생성형 AI 소프트웨어공학 · 2026학년도 2학기 · 2강

소프트웨어 개발 프로세스 모델

그리고 AI 도구 환경 구축: ChatGPT 계정으로 Codex 로그인 · 수업 공용 계정으로 ZCode 연결 · 첫 지시

대전대학교 컴퓨터공학과 · 박상돈

2강 · 수업 안내

지난 시간, 그리고 오늘

지난 시간 요약

소프트웨어공학은 '만드는 과정'의 학문이고, 큰 사고는 늘 과정에서 왔습니다.

구현이 자동화될수록 정의 · 설계 · 검증 · 책임이라는 사람의 몫이 커집니다.

그리고 여러분 노트북에는 이제 Python · VS Code · Git · GitHub 작업대가 있습니다. 오늘 그 위에 AI를 얹습니다.

1

개발 프로세스 모델

폭포수 · 반복적 · 애자일. 순서가 결과를 바꿉니다.

2

애자일 실무 용어

스프린트 · 데일리 스크럼 · 백로그. 팀 프로젝트의 공용어.

3

AI 도구 환경 구축 (실습)

ChatGPT 계정 → Codex 로그인 → ZCode 연결 → 첫 지시.

01

PART 1

개발 프로세스 모델

같은 것을 만들어도 '순서'가 다르면 결과가 다릅니다. 세 가지 대표 모델을 비교합니다.

프로세스 모델이란

소프트웨어를 만드는 **활동들의 순서와 방식**을 정해 둔 틀입니다. 무엇을 먼저 하고, 언제 확인하고,
어떻게 변경에 대응할지에 대한
팀의 합의입니다.

집 짓기로 비유하면 이렇습니다.

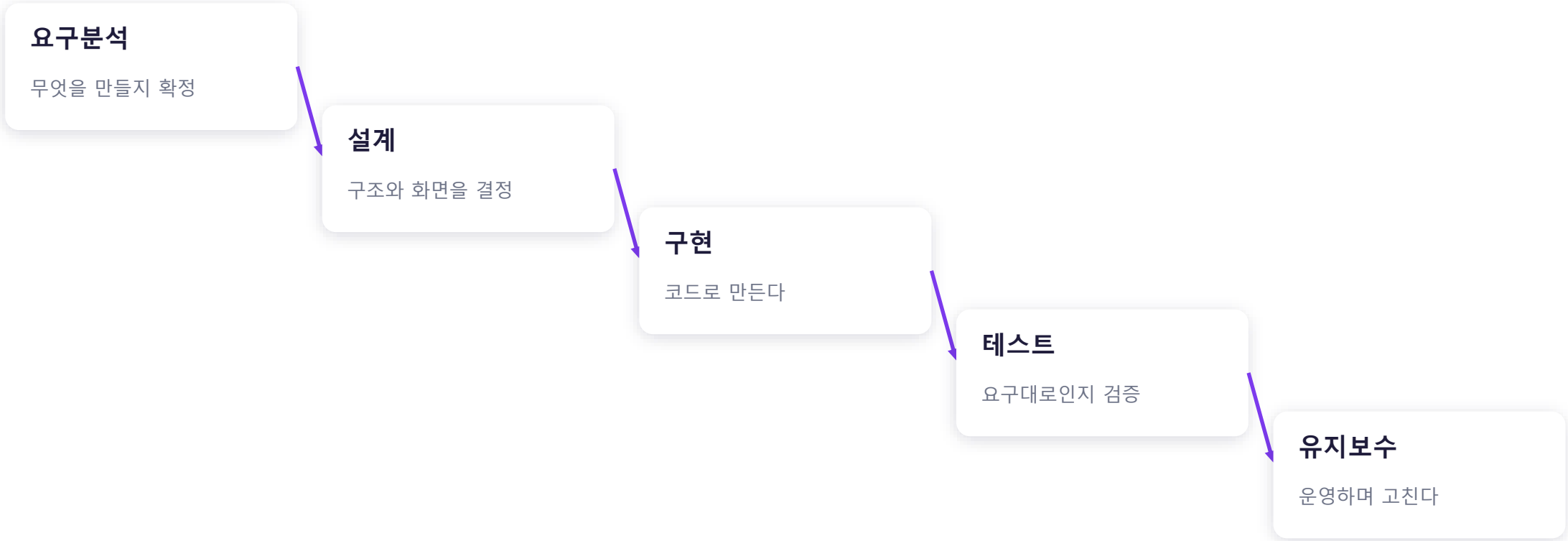
설계도 없이 벽부터 올리면

다 짓고 나서 "현관이 없네요"를 발견합니다. 순서가 틀리면 뒤에서 비용을 치릅니다.

네 명이 각자 순서로 지으면

지붕 팀과 기둥 팀이 서로를 기다리다 충돌합니다. 순서는 팀 전체의 합의여야 합니다.

폭포수(Waterfall) 모델: 한 방향으로 흐른다



핵심 성질
각 단계를 완전히 끝내고 문서로 인수인계하며 다음 단계로 갑니다. 물이 아래로만 흐르듯, 되돌아가는 것을 전제하지 않습니다.

폭포수의 빛과 그림자

장점: 명확함

단계와 산출물이 뚜렷해 계획 · 일정 · 역할 분담이 쉽습니다.

각 단계의 문서가 남아 인수인계와 감리에 강합니다.

요구가 처음부터 확정될 수 있는 프로젝트에서는 여전히 유효합니다.

