Claude Code 기초 — 터미널부터 에이전트·스킬까지 실습 워크숍

> **형식:** 튜토리얼 / 사내(KIST) 워크숍 실습
> **주제:** Claude Code 처음 시작하는 사용자를 대상으로 터미널 기본 조작부터 `CLAUDE.md`, 슬래시 커맨드, 스킬, 에이전트, 플러그인까지 한 번에 따라 해보는 두 시간 분량의 실습형 강의
> **강의자:** 황민호 — 카카오 AI Studio 조직 산하 FDE 팀 (사내 조직을 돌며 AI로 빠르게 시스템을 구축하도록 돕고 AX 미션을 지원)
> **대상:** Claude Code를 일주일 미만 써보았거나 거의 처음 접하는 비개발자 / 연구자

목차

오프닝 — 왜 터미널과 친해져야 하는가

강의자는 카카오 AI Studio 산하 **FDE 팀**(팔란티어의 FDE와 비슷한 미션의 팀)으로 사내 조직을 돌며 AI 기반 시스템 구축과 AX(AI Transformation)를 돕고 있다고 자기소개를 합니다.
청중 대부분이 Claude Code를 일주일 이상 써본 경험이 거의 없어, 강의 톤은 천천히 따라올 수 있게 진행됩니다.

> Claude Code가 개발자가 아닌 분들에게는 굉장히 낯선 환경이지만, **터미널과 친해지는 것**을 오늘의 첫 목표로 잡으면 됩니다. 이 고비만 넘으면 그 뒤에는 **하네스(harness)** 라는 또 다른 세계가 기다리고 있습니다.

하네스 데모 — 책 한 권을 한 시간에

본격적인 실습 전에, 강의자가 준비한 짧은 영상으로 **하네스(harness)** 의 위력을 보여줍니다.

- 단 몇 줄의 프롬프트로 **200~300페이지가 넘는 책**을 만들어내는 시연.
- 다섯 개의 에이전트가 자동으로 생성되어 서로 협업.
- 리서치 에이전트가 책의 소스 자료를 수집.
- 각 에이전트가 **병렬로 집필**, 완료되면 보고.
- 책에 들어갈 **삽화 이미지**까지 생성하여 정확한 위치에 삽입.
- 결과물은 **마크다운**으로 작성된 뒤 **PDF**로 변환.
- 추가 요청 한 줄로 PDF 책을 페이지 넘기는 형태로 보여주는 **웹사이트**까지 자동 구축.
- **총 소요 시간 약 1시간.**

> 오늘 Claude Code를 배우러 오셨으니까, Claude Code 입문서를 사도 되지만 — 하네스를 익히면 **나만의 책**을 직접 만들 수도 있습니다.

이 단계는 강의의 동기부여 데모이며, 본격 학습은 다음 주 이후 세션에서 다룰 예정이라고 안내합니다.

Claude Code의 위치 — 웹·데스크탑과의 차이

환경	동작 위치	자유도
웹 (Claude, ChatGPT, Gemini)	브라우저	매우 제약됨
Claude Desktop	로컬(앱)	제약 있음
Claude Code (터미널)	**로컬, 내 폴더에 직접 액세스**	**자유로움**

> Claude Code는 **내 로컬 환경에서 돌아가는 에이전트**라고 생각하시면 됩니다. 매 프로젝트마다 폴더 하나를 만들고, 그 폴더 안에서 작업합니다.

웹/데스크탑에서는 파일을 일일이 업로드해야 하지만, Claude Code는 **현재 폴더의 파일을 스스로 탐색·읽기·분석**합니다. 강의자는 KIST 회의록 3개가 들어있는 폴더에서 `claude` 를 실행한 뒤 "이 회의록 3개를 합쳐서 한 페이지의 보고서로 요약해줘 한국어로"라고만 입력해 결과를 즉시 받아내는 시연을 보여줍니다.

터미널 기본기 — pwd, cd, mkdir

청중이 PowerShell/터미널 환경에 익숙하지 않으므로, 최소한의 명령어부터 짚고 넘어갑니다.

```
pwd                 # 현재 위치한 폴더 확인 (Print Working Directory)
cd Downloads        # 폴더 이동 (Change Directory)
mkdir kist-test     # 새 폴더 생성 (Make Directory)
cd kist-test        # 새로 만든 폴더로 진입
claude              # 그 위치에서 Claude Code 실행
```

> 우리의 목표는 **폴더를 만들고, 그 위치로 가서 *claude* 를 실행하는 것까지**입니다. 그 이후로는 AI가 모든 것을 다 해줍니다.

