



CLAUDE CODE 기업 교육 · 상세 커리큘럼

모듈 20개 상세 강의안

모듈별 강의목표 · 대상 · 사전지식 · 산출물 · 실습환경 · 시간별 목차

초급 M01-M10

중급 M11-M20

각 모듈은 2~5시간 단위 · 09:00-18:00 코스로 2~3개 조합 가능 · 2026년 6월 기준

BEGINNER · 초급

초급 모듈 도구를 손에, 혼자 쓰는 흐름

10

M01

Claude Code·Claude.ai 완전 입문 — 설치부터 첫 작업까지

초급 · 2.5시간 · 입문 · 출처 공통 보강 +



강의 목표

Claude.ai 채팅과 Claude Code CLI의 차이를 이해하고, 설치·로그인·모델 선택을 끝낸 뒤 첫 작업(파일 읽기·간단 수정·커밋)을 직접 돌려본다. 하루 종일 이어질 본 과정을 따라올 최소 조작 감각을 손에 익힌다.



산출물

내 손으로 돌려본 첫 작업 로그 1건



강의대상

AI 코딩 도구가 거의 처음이거나, ChatGPT는 써봤지만 CLI형 에이전트는 안 써본 엔지니어. 비엔지니어(기획·디자인)도 따라올 수 있음.



사전지식

없음. 터미널을 한 번도 안 열어봤어도 됩니다.



실습환경

공통 환경 + 실습용 샘플 저장소 1개(클론해서 사용)



강의목차 및 내용 (시간별)

시간	내용
0:00-0:30	Claude.ai vs Claude Code — 채팅과 CLI 에이전트의 차이. Opus 4.8·Sonnet 4.6를 언제 쓰나. 비용·한도 개념.
0:30-1:10	설치·로그인 따라 하기. 사내 프록시·방화벽에서 막히는 지점과 우회. (5분 따라 하기: 설치 → claude 실행 → 모델 확인)
1:10-1:50	첫 작업. 샘플 저장소를 읽고 수정 → diff 확인 → 커밋. 'AI에게 맡기고 사람이 검증한다'는 기본 리듬 체득.
1:50-2:30	자주 막히는 5가지(권한·컨텍스트 넘침·영똥한 파일 수정 등)와 대처. 다음 모듈 지도 그려주기.



강의 목표

AI가 코드를 빨리 짜도 프로젝트가 흔들리는 건 '운용'이 없어서다. 요구마다 ID(REQ-ID)를 붙이고, 계획·구현·검증·반영이라는 4게이트를 차례로 통과시키는 기본기를 익힌다. 요구 하나를 게이트 끝까지 직접 밀어 본다.



산출물

REQ-ID 1건의 4게이트 통과 기록 + 우리 팀용 게이트 템플릿 초안



강의 대상

AI로 코드는 빨라졌는데 품질·일관성이 들쭉날쭉한 1년 차, 작업을 추적 가능하게 관리하고 싶은 테크리드.



사전 지식

Git 브랜치·커밋·PR 경험, Claude Code 기본 조작(M01 수준), 마크다운 읽기.



실습 환경

공통 환경 + 회원관리 SaaS 예제 저장소(언어 불문, 개념 중심)



강의 목차 및 내용 (시간별)

시간	내용
0:00-0:40	이런 상황 — 'AI가 짜준 코드가 쌓일수록 통제가 안 된다.' 코드 속도와 프로젝트 안정성은 별개라는 문제 정의.
0:40-1:30	요구 주도란. 모든 작업을 요구(REQ-ID) 단위로 묶고, AI에게 시키는 일도 요구에 연결. 추적성이 결재·보고 문화와 잘 맞는 이유.
1:30-2:20	4게이트(계획→구현→검증→반영) 개념과 각 게이트의 승인 포인트. (5분 따라 하기: 요구 1건에 ID 붙이고 계획 게이트 통과)
2:20-3:00	따라 하기 — 요구 하나를 4게이트 끝까지. 더 깊게: 세션 분리·권위 문서로 가는 길(M11 예고).



강의 목표

Claude Code의 -p(프롬프트) 플래그를 이해하고, 상시 켜둘 호스트(Raspberry Pi 5 또는 사내 리눅스 서버) 위에서 첫 자율 에이전트를 띄운다. 폴더에 파일이 들어오면 알아서 요약 리포트를 만드는 봇을 손으로 완성한다.