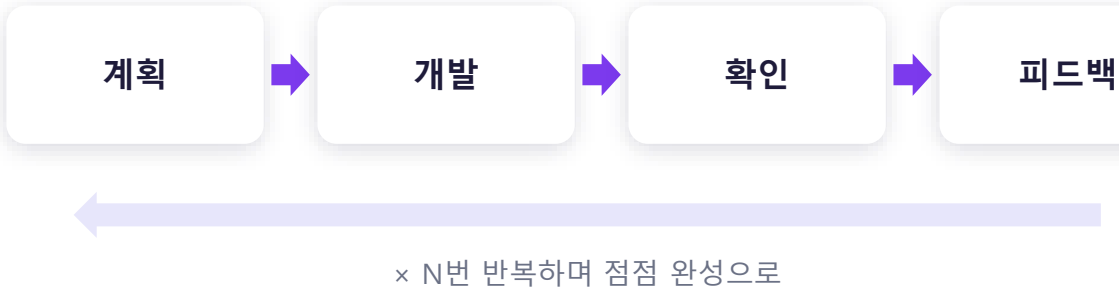
한계: 피드백이 늦다

고객은 맨 끝에야 동작하는 것을 봅니다. "다 만들었는데, 이게 아니에요"가 최악의 시나리오.

뒤늦게 발견된 요구 변경은 앞 단계 전부를 다시 여는 비용을 부릅니다.

처음에 요구를 완벽히 아는 프로젝트는 생각보다 드뭅니다.

반복적(Iterative) 모델: 작은 바퀴를 여러 번



한 번에 전부를 만드는 대신, 계획 → 개발 → 확인 → 피드백의 작은 바퀴를 여러 번 굴려 완성에 다가갑니다. 확인이 매 바퀴마다 있으니 "이게 아니에요"를 일찍 듣습니다.

증분(Incremental): 조각씩 완성

피자를 한 조각씩 완성해 내놓는 방식. 기능 1을 끝내 배포하고, 기능 2를 더 하고... 매 회차 '완성된 일부'가 나옵니다.

반복(Iterative): 전체를 다듬기

그림을 스케치 → 밑색 → 세부 묘사로 완성하는 방식. 매 회차 '전체의 더 나은 버전'이 나옵니다. 실제로는 둘을 섞어 씁니다.

애자일: 2001년, 열일곱 명의 선언

무거운 문서와 계획 중심 개발에 지친 개발자 17명이 모여 네 줄의 가치 선언을 발표했습니다.

개인과 상호작용	>	프로세스와 도구
작동하는 소프트웨어	>	포괄적인 문서
고객과의 협력	>	계약 협상
변화에 대응하기	>	계획을 따르기

오른쪽에도 가치가 있다, 다만 왼쪽을 '더' 높게 친다는 선언입니다. 문서를 버리자는 뜻이 아닙니다.

세 모델 한눈에 비교

구분	폭포수	반복적	애자일
피드백 시점	맨 끝 (전체 완성 후)	매 회차 끝	매일 · 매 스프린트
변경 대응	비용이 매우 크다	다음 회차에 반영	환영 (백로그로 수용)
산출 문서	단계마다 상세 문서	회차별 핵심 문서	필요한 만큼만
잘 맞는 상황	요구 확정 · 규제 · 안전 필수	규모가 크고 요구가 일부 유동적	요구가 불확실 · 빠른 출시와 학습

정답 모델은 없습니다. 상황이 모델을 고릅니다. 다음 장에서 상황별로 골라 봅시다.

상황이 모델을 고른다: 세 가지 사례

병원 의료기기 제어 SW

폭포수 계열

요구가 규격으로 확정되고, 인허가 문서와 단계별 검증이 필수. 변경보다 확실성이 우선입니다.

스타트업의 신규 서비스

애자일

무엇이 먹힐지 아무도 모릅니다. 작게 내놓고 사용자 반응으로 배우는 속도가 생명입니다.

우리 팀 프로젝트

애자일 + AI 반복

작게 만들고 자주 확인. 여기에 AI로 구현 속도까지 붙입니다. 자세한 방식은 2부에서.

02

PART 2

애자일 실무 용어

스프린트 · 데일리 스크럼 · 백로그. 팀 프로젝트에서 그대로 쓸 최소한의 공용어를 정리합니다.

스크럼(Scrum) 한눈에

애자일 가치를 실행하는 가장 널리 쓰이는 틀. 흐름은 이 한 줄입니다.

제품 백로그
할 일 전체의 우선순위 목록

→

스프린트 계획
이번 주기에 할 일을 골라 확정

→

스프린트 (1–4주)
매일 15분 데일리 스크럼

→

동작하는 증분
실제로 돌아가는 결과물 한 덩이

→

리뷰 · 회고
결과와 과정을 돌아보고 개선

다음 스프린트로. 이 바퀴를 프로젝트가 끝날 때까지 반복

왜 이 틀이 강한가
매 스프린트 끝에 '동작하는 것'이 나오고, 매일 15분의 확인이 있으니 문제가 커지기 전에 드러납니다. 폭포수의 "맨 끝의 충격"이 구조적으로 사라집니다.

팀 프로젝트 공용어 네 가지

스프린트

짧고 '고정된' 개발 주기(1–4주). 기간을 늘리지 않고, 못 끝낸 일은 다음 스프린트로 넘깁니다. 우리 프로젝트는 1주 스프린트로 갑니다.

데일리 스크럼

매일 15분, 서서 하는 확인. 어제 한 것 · 오늘 할 것 · 막힌 것 세 가지만 말합니다. 문제 해결 회의가 아니라 '드러내는' 자리입니다.

