생성형 AI 소프트웨어공학 · 2026학년도 2학기 · 1강

오리엔테이션과 소프트웨어공학 개관

그리고 개발 환경 구축: Python · VS Code · Git · GitHub, 첫 저장소에 파일을 올리기까지

대전대학교 컴퓨터공학과 · 박상돈

1강 · 수업 안내

오늘 세 가지를 합니다

1

오리엔테이션

수업 방식과 평가 기준, 그리고 이 수업만의 AI 활용 규칙을 합의합니다.

2

소프트웨어공학 개관

왜 이 학문이 필요해졌는지 실패 사례에서 출발해, AI 시대에 사람에게 남는 일을 정리합니다.

3

개발 환경 구축 (실습)

Python · VS Code · Git을 설치하고, GitHub 저장소를 만들어 첫 파일을 올립니다.

이번 주 성취 기준

- 소프트웨어공학이 왜 필요한지와 이 과목의 전체 구조를 설명할 수 있다.
- 구현이 자동화된 환경에서 개발자에게 남는 역할을 내 말로 설명할 수 있다.
- 개발 환경을 스스로 구축하고 GitHub 저장소에 내 파일을 올릴 수 있다.

01

PART 1

오리엔테이션

이 수업이 무엇을, 어떻게 배우는 수업인지, 그리고 AI를 어디까지 쓸 수 있는지부터 합의합니다.

"생성형 AI 소프트웨어공학"이란

코드를 **'작성'하는 법**이 아니라, AI에게 코드를 **'만들게 하고' 그 결과에 책임지는 법**을 배우는 수업입니다.
구현은 생성형 AI가 맡습니다. 우리는 무엇을 만들지 정의하고, 어떤 구조로 만들지 설계하고,
제대로 만들어졌는지 검증합니다.

정의한다

무엇을 만들 것인가. 요구사항을 말과 글로 명확히 한다.

설계한다

어떤 구조로 만들 것인가. 큰 그림과 경계를 사람이 정한다.

검증한다

제대로 만들어졌는가. 기준을 세우고 결과를 판정한다.

1부 · 오리엔테이션

우리의 작업 방식, "풀 바이브 코딩"

한 학기 내내, 사람이 자연어로 지시하고 AI가 구현하는 방식으로 작업합니다. 키보드로 코드를 치는 시간보다, 무엇을 시킬지 생각하고 결과를 확인하는 시간이 훨씬 깁니다.

그래서 이 수업의 실력은 타자 속도가 아니라 지시의 정확도와 검증의 꼼꼼함으로 결정됩니다.

한 학기 도구 스택

ChatGPT 계정 (무료)
Codex 로그인에 쓰는 내 계정. chatgpt.com 에서 무료 가입 (2주차)

Codex · ZCode (코딩 에이전트)
지시하면 파일을 만들고 고치는 AI 둘. Codex는 VS Code 확장 (GPT-5.6), ZCode는 별도 앱(GLM-5.3, 수업 공용 계정) (2주차)

VS Code
한 학기 내내 쓸 작업대. 다음 주 Codex가 이 안에 들어옵니다 (오늘 설치)

GitHub
모든 결과물의 기록·제출·협업 장소 (오늘 시작)

1부 · 오리엔테이션

한 학기 로드맵

1 – 7주
기초와 언어

환경 구축(1–2주), 생성형 AI 이해와 지시문 작성(3–4주), 요구사항 정의(5주), 설계(6주), 명세 기반 구현, 즉 지시하고 받아내기(7주)

8주
중간고사

1–7주 개념과 실습을 평가 범위로

9 – 13주
품질과 운영

테스트 설계(9주), 결함 수정·유지보수(10주), 배포와 CI(11주), 코딩 에이전트 자동화(12주), 품질·보안·윤리(13주)

14 – 16주
발표·마무리

발표·시연(14), 보강(15), 총정리·회고(16)

과제와 팀 프로젝트
개인 과제는 3·7·9주에 나갑니다. 팀 프로젝트는 5주차에 주제를 정해 학기 내내 키워서 14주차에 배포된 주소로 시연합니다.

1부 · 오리엔테이션

매주 이렇게 진행합니다

개념 강의 + 실습이 한 세트
앞부분은 개념, 뒷부분은 반드시 손으로 해 보는 실습입니다. 실습 결과 확인까지가 그날 수업입니다.

강의 자료는 전부 자체 제작
교재 대신 매주 이 슬라이드와 실습 안내를 LMS에 올립니다. 시험도 이 자료가 기준입니다.

노트북 지참 필수
매주 본인 노트북으로 실습합니다. 설치와 설정은 오늘 다 같이 하니 빈 노트북이어도 괜찮습니다.

팀 프로젝트는 학기 내내
5주차 주제 선정부터 14주차 발표까지, 매주 배우는 내용을 팀 프로젝트에 바로 적용합니다.

평가 기준

항목	비율	내용
출석	10%	매주 실습 체크리스트 통과가 출석 확인을 겸합니다
중간고사	25%	8주차. 1~7주 범위, 개념 + 상황 판단 서술 (AI 도구 없이)
과제	10%	개인 과제 3회. 3주 · 7주 · 9주 공지
기말고사	25%	학기 말. 전 범위, 개념 + 판단 서술 (AI 도구 없이)
프로젝트 (기타1)	20%	14주차. 배포된 결과물 시연 + 발표 + 저장소 기록 + 동료평가
출석 (기타2)	5%	학사 평가 기준에 따른 추가 출석 반영
역량평가 (기타3)	5%	학교 공통 역량평가. 세부 기준은 별도 공지

합계 100%. 지각·결석 처리, 제출 기한 등 세부 규정은 LMS의 강의계획서 원문이 기준입니다.

AI 활용 규칙, 딱 세 가지

① 쓰는 것이 기본입니다

이 수업에서 AI 사용은 부정행위가 아니라 요구사항입니다. 과제와 프로젝트는 AI를 활용해 만드는 것을 전제로 설계되어 있습니다.

② 설명하지 못하면 제출하지 마세요

제출물의 기준은 "내가 설명할 수 있는 것"까지입니다. 코드 한 줄, 설계 한 문장이라도 물었을 때 답하지 못하면 본인 것이 아닙니다. 구두 확인이 있을 수 있습니다.

③ 시험은 AI 없이

중간·기말고사는 도구 없이 봅니다. 개념과 판단력은 본인 머리에 있어야 하고, 그것이 이 수업이 기르려는 능력입니다.

제출물에는 사용한 도구와 핵심 지시문을 함께 기록합니다. 기록 양식은 과제 공지에 포함됩니다.

02

PART 2

소프트웨어공학 개관

코드가 아니라 '만드는 과정'의 학문. 왜 필요해졌는지, 유명한 실패에서부터 배웁니다.

소프트웨어공학이란 무엇인가

> 소프트웨어의 개발·운영·유지보수에 **체계적이고, 규율 있으며, 측정 가능한 접근**을 적용하는 것.
>
> (IEEE의 고전적 정의를 우리말로 풀면)

혼자서 감으로 짜는 '프로그래밍'과, 여러 사람이 만들어 오래 쓰이는 것을 책임지는 '공학'은 다릅니다. 그 차이를 만드는 세 단어가 정의 안에 들어 있습니다.

체계적

정해진 과정을 따른다. 순서와 산출물이 있다.

규율

팀이 합의한 규칙을 지킨다. 개인기가 아니라 약속.

정량적

감이 아니라 측정과 근거로 판단한다.

실패에서 시작합니다

Therac-25

1985–87 · 방사선 치료기

드문 입력 순서에서만 터지는 결함과 검증 부재가 겹쳐 환자들이 과다 피폭. 사망 사고를 포함해 최소 6건. 하드웨어 안전장치를 빼고 소프트웨어만 믿은 대가였습니다.

Ariane 5

1996 · 유럽 우주 로켓

이전 기종에서 재사용한 코드의 수치 변환 오버플로로 발사 37초 만에 자폭. 약 5억 달러가 사라졌습니다. 코드는 멀쩡했습니다. 옛 기종의 '가정'이 새 기종에선 틀렸을 뿐.

Knight Capital

2012 · 미국 증권사

서버 8대 중 1대에 새 버전 배포가 누락되어 잠자던 옛 코드가 깨어남. 45분 만에 약 4억 4천만 달러 손실, 회사는 사실상 무너졌습니다. 배포라는 '과정'의 실패였습니다.

