**신영선의 AI탐구 · 296화**

# AI 코딩에 붙이는 오픈소스 7개

같은 AI라도 **어떤 방식으로 일하게 하느냐**에 따라
결과는 달라집니다.

계획하고, 단순하게 만들고, 화면을 다듬고,
반복 업무를 나만의 작업 방식으로 바꾸는 도구들입니다.

**핵심은 설치가 아니라 첫 실행입니다.**

하나만 골라 설치한 뒤, 작은 작업으로 실제 작동을 확인하세요.

## 목적별로 바로 찾기


01 **Superpowers**
일하는 순서를 설계하기

02 **Ponytail**
복잡하게 만들기 전에 줄이기

03 **Frontend Design**
화면의 시각적 완성도 높이기

04 **Caveman**
답변을 짧고 핵심적으로 받기

05 **Skill Creator**
반복 업무를 나만의 스킬로 만들기

06 **Code Simplifier**
동작은 유지하고 코드를 정리하기

07 **Ralph Loop**
완료 조건까지 수정·검증 반복하기

끝 **실전 조합과 확인표**
도구를 겹치지 않게 쓰는 방법


CLI 도구·MCP 서버를 제외한 스킬·플러그인 선정입니다.
실제 설치·활성화한 목록에서 골랐으며, 사용 빈도·인기·선호도 순위는 아닙니다.

기본 환경: Claude Code  Codex 호환: 도구별 안내  소스 라이선스 ≠ 모델 무료 사용

**빠른 시작**

# 한 번에 7개 말고, 필요한 1개부터.

아래의 설치 명령은 일반 ChatGPT 대화창이나 운영체제 터미널이 아니라, **Claude Code 대화창**에서 실행하는 명령입니다.




### 1 작업용 프로젝트에서 Claude Code 열기

로그인한 로컬 Claude Code를 준비합니다. 일반 claude.ai 채팅·Codex는 설치 방식이 다릅니다. 원본 프로젝트는 커밋하거나 복사본을 만드세요.

### 2 공식 마켓플레이스 확인

대부분의 대화형 Claude Code에서는 기본 등록됩니다. 공식 마켓플레이스가 없다는 오류가 날 때만 아래 명령을 실행하세요.

```
/plugin marketplace add anthropics/claude-plugins-official
```

### 3 도구별 페이지의 설치 명령 실행

Ponytail·Caveman은 각 제작자의 마켓플레이스를 먼저 추가합니다. 출처와 권한을 확인한 뒤 필요한 도구만 설치하세요.

### 4 목록 확인 후 새 세션에서 첫 요청

```
/plugin
```

Installed 탭에서 설치·활성화 상태를 확인합니다. 적용이 안 되면 새 세션을 열거나 공식 안내에 따라 /reload-plugins를 사용합니다. 설치된 것과 실제 실행된 것은 다릅니다.

### 처음이라면

화면 만들기 → Frontend Design
설명 줄이기 → Caveman
큰 개발 작업 → Superpowers

### 익숙해졌다면

반복 업무 표준화 → Skill Creator
코드 정리 → Code Simplifier
한도가 있는 반복 검증 → Ralph Loop

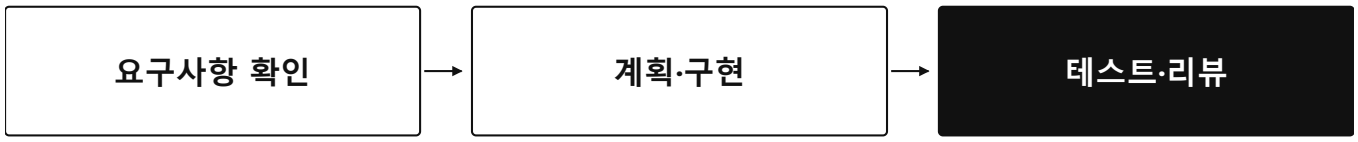
> **비용과 안전**
> 오픈소스는 코드의 이용 조건을 말합니다. Claude·Codex 구독이나 API 사용료는 별도입니다. 훅·에이전트가 추가 실행되면 사용량도 늘어날 수 있습니다. 이 문서 제작 과정에서 새 설치나 실행 테스트를 수행한 것은 아닙니다.

**01 · 일하는 순서를 설계하기**

# Superpowers

바로 코딩하기 전에 요구사항을 정리하고, 계획·구현·테스트·리뷰로 이어지는 개발 절차를 적용합니다.