산출물

파일 감시 → 자동 리포트가 도는 봇 1개



강의대상

야간 배치·모니터링·정기 리포트 같은 무인 작업을 자동화하고 싶은 실무자, 자율 자동화를 처음 제대로 다뤄보는 엔지니어.



사전지식

리눅스 셸 기본(ssh·cd·파일 편집), cron이 뭔지 안다, Claude Code 기본 조작.



실습환경

공통 환경 + Raspberry Pi 5(없으면 리눅스 서버·미니 PC·클라우드 VM), cron 사용 가능 환경



강의목차 및 내용 (시간별)

시간	내용
0:00-0:30	이런 상황 – 매일 같은 시간에 로그를 모아 요약하는 단순 반복. 사람이 붙어 있어야만 돌아가는 작업의 한계.
0:30-1:20	상시 호스트 만들기(핸즈온). SSH 접속, Claude Code 설치, -p 플래그로 한 번에 시키는 비대화형 실행 이해.
1:20-2:20	첫 에이전트(핸즈온). 폴더 감시 → 변화 감지 → 요약 리포트 생성. (5분 따라 하기: cron에 5분마다 실행 등록)
2:20-3:00	돌려보고 로그 확인. 더 깊게: 판단 기준을 가르치고 폭주를 막는 길(M12 예고).



강의 목표

강력한 LLM(엔진)을 테스트·권한·리뷰·지시 파일(하네스)로 감싸 통제한다는 '하네스 엔지니어링' 개념을 잡는다. 작업마다 어디까지 맡길지 정하는 제어 레벨을 설정하고, 나쁜 지시와 좋은 지시의 차이를 직접 비교한다.



산출물

내 작업 3종의 제어 레벨 표 + 좋은 작업 지시 템플릿



강의대상

AI가 짠 코드를 통제해본 적이 거의 없는 1년 차, 팀 도입 전 개념부터 잡고 싶은 테크리드, 상류 산출물을 만드는 기획·PM.



사전지식

터미널·Git 기본(commit/diff/branch), 어느 언어든 코드 리뷰(diff 읽기) 경험.



실습환경

공통 환경 + Claude Code (OpenCode는 중급 M14에서 사용)



강의목차 및 내용 (시간별)

시간	내용
0:00-0:40	이런 상황 – AI에게 '알아서 해줘'라고 던졌다가 엉뚱한 결과를 받는다. 에이전트는 챗봇이 아니라는 점.
0:40-1:30	하네스 엔지니어링이란. 엔진(모델)을 테스트·권한·리뷰·지시 파일 네 띠로 감싸는 그림. AI 시대에 남는 일과 바뀌는 일.
1:30-2:20	제어 레벨 결정 – 작업 위험도에 따라 자동화 수위를 다르게. (5분 따라 하기: 내 작업 하나를 레벨로 분류)
2:20-3:00	나쁜 지시 vs 좋은 지시 비교 실습. 더 깊게: 상류·실장 공정 하네스로 가는 길(M13·M14 예고).

M05

Agent Skills 입문 — 공식 스킬로 반복 작업 없애기

초급 · 3시간 · 입문→실무 · 출처 Agent Skills



강의 목표

'이 작업은 이렇게 한다'를 LLM에게 가르치는 오픈 표준 Agent Skills의 개념을 잡고, 공식·레퍼런스 스킬을 골라 내 작업에 붙여 쓴다. 매년 복붙하던 프롬프트를 스킬 한 번 호출로 끝내는 경험을 손에 익힌다.



산출물

내 반복 작업에 공식 스킬을 붙여 돌려본 사례 1건



강의 대상

같은 지시를 매년 복붙하는 불편을 느낀 엔지니어, 팀 작업 방식을 표준화하고 싶은 실무 리더.



사전 지식

터미널 기본(cd·ls·git), Claude Code 기본 조작, 마크다운 읽기.
YAML/JSON을 눈으로 읽으면 더 수월.