백로그

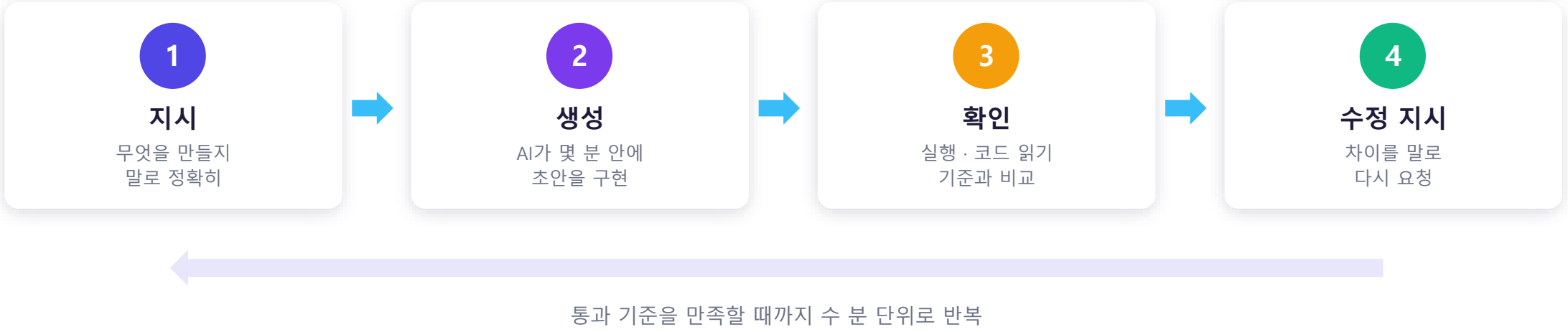
할 일의 우선순위 목록. 새 요구와 변경은 전부 여기로 들어와 순서를 받습니다. "변화를 환영"하는 실제 장치입니다.

회고

스프린트 끝에 결과물이 아니라 '일하는 방식'을 돌아봅니다. 좋았던 것 · 아쉬웠던 것 · 다음에 바꿀 것 하나.

AI 시대의 개발 주기: 스프린트가 분 단위로

구현이 몇 분으로 줄어들면, 병목은 '만들기'에서 '확인하기'로 이동합니다. 그리고 우리의 하루는 이 초소형 스프린트의 반복이 됩니다.



그래서 프로세스는 덜이 아니라 '더' 중요해집니다
바퀴가 빨라질수록 방향(요구·기준)이 틀리면 더 빨리 엉뚱한 곳에 도착합니다. 오늘 실습에서 이 바퀴를 처음으로 한 바퀴 돌려 봅니다.

03

PART 3

AI 도구 환경 구축

VS Code에 Codex를, 노트북에 ZCode를 붙이고 첫 지시를 내립니다. 계정 로그인부터 결과 확인까지.

오늘 만들 환경: 연결 구조 먼저

ChatGPT 계정 (무료)
chatgpt.com 가입
내 것, 비용 없음

로그인

Codex (VS Code 확장)
OpenAI의 코딩 에이전트
엔진 GPT-5.6

Z.ai 공용 계정
교수님 구독(GLM Coding Plan)
수업 시간에 로그인해 드림

로그인

ZCode (데스크톱 앱)
Z.ai의 코딩 에이전트 프로그램
엔진 GLM-5.3

집·과제용: 내 계정으로 쓰는 Codex

VS Code 안에 패널로 들어옵니다. ChatGPT에 로그인만 하면 되고 키가 필요 없습니다.

무료 플랜은 한도가 '제한적 체험' 수준이라 큰 작업엔 모자랄 수 있습니다. 막히면 ZCode로.

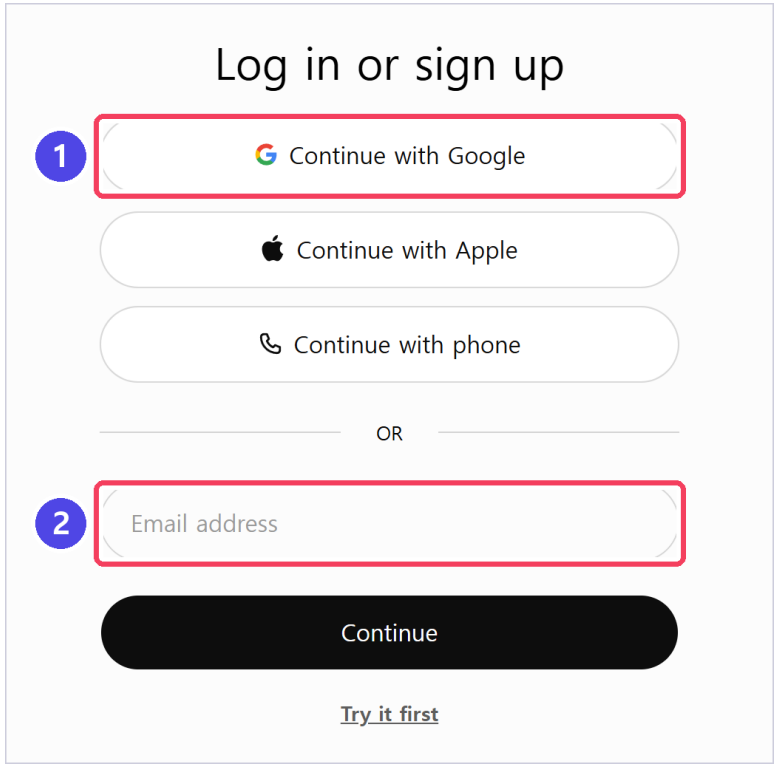
수업용: 반 전체가 나눠 쓰는 ZCode

VS Code와 별개의 프로그램입니다. 같은 hello-sw 폴더를 열어 두면 고친 파일이 양쪽에 다 보입니다.

교수님 계정 하나를 열 명이 나눠 쓰니, 지시는 작게. 계정 정보는 묻지도 적지도 않기.

둘 다 '지시를 받아 파일을 만들고 고치는 에이전트'입니다
수업 시간엔 ZCode(공용 계정), 집과 과제엔 Codex(내 계정). 한쪽이 막히면 다른 쪽으로 바꿔 씁니다. 오늘 실습은 이 네 조각을 위에서부터 하나씩 잇는 과정입니다. 지난주에 만든 작업대(VS Code, Git, GitHub)는 그대로 씁니다.

① ChatGPT 무료 계정 만들기 (Codex 로그인용)



chatgpt.com 의 가입·로그인 화면 (2026년 8월). 구글 계정이 있으면 ①이 제일 빠릅니다.

