AI In-depth

금융권을 위한 AI Agent 활용 기초

상상인증권 리서치센터

김경태 | 글로벌전략/디지털자산 02-3779-3427, kt.kim@sangsanginib.com

상상인증권

CONTENTS

Executive Summary

- Prompt Engineering의 종언과 Context Engineering의 시대 도래: 현재 AI의 사용법은 Prompt Engineering이 아니라 종합적인 작업 계획표: 과거 업무 기록 및 장기 기억, 그리고 현재의 목표, 그리고 데이터의 출처까지 고려. AI 모델이 접근하는 Context 전체를 관리하는 Context Engineering의 이해도가 결과값의 품질을 결정하며, 인간의 역할과 개입은 과거 업무 기록, 그리고 검증/수정 단계에 집중
- 모델 선택은 벤치마크 순위가 아니라 과제 완주율과 도구 호출 안정성, 오류 교정에 투입되는 시간까지 합산한 총비용을 기준으로 판단. 동일 성공률 조건에서도 경량 모델의 성공 건당 비용이 최상위 모델의 7배에 이르는 역전이 확인됨.
- 실행 환경(하네스)은 복수 모델 구동과 메신저 연동이 필요하면 오픈소스 계열(Hermes), 단일 모델 기반의 사무실 검토형 업무가 중심이면 설치와 운용이 간편한 자체 하네스(Claude Code·Codex)가 적합
- AI 대화 내역은 축적되나, 내부 추론과 교정 내용은 컨텍스트 윈도우와 함께 소멸하므로, 작업 규칙은 스킬(SKILL.md)에 파일 형태로 축적해야 다음 대화부터 자동 반영. 스킬은 호출 조건을 기술한 description이 실행 여부를 결정하는 구조이므로, 실제 사용하는 지시 문구를 그대로 기재하는 것이 등록의 관건
- AI의 사용 문법은 무서운 속도로 발전하고 있으나, AI 업무 설계 -> 메모리 구축 -> 자동화의 골격은 유지. AI 활용 역량과 경험의 격차는 시간이 갈수록 누적되는 구조이며, AI 사용법의 편의성은 빠른 속도로 개선되고 있으나, AI 워크플로우 고도화는 지식보다 경험, 노하우의 영역에 더 가깝다고 해석

지시문 작성 요령

- 지시 블록의 필수 요소 — ①무엇을(작업) ②어떤 입력에서(파일·경로) ③어떤 형식으로(표, 엑셀, html) 명시
- 필수 요소가 빠지면 모델이 추측으로 공백을 채우는 구조 — "리포트 분석해줘"(범위·산출·저장 등 세부 요소 부재)
- 표준 루프는 “구체적 지시 → 파일 해석·코드 생성 → 숫자 검증 → 반복 작업은 코드화 → 점검은 사람” 순서로 고정
- 지시문의 범위·형식·저장·제약을 채우는 설계가 성과를 좌우

그림 1. 구체적이지 못한 지시문 예시

프롬프트 >

“리포트 요약, 분석해줘”

지시에 4가지가 빠져 있음

- 범위 — 무엇을, 얼마나 분석할지 설명이 없음
- 형식 — 표·문장·md 중 무엇인지 설명이 없음
- 저장 — 결과를 어디에, 어떤 형태로 남길지 모름
- 제약 — 추측 금지 등 규칙 없음

→ 빈칸을 모델이 추측으로 채움

→ 실행할 때마다 결과가 달라져 재현 불가

자료: 상상인증권 리서치센터

그림 2. 좋은 지시문 예시

프롬프트 >

“reports.json에서 영업이익, 목표가 변경만 표로, md로 작업 폴더 루트에 저장. 원문 근거만 활용”

4가지를 한 문장에 모두 지정

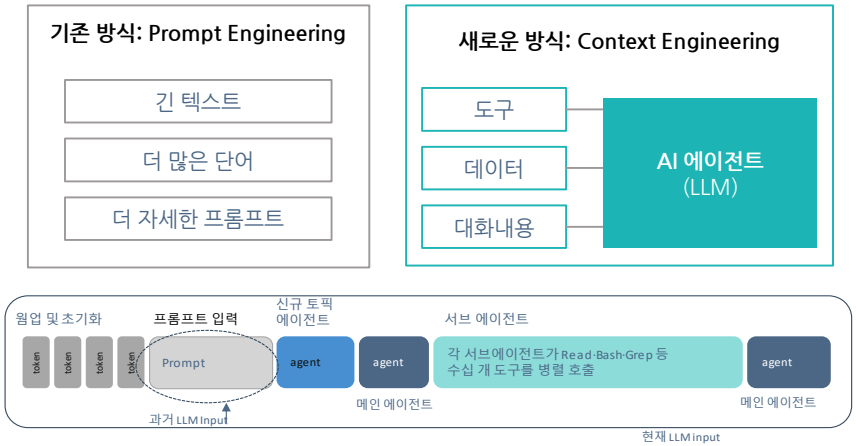
- 범위 — 목표가 변경 항목만
- 형식 — 표 형태 · 마크다운(md)
- 저장 — 작업 폴더 루트
- 제약 — 추측 배제 · 원문 근거만

→ 누가·언제 실행해도 동일한 산출

자료: 상상인증권 리서치센터

Focus Chart 1

기존 챗봇 시대의 AI vs 현재 AI에 들어가는 입력값 차이



자료: Google Brain, 상상인증권 리서치센터

모델 지능이 낮은 만큼 토큰도 많이 소모한다

지표	Fable 5 (low)	Sonnet 5 (high)	GLM-5.2	배율 (GLM/ Fable)
성공율	80%	80%	80%	동일
총비용	$2.27	$8.43	$15.82	7.0x
성공 건당 비용	$0.28	$1.05	$1.98	7.0x
총 토큰 소모량	0.14M	1.75M	10.82M	77.0x
총 소요 시간	14분	59분	124분	8.9x

자료: Vulcanbench, 상상인증권 리서치센터

Prompt → Context Engineering

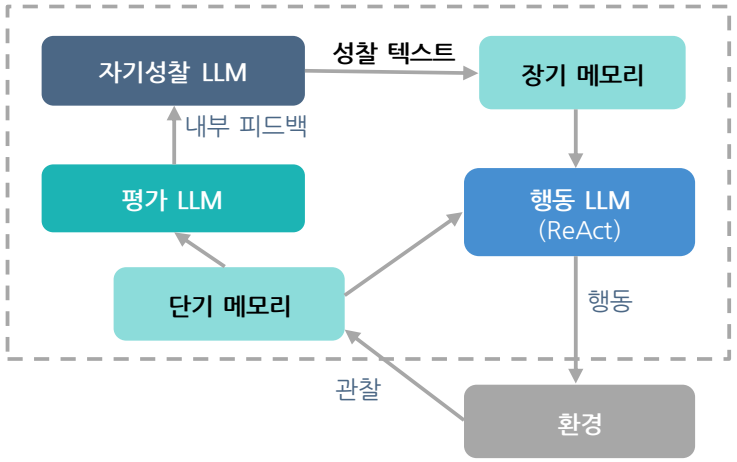
- 과거: 프롬프트를 다듬어 한 번에 정답 유도 & 비슷한 결과물을 재생산 → 사용자 노하우 공유에 의존
- 그러나 과거의 역할극·마법의 문장/레시피('단계별로 생각해' 등)·보상·협박 문구는 이제 효과 미미
- Context Engineering의 시대 → AI 모델의 ①작업 내역(목적·입력·출력·메모리·금지 사항 등) ②완료 조건(Goal) ③참고 데이터베이스 ④검증·검산을 조정하는 시대

AI 모델 구독 전 숙지할 내용

- AI 순위 인용을 위한 벤치마크는 짧은 응답·제한적 과제 위주이며, 공개된 벤치마크는 과적합(정답 암기)에 취약
- 모델 성능이 낮은 만큼 소모되는 토큰량과 시간이 많아 일부 복잡한 작업은 "가성비"라고 단정하기 어려움
- AI 실무작업 선정 기준: ①다단계 작업 성공률/완주율 ②도구 호출 안정성과 실패 후 재개 신뢰도 ③후반까지 지시·제약 유지, 망각 문제(Catastrophic Forgetting) 최소화 ④오류/문맥오염 수정 포함 총비용

Focus Chart 2

현 추론형 LLM의 성찰·평가 루프 이후 최종 답변 도출



자료: Anthropic, 상상인증권 리서치센터

Hermes 사용 도식화: AI 모델 – Hermes 기기 – 메신저 연결

자료: 상상인증권 리서치센터

장기기억은 따로 축적해야 한다

- 에이전트는 행동(ReAct)·평가·자기성찰 LLM이 내부 피드백과 성찰 텍스트를 주고받는 다단 루프 구조
- KV캐시와 컨텍스트 창은 단기 메모리에 해당하는 휘발 영역 —대화 종료와 함께 소멸. 성찰/추론 텍스트 저장 불가
- 사실상 작업/추론 성능의 핵심인 성찰 텍스트·전체 대화는 장기 메모리로 이관해야 보존
- 대화 기록을 벡터 DB·파일로 축적해 다음 대화가 검색·회상하게 하는 것이 장기 기억의 구현

Hermes Agent로 개인 AI 비서 구축

- Hermes는 오픈소스 에이전트 하네스로, 특정 모델에 구애받지 않고 사용 가능
- 메신저(텔레그램·디스코드·슬랙)·PC 연결을 지원 — 자체 Harness가 없는 오픈소스 모델(GLM, Deepseek, Kimi, Exaone) 활용에 필수
- 사용할수록 발전: 채팅 로그에서 스킬을 생성·최적화하고, 과거 대화를 주기적으로 검색·요약해 장기 기억처럼 활용

마크다운(CLAUDE.md), 스킬(SKILL.md) – 반복 작업 정형화

스킬, Description, 절차 파일 예시

Exampleskill.md

Name 스킬-이름 (영문 소문자·하이픈, 폴더명과 동일하게)

Description ①사용자가 실제로 입력할 문구를 따옴표로 나열 ("○○ 정리해줘", "○○ 만들어라"). ②입력이 무엇인지 — 어떤 파일·경로·데이터를 받아 작동하는지. ③산출물이 무엇인지 — 어떤 형식(.xlsx/.docx/.txt)을 어디에 저장하는지. ※ 이 구간만 상시 로드됨 — 호출 여부가 여기서 결정되므로 막연하게 쓰면 스킬이 불리지 않음

스킬 제목 (한 줄)

절차 ← ②몸통: 호출된 때만 로드되므로 길어도 부담 없음

1. 입력을 어디서 가져오는지 — 파일 경로, API, 캘린더 등 출처를 구체적으로
2. 어떤 순서로 처리하는지 — 단계마다 사용할 도구·스크립트·옵션 명시
3. 산출물을 어떤 형식으로 만드는지 — 서식·인코딩·파일명 규칙
4. 어디에 저장하는지 — 정확한 경로 (서브폴더 생성 여부까지)

주의사항

- 반복 실수했던 함정, 계정 차단·인코딩 깨짐 같은 리스크와 대응법
- 참조할 정본 문서가 있으면 경로 명시 ("작업 전 ○○.md를 먼저 Read")

완료 조건 ← ③꼬리: 기계적으로 확인 가능한 검증 항목

- ○ 행 수·건수가 원본과 일치하는가
- ○ 부호·단위·기준일이 맞는가
- ○ 파일이 지정 경로에 존재하는가 ※ 스스로 점검하게 하면 '완료 보고했으나 틀린' 산출물이 감소

자료: Claude Code, 상상인증권 리서치센터

- ①Head(name·description) — 스킬 목록에 항상 떠 있는 유일한 부분으로, 모델은 이 한 줄만 보고 호출 여부를 결정. 따라서 사용자가 실제로 입력할 문구를 그대로 작성 ("세미나 실적 정리해줘")
- ②Body(절차) — 무엇을 받아(입력), 어떤 순서로(단계), 무엇을 내놓는지(형식·저장 경로)를 번호로 기술. 호출된 뒤에만 읽히는 부분이며, 스크립트와 연동
- ③Tail(검증) — 행 수·단위·저장 경로처럼 중요 사항을 기계적으로 판정할 수 있는 완료 조건을 명시. 업무 완료 때마다 자체 점검을 강제하면 완료 보고했으나 환각이 있거나 내용이 틀린 산출물이 감소
- 등록 기준은 반복성 — 스킬이 호출되지 않으면 본문이 아니라 description부터 수정

Chapter 01

Prologue: AI Agent에 맞는 습관

Context Engineering의 시대

상상인증권

Chatbot → RAG → AI Agent: 3번의 전환

- 챗봇 시대(2020~2023): '마법의 단어, 잘 쓰는 비법' 등 레시피 위주의 Prompt Engineering이 성능을 좌우 – 새 정보 전달 수단이 없어 유저 Input에만 의존
- RAG·추론의 시대(2023~2024)에는 검색 기능 + 파일 첨부로 학습 데이터 이상의 Input이 들어갔으며, 추론을 통해 유저의 Prompt를 내부적으로 수정하는 것이 가능해졌으나, 결과값에 대해서는 검증이 부족했음 → 환각 문제 대두
- 에이전트·하네스 시대(2025~): 계획→도구→관찰→수정을 스스로 반복하며 파일·터미널·브라우저를 직접 조작. 이 루프를 제품으로 만든 것이 AI 하네스(Claude Code·Codex·Hermes), 그리고 하네스 내부에 들어갈 지시는 Context(문맥)

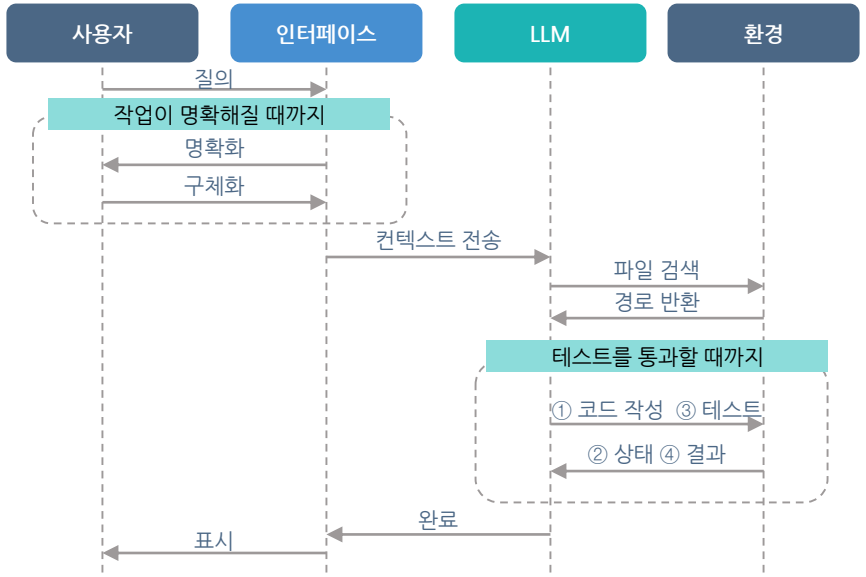
그림 3. 현재 LLM 작업 플로우 – 사용자 프롬프트의 중요도 축소

자료: 상상인증권 리서치센터

Prompt Engineering → Context Engineering

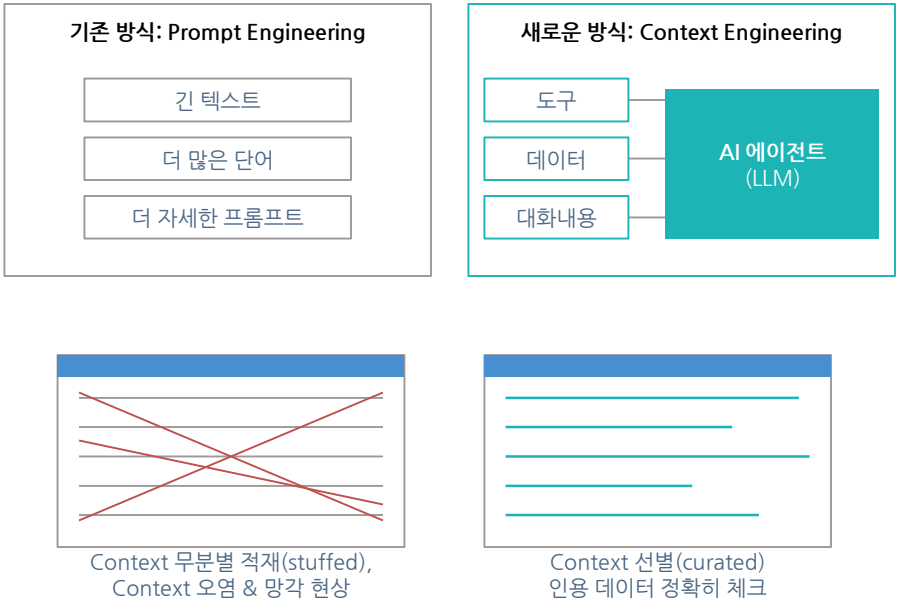
- 과거: 프롬프트를 다듬어 한 번에 정답 유도 & 비슷한 결과물을 재생산 → 사용자 노하우 공유에 의존
- 현재: 작업의 목표·입력·출력·저장·제약을 에이전트(Orchestrator)에 위임하고 산출 파일을 검수
- 역할극·마법의 문장/레시피('단계별로 생각해' 등)·보상·협박 문구는 이제 효과 미미
- Context Engineering의 시대 → AI 모델의 ①작업 내역(목적·입력·출력·메모리·금지 사항) ②완료 조건(Goal) ③참고 데이터베이스 ④검증·검산을 조정하는 시대

