files (x86)\GnuWin32\bin



4.make

2. make 사용해 컴파일

- Windows 10에서 make 사용하기
- 설치 확인

```
■ Anaconda Prompt (miniconda3)

C:\Users\k8s>make -v
GNU Make 3.81
Copyright (C) 2006 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.

This program built for i386-pc-mingw32

C:\Users\k8s>
```

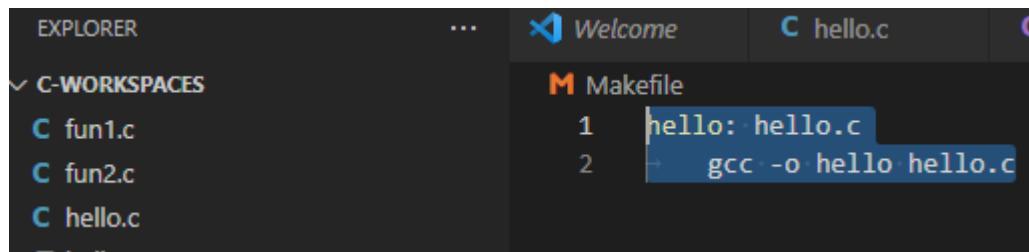
4.make

2. make 사용해 컴파일-

- Makefile 이름 사용 시

hello: hello.c

gcc -o hello hello.c



The screenshot shows the VS Code interface with the Explorer sidebar open, displaying three C workspace files: fun1.c, fun2.c, and hello.c. The main editor area shows a Makefile with the following content:

```
hello: hello.c
gcc -o hello hello.c
```

The second line, "gcc -o hello hello.c", is highlighted with a blue selection bar.



4.make

2. make 사용해 컴파일-

- Makefile 이름 사용 시

C:\DEV\c-workspaces>dir/w/p

C 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: BED0-C858

C:\DEV\c-workspaces 디렉터리

```
[.] [...] fun1.c fun2.c hello.c main.c Makefile Sample.h  
      6개 파일          335 바이트  
      2개 디렉터리 16,826,281,984 바이트 남음
```

C:\DEV\c-workspaces>make

gcc -o hello hello.c

C:\DEV\c-workspaces>dir/w/p

C 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: BED0-C858

C:\DEV\c-workspaces 디렉터리

```
[.] [...] fun1.c fun2.c hello.c hello.exe main.c Makefile Sample.h  
      7개 파일          55,067 바이트  
      2개 디렉터리 16,826,028,032 바이트 남음
```

C:\DEV\c-workspaces>hello

HELLO WORLD !!

C:\DEV\c-workspaces>

4.make

2. make 사용해 컴파일-

- Makefile이란 이름대신 다른 이름(makefile_Hello)을 사용했을 시

```
C:\DEV\c-workspaces>del *.o *.exe
```

```
C:\DEV\c-workspaces>dir/w/p
```

C 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: BEDo-C858

C:\DEV\c-workspaces 디렉터리

```
[.]  [..]  fun1.c  fun2.c  hello.c  main.c  Makefile  Sample.h  
      6개 파일          372 바이트  
      2개 디렉터리  16,825,733,120 바이트 남음
```

```
C:\DEV\c-workspaces>
```

The screenshot shows a dark-themed code editor interface. The Explorer pane on the left lists files and folders: C-WORKSPACES, fun1.c, fun2.c, hello.c, main.c, Makefile, makefile_Hello, and Sample.h. The Editor pane on the right displays a file named makefile_Hello with the following content:

```
1 hello: hello.c  
2     gcc -o hello hello.c
```



4.make

2. make 사용해 컴파일-

- Makefile이란 이름대신 다른 이름(makefile_Hello)을 사용했을 시

```
C:\DEV\c-workspaces>make -f makefile_Hello  
gcc -o hello hello.c
```

```
C:\DEV\c-workspaces>dir/w/p  
C 드라이브의 볼륨에는 이름이 없습니다.  
볼륨 일련 번호: BED0-C858
```

```
C:\DEV\c-workspaces 디렉터리  
[.]      [..]      fun1.c      fun2.c      hello.c      hello.exe      main.c      Makefile      makefile_Hello  
Sample.h  
8개 파일      55,104 바이트  
2개 디렉터리 16,823,992,320 바이트 남음
```

```
C:\DEV\c-workspaces>hello  
HELLO WORLD !!
```

```
C:\DEV\c-workspaces>
```



4.make

Quiz

- 다음 코드를 작성한 후, gcc 컴파일과 make 컴파일을 해 보시오.

input.c

```
#include <stdio.h>
int input()
{
    int num;
    printf("Enter a number : ");
    scanf("%d", &num);
    return num;
}
```

output.c

```
#include <stdio.h>
int output(int num)
{
    int i;
    for(i=0; i<num; i++)
        printf("Hello Linux\n");
    return 0;
}
```



4.make

Quiz

- 다음 코드를 작성한 후, gcc 컴파일과 make 컴파일을 해 보시오.

hi.c

```
#include <stdio.h>
int input();
int output(int);
int WinMain()
{
    int n;
    n=input();
    if(!(output(n)))
    {
        printf("Good Bye!!!\n");
    }
    return 0;
}
```

Makefile_Hi

```
hi: input.o output.o hi.o
    gcc -o hi input.o output.o hi.o
input.o: input.c
    gcc -c input.c
output.o: output.c
    gcc -c output.c
hi.o: hi.c
    gcc -c hi.c
clean:
    rm -f *.o hi
```



4.make

Solution-컴파일 할 경우

C:\DEV\c-workspaces>dir/w/p

C 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: BEDo-C858

C:\DEV\c-workspaces 디렉터리

```
[.]      [...]    fun1.c    fun2.c    hello.c    hello.exe    hi.c    input.c    main.c    Makefile  
makefile_Hello  Makefile_Hi  output.c  Sample.h  
12개 파일      55,670 바이트  
2개 디렉터리  16,827,695,104 바이트 남음
```

C:\DEV\c-workspaces>gcc -c input.c

C:\DEV\c-workspaces>gcc -c output.c

C:\DEV\c-workspaces>gcc -c hi.c

C:\DEV\c-workspaces>dir/w/p

C 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: BEDo-C858

C:\DEV\c-workspaces 디렉터리

```
[.]      [...]    fun1.c    fun2.c    hello.c    hello.exe    hi.c    hi.o    input.c    input.o  
main.c    Makefile    makefile_Hello  Makefile_Hi  output.c    output.o    Sample.h  
15개 파일      58,422 바이트  
2개 디렉터리  16,827,654,144 바이트 남음
```

...



Solution-컴파일 할 경우

```
C:\DEV\c-workspaces>gcc -o hi input.o output.o hi.o
```

```
C:\DEV\c-workspaces>dir/w/p
```

C 드라이브의 볼륨에는 이름이 없습니다.

볼륨 일련 번호: BED0-C858

C:\DEV\c-workspaces 디렉터리

```
[.]      [..]      fun1.c      fun2.c      hello.c      hello.exe      hi.c      hi.exe      hi.o      input.c  
input.o      main.c      Makefile      makefile_Hello      Makefile_Hi      output.c      output.o      Sample.h  
16개 파일      114,626 바이트  
2개 디렉터리 16,827,465,728 바이트 남음
```

```
C:\DEV\c-workspaces>hi
```

Enter a number : 5

Hello Linux

Hello Linux

Hello Linux

Hello Linux

Hello Linux

Good Bye!!!

```
C:\DEV\c-workspaces>
```



4.make

Solution-Makefile 이용할 경우

```
C:\DEV\c-workspaces>del *.o *.exe
```

```
C:\DEV\c-workspaces>dir/w/p
```

C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: BED0-C858

C:\DEV\c-workspaces 디렉터리

```
[.]      [..]      fun1.c      fun2.c      hello.c      hi.c      input.c      main.c      Makefile      makefile_Hello  
Makefile_Hi      output.c      Sample.h  
11개 파일      975 바이트  
2개 디렉터리 16,827,338,752 바이트 남음
```

```
C:\DEV\c-workspaces>make -f Makefile_Hi
```

```
gcc -c input.c  
gcc -c output.c  
gcc -c hi.c  
gcc -o hi input.o output.o hi.o
```

```
C:\DEV\c-workspaces>hi
```

Enter a number : 5

Hello Linux
Hello Linux
Hello Linux
Hello Linux
Hello Linux
Good Bye!!!

```
C:\DEV\c-workspaces>
```



4.make

Linux-Makefile 이용할 경우

```
k8s@DESKTOP-RoEQ2U6:~/c_workspaces$ make
```

Command 'make' not found, but can be installed with:

```
sudo apt install make      # version 4.3-4.1build1, or  
sudo apt install make-guile # version 4.3-4.1build1
```

```
k8s@DESKTOP-RoEQ2U6:~/c_workspaces$ sudo apt install make
```

[sudo] password for k8s:

Reading package lists... Done

Building dependency tree... Done

Reading state information... Done

Suggested packages:

 make-doc

The following NEW packages will be installed:

 make

0 upgraded, 1 newly installed, 0 to remove and 19 not upgraded.

Need to get 180 kB of archives.

After this operation, 426 kB of additional disk space will be used.

```
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 make amd64 4.3-4.1build1 [180 kB]
```

Fetched 180 kB in 2s (107 kB/s)

Selecting previously unselected package make.

(Reading database ... 45499 files and directories currently installed.)

```
Preparing to unpack .../make_4.3-4.1build1_amd64.deb ...
```

Unpacking make (4.3-4.1build1) ...

Setting up make (4.3-4.1build1) ...

```
Processing triggers for man-db (2.10.2-1) ...
```



4.make

Linux-Makefile 이용할 경우

```
k8s@DESKTOP-RoEQ2U6:~/c_workspaces$ make -v
GNU Make 4.3
Built for x86_64-pc-linux-gnu
Copyright (C) 1988-2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
k8s@DESKTOP-RoEQ2U6:~/c_workspaces$
```

『1과목』 프로그래밍 언어 디버깅 및 프로그램 통합





Syllabus

학습목표

- 개발된 모듈별 소스 코드를 디버깅 도구를 활용하여 디버깅을 수행하고 개발된 프로그램을 통합할 수 있다.

눈높이 체크

- 디버깅과 통합을 알고 계신가요?



1. gdb 개요 및 사용법

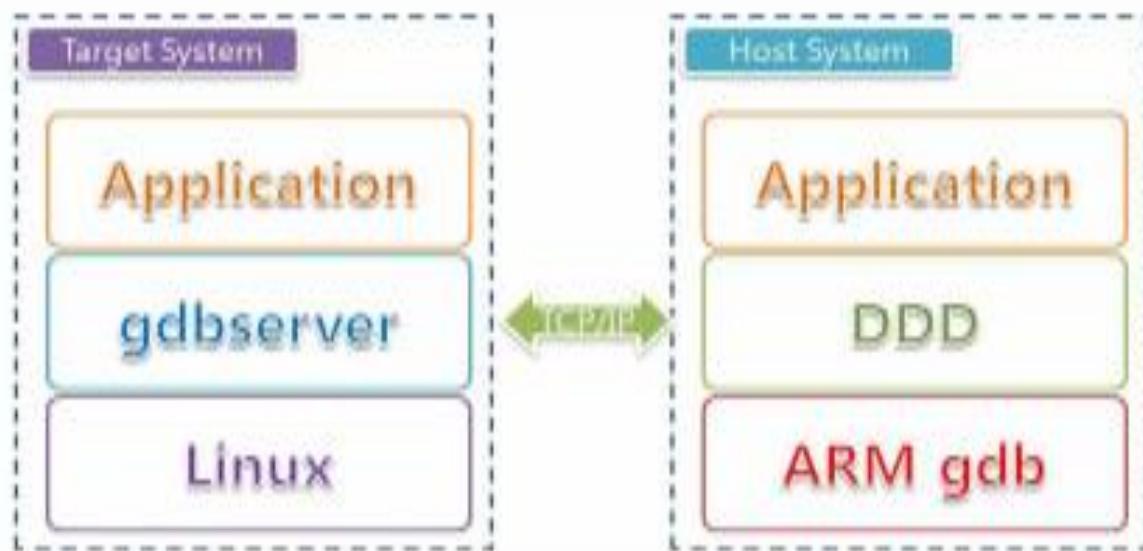
1. gdb의 개요

- 통상은 GDB라고 호칭되는 GNU 디버거(GNU Debugger)는 GNU SW 시스템을 위한 기본 디버거이다. GDB는 많은 유닉스 기반의 시스템에서 동작하는 이식성 있는 디버거로 포트란, 에이다, C, C++ 등의 다양한 프로그래밍 언어를 지원한다.
- 임베디드 시스템에서는 교차 개발 환경의 특성으로 인하여 원격 모드 디버깅이 지원된다. 원격 모드 디버깅은 gdb가 동작하는 시스템과 디버그 대상인 애플리케이션이 동작하는 시스템이 다른 경우에 적용이 가능하다. gcc를 이용하여 실행 파일을 만들고 gdb에 의한 디버깅을 반복 수행한 후 NFS 등을 활용하여 실행 파일을 타깃으로 전송한다.
- GDB는 컴퓨터 프로그램의 처리를 추적하고 수정할 수 있는 다양한 기능들을 제공한다. 유저는 프로그램의 내부 변수들의 값을 주시하거나 변경할 수 있다. 프로그램의 일반적인 실행 과정과 독립적으로 함수를 호출(call)할 수도 있다.

1. gdb 개요 및 사용법

2. 원격 모드 디버깅 예시

- 타깃 시스템에 gdbserver를 설치하고 호스트 시스템에는 ARM 전용 gdb와 GUI 환경에서 디버깅을 위한 DDD를 설치하여 원격 모드 디버깅이 가능하다.
- ARM 교차 개발 환경에서의 원격 모드 디버깅 예시





1. gdb 개요 및 사용법

3. gdb의 기본 사용법

- **gdb**로 디버깅을 하기 위해서는 사전에 **gcc**의 **-g** 옵션을 주어 컴파일을 해 놓아야 한다. 이는 **gdb**가 필요로 하는 부가적인 정보를 컴파일 시 추가해 놓기 위함이다. **gdb**를 시작하기 위한 기본적인 실행문법은 다음과 같다.

```
$ gdb [prog] [core|pid]
```

- **gdb** 명령어 뒤에 디버깅하고자 하는 프로그램 이름을 지정하는 것이다. **core** 파일은 실행 파일이 비정상적으로 종료하였을 경우에 생성되는 **core dump** 파일을 통해 디버깅하기 위함이고 **pid**는 현재 실행 중인 프로그램을 디버깅하기 위함이다.
- **gdb**를 실행하면 **command line prompt**가 출력되고 이때부터는 **gdb** 전용의 명령어를 이용하여 디버깅을 수행



1. gdb 개요 및 사용법

3. gdb의 기본 사용법

● gdb의 기본적인 명령어

명령어	문법	세부 설명
list	list [[file:]n func]	소스 파일의 내용을 출력
help	help [name]	gdb의 명령어 도움말 출력
break	break [[file:]n function] <-> d (del)	breakpoint를 라인이나 함수에 지정
clear	clear [[file:]n function]	breakpoint를 제거
run	run [arglist]	프로그램을 실행
print	print expr	expr의 값을 출력
disp	disp expr <-> undisp	expr의 값을 실행되는 동안 출력
continue	c [n]	breakpoint 이후 계속 진행. 지정된 n번째 breakpoint까지 진행
next	next [n]	멈춘 프로그램에서 다음 n개의 문장을 실행
step	step [n]	함수의 내부로 진입하여 한 문장씩 실행
up	up	현재의 함수를 호출한 함수 출력
down	down	현재의 함수가 호출한 함수 출력
return	ret	현재 함수에서 값을 반환
quit	quit	gdb 종료

2. 디버깅 및 프로그램 통합하기



1. 수행 순서

- **gdb를 이용한 디버깅 개요**

Gdb를 이용한 디버깅





2. 디버깅 및 프로그램 통합하기

2. 실행 방법

C:\DEV\c-workspaces>gdb

GNU gdb (GDB) 8.1

Copyright (C) 2018 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <<http://gnu.org/licenses/gpl.html>>

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.

This GDB was configured as "x86_64-w64-mingw32".

Type "show configuration" for configuration details.

For bug reporting instructions, please see:

<<http://www.gnu.org/software/gdb/bugs/>>.

Find the GDB manual and other documentation resources online at:

<<http://www.gnu.org/software/gdb/documentation/>>.

For help, type "help".

Type "apropos word" to search for commands related to "word".

(gdb) quit

C:\DEV\c-workspaces>



2. 디버깅 및 프로그램 통합하기

2. 실행 방법-Linux

```
k8s@DESKTOP-RoEQ2U6:~/c_workspaces$ gdb
```

Command 'gdb' not found, but can be installed with:

sudo apt install gdb

```
k8s@DESKTOP-RoEQ2U6:~/c_workspaces$ sudo apt install gdb
```

Reading package lists... Done

Building dependency tree... Done

Reading state information... Done

The following additional packages will be installed:

```
libbabeltrace1 libboost-regex1.74.0 libc6-dbg libdebuginfod-common libdebuginfod1 libipt2 libsource-highlight-common  
libsource-highlight4v5
```

Suggested packages:

```
gdb-doc gdbserver
```

The following NEW packages will be installed:

```
gdb libbabeltrace1 libboost-regex1.74.0 libc6-dbg libdebuginfod-common libdebuginfod1 libipt2  
libsource-highlight-common libsource-highlight4v5
```

0 upgraded, 9 newly installed, 0 to remove and 19 not upgraded.

Need to get 18.7 MB of archives.

After this operation, 35.1 MB of additional disk space will be used.

Do you want to continue? [Y/n] Y

...

```
k8s@DESKTOP-RoEQ2U6:~/c_workspaces$ gdb -v
```

GNU gdb (Ubuntu 12.1-oubuntu1~22.04) 12.1

Copyright (C) 2022 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <<http://gnu.org/licenses/gpl.html>>

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law.

```
k8s@DESKTOP-RoEQ2U6:~/c_workspaces$
```



2. 디버깅 및 프로그램 통합하기



3. 예시 코드를 생성

- **gdb_test.c**

```
#include <stdio.h>
```

```
int WinMain() {
    int a = 10;
    int b = 20;
    int result = 0;

    result = a + b;

    return 0;
}
```



2. 디버깅 및 프로그램 통합하기

4. 예시 코드를 컴파일

- **gdb로 디버깅을 하기 위해서 gcc의 -g 옵션을 이용하여 컴파일을 수행한다.**

```
C:\DEV\c-workspaces>gcc -o gdb_test gdb_test.c
```

```
C:\DEV\c-workspaces>dir/w/p
```

C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: BED0-C858

```
C:\DEV\c-workspaces 디렉터리
```

```
[.]      [..]      fun1.c      fun2.c      gdb_test.c      gdb_test.exe      hello.c      hi.c      hi.exe      hi.o  
input.c      input.o      main.c      Makefile      makefile_Hello      Makefile_Hi      output.c      output.o      Sample.h  
17개 파일      114,560 바이트  
2개 디렉터리 16,813,944,832 바이트 남음
```

```
C:\DEV\c-workspaces>gdb_test
```

```
C:\DEV\c-workspaces>
```



2. 디버깅 및 프로그램 통합하기

4. 예시 코드를 컴파일

- a에 10을 넣고 b에 20을 넣고 그 값을 result라는 4바이트짜리 변수에 넣고 종료, 출력문 printf를 쓰지 않았기 때문에 당연히 아무 반응도 없습니다.
- gdb(GNU Debugger)를 통해 디버깅을 해서, 어셈블리 코드로 바꿔서 결과를 보겠습니다.

```
C:\DEV\c-workspaces>gdb gdb_test
GNU gdb (GDB) 8.1
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-w64-mingw32".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from gdb_test...done.
(gdb) disass WinMain
...
```



2. 디버깅 및 프로그램 통합하기

4. 예시 코드를 컴파일

(gdb) disass WinMain

Dump of assembler code for function WinMain:

```
0x0000000000401550 <+0>: push %rbp  
0x0000000000401551 <+1>: mov %rsp,%rbp  
  
0x0000000000401554 <+4>: sub $0x10,%rsp  
0x0000000000401558 <+8>: movl $oxa,-0x4(%rbp)  
0x000000000040155f <+15>: movl $0x14,-0x8(%rbp)  
0x0000000000401566 <+22>: movl $0xo,-0xc(%rbp)  
0x000000000040156d <+29>: mov -0x4(%rbp),%edx  
0x0000000000401570 <+32>: mov -0x8(%rbp),%eax  
0x0000000000401573 <+35>: add %edx,%eax  
0x0000000000401575 <+37>: mov %eax,-0xc(%rbp)  
0x0000000000401578 <+40>: mov $oxo,%eax  
0x000000000040157d <+45>: add $0x10,%rsp  
0x0000000000401581 <+49>: pop %rbp  
0x0000000000401582 <+50>: retq  
0x0000000000401583 <+51>: nop  
0x0000000000401584 <+52>: nop  
---Type <return> to continue, or q <return> to quit---  
0x0000000000401585 <+53>: nop  
0x0000000000401586 <+54>: nop  
0x0000000000401587 <+55>: nop  
0x0000000000401588 <+56>: nop  
0x0000000000401589 <+57>: nop  
0x000000000040158a <+58>: nop  
0x000000000040158b <+59>: nop  
0x000000000040158c <+60>: nop  
0x000000000040158d <+61>: nop  
0x000000000040158e <+62>: nop  
0x000000000040158f <+63>: nop
```

End of assembler dump.

(gdb)

1. <+0>: push %rbp
2. <+1>: mov %rsp,%rbp

이 1번줄과 2번줄은 어셈블리에서 항상 나오는 표현입니다.
레지스터 rbp를 스택에 넣어두고, rsp를 rbp에 넣어줍니다.
참고로 윈도우와는 다르게 리눅스에서는 source 와
destination 의 위치가 바뀝니다.

즉, add %r1, %r2 가 윈도우에서는

$r1 = r1 + r2$

이었다면, 리눅스에서는

$r2 = r2 + r1$



2. 디버깅 및 프로그램 통합하기

4. 예시 코드를 컴파일

(gdb) disass WinMain

Dump of assembler code for function WinMain:

```

0x0000000000401550 <+0>: push %rbp
0x0000000000401551 <+1>: mov %rsp,%rbp
0x0000000000401554 <+4>: sub $0x10,%rsp
0x0000000000401558 <+8>: movl $oxa,-0xc(%rbp)
0x000000000040155f <+15>: movl $0x14,-0x8(%rbp)
0x0000000000401566 <+22>: movl $0xo,-0xc(%rbp)
0x000000000040156d <+29>: mov -0x4(%rbp),%edx
0x0000000000401570 <+32>: mov -0x8(%rbp),%eax
0x0000000000401573 <+35>: add %edx,%eax
0x0000000000401575 <+37>: mov %eax,-0xc(%rbp)
0x0000000000401578 <+40>: mov $0xo,%eax
0x000000000040157d <+45>: add $0x10,%rsp
0x0000000000401581 <+49>: pop %rbp
0x0000000000401582 <+50>: retq
0x0000000000401583 <+51>: nop
0x0000000000401584 <+52>: nop
---Type <return> to continue, or q <return> to quit---
0x0000000000401585 <+53>: nop
0x0000000000401586 <+54>: nop
0x0000000000401587 <+55>: nop
0x0000000000401588 <+56>: nop
0x0000000000401589 <+57>: nop
0x000000000040158a <+58>: nop
0x000000000040158b <+59>: nop
0x000000000040158c <+60>: nop
0x000000000040158d <+61>: nop
0x000000000040158e <+62>: nop
0x000000000040158f <+63>: nop

```

End of assembler dump.

(gdb)

본격적인 main 함수의 표현입니다.

movl \$oxa, -0xc(%rbp)

oxa(=dec. 10)의 값을 %rbp에 넣어주는 표현

그런데 왜 -0xc를 해줄까요? 분명히 int는 4바이트(=ox4)인데?