실습 환경

공통 환경 + 공식 스킬 저장소 접근(github.com/anthropics/skills 등)



강의 목차 및 내용 (시간별)

시간	내용
0:00-0:40	이런 상황 — 문서 변환·데이터 정리 같은 정형 작업을 매년 같은 프롬프트로 시킨다. 사람마다 쓰는 법이 제각각인 문제.
0:40-1:30	Agent Skills란. 스킬이 어떻게 자동 발동되는지(설명 매칭), 스킬 폴더 구조(SKILL.md)를 눈으로 본다.
1:30-2:20	공식·레퍼런스 스킬 활용(핸즈온). 내 작업에 맞는 스킬을 붙여 돌려본다. (5분 따라 하기: 스킬 하나 설치 → 호출)
2:20-3:00	결과 비교 — 복붙 방식과 스킬 방식의 차이. 더 깊게: 나만의 스킬을 만들고 배포하는 길(M15 예고).



강의 목표

Karpathy가 제안한 LLM Wiki 패턴이 왜 RAG와 다른지 이해하고, 원천 자료(raw)-구조화 페이지(wiki)-메타로 나뉜 3층 구조를 잡는다. 환경을 깔고 스키마를 작성해 읽을수록 똑똑해지는 '베이스의 뼈대를 세운다.



산출물

3층 구조로 초기화된 내 지식 베이스 + 스키마 1종



강의대상

흩어진 사내 지식을 한곳에 모으고 싶은 엔지니어, 지식 관리 자동화를 검토하는 실무 리더.



사전지식

터미널 기본, Python 읽기(짧은 스크립트 수정 가능 수준), Git 기본. Claude Code는 처음이어도 됨.



실습환경

공통 환경 + Obsidian(또는 마크다운 뷰어), 빈 지식 저장소



강의목차 및 내용 (시간별)

시간	내용
0:00-0:30	이런 상황 — 좋은 자료를 모아두고도 다시 못 찾는다. 검색만으로는 지식이 쌓이지 않는 문제.
0:30-1:10	LLM Wiki란, RAG와 뭐가 다른가. raw/wiki/메타 3층 구조의 역할 분담.
1:10-1:50	환경 구축·스키마 작성(핸즈온). 지식 페이지가 어떤 모양으로 저장될지 정한다. (5분 따라 하기: 저장소 초기화 → 첫 페이지)
1:50-2:30	첫 자료를 손으로 넣어 구조 확인. 더 깊게: RSS 자동수집·자동 정리로 가는 길(M16 예고).



강의 목표

프로젝트 규칙·말투·금기를 담은 CLAUDE.md를 설계해 Claude Code를 '우리 프로젝트를 아는 동료'로 길들인다. 정형 작업을 Skills로 자동화하고, 세션을 넘어 지식을 쌓는 Memory의 기본을 잡는다.



산출물

우리 프로젝트용 CLAUDE.md 초안 + Skill 1개 적용



강의대상

Claude Code를 일상에 들이기 시작한 1년 차, 매번 같은 맥락을 다시 다시 설명하느라 지친 실무자.



사전지식

터미널·Git 기본, 코드를 읽고 작은 스크립트를 고칠 수 있음. M01 수준의 조작.



실습환경

공통 환경 + 실습용 프로젝트 저장소



강의목차 및 내용 (시간별)

시간	내용
0:00-0:40	이런 상황 — 세션을 새로 열 때마다 프로젝트 규칙을 처음부터 설명한다. 맥락이 휘발되는 한계.
0:40-1:30	CLAUDE.md 설계. 무엇을 적고 무엇을 빼는가, 길이와 비용의 관계. (5분 따라 하기: 규칙 5줄로 시작하는 CLAUDE.md)
1:30-2:20	Skills로 정형 작업 자동화. Memory로 세션을 넘는 지식 축적의 기초.
2:20-3:00	따라 하기 — CLAUDE.md 적용 전후 비교. 더 깊게: n8n·스케줄·병렬 에이전트로 가는 길(M17 예고).



강의 목표

Claude Code를 기능 하나하나로 외우는 대신 '흐름'으로 보는 관점을 잡는다. 컨텍스트를 의도적으로 설계하고, Plan 모드로 먼저 계획을 세운 뒤 TodoWrite로 작업을 쪼개 추적하는 기본 리듬을 익힌다.