그림 4. 현재 AI Agent 작업 구조 도식도 (*Claude 기준)



자료: Anthropic, 상상인증권 리서치센터

그림 5. 현재는 Context를 깔끔하고 명확하게 유지하는 것이 관건



자료: Google Brain, 상상인증권 리서치센터

AI Agent 시대에 맞는 새 방식

- 추론 시절의 습관 대신 AI Agent 시대에 맞는 지식 업데이트 필요
- 1. 목표·입력·출력·제약을 명세 — 학습 이후 정보는 검색·파일·API로 '어디서 가져올지' Source와 신뢰도 지정
- 2. 단계 세분화 대신 목표 지정 후 Subagent 여럿에 위임, 검증만 단계별 수행
- 3. Context Window 효율화를 위해 CLAUDE.md·스킬·노트로 컴퓨터(local drive)에 저장

*별도 앱 설치 없이는 권한 문제로 불가능. Claude Code, Codex 등 Harness만 가능

표 1. 과거 습관과 에이전트 시대 습관

과거 방식	더 이상 통용되지 않는 이유	에이전트 시대의 방식
'마법의 문장' 수집 — 역할극 등	추론 과정 모델 내부에 내장됨 AI 모델이 사용자에게 의도 재확인	목표·입력·출력·제약을 명세
RAG 기술을 이용한 인터넷 검색 대세	수집한 자료의 우열 판단 곤란 → 컨텍스트 오염	"어디서 가져올지" 정확한 Source를 지정
작업을 잘게 세분해 한 단계씩 지시	Subagent가 모델 내부에 탑재	목표 단위로 위임, 검증만 단계별
긴 채팅 히스토리로 맥락·기억을 유지	컨텍스트 창이 가장 희소한 자원 & 대화를 닫으면 소멸	CLAUDE.md·스킬·노트로 파일화
답변을 복사해 문서에 붙여넣기	에이전트가 파일을 직접 작성	파일로 받고 숫자만 원본 대조

자료: 상상인증권 리서치센터

Chapter 02

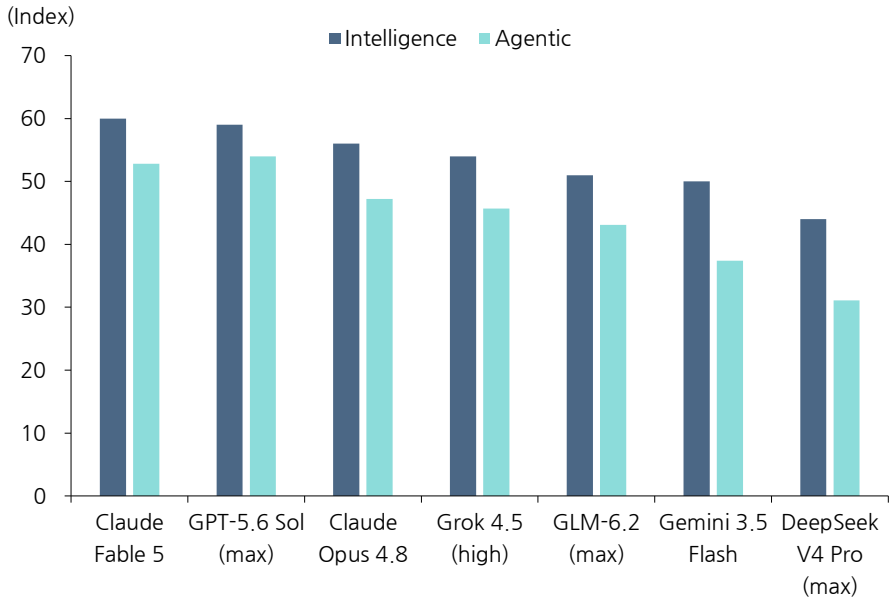
모델/도구 고르기 & Hermes란?

Hermes/Openclaw/Claude Code/Codex/Antigravity…
아직은 AI모델의 춘추/전국시대

벤치마크 점수만 보고 모델의 우열을 가리기는 곤란

- Artificial Analysis, SWE Bench, LLM Arena ELO 등 점수만으로 주력 모델 선정은 위험 — 공개 벤치마크는 짧은 응답·제한적 과제 위주이며, 공개된 벤치마크는 과적합(정답 암기)에 취약—도구 호출·긴 대화가 주력인 실무와는 간극 심함
- 투표형 순위(LLM Arena)는 '보기 좋은 답, 긴 답변'에 치우치고, 아부 문제(Sycophantic Behavior)로 유저에 과하게 동의
- 실무 기준: ①다단계 작업 성공률/완주율 ②도구 호출 안정성과 실패 후 재개 신뢰도 ③후반까지 지시·제약 유지, 망각 문제(Catastrophic Forgetting) 최소화 ④오류/문맥오염 수정 포함 총비용

그림 6. 주요 모델의 Intelligence vs Agentic Index 비교



자료: Artificial Analysis, 상상인증권

표 2. 실무에 경량 모델이 어려운 이유

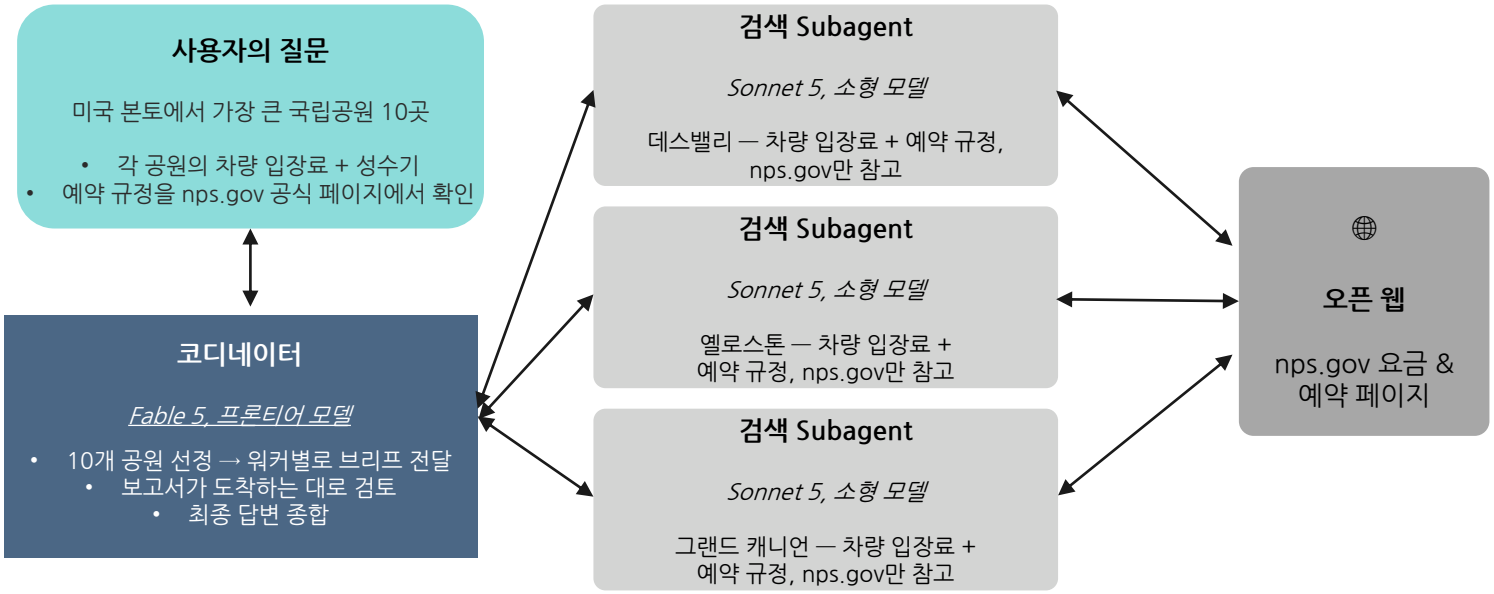
지표	Fable 5 (low)	Sonnet 5 (high)	GLM-5.2	배율 (GLM/ Fable)
성공율	80%	80%	80%	동일
총비용	$2.27	$8.43	$15.82	7.0x
성공 건당 비용	$0.28	$1.05	$1.98	7.0x
총 토큰 소모량	**0.14M**	**1.75M**	**10.82M**	**77.0x**
총 소요 시간	**14분**	**59분**	**124분**	**8.9x**

자료: VulcanBench, 상상인증권

고성능 모델 & 경량 모델 혼합 사용은 시장의 성숙화

- 고급 모델 회의론은 시장성 문제가 아니라 업무 배분·Segmentation·성숙화의 별개 사안으로 판단
- 단순·반복은 Sonnet·GLM·DeepSeek 등 경량·가성비 모델로 충분하나, 직관·판단이 필요한 orchestrator·aggregator는 최상위 모델의 영역 — harness로 지능 향상은 불가
- 고성능 모델이 오류를 빠르게 색출해 오히려 더 낮은 비용에 해결하는 경우도 다수 — Token 단가와 $ per Task는 별개, 저가 모델 비교는 시간 비용 미포함

그림 7. 고성능·경량 모델 혼합 아키텍처 Anthropic 공식 예시: 국립공원 정보 수집 Task



자료: Anthropic, 상상인증권 리서치센터

본인에게 맞는 모델만 사용해도 충분

- 사용자가 많은 Claude Code, Codex, Cursor 등은 노하우 공유가 활발하고 평균 이상의 성능을 보장
- 다만 모델별로 선호가 갈리고 피크 시간대 성능 저하도 빈번
- 이상적 선택법: 공개 순위는 참고만, Openrouter·Copilot 등 병렬 환경에서 직접 시험 — 모델 선택은 개인 취향도 중요
- 가격은 100만 토큰당 달러($/1M tok) 표기 — 한국어는 1.5~2배를 곱해야 체감 비용에 근접

표 3. 주요 AI 랩의 영역별 리더보드 순위 비교

회사	AI 에이전트	코딩(Web)	범용 텍스트	검색/리서치 텍스트	문서 작업	이미지	비디오
OpenAI (GPT)	**2위**	**3위**	5위	**2위**	**2위**	**1위**	5위
Anthropic (Claude)	**1위**	**2위**	**1위**	**1위**	**1위**	-	-
Google (Gemini)	7위	10위	**3위**	4위	**3위**	4위	**1위**
xAI (Grok)	4위	5위	7위	5위	9위	7위	6위
Meta (Muse Spark)	5위	6위	**2위**	-	5위	3위	**3위**
Zhipu AI (GLM)	**3위**	4위	8위	-	8위	-	-
Alibaba (Qwen)	6위	8위	6위	-	6위	9위	4위
Moonshot (Kimi)	8위	**1위**	4위	-	4위	-	-
Bytedance (Seedance)	-	7위	-	-	-	6위	**2위**
비고				Baidu 3위	Mini 7위	MS 3위	

자료: Lmarena, 상상인증권 리서치센터

언어와 토큰 — 한국어는 영어 대비 평균 1.6배

- 한국어 청구비용이 큰 이유: 영어 중심 Tokenizer가 음절·조사·어미를 잘게 분할 — 영어 대비 평균 약 1.6배 (모델별 상이)
- Claude 계열은 한국어 배수가 특히 높아, 자동 로드되는 문장을 방치하면 비용·컨텍스트가 빠르게 포화
  → CLAUDE.md 요약·스킬 설명·공통 규칙은 영어 병기 검토
- 핵심 워크플로우/긴 자료는 영어로 저장하되, 산출물·노트 본문은 한국어로 작업 가능

표 4. 언어별 토큰 소모 배수 (영어 대비)

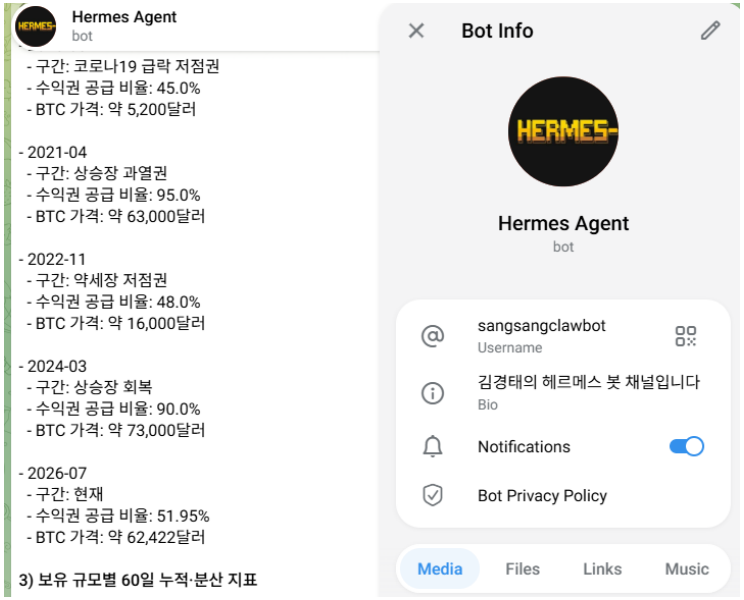
	OpenAI	Gemini 3.1	Qwen3.6	DeepSeek V4	Kimi K2.6	Anthropic	Avg
중국어	1.15x	0.98x	0.85x	0.87x	0.81x	1.65x	**1.02x**
한국어	**1.40x**	**1.29x**	**1.13x**	**1.59x**	**2.02x**	**2.59x**	**1.60x**
스페인어	1.18x	1.17x	1.20x	1.36x	1.63x	1.55x	**1.34x**
프랑스어	1.30x	1.33x	1.34x	1.50x	1.79x	1.71x	**1.48x**
러시아어	1.31x	1.26x	1.27x	1.45x	2.03x	1.96x	**1.51x**
아랍어	1.31x	1.41x	1.23x	1.55x	2.03x	2.74x	**1.64x**
힌디어	1.37x	1.17x	1.85x	2.67x	2.63x	3.11x	**2.00x**
Avg	**1.33x**	**1.22x**	**1.23x**	**1.49x**	**1.76x**	**2.07x**	**1.49x**

자료: Arxiv, 상상인증권 리서치센터

사용 방식 선택 — Hermes vs Claude Code 등

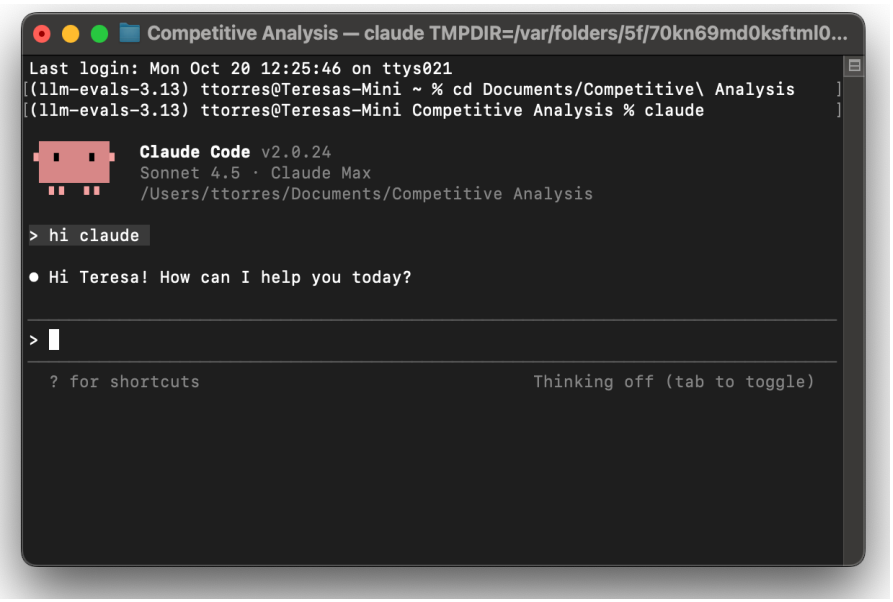
- API 연결 + Hermes: Claude·GPT·Gemini·Grok을 한 틀에서 구동 가능 : 특정 업체 종속과 한도·장애·가격 변동 리스크를 분산, Telegram·Slack 등 메신저와도 연결 가능하여 외부 작업 가능
- 한 업체로 충분하고 메신저가 불필요하면 → Claude Code·Codex 등 전용 하네스. 모델과 도구를 한번에 제공하며, 편의성 및 Tool Calling 작업 (엑셀, 파워포인트 작업 등)에 우위를 보임
- 같은 작업 폴더·md/csv 경로를 공유하면 도구가 달라도 검수 규격·작업 메모리는 하나로 유지

그림 8. 현재 필자가 사용중인 Hermes Agent 텔레그램



자료: Telegram, 상상인증권 리서치센터

그림 9. Claude Code, Codex 등 자체 Harness – CLI, GUI 버전



자료: Claude Code, 상상인증권 리서치센터

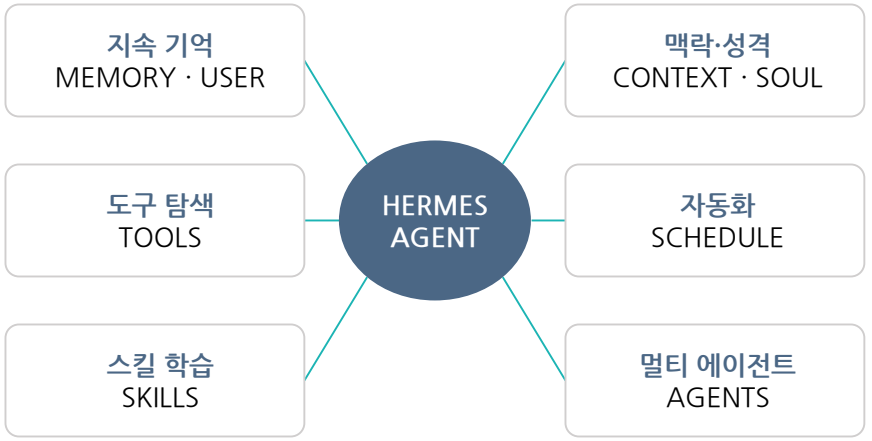
Hermes Agent — 외부 업무가 잦을 경우에는?