역어셈블 과정에서 디버거는 변수의 개수를 셉니다.

main 함수를 보시면 int 변수가

int a

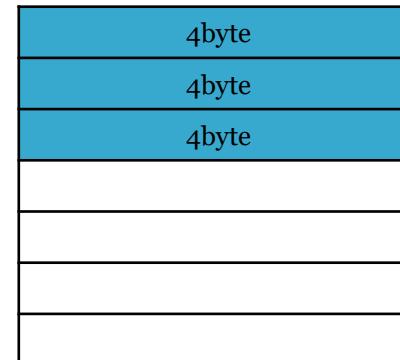
int b

int result

이렇게 3개가 나옵니다. 즉, 4바이트 * 3 = 12바이트 (=hex. 0xc)라는 공간을 "미리" 할당합니다.

(byte address로 하지 말고 4byte 어드레스라고 가정합니다.

즉 1칸에 4바이트) 3칸(12byte)을 할당



[rbp]



2. 디버깅 및 프로그램 통합하기

4. 예시 코드를 컴파일

(gdb) disass WinMain

Dump of assembler code for function WinMain:

```
0x0000000000401550 <+0>: push %rbp
0x0000000000401551 <+1>: mov %rsp,%rbp
0x0000000000401554 <+4>: sub $0x10,%rsp
0x0000000000401558 <+8>: movl $oxa,-0x4(%rbp)
0x000000000040155f <+15>: movl $0x14,-0x8(%rbp)
0x0000000000401566 <+22>: movl $0x0,-0xc(%rbp)
0x000000000040156d <+29>: mov -0x4(%rbp),%edx
0x0000000000401570 <+32>: mov -0x8(%rbp),%eax
0x0000000000401573 <+35>: add %edx,%eax
0x0000000000401575 <+37>: mov %eax,-0xc(%rbp)
0x0000000000401578 <+40>: mov $0x0,%eax
0x000000000040157d <+45>: add $0x10,%rsp
0x0000000000401581 <+49>: pop %rbp
0x0000000000401582 <+50>: retq
0x0000000000401583 <+51>: nop
0x0000000000401584 <+52>: nop
---Type <return> to continue, or q <return> to quit---
0x0000000000401585 <+53>: nop
0x0000000000401586 <+54>: nop
0x0000000000401587 <+55>: nop
0x0000000000401588 <+56>: nop
0x0000000000401589 <+57>: nop
0x000000000040158a <+58>: nop
0x000000000040158b <+59>: nop
0x000000000040158c <+60>: nop
0x000000000040158d <+61>: nop
0x000000000040158e <+62>: nop
0x000000000040158f <+63>: nop
```

End of assembler dump.

(gdb)

변수가 나오는 순서에 따라서 맨 아래서부터 메모리 위치를 배정해줍니다.

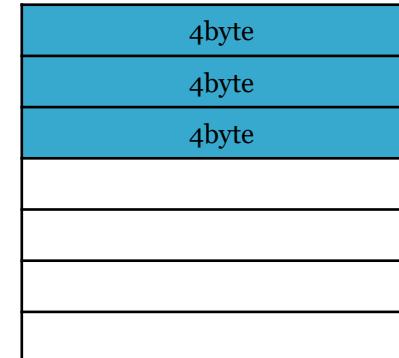
int a = 10;

즉, 10의 값(=hex. ox_a)을 맨 아래인 rbp - 12byte 위치에 넣어 줍니다.

이것을 어셈블리로 표현하면

movl \$oxa, -c(%rbp)

가 되는 것입니다.



[rbp]



2. 디버깅 및 프로그램 통합하기

4. 예시 코드를 컴파일

(gdb) disass WinMain

Dump of assembler code for function WinMain:

```
0x0000000000401550 <+0>: push %rbp
0x0000000000401551 <+1>: mov %rsp,%rbp
0x0000000000401554 <+4>: sub $0x10,%rsp
0x0000000000401558 <+8>: movl $0xa,-0x4(%rbp)
0x000000000040155f <+15>: movl $0x14,-0x8(%rbp)
0x0000000000401566 <+22>: movl $0xo,-0xc(%rbp)
0x000000000040156d <+29>: mov -0x4(%rbp),%edx
0x0000000000401570 <+32>: mov -0x8(%rbp),%eax
0x0000000000401573 <+35>: add %edx,%eax
0x0000000000401575 <+37>: mov %eax,-0xc(%rbp)
0x0000000000401578 <+40>: mov $0xo,%eax
0x000000000040157d <+45>: add $0x10,%rsp
0x0000000000401581 <+49>: pop %rbp
0x0000000000401582 <+50>: retq
0x0000000000401583 <+51>: nop
0x0000000000401584 <+52>: nop
---Type <return> to continue, or q <return> to quit---
0x0000000000401585 <+53>: nop
0x0000000000401586 <+54>: nop
0x0000000000401587 <+55>: nop
0x0000000000401588 <+56>: nop
0x0000000000401589 <+57>: nop
0x000000000040158a <+58>: nop
0x000000000040158b <+59>: nop
0x000000000040158c <+60>: nop
0x000000000040158d <+61>: nop
0x000000000040158e <+62>: nop
0x000000000040158f <+63>: nop
```

End of assembler dump.

(gdb)

다음 변수인 int b = 20; 는 -12 바이트 위치가 아닌, 4바이트가 줄어든 -8 바이트 위치에 할당해줍니다.
즉, 20의 값을 -0x8(%rbp) (=%rbp - 8byte)에 넣어줍니다.
20은 hexa 값으로 0x14입니다. ($1 * 16 + 4 * 1$)



2. 디버깅 및 프로그램 통합하기

4. 예시 코드를 컴파일

(gdb) disass WinMain

Dump of assembler code for function WinMain:

```
0x0000000000401550 <+0>: push %rbp
0x0000000000401551 <+1>: mov %rsp,%rbp
0x0000000000401554 <+4>: sub $0x10,%rsp
0x0000000000401558 <+8>: movl $oxa,-0x4(%rbp)
0x000000000040155f <+15>: movl $0x14,-0x8(%rbp)
0x0000000000401566 <+22>: movl $ox0,-0xc(%rbp)
```

```
0x000000000040156d <+29>: mov -0x4(%rbp),%edx
0x0000000000401570 <+32>: mov -0x8(%rbp),%eax
0x0000000000401573 <+35>: add %edx,%eax
0x0000000000401575 <+37>: mov %eax,-0xc(%rbp)
0x0000000000401578 <+40>: mov $ox0,%eax
0x000000000040157d <+45>: add $0x10,%rsp
0x0000000000401581 <+49>: pop %rbp
0x0000000000401582 <+50>: retq
0x0000000000401583 <+51>: nop
0x0000000000401584 <+52>: nop
```

---Type <return> to continue, or q <return> to quit---

```
0x0000000000401585 <+53>: nop
0x0000000000401586 <+54>: nop
0x0000000000401587 <+55>: nop
0x0000000000401588 <+56>: nop
0x0000000000401589 <+57>: nop
0x000000000040158a <+58>: nop
0x000000000040158b <+59>: nop
0x000000000040158c <+60>: nop
0x000000000040158d <+61>: nop
0x000000000040158e <+62>: nop
0x000000000040158f <+63>: nop
```

End of assembler dump.

(gdb)

마지막 변수는 int result = 0; 이므로,

0의 값을 -4바이트 줄어든 위치에 넣어줍니다.



2. 디버깅 및 프로그램 통합하기

4. 예시 코드를 컴파일

(gdb) disass WinMain

Dump of assembler code for function WinMain:

```
0x0000000000401550 <+0>: push %rbp
0x0000000000401551 <+1>: mov %rsp,%rbp
0x0000000000401554 <+4>: sub $0x10,%rsp
0x0000000000401558 <+8>: movl $0xa,-0x4(%rbp)
0x000000000040155f <+15>: movl $0x14,-0x8(%rbp)
0x0000000000401566 <+22>: movl $0xo,-0xc(%rbp)
0x000000000040156d <+29>: mov -0x4(%rbp),%edx
0x0000000000401570 <+32>: mov -0x8(%rbp),%eax
0x0000000000401573 <+35>: add %edx,%eax
0x0000000000401575 <+37>: mov %eax,-0xc(%rbp)
0x0000000000401578 <+40>: mov $0xo,%eax
0x000000000040157d <+45>: add $0x10,%rsp
0x0000000000401581 <+49>: pop %rbp
0x0000000000401582 <+50>: retq
0x0000000000401583 <+51>: nop
0x0000000000401584 <+52>: nop
---Type <return> to continue, or q <return> to quit---
0x0000000000401585 <+53>: nop
0x0000000000401586 <+54>: nop
0x0000000000401587 <+55>: nop
0x0000000000401588 <+56>: nop
0x0000000000401589 <+57>: nop
0x000000000040158a <+58>: nop
0x000000000040158b <+59>: nop
0x000000000040158c <+60>: nop
0x000000000040158d <+61>: nop
0x000000000040158e <+62>: nop
0x000000000040158f <+63>: nop
```

End of assembler dump.

(gdb)

이 것은 계산을 위해서 값을 옮기는 과정입니다.
계산 식은

result = a + b;
해당 인수들의 주소는
&a = -0xc(%rbp)
&b = -0x8(%rbp)

여기서는 %rbp - oxc 위치에 있는 값(=10)을 edx 레지스터에 넣어주었습니다.

rbp에서 직접적으로 계산을 하지 않고 edx 와 eax 같은 범용 레지스터에 임시적으로 옮겨놓고, 계산을 한 후 그 결과 값을 다시 rbp에 가져오게 됩니다.



2. 디버깅 및 프로그램 통합하기

4. 예시 코드를 컴파일

(gdb) disass WinMain

Dump of assembler code for function WinMain:

```
0x0000000000401550 <+0>: push %rbp
0x0000000000401551 <+1>: mov %rsp,%rbp
0x0000000000401554 <+4>: sub $0x10,%rsp
0x0000000000401558 <+8>: movl $0xa,-0x4(%rbp)
0x000000000040155f <+15>: movl $0x14,-0x8(%rbp)
_0x0000000000401566 <+22>: movl $0xo,-0xc(%rbp)
_0x000000000040156d <+29>: mov -0x4(%rbp),%edx
0x0000000000401570 <+32>: mov -0x8(%rbp),%eax
0x0000000000401573 <+35>: add %edx,%eax
0x0000000000401575 <+37>: mov %eax,-0xc(%rbp)
0x0000000000401578 <+40>: mov $0xo,%eax
0x000000000040157d <+45>: add $0x10,%rsp
0x0000000000401581 <+49>: pop %rbp
0x0000000000401582 <+50>: retq
0x0000000000401583 <+51>: nop
0x0000000000401584 <+52>: nop
---Type <return> to continue, or q <return> to quit---
0x0000000000401585 <+53>: nop
0x0000000000401586 <+54>: nop
0x0000000000401587 <+55>: nop
0x0000000000401588 <+56>: nop
0x0000000000401589 <+57>: nop
0x000000000040158a <+58>: nop
0x000000000040158b <+59>: nop
0x000000000040158c <+60>: nop
0x000000000040158d <+61>: nop
0x000000000040158e <+62>: nop
0x000000000040158f <+63>: nop
```

End of assembler dump.

(gdb)

rbp -8byte의 위치에 있는 값(=20)을 eax 레지스터에 넣어줍니다.



2. 디버깅 및 프로그램 통합하기

4. 예시 코드를 컴파일

(gdb) disass WinMain

Dump of assembler code for function WinMain:

```
0x0000000000401550 <+0>: push %rbp
0x0000000000401551 <+1>: mov %rsp,%rbp
0x0000000000401554 <+4>: sub $0x10,%rsp
0x0000000000401558 <+8>: movl $0xa,-0x4(%rbp)
0x000000000040155f <+15>: movl $0x14,-0x8(%rbp)
_0x0000000000401566 <+22>: movl $0x0,-0xc(%rbp)
_0x000000000040156d <+29>: mov -0x4(%rbp),%edx
0x0000000000401570 <+32>: mov -0x8(%rbp),%eax
0x0000000000401573 <+35>: add %edx,%eax
0x0000000000401575 <+37>: mov %eax,-0xc(%rbp)
0x0000000000401578 <+40>: mov $0x0,%eax
0x000000000040157d <+45>: add $0x10,%rsp
0x0000000000401581 <+49>: pop %rbp
0x0000000000401582 <+50>: retq
0x0000000000401583 <+51>: nop
0x0000000000401584 <+52>: nop
---Type <return> to continue, or q <return> to quit---
0x0000000000401585 <+53>: nop
0x0000000000401586 <+54>: nop
0x0000000000401587 <+55>: nop
0x0000000000401588 <+56>: nop
0x0000000000401589 <+57>: nop
0x000000000040158a <+58>: nop
0x000000000040158b <+59>: nop
0x000000000040158c <+60>: nop
0x000000000040158d <+61>: nop
0x000000000040158e <+62>: nop
0x000000000040158f <+63>: nop
```

End of assembler dump.

(gdb)

eax = eax + edx

eax에는 **20**이 들어가 있었고, **edx**에는 **10**이 들어가 있음.

최종적으로

eax = 20 + 10

과 동일합니다.

eax에 결과 값 **30**이 들어가게 됩니다.



2. 디버깅 및 프로그램 통합하기

4. 예시 코드를 컴파일

(gdb) disass WinMain

Dump of assembler code for function WinMain:

```
0x0000000000401550 <+0>: push %rbp
0x0000000000401551 <+1>: mov %rsp,%rbp
0x0000000000401554 <+4>: sub $0x10,%rsp
0x0000000000401558 <+8>: movl $0xa,-0x4(%rbp)
0x000000000040155f <+15>: movl $0x14,-0x8(%rbp)
_0x0000000000401566 <+22>: movl $0xo,-0xc(%rbp)
_0x000000000040156d <+29>: mov -0x4(%rbp),%edx
0x0000000000401570 <+32>: mov -0x8(%rbp),%eax
0x0000000000401573 <+35>: add %edx,%eax
0x0000000000401575 <+37>: mov %eax,-0xc(%rbp)
0x0000000000401578 <+40>: mov $0xo,%eax
0x000000000040157d <+45>: add $0x10,%rsp
0x0000000000401581 <+49>: pop %rbp
0x0000000000401582 <+50>: retq
0x0000000000401583 <+51>: nop
0x0000000000401584 <+52>: nop
---Type <return> to continue, or q <return> to quit---
0x0000000000401585 <+53>: nop
0x0000000000401586 <+54>: nop
0x0000000000401587 <+55>: nop
0x0000000000401588 <+56>: nop
0x0000000000401589 <+57>: nop
0x000000000040158a <+58>: nop
0x000000000040158b <+59>: nop
0x000000000040158c <+60>: nop
0x000000000040158d <+61>: nop
0x000000000040158e <+62>: nop
0x000000000040158f <+63>: nop
```

End of assembler dump.

(gdb)

계산이 끝났으니 값을 옮겨줘야합니다.

eax 레지스터에 있는 값을 -0x4(%rbp), 즉 %rbp - 4byte 위치에 넣어줍니다.

저희는 이미 위의 과정들을 통하여 어떤 변수(값)가 어느 위치에 있는지 알고 있습니다.

rbp에서 4byte 줄어든 위치는 result 변수가 할당되어 있는 자리입니다.

즉, result 변수에 eax 레지스터의 값, 30이 들어가게 됩니다.



2. 디버깅 및 프로그램 통합하기

4. 예시 코드를 컴파일

(gdb) disass WinMain

Dump of assembler code for function WinMain:

```
0x00000000000401550 <+0>: push %rbp
0x00000000000401551 <+1>: mov %rsp,%rbp
0x00000000000401554 <+4>: sub $0x10,%rsp
0x00000000000401558 <+8>: movl $0xa,-0x4(%rbp)
0x0000000000040155f <+15>: movl $0x14,-0x8(%rbp)
_0x00000000000401566 <+22>: movl $0x0,-0xc(%rbp)
_0x0000000000040156d <+29>: mov -0x4(%rbp),%edx
0x00000000000401570 <+32>: mov -0x8(%rbp),%eax
0x00000000000401573 <+35>: add %edx,%eax
0x00000000000401575 <+37>: mov %eax,-0xc(%rbp)
0x00000000000401578 <+40>: mov $0x0,%eax
0x0000000000040157d <+45>: add $0x10,%rsp
0x00000000000401581 <+49>: pop %rbp
0x00000000000401582 <+50>: retq
0x00000000000401583 <+51>: nop
0x00000000000401584 <+52>: nop
---Type <return> to continue, or q <return> to quit---
0x00000000000401585 <+53>: nop
0x00000000000401586 <+54>: nop
0x00000000000401587 <+55>: nop
0x00000000000401588 <+56>: nop
0x00000000000401589 <+57>: nop
0x0000000000040158a <+58>: nop
0x0000000000040158b <+59>: nop
0x0000000000040158c <+60>: nop
0x0000000000040158d <+61>: nop
0x0000000000040158e <+62>: nop
0x0000000000040158f <+63>: nop
```

End of assembler dump.

(gdb)

다음에 있을 연산이나 작업을 위해, eax 레지스터를 초기화 합니다.



2. 디버깅 및 프로그램 통합하기

4. 예시 코드를 컴파일

(gdb) disass WinMain

Dump of assembler code for function WinMain:

```
0x0000000000401550 <+0>: push %rbp
0x0000000000401551 <+1>: mov %rsp,%rbp
0x0000000000401554 <+4>: sub $0x10,%rsp
0x0000000000401558 <+8>: movl $0xa,-0x4(%rbp)
0x000000000040155f <+15>: movl $0x14,-0x8(%rbp)
_0x0000000000401566 <+22>: movl $0xo,-0xc(%rbp)
_0x000000000040156d <+29>: mov -0x4(%rbp),%edx
0x0000000000401570 <+32>: mov -0x8(%rbp),%eax
0x0000000000401573 <+35>: add %edx,%eax
0x0000000000401575 <+37>: mov %eax,-0xc(%rbp)
0x0000000000401578 <+40>: mov $0xo,%eax
0x000000000040157d <+45>: add $0x10,%rsp
0x0000000000401581 <+49>: pop %rbp
0x0000000000401582 <+50>: retq
0x0000000000401583 <+51>: nop
0x0000000000401584 <+52>: nop
---Type <return> to continue, or q <return> to quit---
0x0000000000401585 <+53>: nop
0x0000000000401586 <+54>: nop
0x0000000000401587 <+55>: nop
0x0000000000401588 <+56>: nop
0x0000000000401589 <+57>: nop
0x000000000040158a <+58>: nop
0x000000000040158b <+59>: nop
0x000000000040158c <+60>: nop
0x000000000040158d <+61>: nop
0x000000000040158e <+62>: nop
0x000000000040158f <+63>: nop
```

End of assembler dump.

(gdb)

모든 프로그램이 끝났습니다. 1번줄에서 stack memory에 save 해 두었던 값을 rbp에 다시 옮겨줍니다.



2. 디버깅 및 프로그램 통합하기

4. 예시 코드를 컴파일

(gdb) disass WinMain

Dump of assembler code for function WinMain:

```
0x00000000000401550 <+0>: push %rbp
0x00000000000401551 <+1>: mov %rsp,%rbp
0x00000000000401554 <+4>: sub $0x10,%rsp
0x00000000000401558 <+8>: movl $0xa,-0x4(%rbp)
0x0000000000040155f <+15>: movl $0x14,-0x8(%rbp)
_0x00000000000401566 <+22>: movl $0x0,-0xc(%rbp)
_0x0000000000040156d <+29>: mov -0x4(%rbp),%edx
0x00000000000401570 <+32>: mov -0x8(%rbp),%eax
0x00000000000401573 <+35>: add %edx,%eax
0x00000000000401575 <+37>: mov %eax,-0xc(%rbp)
0x00000000000401578 <+40>: mov $0x0,%eax
0x0000000000040157d <+45>: add $0x10,%rsp
0x00000000000401581 <+49>: pop %rbp
0x00000000000401582 <+50>: retq
```

return 0; 의 의미

```
0x00000000000401583 <+51>: nop
0x00000000000401584 <+52>: nop
---Type <return> to continue, or q <return> to quit---
```

```
0x00000000000401585 <+53>: nop
0x00000000000401586 <+54>: nop
0x00000000000401587 <+55>: nop
0x00000000000401588 <+56>: nop
0x00000000000401589 <+57>: nop
0x0000000000040158a <+58>: nop
0x0000000000040158b <+59>: nop
0x0000000000040158c <+60>: nop
0x0000000000040158d <+61>: nop
0x0000000000040158e <+62>: nop
0x0000000000040158f <+63>: nop
```

End of assembler dump.

(gdb)



2. 디버깅 및 프로그램 통합하기

5. Linux에서 수행

```
k8s@DESKTOP-RoEQ2U6:~/c_workspaces$ cat > gdb_test.c
```

```
#include <stdio.h>
```

```
int main() {
    int a = 10;
    int b = 20;
    int result = 0;

    result = a + b;

    return 0;
}
```

```
k8s@DESKTOP-RoEQ2U6:~/c_workspaces$ gcc -o gdb_test gdb_test.c
```

```
k8s@DESKTOP-RoEQ2U6:~/c_workspaces$ gdb gdb_test
```

```
GNU gdb (Ubuntu 12.1-oubuntu1~22.04) 12.1
```

```
Copyright (C) 2022 Free Software Foundation, Inc.
```

```
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
```

```
This is free software: you are free to change and redistribute it.
```

```
There is NO WARRANTY, to the extent permitted by law.
```

```
Type "show copying" and "show warranty" for details.
```

```
This GDB was configured as "x86_64-linux-gnu".
```

```
Type "show configuration" for configuration details.
```

```
For bug reporting instructions, please see:
```

```
<https://www.gnu.org/software/gdb/bugs/>.
```

```
Find the GDB manual and other documentation resources online at:
```

```
<http://www.gnu.org/software/gdb/documentation/>.
```

```
For help, type "help".
```

```
Type "apropos word" to search for commands related to "word"...
```

```
Reading symbols from gdb_test...
```

```
(No debugging symbols found in gdb_test)
```



2. 디버깅 및 프로그램 통합하기

5. Linux에서 수행

(gdb) disass main

Dump of assembler code for function main:

```
0x0000000000001129 <+0>: endbr64
0x000000000000112d <+4>: push %rbp
0x000000000000112e <+5>: mov %rsp,%rbp
0x0000000000001131 <+8>: movl $oxa,-0xc(%rbp)
0x0000000000001138 <+15>: movl $0x14,-0x8(%rbp)
0x000000000000113f <+22>: movl $ox0,-0x4(%rbp)
0x0000000000001146 <+29>: mov -0xc(%rbp),%edx
0x0000000000001149 <+32>: mov -0x8(%rbp),%eax
0x000000000000114c <+35>: add %edx,%eax
0x000000000000114e <+37>: mov %eax,-0x4(%rbp)
0x0000000000001151 <+40>: mov $ox0,%eax
0x0000000000001156 <+45>: pop %rbp
0x0000000000001157 <+46>: ret
```

End of assembler dump.