셋 다 '코딩 실력'의 문제가 아니었습니다. 그렇다면 무엇의 문제였을까요?

세 사고의 공통점: 과정이 무너졌다

가정이 검증되지 않았다
Ariane 5: "이 값은 범위를 안 넘는다"는 옛 기종의 가정을 아무도 새 기종에서 다시 확인하지 않았습니다.

테스트가 현실을 덮지 못했다
Therac-25: 숙련된 조작자의 빠른 입력 순서라는 현실적 사용 패턴이 시험 항목에 없었습니다.

변경과 배포가 관리되지 않았다
Knight Capital: 무엇이 어디에 배포되었는지 확인하는 절차가 없었고, 8대 중 1대의 누락을 아무도 몰랐습니다.

그래서 '공학'이 필요합니다
개인의 실력이 아니라 팀의 과정입니다. 요구를 확인하고, 검증하고, 변경을 관리하는 체계가 사고를 막습니다. 이 수업이 다루는 것이 바로 그 과정입니다.

소프트웨어는 왜 유독 어려운가

다리나 건물을 짓는 공학에는 수백 년의 노하우가 있는데, 소프트웨어는 왜 아직도 자주 무너질까요. 소프트웨어라는 재료 자체의 성질 때문입니다.

보이지 않는다

진척도, 구조, 부실 공사가 눈에 안 보입니다. "거의 다 됐어요"를 확인할 방법이 마땅치 않습니다.

복잡도가 폭발한다

상태와 입력의 조합이 천문학적으로 늘어나, 모든 경우를 머리로도 시험으로도 다 덮기 어렵습니다.

바꾸기 쉬워 보인다

벽돌과 달리 코드는 즉시 고칠 수 있습니다. 그래서 아무나, 아무 때나 바꾸다 사고가 납니다.

요구가 계속 자란다

완공이 없습니다. 쓰이는 소프트웨어일수록 요구가 늘고, 변경은 죽을 때까지 계속됩니다.

이 네 성질은 구현을 사람이 하든 AI가 하든 사라지지 않습니다. 그래서 다음 질문이 중요해집니다.

구현이 자동화되면,
개발자는 사라질까?

생성형 AI는 초안 코드를 몇 초 만에 만들어 냅니다.
코드 '작성'이 병목이 아니게 된 시대. 그렇다면 사람의 일은 무엇이 남을까요.

구현이 자동화되어도 남는 일

무엇을 만들지 정의한다
요구사항을 명확한 말과 글로. AI는 시키지 않은 것을 만들어 주지 않습니다. (5주차 · 요구공학)

어떤 구조로 만들지 판단한다
화면 흐름, 데이터 구조, 기능의 경계. 좋은 설계를 고르는 안목은 사람의 몫입니다. (6주차 · 설계)

"됐다"의 기준을 세운다
인수 조건. 결과물을 받았을 때 통과·불통과를 판정할 기준을 미리 정합니다. (8·9주차)

정말 되는지 검증한다
테스트 설계와 결함 발견. 통과했는데도 남아 있는 결함을 찾아내는 눈. (9·10주차)

품질·보안·윤리에 책임진다
AI가 만든 취약점, 라이선스, 개인정보. 최종 책임은 이름을 올린 사람에게. (13주차)

이 다섯 줄이 곧 이 과목의 목차입니다.

이 과목의 구조: 생명주기를 따라갑니다

소프트웨어가 태어나서 운영되기까지의 여정을 '생명주기'라고 부릅니다. 우리 수업의 주차 구성이 곧 이 생명주기입니다.

1. **요구**
   무엇을 만들까 (5주)
2. **설계**
   어떤 구조로 (6주)
3. **구현·지시**
   AI에게 맡겨 받아내기 (7주)
4. **검증**
   테스트 설계 (9주)
5. **유지보수**
   결함 재현과 수정 (10주)
6. **배포·자동화**
   CI/CD와 에이전트 (11–12주)

그 전에, 1–4주는 준비 기간
이 여정을 떠나기 위한 장비(개발 환경 · AI 도구)와 언어(지시문 작성법)를 먼저 갖춥니다. 오늘이 그 첫날입니다.

03

PART 3

개발 환경 구축

여러분 노트북에 한 학기 치 작업대를 만듭니다. 설치부터 GitHub 첫 업로드까지, 오늘 다 끝냅니다.

오늘 설치할 네 가지

Python
우리가 다룰 언어의 실행기. AI가 만든 코드를 내 컴퓨터에서 돌리기 위해 필요합니다.

VS Code
한 학기 내내 쓸 작업대(에디터). 다음 주에 AI 에이전트 Codex가 이 안에 들어옵니다.

Git
변경 기록을 남기는 도구. 누가 언제 무엇을 바꿨는지 전부 기록됩니다.

GitHub
기록을 보관·공유하는 곳. 계정 하나가 여러분의 개발 이력서가 됩니다.

왼쪽부터 순서대로, 다 같이 진행합니다. 각 단계의 '확인 명령'이 통과되어야 다음 단계로 갑니다.

잠깐, '터미널'이 뭔가요? (오늘 확인 명령을 치는 곳)

글자로 명령하는 창

마우스로 클릭하는 대신, 명령어를 글자로 치고 Enter를 누르면 컴퓨터가 결과를 글자로 보여 주는 창입니다. Windows에서는 '명령 프롬프트(cmd)' 또는 'Windows 터미널', Mac은 '터미널'.

여는 법 (Windows)

키보드의 ⊞ Win 키를 누르고 cmd 라고 입력한 뒤 Enter. 검은(또는 파란) 창이 뜨면 성공입니다. 잠시 후 설치할 VS Code 안에도 터미널이 들어 있어서, 그때부터는 VS Code 안에서 씁니다.

⊞ Win → cmd 입력 → Enter

'먹는다 / 안 먹는다'

명령을 쳤을 때 결과(버전 숫자 등)가 나오면 '먹는 것', "'python'은(는) 내부 또는 외부 명령... 이 아닙니다" 같은 빨간·회색 오류가 나오면 '안 먹는 것'입니다. 안 먹으면 손 드세요.

명령 프롬프트 (예시)

```
C:\Users\hong> python --version
Python 3.14.7

C:\Users\hong> git --version
git version 2.55.0.windows.1

C:\Users\hong> cd hello-sw      # 폴더 이동
```

오늘의 규칙: 슬라이드의 명령을 '그대로' 칩니다

띄어쓰기, 대소문자, 하이픈(-) 개수까지 똑같이. 명령 앞의 'C:₩Users₩hong>' 는 컴퓨터가 알아서 보여 주는 부분이라 치지 않습니다. 결과 화면의 숫자(버전)는 컴퓨터마다 달라도 괜찮습니다.

① Python 설치 (1) 설치 파일 내려받기

python.org/downloads. Windows에서 접속하면 자동으로 Windows용 화면이 나옵니다 (2026년 8월 기준 화면)

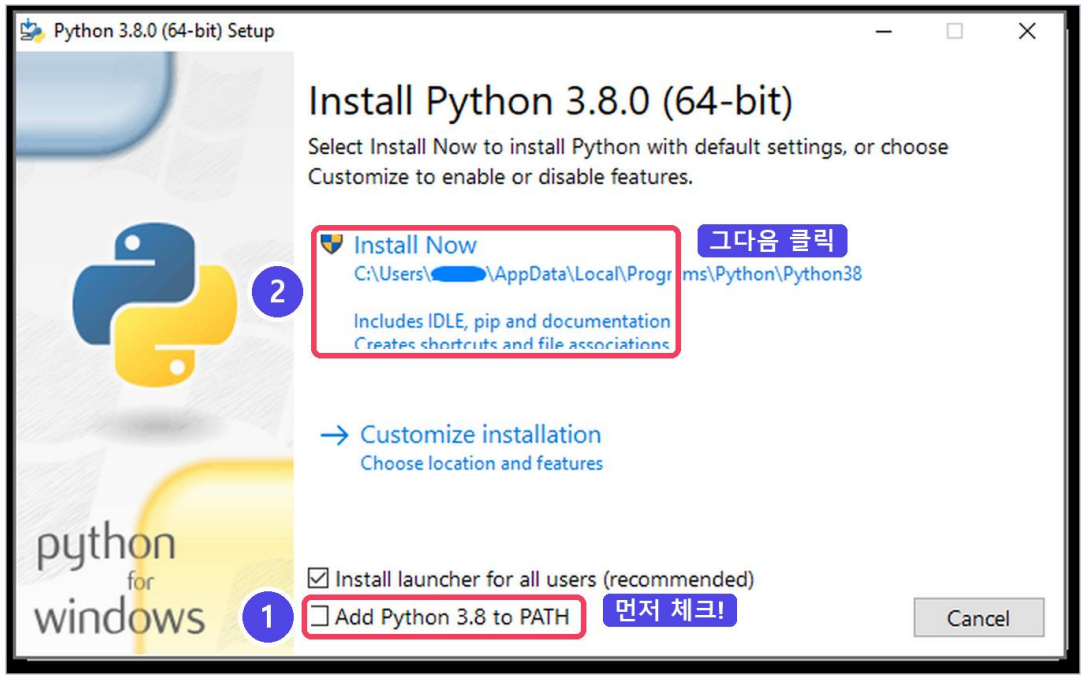
1 **주소창에 python.org/downloads**
브라우저(크롬·엣지) 맨 위 주소창에 입력하고 Enter.