- Hermes는 오픈소스(MIT) 에이전트 하네스로, 특정 모델 회사에 묶이지 않고 사용 가능. Openclaw(Clawdbot)와 유사
- 여러 모델의 협업과 메신저(텔레그램·디스코드·슬랙)·PC 연결을 지원 — 자체 Harness가 없는 오픈소스 모델(GLM, Deepseek, Kimi, Exaone) 활용에 필수
- 사용할수록 개선되는 것이 핵심: 채팅 로그에서 스킬을 자동 생성·최적화하고, 과거 대화를 검색·요약해 장기 기억처럼 활용
- AI API Key 호출용 & 코드 실행용 서버가 있으면 편리하며, 주로 맥미니를 사용하는 사례가 다수

그림 10. Hermes 셋업 구성도 (API·서버·메신저 연동)

자료: 상상인증권 리서치센터

그림 11. Hermes 핵심 역량 6가지

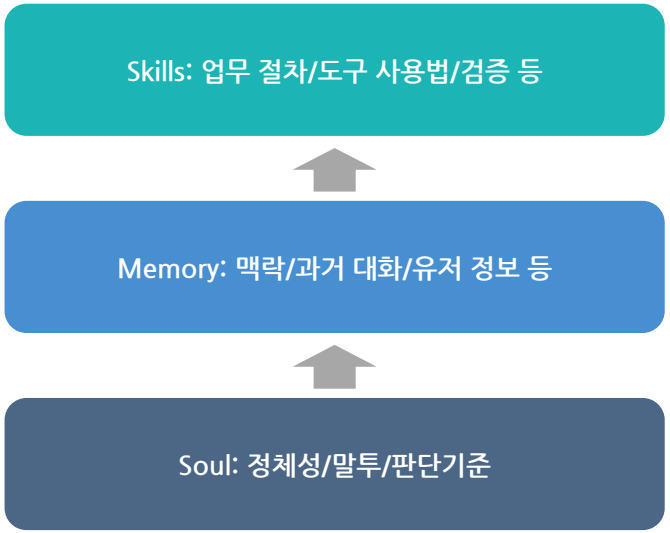


자료: NousResearch, 상상인증권 리서치센터

Hermes 요소 — SOUL(성격) · 스킬 · Third Party App 연동

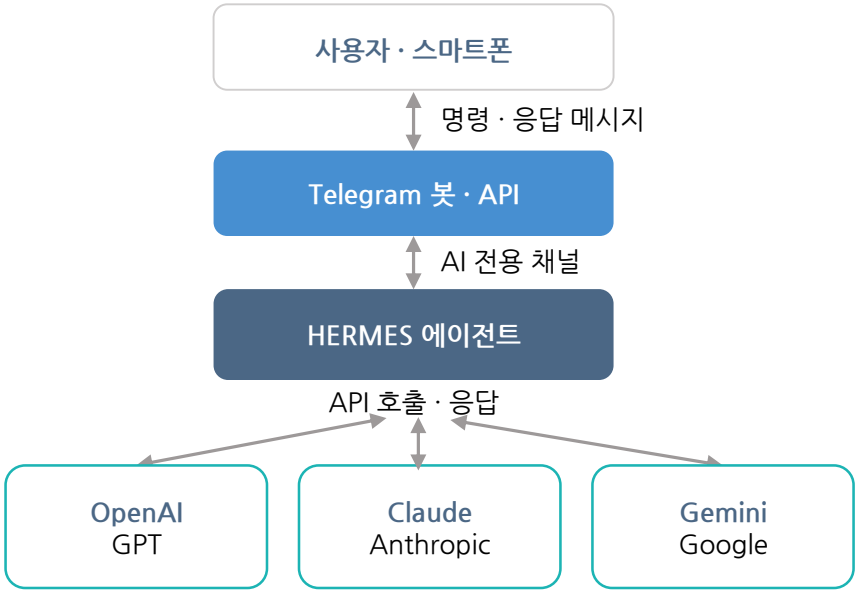
- SOUL.md: 모델의 성격/어투/보고 형식을 반복 설명하지 않으려면 SOUL.md(~/.hermes/SOUL.md)에 고정이 필요. 리서치 업무에는 사실 여부 확인 등 제약 사항, 성격을 SOUL에 저장해 두면 매번 같은 톤이 적용되어 검수 기준이 맞춰짐
- Skills: 반복 절차는 /skills로 검색·설치하고, 복잡한 작업 후 절차를 스킬로 남기면 다음부터 같은 유형이 더 안정·빠름(사용 중 스스로 다듬음) SKILL.md와 동일. 원하는 형태의 산출물을 Skill화 하여 저장하여 다음에도 재활용 하는 것이 핵심
- Telegram, Slack 등 모바일 연동: Telegram API, Telegram Bot, Slackbot 등을 활용해 AI 전용 채널을 만든 이후, 외부에서도 스마트폰을 통해 컴퓨터에서 가능한 작업을 수행 가능하며, 반복 스케줄링에도 편리

그림 12. Hermes 3계층 — Soul · Memory · Skills



자료: NousResearch, 상상인증권 리서치센터

그림 13. Telegram · Hermes · LLM 연동 구조



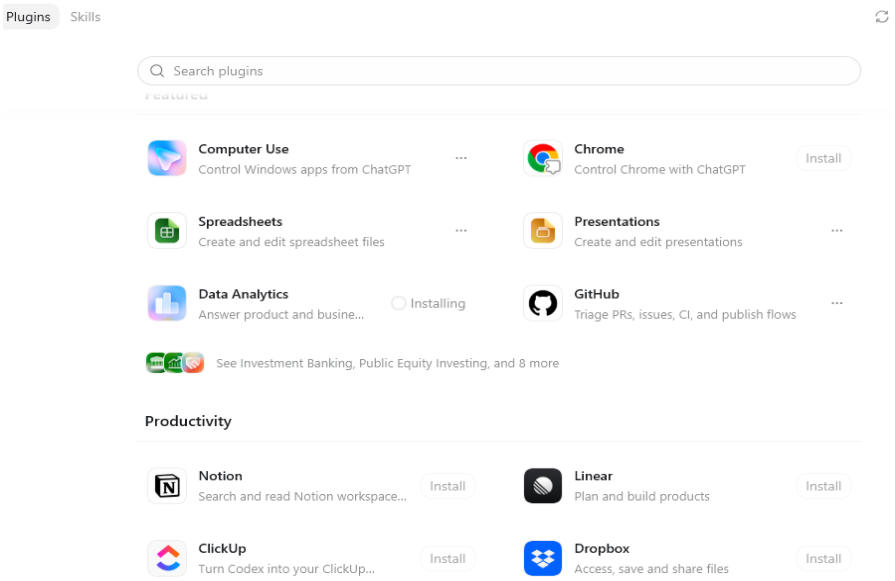
자료: 상상인증권 리서치센터

모델 자체 Harness: Claude Code, Codex, Antigravity 등

- 모델 자체 Harness인 Claude Code/Codex/Antigravity/Grok Build 등은 주로 CLI(명령프롬프트)·GUI(앱) 버전 제공 — 터미널 또는 앱 다운로드로 설치. Git 등 애드온이 필요하나 외부 하네스 대비 설치, 사용이 훨씬 간편
- 프런티어 랩이 오픈소스 하네스의 기능을 빠르게 흡수해 격차 축소 중 —Telegram 등 Third Party 연결은 여전히 제한

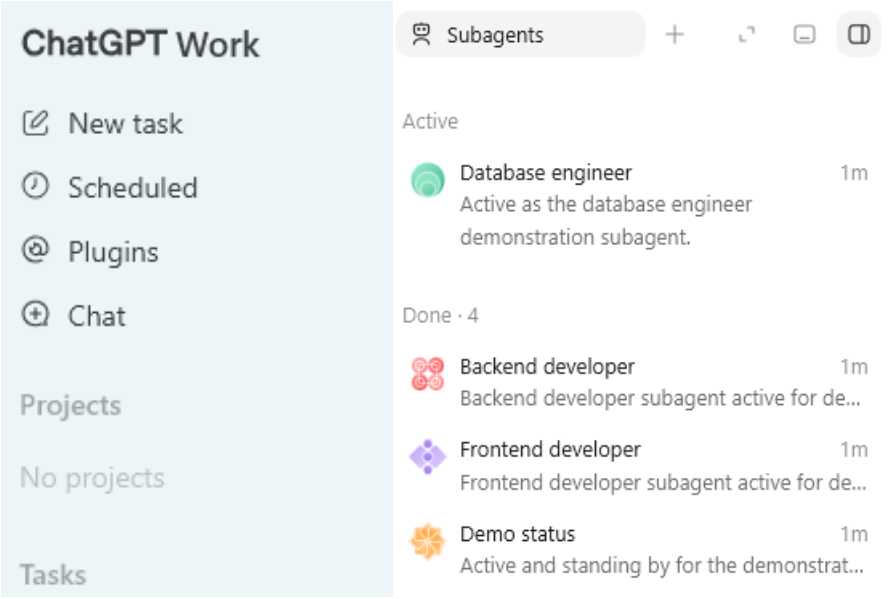
**Claude Cowork, GPT Workspace 등 샌드박스형·팀 공용 앱은 개인적으로 비추천 — 컴퓨터 사용이 제약되고 Skill 호환이 불완전

그림 14. Codex 플러그인 마켓 — Third Party App 연결



자료: OpenAI, 상상인증권 리서치센터

그림 15. ChatGPT Work — 서브에이전트 실행 화면

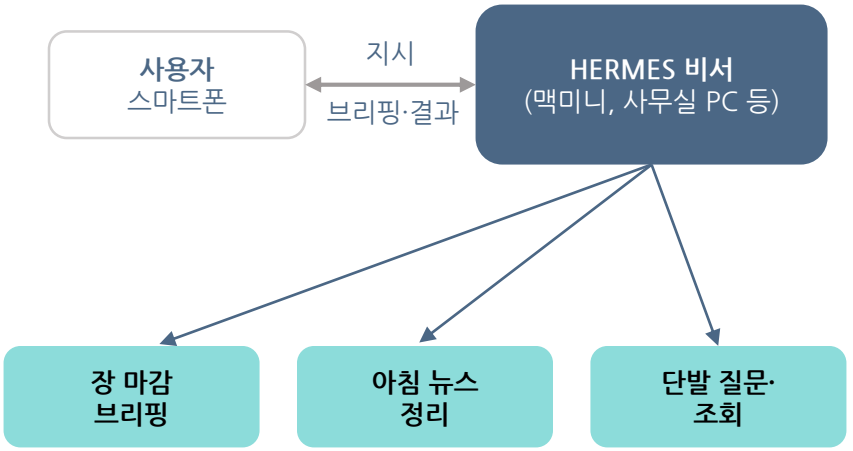


자료: OpenAI, 상상인증권 리서치센터

결론: Hermes / Claude Code – 둘 다 사용해도 괜찮다

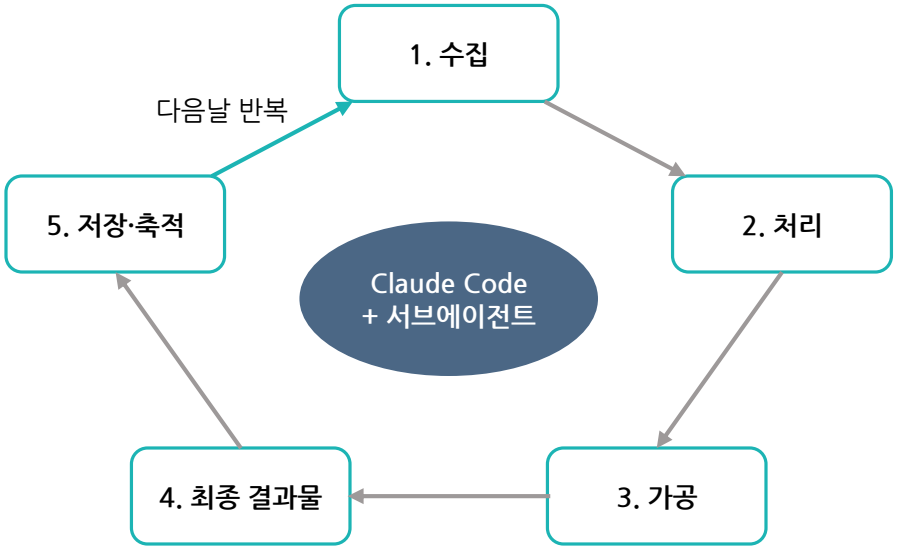
- 사무실 작업은 Claude Code — 보고서·엑셀·차트 등 중간 검토가 필요한 업무
- 수집→가공→작성→검산이 길수록 오류를 즉시 교정해야 재작업 비용 감소
- 외부·이동 시간은 Hermes — 장 마감 브리핑·아침 뉴스 등 '비서형' 업무 전담에 유리
- 다만 앱 버전 Harness는 Skill·memory·md는 계정이 아닌 기기에 저장되어 공유 불가하며 다소 락인 효과가 있음
  → 장소를 가리지 않는 접근에는 Hermes가 유리

그림 16. Hermes — 개인 텔레그램 비서



자료: 상상인증권 리서치센터

그림 17. Claude Code · 서브에이전트 일일 반복 루프



자료: 상상인증권 리서치센터

Chapter 03

AI Agent 사용법

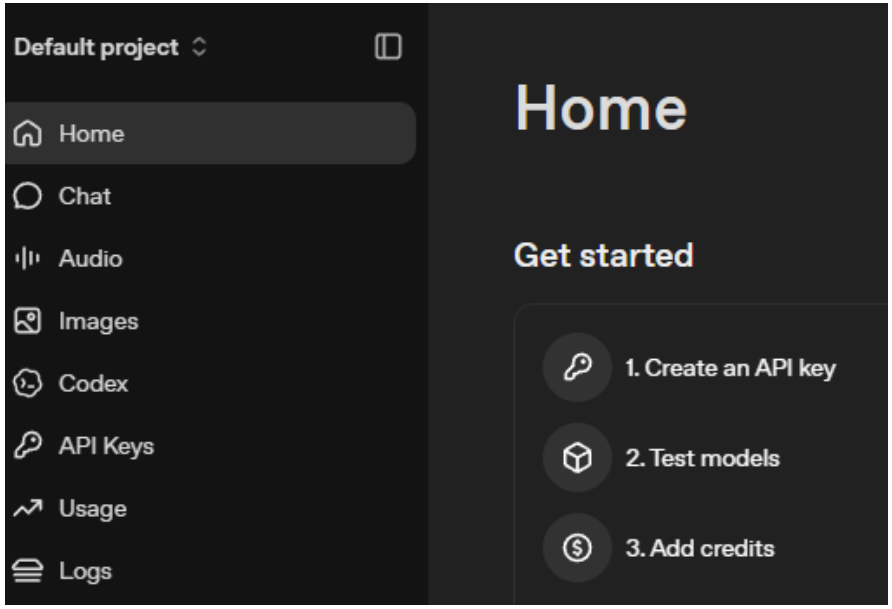
*Claude Code 메인

상상인증권

사용하기 전: API Key 사용법 – 보안에 유의 필요

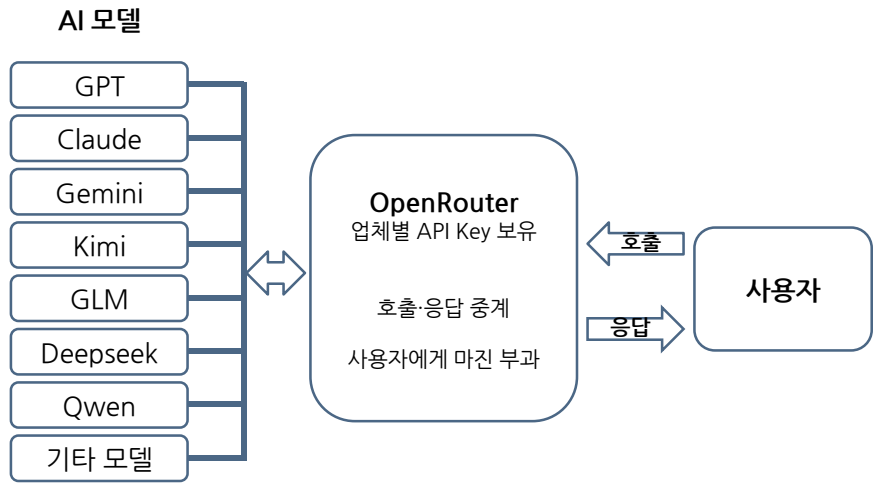
- API는 프로그램끼리 데이터를 주고받는 창구이며 API 키는 아이디— Hermes가 Claude, GPT 연동을 위해 필요
- 경로① 정액 구독(Pro·Max 등 고급 플랜): 가입 → 데스크톱 앱(Claude Code, Codex 등) 설치 → (OAuth방식) 로그인 시 브라우저 인증 창에서 승인, API 키 발급 없이 바로 사용 *Claude 정액 구독은 API Key가 발급되지 않아 Hermes, Openclaw 연결에는 사용 불가
- 경로② API 종량제: console.anthropic.com, openai.com/api 등 종량제 전문 페이지에서 결제 수단 등록 → API 키 발급 → 도구 설정에 입력. 사용량(토큰)만큼 과금되고 한도 설정 가능 — Hermes 등 외부 하네스·자동화 연결에 필수