(gdb) quit

k8s@DESKTOP-RoEQ2U6:~/c_workspaces\$



3. 디버깅 및 프로그램 통합하기 예

Quiz1

- 다음 <보기>처럼 비트 연산처리하는 프로그램을 구현하고 디버깅을 수행해보자.
- <보기>

```
select menu: 1.AND 2.OR 3.XOR 4.exit => 1
```

```
input two characters => a b
```

```
01100001 AND 01100010 = 01100000
```

```
select menu: 1.AND 2.OR 3.XOR 4.exit => 2
```

```
input two characters => ab
```

```
01100001 OR 01100010 = 01100011
```

```
select menu: 1.AND 2.OR 3.XOR 4.exit => 3
```

```
input two characters => ab
```

```
01100001 XOR 01100010 = 00000011
```

```
select menu: 1.AND 2.OR 3.XOR 4.exit => 4
```



3. 디버깅 및 프로그램 통합하기 예

Solution1

- **bitoperation.c**

```
#include <stdio.h>

void printbit(char ch);

int WinMain()
{
    char a, b, c;
    int select;

    while (1) {
        printf("\nselect menu: 1.AND 2.OR 3.XOR 4.exit => ");
        scanf("%d", &select);

        if (select == 1) {
            printf("input two characters => ");
            scanf("\n%c %c", &a, &b);
            c = a&b;
            printbit(a);
            printf(" AND ");
            printbit(b);
            printf(" = ");
            printbit(c);
        } else if (select == 2) {
```



3. 디버깅 및 프로그램 통합하기 예

Solution1

```
• printf("input two characters => ");
  scanf("\n%c %c", &a, &b);
  c = a|b;
  printbit(a);
  printf(" OR ");
  printbit(b);
  printf(" = ");
  printbit(c);
} else if (select == 3) {
  printf("input two characters => ");
  scanf("\n%c %c", &a, &b);
  c = a^b;
  printbit(a);
  printf(" XOR ");
  printbit(b);
  printf(" = ");
  printbit(c);
} else if (select == 4) {
  break;
} else {
  printf("select error\n");
}
}

return 0;
}
```



3. 디버깅 및 프로그램 통합하기 예

Solution1

```
void printbit(char ch)
{
    int i;

    for (i=0; i<8; i++) {
        printf("%d", (ch&0x01) ? 1 : 0);
        ch = ch<<1;
    }
}
```

```
C:\DEV\c-workspaces>gcc -o bitoperation bitoperation.c
C:\DEV\c-workspaces>bitoperation

select menu: 1.AND 2.OR 3.XOR 4.exit => 1
input two characters => AB
01000001 AND 01000010 = 01000000
select menu: 1.AND 2.OR 3.XOR 4.exit => 2
input two characters => AB
01000001 OR 01000010 = 01000011
select menu: 1.AND 2.OR 3.XOR 4.exit => 3
input two characters => AB
01000001 XOR 01000010 = 00000011
select menu: 1.AND 2.OR 3.XOR 4.exit => 4

C:\DEV\c-workspaces>
```

『1과목』 프로그래밍 언어

Python





학습목표

- 파이썬 연산처리, 파이썬 변수, 파이썬 자료형 등에 대해 알 수 있다.

눈높이 체크

- 플랫폼의 개념을 알고 계신가요?



본 강좌의 의의

- 파이썬은 웹 서비스뿐만 아니라, AI/ 딥러닝 등데이터 분석에서도 널리 쓰이는 언어로, 쉽지만 강력한 언어이다.
- Python 기초 문법과 Coding 작성법 이해를 통해 프로그램을 제작 할 수 있다.
- 프로그램 설치법, 코드를 작성할 수 있다.
- Python 언어를 활용하여 필요 프로그램을 직접 만들 수 있다.



Syllabus

파이썬 소개(Python Introduction)

- 파이썬 창시자 귀도 반 로섬은 네덜란드 출신 컴퓨터 프로그래머로, 1989년 크리스마스 주간에 연구실에서 파이썬을 만들어 1991년에 공식 발표
- 플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑 (*dynamically typed*) 대화형 언어이다.
- 파이썬은 초보자부터 전문가까지 폭넓은 사용자층을 보유하고 있다.

A screenshot of a web browser displaying Guido van Rossum's personal homepage at gvanrossum.github.io. The page features a portrait of Guido van Rossum, a link to his personal home page, and two small links at the bottom.

← → ⌂ gvanrossum.github.io ⌂ ☆ ★ ☰ 🔍 :

Guido van Rossum - Personal Home Page

["Gawky and proud of it."](#)
[Who I Am](#)

<https://gvanrossum.github.io/>



Syllabus

파이썬 소개(Python Introduction)

- 파이썬이라는 이름은 로섬이 좋아하는 TV 프로그램인 Monty python 코미디언 그룹의 Flying circus 코메디 프로(1969~1974년 영국 BBC에서 방송)에서 따왔습니다.
- Silly Olympics (바보 올림픽): Flying circus 코메디 프로 중 , <https://www.youtube.com/watch?v=qgSzGIkFq2A>
- 파이썬의 사전적 의미가 그리스 신화에 나오는 큰 비단뱀을 뜻해서 파이썬 책이나 로고에 뱀 모양이 많이 나옵니다.





Syllabus

파이썬 소개(Python Introduction)

- 파이썬 프로그래밍 언어는 Google Trends를 통해 분석한 결과, 2023년 기준 가장 인기있는 언어 1위

The screenshot shows the PYPL website interface. At the top, there's a navigation bar with links for 'PYPL Index', '10 TOP IDE', '10 TOP ODE', and '10 TOP DB'. Below the navigation bar, the title 'PYPL PopularitY of Programming Language' is displayed. A sub-header indicates the data is 'Worldwide, Feb 2023 compared to a year ago'. A table lists the top 10 languages based on popularity:

Rank	Change	Language	Share	Trend
1		Python	27.7 %	-0.7 %
2		Java	16.79 %	-1.3 %
3		JavaScript	9.65 %	+0.6 %
4	↑	C#	6.97 %	-0.5 %
5	↓	C/C++	6.87 %	-0.6 %
6		PHP	5.23 %	-0.8 %
7		R	4.11 %	-0.1 %

The footer of the website reads 'The PYPL PopularitY of Programming Language'.

PYPL:Popularity of Programming Language: <http://pypl.github.io/PYPL.html>



Syllabus

파이썬 특징(Python Features)

- 쉬운 문법과 간결함

- 사람의 사고 체계와 유사하고 문법 자체가 아주 쉽고 간결함
- 소스 코드가 이해하기 쉽어 공동 작업과 유지 보수가 쉬움

- 동적 타이핑(dynamic typing) 범용 프로그래밍 언어

- 프로그램의 실행 시점에서 프로그램 변수의 타입을 결정하는 언어

- 다양한 플랫폼 지원과 호환성

- 다양한 플랫폼에서 동작하도록 지원
- C언어와의 호환성이 매우 뛰어남

- 풍부한 라이브러리(모듈)

- 다양한 파이썬 내부 라이브러리 뿐만 아니라 다양한 서드 파티 third party 라이브러리 사용 가능

- 풀 언어(glue language)

- 다른 언어로 쓰인 모듈들을 연결하는 언어로 자주 이용
- C나 C++ 언어로 만든 프로그램을 파이썬에서 사용 가능

- 다양한 언어의 문자 처리

- 유니코드 문자열을 지원해서 다양한 언어의 문자 처리 가능



Syllabus

파이썬 응용(Python Applications)

● 능력 단위

분류번호

시스템 유틸리티	<ul style="list-style-type: none">인증 / 권한 관리캐시 관리 / 파일 시스템 도구코드 분석 / 빌드, 디버깅 도구
GUI(Graphic User Interface) 개발	<ul style="list-style-type: none">윈도우 형태의 프로그래밍 가능Tkinter 패키지
웹 프로그래밍	<ul style="list-style-type: none">웹에서 회원 가입, 게시판, 블로그 등의 프로그램 제작 가능Django, Flask 등 파이썬 기반 웹 프레임워크
데이터베이스	<ul style="list-style-type: none">Oracle, MySQL, PostgreSQL 등의 데이터베이스에 접근 가능한 도구 제공데이터베이스를 이용한 다양한 프로그램 제작 가능
데이터 분석 및 시각화	<ul style="list-style-type: none">데이터 처리, 탐색, 분석, 시각화까지 구현 가능NumPy, Pandas, DataFrame, Matplotlib 패키지
기계학습/딥러닝 모델	<ul style="list-style-type: none">모델 생성과 데이터 학습 등을 위한 편리한 도구 제공Scikit-learn, TensorFlow, PyTorch, Keras 패키지
이미지 처리/컴퓨터 비전	<ul style="list-style-type: none">컴퓨터 비전과 관련된 다양한 처리와 알고리즘을 제공OpenCV 패키지
자연어 처리	<ul style="list-style-type: none">텍스트 데이터나 자연어 처리를 위한 다양한 모델 제공WordCloud, Gensim 패키지
게임 개발	<ul style="list-style-type: none">게임 개발에 필요한 이미지, 소리, 입력 장치 등의 효율적인 처리 방법을 제공PyGame 패키지
기타	<ul style="list-style-type: none">로봇, 사물인터넷, 클라우드 컴퓨팅 등 수 많은 분야에서 다양한 응용으로 활용

자세한 내용들은 Awesome Python(<https://awesome-python.com>) 참고



2. Pre-Requisites

파이썬 설치 및 환경 개요

- 파이썬을 이용하는 방법은 다양하게 존재하며 사용하기 원하는 환경에 따라서 선택
- 개인 컴퓨터에서 파이썬 사용시에는 홈페이지 (<https://www.python.org>)에서 설치 파일을 다운로드하여 설치 할 수 있으나,
- 파이썬 기본 에디터보다는 다양한 통합개발환경(IDE)을 활용하여 프로그래밍 가능
- 대표적인 통합개발환경(IDE):
PyCharm(<https://www.jetbrains.com/pycharm>), Visual Studio Code(<https://code.visualstudio.com>)
- 본 강의에서는 윈10+Miniconda에서 Visual Studio Code를 IDE로 파이썬을 실습



2. Pre-Requisites

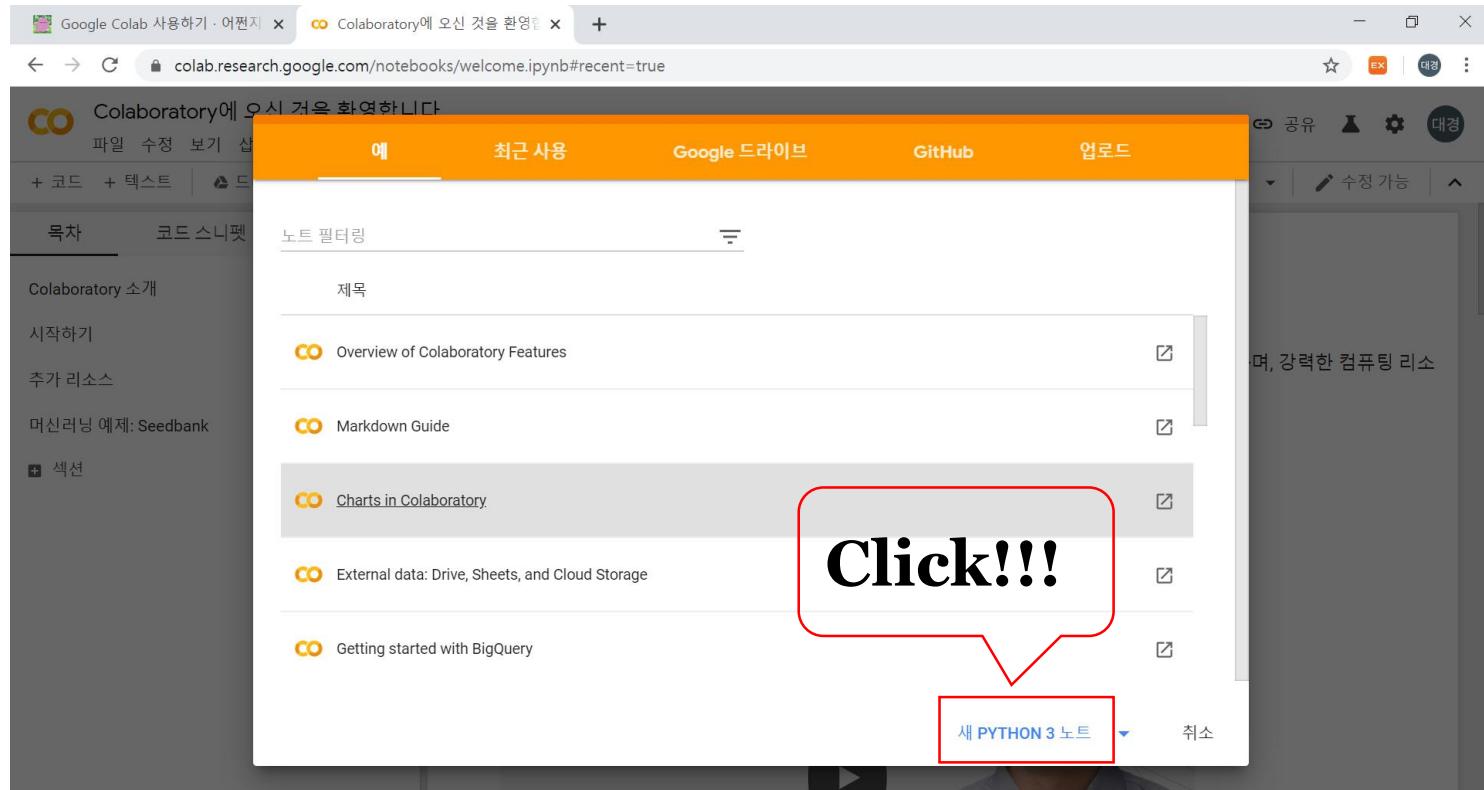
Google Colab 이란?

- Google Colab은 AI개발자들을 위해 구글에서 제공하는 무료 클라우드 서비스
- 풀 네임은 Google Colaboratory
- Google Drive + Jupyter Notebook
 - Google Drive처럼 협업 가능(동시에 수정 가능)
- <https://colab.research.google.com/>로 접속시 사용 가능
- 컴퓨터 사양(19년 12월 기준)
 - Ubuntu 17.10
 - CPU 제논 2.3GHz
 - 메모리 13G
 - GPU : K80 또는 T4 :
 - TPU도 사용 가능
- GPU 사용시 최대 12시간
- Github의 소스 코드를 Colab에서 사용 가능

2. Pre-Requisites

Google Colab 이란?

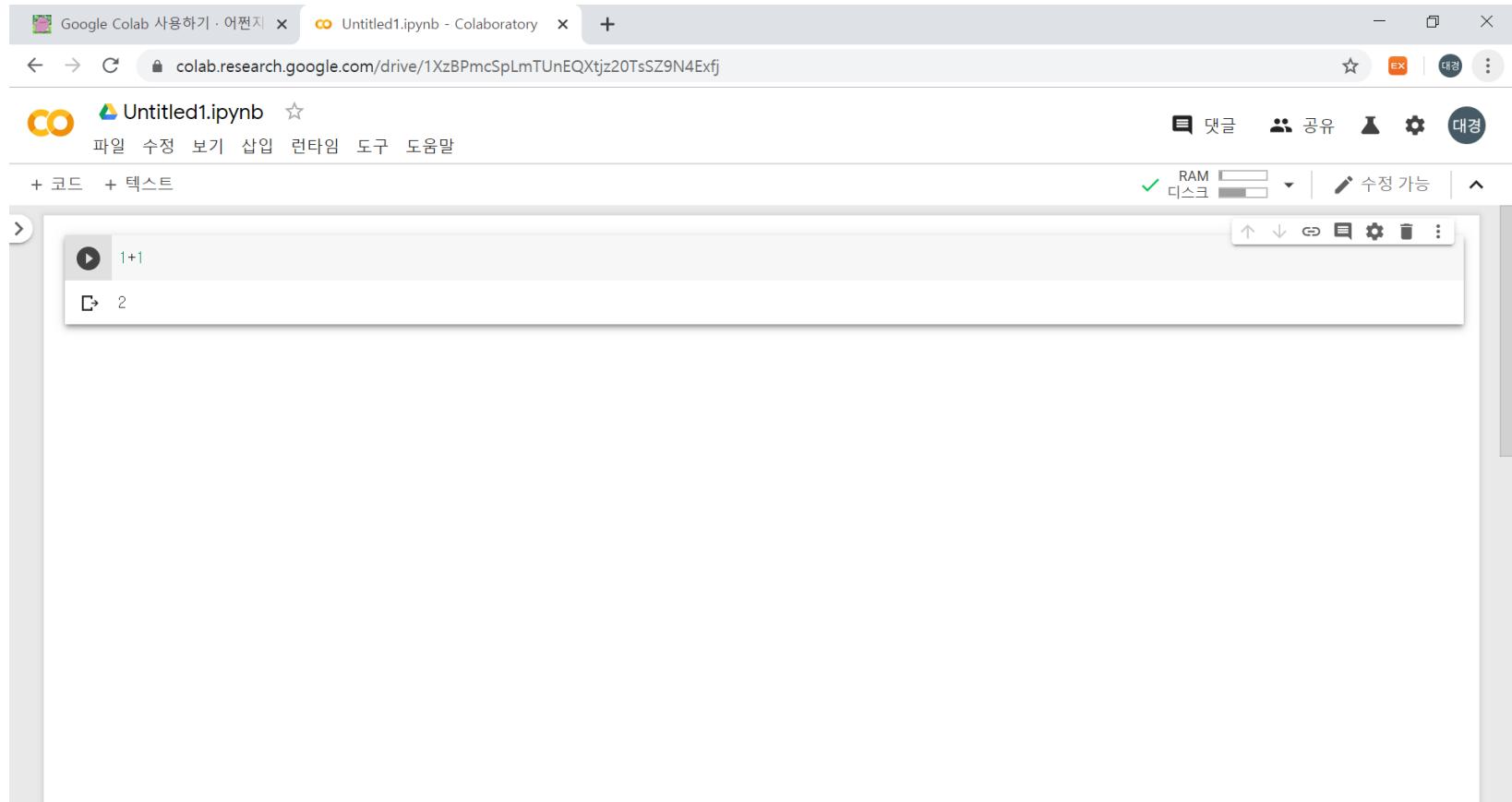
- <https://colab.research.google.com/>로 접속시 사용 가능



2. Pre-Requisites

Google Colab 이란?

- Python3로 편집기 실행





2. Pre-Requisites

Google Colab 리눅스 명령어 사용

- 명령문 앞에 !을 붙인다.

- # OS

```
!cat /etc/issue.net
```

Ubuntu 18.04.2 LTS

- # CPU 사양

```
!head /proc/cpuinfo
```

processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 85
model name : Intel(R) Xeon(R) Platinum 8124M CPU @ 3.00GHz
stepping : 4
microcode : 0x1000157
cpu MHz : 2999.998
cache size : 25344 KB
physical id : 0

- # 메모리 사양

```
!head -n 3 /proc/meminfo
```

MemTotal: 7814692 kB
MemFree: 533060 kB
MemAvailable: 6261620 kB



2. Pre-Requisites

Google Colab 리눅스 명령어 사용

- 명령문 앞에 !을 붙인다.

!df -h

```
----  
Filesystem      Size  Used Avail Use% Mounted on  
none            4.8G  60M  4.5G  2% /  
tmpfs           64M   0  64M  0% /dev  
tmpfs           3.8G   0  3.8G  0% /sys/fs/cgroup  
shm             64M   0  64M  0% /dev/shm  
/dev/nvmeon1p1  117G  69G  49G  59% /goorm  
/dev/nbd1p1    4.8G  60M  4.5G  2% /workspace  
tmpfs           3.8G   0  3.8G  0% /proc/acpi  
tmpfs           3.8G   0  3.8G  0% /proc/scsi  
tmpfs           3.8G   0  3.8G  0% /sys/firmware
```

!python –version

```
----  
Python 3.7.4
```

```
import tensorflow as tf  
print(tf.__version__)
```

```
----  
2.8.2
```



3. 윈10에 Python 환경 구성

Miniconda 다운로드

- 윈10에서 Miniconda Prompt를 이용해 가상환경 만들기
 - 가상환경(Virtual Environment)을 이용하면 Python 버전 간의 의존성을 고려해서 가상의 격리된 환경을 만들어줌으로써 버전이 다름으로 인해 발생할 수 있는 호환이나 충돌 문제를 미연에 방지할 수 있음.
- 1. 윈도우 환경이라면 Miniconda3 Windows 64-bit를 다운로드합니다.
 - <https://docs.conda.io/en/latest/miniconda.html>

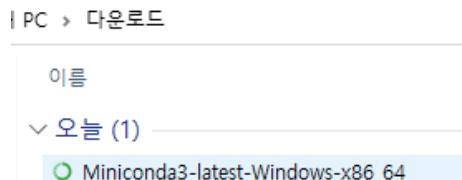
The screenshot shows a web browser displaying the official Miniconda documentation at <https://docs.conda.io/en/latest/miniconda.html>. The page has a green header bar with the Conda logo and navigation links for Conda, Conda-build, and Miniconda. The main content area is titled "System requirements" and lists several bullet points about the software's compatibility with different operating systems and architectures. Below this, there is a note about local vs. system-wide installations. A large red callout box highlights the "Latest Miniconda Installer Links" section, which contains three download links for Windows, macOS, and Linux. The Windows link is specifically highlighted with a red border. At the bottom of the page, there is a table of SHA256 hash values for each installer.

Platform	Name	SHA256 hash
Windows	Miniconda3 Windows 64-bit	1acbc2e8277dd54a5f724896c7edee112d068529588d944702966c867e7e9cc
	Miniconda3 Windows 32-bit	4fb64e6c9c28b88beab16994bfba4829110ea3145baa60bda5344174ab65d462
macOS	Miniconda3 macOS Intel x86 64-bit bash	007baef18dc7b6f2ca6209b5a0c9bd2f283154152f82becf787aac709a51633

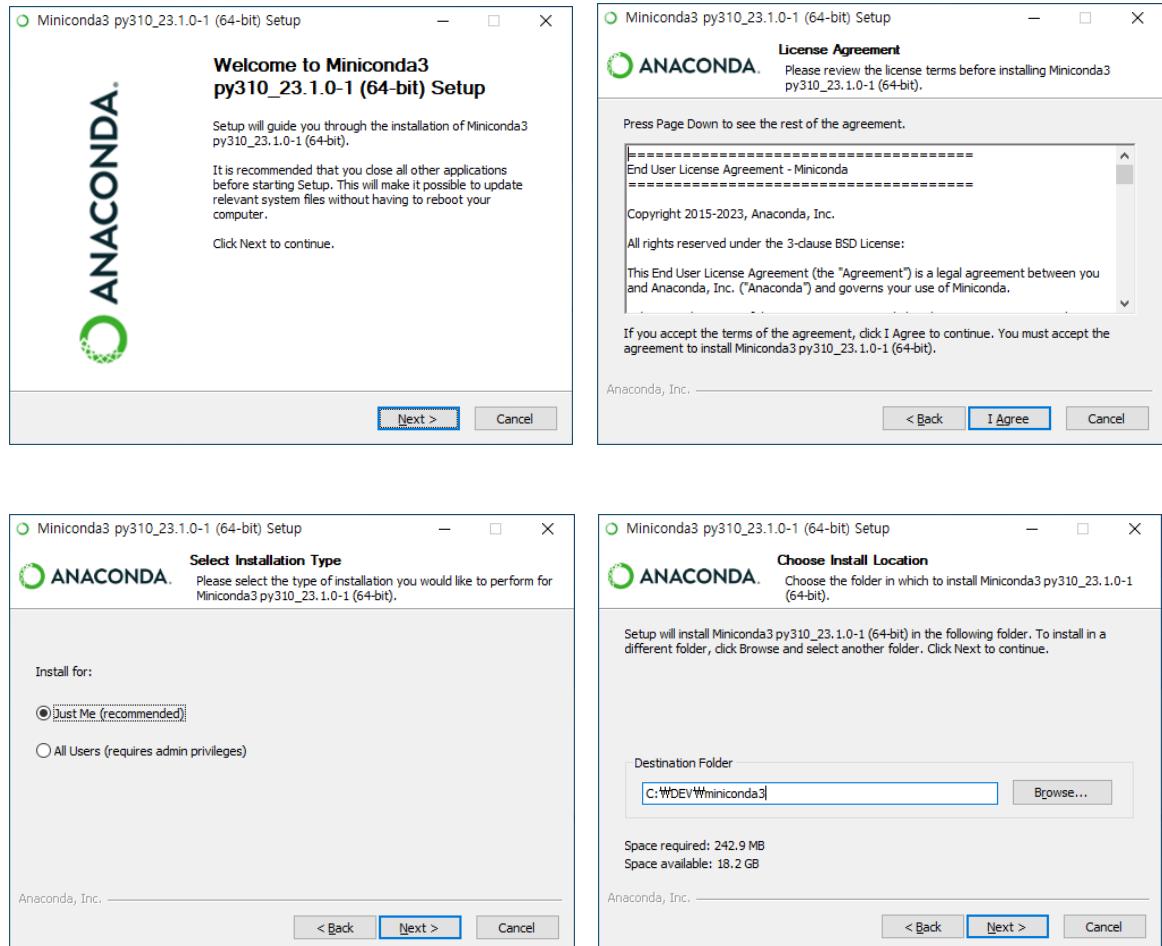
3. 윈10에 Python 환경 구성

Miniconda 설치

● Next 클릭



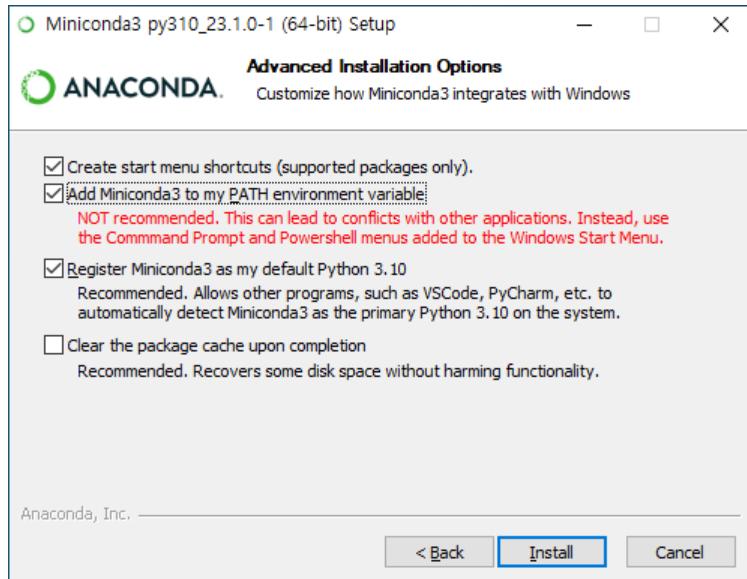
- ❖ All Users 체크 시 Just Me와 설치되는 경로가 다르며, 또한 패키지 설치나 업그레이드 시 CMD(커맨드)창을 관리자 권한으로 실행해야 함
- ❖ C:\DEV\Miniconda3로 경로 설정하고 Next