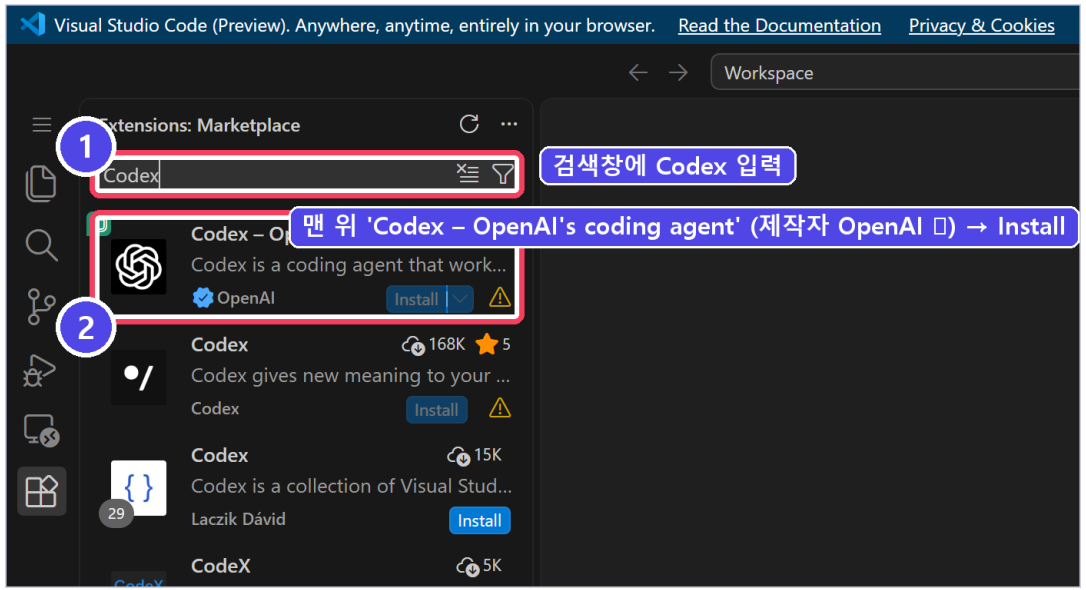
1 **주소창에 chatgpt.com → 오른쪽 위 'Sign up for free'**
이미 계정이 있으면 'Log in'만 하면 됩니다. 학교 구글 계정이 막히면 개인 계정으로.

2 **'Continue with Google' 또는 이메일 주소 → Continue**
이메일로 하면 인증 코드가 메일로 옵니다. 이름과 생년월일을 묻는 화면이 이어집니다.

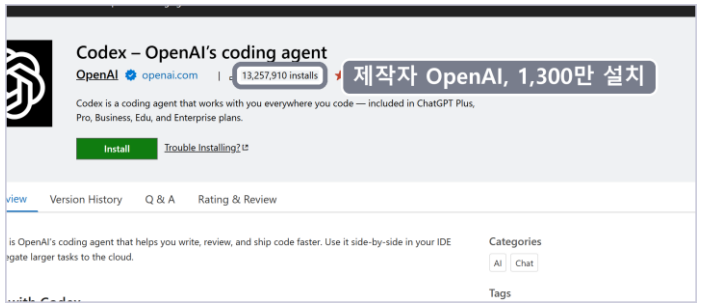
3 **로그인된 채로 두기**
다음 장에서 Codex가 이 로그인을 그대로 씁니다. 무료 플랜 그대로, 결제 정보는 넣지 않습니다.

무료 플랜에도 Codex가 들어 있습니다
OpenAI 도움말(2026년 8월) 기준으로 Codex는 Free·Go 플랜을 포함한 모든 ChatGPT 플랜에서 쓸 수 있습니다. 다만 무료는 '제한적 체험' 수준이라 공식 한도 숫자가 없고, 5시간 단위와 주 단위 한도가 같이 걸립니다. 막히면 수업 공용 ZCode로 바꾸거나, ChatGPT 앱에서 GPT-5.6 Luna에게 물어 코드를 붙여넣으면 됩니다.

② VS Code에 Codex 확장 설치



지난주와 같은 확장 메뉴. ① Codex 검색 → ② 맨 위 'Codex – OpenAI's coding agent' (제작자 OpenAI, 파란 체크) Install



같은 확장의 마켓플레이스 페이지 (marketplace.visualstudio.com). 설치 수 1,300만

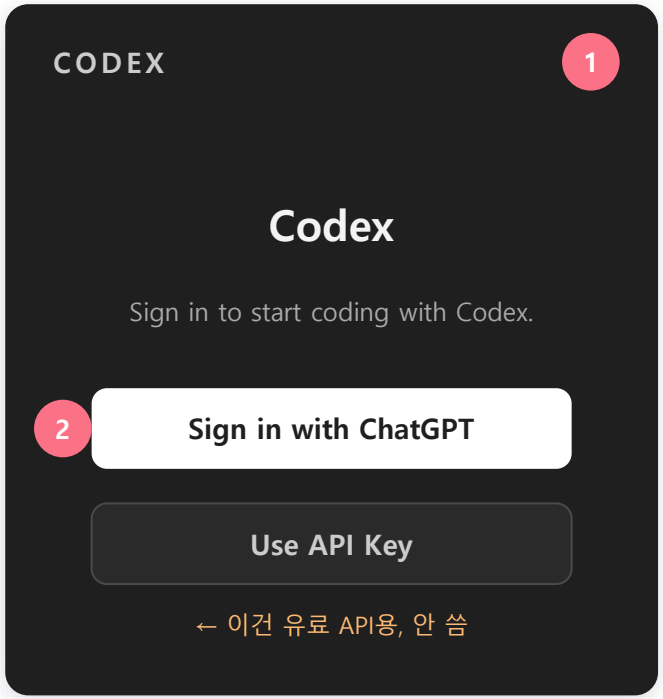
1 **확장 아이콘(Ctrl+Shift+X) → Codex 검색**
비슷한 이름(Codex, Codex-AI, codex-vscode...)이 많습니다. 제작자가 'OpenAI'이고 파란 체크가 있는 맨 위 항목이 정답.