2 **노란 버튼 말고, 그 아래 링크**
노란 버튼은 'Python install manager'라는 별도 프로그램입니다. 우리는 그 아래 "Or get the standalone installer for Python 3.14.x"의 파란 링크를 클릭 → python-3.14.x-amd64.exe 가 내려받아집니다.

3 **받은 파일 실행**
브라우저 오른쪽 위 다운로드 목록 또는 '다운로드' 폴더에서 python-3.14.x-amd64.exe 를 더블클릭 → 설치 화면이 뜹니다 (다음 장).

? **버전 숫자가 슬라이드와 달라요**
3.14.8, 3.15.0처럼 날짜에 따라 바뀝니다. 3.11 이상이면 무엇이든 괜찮습니다. Mac은 같은 화면에서 'macOS' 링크로 들어가 .pkg 파일을 받아 설치하면 됩니다 (Mac은 PATH 체크가 없습니다).

① Python 설치 (2) 설치 화면에서 딱 두 번 클릭



출처: docs.python.org (3.8 화면, 배치 동일). 최신 버전은 체크박스 글자가 'Add python.exe to PATH'

1 **맨 아래 'Add python.exe to PATH' 체크**
이걸 켜야 어느 폴더에서든 python 명령이 '먹습니다'. 오늘 실습에서 가장 많이 놓치는 지점입니다.

2 **Install Now → 'Setup was successful' → Close**

확인: 터미널(명령 프롬프트)을 '새로' 열고

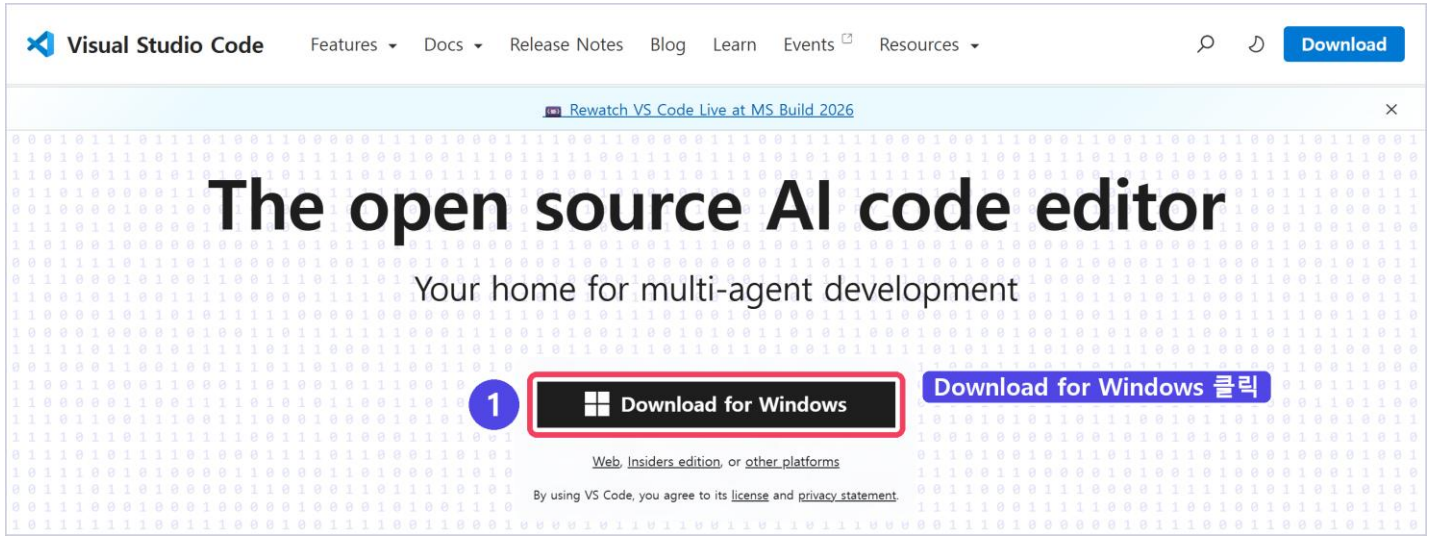
```
python --version
# Python 3.14.7  ← 버전 숫자가 찍히면 성공

py --version
# 위가 안 되면 Windows에선 py 로도 확인
# Mac은  python3 --version
```

PATH 체크를 놓쳤다면?
Windows 설정 → 앱 → 'Python 3.14' 제거 → 설치 파일을 다시 실행해서 체크하고 재설치. 1분이면 됩니다.

'python'을 치면 Microsoft Store가 열려요
PATH가 안 잡힌 신호입니다. 위와 같이 재설치하거나, 일단 py --version 으로 확인하고 넘어가도 됩니다.

② VS Code 설치: 내려받고, 전부 기본값으로 Next



code.visualstudio.com. 접속하면 내 운영체제에 맞는 'Download for Windows' 버튼이 보입니다

1 **접속 → 'Download for Windows' 클릭**
VSCodeUserSetup-x64-1.xx.exe 가 내려받아집니다.

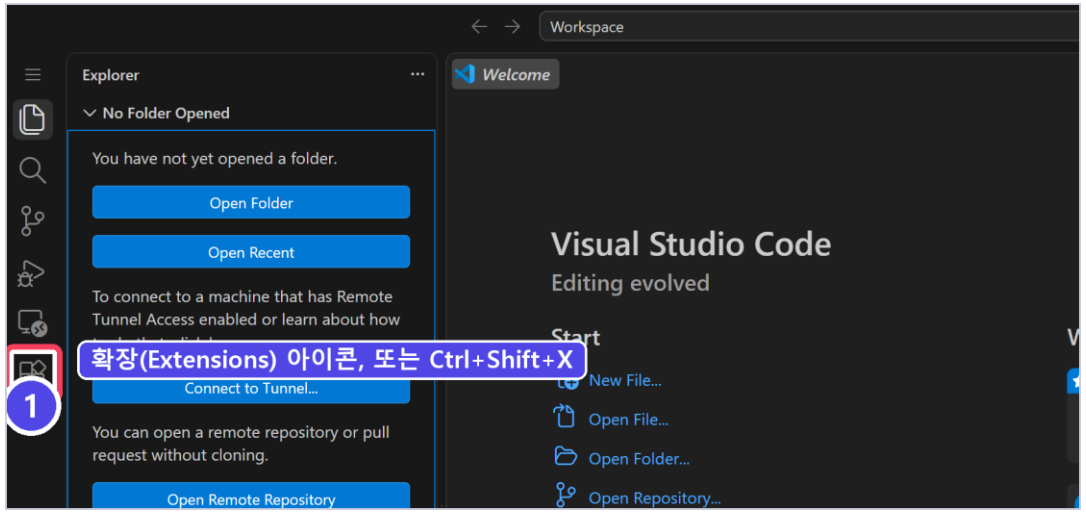
2 **실행 → 'I accept' → Next만 계속**
설치 위치, 시작 메뉴, 추가 작업 화면이 차례로 나오는데 전부 기본값 그대로 Next → Install → Finish. ('Launch Visual Studio Code' 체크는 켜 둔 채로.)