3. 윈10에 Python 환경 구성

Miniconda 설치

- Register Miniconda3 as my default Python 3.10 '체크하고 Install 클릭

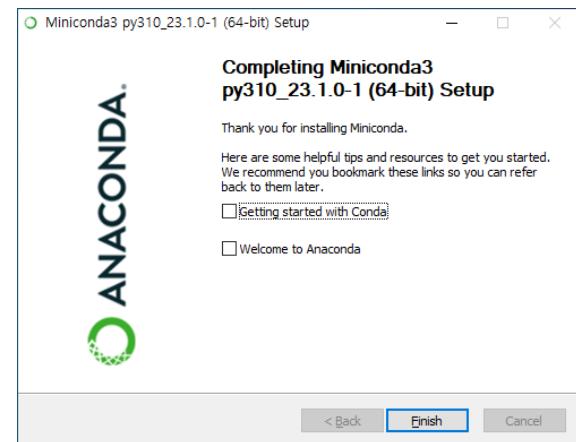
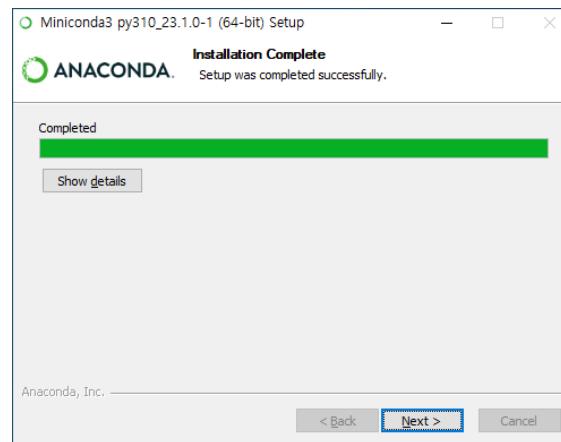
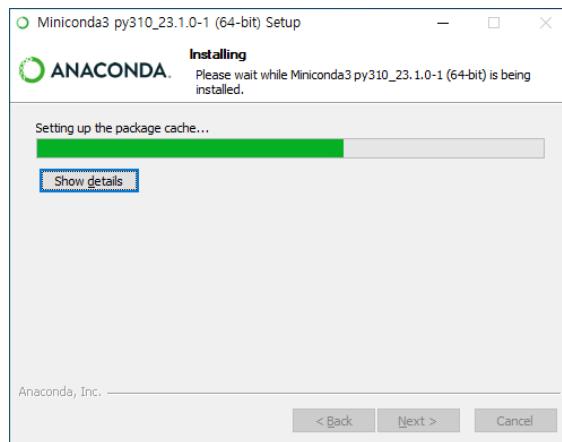


- Add Miniconda3 to my PATH environment variable:
 - PATH 환경 변수에 Miniconda를 추가할지 선택
 - Miniconda 외에 다른 파이썬 인터프리터를 환경변수에 등록해서 사용한다면 체크 해제
 - Miniconda 만을 사용하는 경우, Miniconda가 주력일 경우로 윈도우 cmd 창에서 파이썬을 실행할 경우, 현재 PC에 파이썬을 설치한 적이 없는 경우에는 체크
 - 체크할 경우 윈도우 cmd 창 경로와 상관없이 Miniconda를 파이썬으로 인식
- Register Miniconda3 as my default Python 3.8:
 - Miniconda를 기본 파이썬으로 등록할지 여부를 선택
 - 체크할 경우 개발 도구나 에디터에서 Miniconda를 파이썬으로 인식

3. 윈10에 Python 환경 구성

Miniconda 설치

- Next 클릭

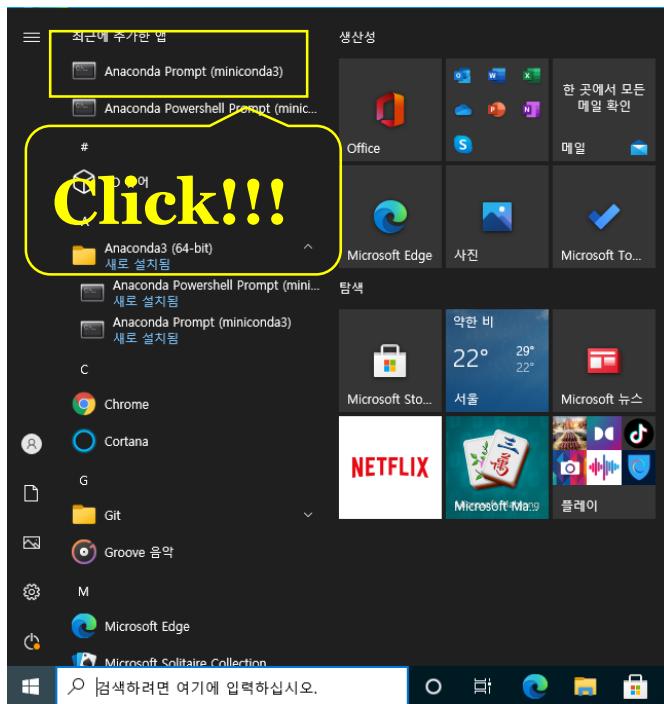




3. 윈10에 Python 환경 구성

Miniconda 설치

2. 원도우키를 누르고 anaconda를 입력시 보이는 Anaconda Prompt를 실행시킵니다. conda 위치를 시스템 변수 PATH에 추가하여 일반 명령프롬프트에서 사용 가능하게 할 수 있지만 Miniconda 공식 문서에서 권장하지 않습니다.





3. 윈10에 Python 환경 구성

Miniconda 설치

(base) C:\Users\k8s>python

Python 3.9.12 (main, Apr 4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.

```
>>> 3+5
8
>>> quit()
```

(base) C:\Users\k8s>

- pip을 업그레이드

(base) C:\Users\k8s>pip install --upgrade pip

Requirement already satisfied: pip in c:\dev\miniconda3\lib\site-packages (22.1.2)

(base) C:\Users\k8s>

- Jupyter notebook 설치

(base) C:\Users\k8s>pip install jupyter

Collecting jupyter
 Downloading jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
Collecting jupyter-console
 Downloading jupyter_console-6.4.4-py3-none-any.whl (22 kB)

...



3. 윈10에 Python 환경 구성

Jupyter notebook 시작

(base) C:\Users\k8s>jupyter notebook

```
[I 13:19:25.919 NotebookApp] Writing notebook server cookie secret to C:\Users\k8s\AppData\Roaming\jupyter\runtime\notebook_cookie_secret
[I 13:19:27.186 NotebookApp] Serving notebooks from local directory: C:\Users\k8s
[I 13:19:27.186 NotebookApp] Jupyter Notebook 6.5.2 is running at:
[I 13:19:27.186 NotebookApp] http://localhost:8888/?token=fo522c7dc31b7fdcoc7c98579bc2132abedeb2fb10293cod
[I 13:19:27.186 NotebookApp] or http://127.0.0.1:8888/?token=fo522c7dc31b7fdcoc7c98579bc2132abedeb2fb10293cod
[I 13:19:27.186 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 13:19:27.278 NotebookApp]
```

To access the notebook, open this file in a browser:

file:///C:/Users/k8s/AppData/Roaming/jupyter/runtime/nbserver-8104-open.html

Or copy and paste one of these URLs:

http://localhost:8888/?token=fo522c7dc31b7fdcoc7c98579bc2132abedeb2fb10293cod

or http://127.0.0.1:8888/?token=fo522c7dc31b7fdcoc7c98579bc2132abedeb2fb10293cod

Name	Last Modified	File size
2년 전		
2년 전		
6달 전		
6달 전		
2시간 전		
2년 전		
2년 전		
2년 전		
9년 전		



3. 윈10에 Python 환경 구성

가상환경 만들기

(base) C:\Users\k8s>conda create -n py38_basic pip python=3.8

Collecting package metadata (current_repodata.json): done
Solving environment: done

```
## Package Plan ##

environment location: C:\DEV\miniconda3\envs\py38_basic

added / updated specs:
- pip
- python=3.8
```

...

Proceed ([y]/n)? y

...

```
done
#
# To activate this environment, use
#
# $ conda activate py38_basic
#
# To deactivate an active environment, use
#
# $ conda deactivate
```

(base) C:\Users\k8s>conda env list

```
# conda environments:
#
base          * C:\DEV\miniconda3
py38_basic      C:\DEV\miniconda3\envs\py38_basic
```

(base) C:\Users\k8s>



3. 윈10에 Python 환경 구성

가상환경 만들기

```
(base) C:\Users\k8s>conda activate py38_basic
```

```
(py38_basic) C:\Users\k8s>pip install ipykernel
```

```
Collecting ipykernel
  Using cached ipykernel-6.21.2-py3-none-any.whl (149 kB)
Collecting tornado>=6.1
  Using cached tornado-6.2-cp37-abi3-win_amd64.whl (425 kB)
Collecting matplotlib-inline>=0.1
  Using cached matplotlib_inline-0.1.6-py3-none-any.whl (9.4 kB)
Collecting packaging
  Using cached packaging-23.0-py3-none-any.whl (42 kB)
Collecting pyzmq>=20
  Downloading pyzmq-25.0.0-cp38-cp38-win_amd64.whl (975 kB)
----- 975.6/975.6 kB 8.8 MB/s eta 0:00:00
```

...

```
Successfully installed asttokens-2.2.1 backcall-0.2.0 colorama-0.4.6 comm-0.1.2 debugpy-1.6.6 decorator-5.1.1 executing-1.2.0 importlib-metadata-6.0.0 ipykernel-6.21.2 ipython-8.10.0 jedi-0.18.2 jupyter-client-8.0.3 jupyter-core-5.2.0 matplotlib-inline-0.1.6 nest-asyncio-1.5.6 packaging-23.0 parso-0.8.3 pickleshare-0.7.5 platformdirs-3.0.0 prompt-toolkit-3.0.36 psutil-5.9.4 pure-eval-0.2.2 pygments-2.14.0 python-dateutil-2.8.2 pywin32-305 pyzmq-25.0.0 six-1.16.0 stack-data-0.6.2 tornado-6.2 traitlets-5.9.0 wewidth-0.2.6 zipp-3.13.0
```

```
(py38_basic) C:\Users\k8s>python -m ipykernel install --user --name py38_basic --display-name "py38_basic"
```

```
Installed kernelspec py38_basic in C:\Users\k8s\AppData\Roaming\jupyter\kernels\py38_basic
```

```
(py38_basic) C:\Users\k8s>
```

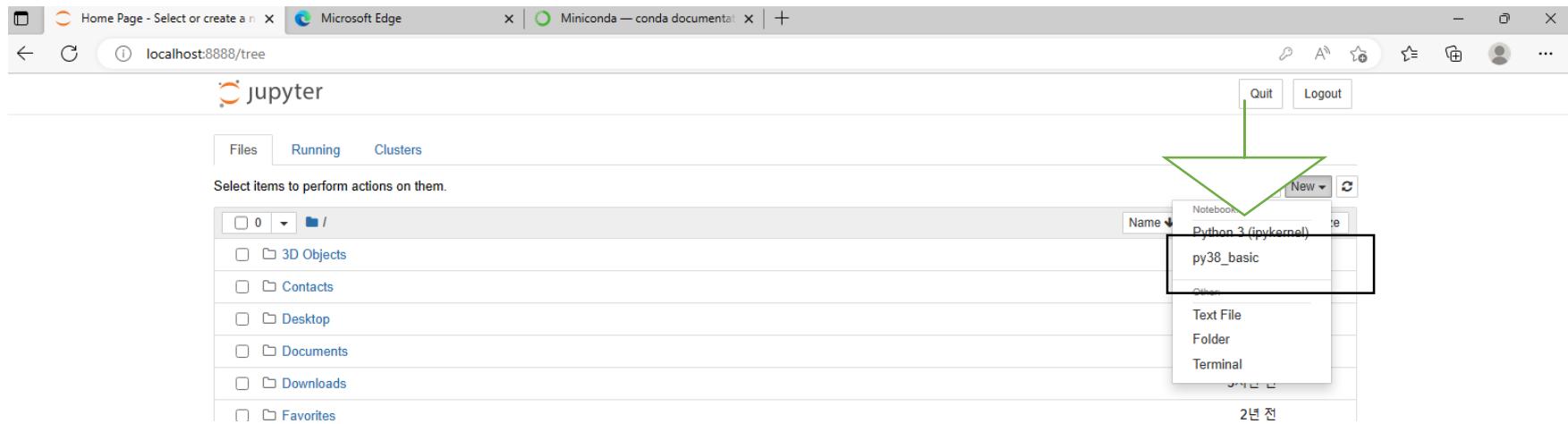
3. 윈10에 Python 환경 구성

가상환경 만들기

(py38_basic) C:\Users\k8s>conda deactivate

(base) C:\Users\k8s>jupyter notebook

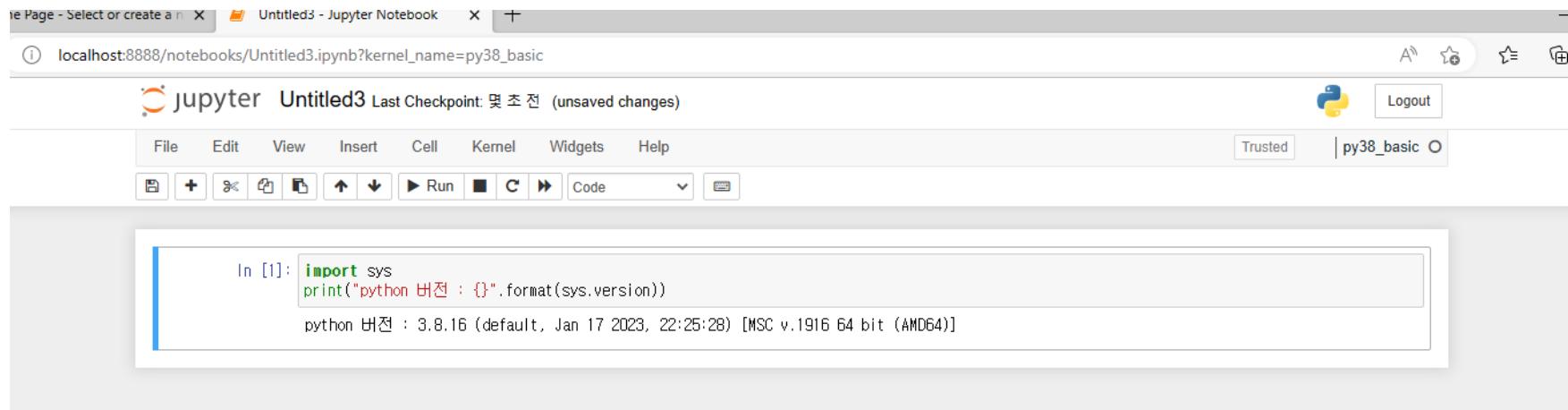
```
[I 14:02:34.315 NotebookApp] Serving notebooks from local directory: C:\Users\k8s
[I 14:02:34.315 NotebookApp] Jupyter Notebook 6.5.2 is running at:
[I 14:02:34.315 NotebookApp] http://localhost:8888/
[I 14:02:34.315 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```



3. 윈10에 Python 환경 구성

가상환경 테스트

```
import sys  
print("python 버전 : {}".format(sys.version))
```



The screenshot shows a Jupyter Notebook interface running in a web browser. The title bar indicates the page is 'Untitled3 - Jupyter Notebook'. The URL in the address bar is 'localhost:8888/notebooks/Untitled3.ipynb?kernel_name=py38_basic'. The main window displays a single code cell with the following content:

```
In [1]: import sys  
print("python 버전 : {}".format(sys.version))
```

When run, the cell outputs:

```
python 버전 : 3.8.16 (default, Jan 17 2023, 22:25:28) [MSC v.1916 64 bit (AMD64)]
```



3. 윈10에 Python 환경 구성

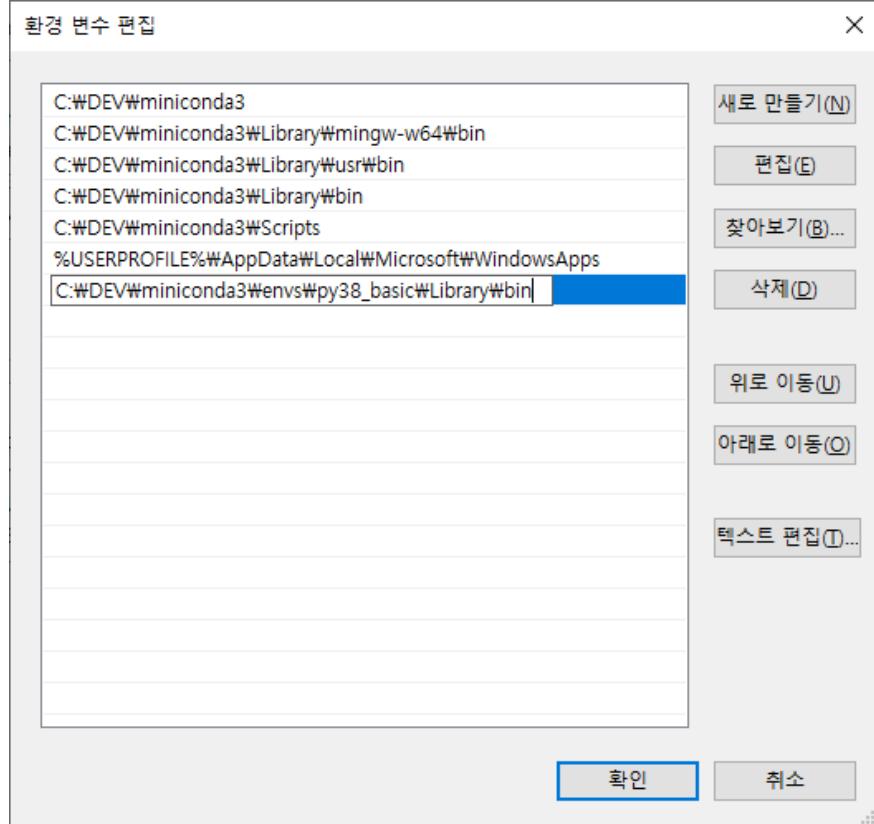
'사용자 변수'에 있는 Path에 다음과 같이 설정

Miniconda3

Miniconda3\Library\bin

Miniconda3\Scripts

C:\DEV\miniconda3\envs\py38_basic\Library\bin





3. 윈10에 Python 환경 구성

가상환경 삭제

conda remove --name 가상환경이름 –all

conda env remove -n 가상환경이름



4. flask 가상환경

flask 가상환경 생성

```
(base) C:\Users\k8s>conda create -n flask_37 python=3.7
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

```
...
```

```
done
#
# To activate this environment, use
#
# $ conda activate flask_37
#
# To deactivate an active environment, use
#
# $ conda deactivate
```

```
(base) C:\Users\k8s>conda activate flask_37
```

```
(flask_37) C:\Users\k8s>conda list
```

```
# packages in environment at C:\DEV\miniconda3\envs\flask_37:
```

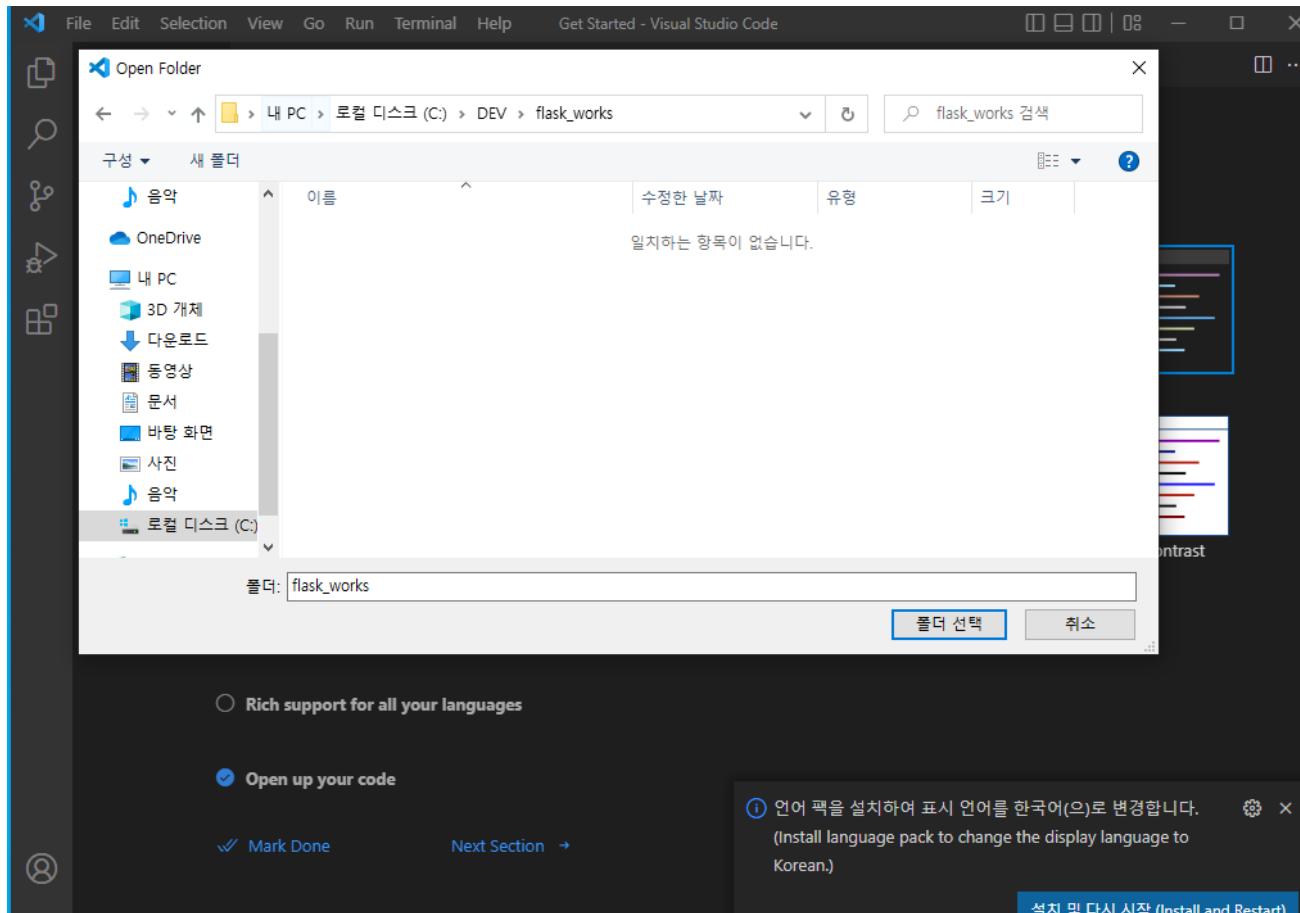
```
#
# Name      Version    Build Channel
ca-certificates 2022.4.26  haa95532_0
certifi        2022.6.15  py37haa95532_0
openssl         1.1.10   h2bbff1b_0
pip            21.2.4    py37haa95532_0
python          3.7.13   h6244533_0
setuptools      61.2.0    py37haa95532_0
sqlite          3.38.5   h2bbff1b_0
vc              14.2     h21ff451_1
vs2015_runtime 14.27.29016 h5e58377_2
wheel           0.37.1    pyhd3eb1bo_0
wincertstore    0.2      py37haa95532_2
```

```
(flask_37) C:\Users\k8s>conda deactivate
```