2 **Install → 왼쪽 세로 막대에 Codex 아이콘 생김**
안 보이면 Ctrl+Shift+P를 누르고 'Codex: Open Codex Sidebar'를 실행하면 열립니다.

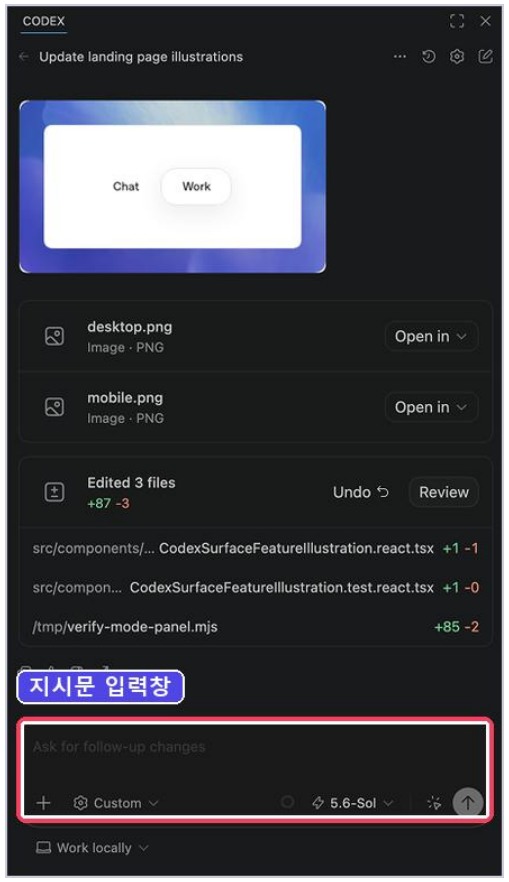
3 **아이콘을 누르면 오른쪽에 Codex 패널**
처음엔 로그인 화면이 뜹니다. 로그인은 다음 장에서.

Codex가 뭐였죠?
VS Code 안에서 '이 파일 만들어 줘, 저 기능 고쳐 줘' 라고 말하면 실제로 파일을 만들고 고쳐 주는 OpenAI의 코딩 에이전트입니다. 두뇌(AI 모델)는 ChatGPT 계정으로 빌려 쓰기 때문에 키가 필요 없고, 로그인만 하면 됩니다.

③ Codex 로그인: Sign in with ChatGPT



Codex 패널 첫 화면 구성 요약 (실제 화면은 확장 설치 후 VS Code 안에서만 보임). 버튼 이름은 OpenAI 공식 문서 기준.



로그인 후의 Codex 패널 예시 (출처: OpenAI 공식 문서). 맨 아래가 지시문 입력창.

1. **왼쪽 막대의 Codex 아이콘 → 패널 열기**
   안 보이면 Ctrl+Shift+P → 'Codex: Open Codex Sidebar'.

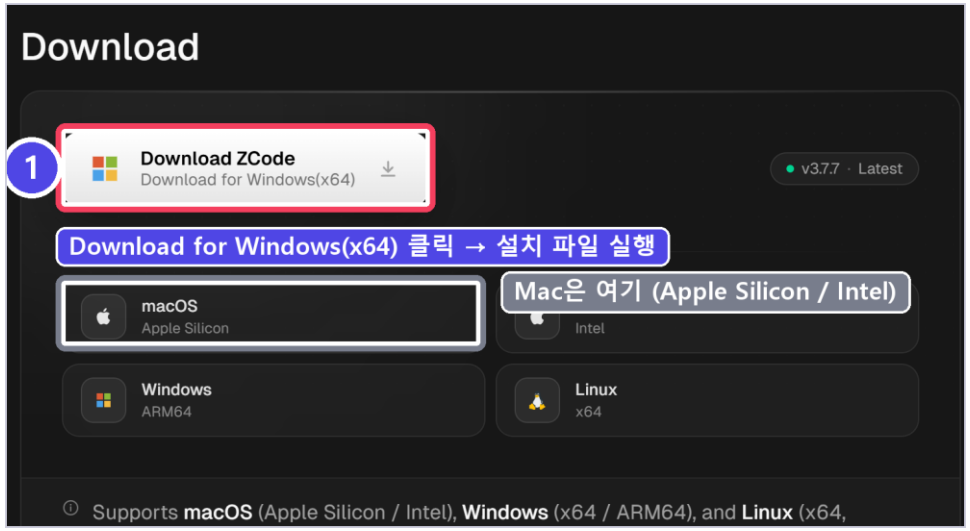
2. **'Sign in with ChatGPT' 클릭 → 브라우저가 열림**
   VS Code가 잠깐 브라우저로 넘깁니다. 팝업 허용을 물으면 허용.

3. **브라우저에서 ChatGPT 로그인 → 허용**
   아까 만든 계정으로 로그인하고 'Allow'(허용)를 누릅니다.
   브라우저가 'Visual Studio Code 열기'를 물으면 열기.

4. **VS Code로 돌아오면 입력창 → '안녕' 보내 보기**
   응답이 오면 연결 성공. 이 입력창에 앞으로 지시문을 씁니다.

브라우저에서 로그인은 됐는데 VS Code로 안 돌아오면,
패널의 'Sign in with ChatGPT'를 한 번 더 누르고 브라우저의
'열기' 허용을 누르세요. 학교 계정이 막히면 개인 계정으로.