3 **VS Code 창이 뜨면 통과**
처음엔 영어 화면입니다. 다음 장에서 한국어 팩과 Python 확장을 설치합니다.

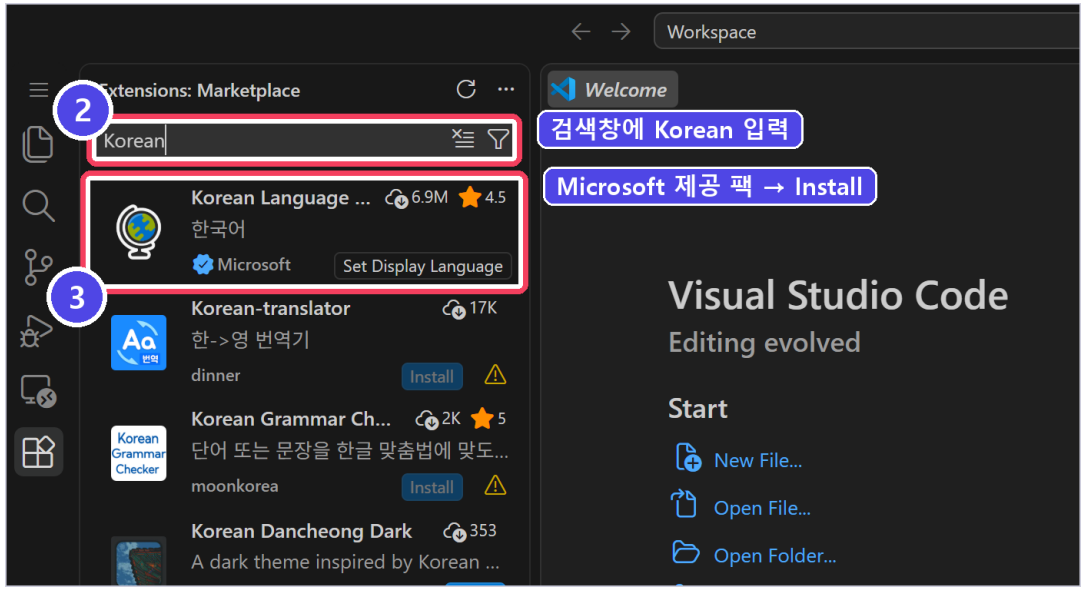
'확장(Extension)'이라는 개념. 다음 장에서 두 개 설치합니다

VS Code는 처음엔 빈 작업대입니다. 필요한 공구를 '확장'으로 끼워 넣는 구조라서, 오늘은 한국어 팩(화면을 한국어로)과 Python(파이썬 코드 색칠·실행 도우미) 두 개만 설치합니다. 다음 주에는 이 목록에 Codex가 추가됩니다. AI에게 지시하면 파일을 만들고 고쳐 주는 코딩 에이전트가 이 자리에 들어옵니다.

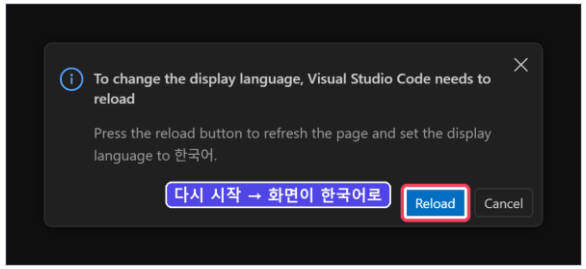
② VS Code 확장 설치 (1) 한국어 팩



① 왼쪽 세로 막대의 '확장' 아이콘 (네모 블록 네 개). 단축키 Ctrl+Shift+X



② 검색창에 Korean 입력 → ③ 'Korean Language Pack for Visual Studio Code' (Microsoft)의 Install



④ 설치 직후 알림에서 'Change Language and Restart'

4

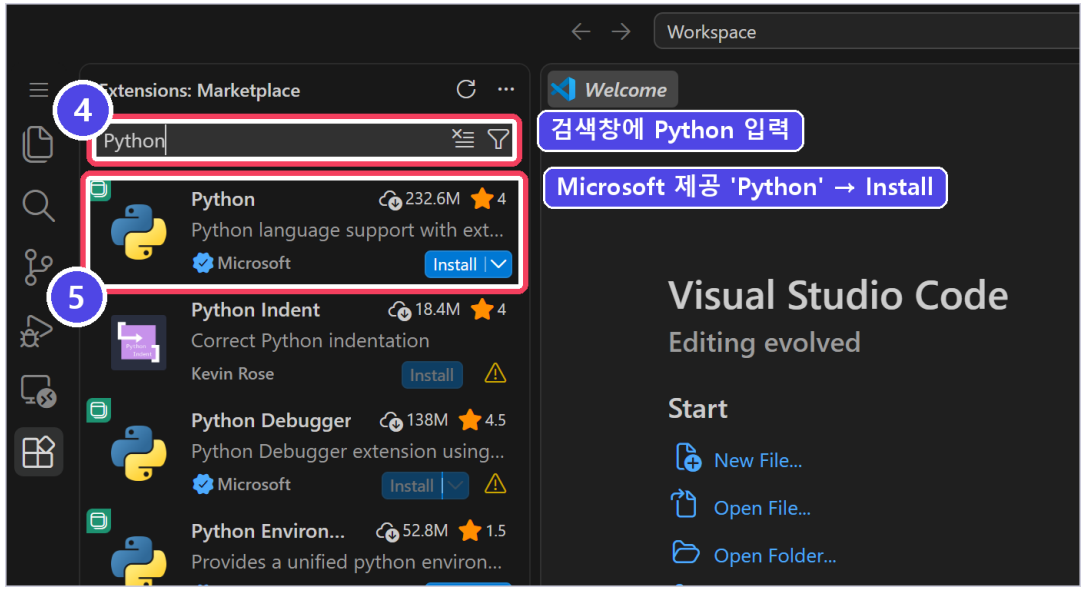
다시 시작하면 메뉴가 한국어로

'파일 · 편집 · 보기 … 터미널 · 도움말' 처럼 바뀌면 성공. 알림을 놓쳤으면 VS Code를 껐다 켜면 됩니다.

왜 한국어 팩부터?

이후 슬라이드의 메뉴 이름을 한국어로 안내하기 때문입니다. 영어 그대로 쓰고 싶은 사람은 건너뛰어도 됩니다.

② VS Code 확장 설치 (2) Python



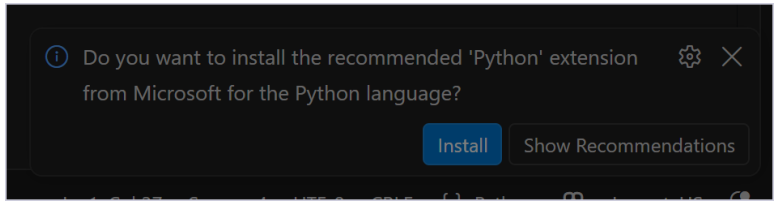
④ 검색창에 Python 입력 → ⑤ 맨 위 'Python' (Microsoft, 다운로드 2억 회 이상) 의 Install

5 **Microsoft가 만든 'Python'을 고르세요**

비슷한 이름이 여러 개 나옵니다. 제작자가 Microsoft이고 파란 체크 표시가 있는 맨 위 항목이 정답입니다. 함께 'Python Debugger', 'Pylance'가 자동으로 딸려 옵니다.

6 **Install 버튼이 톱니바퀴(⚙)로 바뀌면 완료**

설치는 10~30초. 왼쪽 확장 목록의 '설치됨(INSTALLED)' 칸에 한국어 팩과 Python 두 개가 보이면 이 단계 통과입니다.