윈도우 사용자는 **PowerShell**, 맥 사용자는 기본 터미널을 사용하면 됩니다.

첫 실행과 폴더 단위 작업

`claude` 를 실행한 위치의 **폴더 전체가 Claude Code의 작업 영역**이 됩니다. 즉, 그 폴더 하위의 파일에 대해 읽기·쓰기·생성 권한을 가지게 되는 것이죠.
작업 중 유용한 단축 동작:

동작	설명
`ESC`	진행 중인 작업을 즉시 취소
작업 중 추가 타이핑	현재 작업이 끝난 뒤 다음 작업으로 큐잉됨
`@` (골뱅이)	파일/폴더를 참조 컨텍스트로 첨부. 입력 후 `Tab` 으로 선택 확정

CLAUDE md — 항상 따라붙는 지침 파일

`CLAUDE.md` 는 **모든 명령마다 자동으로 포함되는 지침 파일**입니다. 따라서 너무 길지 않게, 그러나 핵심 규칙을 **명료하게** 담아야 합니다.

CLAUDE.md의 4개 레이어

레이어	적용 범위
Managed (회사 차원)	회사의 모든 멤버에게 적용
유저 글로벌	내 로컬 어떤 프로젝트에서든 적용
프로젝트	해당 프로젝트(폴더) 안에서만 적용
로컬 (Local)	동료와 공유하지 않는 내 PC만의 추가 지침

만들기 — 프롬프트 한 줄

Claude Code 안에서 직접 만들도록 시킵니다.

```
이 폴더에 CLAUDE.md 파일을 만들어 주세요.
지침은 다음과 같습니다.
- 답변은 항상 한글로 작성한다
- 새로 만드는 파일 이름에는 항상 _new를 붙인다
```

이렇게 만들면 그 이후의 모든 작업이 이 지침을 따라 동작합니다. 한글로 답변하고, 생성되는 파일에는 자동으로 `_new` 접미사가 붙습니다.

> *`CLAUDE.md`* 는 AI에게 시켜서 만들 수도 있지만 **직접 손으로 수정해도 됩니다.** 원본 파일을 수정 금지로 둔다거나, 파이썬 코드 작성 규칙을 강제한다거나, 출력 문체를 지정하는 식으로 활용할 수 있습니다.

실습 1 — 미스토스 모델 리서치 → 웹사이트

청중이 직접 결과를 만들어 보는 첫 실습입니다.

1단계 — 웹 리서치

```
미스토스(Mistos) 모델을 조사하고 관련된 내용을 파일로 저장합니다.
```

Claude Code에는 **WebSearch / WebFetch** 같은 도구가 기본 탑재되어 있어, 별도 설정 없이도 인터넷에서 자료를 모으고 출처 링크까지 포함한 마크다운 파일로 정리해 줍니다. `CLAUDE.md` 지침대로 파일명은 `mistos-research_new.md` 같은 형태로 저장됩니다.

2단계 — 정적 웹사이트 생성

```
조사된 내용을 토대로 세련되고 모던한 웹사이트를 만들어 주세요.
배경에는 애니메이션 효과가 극대화된 요소를 추가해 주세요.
```

3단계 — 테마/브랜드 톤 변경

추가 프롬프트로 즉석에서 디자인을 바꿉니다.

```
화이트 컬러의 Apple 사이트 스타일로 수정해 주세요.
```

> 요즘에 유행하는 팁은 **브랜드 이름을 넣는 것**입니다. “Apple 사이트 스타일”, “Stripe 스타일” 같이 구체적인 브랜드를 지정하면 디자인 결과물이 훨씬 좋아집니다.

디자인 시스템 추출과 스킬 만들기

웹 서비스 개발 흐름은 보통 디자이너가 먼저 시스템을 만들고 개발자가 그것을 받아서 구현하는 방식이지만, AI에서는 **반대 방향**으로도 가능합니다.

디자인 시스템 추출

```
이 사이트의 디자인 요소를 추출해서 별도의 디자인 시스템을 만들어 주세요.
```

추출되는 요소들:

- 컬러 팔레트
- 타이포그래피
- 컴포넌트 (버튼 스타일, 카드 등)
- 배치/레이아웃 규칙
- 애니메이션 효과

Skill로 만들기

> Claude Code의 **Skill**은 한 번 이상 반복해서 할 작업들을 묶어 저장해 두면, 다음에 어느 프로젝트에서든 재활용할 수 있게 만들어 주는 구성요소입니다.