그림 18. API 키 토큰 종량제($/1M Token) 홈페이지



자료: OpenAI, 상상인증권 리서치센터

그림 19. OpenRouter 등 Vendor 사이트는 여러 모델 활용 가능



자료: Openrouter, 상상인증권 리서치센터

Claude Code 첫 실행 — 작업 폴더 지정부터

- Claude Code 등 AI Agent는 대부분 작업에 컴퓨터 활용이 필수적
- 전용 작업 폴더: 에이전트는 '지금 열린 폴더'가 곧 작업 공간이자 읽기·쓰기 허용 범위 — 문서함·바탕화면이 아닌 전용 폴더(예: C:/work/ai)에서 실행, 산출물은 대화창 답변이 아니라 폴더 내부에서 파일이 저장됨
- 필요한 모든 것을 한 폴더에: 입력 데이터(xlsx·csv·pdf)·스킬·스크립트(py)·작업 규칙(CLAUDE.md)을 모아 두면 검수·재사용

그림 20. Setup 프롬프트 예시— CLI 버전 (Command Prompt) vs GUI 버전 (앱)

<Claude Code 작업 폴더 첫 세팅 Prompt 예시>

1. 유저 인터뷰
- 직무, 주간 반복 업무 종류 3~5개
- 사용 도구·데이터, 산출물 형식
- 보안 제약 (외부 유출 금지, 참조 폴더)

2. 제안
- 폴더 구조: 루트 + /archive + /memory, - CLAUDE.md 초안: 사용자 정보, 저장 위치, 파일명 규칙, 약어 사전, 보안 규칙
- /memory 스킬 1~2개: 최다 반복 업무의 단계별 작업법

3. 승인 후 생성 → 테스트 1회로 작동 확인
공통: 덮어쓰기 전 확인, 네트워크 드라이브 읽기 전용, CLAUDE.md 최소화(상세 절차는 /memory)

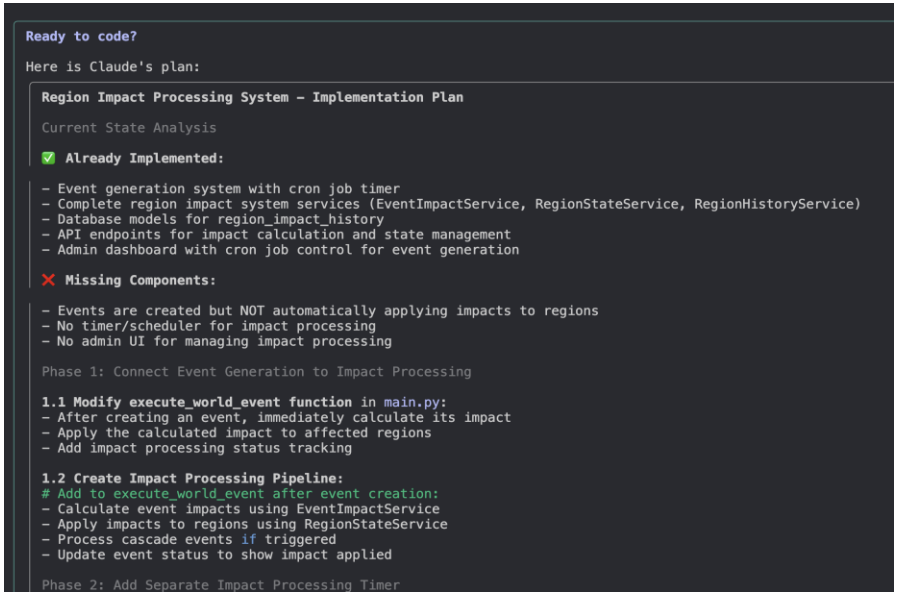
Prompt 실행 | Workspace 생성

자료: Claude Code, 상상인증권 리서치센터

업무 루프화/자동화 프로세스— 계획 · 서브 에이전트 · 목표

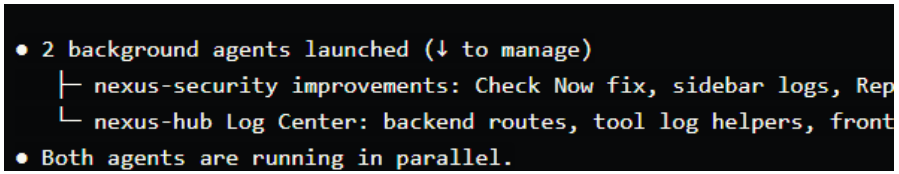
- 계획 모드: '계획부터 세워' 지시 시 파일 수정 전에 계획서를 먼저 수령 — 실수 범위가 줄고, 오류는 실행 전에 교정
- 서브 에이전트: 긴 문서나 여러 출처를 한 대화에 담으면 Context Window 포화 → 별도 에이전트로 분산
- 목표(Goal): AI가 판정 가능한 조건일 때만 재시도 루프 가동 — '그럴듯할 때까지' 같은 모호한 조건 지양

그림 21. 계획모드 – Anthropic 예시

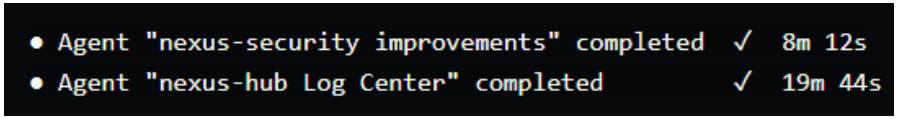


자료: Anthropic Blog, 상상인증권 리서치센터

그림 22. 서브 에이전트 & Goal – CLI 창 예시, GUI에서도 가능



Sub Agent 병렬 실행 예시



Orchestrator - Goal 설정 이후 완료 점검

자료: Claude Code, 상상인증권 리서치센터

서브 에이전트(Subagent) 정의 — 역할의 장기메모리화

- 서브 에이전트는 메인 에이전트(Orchestrator)가 위임한 작업을 별도 컨텍스트 윈도우에서 수행하고 결과 요약만 반환하는 하위 AI로, 리서치·편집·데이터 검증처럼 역할 단위 분리와 병렬 실행이 가능
- 역할 정의는 .claude/agents/ 경로에 md 파일로 저장하며, Head에 이름·발동 조건·허용 도구·사용 모델을, 본문에 작업 범위·판단 기준·종료 조건을 계약서 형식으로 기술
- 파일로 명세서를 저장하면 대화가 바뀌어도 동일한 품질의 분업이 재현 & 메인 에이전트의 컨텍스트 오염도 차단

그림 23. 서브 에이전트 실제 md 파일 - 리서치작성자

.claude\agents\리서치작성자.md

Name	리서치작성자
Description	한국어 금융 리서치 원고 작성 전담. "리서치작성자로 써줘", "이 자료로 시황/기업분석 원고 작성" 요청 시 호출. 수집·계산이 끝난 자료를 받아 글만 쓴다. 데이터 수집·코드 작성·파일 대량 탐색에는 사용하지 말 것.
Tools	Read, Grep, Glob, Write
Model	opus

역할

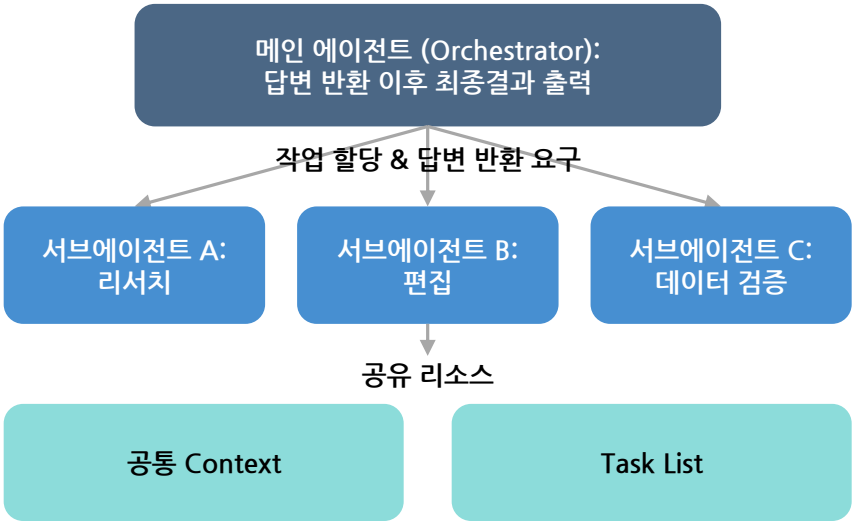
수집·검증이 끝난 자료(md 노트, 표, 데이터 요약)를 입력받아 상상인증권 리서치센터 기준의 한국어 원고를 작성한다. 글쓰기 전담 — 데이터 수집, 수치 계산, 코드 작성은 범위 밖이다.

범위

- 시황 메모, 기업 분석, 실적 요약, 투자 논리(thesis), 세미나 자료 원고
- 입력: 지정된 자료 파일(경로를 지시문에서 받음)만 읽는다
- 출력: 지정된 경로에 md 파일 1개로 저장 (기본: 작업 폴더 루트)

자료: 상상인증권 리서치센터

그림 24. 메인-서브 에이전트의 역할 분리 예시

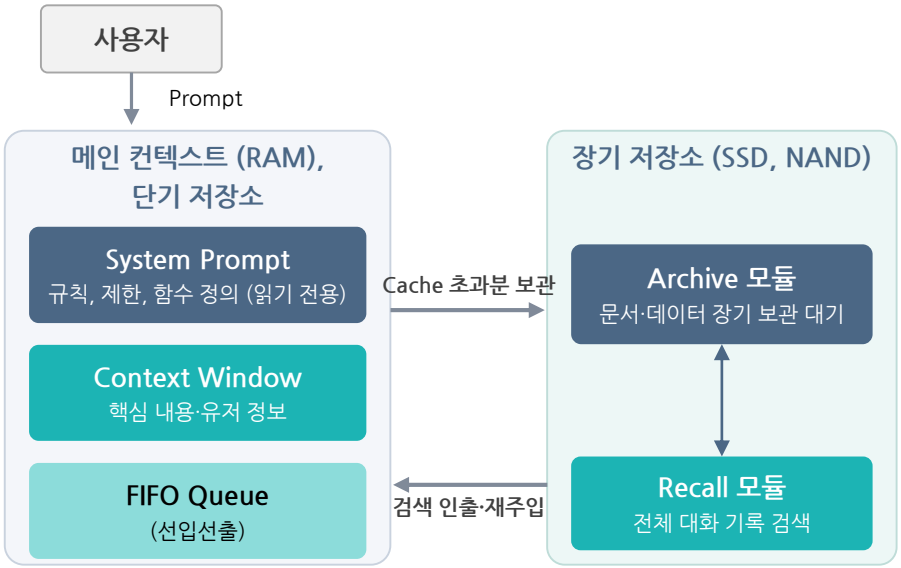


자료: Anthropic, 상상인증권 리서치센터

에이전트 기억 — 사용자는 컨텍스트 윈도우가 병목

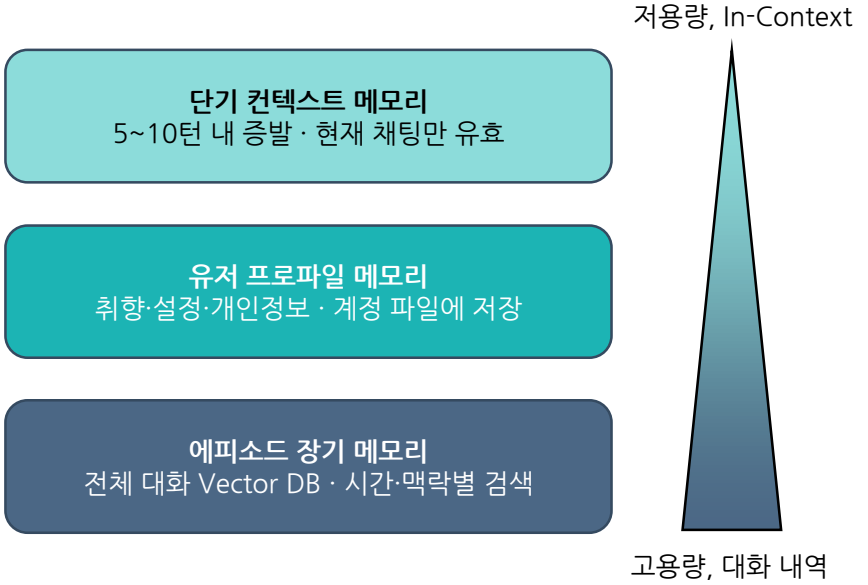
- 한 대화의 지시문·메시지·파일·도구 결과는 컨텍스트 윈도우 안에서 처리되며, 창이 차면 과거 대화가 요약(Compact) 또는 삭제(Pruning)되어 수치·세부 지시부터 소실
- 컨텍스트 윈도우는 보통 선입선출(FIFO) 구조로 관리되어 오래된 내용부터 밀려나므로, 보존이 필요한 정보는 별도 저장소에 보관(Archive)한 뒤 검색으로 재호출(Recall)하는 이원 구조가 필요
- 기억은 수명에 따라 계층화 — 수 턴 내 증발하는 단기 컨텍스트, 사용자 설정을 담는 프로파일 메모리, 전체 대화를 축적하는 에피소드 장기 메모리로 구분되며 저장 위치와 지속 기간이 각기 상이