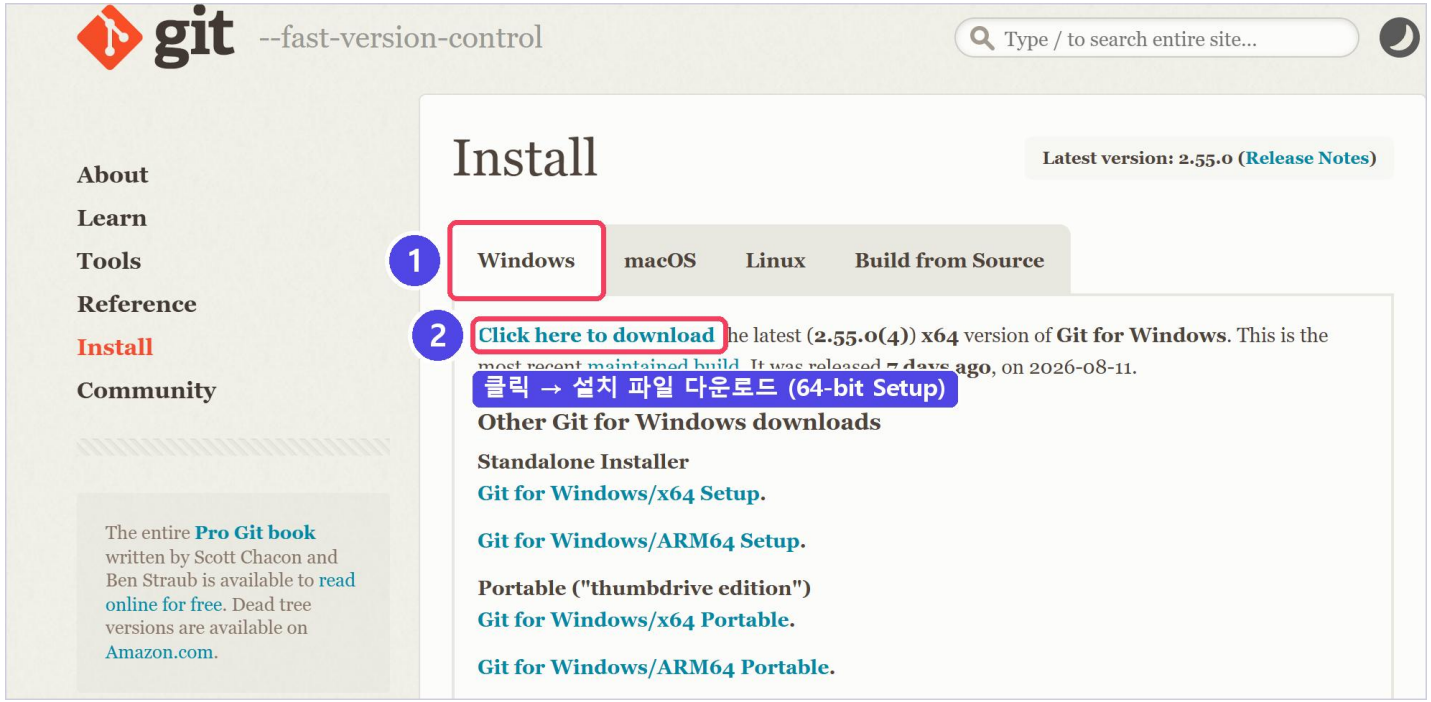


나중에 .py 파일을 열었을 때 이런 알림이 뜨면 그냥 Install을 누르면 됩니다.

이 단계 통과 기준

VS Code 메뉴가 한국어로 보이고, 확장 아이콘을 눌렀을 때 '설치됨' 목록에 Korean Language Pack 과 Python 이 있다. 둘 다 되면 손 들어서 확인받고 다음으로.

③ Git 설치: 화면이 많이 나와도 전부 Next



git-scm.com → 왼쪽 메뉴 'Install' → 'Windows' 탭. 'Click here to download' 가 64비트 설치 파일입니다

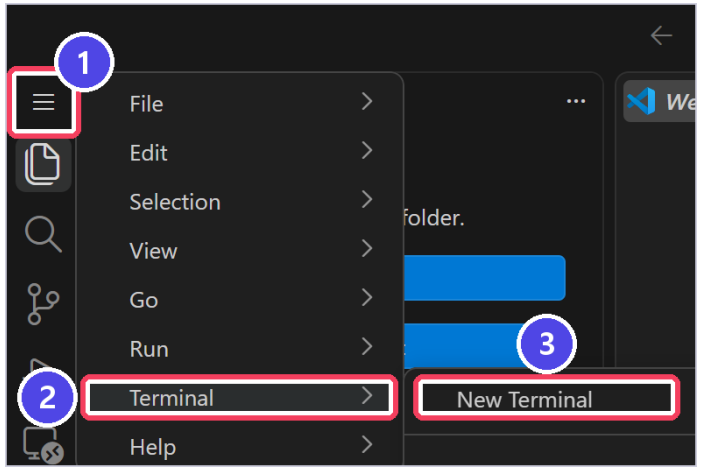
1 **git-scm.com → Install → Windows 탭**
주소창에 git-scm.com 입력 → 페이지 왼쪽 'Install' 클릭 → 'Windows' 탭이 선택돼 있는지 확인.

2 **'Click here to download' 클릭**
Git-2.xx.x-64-bit.exe 가 내려받아집니다. (Mac은 macOS 탭. 또는 터미널에 git 을 치면 설치 안내가 뜹니다.)

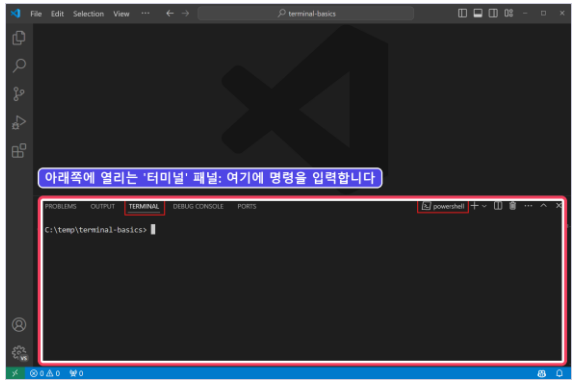
3 **실행 후 Next만 계속 → Finish**
에디터 선택, 브랜치 이름, PATH, 줄바꿈 등 낯선 옵션 화면이 계속 나옵니다. 하나도 바꾸지 말고 Next. 나중에 언제든 바꿀 수 있습니다.

Git이 뭐였죠?
파일의 변경 기록을 남기는 도구입니다. '누가, 언제, 무엇을' 바꿨는지 전부 남고, 되돌릴 수도 있습니다. 설치가 끝나도 아이콘이나 창은 안 뜹니다. 터미널에서 git 명령이 먹는지로 확인합니다 (다음 장).

③ Git 확인과 첫 설정: VS Code 안의 터미널에서



① 메뉴 → ② 터미널(Terminal) → ③ 새 터미널(New Terminal). 단축키 Ctrl+` (숫자 1 왼쪽의 ` 키). PC 설치판은 창 위쪽에 메뉴가 바로 보입니다.



아래쪽에 열리는 터미널 패널 (출처: VS Code Docs)

VS Code 터미널 (따옴표 안만 본인 것으로 바꿔서 입력)

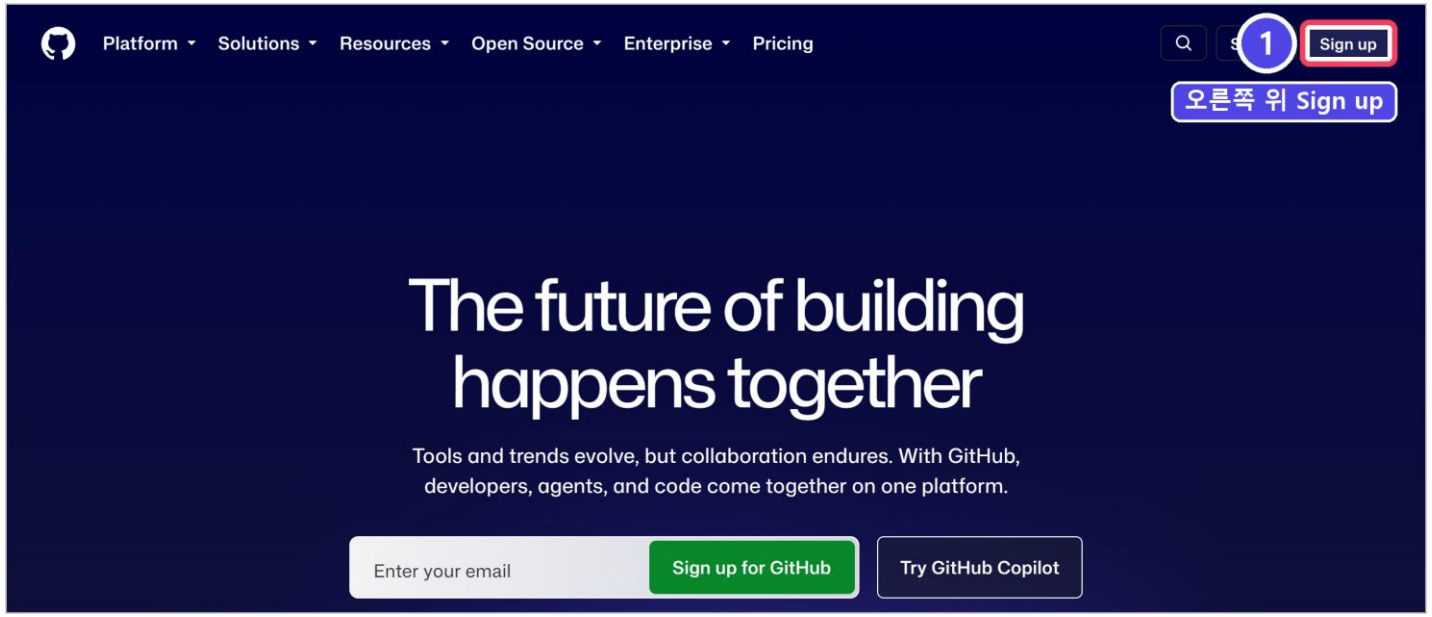
```
git --version
# git version 2.55.x  ← 숫자가 찍히면 설치 성공
git config --global user.name "Hong Gildong"
git config --global user.email "you@dju.kr"
git config --global --list      # 등록된 설정 확인
```