산출물

Plan→Todo로 쪼갠 작업 흐름 1건 + 컨텍스트 설계 메모



강의대상

Claude Code를 좀 써봤지만 '기능은 아는데 흐름으로 못 엮는' 1년 차, 개발 흐름을 다듬고 싶은 실무자.



사전지식

터미널·Git 기본(branch-diff), JSON 읽기(settings.json), M01 수준 조작



실습환경

공통 환경 + 실습용 저장소, settings.json 편집 가능



강의목차 및 내용 (시간별)

시간	내용
0:00-0:40	이런 상황 — 큰 작업을 통째로 시켰다가 중간에 길을 잃는다. 한 방 지시의 한계.
0:40-1:30	컨텍스트 설계 — CLAUDE.md·memory로 무엇을 미리 깔아둘지. 흐름이라는 관점 잡기.
1:30-2:20	Plan 모드로 먼저 계획, TodoWrite로 작업 분해·추적. (5분 따라 하기: 작업 하나를 Plan→Todo로)
2:20-3:00	Slash 커맨드·Skills로 정형화 맛보기. 더 깊게: Subagents·워크트리 병렬로 가는 길(M18 예고).



강의 목표

임베디드-IoT를 한 번도 안 다뤄본 소프트웨어 엔지니어가 Claude Code를 페어 삼아 ESP32 보드에 불을 켜고 WiFi에 올린다. '낯선 영역도 AI와 함께라면 끝까지 간다'는 감각과, 말기고 검증하는 리듬을 손에 익힌다.



산출물

WiFi에 불어 동작하는 ESP32 보드



강의대상

AI 코딩 도구를 익숙한 일에만 쓰던 엔지니어, 신기술 PoC를 빠르게 돌려보고 싶은 실무자.



사전지식

터미널 기본, 아무 언어로든 코드를 읽고 고쳐본 경험, AI 코딩 도구 사용 경험. C/C++ 전자회로 지식 불필요.



실습환경

공통 환경 + ESP32 개발보드·브레드보드·LED·점퍼선 키트(과정 제공), USB 케이블, 아두이노/PlatformIO



강의목차 및 내용 (시간별)

시간	내용
0:00-0:30	이런 상황 — '임베디드는 내 영역이 아니라' 손도 못 댄다. 진입장벽이라는 한계.
0:30-1:20	Claude Code에게 일 시키기 + 개발환경 세팅(핸즈온). LED 점등으로 첫 성공 경험.
1:20-2:20	WiFi 연결(핸즈온). 보드를 네트워크에 올리고 상태 확인. (5분 따라 하기: LED를 코드로 깜빡이기)
2:20-3:00	막힐 때 AI에게 묻는 법, 에러를 함께 푸는 법. 더 깊게: 적외선·MQTT 원격제어로 가는 길(M19 예고).

M10

토큰·요금 메커니즘과 컨텍스트 절감 핵심(T01~T10)

초급 · 3시간 · 실무 · 출처 토큰 절약



강의 목표

토큰이 왜·어디서 녹는지 요금·토큰 메커니즘부터 잡는다. 비용의 큰 몫을 차지하는 컨텍스트를 다스리는 핵심 기법 10가지(T01~T10)를 익히고, 내 사용량을 실측해 새는 곳을 직접 찾는다.



산출물

내 사용량 베이스라인 + 컨텍스트 절감 체크리스트(T01~T10)



강의 대상

Claude Code 일상 사용자 중 비용·사용 제한이 신경 쓰이는 1년 차, 차, 개인 청구가 통장에 직결되는 1인 개발자.



사전 지식

Claude Code를 어느 정도 써본 경험(세션·파일 읽기·PR 정도), 터미널 기본, 회계 지식 불필요.



실습 환경

공통 환경 + 사용량 실측 도구(token-meter), 본인 사용 로그



강의 목차 및 내용 (시간별)

시간	내용
0:00-0:40	이런 상황 – 무심코 쓰다 사용 제한·청구서에 놀란다. 토큰이 어떻게 매겨지는지 모르는 한계.
0:40-1:20	요금·토큰 메커니즘, 입력·출력·캐시 토큰의 차이, 컨텍스트가 비용의 큰 몫인 이유.
1:20-2:20	컨텍스트 절감 핵심 T01~T10(핸즈온). 불필요한 파일 안 읽히기·세션 분할·요약 활용. (5분 따라 하기: 내 사용량 실측)
2:20-3:00	베이스라인 측정 → 절감 전후 비교. 더 깊게: 모델·습관·자동화 심화로 가는 길(M20 예고).