명령은 다음과 같이 줍니다.

```
이 디자인 시스템을 my-design 이라는 스킬로 만들어 주세요.
```

스킬 생성 중 권한 요청 다이얼로그가 뜨면:

옵션	의미
1	이번 한 번만 허용
2	같은 작업은 더 이상 묻지 않음

스킬이 저장되는 위치

스코프	위치
프로젝트 한정	`<프로젝트>/.claude/skills/`
사용자 전역	`~/.claude/skills/` (윈도우는 `C:\Users\<사용자>\.claude\skills\`)

이 위치 아래에 `my-design` 폴더가 생기고, 그 안에 Apple 톤 화이트 디자인 시스템의 규칙들이 정리되어 등록됩니다.

실습 2 — Hugging Face Daily Papers 정리

연구자 청중을 위한 두 번째 실습. **Hugging Face의 Daily Papers** 리스트를 자동 정리하는 흐름입니다.

```
https://huggingface.co/papers
이 사이트의 오늘자 논문들을 정리해 줘.
```

이때 Claude Code는:

1. URL을 **WebFetch**로 가져옵니다.
2. 리스트 페이지에는 초록만 있으므로, **각 논문 상세 페이지**를 개별적으로 다시 패치합니다.
3. 결과물을 분류별로 정리한 마크다운 파일로 저장합니다.

이 시점 결과: 그날 올라온 논문 약 16편을 분류·요약한 파일이 폴더에 떨어집니다.

스킬 재활용 — 같은 디자인으로 두 번째 사이트

논문 정리 파일을 컨텐츠로, 앞에서 만든 `my-design` 스킬을 디자인 가이드로 사용합니다.

```
@papers-research_new.md
이 논문 내용을 토대로 my-design 스킬을 사용해서
동일한 디자인으로 논문을 소개하는 웹사이트를 만들어 주세요.
```

`@` (골뱅이) 기호로 파일을 참조 컨텍스트에 명시적으로 붙입니다. 결과적으로 미스토스 사이트와 **시각적으로 일관된** 두 번째 사이트가 생성됩니다.

> **`@` 는 그 파일/폴더를 "참고해서" 작업하라는 의미**입니다. 파일도, 폴더도 지정할 수 있습니다.

GPT-Image-2 스킬과 인포그래픽

강의자는 별도 스킬 압축 파일에 **`gpt-image-2`** 스킬을 함께 배포했다고 안내합니다.

- 최근 출시된 OpenAI의 **GPT Image 2** 모델을 활용해 이미지를 생성하는 스킬.
- 만드는 방법은 **공식 개발자 문서를 복사 → "이걸 토대로 GPT Image 2 스킬로 만들어 줘"** 라고 시키는 것뿐.
- 다음 프롬프트로 강의 흐름을 한 장의 인포그래픽으로 시각화한 예시를 보여줍니다.

```
지금까지 했던 내용들을 GPT Image 2로 인포그래픽으로 그려 줘.
가로 비율로 만들어 줘.
```

> 좋은 스킬을 한 번 잘 만들어 두면 **계속 재활용**할 수 있고, 어려운 작업도 훨씬 쉽게 처리할 수 있습니다. (스킬에 대한 더 깊은 내용은 다다음 주 두 번째 세션에서 다룰 예정.)

플러그인과 마켓플레이스 — Document Skills 설치

PDF 논문을 분석하기 위해 **공식 플러그인**을 설치합니다.

UI에서 설치

Claude Code 안에서 슬래시 명령:

```
/plugin
```

UI를 따라 들어가서 **Marketplace → Add Marketplace** 를 선택하고 다음을 입력합니다.

```
anthropic/skills
```

추가된 마켓플레이스 안의 플러그인 중 **Document Skills**를 스페이스로 선택하고 엔터로 설치합니다. 이 플러그인은 **PDF, PPTX, 엑셀** 등 사내에서 자주 쓰는 문서 포맷을 Claude Code가 직접 읽을 수 있도록 해 줍니다.