이름·이메일 = 기록에 남는 서명
앞으로 남길 모든 커밋(기록)에 붙습니다. 이메일은 잠시 후 만들 GitHub 계정과 같은 것으로. 그래야 GitHub가 '이 기록은 이 사람 것'이라고 연결해 줍니다. 회사·팀에서도 똑같이 합니다.

'git'을 찾을 수 없다고 나와요
설치 전에 켜 둔 VS Code는 새 프로그램을 모릅니다. VS Code를 완전히 끄고 다시 켠 뒤 새 터미널에서 다시 시도하세요.

통과 기준: git --version 에 숫자가 나오고, git config --global --list 에 user.name 과 user.email 이 보인다.

④ GitHub 계정 만들기



github.com 첫 화면. 오른쪽 위 'Sign up' (가운데 이메일 칸에 주소를 넣고 'Sign up for GitHub'를 눌러도 같은 절차)

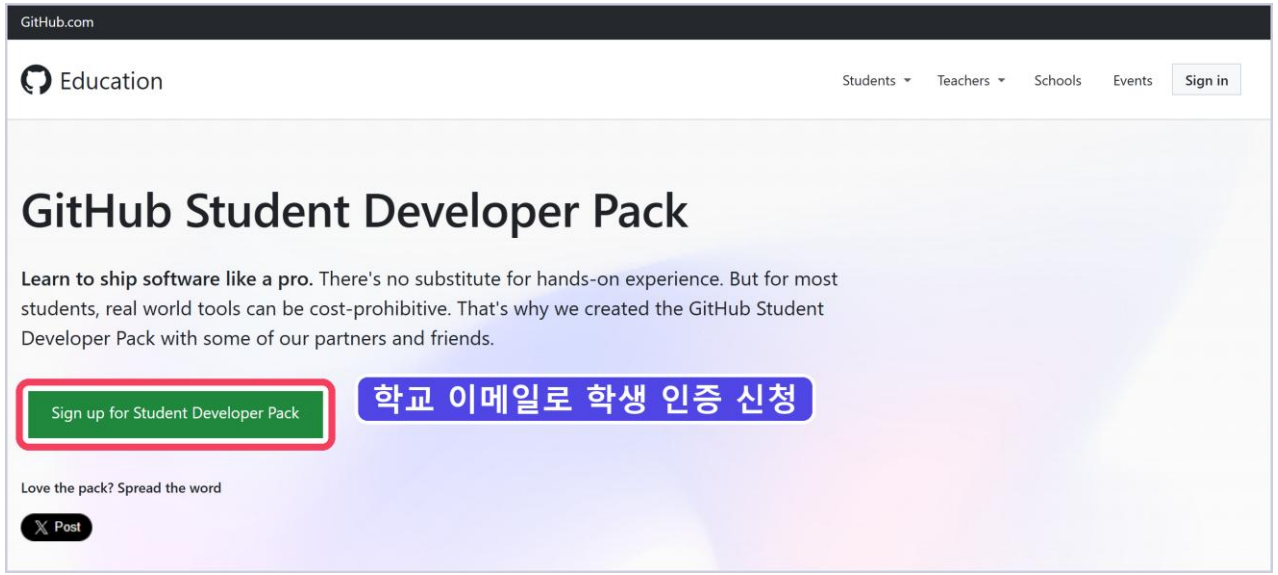
1 **github.com → Sign up**
이메일 → 비밀번호(15자 이상 또는 8자+숫자+소문자) → 아이디(Username) → 국가 순서로 입력하고 Continue.

2 **이메일 = Git에 등록한 그 주소**
학교 이메일을 권합니다 (학생 혜택 인증에도 쓰임). '사람인지 확인' 퍼즐이 나오면 풀고, 이메일로 온 8자리 코드를 입력하면 가입 완료.

3 **로그인된 첫 화면(대시보드)이 보이면 통과**
왼쪽에 'Top repositories', 오른쪽 위에 + 버튼과 내 프로필 사진이 보이는 화면입니다.

아이디(Username)는 신중하게. github.com/아이디 가 여러분의 공개 포트폴리오 주소
몇 년 뒤 입사 지원서에 그대로 적을 주소입니다. 실명 기반(hong-gildong)이나 깔끔한 닉네임을 추천하고, 장난스러운 아이디는 피하세요. 나중에 바꿀 수는 있지만 링크가 다 깨집니다.

(선택) 학생 인증 혜택, GitHub Student Developer Pack



education.github.com/pack. 초록 버튼으로 신청. 오늘 필수는 아니고, 나중에 여유 있을 때 하면 됩니다

주소를 못 외우겠으면 검색창에 'GitHub Student Pack' 이라고 치면 첫 번째로 나옵니다. 신청은 5분, 승인은 보통 1~7일.

무엇을 주나요

학교 이메일로 재학생 인증을 하면, 여러 유료 개발 도구를 재학 중 무료로 씁니다 (예: GitHub Copilot Pro, JetBrains 전체 IDE, 여러 클라우드 크레딧·도메인 등. 목록은 자주 바뀝니다).

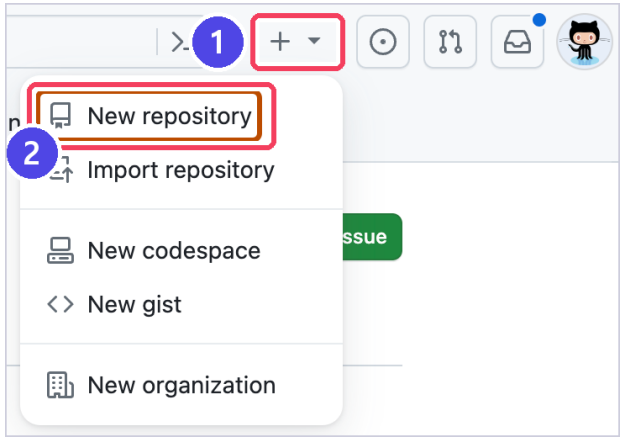
신청 방법

GitHub에 로그인한 상태로 위 주소 접속 → 'Sign up for Student Developer Pack' → 학교 이메일 인증 + 학생증(또는 재학증명서) 사진 업로드 → 며칠 안에 승인 메일.

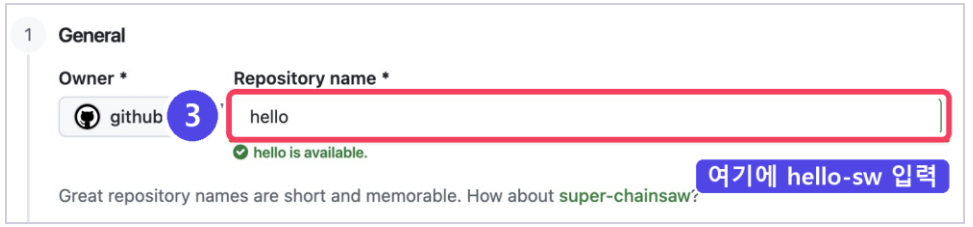
주의

GitHub 계정의 이메일에 학교 이메일이 추가·인증되어 있어야 합니다. 이번 주 과제와는 무관하니, 실습이 끝나고 집에서 천천히 하세요.