INTERMEDIATE · 중급

중급 모듈

팀에 들이고, 스스로 굴러가게

10



강의 목표

작업마다 세션을 분리해 맥락 오염을 막고, 무엇이 정답인지 가리는 권위 문서를 두어 일관성을 잡는다. CodeRabbit 자동 리뷰를 PR 흐름에 붙여, 혼자서도 PM·구현·리뷰 3역이 도는 운용 체계를 완성한다.



산출물

우리 팀용 운용 가이드 + 세션·권위 문서·리뷰 템플릿 세트



강의대상

M02를 들었거나 동등 수준인 엔지니어, AI 도구 운용을 팀에 정착 시켜야 하는 팀장, 개발 생산성·QA 담당자.



사전지식

M02(요구 주도·4게이트) 또는 동등 경험, GitHub PR 운용, Claude Code 실사용 경험.



실습환경

공통 환경 + GitHub(CodeRabbit 연동 가능 저장소), 회원관리 SaaS 예제



강의목차 및 내용 (시간별)

시간	내용
0:00-0:40	이런 상황 — 한 세션에 이것저것 시키다 맥락이 뒤엉킨다. 긴 세션이 품질을 떨어뜨리는 한계.
0:40-1:30	세션 분리 운용 — 작업별로 깨끗한 세션을 쓰는 규율. 권위 문서로 '무엇이 정답인지' 한곳에 고정.
1:30-2:20	CodeRabbit 자동 리뷰를 PR에 연결(핸즈온). AI 리뷰와 사람 리뷰의 역할 나누기. (5분 따라 하기: PR에 자동 리뷰 붙이기)
2:20-3:00	막힘·진척 관리, 3역 체제 점검. 더 깊게: 우리 팀 운용 가이드로 굳히기.



강의 목표

자율 봇에게 CLAUDE.md로 판단 기준을 가르쳐 엉뚱한 행동을 줄이고, 비용 상한과 타임아웃으로 폭주를 막는다. 여러 에이전트를 한 호스트에서 충돌 없이 공존 시키는 법까지 다뤄, 무인 자동화를 실운영 수준으로 끌어올린다.



산출물

판단 기준·비용 상한·타임아웃이 적용된 봇 + 다중 에이전트 구성도



강의대상

M03를 들었거나 자율 봇을 한 번 띄워본 엔지니어, 비용·리스크가 걱정되는 자동화 담당자.



사전지식

M03(첫 자율 봇) 또는 동등 경험, 리눅스 셸·cron, 짧은 스크립트 수정.



실습환경

공통 환경 + 상시 호스트(pi 5 또는 리눅스 서버), 비용 모니터링 가능 설정



강의목차 및 내용 (시간별)

시간	내용
0:00-0:40	이런 상황 — 무인 봇이 한밤중에 같은 작업을 반복하며 비용을 태운다. 통제 없는 자동화의 위험.
0:40-1:30	CLAUDE.md로 판단 기준 가르치기 — '이럴 땐 멈춰라'를 명시. 프롬프트 설계로 행동 범위 좁히기.
1:30-2:20	비용 상한·타임아웃 제어(핸즈온). 폭주를 코드로 막는다. (5분 따라 하기: 봇에 비용 상한 걸기)
2:20-3:00	다중 에이전트 공존 — 리소스 경합·충돌 다루기. 더 깊게: 한국 회사 적용 시 점검 항목.

M13

상류 공정 하네스 — 요건·도메인 분할·수용 기준·비기능 요건

중급 · 3시간 · 실무→팀 도입 · 출처 하네스



강의 목표

AI에게 구현을 맡기기 전, 상류 공정(요건·설계)을 다스려 결과 품질을 끌어올린다. Input을 관리하고 도메인을 분할하며, 수용 기준과 비기능 요건을 명시해 하류(구현)로 깔끔하게 넘기는 법을 익힌다.



산출물

한 기능에 대한 요건·수용 기준·비기능 요건 문서 세트



강의 대상

M04를 들었거나 동등 수준인 엔지니어, 요건·설계 산출물을 만드는 PL·기획·PM.