5. Visual Studio Code

Visual Studio Code 사용

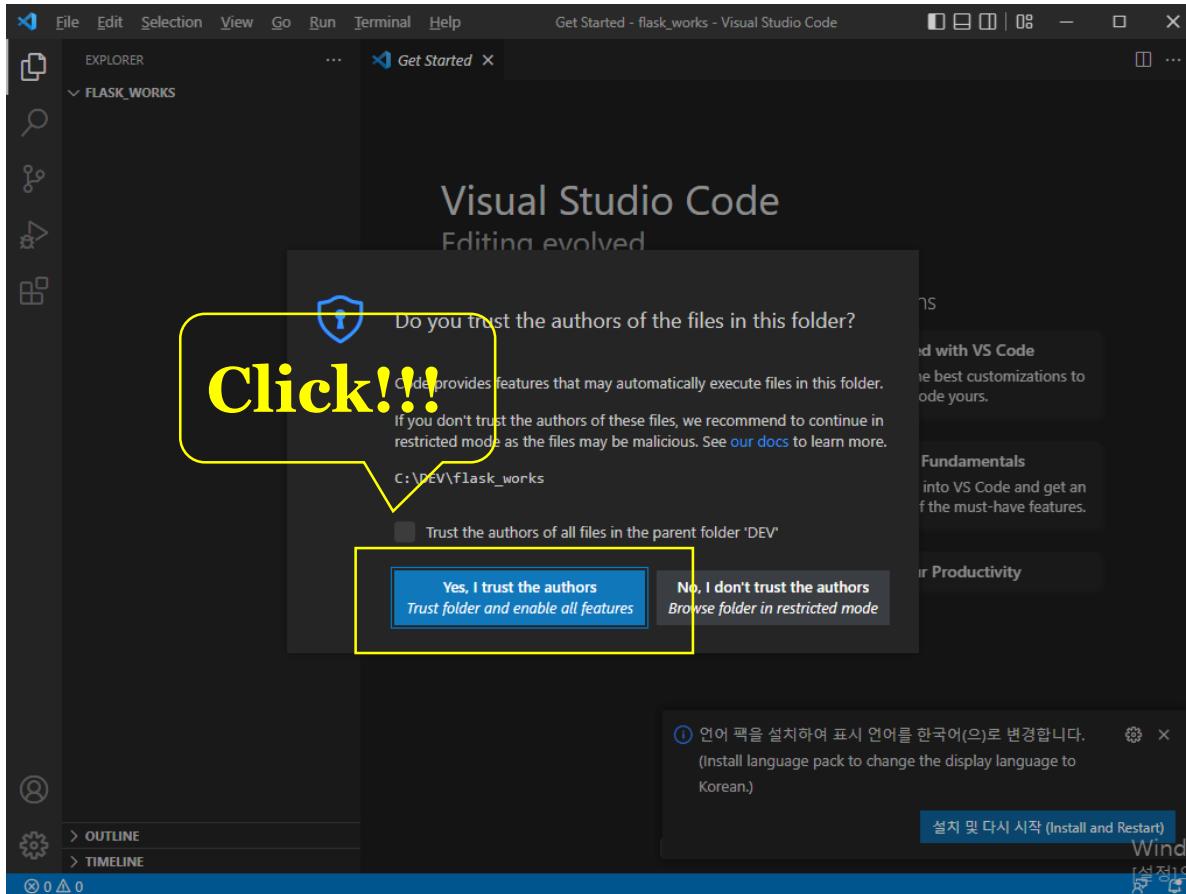
- Visual Studio Code를 실행하여 메뉴에서 File > Open Folder를 선택



5. Visual Studio Code

Visual Studio Code 사용

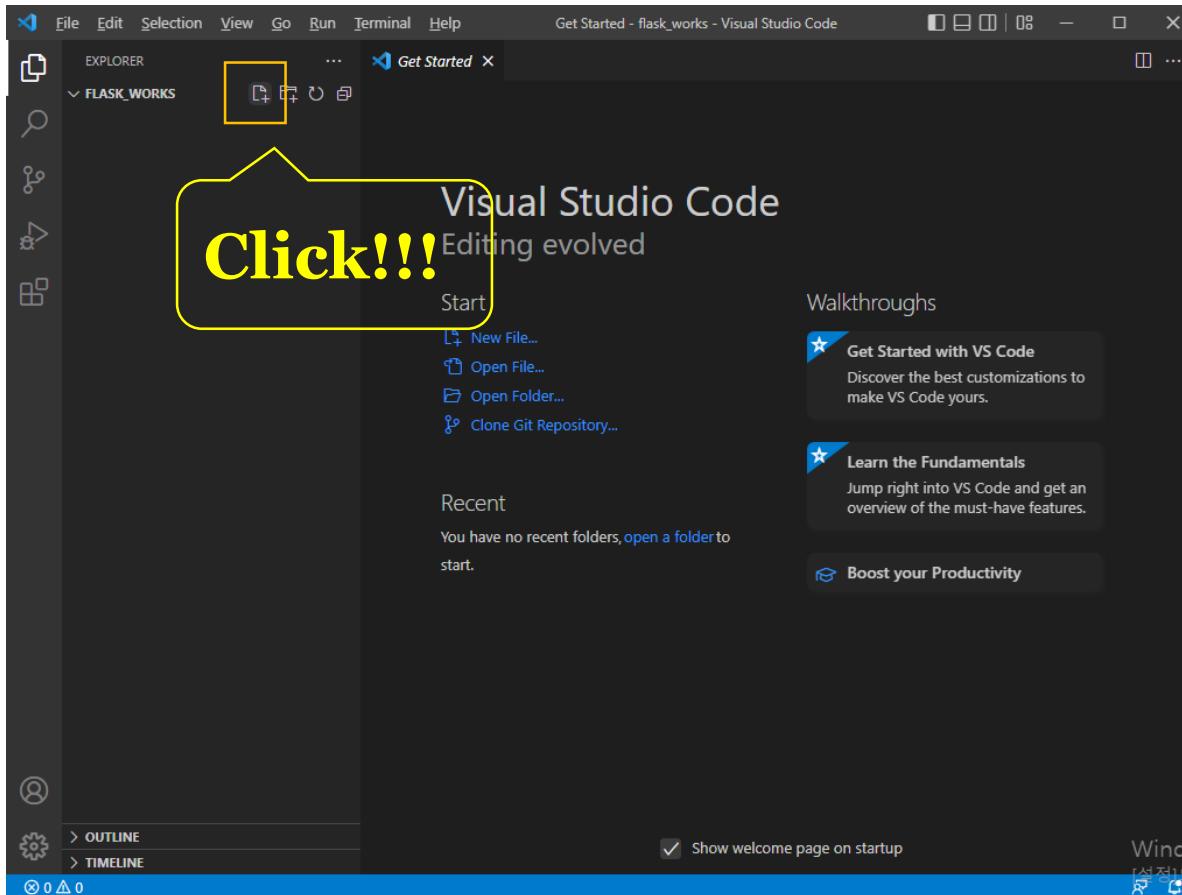
- Yes, I trust the authors 클릭



5. Visual Studio Code

Visual Studio Code 사용

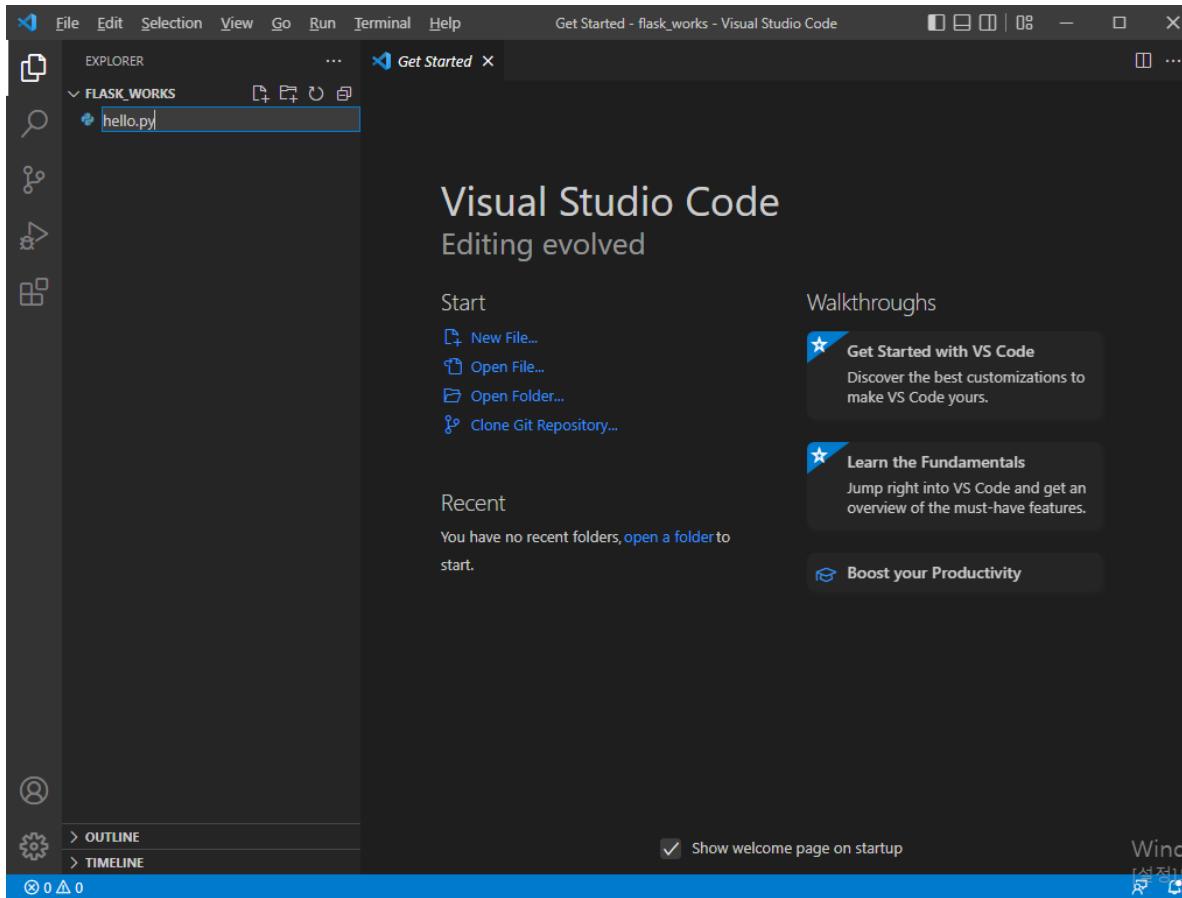
- 해당 폴더가 Visual Studio Code 왼쪽에 보이게 됩니다. New File 아이콘을 클릭



5. Visual Studio Code

Visual Studio Code 사용

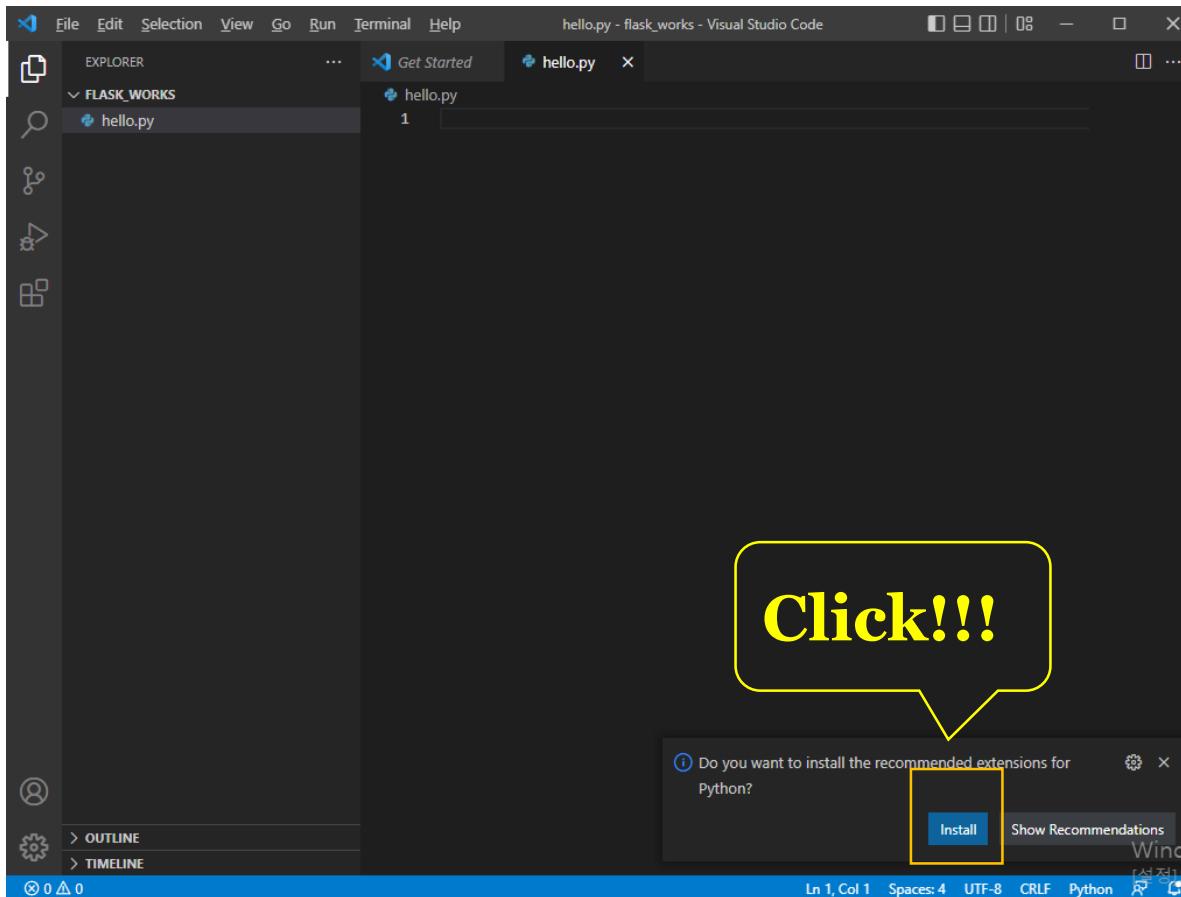
- `hello.py`를 입력 후, 엔터를 누르면 해당 파일이 추가되면서 오른쪽에 `hello.py` 파일이 열리게 됩니다.



5. Visual Studio Code

Visual Studio Code 사용

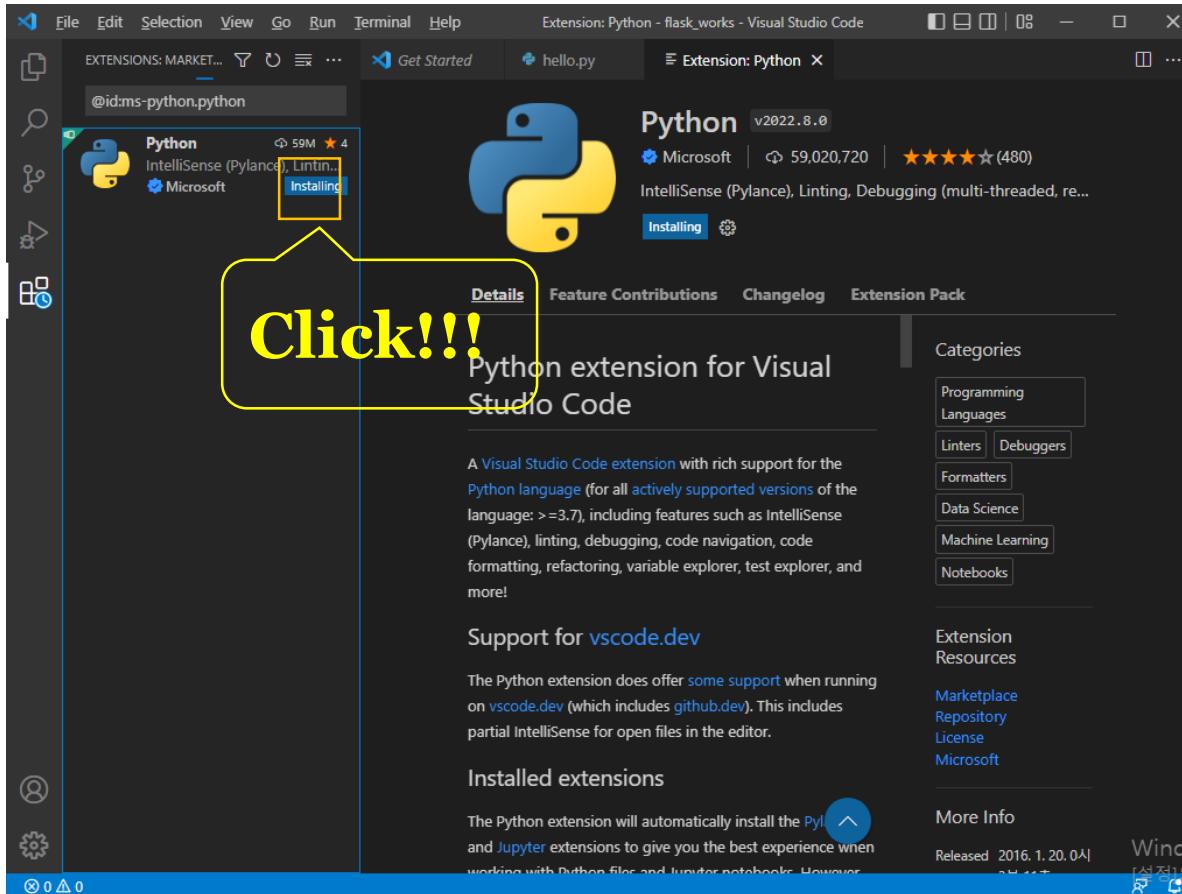
- 오른쪽 아래에 Python 확장을 설치하는지 물어보는 메시지 박스가 보이면 설치



5. Visual Studio Code

Visual Studio Code 사용

- 오른쪽 아래에 Python 확장을 설치하는지 물어보는 메시지 박스가 보이면 설치



5. Visual Studio Code

Visual Studio Code 사용

- 올바르게 인터프리터가 설정이 되었는지 확인해봅니다.

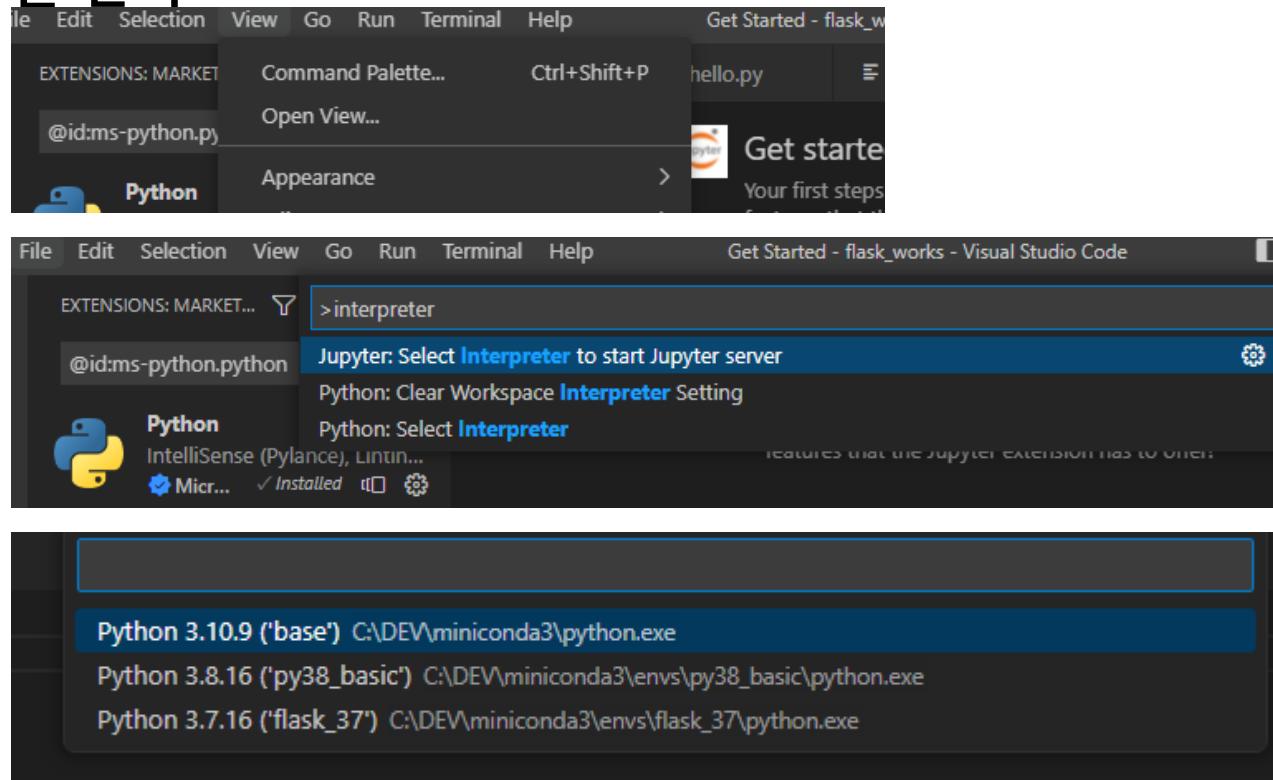
The screenshot shows the Visual Studio Code interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. A status bar at the top right indicates "hello.py - flask_works - Visual Studio Code". The left sidebar contains icons for file explorer, search, and other tools. The main workspace shows a single file named "hello.py" with the number "1" on the first line. Below the workspace, the Python extension is visible, showing it's active and providing IntelliSense (Pylance), Linting, and Microsoft support. At the bottom, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is selected, displaying command-line output:

```
C:\DEV\flask_works>C:/DEV/miniconda3/Scripts/activate.bat  
(base) C:\DEV\flask_works>conda env list  
# conda environments:  
#  
base * C:\DEV\miniconda3  
flask_37 C:\DEV\miniconda3\envs\flask_37  
nv38_basic C:\DEV\miniconda3\envs\nv38_basic
```

5. Visual Studio Code

Visual Studio Code 사용

- 인터프리터 환경 설정이 원활치 않을 때, Conda에서 생성한 가상환경의 인터프리터로 변경해야 합니다.
- Ctrl + Shift + P를 누른 후, interpreter를 입력하여 검색되는 다음 항목을 선택





5. Visual Studio Code

가상 환경 변경

(base) C:\DEV\flask_works>conda activate flask_37

(flask_37) C:\DEV\flask_works>

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

(base) C:\DEV\flask_works>conda env list
# conda environments:
#
base          *  C:\DEV\miniconda3
flask_37      C:\DEV\miniconda3\envs\flask_37
py38_basic    C:\DEV\miniconda3\envs\py38_basic