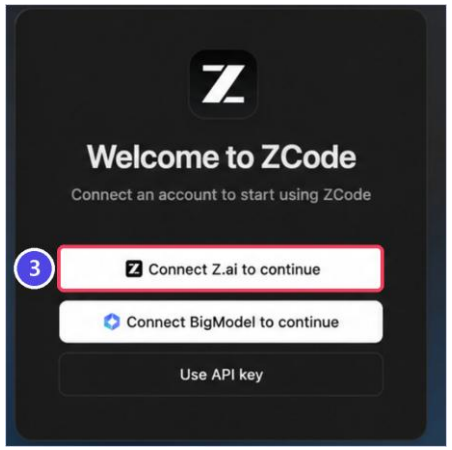
④ ZCode 설치 (zcode.z.ai)



zcode.z.ai/en/docs/install 의 다운로드 영역 (2026년 8월, v3.7.7). ① Windows(x64), Mac은 아래 칸.

1 **zcode.z.ai 접속 → Download**
Windows는 x64, 최근 맥북은 Apple Silicon. 무료 프로그램이고 가입 없이 받습니다.

2 **설치 파일 실행 → 마법사 따라 완료 → ZCode 실행**
Windows는 시작 메뉴 또는 바탕화면 아이콘. 'PC 보호' 경고가 뜨면 추가 정보 → 실행.

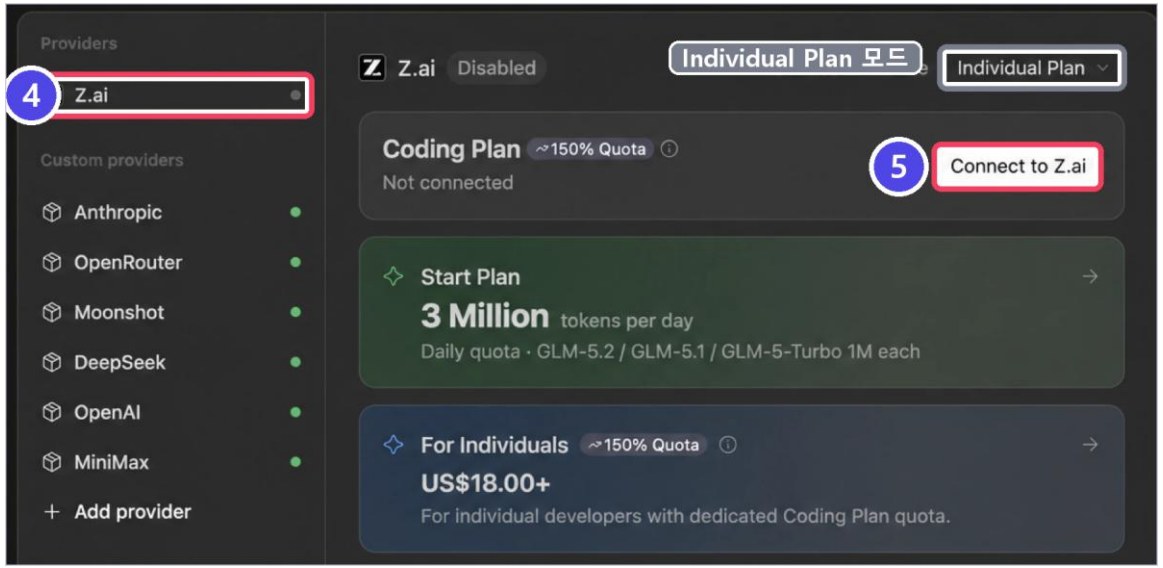


ZCode 첫 실행 화면 (출처: ZCode 공식 문서)

3 **첫 화면에서 'Connect Z.ai to continue'**
맨 위 버튼입니다. 브라우저가 열리면 교수님이 수업 공용 계정으로 로그인해 드립니다. 'Connect BigModel'과 'Use API key'는 쓰지 않습니다.

ZCode는 VS Code 확장이 아니라 따로 도는 프로그램입니다(Windows·Mac·Linux). 노트북에 설치가 막혀 있으면 오늘은 Codex로만 진행해도 됩니다.

⑤ ZCode에 수업 공용 계정 연결



ZCode 설정 → Model settings의 Z.ai 화면 (출처: ZCode 공식 문서). 'Individual Plan' 모드에서 'Connect to Z.ai'를 누르면 계정 로그인으로 넘어갑니다. 첫 화면에서 이미 연결했다면 여기서 확인만 합니다.

1. **'Connect Z.ai to continue' → 브라우저가 열림**
   교수님이 와서 수업 공용 계정으로 로그인해 드립니다. 비밀번호는 알려드리지 않습니다.

2. **브라우저에서 '허용(Authorize)' → ZCode로 돌아옴**
   인증이 끝나면 ZCode가 계정을 자동으로 묶습니다. 왼쪽 아래에 'Connected'가 보이면 성공.

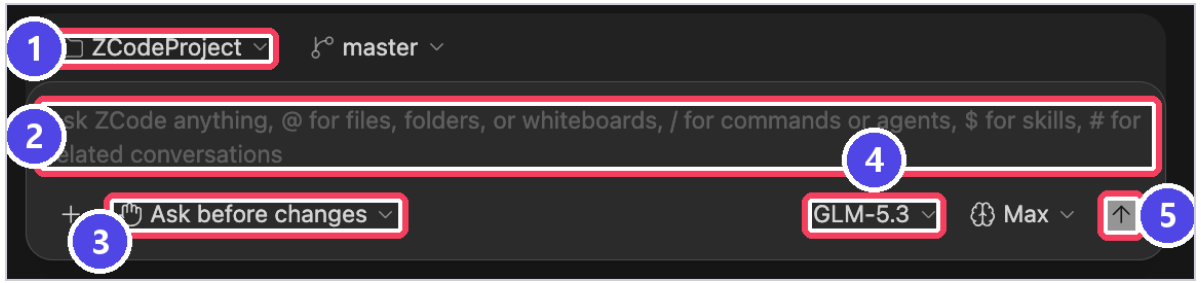
3. **모델은 GLM-5.3이 자동으로 잡힘**
   교수님 계정의 GLM Coding Plan 한도를 씁니다. 따로 키를 넣지 않습니다.