첫 저장소(Repository) 만들기: 이름은 hello-sw



① 오른쪽 위 + 버튼 → ② New repository (출처: GitHub Docs)



③ Owner는 본인 아이디, Repository name 칸에 hello-sw (영문 소문자·하이픈만). 초록 체크가 뜨면 사용 가능한 이름 (출처: GitHub Docs)

같은 화면 아래쪽에서 고를 것 (양식 요약)

4 **Public 선택**
누구나 볼 수 있는 공개 저장소. 과제 확인을 위해 Public으로 통일합니다.

5 **'Add a README file' 체크**
설명 파일 하나가 들어간 상태로 시작합니다. 그래야 바로 내 컴퓨터로 내려받아(clone) 작업할 수 있습니다.

6 **초록 'Create repository' 버튼**
누르면 저장소 페이지가 열립니다. 그 페이지가 다음 장의 화면입니다.

저장소(Repository)란

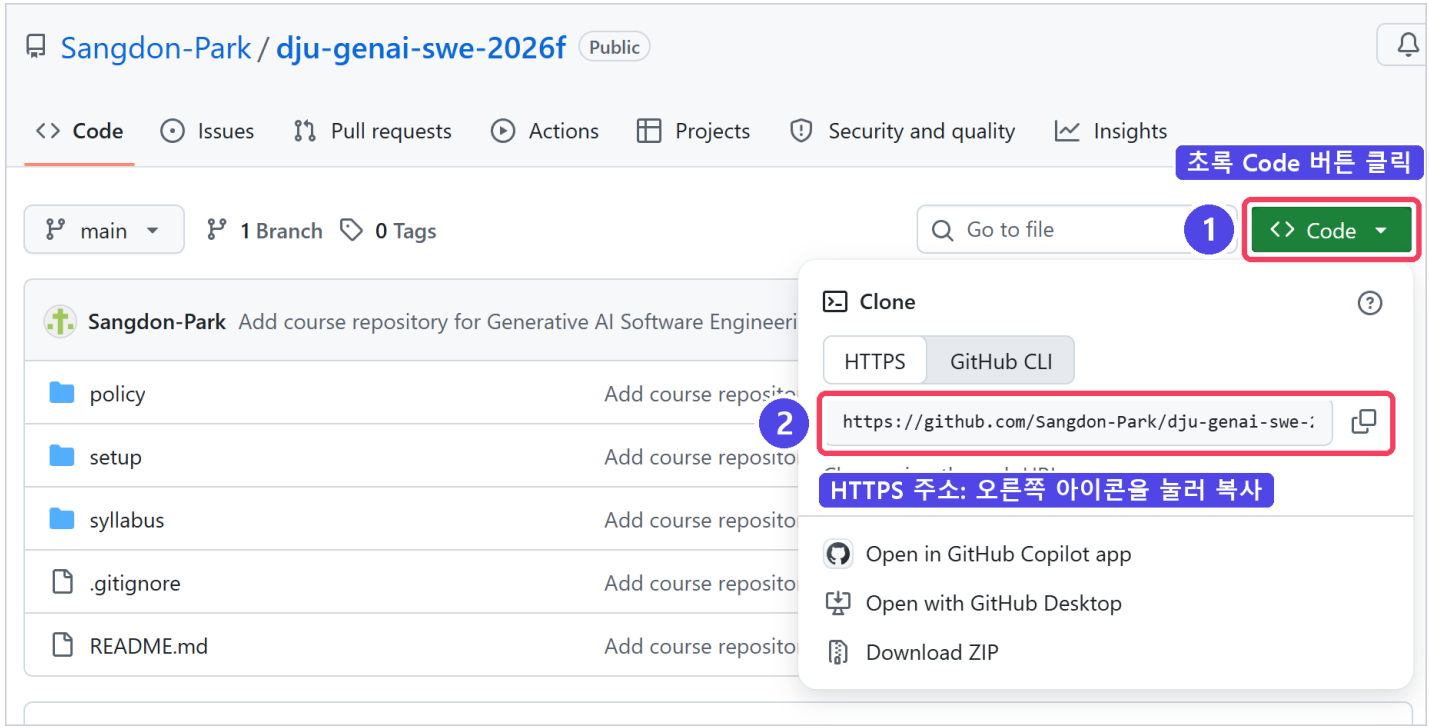
프로젝트 하나의 모든 파일과 변경 기록이 담기는 '서랍'입니다. 앞으로 과제 하나, 프로젝트 하나마다 저장소를 하나씩 만들게 됩니다.

이름 규칙: 영문 소문자와 하이픈(-)만. 띄어쓰기·한글은 피합니다.

README는 '이 저장소가 뭔지' 적어 두는 첫 파일입니다. GitHub가 저장소 페이지 아래에 자동으로 보여 줍니다.

확인: 주소창이 github.com/본인아이디/hello-sw 이고, 파일 목록에 README.md 하나가 보이면 통과.

내 컴퓨터로 가져오기(clone): 저장소 주소 복사



예시 화면은 제 강의 저장소(dju-genai-swe-2026f)입니다. 여러분은 자기 hello-sw 저장소 페이지에서 똑같이 하면 됩니다

1 초록 Code 버튼 클릭
저장소 페이지 오른쪽 위, 파일 목록 바로 위에 있습니다.

2 HTTPS 주소 옆 복사 아이콘 클릭
https://github.com/아이디/hello-sw.git 형태입니다. 'HTTPS' 탭이 선택돼 있는지 확인 (SSH 아님).

VS Code 터미널 (주소는 Ctrl+V)

```
git clone https://github.com/ID/hello-sw.git
cd hello-sw
```

clone = 저장소를 내 컴퓨터로 통째로 복사. 그런데 어디에 복사됐나요?
터미널에 표시된 현재 폴더(예: C:₩Users₩내이름) 안에 hello-sw 폴더가 생깁니다. cd hello-sw 로 그 폴더에 들어간 다음, VS Code 메뉴 [파일 → 폴더 열기]에서 그 hello-sw 폴더를 열면 왼쪽 탐색기에 README.md 가 보입니다.

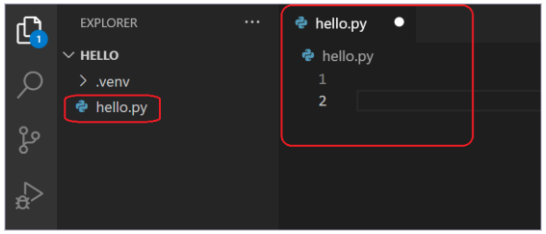
hello.py 만들어 GitHub에 올리기: add · commit · push

1 **VS Code에서 hello-sw 폴더 열기**

메뉴 [파일 → 폴더 열기] → 아까 생긴 hello-sw 선택. '작성자를 신뢰합니까?' 창이 뜨면 '예'.