(base) C:\DEV\flask_works>conda activate flask_37
(flask_37) C:\DEV\flask_works>
```



5. Visual Studio Code

flask 패키지 설치

(flask_37) C:\DEV\flask_works>pip install flask

Collecting flask

 Downloading Flask-2.2.3-py3-none-any.whl (101 kB)

101.8/101.8 kB 1.5 MB/s eta 0:00:00

Collecting importlib-metadata>=3.6.0

...

Installing collected packages: zipp, typing-extensions, MarkupSafe, itsdangerous, colorama, Werkzeug, Jinja2, importlib-metadata, click, flask

Successfully installed Jinja2-3.1.2 MarkupSafe-2.1.2 Werkzeug-2.2.3 click-8.1.3 colorama-0.4.6 flask-2.2.3 importlib-metadata-6.0.0 itsdangerous-2.1.2 typing-extensions-4.5.0
zipp-3.13.0

(flask_37) C:\DEV\flask_works>



5. Visual Studio Code

flask 패키지 설치 테스트

- **hello.py** 작성

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```



5. Visual Studio Code

flask 패키지 설치 테스트

- 터미널에서 다음 명령을 실행

(flask_37) C:\DEV\flask_works>dir

```
C 드라이브의 볼륨에는 이름이 없습니다.  
볼륨 일련 번호: BEDo-C858
```

```
C:\DEV\flask_works 디렉터리
```

```
2022-06-25 오후 03:46 <DIR> .
2022-06-25 오후 03:46 <DIR> ..
2022-06-25 오후 03:16 0 hello.py
2022-06-25 오후 03:46 <DIR> __pycache__
    1개 파일 0 바이트
    3개 디렉터리 12,450,320,384 바이트 남음
```

(flask_37) C:\DEV\flask_works>set FLASK_APP=hello

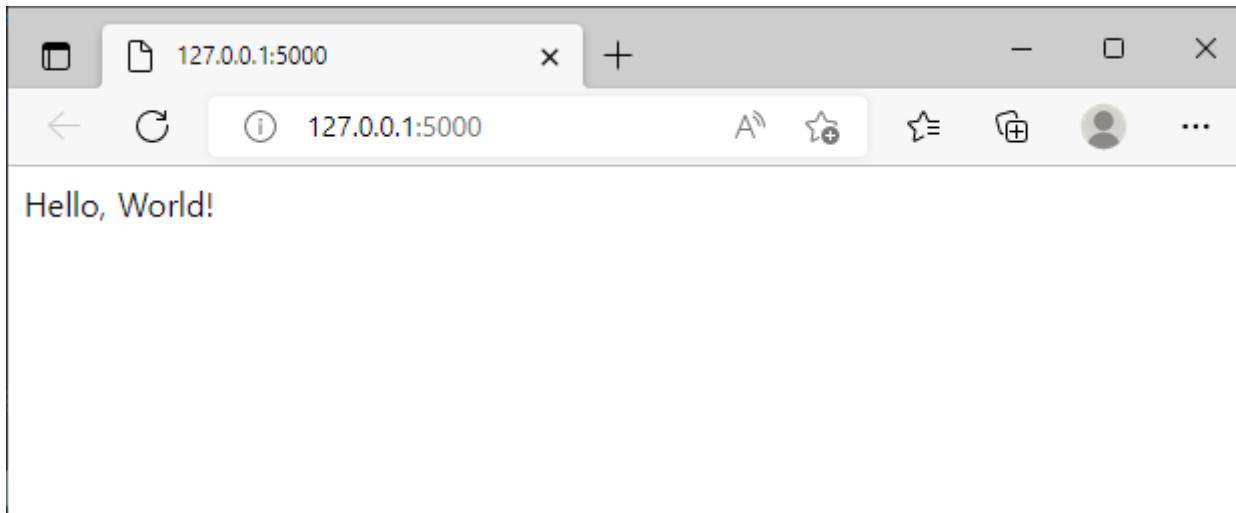
(flask_37) C:\DEV\flask_works>flask run

```
* Serving Flask app 'hello' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
```

5. Visual Studio Code

flask 패키지 설치 테스트

- **127.0.0.1:5000**





6. 파이썬 입력 방식

REPL(Read–Eval–Print Loop) 방식

- REPL(Read–Eval–Print Loop, '레플'이라고 읽는다) 방식은 주피터 노트북이나 파이썬(또는 IPython) 콘솔을 실행해서 명령어를 한 줄씩 입력하며 실행 상황을 지켜보는 방식이다. 파이썬 프로그래밍을 공부하거나 코드의 초기 버전을 제작할 때 유용한 방식이다.
- Ipython

```
(base) C:\Users\k8s>ipython
Python 3.10.9 | packaged by conda-forge | (main, Mar 28 2023, 16:40:45) [GCC 11.2.0]
Type 'copyright', 'credits' or 'license' for additional copyright notices
IPython 8.10.0 -- An enhanced Interactive Python environment

In [1]: 1 + 2
Out[1]: 3

In [2]: quit

(base) C:\Users\k8s>
```

- Jupyter Notebook

```
!python --version
Python 3.8.16
```

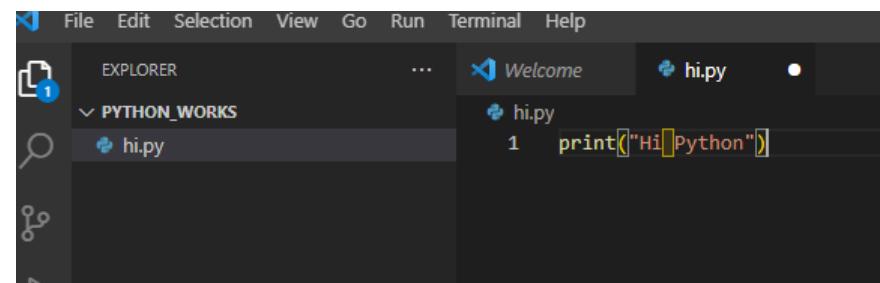
6. 파이썬 입력 방식

스크립트 실행 방식

- 코드를 모두 완성한 후 전체 코드를 한 번에 실행하고 싶을 때는 코드를 .py 확장자 파일로 만든다. 이렇게 만든 코드 파일을 스크립트 (script)라고 한다. 스크립트는 주피터 노트북이나 아이파이썬 (IPython) 콘솔을 실행하지 않고 터미널에서 직접 실행할 수 있다.
- 다만 스크립트에서는 변수 이름만 입력해서 변수의 값, 즉 표현 (representation)이 표시되게 하는 것은 불가능하므로 화면에 무언가를 표시하려면 반드시 명시적으로 print 명령을 사용해야 한다.
- Visual Studio Code

```
(base) C:\DEV\python_works>conda env list
```

```
# conda environments:  
#  
base      * C:\DEV\miniconda3  
flask_37   C:\DEV\miniconda3\envs\flask_37  
py38_basic C:\DEV\miniconda3\envs\py38_basic
```



```
(base) C:\DEV\python_works>conda activate py38_basic
```

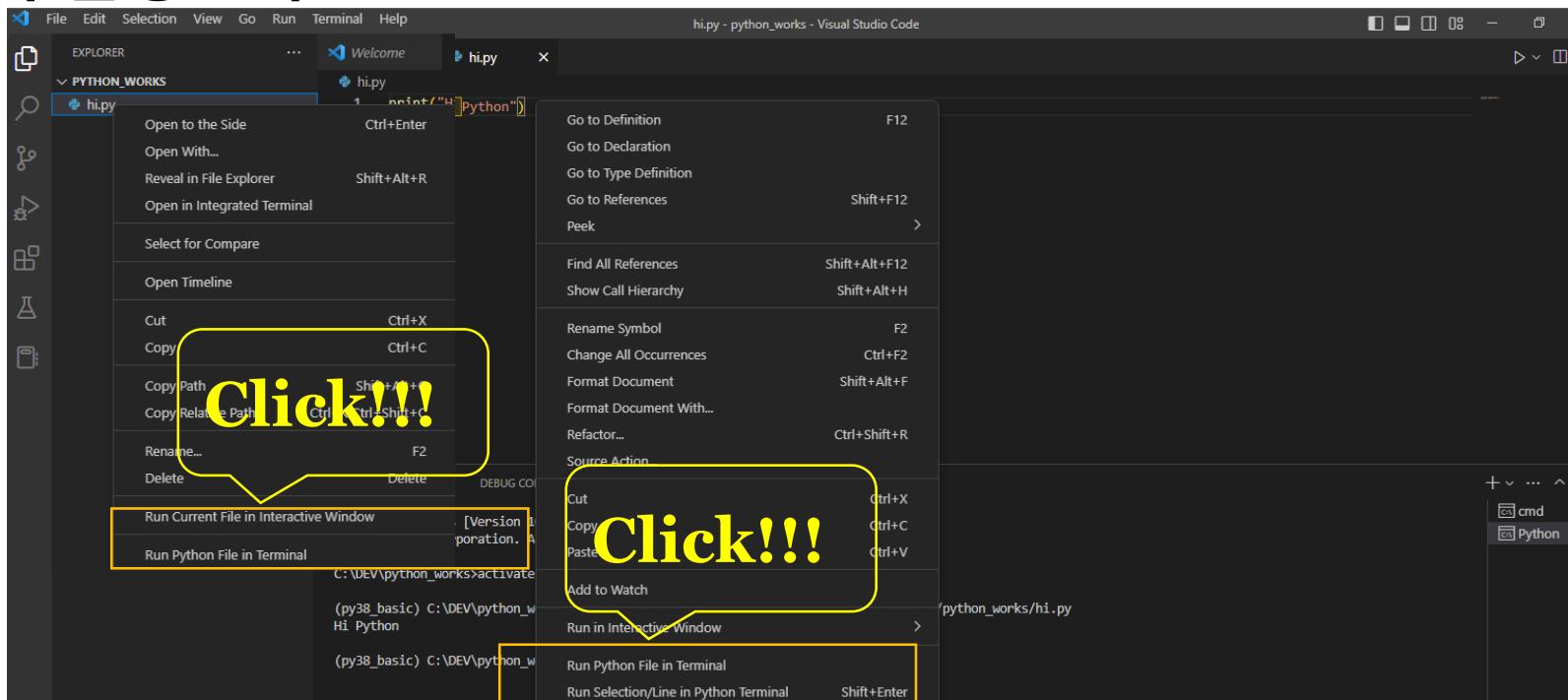
```
(py38_basic) C:\DEV\python_works>
```



6. 파이썬 입력 방식

스크립트 실행 방식

- 특히, Visual Studio Code에서 코드를 실행할 때 **Ctrl + Shift + P**를 누른 후, interpreter를 입력하여 선택한 것을 기본으로 하기 때문에 실행 가능한 가상환경을 선택한 후 코드를 실행해야 한다. 그렇지 않으면 기본으로 선택된 가상환경 아래에서 실행된다.





6. 파이썬 입력 방식

몇 가지 코딩 규칙

● Jupyter notebook에서 실습

(py38_basic) C:\DEV\python_works>jupyter notebook

```
[I 09:07:12.600 NotebookApp] Serving notebooks from local directory: C:\DEV\python_works
[I 09:07:12.621 NotebookApp] Jupyter Notebook 6.5.2 is running at:
[I 09:07:12.621 NotebookApp] http://localhost:8888/
[I 09:07:12.621 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[I 09:07:48.735 NotebookApp] 302 GET /tree (::1) 22.83000oms
[I 09:08:07.516 NotebookApp] 302 POST /login?next=%2Ftree (::1) 786.49000oms
```

localhost:8888/tree

jupyter

Files Running Clusters

Select items to perform actions on them.

0 /

hi.py

Click!!!

Upload New

Notebook: Python 3 (ipykernel)

Name: py38_basic 9 B

Other: Text File Folder Terminal



6. 파이썬 입력 방식

몇 가지 코딩 규칙

- 띄어쓰기 규칙

- 코드의 처음과 마지막에는 빈칸을 넣지 않는다.
- 빈칸은 항상 하나만 넣는다.
- 숫자나 연산 기호 사이는 한 칸을 띄운다.
- 괄호를 연 직후나 닫기 직전에는 빈칸을 넣지 않는다.
- # 기호는 주석(설명문, comment)을 뜻하는 기호

▶ 1 + 1

▷ 2

▶ 1+1 # 모두 불여쓰지 않는다.

▷ 2

▶ 1 + 1 # 너무 많이 띄우지 않는다. 한 칸만 띄운다.

2



6. 파이썬 입력 방식

몇 가지 코딩 규칙

● 예약어 (Reserved Words):

- 예약어는 상수 또는 변수나 다른 식별자의 이름으로 사용 할 수 없다.
- 예약어는 모두 소문자이다.

```
import keyword  
keyword.kwlist
```

```
['False',  
'None',  
'True',  
'and',  
'as',  
'assert',  
'async',  
'await',  
'break',  
'class',  
'continue',  
'def',  
'del',  
'elif',  
'else',  
'except',  
'finally',  
'for',  
'from',  
'global',  
'if',  
'import',  
'in',  
'is',  
'lambda',  
'nonlocal',  
'not',  
'or',  
'pass',  
'raise',  
'return',  
'try',  
'while',  
'with',  
'yield']
```



6. 파이썬 입력 방식

몇 가지 코딩 규칙

● 줄 들여쓰기 (Lines and Indentation)

- 파이썬 프로그램 시 첫번째 주의 사항.
- 클래스와 함수 정의시 코드 블럭을 표시하거나 흐름 제어 할 중괄호가 없다.
- 코드블럭은 줄 들여쓰기에 의해 엄격히 적용 된다.
- 들여쓰기는 공백(Space), 탭(Tab) 모두 가능.
- 단, 들여쓰기 공백 수는 블럭내에 동일 수로 해야 함.

```
if True:  
    print("True")  
else:  
    print("False")
```

True

정상!!!

```
if True:  
    print("Answer")  
    print("True")  
else:  
    print("Answer")  
    print("False")  
  
File <stokenize>:6  
      print("False")  
      ^
```

에러!!!

IndentationError: unindent does not match any outer indentation level



6. 파이썬 입력 방식

몇 가지 코딩 규칙

- 멀티 라인 문 (Multi-Line Statements)

- 파이썬은 일반적으로 새로운 줄이 되면 끝난다. 그러나 줄 연속 문자(\)의 사용으로 줄을 계속 사용할 수 있다.

```
total = 'item_one' + #
       'item_two' + #
       'item_three'

print(total)
```

item_one, item_two, item_three

- [], {}에 포함 된 문이나 () 괄호는 줄 연속 문자를 사용할 필요가 없다.

```
days = ['Monday', 'Tuesday',
        'Wednesday', 'Thursday', 'Friday']
print(days)
```

['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']



6. 파이썬 입력 방식

몇 가지 코딩 규칙

- 멀티 라인 문 (Multi-Line Statements)

- 파이썬은 일반적으로 새로운 줄이 되면 끝난다. 그러나 줄 연속 문자(\)의 사용으로 줄을 계속 사용할 수 있다.

```
total = 'item_one' + #
       'item_two' + #
       'item_three'

print(total)
```

item_one, item_two, item_three

- [], {}에 포함 된 문이나 () 괄호는 줄 연속 문자를 사용할 필요가 없다.

```
days = ['Monday', 'Tuesday',
        'Wednesday', 'Thursday', 'Friday']
print(days)
```

['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']



6. 파이썬 입력 방식

몇 가지 코딩 규칙

● 인용 (Quotation)

- `single(''), double("")`, `triple("'" or "'''')` 를 사용하여 문자열을 감싸서 사용한다.
- `triple` 따옴표는 여러 줄에 걸쳐 문자열을 사용할 수 있다.

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
print(word)
print(sentence)
print(paragraph)
```

```
word
This is a sentence.
This is a paragraph. It is
made up of multiple lines and sentences.
```



6. 파이썬 입력 방식

몇 가지 코딩 규칙

- 주석 (Comments)

- 기호 (#)를 사용해 주석을 시작한다.
- 기호 (#) 후 물리적 라인 끝까지 모든 문자를 주석으로 간주한다.

```
# First comment
print("Hello, Python!") # second comment
# last comment
```

Hello, Python!



6. 파이썬 입력 방식

몇 가지 코딩 스타일

- PEP 8 – Style Guide for Python Code
- <https://peps.python.org/pep-0008/>

Python Enhancement Proposals

Python » PEP Index » PEP 8



Contents

- Introduction
- A Foolish Consistency is the Hobgoblin of Little Minds
- Code Lay-out
 - Indentation
 - Tabs or Spaces?
 - Maximum Line Length
 - Should a Line Break Before or After a Binary Operator?
 - Blank Lines
 - Source File Encoding
 - Imports
 - Module Level Dunder Names
- String Quotes
- Whitespace in Expressions and Statements
 - Pet Peeves
 - Other Recommendations
- When to Use Trailing Commas
- Comments
 - Block Comments
 - Inline Comments
 - Documentation Strings
- Naming Conventions

PEP 8 – Style Guide for Python Code

Author: Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>

Status: Active

Type: Process

Created: 05-Jul-2001

Post-History: 05-Jul-2001, 01-Aug-2013

▶ Table of Contents

Introduction

This document gives coding conventions for the Python code comprising the standard library in the main Python distribution. Please see the companion informational PEP describing [style guidelines for the C code in the C implementation of Python](#).

This document and [PEP 257](#) (Docstring Conventions) were adapted from Guido's original Python Style Guide essay, with some additions from Barry's style guide [2].

This style guide evolves over time as additional conventions are identified and past conventions are rendered obsolete by changes in the language itself.

Many projects have their own coding style guidelines. In the event of any conflicts, such project-specific guides take precedence for that project.



6. 파이썬 입력 방식

몇 가지 코딩 스타일

- **Google Style Guides**

- <https://google.github.io/styleguide/>
[styleguide](https://google.github.io/styleguide/)

Google Style Guides

Every major open-source project has its own style guide: a set of conventions (sometimes arbitrary) about how to write code for that project. It is much easier to understand a large codebase when all the code in it is in a consistent style.

"Style" covers a lot of ground, from "use camelCase for variable names" to "never use global variables" to "never use exceptions." This project ([google/styleguide](https://google.github.io/styleguide/)) links to the style guidelines we use for Google code. If you are modifying a project that originated at Google, you may be pointed to this page to see the style guides that apply to that project.

- [AngularJS Style Guide](#)
- [Common Lisp Style Guide](#)
- [C++ Style Guide](#)
- [C# Style Guide](#)
- [Go Style Guide](#)
- [HTML/CSS Style Guide](#)
- [JavaScript Style Guide](#)
- [Java Style Guide](#)
- [Objective-C Style Guide](#)
- [Python Style Guide](#)

6. 파이썬 입력 방식

몇 가지 코딩 스타일

- Airbnb JS Style Guides
- <https://github.com/Airbnb/javascript>

The screenshot shows the GitHub repository page for `airbnb/javascript`. The repository is public, has 25k forks, and 132k stars. The master branch is selected. The main navigation bar includes Code, Issues (97), Pull requests (61), Actions, Wiki, Security, and Insights. On the right, there's an 'About' section for the JavaScript Style Guide, listing tags like javascript, styleguide, eslint, es6, style-linter, linting, style-guide, es2015, naming-conventions, arrow-functions, es2017, es2016, tc39, and es2018. Below the navigation, a list of recent pull requests is displayed:

Author	Commit Message	Time Ago
syedmouaazfarrukh and ljharb	[readme] added JavaScript style guide	last week
	[Tests] re-enable tests disabled for the eslint configuration	2 years ago
	[guide] [css] Fixed Italic subtitle in css-in-js README	6 years ago
	[Docs] HTTP => HTTPS	2 years ago
	[eslint config] [deps] update eslint-plugin-javascript	last week
	[eslint config] [patch] Fixed handle and on... (Patch)	3 years ago



6. 파이썬 입력 방식

몇 가지 코딩 스타일

● NHN Coding Convention

- <https://nuli.navercorp.com/tool/codingCon>

The screenshot shows the homepage of the NULI website. At the top, there is a navigation bar with links for 체험 (Experience), 지침 (Guidelines), 교육 (Education), 세미나 (Seminar), 커뮤니티 (Community), and **개발도구** (Development Tools). Below the navigation bar is a large blue banner with the text "기술로 동행하다." (Traveling with technology) and "널리 사람을 이롭게 하기 위한 다양한 기술로 접근성을 실천하고 싶습니다." (Technology Doesn't Leave Anyone Behind). To the right of the banner, there is an illustration of four people working together on computers. On the left side of the main content area, there is a sidebar with links for Chart Library Nwagon, WAI-ARIA UI, Coding Convention, Font Simulation, and NUF Dreamweaver Extension. The main content area has a heading "Coding Convention" and a paragraph explaining the relationship between markup development and various technologies like design, browser, script, and accessibility. It also mentions the purpose of coding conventions and the benefits for other users.

기술로 동행하다.

널리 사람을 이롭게 하기 위한 다양한 기술로 접근성을 실천하고 싶습니다.

Technology Doesn't Leave Anyone Behind

개발도구

Chart Library Nwagon

WAI-ARIA UI

[Coding Convention](#)

Font Simulation

NUF Dreamweaver Extension

About NULI NAVER Accessibility WIT블로그 | 로그인

Coding Convention

마크업 개발은 프론트-엔드 페이지의 기본 골격을 형성하기 때문에 디자인, 브라우저, 스크립트, 성능, 접근성 등과 긴밀한 관계가 있습니다.
즉, 마크업 개발을 잘 해야 모든 브라우저에서 콘텐츠를 손쉽 없이, 빠르고 쉽게 사용자에게 전달할 수 있습니다.

코딩 컨вен션은 이러한 조건을 만족시키기 위해 마크업 개발자가 지켜야 할 표준을 제시합니다.

또한 다른 사람도 코드를 빠르고 쉽게 이해할 수 있도록 하여 유지보수 비용을 절감하고 업무 효율을 높입니다.

Coding Convention (PDF 국문) 다운로드 ↴

Font Simulation

NUF Dreamweaver Extension



6. 파이썬 입력 방식

몇 가지 코딩 스타일

- **Toast FE Guide**
- <https://ui.toast.com/fe-guide/en>

TOAST UI

APPLICATIONS

COMPONENTS

TOOLS

FE GUIDE

POSTS

FE Guide

Get packed with compass and map for Front-End Javascript development voyage with FE Guide.

HTML/CSS/Sass

In web development, HTML forms the basic skeletal structure, and CSS decides how the markup language will be presented. Both are closely related to service performance and accessibility. In other words, both HTML and CSS must be written well in order for all browsers to fully express the contents without significant loss. This guide serves to help developers write consistent codes to facilitate cooperation and to minimize maintenance and expansion cost.



7. 파이썬 연산처리

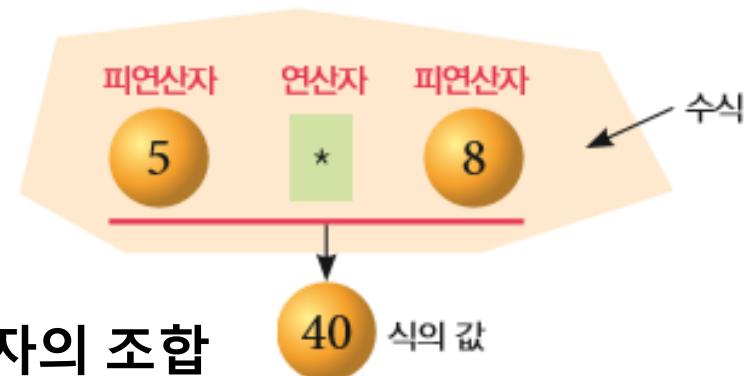
정수 연산

- 덧셈, 뺄셈, 곱셈, 나눗셈

- + : 키보드 +
- - : 키보드 -
- × : 키보드 *
- ÷ (몫을 구하기) : 키보드 // (두 글자로 된 연산기호이므로 띄우지 않고 연속해서 기호를 타이핑한다.)
- ÷ (소수점까지 나누기) : 키보드 /
- 나머지 : 키보드 %
- 제곱 : 키보드 **

- 연산자와 피연산자

- 수식(expression)=피연산자들과 연산자의 조합
- 연산자(operator)는 연산을 나타내는 기호
- 피연산자(operand)는 연산의 대상이 되는 것



7. 파이썬 연산처리

정수 연산

2 + 4 - 5

1

2 * 4

8

11 // 4 # 11을 4로 나눈 몫은 2

2

11 % 4 # 11을 4로 나눈 나머지는 3

3

10 / 4 # $10/4 = 2.5$

2.5

10 // 4 # $10/4 = 2$

2

2 ** 3 # 2의 세제곱은 8

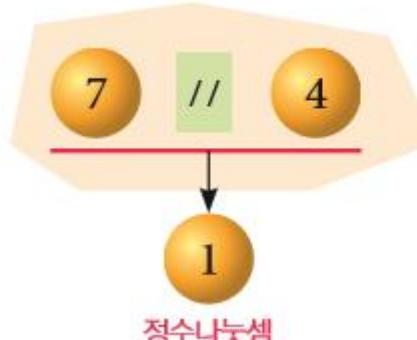
8

2 ** -1 # 음수 제곱은 역수. $2^{-1} = 1/2$

0.5

1000 ** 0 # 모든 수의 0제곱은 1

1





7. 파이썬 연산처리

실수 사칙연산

- 실수에 대해서도 사칙연산 가능

$1.1 + 2.4$

3.5

$4.2 - 1.2$

3.0

$1.2 * 2.2$

2.64

$5.4 / 2.7$

2.0

7. 파이썬 연산처리



Quiz: 사칙연산

- 현재 5000원이 있고 사탕의 가격이 120원이라고 하자. 최대한 살 수 있는 사탕의 개수와 나머지 돈은 얼마인가?