4. **확인: 왼쪽 아래 톱니바퀴 → Model settings → Z.ai**
   Connection mode가 'Individual Plan', Coding Plan이 'Connected'로 보이면 됩니다.

5. **끊겼으면 'Connect to Z.ai'로 다시**
   'Not connected'면 이 버튼으로 다시 로그인합니다. 그때도 교수님이 해 드립니다.

공용 계정은 수업용 노트북에서만 씁니다. 비밀번호를 묻지도, 적지도, 다른 기기에 로그인하지도 않습니다. 열 명이 계정 하나를 같이 쓰는 거라 한 명이 큰 작업을 돌리면 모두가 느려집니다. 지시는 작게.

⑥ ZCode 첫 화면: 폴더 열고 '안녕' 보내기



ZCode 새 작업 화면의 입력 영역 (출처: ZCode 공식 문서). ① 작업 폴더 ② 입력창 ③ 모드 ④ 모델 ⑤ 보내기

ZCode 화면은 이렇게 생겼습니다

가운데가 에이전트와의 대화. 왼쪽에 파일 목록, 아래에 터미널과 Git, 오른쪽에 미리보기를 열 수 있습니다.

입력창 위의 폴더 이름이 '지금 작업하는 폴더'입니다. 지난주 hello-sw 폴더를 열어 두면 Codex와 같은 파일을 봅니다.

'+'로 파일을 첨부하고, @로 파일 이름을 골라 넣을 수 있습니다. 오늘은 글자만 칩니다.

1 **Open Workspace → hello-sw 폴더 선택**
지난주 clone한 폴더입니다. 보통 C:₩Users₩내이름₩hello-sw.

2 **입력창에 '안녕' → 보내기(↑)**
응답이 오면 연결 성공. 첫 응답은 몇 초 걸릴 수 있습니다.

3 **모드는 'Ask before changes' 그대로**
파일을 고치기 전에 허락을 물어보는 기본 모드. 오늘은 이것만 씁니다.

4 **모델은 GLM-5.3**
옆의 'Max'는 생각 깊이. 기본값 그대로 둡니다.

5 **'Rate limit'·1302가 보이면 잠깐 대기**
반 전체가 같은 계정을 써서 줄을 선 겁니다. 몇 초 뒤 다시 보내면 됩니다.

여기까지 되면 두 에이전트가 다 붙은 겁니다. Codex는 VS Code 안에서, ZCode는 따로 뜬 창에서, 같은 hello-sw 폴더를 봅니다. 다음 장, 비밀번호 얘기를 하고 첫 지시로 갑니다.

키와 계정은 비밀번호입니다

코드에 직접 붙여넣지 않는다

api_key = "..." 처럼 소스 코드에 키를 박는 순간, 그 파일이 가는 모든 곳에 비밀번호가 따라갑니다. 앞으로 직접 키를 다룰 때 첫 번째 규칙입니다.

GitHub에 올라간 키는 몇 분 안에 수집됩니다

공개 저장소를 훑는 봇이 상시 돌고 있습니다. 키든 비밀번호든 새면 남이 내 이름으로 요청을 보내고, 한도와 요금과 계정이 한꺼번에 위험해집니다.

유출이 의심되면 즉시 알리기 · 비밀번호 바꾸기

수업 공용 계정이 이상하면 교수님에게 바로 알리세요. 내 ChatGPT 비밀번호가 샌 것 같으면 바로 바꾸고 로그인된 기기를 확인합니다. 실수를 숨기는 게 제일 나쁜 선택입니다.

그래서 비밀은 '도구 설정 안'에만 둡니다

ZCode는 교수님 계정 로그인으로, Codex는 내 ChatGPT 로그인으로 연결했습니다. 코드·메모·채팅·깃허브에는 키도 비밀번호도 적지 않습니다. 공용 계정 비밀번호는 묻지도 적지도 않습니다.

④ 첫 지시: 일부러 '작게' 시킵니다

ZCode(또는 Codex) 입력창에 보낼 지시문 (그대로 입력)

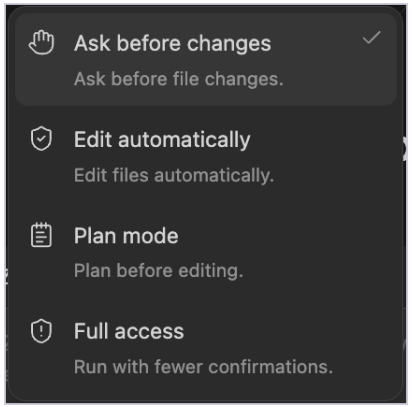
"temp.py 파일을 만들어 줘. 섭씨 온도를 화씨로 바꾸는 함수 c_to_f를 작성하고, 25도를 변환한 결과를 출력해."

왜 이렇게 '작게'?

결과를 눈으로 전부 확인할 수 있는 크기이기 때문입니다. 확인 못 할 크기로 시키는 것이 초보의 1번 실수입니다.

지시문에 들어간 세 가지

만들 파일 이름(temp.py) · 만들 것(함수 c_to_f) · 확인 방법(25도 변환 출력). 좋은 지시의 최소 골격입니다.



실행·수정 전에 이렇게 물어봅니다. ZCode 기본 모드 'Ask before changes'. 읽고 승인하는 습관을 오늘부터 (출처: ZCode 공식 문서)

ZCode가 'temp.py를 만들겠다'며 내용을 보여 주면 → 읽고 승인(Allow) → 파일이 생기면 성공. Codex도 같은 식으로 물어봅니다. 다음 장에서 결과를 확인합니다.