MIT | Claude Code · Codex 공식 설치 경로 제공




| 사용 전 | 기대 변화 |
| --- | --- |
| **요청 직후 코드부터 작성** | **요구사항 합의 후 단계별 구현** |

## 설치

아래 명령은 Claude Code 대화창에서 한 줄씩 실행합니다.

```
/plugin install superpowers@claude-plugins-official
```

## 첫 요청은 이렇게

> 예약 관리 화면을 만들고 싶어. 먼저 필요한 기능을 질문하고, 설계를 내가 확인한 뒤 구현 계획을 세워줘.

## 작동 확인

코드부터 작성하지 않고 요구사항·설계 확인이 먼저 나오는지 봅니다.

> **알아둘 점**
> 큰 작업에 적합합니다. 간단한 수정에는 전체 절차가 부담이 될 수 있으니 범위를 먼저 좁히세요.

**Codex에서 쓰려면**

Codex의 플러그인 목록에서 Superpowers를 검색해 설치합니다. Claude Code와는 별도로 설치해야 합니다.

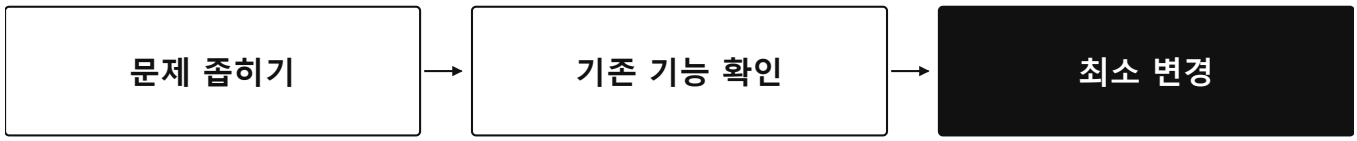
**공식 저장소** github.com/obra/superpowers

**02 · 복잡하게 만들기 전에 줄이기**

# Ponytail

이미 있는 기능과 가장 단순한 해결책부터 검토하게 해, 불필요한 추상화와 코드를 줄이도록 돕습니다.

MIT | Claude Code · Codex 공식 설치 경로 제공




| 사용 전 | 기대 변화 |
| --- | --- |
| **새 라이브러리·복잡한 구조 추가** | **기존 기능 재사용·최소 변경** |

## 설치

아래 명령은 Claude Code 대화창에서 한 줄씩 실행합니다.

```
/plugin marketplace add DietrichGebert/ponytail
/plugin install ponytail@ponytail
```

## 첫 요청은 이렇게

> 이 기능을 가장 단순하게 해결해줘. 새 도구를 추가하기 전에 기존 코드로 해결 가능한지 보고, 안 해도 되는 작업도 알려줘.

## 작동 확인

새 구조를 만들기 전에 기존 기능 재사용·작업 생략 가능성을 검토하는지 봅니다.

> **알아둘 점**
> 짧은 코드 자체가 목표는 아닙니다. 입력 검증·오류 처리·보안·접근성은 생략하지 않습니다.

**Codex에서 쓰려면**

공식 README에 Codex 플러그인 설치 경로가 있습니다. Codex용 설치·훅 신뢰 절차를 별도로 따르세요.

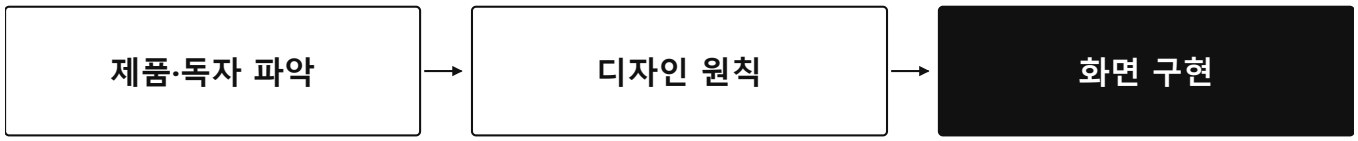
**공식 저장소** github.com/DietrichGebert/ponytail

**03 · 화면의 시각적 완성도 높이기**

# Frontend Design

제품의 목적에 맞춰 폰트·색상·레이아웃을 의도적으로 선택하게 해, 비슷비슷한 AI 화면을 줄입니다.

Apache-2.0 | Claude Code 플러그인 · Codex는 스킬 방식




| 사용 전 | 기대 변화 |
| --- | --- |
| **기본 템플릿을 그대로 반복** | **제품에 맞는 화면 원칙 적용** |

## 설치

아래 명령은 Claude Code 대화창에서 한 줄씩 실행합니다.

```
/plugin install frontend-design@claude-plugins-official
```

## 첫 요청은 이렇게

> 독서 기록 서비스의 첫 화면을 만들어줘. 흑백 톤, 넉넉한 여백, 읽기 쉬운 한글 폰트로 디자인 방향부터 제안하고 모바일에서도 확인해줘.