Solution

```
: myMoney = 5000;
candyPrice = 120;
# 최대한 살 수 있는 사탕 수
numCandies = myMoney//candyPrice
print('최대한 살 수 있는 사탕 수 = ', numCandies)
# 최대한 사탕을 구입하고 남은 돈
change = myMoney % candyPrice;
print('최대한 사탕을 구입하고 남은 돈 = ', change)
```

최대한 살 수 있는 사탕 수 = 41
최대한 사탕을 구입하고 남은 돈 = 80



7. 파이썬 연산처리

연산 순서와 괄호

- 곱셈과 나눗셈을 덧셈이나 뺄셈보다 먼저 한다.

3 + 2 * 4

11

3 + (2 * 4)

11

(3 + 2) * 4

20

- Quiz

$$100 \div [3 \times \{10 - (3 \times 2)\} + 8]$$



7. 파이썬 연산처리

부등식의 참과 거짓 계산

- 참과 거짓을 계산하는 부등식 연산

- $>$: 키보드 $>$
- $<$: 키보드 $<$
- $=$: 키보드 $==$
- \neq : 키보드 $!=$
- \geq : 키보드 \geq
- \leq : 키보드 \leq

2 > 1

True

2 > 2

False

2 == 2

True

2 != 1

True

3 <= 3

True



7. 파이썬 연산처리

부울리언 대수

- 참과 거짓에 대해 & (AND) 라는 연산과 | (OR) 라는 부울리언 대수(Boolean algebra) 연산
- & (AND) 연산은 두 값이 모두 참일 때만 답이 참이 된다. 즉, 하나라도 거짓이면 답은 거짓이다.
- | (OR) 연산은 두 값이 모두 거짓일 때만 답이 거짓이 된다. 즉, 하나라도 참이면 답은 참이다.

```
(2 > 0) & (2 < 3)
```

True

```
(2 > 2) | (2 < 3)
```

True



7. 파이썬 연산처리

부울리언 대수

True & True

True

True & False

False

False & True

False

False & False

False

True | True

True

True | False

True

False | True

True

False | False

False

8. 변수

변수란?

- 변수는 컴퓨터의 메모리 안에 만들어지는 공간으로 우리는 여기에 숫자나 문자를 저장할 수 있다.
- 지역 변수 – 함수 안에서 선언되는 변수
- 전역 변수 – 함수 외부에서 선언되는 변수
- 인스턴스 변수 – 클래스 안에 선언된 변수, 앞에 `self.`가 붙는다.
- 앞서서 숫자 계산을 했을 때, 다음 같은 특징이 있다.

```
: 1 + 3
: 4

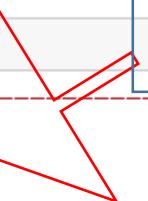
: 1 + "3"
: TypeError
Cell In[63], line 1
----> 1 1 + "3"

TypeError: unsupported operand type(s) for +: 'int' and 'str'

: 1 + int("3")
: 4
```

숫자와 문자는 연산이 안 됨

Traceback (most recent call last)



8. 변수

변수란?

- 그렇다면 직접, 키보드로 숫자 값을 직접 입력 받을 경우 `input()`을 이용

```
input("첫번 째 숫자를 입력하시오.: ")
```

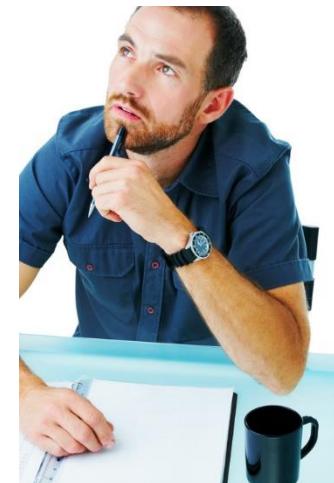
```
첫번 째 숫자를 입력하시오.: 1
```

```
'1'
```

```
input("두번 째 숫자를 입력하시오.: ")
```

```
두번 째 숫자를 입력하시오.: 
```

- 그러나, 이 상태로는 연산을 할 수 없다.
- 변수가 필요





8. 변수

변수·함수·유형 표기 방식

- 문자, 숫자, 밑줄(_)로 구성됩니다.
- 숫자는 처음에 나올 수 없습니다.
- 대소문자를 구분합니다.
- 예약어 사용 불가.
- 파이썬은 변수나 함수 이름은 스네이크 표기법을, 클래스 이름은 파스칼 표기법을 쓰는 관례를 사용
- **camelCase**
- **PascalCase**



8. 변수

변수·함수·유형 표기 방식

● 카멜 표기법(Camel case)

- camelCase
- 좀 더 공식화된 용어로는 중간 대문자(medial caps)라는 용어도 있다. '카멜'은 '낙타'를 뜻 한다. '카멜백 표기법(camelback case)'이라고도 불렸지만 요즘은 거의 쓰이지 않는다. 낙 타의 등에 혹이 불룩하게 나와 있는 것에서 따온 이름으로, 각 단어의 첫 글자를 대문자로 표 기하되 첫 단어의 첫 글자는 소문자로 표기하는 방식이다. Java가 변수나 함수 이름에 이를 사용하고 있으며, 자바스크립트, C#, Dart와 같은 많은 언어들이 사용하고 있다.

● 파스칼 표기법(Pascal case)

- PascalCase
- 각 단어의 첫 글자를 대문자로 표기한다. 첫 단어의 첫 글자도 대문자로 표기한다. 요즈음은 그냥 이것도 카멜 표기법이라고 부른다. 구분해서 부를 때는 이쪽은 대문자 카멜 표기법 (upper camel case)으로, 첫 단어 첫 글자를 소문자로 쓰는 방식은 소문자 카멜 표기법 (lower camel case)이라고 부른다. 파스칼 언어에서 이러한 표기법을 사용했기 때문에 붙은 이름이다. 변수나 함수 이름으로는 카멜백 표기법을 사용하는 언어라고 해도 적어도 클래스 이름은 파스칼 표기법을 사용하는 게 보통이다. C#은 변수 이름만 카멜 표기법을 쓰며 메서드 이름이나 클래스 이름은 파스칼 표기법을 쓴다. 또는 기본 유형은 카멜 표기법 또는 스네이크 표기법을 사용하더라도 사용자 정의 유형은 파스칼 표기법을 사용하기도 한다. 당연히 파스칼은 변수 이름도 파스칼 표기법을 사용한다.



8. 변수

변수·함수·유형 표기 방식

● 스네이크 표기법(Snake case)

- SNAKE_CASE, snake_case
- 땅바닥을 기어다니는 뱀의 모양과 비슷하다고 해서 붙은 이름. 모두 소문자(혹은 모두 대문자)로 쓰되, 띄어쓰기를 밑줄(_) 기호로 대체하는 것이다. C, C++과 같은 언어들이 이 방법을 사용한다. 변수나 함수, 유형, 클래스 이름은 소문자를, 상수 및 매크로 이름은 대문자를 사용하는 게 관례. Java도 static final 변수(사실상 상수)와 enum 값 이름에 대문자 스네이크 표기법을 쓴다. 파이썬은 변수나 함수 이름은 스네이크 표기법을, 클래스 이름은 파스칼 표기법을 쓰는 관례를 사용한다. 띄어쓰기를 하면 안 되는 파일 이름을 적을 때에도 이 방법을 사용한다. 포트란이나 코볼처럼 문자열을 제외하고 대소문자 구별을 안 하는 언어라면 스네이크 표기법.

● 케밥 표기법(Kebab case)

- KEBAB-CASE, kebab-case
- 케밥 꼬치의 모양을 보고 따온 말로, 꼬치에 재료가 듬성듬성 꽂혀 있는 것과 비슷한 모양이라는 의미다. 모두 소문자(혹은 모두 대문자)로 쓰되, 띄어쓰기를 하이픈(-) 기호로 대체하는 것이다. 하이픈은 뺄셈 기호로도 쓰이기 때문에 일반 프로그래밍 언어에서는 이름에 잘 안 쓰는 편이며 많은 언어가 변수나 함수, 유형 이름에 하이픈을 못 쓰게 한다. CSS는 연산을 하지 않으므로 이 방식을 사용한다. HTML도 id나 class의 이름은 케밥 표기법을 쓴다. 이들은 주로 CSS에서 쓰이기 때문. URL에도 종종 쓰이며, 띄어쓰기를 하면 안 되는 파일 이름에도 가끔 쓰인다.



8. 변수

변수·함수·유형 표기 방식

● 헝가리안 표기법(Hungarian notation)

- strHungarianNotation
- 영어로는 대소문자를 뜻하는 case 대신에 표기법을 뜻하는 notation을 사용하는데, 이는 대소문자 문제가 아니기 때문이다. 파스칼 표기법처럼 첫 단어를 포함해서 모든 단어의 첫 글자를 대문자로 하되, 그 앞에 변수의 유형을 뜻하는 약어를 소문자로 붙여 준다. 정수(int)형이라면 i, 문자열(string)이라면 str을 붙이는 식이다. 이 방법을 처음 고안한 마이크로소프트의 개발자[이와 같은 표기법을 생각한 것은 MS로 오기 전 제록스의 팔로-알토 연구소에 있었을 때라고 한다]이와 같은 표기법을 생각한 것은 MS로 오기 전 제록스의 팔로-알토 연구소에 있었을 때]가 헝가리인이었기 때문에 이런 이름이 붙었다. 윈도우 API에서 이 방식을 사용했으며, MFC에도 썼지만 지금은 잘 안 쓰는 방식이다. 마이크로소프트조차도 이전 윈도우 API와 호환성을 위해서 남겨놓았을 뿐, 더 이상 쓰지 않으며 .NET 프레임워크 프로그래밍에 사용하는 일반 명명 규칙에서 아예 쓰지 말라고 못박았다. 그래도 요즘도 가끔 쓰이는 경우는 있다. 구글은 한때 글로벌 상수 이름 앞에 소문자 k를 붙였는데, 최근에는 가이드라인을 통해 이렇게 하지 말라고 권고하고 있지만 오픈 소스 저장소에 올라와 있는 구글 코드 중에는 여전히 이를 사용한 코드가 종종 보인다. 단, 구글의 C++ 코딩 가이드라인에서는 전역 상수 앞에 k를 붙이도록 하고 있다.



8. 변수

변수 선언

- 앞선 예를, 변수를 통해 해결해 보자.

```
number1=int(input("첫번 째 숫자를 입력하시오.: "))
```

첫번 째 숫자를 입력하시오.: 1

```
number2=int(input("두번 째 숫자를 입력하시오.: "))
```

두번 째 숫자를 입력하시오.: 3

```
print(number1)
```

1

```
print(id(number1))
```

140716101544864

```
type(number1)
```

int

```
print(number1 + number2)
```

4

```
print("숫자 1과 숫자 2의 합은: ",number1 + number2)
```

숫자 1과 숫자 2의 합은: 4

```
print("숫자 1과 숫자 2의 합은: {}".format(number1 + number2))
```

숫자 1과 숫자 2의 합은: 4



8. 변수

변수 선언

- 그렇다면, 변수란 무엇인가?

```
x = 1      # x는 정수입니다.  
x = 'hello' # 여기서는 x가 문자열입니다.  
x = [1, 2, 3] # 여기서는 x가 리스트입니다.  
x  
[1, 2, 3]
```

가장 끝 값이 출력

```
a=10
```

```
b=10
```

```
a, b
```

```
(10, 10)
```

```
a is b
```

```
True
```

```
id(a), id(b)
```

```
(140716101545152, 140716101545152)
```

곧, 변수 a와 b는
같은 값 10

같은 주소 140716101545152를
가리키고 있다.

따라서, 파이썬에서 변수란?
특정 값이 저장된 주소를 가리키는 포
인터이다.



8. 변수

변수 선언

- 다음 예를 통해, 좀 더 이해해 보자.

```
b = 30
```

```
a, b
```

```
(10, 30)
```

```
a is b
```

```
False
```

```
id(a), id(b)
```

```
(140716101545152, 140716101545792)
```

b 값이 달라지며, 모든 것이
달라졌다.

```
b = 10
```

```
a, b
```

```
(10, 10)
```

```
a is b
```

```
True
```

```
id(a), id(b)
```

```
(140716101545152, 140716101545152)
```

b 값이 같아지며, 모든 것이
다시 같아졌다.

따라서, 파이썬에서 변수란?
값이 같으면 같은 곳을, 다르면 다른 곳을 가
리키게 된다.



8. 변수

상수란?

- 상수(constant)는 한번 값이 결정되면 절대로 변경되지 않는 변수.
 - **TAX_RATE = 0.35**
 - **PI = 3.141592**
 - **MAX_SIZE = 100**

```
TAX_RATE = 0.35
PI = 3.141592
MAX_SIZE = 100
print(TAX_RATE,PI,MAX_SIZE )
```

0.35 3.141592 100



9. 자료형

자료형이란?

- 파이썬에서 자료형은 유형에 따라서 논리형, 수치형, 문자형, 구조형으로 구분
- 일반적으로 숫자와 문자, 그리고 구조에 따른 저장을 위해서 여려가지 구분된 자료형을 제공

유형	자료형	설명	선언
논리형	불리언(boolean type)	• 참(True) 또는 거짓(False)을 표현할 때 사용	• b = True
수치형	정수(integer type)	• 자연수를 포함해 값의 영역이 정수로 한정된 값	• i = 10
수치형	2진수 정수(binary type)	• 2진수 자료형(숫자가 '0b' 또는 '0B'로 시작)	• b = 0b010
수치형	8진수 정수(octal type)	• 8진수 자료형(숫자가 0o 또는 0O로 시작)	• o = 0o130
수치형	16진수 정수(hexadecimal type) 1	• 16진수 자료형(0x로 시작)	• h = 0xABCD
수치형	실수(floating-point type)	• 소수점이 포함된 값	• f = 12.34
수치형	복소수(complex type)	• 복소수 사용을 위한 자료형	• j = 1 + 23j
문자형	문자열(string type)	• 값이 문자로 출력되는 자료형	• s = "Suan"



9. 자료형

자료형이란?

● ...

유형	자료형	설명	선언
시퀀스형	리스트(list type)	<ul style="list-style-type: none">여러 요소를 묶어 하나의 변수로 사용	<ul style="list-style-type: none"><code>l = [1, 2, 3]</code>
시퀀스형	튜플(tuple type)	<ul style="list-style-type: none">리스트와 유사하지만 생성, 삭제, 수정 불가	<ul style="list-style-type: none"><code>t = (1, 2, 3)</code>
시퀀스형	범위(range type)	<ul style="list-style-type: none">숫자의 불변한 시퀀스 형태	<ul style="list-style-type: none"><code>range(10)</code>
집합형	집합(set type)	<ul style="list-style-type: none">중복과 순서가 없는 집합을 위한 자료형	<ul style="list-style-type: none"><code>s = {1, 2, 3}</code>
집합형	불변 집합(frozenset type) 불	<ul style="list-style-type: none">불변한 집합을 위한 자료형	<ul style="list-style-type: none"><code>fs = frozenset{1, 2, 3}</code>
매핑형	딕셔너리(dictionary type)	<ul style="list-style-type: none">키(key)와 값(value)이 쌍(pair)으로 들어간 자료형	<ul style="list-style-type: none"><code>d = {1:'One', 2:'Two'}</code>
바이트 시퀀스형	바이트(byte type)	<ul style="list-style-type: none">바이트 값을 가지는 자료형	<ul style="list-style-type: none"><code>byte = b'hello'</code>
바이트 시퀀스형 b	bytearray type	<ul style="list-style-type: none">바이트의 시퀀스 형태로 수정 가능한 자료형	<ul style="list-style-type: none"><code>ba = bytearray(b'hello')</code>
바이트 시퀀스형	memoryview type	<ul style="list-style-type: none">바이트의 메모리 값으로 표현한 자료형	<ul style="list-style-type: none"><code>mv = memoryview(b'hello')</code>



9. 자료형

자료형이란?

- 파이썬에서 변수의 메모리 크기를 알기 위해서는 sys 패키지의 `getsizeof` 명령을 사용한다.

```
[ ] from sys import getsizeof
```

```
[ ] a = 1
getsizeof(a)
```

28

```
[ ] b = "1"
getsizeof(b)
```

50



9. 자료형

자료형이란?

- 변수나 값의 자료형을 알아보려면 `type` 명령을 사용하면 된다.
`None` 타입은 변수가 비어있음을 표시할 때 사용한다.

```
[ ] type(None)
```

```
NoneType
```

```
[ ] type(True)
```

```
bool
```

```
[ ] type(1)
```

```
int
```

```
[ ] type(3.14)
```

```
float
```



9. 자료형

자료형이란?

- 복소수를 사용할 때는 허수부 숫자 뒤에 j문자를 붙인다.

```
[ ] type(2 + 3j)
```

complex

```
[ ] type("abc")
```

str

```
[ ] type((1, 2, 3))
```

tuple

```
[ ] type([1, 2, 3])
```

list

```
[ ] type({"A": 10, "B": 20, "C": 30})
```

dict



9. 자료형

자료형이란?

- 문자열과 숫자형은 연산 불가능

```
"I have" + 2 + "pens."
```

```
-----  
TypeError
```

```
Cell In[53], line 1
```

```
----> 1 "I have" + 2 + "pens."
```

```
Traceback (most recent call last)
```

```
TypeError: can only concatenate str (not "int") to str
```

● 문자

- +, * 연산자

```
'py'+'thon'
```

```
'python'
```

```
'py' 'thon'
```

```
'python'
```

```
'py'*3
```

```
'pypypy'
```



9. 자료형

자료형이란?

- 인덱싱 & 슬라이싱

p	y	t	h	o	n
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

```
'python'[0]
```

```
'p'
```

```
'python'[5]
```

```
'n'
```

```
'python'[1:4]
```

```
'yth'
```

```
'python'[-2:]
```

```
'on'
```



9. 자료형

부동소수점 실수 자료형

- 파이썬에서는 정수(integer)뿐 아니라 소수점을 포함하는 실수(real number)도 다룰 수 있다. 정수 데이터에는 소수점이 없고 실수 데이터에는 소수점이 붙어 있다.

```
10 # 정수
```

```
10
```

```
10.0 # 실수
```

```
10.0
```

- 정수부가 0인 실수는 0을 생략할 수 있다.

```
.1 # .1 = 0.1
```

```
0.1
```



9. 자료형

부동소수점 실수 자료형

- 숫자 계산을 하는 경우 계산에 쓰인 숫자 중 하나라도 소숫점이 있는 실수가 있으면 계산결과는 실수가 된다. 나눗셈 연산의 결과는 입력에 상관없이 항상 실수로 처리한다. 예를 들어 10×5 는 정수로 취급 하지만 10.0×5 나 $10 \div 5$ 는 실수로 취급한다.

```
[ ] 10 * 5 # 정수
```

50

```
[ ] 10.0 * 5 # 실수
```

50.0

```
[ ] 10 / 5 # 실수
```

2.0



9. 자료형

부동소수점 실수 자료형

- 정수와 실수는 컴퓨터 메모리에 저장되는 방식이 다르다. 이를 데이터의 자료형(type)이 다르다고 한다. 파이썬에서 정수는 int라는 정수 자료형으로 처리한다. 실수는 float라는 부동소수점(floating point number) 자료형으로 처리된다. 파이썬에서 데이터의 자료형을 알려면 type 명령을 사용한다.

```
[ ] type(10)
```

```
int
```

```
[ ] type(10.0)
```

```
float
```



9. 자료형

부동소수점 실수 자료형

- 프로그래밍 언어는 IEEE 754라는 국제표준에 따라 실수를 부동소수점 방식으로 표현한다.
- 부동소수점 방식에서는 숫자를 정수로 된 유효숫자와 정수로 된 지수의 곱으로 표현한다. 예를 들어 십진수 부동소수점 방식에서 123.456 이란 숫자는 123456×10^{-3} 이므로 123456이라는 정수 유효숫자와 -3이라는 정수 지수로 나타낼 수 있다.
- 파이썬에서는 유효숫자e지수라는 방법으로 부동소수점 형태를 직접 표현한다.

$$\text{유효숫자} e \text{지수} = \text{유효숫자} \times 10^{\text{지수}}$$

```
123e2 # 123e2 = 123.0 x 100 = 12300.0
```

12300.0

```
123e-2 # 123e-2 = 123.0 x 0.01 = 1.23
```

1.23

```
123.456e-3 # 123.456e-3 = 123.456 x 0.001 = 0.123456
```

0.123456



9. 자료형

자료형 변환(Data Type Conversion)

- 실수를 정수로 변환하거나 정수를 실수로 변환하려면 `int` 명령과 `float` 명령을 사용한다.

```
[ ] int(1.0) # 실수를 정수로 변환
```

```
1
```

```
[ ] float(1) # 정수를 실수로 변환
```

```
1.0
```

- 만약 정수형으로 변환하려는 숫자가 정확히 정수로 표현될 수 없으면 소숫점 이하의 숫자를 버린다.

```
[ ] int(3.14)
```

```
3
```

```
[ ] int(3.9)
```

```
3
```

```
[ ] int(-3.9)
```

```
-3
```



9. 자료형

자료형 변환(Data Type Conversion)

- 아래의 함수들을 통해 데이터형의 변환을 할 수 있다.

Function	Description
int(x [,base])	Converts x to an integer. base specifies the base if x is a string.
long(x [,base])	Converts x to a long integer. base specifies the base if x is a string.
float(x)	Converts x to a floating-point number.
complex(real [,imag])	Creates a complex number.
str(x)	Converts object x to a string representation.
repr(x)	Converts object x to an expression string.
eval(str)	Evaluates a string and returns an object.
tuple(s)	Converts s to a tuple.
list(s)	Converts s to a list.
set(s)	Converts s to a set.
dict(d)	Creates a dictionary. d must be a sequence of (key,value) tuples.
frozenset(s)	Converts s to a frozen set.



9. 자료형

자료형 변환(Data Type Conversion)

- 아래의 함수들을 통해 데이터형의 변환을 할 수 있다.

Function	Description
chr(x)	Converts an integer to a character.
unichr(x)	Converts an integer to a Unicode character.
ord(x)	Converts a single character to its integer value.
hex(x)	Converts an integer to a hexadecimal string.
oct(x)	Converts an integer to an octal string.



9. 자료형

NaN과 Inf

- IEEE 754 표준에 따른 부동소수점 자료형은 특별한 두 가지 값을 표현할 수 있다. 하나가 NaN(Not a Number)이고 또 하나는 Inf(Infinity)이다. NaN는 “숫자가 아닌 것”, Inf는 “무한대”를 뜻한다. 파이썬에서는 다음과 같은 명령으로 표현한다.

```
[ ] float("NaN")
```

```
nan
```

```
[ ] float("Inf")
```

```
inf
```

```
[ ] float("-Inf")
```

```
-inf
```



9. 자료형

Quiz : 변수 값 교환

- 변수 x 와 변수 y의 값을 서로 바꾸는 프로그램을 작성해보자. 다음과 같은 프로그램으로는 변수의 값을 교환할 수 없다. 왜 그럴까? 그리고 해결 방법은 무엇일까?