⑤ 결과 확인: 바퀴를 끝까지 돌립니다

temp.py

```python
# AI가 만든 temp.py 예시 (여러분 것과 다를 수 있음)
def c_to_f(c):
    return c * 9 / 5 + 32

print(c_to_f(25))

# 터미널에서 실행
python temp.py            # → 77.0
```

1. 실행한다
python temp.py 로 77.0이 나오는지.

2. 코드를 읽는다
무슨 일을 하는지 설명할 수 있는가. 설명 못 하면 내 것이 아닙니다(1강 규칙 ②).

3. 다르면 말로 수정 지시
"소수점 한 자리로 반올림해 줘"처럼, 기대와 결과의 차이를 말로 정리해 다시 요청합니다.

지시 → 생성 → 확인 → 수정 지시. 2부에서 본 초소형 스프린트를 방금 여러분이 한 바퀴 돌렸습니다.

두 도구, 언제 무엇을 쓰나

Codex (내 ChatGPT 무료 계정)

언제 집에서, 과제할 때, 혼자 공부할 때. 내 계정이라 유출 걱정이 없습니다.

한도 무료 플랜은 '제한적 체험' 수준. 5시간 창과 주간 한도가 같이 걸리고 공식 숫자는 없습니다.

막히면 ChatGPT 앱에서 GPT-5.6 Luna에게 묻고 코드를 붙여넣기(글 채팅은 사실상 무제한), 또는 수업 시간엔 ZCode.

ZCode (수업 공용 Z.ai 계정)

언제 수업 시간 실습. 모델 GLM-5.3이 똑똑하고 한도가 넉넉합니다.

한도 5시간 단위 양을 반 전체가 나눠 씁니다(열 명이면 1인당 지시 100번 넘게 가능). 동시에 몰리면 1302, 잠깐 대기.

규칙 계정 정보 묻지·적지 않기, 다른 기기 로그인 금지, 큰 작업 금지(모두가 느려짐), 학기 끝나면 로그아웃.

둘 다 같은 일을 합니다. 한쪽이 막히면 다른 쪽으로.
지시문은 양쪽에 똑같이 쓰면 됩니다. 다만 3강의 비교 실험처럼 '지시문만 바꿔서 비교'하는 날에는 반드시 같은 도구, 같은 모델로 세 실험을 돌려야 비교가 됩니다.

잘 안 될 때: 네 가지 점검

Codex 로그인이 안 돼요

브라우저에서 로그인은 됐는데 VS Code로 안 돌아오면 'Sign in with ChatGPT'를 다시 누르고 브라우저의 '열기'를 허용. 학교 계정이 막히면 개인 계정으로.

한도 초과 · 기다리는 중

Codex 무료 한도가 차면 ChatGPT 앱의 Luna에게 묻거나 ZCode로. ZCode의 5시간 한도가 다 찼으면 그 시간대는 Codex로. 둘 다 시간이 지나면 한도가 다시 찹니다.

ZCode 연결이 끊겼어요 · 1302

왼쪽 아래가 'Connected'인지 확인. 'Connect'로 바뀌어 있으면 다시 로그인이 필요하니 질문하세요. 1302·'Rate limit'은 반 전체가 동시에 눌러 줄을 선 것, 몇 초 뒤 다시.

그래도 안 되면

옆 사람의 설정 화면과 비교하고, 질문하세요. 오류 메시지를 지우지 말고 그대로 보여 주는 것이 가장 빠릅니다.

한도가 다 찼을 때: 수업 중엔 ZCode(공용 계정), 집에선 ChatGPT 앱에서 GPT-5.6 Luna에게 물어보고 코드를 붙여넣으세요. 둘 다 안 되면 다음 수업 전에 질문으로 남겨 주세요.

실습 체크리스트: 나가기 전에

- [x] ChatGPT 무료 계정에 로그인했다.
- [x] VS Code에 Codex 확장을 설치하고 ChatGPT로 로그인해 '안녕' 응답을 받았다.
- [x] ZCode를 설치하고 수업 공용 계정으로 연결해 '안녕' 응답을 받았다.
- [x] 지시로 temp.py 가 생성되고, 실행하면 77.0 이 나온다.
- [x] temp.py 를 hello-sw 저장소에 push하고 저장소 주소를 제출했다.

오늘의 출석 인증: temp.py를 지난주 hello-sw 저장소에 push하고, 저장소 주소를 LMS 2주차 게시판에 제출하세요. (키는 코드에 없죠? 그래서 올려도 안전합니다.)

2강 · 정리

오늘의 세 문장

1. 정답 프로세스 모델은 없습니다. 요구의 확실성과 변경 가능성이라는 '상황'이 폭포수 · 반복 · 애자일 중에서 고릅니다.

2. 애자일은 짧은 고정 주기와 잦은 확인의 문화입니다. AI는 그 주기를 분 단위 '지시 → 생성 → 확인 → 수정' 루프로 압축합니다.

3. 내 작업대에 AI가 연결되었습니다. 키는 도구 설정 안에만, 지시는 확인 가능한 크기로. 이 두 습관이 오늘의 진짜 결과물입니다.

예고 · 3강

다음 주: 생성형 AI의 이해와 지시문 작성 기초

생성형 AI는 어떻게 동작하는가

왜 같은 지시에 다른 답이 나오는지, 왜 그럴듯한 거짓말을 하는지. 원리를 알아야 부릴 수 있습니다.

지시문 작성 기초

역할 · 맥락 · 요구 · 형식. 결과 품질을 바꾸는 지시문의 구조를 배우고 직접 비교 실험합니다.

예고: 과제 1

같은 기능을 서로 다른 지시문 세 가지로 요청하고 결과를 비교·분석하는 과제가 3주차에 공지됩니다.

오늘 연결한 환경을 그대로 사용하니, 환경이 불안한 분은 이번 주 안에 꼭 정리해 두세요.

temp.py push와 저장소 주소 제출 잊지 마시고, 수고하셨습니다!