사전 지식

M04(하네스 입문) 또는 동등 경험, 자기 회사 개발 프로세스(요건→설계→구현→리뷰→배포)를 대략 안다.



실습 환경

공통 환경 + Claude Code, 예제 기능 시나리오(승인 라인 변경 등)



강의 목차 및 내용 (시간별)

시간	내용
0:00-0:40	이런 상황 — 요건이 모호하니 AI가 그럴듯하지만 빛나간 코드를 낸다. 입력이 흐리면 출력도 흐린 한계.
0:40-1:30	Input 관리·도메인 분할 — 큰 요구를 AI가 다루기 좋은 크기로 쪼개는 법.
1:30-2:20	수용 기준·비기능 요건 명시(핸즈온). '무엇이 완료인가'를 검증 가능하게 적기. (5분 따라 하기: 기능 1개의 수용 기준 작성)
2:20-3:00	하류 전달 — 구현 공정으로 넘기는 포맷. 더 깊게: 실장 공정 제어(M14)와 잇기.

M14

실장 공정 하네스 — 권한·커맨드·스킬·hooks·작은 차분 + OpenCode

중급 · 3.5시간 · 팀→에이전트 · 출처 하네스



강의 목표

구현 단계에서 AI를 통제하는 안전장치를 직접 짠다. 지시 파일·권한, 정형 커맨드, 스킬·서브에이전트, hooks/plugins로 가드레일을 두르고, 작은 차분(diff) 단위 리뷰로 품질을 지킨다. Claude Code와 오픈소스 OpenCode를 함께 굴러본다.



산출물

권한·커맨드·hooks가 적용된 실습 저장소 + 작은 차분 리뷰 1건



강의대상

M13 또는 동등 수준인 엔지니어, 품질·보안·감사 관점에서 AI 도입을 통제해야 하는 담당자.



사전지식

M04·M13 또는 동등 경험, 터미널·Git diff 읽기, JSON/설정 파일 다루기.



실습환경

공통 환경 + Claude Code + OpenCode(오픈소스, MIT), hooks 사용 가능 저장소



강의목차 및 내용 (시간별)

시간	내용
0:00-0:30	이런 상황 — AI가 한 번에 너무 많이 바뀌 리뷰가 불가능해진다. 큰 차분의 위험.
0:30-1:20	지시 파일·권한 설정 — 무엇을 건드리게 하고 무엇을 막을지. 정형 커맨드로 반복 작업 고정.
1:20-2:20	스킬·서브에이전트 + hooks/plugins로 가드레일(핸즈온). (5분 따라 하기: 커밋 전 hook으로 검사 걸기)
2:20-3:10	작은 차분 리뷰 — 변경을 작게 끊어 사람이 따라잡게. 통합 사례(승인 라인 변경) 실습.
3:10-3:30	OpenCode 핸즈온 — 같은 원리를 다른 도구로. 더 깊게: 표준 플로우·추적성·조직 도입.



강의 목표

skill-creator로 우리 팀 작업을 스킬로 만들어, 매번 반복하던 프롬프트를 팀이 공유하는 자산으로 바꾼다. 만든 스킬을 사내에 안전하게 배포·버전 관리·운영하는 절차까지 갖춰, 제각각인 사용법을 한 품질로 통일한다.



산출물

동작하는 사내 스킬 1개 + 배포 패키지



강의대상

M05를 들었거나 공식 스킬을 써본 엔지니어, AI 사용을 표준화·정착 착시시켜야 하는 팀장·플랫폼 엔지니어.



사전지식

M05(스킬 입문) 또는 동등 경험, 마크다운·YAML 읽기, Git 기본.



실습환경

공통 환경 + skill-creator, 사내 배포 채널(저장소·레지스트리) 모사 환경



강의목차 및 내용 (시간별)

시간	내용
0:00-0:40	이런 상황 — 좋은 작업 방식이 한 사람 머릿속에만 있다. 노하우가 안 퍼지는 한계.
0:40-1:40	skill-creator로 스킬 만들기(핸즈온). 우리 팀 작업 하나를 SKILL.md로 정리. (5분 따라 하기: 최소 스킬 1개 생성)
1:40-2:30	안전 배포 — 누가 어떻게 받게 할지, 버전·권한·보안 점검. 사내 보안 정책 위에 얹기.
2:30-3:00	운영 — 스킬을 어떻게 업데이트·폐기하나. 더 깊게: 조직 표준·거버넌스로 키우기.