UI 설치가 막힐 때 — CLI로 설치

GitHub 접근 등에서 에러가 나는 경우, 공식 GitHub 리포 (`anthropic/skills`)의 README에 있는 두 줄을 그대로 복사해 Claude Code에 붙여넣어 실행합니다.

```
/plugin marketplace add anthropic/skills
/plugin install document-skills
```

설치 후에는 깔끔한 반영을 위해 `/exit` 로 Claude Code를 한 번 종료한 뒤 다시 실행합니다. 정상 적용되었는지는 다음으로 확인합니다.

```
/pdf
/pptx
```

해당 슬래시 커맨드가 자동완성에 나타나면 설치 성공입니다.

> 한국 사용자에게 자주 필요한 **HWP**는 Anthropic 공식에는 없지만, 강의자가 별도로 만들어 둔 HWP 스킬을 압축 파일에 포함해 함께 배포하고 있다고 안내합니다. 압축을 풀어 스킬 폴더에 복사하기만 하면 됩니다.

서브에이전트 만들기 — 논문 분석 에이전트

PDF 다운로드 → 분석을 자동화하는 **서브에이전트**를 만들어 봅니다.

PDF 다운로드

Hugging Face 논문 상세 URL을 그대로 붙여 넣고 시킵니다.

```
이 URL의 관련 PDF 파일 다운로드.
```

Claude Code는 arXiv 링크를 찾아 PDF를 현재 폴더에 받아 둡니다 (예: 36페이지짜리 논문 1편).

에이전트 정의 만들기

스킬과 마찬가지로 **AI에게 만들어 달라고 시키면 됩니다.** 이 프로젝트 한정으로 만들 것이므로 `.claude/` 하위 경로를 명시합니다.

```
.claude 하위에 논문 PDF를 분석하는 에이전트를 추가해 주세요.
이름은 paper-analyzer 로 합니다.
```

생성되는 에이전트 파일에는 다음 메타정보가 자동으로 채워집니다.

항목	내용
name	`paper-analyzer`
description	에이전트가 하는 일 한 줄 요약
tools	파일 읽기 / Bash 실행 등 사용 가능 도구
model	어떤 모델을 사용할지 (가장 좋은 Opus 모델로 설정됨)
system prompt	"학술 논문 분석 전문가" 페르소나, 5분 안에 핵심 파악 가능한 리포트 생성을 목적으로 한 원칙·입력 형식 등

에이전트 로딩과 사용

방금 만든 에이전트는 **동적으로 즉시 로딩되지 않습니다.** Claude Code를 한 번 종료했다 다시 실행해야 인식됩니다.

```
/exit
claude
/agents
```

`/agents`로 현재 사용 가능한 에이전트 목록을 확인할 수 있습니다. 등록되었으면 다음과 같이 호출합니다.

```
@my-paper.pdf
paper-analyzer 에이전트를 이용해서 논문을 분석합니다.
```

에이전트 고도화

생성 시점의 기본 정의는 일반론에 가깝습니다. 본인의 도메인 지식으로 추가 지침을 넣으면 깊이가 달라집니다.

- 초록 위주로 분석
- **한계점(limitations)** 위주 분석
- 본문 주장에 대한 **검증·반박** 위주

병렬 에이전트와 컨텍스트 분리

웹 챗봇과 가장 차별화되는 부분은 **병렬 처리**와 **컨텍스트 분리**입니다.

```
나머지 11개 논문도 paper-analyzer 에이전트로 분석해서 파일로 저장해 주세요.
6개의 에이전트로 병렬로 진행해 주세요.
```

이렇게 하면 6개의 에이전트가 동시에 스폰되어 12개 논문(이미 분석된 1편 + 11편)을 나눠 처리합니다.

> 이런 병렬화는 **웹사이트(Claude.ai)에서는 되지 않습니다.** Claude Code를 썼을 때 비로소 우리가 진짜 생각하는 대로 AI가 많은 것을 해줄 수 있게 됩니다.

컨텍스트 절약 효과

각 서브에이전트는 **자체 컨텍스트 윈도우**를 가집니다. 메인 세션은 결과 요약만 한 줄 받기 때문에, 메인의 컨텍스트를 거의 소모하지 않으면서도 큰 작업을 처리할 수 있습니다. 이게 컨텍스트 관리의 핵심 패턴 중 하나입니다.