그림 25. 컨텍스트 윈도우·실제 하드웨어 사용 구조



자료: 상상인증권 리서치센터

그림 26. Context Window의 저장/보관 계층 개념

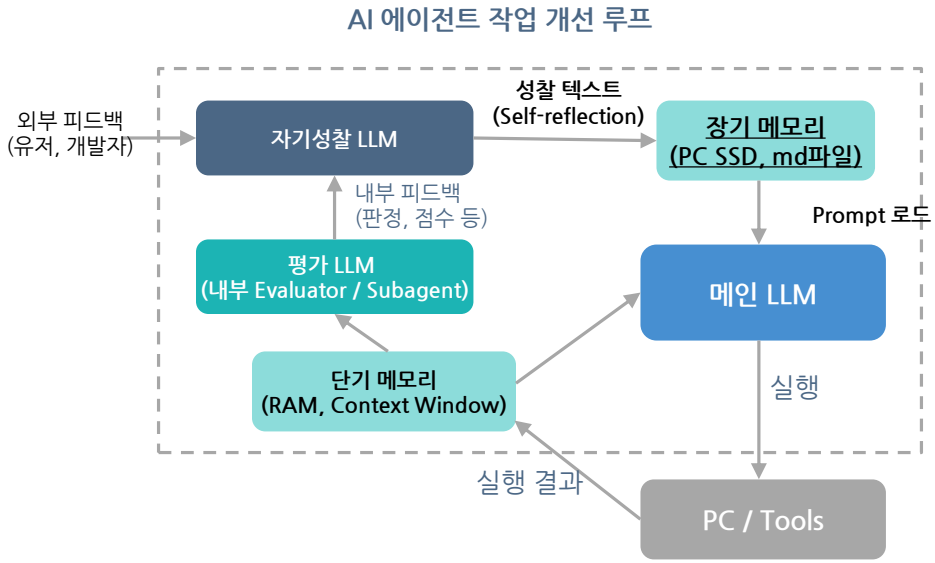


자료: 상상인증권 리서치센터

KV Cache의 한계와 단기 메모리 / 장기 메모리 구분

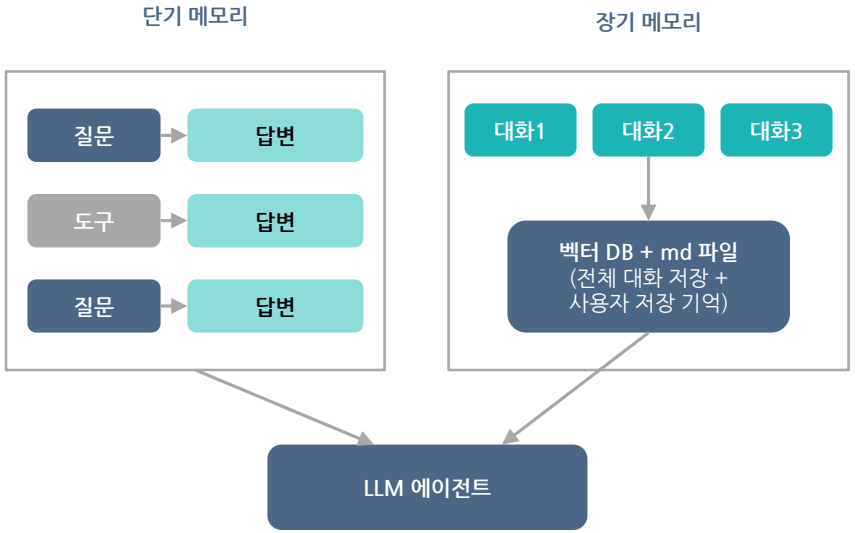
- AI 에이전트는 품질 개선을 위해 메인·평가·자기성찰 LLM이 외부/내부 피드백과 성찰 텍스트를 주고받는 다단 루프 구조
- 루프 횟수에 따라 단기/장기 메모리가 축적되어 그에 따른 계산량/데이터 저장량이 기하급수적으로 증가
- KV캐시와 컨텍스트 창은 단기 메모리에 해당하는 휘발 영역 —대화 종료와 함께 소멸. 성찰/추론 텍스트 저장 불가
- 따라서 사실상 작업/추론 성능의 핵심인 성찰 텍스트·대화는 장기 메모리로 이관해야 보존 — 대화 기록을 벡터 DB·파일로 축적해 다음 대화가 검색·회상하게 하는 것이 장기 기억의 구현

그림 27. AI 에이전트의 작업, 성찰, 메모리 루프 구조



자료: 상상인증권 리서치센터

그림 28. 현 프런티어 LLM 에이전트는 단기/장기 메모리 모두 참조

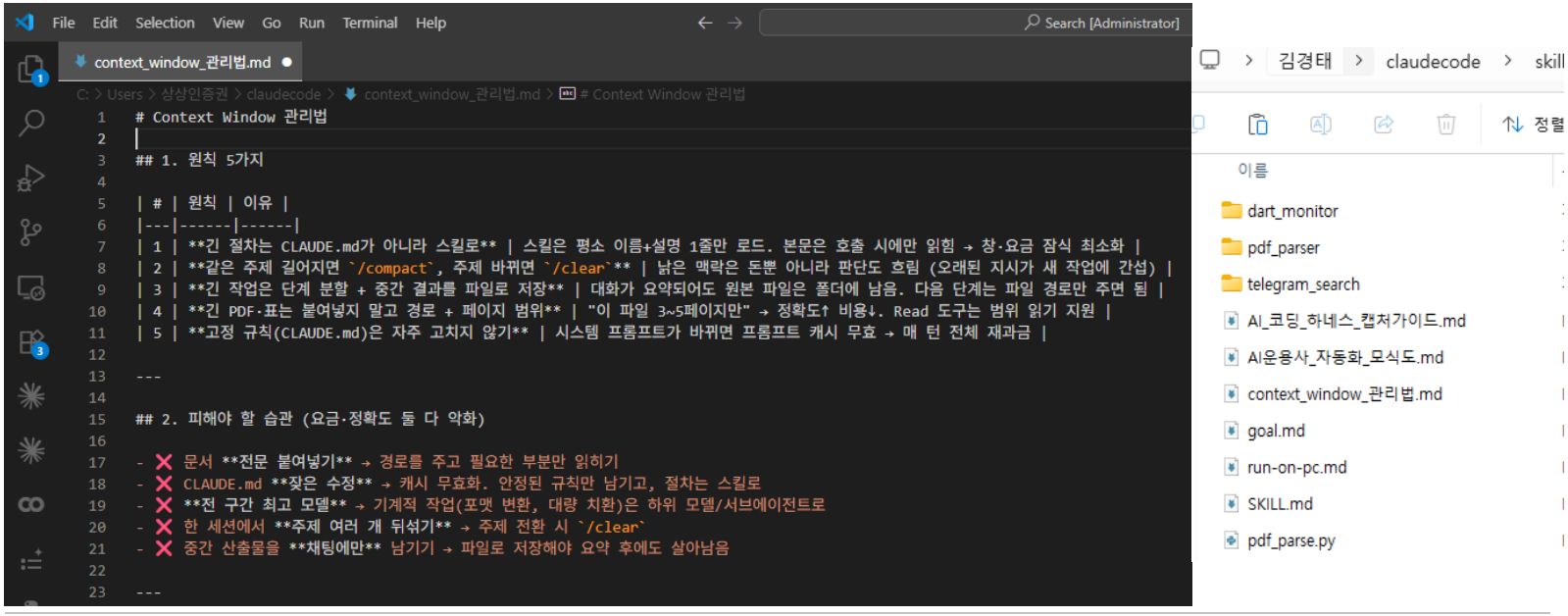


자료: 상상인증권 리서치센터

Markdown 파일 (.md) : 장기 기억의 핵심

- 앞선 장기 메모리의 가장 단순한 실무 구현이 Markdown(.md) 파일 — 프로젝트 루트의 CLAUDE.md는 대화 시작 시 자동 로드되어 폴더의 목적·규칙·경로를 모든 작업의 출발점에 주입
- System Prompt와 유사하게, 매 응답 전에 md 내용이 먼저 입력되는 구조: 대화 간 기억이 유지
- 따라서 대화가 길어져 컨텍스트가 포화되면 진행 내역 & 다음 할 일을 md에 기록한 뒤 새 대화에서 이어가는 것이 효율적

그림 29. 사내 프로젝트 CLAUDE.md 실제 내용

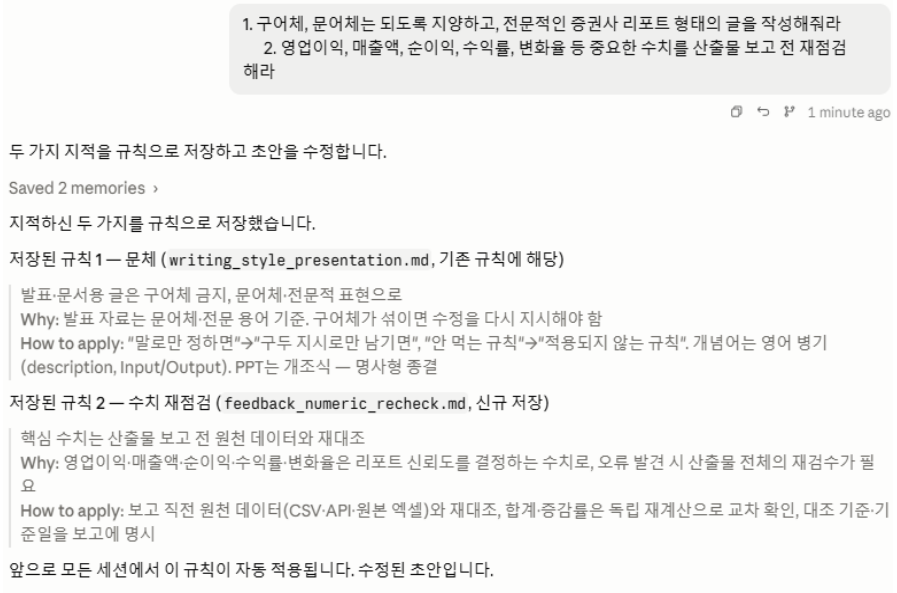


자료: 상상인증권 리서치센터

장기 기억 구현으로 실수 재반복 문제 해결

- 문제 지적/답변 교정 직후 같은 대화 내에서 Markdown(.md) 파일에 저장하는 방식이 가장 효과적 (아래 그림 참조)
- 중요 규칙을 고정하는 요령은 왜(Why)와 어떻게(How)의 분리 — 오답만 기재할 경우 저성능 AI는 문제 재발이 빈번함
- 1. 왜 문제가 생기는지와 2. 다음에 어떻게 할지를 함께 메모리에 기재해야 유사 상황까지 대응 가능

그림 30. 오류 지적 대화 → 규칙 고정



자료: 상상인증권 리서치센터

그림 31. 메모리 점검 & 정리 → Skill 승격 또는 삭제

자료: 상상인증권 리서치센터

(작업명).md로 메모리 계층화, 서열화하기 팁

- 계층화의 본질은 로드 범위: 전역 – 프로젝트 - 하위 폴더 등 → 규칙의 적용 범위와 저장 위치를 일치
- 넓은 계층은 보편 규칙, 좁은 계층은 구체 규칙 — 사용자 정보·전역 원칙은 경로 최상단에 배치, 그리고 인코딩·데이터 형식 같은 세부는 해당 폴더로 → 컨텍스트 오염, 낭비 없이 필요한 규칙만 로드
- 자동 메모리(CLAUDE.md, AGENT.md)는 에이전트가 스스로 기록·회상하는 유일한 계층 — 방치할 경우 오류가 축적되어 소위 "나쁜 습관"을 학습할 위험 존재

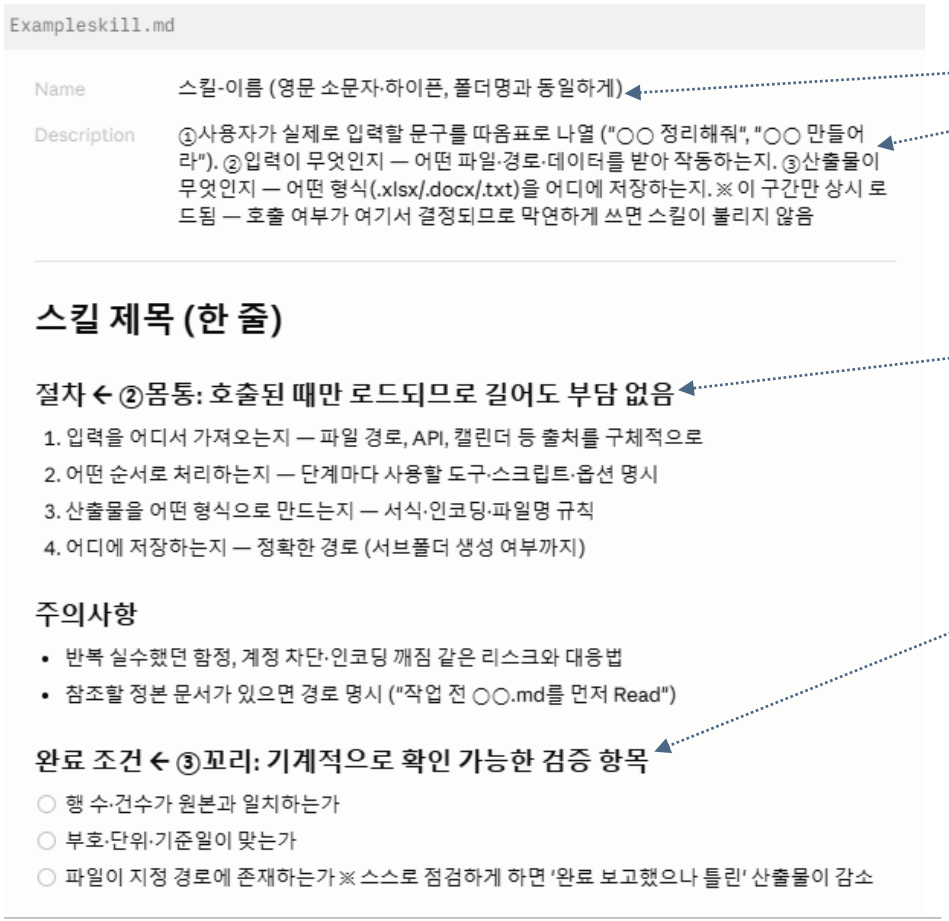
표 5. CLAUDE.md 메모리 4계층

계층	로딩 시점	기재 내용	예시
전역 ~/.claude/CLAUDE.md	전 프로젝트 상시 로드	사용자 정보·기본 작업 폴더 규칙	"산출물은 항상 작업 폴더 루트에 저장"
프로젝트 CLAUDE.md	해당 폴더 실행 시 로드	목적·규칙·결정·다음 할 일	"목표가는 본문이 아닌 표에서"
하위 폴더 CLAUDE.md	하위 폴더 작업 시 추가 로드	모듈·데이터별 세부 규칙	"본 폴더 CSV는 EUC-KR 형태로 저장"
자동 메모리 memory/	에이전트가 자체 기록·회상	학습한 사실·피드백·상태	주기 점검 — 오류 항목 삭제

자료: 상상인증권 리서치센터

특수 Markdown - 스킬(SKILL.md) 개요

그림 32. 스킬·절차 파일 예시



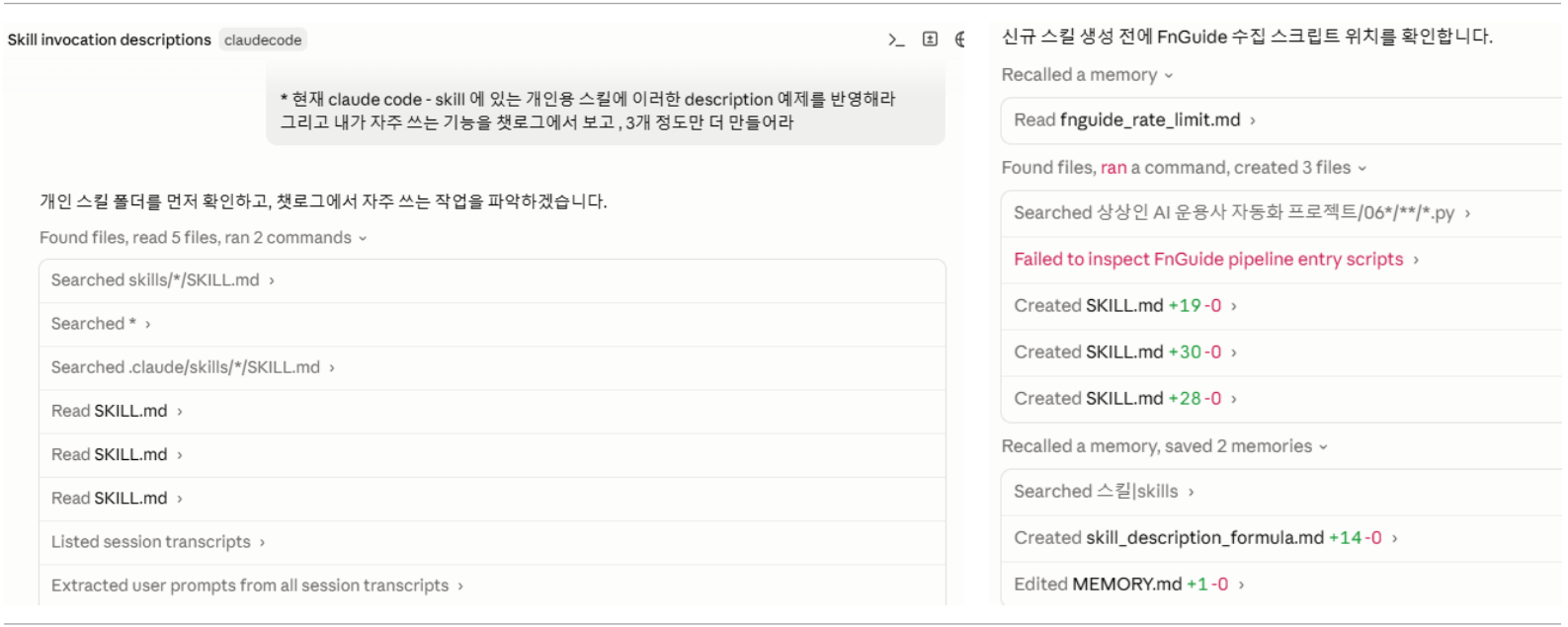
자료: 상상인증권 리서치센터

- ①Head(name·description) — 스킬 목록에 항상 떠 있는 유일한 부분으로, 모델은 이 한 줄만 보고 호출 여부를 결정. 따라서 사용자가 실제로 입력할 문구를 그대로 작성 ("세미나 실적 정리해줘")
- ②Body(절차) — 무엇을 받아(입력), 어떤 순서로(단계), 무엇을 내놓는지(형식·저장 경로)를 번호로 기술. 호출된 뒤에만 읽히는 부분이며, 스크립트와 연동
- ③Tail(검증) — 행 수·단위·저장 경로처럼 중요 사항을 기계적으로 판정할 수 있는 완료 조건을 명시. 업무 완료 때마다 자체 점검을 강제하면 완료 보고했으나 환각이 있거나 내용이 틀린 산출물이 감소
- 등록 기준은 반복성 — 스킬이 호출되지 않으면 본문이 아니라 description부터 수정