강의 목표

M06에서 세운 지식 베이스를 자동으로 굴린다. RSS로 자료를 모으고, 데일리·위클리 리포트를 자동 생성하며, 서버에이전트로 자료를 자동 정리(ingest)하는 흐름까지 완성한다. 읽을수록 똑똑해지는 '제2의 뇌'를 실제로 돌린다.



산출물

RSS 수집 + 데일리/위클리가 자동으로 도는 지식 베이스



강의대상

M06를 들었거나 지식 베이스 뼈대가 있는 엔지니어, 지식 관리 자동화를 도입하려는 실무 리더.



사전지식

M06(LLM Wiki 입문) 또는 동등 경험, Python 읽기·수정, Git 기본.



실습환경

공통 환경 + 지식 저장소, RSS 소스 몇 개, cron/스케줄러



강의목차 및 내용 (시간별)

시간	내용
0:00-0:30	이런 상황 — 자료를 손으로 모으고 정리하다 결국 안 한다. 수동 수집의 한계.
0:30-1:20	RSS 자동수집(핸즈온). 원천 자료를 raw 층에 자동으로 쌓기.
1:20-2:20	데일리·위클리 리포트 자동화(핸즈온). 모인 자료를 요약·구조화해 wiki 층으로. (5분 따라 하기: 데일리 리포트 1회 자동 생성)
2:20-3:00	서브에이전트로 자동 ingest — 사람 손 없이 지식이 자라게. 더 깊게: 사내 보안·거버넌스 위에 얹기.



강의 목표

M07에서 길들인 Claude Code를 멈추지 않고 도는 운영 체계로 확장한다. n8n으로 외부 도구와 잇고, 스케줄로 24/7 실행하며, Hooks로 품질 게이트를 건다. 병렬 에이전트와 다른 모델로의 위임 체인까지 Opus 4.8 기준으로 굴러본다.



산출물

스케줄로 도는 자동화 1건 + Hooks 품질 게이트 + 병렬/위임 구성도



강의대상

M07를 들었거나 동등 수준인 엔지니어, Claude Code를 팀에 표준화 준비하려는 테크리드, 반복 업무 자동화 담당자.



사전지식

M07(CLAUDE.md·Skills·Memory) 또는 동등 경험, cron/스케줄러, JSON/YAML.



실습환경

공통 환경 + n8n(로컬 또는 사내), Hooks·스케줄 사용 가능 저장소



강의목차 및 내용 (시간별)

시간	내용
0:00-0:30	이런 상황 — 사람이 켜줘야만 자동화가 돈다. 반쪽짜리 자동화의 한계.
0:30-1:20	n8n 연계 + 24/7 스케줄 실행(핸즈온). 외부 도구와 잇고 시간 맞춰 돌리기.
1:20-2:10	Hooks x CLI 품질 게이트 — 자동 실행에도 품질을 강제. 권한·보안·한국 회사 맥락. (5분 따라 하기: 스케줄 작업에 품질 게이트)
2:10-3:00	병렬 에이전트(서브에이전트 vs Agent Teams), 뿔모델 위임 체인(Opus↔Sonnet).
3:00-3:30	바로 쓰는 운영 레시피, 비용·안정성 점검. 더 깊게: 워크플로 조립(M18)과 잇기.



강의 목표

M08에서 잡은 흐름 관점을 실제 자동 흐름으로 조립한다. Subagents로 위임하고, 워크트리로 병렬 개발하며, Scheduled agents로 장기 태스크를 돌린다. 모델 선택·프롬프트 캐시·배치·라우팅으로 비용까지 누른 뒤 나만의 워크플로를 설계한다.



산출물

내 업무에 끼워 넣은 워크플로 1종 + settings.json 레시피



강의대상

M08를 들었거나 동등 수준인 엔지니어, 팀 개발 흐름을 표준화·자동화하려는 테크리드.



사전지식

M08(워크플로 입문) 또는 동등 경험, git worktree, JSON(settings.json), cron 경험.