- - 아래 -

x = 10 #(1)

y = 20 #(2)

x = y #(3)

y = x #(4)



9. 자료형

Solution : 변수 값 교환

temp = x

x = y

y = temp

Quiz : 변수 값 교환

```
: x = 10  
y = 20  
print("x = {}, y = {}".format(x,y))
```

x = 10, y = 20

```
: temp = x  
x = y  
y = temp
```

```
: print("x = {}, y = {}".format(x,y))
```

x = 20, y = 10

『1과목』 프로그래밍 언어 GUI 도구





1. 인프라 설계 준비

학습목표

- 실습 환경에 맞는 GUI 도구를 선정한다.

눈높이 체크

- PyQt를 알고 계신가요?



1. PyQt

PyQt 개요

- PyQt는 파이썬 플러그인으로 구현된 크로스 플랫폼 GUI 툴킷 Qt의 파이썬 바인딩이다. PyQt는 영국 회사 리버뱅크 컴퓨팅이 개발한 자유 소프트웨어이다. PyQt는 마이크로소프트 윈도, 다양한 유닉스 파생판(리눅스, 맥OS 포함)을 지원한다



1. PyQt

PyQt5 패키지 설치와 창 뛰우기

- **PyQt-1.py**

```
import sys
from PyQt5.QtWidgets import *

app = QApplication(sys.argv)
qw = QWidget()
qw.show()
sys.exit(app.exec_())
```

Microsoft Windows [Version 10.0.19045.2604]
(c) Microsoft Corporation. All rights reserved.

C:\DEV\python_works>activate tf38_cpu

(tf38_cpu) C:\DEV\python_works>python PyQt-1.py
Traceback (most recent call last):
 File "PyQt-1.py", line 2, in <module>
 from PyQt5.QtWidgets import *
ModuleNotFoundError: No module named 'PyQt5'

(tf38_cpu) C:\DEV\python_works>



1. PyQt

PyQt5 패키지 설치와 창 뛰우기

(tf38_cpu) C:\DEV\python_works>pip install PyQt5

Collecting PyQt5

 Downloading PyQt5-5.15.9-cp37-abi3-win_amd64.whl (6.8 MB)

6.8/6.8 MB 6.3 MB/s eta 0:00:00

Collecting PyQt5-sip<13,>=12.11

 Downloading PyQt5_sip-12.11.1-cp38-cp38-win_amd64.whl (78 kB)

78.1/78.1 kB 4.5 MB/s eta 0:00:00

Collecting PyQt5-Qt5>=5.15.2

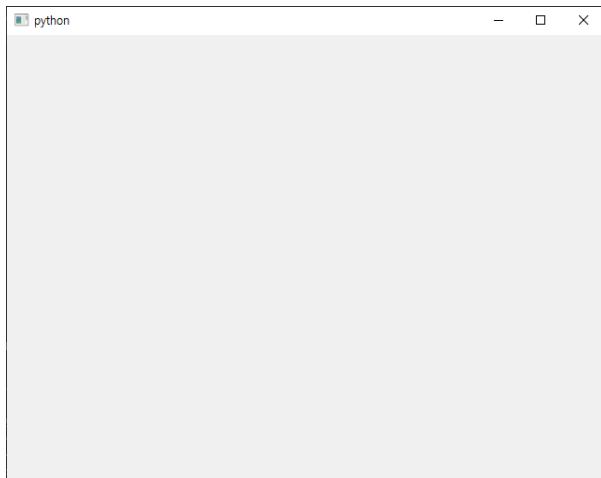
 Downloading PyQt5_Qt5-5.15.2-py3-none-win_amd64.whl (50.1 MB)

50.1/50.1 MB 2.7 MB/s eta 0:00:00

Installing collected packages: PyQt5-Qt5, PyQt5-sip, PyQt5

Successfully installed PyQt5-Qt5-5.15.2 PyQt5-sip-12.11.1 PyQt5-5.15.9

(tf38_cpu) C:\DEV\python_works>python PyQt-1.py





1. PyQt

간단한 예제

- **PyQt-2.py**

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton

def main():
    app = QApplication(sys.argv)
    widgets = QWidget()
    widgets.resize(200,50)
    widgets.setWindowTitle("PyQt5 GUI")
    # 푸시 버튼
    btn1 = QPushButton('O', widgets)
    btn1.resize(100,30)
    btn1.move(0,20)
    btn2 = QPushButton('X', widgets)
    btn2.resize(100,30)
    btn2.move(100,20)
    # 푸시 버튼 클릭
    btn1.clicked.connect(lambda: print("O 클릭"))
    btn2.clicked.connect(lambda: print("X 클릭"))
    widgets.show()
    sys.exit(app.exec_())
main()
```



1. PyQt

간단한 예제

- **PyQt-2.py**

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton

def main():
    app = QApplication(sys.argv)
    widgets = QWidget()
    widgets.resize(200,50)
    widgets.setWindowTitle("PyQt5 GUI")
    # 푸시 버튼
    btn1 = QPushButton('O', widgets)
    btn1.resize(100,30)
    btn1.move(0,20)
    btn2 = QPushButton('X', widgets)
    btn2.resize(100,30)
    btn2.move(100,20)
    # 푸시 버튼 클릭
    btn1.clicked.connect(lambda: print("O 클릭"))
    btn2.clicked.connect(lambda: print("X 클릭"))
    widgets.show()
    sys.exit(app.exec_())
main()
```

1. PyQt

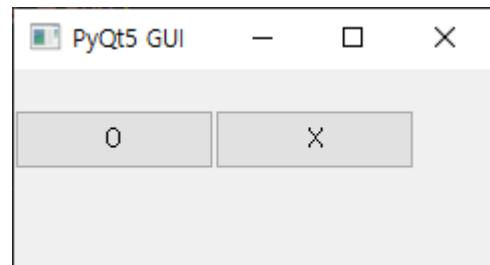
간단한 예제

- **PyQt-2.py**

```
(tf38_cpu) C:\DEV\python_works>python PyQt-2.py
```

O 클릭
X 클릭

```
(tf38_cpu) C:\DEV\python_works>
```





1. PyQt

위젯의 타이틀 및 배경색 설정하기

- **PyQt-3.py**

```
import sys
from PyQt5.QtWidgets import QApplication
from PyQt3Widget import PyQt3Widget
app = QApplication(sys.argv)
myw = PyQt3Widget()
myw.show()
sys.exit(app.exec_())
```

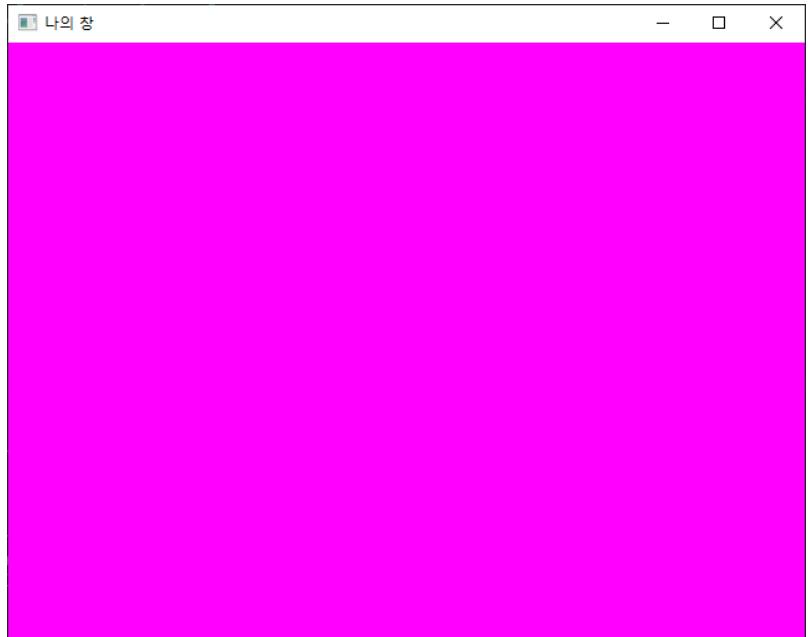
- **PyQt3Widget.py**

```
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *
class PyQt3Widget(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("나의 창")
        pal = QPalette()
        pal.setColor(QPalette.Background,QColor(255,0,255))
        self.setAutoFillBackground(True)
        self.setPalette(pal)
```

1. PyQt

위젯의 타이틀 및 배경색 설정하기

(tf38_cpu) C:\DEV\python_works>python PyQt-3.py





1. PyQt

이벤트 핸들러

- **PyQt-4.py**

```
import sys
from PyQt5.QtWidgets import QApplication
from PyQt4Widget import PyQt4Widget
app = QApplication(sys.argv)
myw = PyQt4Widget()
myw.show()
sys.exit(app.exec_())
```



1. PyQt

이벤트 핸들러

- **PyQt4Widget.py**

```
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *
class PyQt4Widget(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("테스트 입력 및 설정")
        self.te = QTextEdit(self)
        self.te.resize(400,90)
        self.resize(1000,600)
        self.btn = QPushButton("확인",self)
        self.btn.move(430,0)
        self.btn.resize(200,90)
        self.lb = QLabel("[테스트]",self)
        self.lb.move(0,110)
        self.btn.clicked.connect(self.BtnClick)
    def BtnClick(self):
        txt = self.te.toPlainText()
        self.lb.setText(txt)
        self.te.setText("")
```

1. PyQt

이벤트 핸들러

(tf38_cpu) C:\DEV\python_works>python PyQt-4.py





1. PyQt

리스트 박스 사용하기

- **PyQt-5.py**

```
import sys
from PyQt5.QtWidgets import QApplication
from PyQt5Widget import PyQt5Widget
app = QApplication(sys.argv)
myw = PyQt5Widget()
myw.show()
sys.exit(app.exec_())
```



1. PyQt

이벤트 핸들러

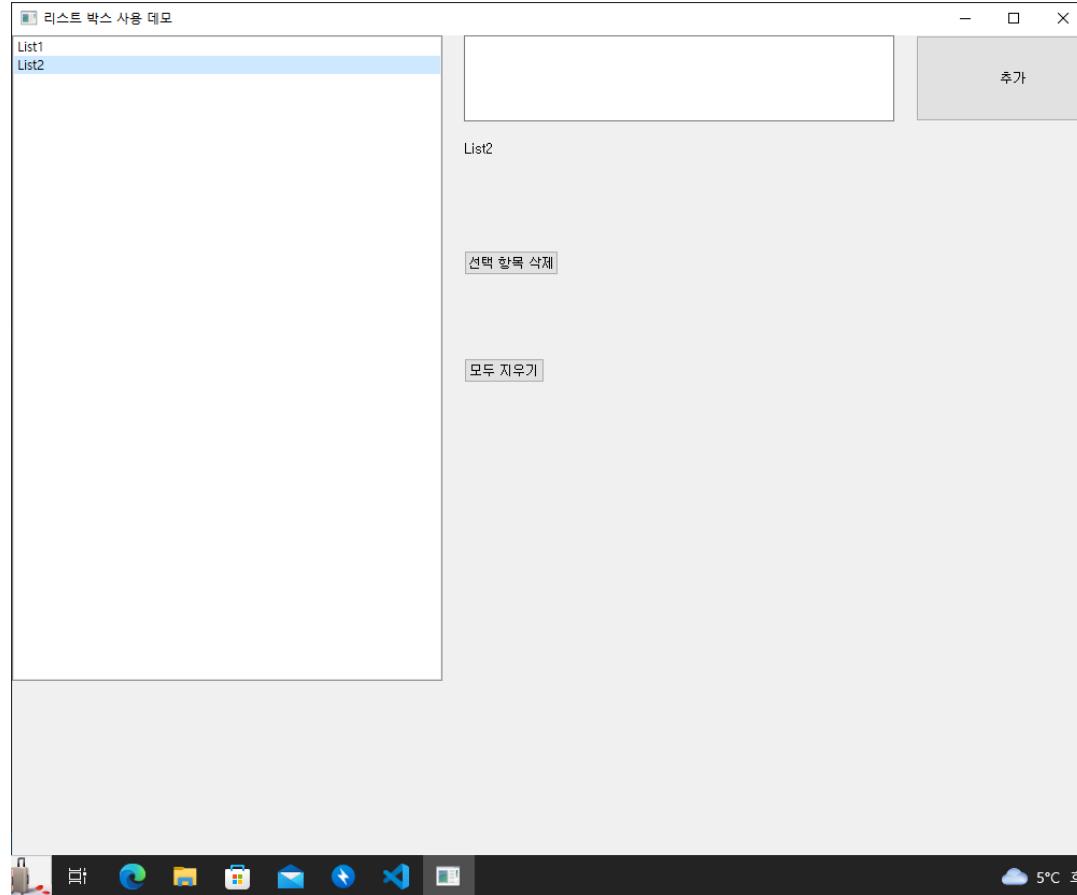
- **PyQt5Widget.py**

```
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *
class PyQt5Widget(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("리스트 박스 사용 데모")
        self.resize(1000,800)
        self.lbox = QListWidget(self)
        self.lbox.resize(400,600)
        self.te=QTextEdit(self)
        self.te.move(420,0)
        self.te.resize(400,80)
        self.btn = QPushButton("추가",self)
        self.btn.move(840,0)
        self.btn.resize(180,80)
        self.btn.clicked.connect(self.AddItem)
        self.lb_sel = QLabel("[테스트]",self)
        self.lb_sel.move(420,100)
        self.lbox.currentItemChanged.connect(self.OnLBoxSelectChange)
        self.btn_remove = QPushButton("선택 항목 삭제",self)
        self.btn_remove.move(420,200)
        self.btn_remove.clicked.connect(self.RemoveItem)
        self.btn_clear = QPushButton("모두 지우기",self)
        self.btn_clear.move(420,300)
        self.btn_clear.clicked.connect(self.ClearAll)
    def ClearAll(self):
        self.lbox.clear()
    def RemoveItem(self):
        index = self.lbox.currentRow()
        self.lbox.takeItem(index)
    def AddItem(self):
        data = self.te.toPlainText()
        self.lbox.addItem(data)
        self.te.setText("")
    def OnLBoxSelectChange(self):
        item = self.lbox.currentItem()
        self.lb_sel.setText(item.text())
```

1. PyQt

이벤트 핸들러

(tf38_cpu) C:\DEV\python_works>python PyQt-5.py





2. Tkinter

Tkinter 소개

- Tkinter는 Tcl/Tk에 대한 파이썬 Wrapper로서 Tcl/Tk를 파이썬에 사용할 수 있도록 한 Lightweight GUI 모듈이다. Tcl은 Tool Command Language의 약자로서 일종의 프로그래밍 언어이며, Tk는 크로스 플랫폼에 사용되는 일종의 GUI 툴킷이다. Tkinter는 타 GUI 프레임워크나 툴킷에 비해 지원되는 위젯들이 부족하고 UI도 그렇게 예쁘지 않다는 단점이 있지만, Python 설치시 기본적으로 내장되어 있는 파이썬 표준 라이브러리이기 때문에 쉽고 간단한 GUI 프로그램을 만들 때 활용될 수 있다.



2. Tkinter

Tkinter 소개

- 주의할 점은 라이브러리 이름이 **tkinter**라고 해서 **pip install tkinter**라고 입력하면 설치되지 않습니다.
- **pip install tk**라고 입력해야 합니다.

(tf38_cpu) C:\DEV\python_works>pip install tkinter

ERROR: Could not find a version that satisfies the requirement tkinter (from versions: none)

ERROR: No matching distribution found for tkinter

(tf38_cpu) C:\DEV\python_works>pip install tk

Collecting tk

 Downloading tk-0.1.0-py3-none-any.whl (3.9 kB)

Installing collected packages: tk

Successfully installed tk-0.1.0

(tf38_cpu) C:\DEV\python_works>



2. Tkinter

Tkinter 설치와 기본 문장

- Tkinter는 파이썬에 기본 내장되어 있기 때문에 PyQt처럼 별도로 설치할 필요가 없다. Tkinter를 사용하기 위해서는 먼저 `tkinter` 모듈을 아래와 같이 `import` 해야 한다 (주: Python 2에서는 Tkinter를 `import` 하고, Python 3에서는 `tkinter`를 `import` 한다). `tkinter` 모듈을 `import`한 다음에는 `Tk` 클래스 객체(`root`)를 생성하고, 이 객체의 `mainloop()` 메서드를 호출한다. 아래 코드와 같이 이런 기본 문장들을 수행하면, 빈 다이얼로그가 화면에 표시된다.
- Tk-1.py

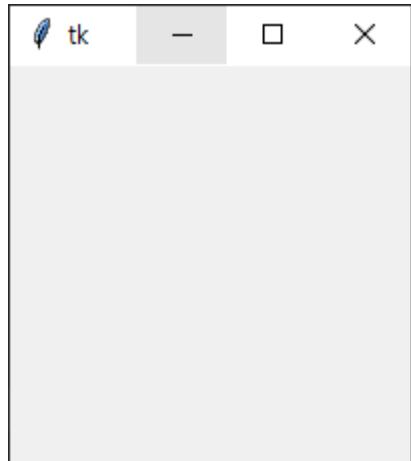
```
from tkinter import *
root = Tk()
root.mainloop()
```

2. Tkinter

Tkinter 설치와 기본 문장

- `mainloop()`는 이벤트 메시지 루프로서 키보드나 마우스 혹은 화면 Redraw와 같은 다양한 이벤트로부터 오는 메시지를 받고 전달하는 역할을 한다.

(tf38_cpu) C:\DEV\python_works>python Tk-1.py



2. Tkinter

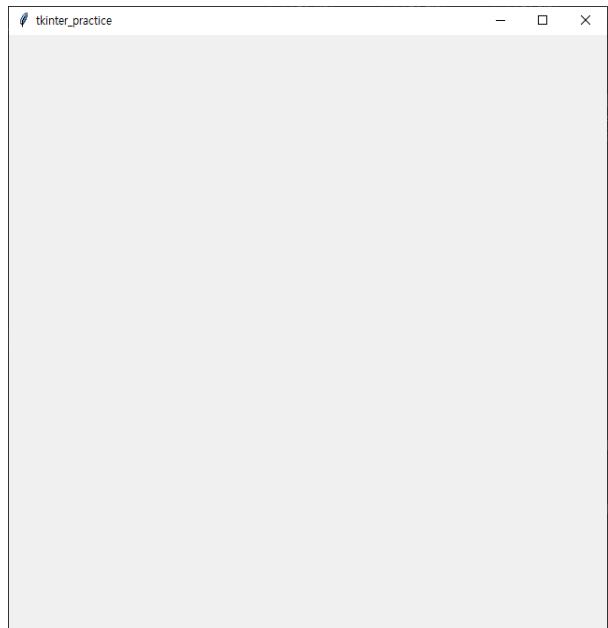
타이틀 지정 및 크기설정

- 타이틀은 위에서 본 윈도우 창의 제목을 말한다. 처음 아무 속성을 적지 않으면 tk로 지정되는데, 이것을 title 속성을 이용하여 바꿀 수 있다.
- 크기는 geometry 속성을 이용하여 바꿀 수 있다. 이때 파이썬의 *가 아닌 x로 사용해야 한다.
- 윈도우의 이름은 tkinter_practice로 바뀌었으며, 크기는 640x640으로 아주 커진 모습을 볼 수 있다.

● Tk-2.py

```
from tkinter import *

root = Tk() # GUI 생성
root.title("tkinter_practice") #상단의 타이틀 지정
root.geometry("640x640") # 크기 설정 (640x640)
root.mainloop() # GUI가 보이고 종료될때까지 실행함
```

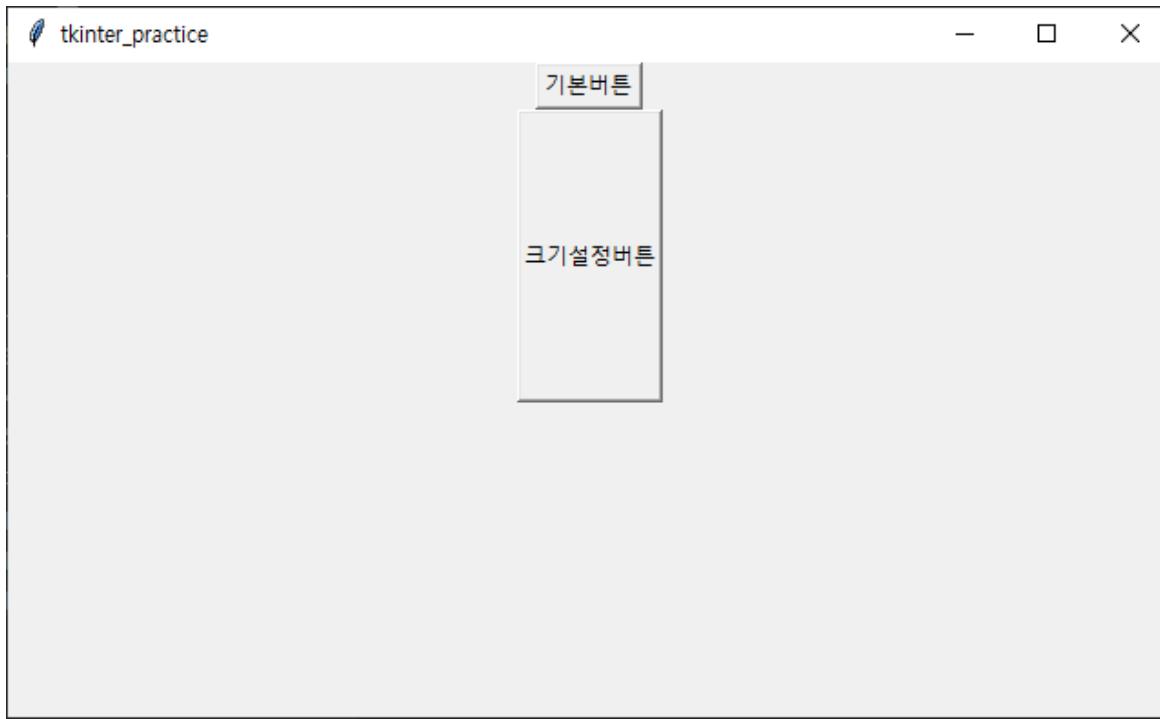


2. Tkinter

버튼 생성

- Btn1은 (현재 윈도우 , 버튼 이름)으로 설정하였다. 위에서 root = Tk()로 윈도우를 생성하였기 때문에 현재윈도우는 root이고 버튼 이름은 각자 설정하면 된다.
- Btn2는 (현재윈도우 , 넓이 , 높이 , 버튼이름)으로 설정하였다.

(tf38_cpu) C:\DEV\python_works>python Tk-3.py





2. Tkinter

버튼 생성

- Tk-3.py

```
from tkinter import *

root = Tk() # GUI 생성
root.title("tkinter_practice") #상단의 타이틀 지정
root.geometry("640x640") # 크기 설정 (640x640)

btn1 = Button(root, text = "기본버튼") #root로 지정한 윈도우에 button 생성
btn1.pack() # 윈도우상에 상대 위치로 위젯을 배치

btn2 = Button(root, width = 10 , height = 10 , text="크기설정버튼")
btn2.pack()

root.mainloop() # GUI가 보이고 종료될때까지 실행함
```

Unit A

참고자료





문헌

1. <http://www.ncs.go.kr>
2. NELLDAL/E/JOHN LEWIS지음, 조영석/김대경/박찬영/송창근 역, 단계별로 배우는 컴퓨터과학, 흥룡과학출판사, 2018
3. 기초부터 활용까지 C/C++ Programming, 김대경, 이귀봉, 삼양미디어, 2009-01-30
4. 기타 서적 및 웹 사이트 자료 다수 참조



감사합니다.

❖ Mobile: 010-9591-1401
❖ E-mail: onlooker2zip@naver.com / dkkim@lklab.org

Copyright 2022. DaeKyeong all rights reserved