개별 업무 맞춤 커스텀 Skill 생성 방법

- 만드는 법: 채팅창에 "방금 이 작업, 스킬로 만들어줘"라고 작성 — 모델이 방금 수행한 절차를 컨텍스트 윈도우 내에 기억한 상태에서 .claude/skills/ 아래에 SKILL.md를 직접 생성
- 별도 코딩·문법 학습 불필요 — 작업 성공 직후 맥락이 살아있을 때 등록해야 품질이 가장 높음
- 첫 시도에 해내기보다 실전에서 다듬는 형태가 이상적. 예시) 호출되지 않으면 description에 호출 조건 추가, 산출물이 부족하면 Body 내용 보강, 실수가 반복되면 Tail 검증 조건 추가 등 지속적인 개선 필요

그림 33. Claude Code Skill 생성 과정



자료: 상상인증권 리서치센터

스킬은 호출되지 않으면 무용지물— 설명에 신경쓰자

- 모델은 스킬을 보수적으로 호출 — 사용자 지시와 description이 겹쳐 보일 때만 본문을 펼치고, 애매하면 스킬 없이 일반 대화로 처리. 절차 본문이 아무리 정교해도 description이 막연하면 호출 자체가 발생하지 않는 구조
- Description 3단 공식 — ①사용자가 실제로 입력할 트리거 문구를 따옴표 그대로 명시("세미나 실적 정리해줘") ②스킬이 받는 입력(파일 경로·캘린더·DB 등) ③산출물의 형태와 저장 위치(.xlsx·.docx 등)
- 그래도 호출되지 않으면 — 지시문에 스킬 이름을 직접 적어 1회 강제 호출한 뒤, 그때 쓴 문구를 description에 추가. 실제 사용 문구가 쌓일수록 자동 호출 확률이 올라가는, 쓸수록 트리거가 정교해지는 구조

그림 34. 스킬 description — 고반복 업무는 Skill 호출이 자주 되도록 하는 것이 중요

스킬별 Description (트리거 → 입력 → 산출물)

1. pdf-parse
   - 트리거 문구: "PDF 파싱해줘", "이 PDF 텍스트 추출해줘", "PDF 내용 정리해줘", .pdf 파일 첨부 + 내용 파악·요약·리포트 반영 요구
   - 입력: 로컬 PDF 파일 경로 (대용량·한글·2단 레이아웃 포함)
   - 산출물: `=== [page N] ===` 페이지 구분자가 있는 UTF-8 .txt 추출 파일
2. telegram-search
   - 트리거 문구: "/telegram-search", "텔레그램에서 ~ 종합해줘/정리해줘/찾아줘", "텔레그램 ~ 공포/이슈/반응 알려줘", 특정 테마(AI CAPEX·종목·매크로 이벤트)의 채널 여론 요구
   - 입력: 주제 키워드(클러스터 + 문맥어)와 기간, 로컬 아카이브 `telegram_archive/raw/`
   - 산출물: 날짜 역순·중복 제거 추출 .txt + Bear/Bull·수치·채널 스탠스 종합 보고
3. seminar-report
   - 트리거 문구: "세미나 실적 정리해줘", "세미나 내역 정리", "OO월 세미나 보고서 만들어줘"
   - 입력: Google Calendar(kktkkt2560@gmail.com) 월별 전수 조회 + 네트워크 템플릿 `형식.xlsx`
   - 산출물: 7컬럼 형식의 `김경태_세미나실적_YYYY.MM-MM.xlsx` (claudecode 루트 저장)

자료: 상상인증권 리서치센터

Chapter 04

실제 업무 활용 예시

금융권에서 자주 접하는 종류

① PDF 텍스트 추출/Parsing(정리) – 금융권용 pdf parser

- 리포트·공시·기관 보고서·논문 모두 '어디서 무엇을 어떤 형식으로 어디에'의 단일 문법으로 처리 — 지시는 ①입력 ②구간·항목 ③컬럼·단위 ④저장 ⑤제약(추측 배제·원문만)
- 긴 PDF는 전체 추출 후 "이 구간만 표로"처럼 두 번에 나눠 지시 — 원문에 없는 칸은 공란, 추출 직후 자릿수·단위·연도·별도/연결을 원문 2~3건과 대조
- 추출 단계의 결함은 후단 종합·의견에서 복구 불가 → 우선순위는 정확한 추출

그림 35. 추출 지시의 구성 요소

.claude\skills\pdf-parse\SKILL.md

Name	pdf-parse
Description	사용자가 "PDF 파싱해줘", "이 PDF 텍스트 추출해줘", "PDF 내용 정리해줘"라고 하거나 .pdf 파일을 첨부하며 내용 파악·요약·리포트 반영을 요구할 때 사용. 입력 — 로컬 PDF 파일 경로(대용량 수십 페이지, 한글, 2단 레이아웃 포함; Read 도구 대신 이 스킬 우선).

skills\pdf_parser\pdf_parse.py

```
# -*- coding: utf-8 -*-
"""PDF 텍스트 파서 — PyMuPDF(fitz) 기반, pypdf 폴백
사용법:
    python pdf_parse.py <file.pdf>                # 전체 텍스트 stdout
    python pdf_parse.py <file.pdf> --pages 1-5    # 페이지 범위 (1-base)
    python pdf_parse.py <file.pdf> --pages 1,3,7
    python pdf_parse.py <file.pdf> --out out.txt  # 파일 저장 (utf-8)
    python pdf_parse.py <file.pdf> --mode blocks  # 블록 단위(레이아웃 순 정렬)
    python pdf_parse.py <file.pdf> --info         # 페이지수/메타데이터만
"""
import argparse
```

자료: 상상인증권 리서치센터

그림 36. 추출 결과 표

0. 문서 유형 분류 (세부 작업 전 최우선)

본격 추출 전에 `--info` + 앞 1~3페이지만 추출(`--pages 1-3`)해 문서 유형을 먼저 판별하고, 아래 유형별 세부 작업을 적용한다. 유형이 애매하면 사용자에게 확인.

유형	판별 단서	유형별 세부 작업
증권사 리포트	표지의 목표주가·투자의견·애널리스트명, 2단 레이아웃	`--mode blocks` 기본 적용. 목표주가·투자의견·실적 전망 표를 8컬럼 CSV로 추출. 타사 의견 인용 금지 원칙 유의
공시·사업보고서	DART 서식, 목차에 "사업의 내용"·"재무에 관한 사항"	필요 섹션만 페이지 한정 추출. 별도/연결 구분을 모든 수치에 필수 명기. 주석(각주)까지 확인
학술 논문	Abstract·References, 저자 소	초록→결론→방법론 순으로 읽기. 수치보다

핵심 수치 CSV 추출 (실적 테이블·컨센서스·목표주가 등)

사용자가 수치 추출을 요구하면(예: "숫자 뽑아줘", "실적 표만 정리", "컨센서스 추출") 텍스트 추출에서 멈추지 말고 아래 절차로 CSV까지 산출한다.

1. 위 표준 워크플로로 .txt 추출 후, 실적 테이블·차트 수치·컨센서스·목표주가·투자의견이 있는 페이지를 grep으로 특정 (표가 깨져 있으면 해당 페이지만 `--mode blocks` 로 재추출)
2. 아래 표준 8컬럼 스키마로 CSV 작성. 컬럼 구조는 문서가 달라도 절대 바꾸지 않는다 — 여러 문서의 CSV를 그대로 이어 붙여(concat) 축적·관리하기 위함

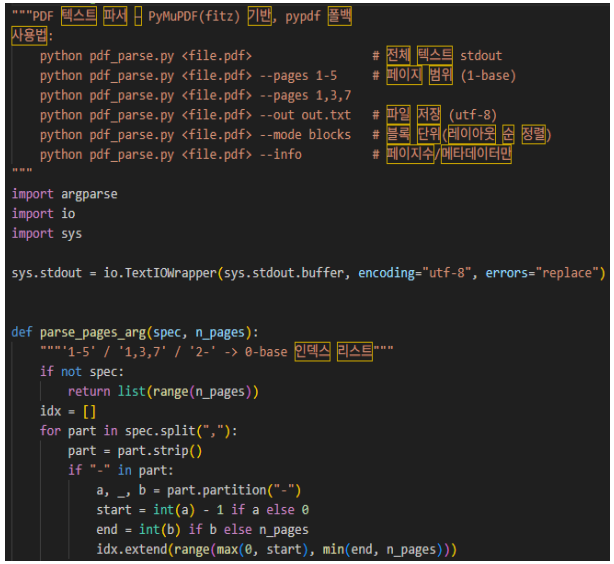
자료: 상상인증권 리서치센터

① 구동 예제 — KRX 지수 방법론 PDF 표 추출

- 앞 페이지의 지시 요소를 그대로 쓴 실행: 대상 PDF / 추출 항목 / 표 형식 / 저장 경로 / 제약(원문 근거만, 추측 금지)
- 도구 로그(파일 찾기→PDF 읽기→표 저장)가 곧 검수 지도 — 조항 번호를 달면 원문 추적 용이
- 끝난 뒤 원문 2~3건과 숫자·단위를 대조. 틀리면 페이지를 한정해 재지시
- 같은 흐름이 2회 이상이면 스킬로 고정. '지시→실행→검수→스킬' 표준 루프의 실물
- 아래: 실제 실행 기록 화면. 로그와 산출 경로를 같이 확인

그림 37. KRX 방법론 PDF 표 추출 실행 기록

PDF 업로드 → Skill 실행

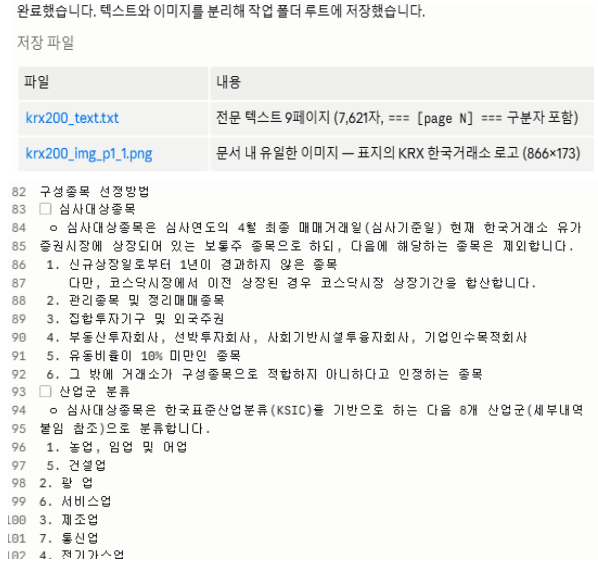


```
"""PDF 텍스트 파서 - PyMuPDF(fitz) 기반, pypdf 폴백
사용법:
    python pdf_parse.py <file.pdf>                # 전체 텍스트 stdout
    python pdf_parse.py <file.pdf> --pages 1-5    # 페이지 범위 (1-base)
    python pdf_parse.py <file.pdf> --pages 1,3,7
    python pdf_parse.py <file.pdf> --out out.txt  # 파일 저장 (utf-8)
    python pdf_parse.py <file.pdf> --mode blocks  # 블록 단위(레이아웃 순 정렬)
    python pdf_parse.py <file.pdf> --info         # 페이지수/메타데이터만
"""
import argparse
import io
import sys

sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding="utf-8", errors="replace")


def parse_pages_arg(spec, n_pages):
    """'1-5' / '1,3,7' / '2-' -> 0-base 인덱스 리스트"""
    if not spec:
        return list(range(n_pages))
    idx = []
    for part in spec.split(","):
        part = part.strip()
        if "-" in part:
            a, _, b = part.partition("-")
            start = int(a) - 1 if a else 0
            end = int(b) if b else n_pages
            idx.extend(range(max(0, start), min(end, n_pages)))
```

Python 스크립트 실행 → 작업 수행



완료했습니다. 텍스트와 이미지를 분리해 작업 폴더 루트에 저장했습니다.

저장 파일

파일	내용
krx200_text.txt	전문 텍스트 9페이지 (7,621자, === [page N] === 구분자 포함)
krx200_img_p1_1.png	문서 내 유일한 이미지 — 표지의 KRX 한국거래소 로고 (866×173)

```
82  구성종목 선정방법
83  □ 심사대상종목
84    ㅇ 심사대상종목은 심사연도의 4월 최종 매매거래일(심사기준일) 현재 한국거래소 유가
85  증권시장에 상장되어 있는 보통주 종목으로 하되, 다음에 해당하는 종목은 제외합니다.
86   1. 신규상장일로부터 1년이 경과하지 않은 종목
87      다만, 코스닥시장에서 이전 상장된 경우 코스닥시장 상장기간을 합산합니다.
88   2. 관리종목 및 정리매매종목
89   3. 집합투자기구 및 외국주권
90   4. 부동산투자회사, 선박투자회사, 사회기반시설투융자회사, 기업인수목적회사
91   5. 유동비율이 10% 미만인 종목
92   6. 그 밖에 거래소가 구성종목으로 적합하지 아니하다고 인정하는 종목
93  □ 산업군 분류
94    ㅇ 심사대상종목은 한국표준산업분류(KSIC)를 기반으로 하는 다음 8개 산업군(세부내역
95  붙임 참조)으로 분류합니다.
96   1. 농업, 임업 및 어업
97   5. 건설업
98  2. 광 업
99  6. 서비스업
100 3. 제조업
101 7. 통신업
102 4. 전기가스업
```

결과값 저장 → AI 모델이 추가 활용

자료: KRX, 상상인증권 리서치센터

① 구동 예제 — 리포트 PDF에서 컨센서스 추출

- 리포트 PDF 파싱/추출 지시는 – 1. [대상] 파일·커버리지, 2. [추출] 의견·목표가·직전, 3. [출력] 표 형식, 4. [저장] 경로, 5. [제약] 원문만 5개 라벨로 정형화
- 산출물 검수 순서 — ①표 행 수와 원문 종목 수 일치 ②임의 2행의 수치를 원문과 대조 ③원문 부재 항목의 '없음' 처리 점검 — 표본 대조로 전수 신뢰 확보
- 증권사 리포트뿐 아니라 공시·기관 보고서·논문에도 같은 골격을 그대로 적용 가능

그림 38. 추출 지시·산출 사례

파일	내용
skhynix_260130_text.txt	전문 텍스트 8페이지 (11,584자)
skhynix_260130_실적테이블.csv	표1. SK하이닉스 실적 Table — 20행 × 11컬럼 (분기 1Q25~4Q26E + 연간 2024/2025E/2026E), UTF-8 BOM이라 엑셀에서 바로 열림

검수 (원문 대조 3건)

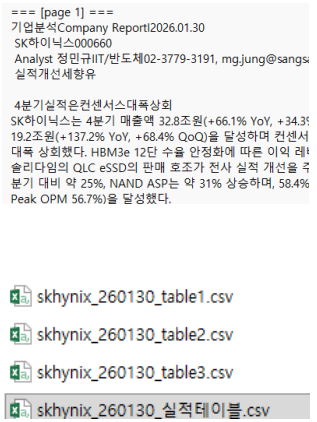
- 매출액 4Q25P 32,827십억원 — 본문 "4분기 매출액 32.8조원" 일치
- 영업이익 2026E 123,608십억원 — 본문 "2026년 영업이익 41.0% 상향" 문맥의 123.6조원 일치
- NAND 영업이익 1Q25 -65 / 2Q25 -142 → 3Q25 흑자전환 — 본문 "NAND YoY 흑자전환" 서술과 정합

테이블 핵심 (십억원)

항목	4Q25P	1Q26E	2025E	2026E
매출액	32,827	43,510	97,147	190,810
영업이익	19,170	27,512	47,206	123,608
OPM (%)	58.4	63.2	48.6	64.8
DRAM OPM (%)	67.4	73.1	59.8	74.4

자료: 상상인증권 리서치센터

그림 39. 리포트 텍스트 추출(.txt), 표 엑셀 추출(.csv)