결과를 사이트에 다시 연결

```
분석된 논문 내용을 토대로 논문 소개 사이트에 상세 페이지를 만들어 연결합니다.
```

이렇게 하면 앞서 만든 논문 소개 사이트의 각 카드에 **상세 페이지 열기** 링크가 추가되고, AI가 분석한 내용이 동일한 디자인 시스템으로 렌더링되어 표시됩니다. 매일 자동화하면 **데일리 페이퍼 소개 사이트**가 하루도 안 걸려 만들어집니다.

사용량 관리 — usage 와 주간 리밋

본격적으로 쓰기 시작하면 곧바로 부딪히는 벽이 **사용량 한도**입니다.

```
/usage
```

이 명령으로 현재 세션 사용량과 이번 주 사용량을 확인할 수 있습니다.

단위	설명
세션 한도	최초 실행 시점부터 **5시간** 윈도우. 그 안에서 100%를 다 쓰면 윈도우 종료까지 사용 불가
주간 리밋	개발자들이 가장 고통스러워하는 부분. 차버리면 다음 주까지 대기

> 강의 도중 청중 일부의 **Pro 요금제 토큰이 모두 소진**되는 상황이 실제로 발생했습니다. 강의자는 "주변에서 Pro로 시작했다가 결국 다 200달러 플랜으로 가더라"고 언급합니다. **토큰을 알뜰하게 쓰는 법**이 핵심 역량이 되는 시대.

컨텍스트 윈도우와 compact

```
/context
```

명령을 치면 현재 세션이 어떤 내용으로 컨텍스트를 채우고 있는지 시각화해서 보여줍니다.

> 현재 Claude Code가 제공하는 컨텍스트 윈도우는 **100만 토큰**. 이걸 어떻게 잘 쓰느냐가 응답 품질을 좌우합니다.

채워지는 영역(예시):

- 시스템 프롬프트
- Skills
- 사용자/AI 메시지(보라색 영역) — 작업할수록 점점 채워짐

컨텍스트가 가득 차면 **compact**가 자동 발생해 압축됩니다. 그러나 compact 시점에 손실(lost data)이 발생하므로, AI 성능이 약간 떨어질 수 있습니다.

```
/compact
```

> Anthropic의 권장은 **꽉 차기를 기다리지 말고 적정 시점에 수동으로 `/compact`** 를 실행하는 것입니다. 더 좋은 전략은 큰 작업을 서브에이전트에 분리해서 메인 컨텍스트가 천천히 차도록 운용하는 것입니다.

세션 이어가기 — continue, resume, clear

PC를 껐다 켰거나 실수로 종료한 경우 등, 이전 맥락을 이어가는 방법.

명령	동작
`claude --continue`	가장 최근 세션을 그대로 이어서 시작
`claude --resume`	과거 세션 목록을 띄워 선택해서 이어가기
`/clear`	종료하지 않고도 컨텍스트를 초기화하여 새 세션처럼 시작
`/exit`	Claude Code 종료 (맥락 끊김)

세션 이력은 다음 위치에 자동 저장됩니다.

```
~/.claude/projects/<프로젝트별 폴더>/<세션 폴더>
```

캐싱 주의

요청 사이의 **prompt caching**은 유효시간이 있어, 너무 오래 뒤에 resume하면 캐시 효과 없이 처음부터 전체를 다시 실어 보내게 됩니다. 비용·지연 면에서 손해이므로, 끊긴 지 한참 된 작업은 차라리 **새 세션을 여는 편이 유리**할 때도 있습니다.

> 다른 PC로 옮겨서 같은 세션을 이어가려면 `~/.claude/projects/` 아래 파일을 직접 옮겨야 합니다. 새로 설치된 PC에는 과거 이력이 남아있지 않습니다 — 모두 **로컬에만 저장**되기 때문입니다.

실용 스킬 — Worklog와 Manpower

강의자가 자작·배포한 두 가지 스킬을 소개합니다.

Worklog 스킬

내가 최근에 했던 작업을 **주간보고 형태로 자동 정리**해 줍니다.

```
worklog 스킬로 최근 일주일 동안 했던 작업을 정리해 줘.
```

- `~/.claude/projects/` 아래의 세션 로그를 분석.
- 어떤 프로젝트에서 어떤 작업을 했는지 깔끔하게 보고서 형태로 출력.
- 개발자처럼 거의 모든 일을 Claude Code로 처리하기 시작하면, 본인의 활동을 관리하는 데 매우 편리.

Manpower 스킬

내가 **AI를 얼마나 잘 쓰고 있는지**를 분석해 주는 메타 스킬.