실습환경

공통 환경 + 워크트리 사용 가능 저장소, Scheduled agents 설정



강의목차 및 내용 (시간별)

시간	내용
0:00-0:30	이런 상황 — 흐름은 머릿속에 있는데 매번 손으로 굴린다. 자동화 안 된 흐름의 한계.
0:30-1:20	Subagents로 위임 + 워크트리 병렬 개발(한즈온). 여러 갈래를 동시에.
1:20-2:10	Hooks 자동화 + Scheduled agents·장기 태스크. (5분 따라 하기: 장기 작업을 스케줄에 올리기)
2:10-3:00	비용 최적화 — 모델 선택·프롬프트 캐시·배치·라우팅.
3:00-3:30	나만의 워크플로 설계 — 블록을 끼워 맞춰 흐름 완성. 더 깊게: settings.json·레시피 정리.

M19

IoT 완성 — 적외선 제어·MQTT 외부망 원격조작·타이머

중급 · 3시간 · 실무→에이전트 · 출처 에어컨 IoT



강의 목표

M09에서 WiFi에 올린 ESP32를 실제 쓸 수 있는 제품 수준으로 완성한다. 적외선으로 에어컨을 제어하고, MQTT로 외부망에서 원격 조작하며, 타이머로 예약 동작까지 붙인다. 맡기고 검증하는 워크플로로 낯선 영역을 끝까지 밀어붙인다.



산출물

폰으로 원격제어되는 에어컨 컨트롤러 + 데모 영상



강의대상

M09를 들었거나 ESP32를 WiFi에 올려본 엔지니어, IoT PoC를 완성 성품까지 끌고 가고 싶은 실무자.



사전지식

M09(IoT 입문) 또는 동등 경험, 코드 읽기·수정, API-HTTP-네트워크에 대한 막연한 감.



실습환경

공통 환경 + M09 키트 + 적외선 송수신 모듈, MQTT 브로커(클라우드/사내), 외부망 접근 경로



강의목차 및 내용 (시간별)

시간	내용
0:00-0:30	이런 상황 — WiFi엔 올렸는데 실제로 가전을 못 움직인다. 데모와 제품 사이의 간극.
0:30-1:20	적외선 제어(핸즈온). 리모컨 신호를 읽고 흉내 내 에어컨을 켜고 끄기.
1:20-2:20	MQTT 외부망 원격조작(핸즈온). 집 밖에서 명령 보내기. (5분 따라 하기: 폰에서 토픽으로 켜기)
2:20-3:00	타이머·예약 동작 붙이기, 데모 영상 촬영. 더 깊게: 보안(인증·암호화)과 실서비스 고려사항.

M20

비용 운용 심화 – 모델 습관(T11~T22) + 자동화·모니터링(T23~T31)

중급 · 3시간 · 팀 도입 · 출처 토큰 절약



강의 목표

M10의 컨텍스트 절감을 넘어, 모델 선택과 작업 습관(T11~T22)으로 비용을 더 누른다. 자동화와 모니터링(T23~T31)으로 절감을 일회성이 아닌 팀의 운용 체계로 굳혀, 1인당 월 비용을 눈에 띄게 줄인다.



산출물

팀 비용 모니터링 대시보드 초안 + 절감 자동화 체크리스트 (T11~T31)



강의대상

M10를 들었거나 동등 수준인 엔지니어, AI 도구 비용·라이선스를 관리하는 팀장·FinOps 담당자.



사전지식

M10(비용 입문) 또는 동등 경험, 터미널·사용량 로그 보기.



실습환경

공통 환경 + token-meter, 팀 사용 로그(또는 모사 데이터), 모니터링 설정



강의목차 및 내용 (시간별)

시간	내용
0:00-0:30	이런 상황 – 한 번 줄였다가 도로 늘어난다. 절감이 습관으로 안 굳는 한계.
0:30-1:20	모델 선택·작업 습관 T11~T22 – Opus와 Sonnet을 언제 가르나, 어떤 습관이 토큰을 아끼나.
1:20-2:20	자동화·모니터링 T23~T31(핸즈온). 새는 곳을 자동으로 잡고 알림 걸기. (5분 따라 하기: 사용량 알림 설정)
2:20-3:00	팀 베이스라인 → 목표 → 추적 체계 세우기. 더 깊게: 결재·예산 보고에 쓰는 지표 만들기.