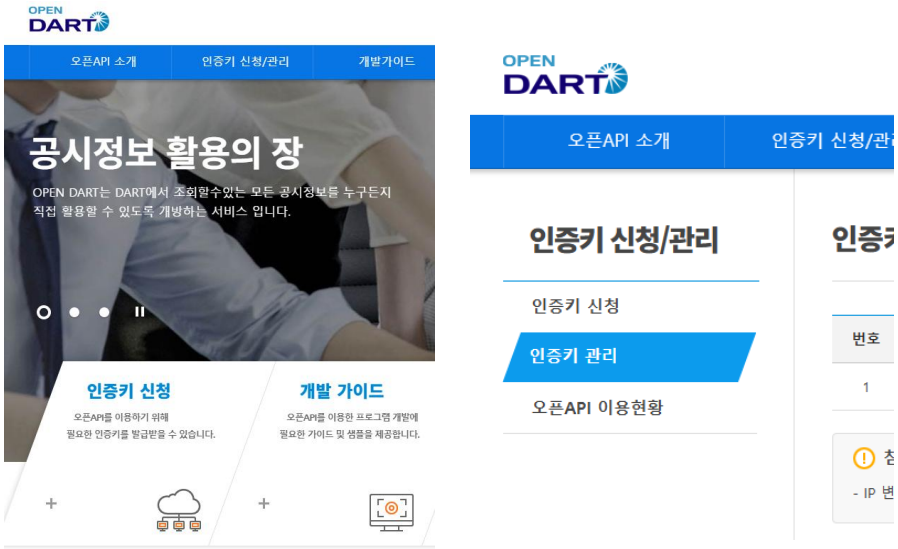
항목	1Q25	2Q25	3Q25	4Q25P
매출액	17,639	22,232	24,449	32,827
DRAM	14,037	17,124	19,084	24,948
NAND	3,229	4,728	5,024	7,550
기타	373	380	341	328
매출원가	7,537	10,249	10,420	10,250
매출총이익	10,102	11,983	14,029	22,577
GPM (%)	57.3	53.9	57.4	68.8
판관비	2,661	2,770	2,646	3,407
영업이익	7,441	9,213	11,383	19,170
DRAM	7,468	9,452	11,221	16,815
NAND	-65	-142	332	2,341
기타	37	-98	-170	14
OPM (%)	42.2	41.4	46.6	58.4
DRAM	53.2	55.2	58.8	67.4
NAND	-2	-3	6.6	31
기타	10	-25.7	-49.8	4.3
세전이익	9,299	8,723	14,790	17,653
당기순이익	8,108	6,996	12,598	15,246
지배주주순	8,107	6,997	12,595	15,243

자료: 상상인증권 리서치센터

② DART 공시/수출데이터/텔레그램 등 API 호출

- 사용법 3단계 — ①API 키 무료 발급 후 config에 1회 저장(DART·ECOS는 즉시) ② "최근 공시 업데이트", "수출 데이터 갱신" 수준의 자연어 지시 ③코드 작성·실행·표 정리는 AI가 수행
- 호출 결과를 로컬 DB에 증분 적재(신규분만 추가)하도록 한 번 만들면, 이후 스킬 호출 한 번으로 시계열 자동 유지 가능
- 활용은 조회를 넘어 상시 모니터링으로 확장 — 관심 종목 공시 알림, 수출 YoY 추이, 텔레그램 키워드 검색 등

그림 40. DART API Key 발급 이후 상시 활용 가능



자료: 금융감독원 Open DART, 상상인증권 리서치센터

그림 41. DART 공시·수출 데이터 호출 결과

수출 호출·축적	13 파이프라인 ecos.py → run_all.py (워터마크 증분, macro.duckdb)

관세청 HS세부 API의 500 오류(사람 경계)와 스로틀·증분 원칙([[pipeline-scale-incremental]])도 스킬에 명시해 함정을 승계했습니다.

2. 예시 실행 결과 (실측)

① DART API 호출 — SK하이닉스 2026-06-01~07-16 공시 35건 수신(status 000). 최근 흐름이 그 자체로 유의미합니다: 7/10 해외증권시장 주권 상장 결정·DR 발행·유상증자 → 7/13 주요사항보고서 → 7/15 발행결과·IR 개최, 즉 美 상장 유상증자 일정이 공시로 실시간 추적되는 상태입니다.

② 수출 축적분 조회 — macro.duckdb `macro_series` 424,179행 중 통관 수출금액(ECOS 901Y118.T002) 317개월(2000-01~2026-05) 축적:

월	수출(백만달러)	YoY%
2026-05	87,821	+53.4
2026-04	85,830	+47.9
2026-03	87,321	+50.4

자료: 상상인증권 리서치센터

③ 데이터 파이프라인 – 적재장소 구축 : Obsidian

- Obsidian은 링크로 정리되는 메모 앱(Evernote, Notion). API 호출이 편리하여 AI Agent 최적 앱
- 종목마다 노트를 두고 공시·뉴스·텔레그램·리포트 등 데이터 소스 라벨·날짜와 함께 타임라인으로 적재 가능
- 노트 속성을 통해 섹터·테마·최근 의견·중요도 등을 기준으로 종목 분류 가능
- 사용 예) "이 종목의 최근 한 달 공시를 정리해줘" 요청 → AI 에이전트가 Obsidian에 쌓인 타임라인을 요약, 데이터 소스에 따라 "텔레그램발 루머 제외, 공시만" 같은 선별 지시도 즉시 처리

그림 42. 종목 이슈 수집 – 처리

자료: 상상인증권 리서치센터

그림 43. 소스 라벨 타임라인

예시)

자료: 상상인증권 리서치센터

③ Obsidian — 그래프 연결 지식

- Obsidian은 데이터를 두뇌 신경망, 거미줄처럼 연결하는 시각화 & 구조화에 특화됨
- 예) [삼성전자]를 입력하면 관련 노트가 자동 연결, 표기 통일만으로 지식 구조 형성
- 역할 분담 — 본문(신호·공시·뉴스)은 에이전트가 자동 기록, 사람은 판단 메모만. 재생성되어도 사람 메모는 보존
- 그래프 뷰는 연결의 결과물 - AI Agent도 이 연결망을 읽어 관계를 파악

그림 44. 브레인·데일리

자료: 상상인증권 리서치센터

그림 45. Obsidian 그래프 뷰 예시

자료: Obsidian, 상상인증권 리서치센터

④ 기존 SQL 기반 DB와의 연계 – MySQL, MongoDB 등

- 운영 중인 사내 DB를 그대로 연결 —DB 경로·접속 정보만 알려주면 에이전트가 직접 조회해 근거로 사용
- SQL을 몰라도 무방 — "이 DB에서 ~를 조회해 정리해라" 지시만으로 조회문 작성·실행은 에이전트 수행 — 사내 DB 전체를 리서치 자료로 활용(방식은 표 6)
- 원본 무결성 유지 — 읽기 전용 계정만 부여해 수정·삭제 원천 차단, 조회 결과는 노트로만 축적

표 6. 데이터베이스 연계 방식

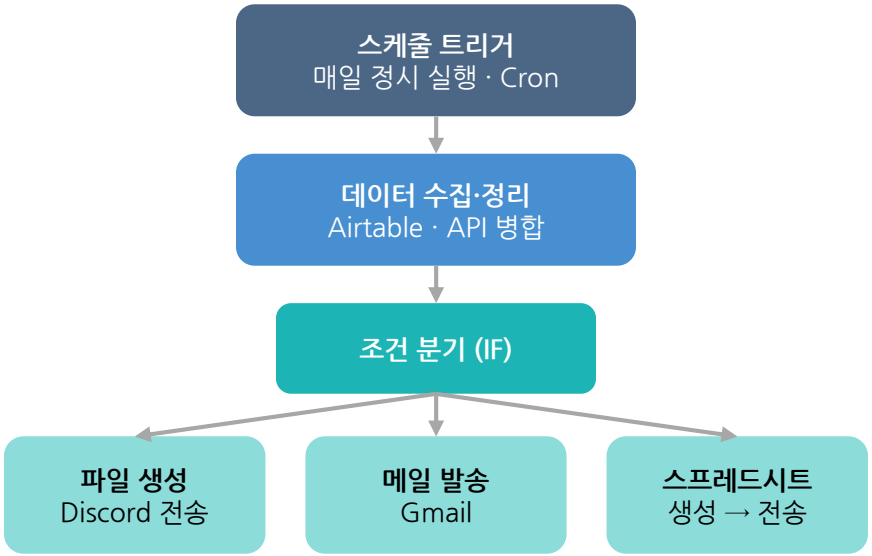
데이터베이스	데이터 종류 (예)	연결 방식
SQL (RDBMS)	MySQL·PostgreSQL — 정형 schema, 엑셀 데이터	에이전트가 조회문 작성·실행 — 읽기 전용 계정 권장
MongoDB (문서형)	뉴스·공시·텍스트 등 비정형 축적분	조회 결과를 노트 피드 근거로 기록
DuckDB (파일 분석)	로컬 시세·수급·지표 — 일일 체인의 수치 원장	체인이 집계 → 뷰 갱신의 입력으로 직결
자체 데이터	Excel·CSV·API — 부서 고유 데이터	경로·접속 정보만 제공하면 에이전트가 읽어 연계

자료: 상상인증권 리서치센터

⑤ 데일리 스케줄러 등록 – 반복은 되도록 LLM이 아닌 코드

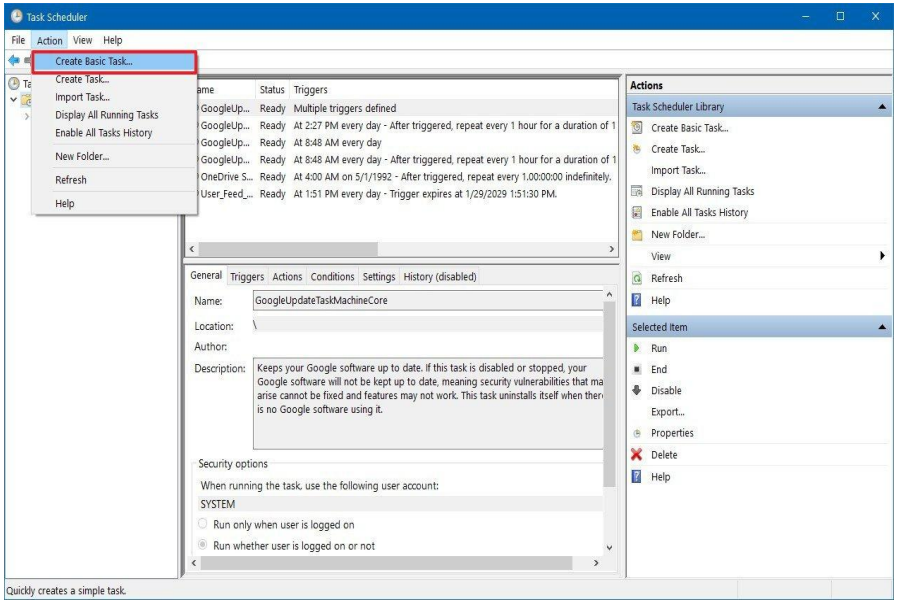
- 매일 작업은 반복 지시 대신 AI로 코드화 → 수집·정제·표 작성 절차를 스크립트로 한 번 만들면 이후에는 토큰 소모가 없고, 입력이 같으면 출력도 동일
- 실행은 Windows 작업 스케줄러 담당 — 원하는 시각에 등록하면 사람·AI 개입 없이 매일 자동 실행
- 사람은 로그·산출물 점검과 코드 수정 담당, AI는 해석에 집중

그림 46. 자동화 체인 사례



자료: 상상인증권 리서치센터

그림 47. 로컬 데이터 처리



자료: DuckDB, 상상인증권 리서치센터

Chapter 05

AI 시대의 향후 시사점

AI의 하루는 사람의 1년?

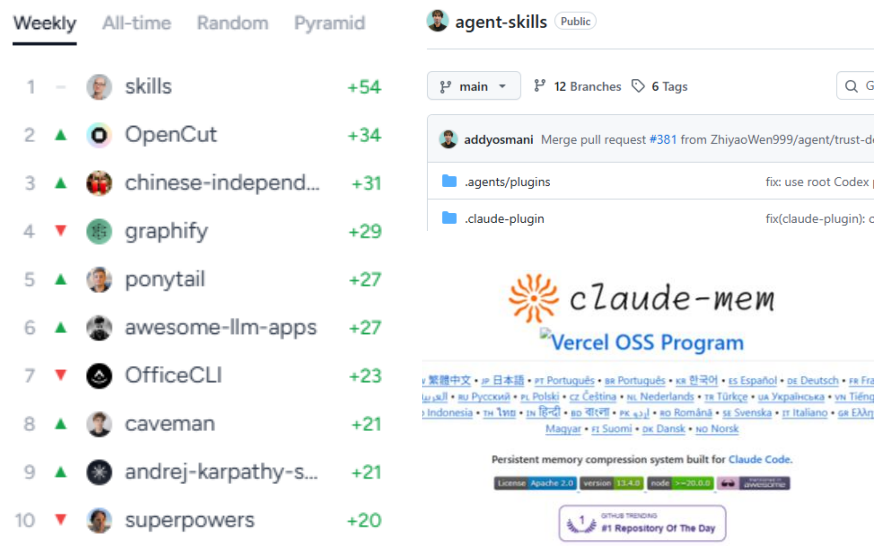
AI 사용법 빈부격차 시대 vs 기다리면 해결

- 지나치게 빠른 기술 업데이트 주기: 모델·하네스·스킬 문법이 수개월 단위로 바뀌어 매일 소식을 따라가도 전 영역 직접 검증은 곤란. 주요 업체가 오픈소스에서 검증된 기법을 자체 하네스로 흡수하는 구조가 반복 (GitHub 상위 Repository도 대부분 AI 활용 보조 기술)
- 정보의 소음: X·Threads·블로그 활용법 글 상당수는 해외 게시물의 무검증 번역·재유포라 실무 적용 가능성은 별개로 봐야 함. 특히 경량 하드웨어로 오픈소스 모델을 구동해 OpenAI, Anthropic 모델에 필적한다는 글은 각별히 유의
- 변하지 않는 축: 프런티어 모델에 제대로 지시만 하면 해내지 못하는 일이 거의 없다는 사실. 메모리(CLAUDE.md·스킬·노트)·자동화(스크립트·체인) 구축은 유행과 무관 — 본편에서 다룬 내용
- 결국 모두가 편리해지는 방향으로 갈 것인가?: Prompt Engineering의 중요도 하락, OpenClaw·Hermes 전용 기능의 메인 하네스(Codex·Claude Code) 흡수 속도를 볼 때, 복잡한 사용법 없이 누구나 사용하기 편한 형태로 진화할 것이라는 시각도 존재
- 다만 그때까지 AI 업무에 익숙한 사용자와 그렇지 않은 사용자의 “AI 빈부격차”는 유지되고, 워크플로우 최적화 격차는 더욱 심화될 것

문제점: 시작도 힘든데, 따라가기는 더 힘들다

- 생성형 AI 실무는 기술 주기가 매우 짧아 모델/하네스/스킬 문법이 수시로 변경. 주요 업체가 오픈소스 기법을 자체 하네스에 흡수하는 과정에서 실무에 유효한 것과 데모용이 섞여 들어옴
- 유튜브/Github/커뮤니티의 수많은 사용법 영상·후기 상당수는 시연 환경에서만 성립하거나 몇 달 뒤 문법이 바뀌어 재현 불가. 전부 따라가는 것은 불가능하며 그럴 필요도 없음
- 따라서 본편에 다룬 모델 선택/하네스/메모리/체인 구축이 우선 — 편의 기능은 그 위에 부착해도 늦지 않음