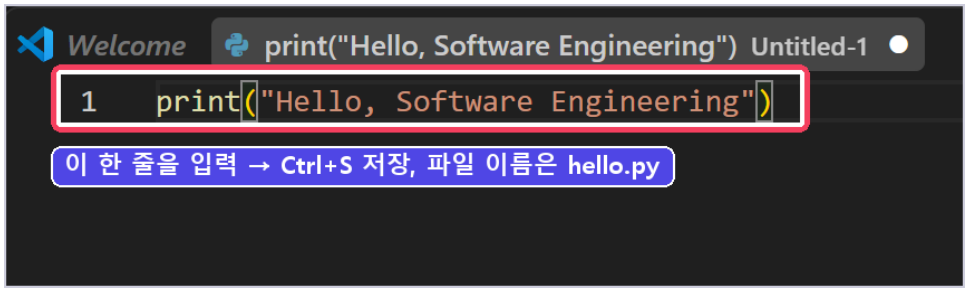
2 **새 파일 hello.py 만들기**

왼쪽 탐색기의 폴더 이름 옆 '새 파일' 아이콘 클릭 → 이름 hello.py 입력 → Enter. (또는 [파일 → 새 텍스트 파일] 후 저장할 때 hello.py)



탐색기와 편집기 탭에 hello.py 가 보이면 OK (출처: VS Code Docs)

3 **한 줄 입력하고 저장 (Ctrl+S)**



VS Code 터미널 (hello-sw 폴더 안에서, 한 줄씩 Enter)

```
git add hello.py
git commit -m "first commit"
git push
```

add: 올릴 파일 고르기

'이 파일을 다음 기록에 포함하겠다'는 표시. 장바구니에 담는 것과 비슷합니다.

commit: 메시지를 붙여 기록 남기기

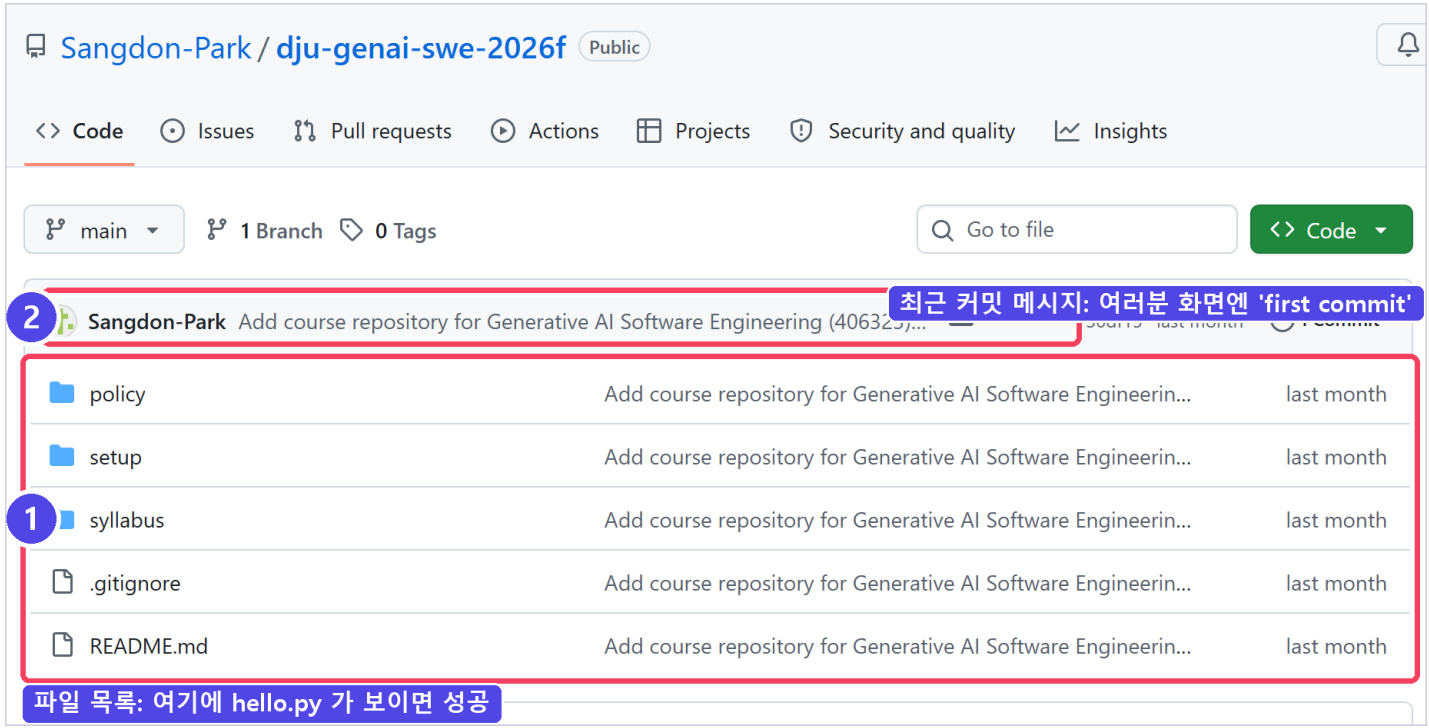
내 컴퓨터 안에 '사진 한 장'을 찍어 저장하는 것. -m 뒤 따옴표 안이 이 기록의 제목입니다.

push: 기록을 GitHub로 올리기

지금까지의 기록을 인터넷(GitHub 저장소)으로 전송. 이걸 해야 다른 사람·다른 컴퓨터에서 보입니다.

첫 push 때 브라우저에 GitHub 로그인 창이 뜹니다 → 로그인하고 'Authorize'. 한 번만 하면 이후엔 묻지 않습니다.

성공했는지 확인하기, 그리고 자주 막히는 세 곳



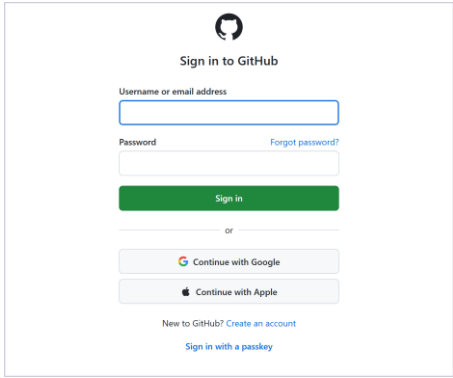
브라우저에서 내 저장소 페이지를 새로고침(F5). 파일 목록에 hello.py, 그 위에 'first commit'이 보이면 성공. (예시는 제 강의 저장소 화면)

성공했으면 저장소 주소를 LMS에 제출
브라우저 주소창의 github.com/본인아이디/hello-sw 를 복사해 LMS 1주차 게시판에 올립니다 (오늘 출석 인증). 다음 장 체크리스트도 확인.

인증 창이 안 떠요
터미널 메시지에 나온 링크(https://github.com/login/...)를 복사해 브라우저에 직접 붙여 넣고 로그인하세요.

push가 거부돼요 (403 / permission denied)
clone한 주소의 아이디가 본인 것인지 확인. 다른 사람 저장소를 clone했을 때 가장 자주 납니다. git remote -v 로 주소 확인.

한글이 깨져요
파일 인코딩을 UTF-8로. VS Code 오른쪽 아래에 'UTF-8'이 보이면 정상. 기본값이라 오늘은 거의 없습니다.



첫 push 때 브라우저에 뜨는 GitHub 로그인 화면. 방금 만든 계정으로 로그인 → Authorize

실습 체크리스트: 나가기 전에

- [ ] 터미널에서 python --version 이 버전을 출력한다.
- [ ] VS Code가 실행되고, 한국어 팩과 Python 확장이 설치되어 있다.
- [ ] git --version 이 동작하고 user.name / user.email 이 등록되어 있다.
- [ ] GitHub에 내 계정으로 로그인할 수 있다.
- [ ] 내 hello-sw 저장소에 hello.py 가 올라가 있다.

다섯 개 모두 통과가 오늘의 출석 인증입니다. 마지막 항목의 저장소 주소를 LMS 1주차 게시판에 제출하세요.

✔ 1강 · 정리

오늘의 세 문장

1. 소프트웨어공학은 코드가 아니라 '만드는 과정'의 학문입니다. 큰 실패는 언제나 과정에서 왔습니다.

2. 구현이 자동화될수록 정의 · 설계 · 검증 · 책임이라는 사람의 몫이 오히려 커집니다. 그 몫이 이 과목의 목차입니다.

3. 오늘 만든 Python · VS Code · Git · GitHub 환경이 한 학기의 작업대입니다. 다음 주, 이 작업대에 AI를 연결합니다.

예고 · 2강

다음 주: 프로세스 모델, 그리고 AI 연결

개발 프로세스 모델

폭포수 · 반복적 · 애자일. 만드는 '순서'가 결과를 어떻게 바꾸는지.

AI 도구 환경 구축 (실습)

ChatGPT 계정 → VS Code에 Codex 로그인 → ZCode 설치·공용 계정 연결 → 첫 지시.

준비물

ChatGPT 무료 계정(chatgpt.com). 없으면 미리 만들어 오세요.

오늘 설치한 환경 그대로의 노트북. 새로 설치할 것은 Codex 확장과 ZCode 프로그램입니다.

ZCode용 Z.ai 공용 계정은 수업 시간에 교수님이 로그인해 드립니다. 미리 준비할 건 없습니다.

오늘 수업은 여기까지입니다. 체크리스트 다섯 개, 확인하고 가세요. 수고하셨습니다!