- 팔각형(레이더) 차트로 시각화.

- 어느 영역이 부족한지, 매주 성장하고 있는지를 추적.
- 조직 차원에서 직원들의 AX 정도를 측정하는 도구로도 활용 가능.

Q&A

세션 말미의 청중 질문 정리.

Antigravity와의 관계

> Antigravity는 IDE 기반이라 UI에서 제어할 수 있지만, Claude Code는 더 스마트하게 동작합니다.
> Antigravity 안에서 별도 터미널을 띄워 Claude Code를 실행하는 것도 가능 — 본인이 편한 방식으로 쓰면 됩니다.

Comet 등 다른 옵션

> Comet은 출시가 늦어 아직 기능이 부족하고 불안정한 부분이 있지만, 최근 1.0이 런칭되어 터미널이 어렵다면 시도해볼 만합니다.

다른 문서 포맷 (PPTX, 엑셀, HWP 등)

> Document Skills 플러그인 하나로 **PDF / PPTX / 엑셀**이 모두 커버됩니다. **HWP**는 Anthropic 공식에는 없지만 강의자가 만든 스킬을 배포 압축 파일에 포함했습니다. 텍스트 변환이 가능한 라이브러리가 있는 포맷은 모두 스킬로 감싸 쓸 수 있고, **순수 바이너리 포맷**은 어려울 수 있습니다.

세션 키워드 검색

> 현재 기본 명령으로는 안 되지만, `~/.claude/projects/` 아래 파일들이 모두 텍스트로 저장되어 있으므로 Claude Code 자체에 그 폴더를 분석시켜 키워드가 들어있는 세션 ID를 찾아내라고 시키는 방식이 가능합니다.

핸드아웃 자료

강의자는 사내 게시판에 다음 자료들을 함께 공유했다고 안내합니다.

- **압축 스킬 묶음** — `gpt-image-2`, `worklog`, `manpower`, `hwp` 등
- **핸드아웃 PDF** — 자주 쓰는 슬래시 커맨드와 명령어 모음

핵심 정리

1. **터미널 + 폴더 단위 사고**가 Claude Code 사용의 출발점입니다. `pwd`, `cd`, `mkdir`, `claude` 네 가지만 익히면 그 다음은 AI가 다 합니다.
2. `CLAUDE.md` 는 모든 명령에 자동 첨부되는 지침. 한글 답변·파일명 규칙·코드 스타일 등 **반복하기 싫은 규칙**을 여기에 명료하게 적어 둡니다. Managed / 글로벌 / 프로젝트 / 로컬 4개 레이어로 운용합니다.

3. `@` **참조**로 파일·폴더를 컨텍스트에 명시적으로 붙이고, `/agents`, `/plugin`, `/usage`, `/context`, `/compact`, `/clear`, `/exit` 등의 슬래시 커맨드로 세션을 통제합니다.
4. **Skill** = 반복 작업의 재활용 단위. 한 번 만들어 두면 어느 프로젝트에서든 동일 결과를 보장합니다. 디자인 시스템·이미지 생성·문서 분석 등에 강력합니다.
5. **Plugin / Marketplace** 로 공식·서드파티 스킬 묶음을 한 번에 설치합니다. 공식 마켓플레이스는 `anthropic/skills`, 핵심 플러그인은 **Document Skills**.
6. **서브에이전트(subagent)** 는 자체 컨텍스트 윈도우를 가진 **전용 작업자**. 페르소나·도구·모델을 지정해 만들고, **병렬 스폰**으로 큰 작업을 빠르게 처리할 수 있습니다 (웹 챗봇 대비 가장 큰 차별점).
7. **컨텍스트 윈도우 100만 토큰**은 무한이 아닙니다. 메인을 가볍게 두고 무거운 작업은 에이전트로 분리, 적정 시점 `/compact`, 끊긴 작업은 `--continue` / `--resume`으로 이어 갑니다.
8. **사용량(Usage)** 이 결국 가장 큰 제약입니다. 5시간 세션 한도와 주간 리밋을 의식하며, **Pro → Max($200)** 로 자연스럽게 옮겨가게 됩니다. 토큰은 자고 일어나면 다시 살아납니다.
9. **하네스(harness)** 는 다음 단계의 학습 주제. 여러 에이전트가 협업해 책 한 권·웹사이트·이미지·PDF까지 한 번에 산출하는 구조이며, 이번 강의는 그 전 단계의 기초입니다.