그림 48. Github 순위권 상위에는 AI Agent 애드온이 다수 포진



자료: GitHub, 상상인증권 리서치센터

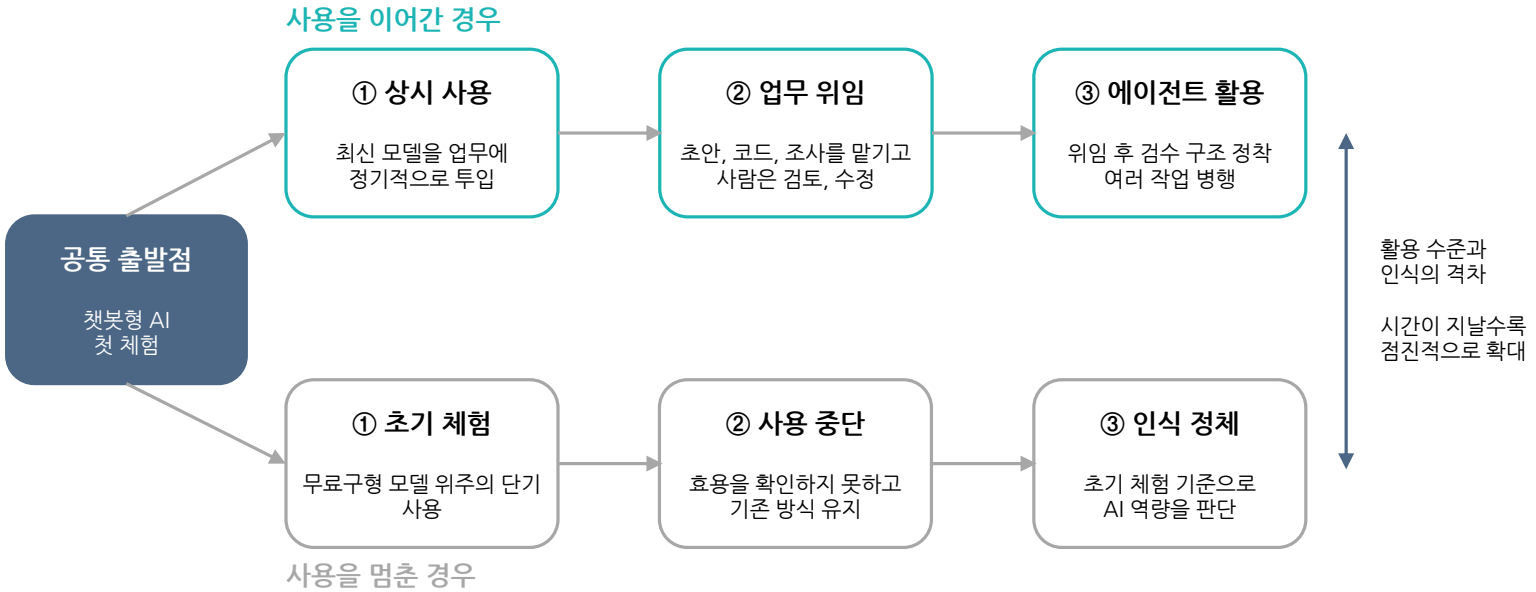
그림 49. 유튜브에 매주 쏟아지는 AI Agent 강의·후기

자료: YouTube, 상상인증권 리서치센터

그럼에도 반드시 가야 할 길이라면, 익숙해져야 한다

- 목표·입력·출력·제약이 명확한 작업에서 AI가 수행하지 못하는 영역은 거의 소멸 – 사무직 대체는 장기적으로 불가피
- 자동화는 확장 가능 — 한 업무에 익숙해지면 업무 전반으로 확산되고, AI를 관리하는 역량이 시간 대비 가치창출을 결정
- ①코딩·실행 자동화는 지시로 가능 ②API 연동이 수동 복사·붙여넣기 감소의 전제 ③확보된 시간은 분석·판단에 재배분
- 본 자료는 완결 매뉴얼이 아닌 시작점 — 방대한 AI 사용 체계를 전부 담기는 어려움. 실천으로 직접 감을 익히는 것이 미래 자산이 될 것

그림 50. Andrej Karpathy – AI 시대의 AI 활용도 빈부격차



자료: Andrej Karpathy, 상상인증권 리서치센터

Chapter 06

부록 : 용어사전, 참고자료, FAQ 등

① 시작 로드맵 — 4주

표 7. 4주 시작 로드맵

단계	할 일
1주차 환경 구축 (Ch.3)	① 이용 경로 선택 — 정액 구독(OAuth 로그인) 또는 API 종량제(키 발급·한도 설정) ② 전용 작업 폴더 생성 — 바탕화면이 아닌 전용 경로에 데이터·스크립트·CLAUDE.md 집결 ③ 첫 세팅 프롬프트 실행 — 유저 인터뷰 → 폴더 구조·CLAUDE.md 초안 → 테스트 1회 ④ 기본 문법 습득 — /clear·/compact·Esc, 접두 문자 #·@·!
2주차 PDF 추출과 검수 (Ch.4 ①)	① PDF 추출 1건 지시 — 대상·항목·형식·저장·제약(원문만) 5개 라벨 ② 긴 문서는 분할 지시 — 원문에 없는 칸은 공란 유지 ③ 원문 대조 검수 — 행 수 일치, 임의 2행의 수치·단위 대조 ④ 오류는 페이지 한정 재지시 — '지시→실행→검수' 루프 체득
3주차 반복의 자산화: 스킬·메모리 (Ch.3)	① 성공 직후 "방금 이 작업, 스킬로 만들어줘" 등록 ② description 3단 공식 — 트리거 문구·입력·산출물, 미호출 시 문구 추가 ③ 교정 직후 # 한 줄 등록 — Why와 How를 함께 기재 ④ 반복 횟수별 승격 — 1회 대화·2회 CLAUDE.md·3회+ 스킬, 오류 규칙 즉시 삭제
4주차~ 데이터 연결 & 자동화 (Ch.4 ②~⑤)	① DART 등 API 키 1회 등록 — 로컬 DB 증분 적재로 시계열 유지 ② 종목 노트 2~3개 개설 — 소스 라벨 타임라인·상단 속성 형식 고정 ③ 사내 DB 읽기 전용 연결 — 조회는 에이전트에 위임 ④ 반복 작업 코드화 후 스케줄러 등록

자료: 상상인증권 리서치센터

② 용어 미니 사전

표 8. 필수 용어 미니 사전

용어	설명	비고
토큰 (Token)	모델이 읽고 쓰는 최소 단위이자 과금 단위 단어·문장이 아닌 '서브워드(단어 조각)'인 경우도 있음	한국어는 영어보다 비효율적 보통 Output이 Input보다 5배 이상 비용
컨텍스트 윈도우 (Context Window)	한 번의 응답에서 참조 가능한 토큰 총량의 상한 지시문·대화·파일·도구 결과가 모두 포함 ('작업대 크기')	실무의 최대 병목 짧을수록 모델 성능 향상 길수록 Context Rot(성능 저하) 발생
컴팩션 / 트리밍 (압축)	컨텍스트가 차오르면 공간 확보 — 컴팩션 = 과거 대화를 요약으로 치환, 트리밍 = 단순 절삭	비가역적, 세부 내용부터 소멸 중요 결과는 메모리 파일로 저장 필요
KV(Key Value) 캐시	이미 처리한 토큰의 Key/Value를 저장해 재계산을 피하는 추론 최적화 장치 Attention Mechanism 비용 감소에 주효	메모리, 낸드 수요의 핵심 길수록 KV 캐시 HBM 포화
스킬 (Skill)	반복 절차를 문서로 패키징 — 관련 작업 감지 시 본문 지시문 대로 작동	컨텍스트 절약형 반복작업 설계 스킬명 수동 호출 가능 AI버전 업무 매뉴얼
서브 에이전트	본체가 위임한 작업을 별도 컨텍스트 윈도우에서 수행하는 하위 AI — 지시 작업 후 결과 요약만 반환	본체 컨텍스트 오염 방지 + Context Window 실사용 범위 확장 병렬 실행 가능
하네스 (Harness)	모델 실행 환경 일체 — 지시서·메모리·도구·권한·검증 Codex, Claude Code가 대표 사례	기존 Prompt Engineering 대체

자료: 상상인증권 리서치센터

③ Hermes 대화 내 명령어

- 대화 제어는 슬래시(/) 명령. Claude Code와 같은 습관을 유지하면 전환 부담이 작음
- 주제가 바뀌면 /new·/reset으로 창을 비울 것. 작업 창이 곧 비용이므로, 낡은 대화를 붙인 채 다음 주제로 넘어가지 않음
- 모델을 바꿔 비교할 때 /model, 고난도 종합 전 추론 강도를 높일 때 /reasoning, 장기 대화가 느려질 때 /compress
- 비용·사용량은 /usage·/insights, 터미널 보조는 hermes -c(이어하기)·dashboard(웹 화면)·insights(통계)
- 가정 실험은 /branch·/fork, 잘못된 지시 재개는 /undo·/retry. 아래 표에 '언제 어떤 명령'인지 정리

표 9. Hermes 대화 내 핵심 명령어

입력	기능	사용 시점
/new (=/reset)	신규 대화 시작 — Claude Code의 /clear에 해당	주제 전환 시
/model [name] --global	대화 또는 전역 기본 모델 전환	Claude·GPT·로컬 비교 시
/reasoning low~xhigh	reasoning effort(추론 강도) 조정 — Ch.1 ③의 실물	고난도 종합 작업 전 상향
/compress [주제]	context 수동 압축 — compaction 수동 실행	장기 대화 응답 지연 시
/usage · /insights	대화 토큰 사용량 · 기간별 통계	비용 모니터링
/branch (=/fork)	대화 분기 — 원본 보존 별도 실험	가정 변경 실험 시
/undo · /retry	직전 문답 취소 · 재전송	오발송 지시 재개

자료: 상상인증권 리서치센터

④ 일상 필수 문법 7개

- /clear: 주제 전환 시 대화·작업 창을 비움. 작업 창이 곧 비용이므로 '새 주제=비우기'
- /compact: 같은 작업이 길어져 느려질 때 대화를 요약으로 압축해 창을 가볍게. /model: 고난도 종합·의견 전에 상위 모델로(단순 추출은 최상위 불필요)
- Esc: 방향이 틀리면 즉시 중단. 잘못된 실행을 방치하는 비용이 더 큼. ↑: 직전 지시를 불러 일부만 고쳐 재실행
- Shift+Tab: 권한 모드(파일 조작 시 신중). /help: 기타 명령 조회. 아래 표에 입력·기능·사용 시점을 정리

표 10. Claude Code 필수 명령어·단축키

입력	기능	사용 시점
/clear	대화·컨텍스트를 비우고 새로 시작	주제 전환 시마다 — context는 곧 비용
/compact	지금까지 대화를 요약해 압축	같은 작업이 길어져 느려질 때
/model	모델 전환	고난도 종합 작업 전 상위 모델로
Esc	실행 중인 작업 중단	방향 오류 시 즉시 중단
↑ (위 화살표)	직전 지시 다시 불러오기	지시 일부 수정 후 재실행 시
Shift+Tab	권한 모드 전환(확인 생략 등)	반복 승인 생략 — 파일 조작 시 신중
/help	명령 목록·도움말	기타 명령 조회 시

자료: 상상인증권 리서치센터

⑤ 특수 문법 3개

- 메시지 맨 앞 글자 하나로 처리 방식이 바뀜. (# 이 메모리 관리의 핵심)
- # 메모: 교정 직후 규칙을 즉시 저장 — #을 생략하면 다음 대화에 같은 실수가 반복되기 쉬움
- @ 경로: 파일·폴더를 지시에 정확히 첨부(Tab 자동완성). 경로 오타로 엉뚱한 파일을 읽는 사고를 예방
- ! 명령: 모델 없이 터미널만 실행(토큰 0). 파일 목록·git 상태 같은 단순 확인에 유리
- Shift+Tab: 큰 변경 전 실행 없이 계획만

표 11. 접두 문자·단축키 특수 문법

입력	기능	사용 시점
# 메모	한 줄을 메모리에 저장 — 저장 위치(프로젝트/전역) 선택	교정 직후 즉시 규칙화 — 메모리 관리 기본
@ 경로	파일·폴더를 지시에 첨부 — Tab 자동완성	대상 파일을 경로 오류 없이 지정
! 명령	터미널 명령 직접 실행 — 모델 미경유(0토큰)	파일 목록·git 상태 등 단순 확인
Shift+Tab	권한 모드 순환 — 계획 모드 진입	대규모 변경 전 '실행 없이 계획만'
Esc Esc	대화를 이전 시점으로 되돌림(rewind)	잘못된 방향의 일괄 취소
Ctrl+R	축약 출력을 전체 로그로 전환	도구 호출·산출 경로 상세 확인

자료: 상상인증권 리서치센터

FAQ - 1

[Hermes 관련]

- Q1. Hermes, Openclaw를 사용하기 위해선 반드시 맥미니가 필요한가?
- A1. 필수는 아닙니다. 필요한 것은 에이전트를 상시 구동할 실행 환경이며, 일반 Windows PC로도 충분합니다. 맥미니 사례가 많은 것은 저전력·무소음이 24시간 구동에 적합하고, 초기 Hermes가 Linux 계통 OS만 지원해 사용법이 Mac 중심으로 형성된 영향입니다.

- Q2. Hermes, Openclaw 설치법 및 사용법이 복잡한데, 본 자료에서 다루지 않은 이유는?
- A2. 설치는 1회성 작업이라 공식 문서·AI 챗봇·유튜브 자료를 참고하는 편이 정확하기 때문입니다. 본 자료는 유행과 무관하게 유지되는 축인 지시문·메모리·검증·자동화 구조에 집중했습니다.

- Q3. 텔레그램으로 Hermes에 지시하면, 회사 자료가 외부 메신저 서버에 남는 것이 아닌가?
- A3. 텔레그램은 명목상 대화 내역을 볼 수 없지만, 지시·응답 텍스트가 메신저 서버를 경유하는 것은 구조상 불가피하므로, 미공개 자료·고객 정보는 메신저 경로에 올리지 않는 것이 원칙입니다.

FAQ - 2

[LLM 모델 관련]

- Q1. 월 비용은 어느 정도로 예상해야 하는가?
- A1. 정액 구독(월 $100 이상, 플랜별 상이)과 API 종량제 중 선택해야 합니다. 입문 단계는 정액 구독 하나로 하네스까지 바로 사용 가능한 경우가 많습니다. 한국어 위주 사용 시 영어 대비 1.5~2배의 토큰 배수를 감안해야 체감 비용에 가깝고, 반복 작업을 코드로 전환하면 같은 예산으로도 처리량이 수 배 차이 납니다.

- Q2. GLM, DeepSeek, Kimi 등 중국계 모델을 회사 업무에 사용해도 되는가?
- A2. 성능·가격상 일부 업무에는 충분하나, 데이터 처리 관할·약관 문제로 API 사용 시 데이터가 학습용으로 중국 기업에 유출될 가능성이 다분합니다. 미공개 정보 업무에는 보수적으로 접근하고, 공개 자료 가공·대량 단순 작업에 쓰는 것이 현실적 절충안입니다.

- Q3. 오픈소스 모델을 로컬 또는 클라우드 디바이스에서 구동하면 보안 문제가 없지 않은가?
- A3. 맥스튜디오·Spark 등 AI PC로도 Parameter가 큰 고급 모델은 정밀도를 FP2, FP4까지 낮춰야 하는데, 그 수준에서는 추론만 하더라도 성능이 크게 떨어져 실무 활용이 어렵습니다. 현재 RAM·NAND 가격 기준, 로컬 모델이 원활히 돌아갈 환경 구축에만 수천만원 이상을 감안해야 합니다.

FAQ - 3

[그 외]

- Q1. Claude Code, Codex 등은 개발자용 코딩 도구가 아닌가?
- A1. 명칭과 달리 실체는 파일을 읽고 쓰는 범용 에이전트로, 본편의 예시도 전부 비개발 업무입니다. 코드는 자동화를 저렴하게 하는 수단이며, 개발자 도구로 출발한 덕분에 파일·터미널·검증 도구가 가장 성숙하다는 점이 오히려 리서치/운용 업무에 유리합니다.

- Q2. MCP라는 용어를 자주 듣는데, 본 자료에서 다루지 않은 이유는?
- A2. MCP(Model Context Protocol)는 에이전트와 외부 도구(DB·캘린더·사내 시스템)를 연결하는 표준 규격으로, 본편 API 연동·DB 연계의 표준화 버전입니다. 입문 단계에서는 스킬·스크립트가 더 단순하고 관리가 쉬워 우선 배치했으며, 외부 도구가 늘어나면 자연스럽게 도달할 다음 학습 항목입니다.

* 다만 API 대비 연결 환경이 불안정해 데이터 호출 실패가 잦아 메인 작업용으로는 추천하지 않습니다.

Compliance Notice

- 본 자료에 기재된 내용들은 작성자 본인의 의견을 정확하게 반영하고 있으며 외부의 부당한 압력이나 간섭 없이 작성되었음을 확인합니다. (작성자: 김경태)
- 본 자료는 고객의 증권투자를 돕기 위한 정보제공을 목적으로 제작되었습니다. 본 자료에 수록된 내용은 당사 리서치센터가 신뢰할 만한 자료 및 정보를 바탕으로 작성한 것이나, 당사가 그 정확성이나 완전성을 보장할 수 없으므로 참고자료로만 활용하시기 바라며 유가증권 투자 시 투자자 자신의 판단과 책임하에 최종결정을 하시기 바랍니다. 따라서 본 자료는 어떠한 경우에도 고객의 증권투자 결과에 대한 법적 책임소재의 증빙자료로 사용될 수 없습니다.
- 본 자료는 당사의 저작물로서 모든 저작권은 당사에게 있으며 어떠한 경우에도 당사의 동의 없이 복제, 배포, 전송, 변형될 수 없습니다.
- 동 자료는 제공시점 현재 기관투자가 또는 제3자에게 사전 제공한 사실이 없습니다.
- 동 자료의 추천종목은 전일 기준 현재당사에서 1% 이상 보유하고 있지 않습니다.
- 동 자료의 추천종목은 전일 기준 현재 당사의 조사분석 담당자 및 그 배우자 등 관련자가 보유하고 있지 않습니다.
- 동 자료의 추천종목에 해당하는 회사는 당사와 계열회사 관계에 있지 않습니다.


상상인증권
sangsangin securities