## 작동 확인

제품 목적에 맞는 디자인 방향을 설명하고, 폰트·여백·계층을 일관되게 적용하는지 봅니다.

> **알아둘 점**
> 이 스킬은 디자인 지침입니다. 실제 브라우저 테스트 도구가 자동으로 추가되는 것은 아닙니다.

**Codex에서 쓰려면**

스킬 설치 도구에서 anthropics/skills의 frontend-design을 선택해 Codex에 설치합니다. Claude 전용 플러그인 명령은 그대로 쓰지 않습니다.

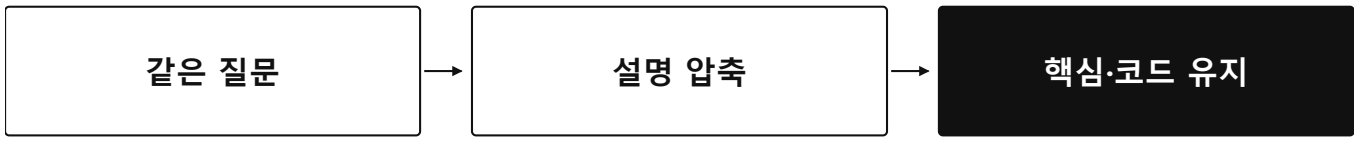
**공식 저장소** github.com/anthropics/skills/tree/main/skills/frontend-design

**04 · 답변을 짧고 핵심적으로 받기**

# Caveman

코드·명령어·오류 메시지는 유지하면서 불필요하게 긴 설명을 줄이는 커뮤니케이션 스킬입니다.

Apache-2.0 (현재 저장소) | Claude Code 플러그인 · Codex 스킬 경로 제공




| 사용 전 | 기대 변화 |
| --- | --- |
| **장황한 작업 설명** | **결론·변경·미확인만 간결하게** |

## 설치

아래 명령은 Claude Code 대화창에서 한 줄씩 실행합니다.

```
/plugin marketplace add JuliusBrussee/caveman
/plugin install caveman@caveman
```

## 첫 요청은 이렇게

> caveman 모드로 답해줘. 결론, 바꾼 것, 확인하지 못한 것만 짧게 알려주되 코드와 오류 메시지는 원문을 유지해줘.

## 작동 확인

답변이 짧아져도 부정·숫자·위험 안내·실행 결과가 빠지지 않는지 봅니다.

> **알아둘 점**
> 이 가이드는 답변 압축 스킬만 다룹니다. 최신 버전의 프록시·자동 모델 선택·클라우드 기능은 별도 기능이며 여기서 설정하지 않습니다.

**Codex에서 쓰려면**

공식 README의 "스킬만 설치" 경로로 Codex를 선택합니다. Codex에서는 $caveman 호출 방식이 안내되어 있습니다.

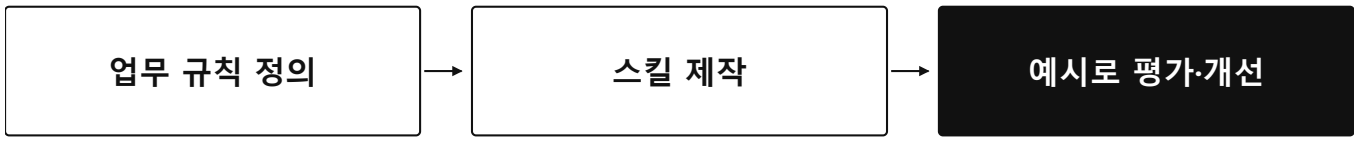
**공식 저장소** github.com/JuliusBrussee/caveman

**05 · 반복 업무를 나만의 스킬로 만들기**

# Skill Creator

자주 하는 업무의 목적·입력·결과 형식을 정리해 스킬로 만들고, 예시 과제로 테스트·개선합니다.

Apache-2.0 | Claude Code 플러그인 · Codex는 스킬 방식




| 사용 전 | 기대 변화 |
| --- | --- |
| **매번 같은 지시를 길게 반복** | **재사용 가능한 업무 절차** |

## 설치

아래 명령은 Claude Code 대화창에서 한 줄씩 실행합니다.

```
/plugin install skill-creator@claude-plugins-official
```

## 첫 요청은 이렇게

> 매주 회의록에서 결정사항과 할 일을 추출하는 스킬을 만들어줘. 입력 예시, 출력 형식, 사람이 확인할 조건부터 묻고 테스트 예시도 만들어줘.

## 작동 확인

스킬 파일만 만드는 데서 끝나지 않고, 예시 입력·기대 결과·테스트 기준까지 만드는지 봅니다.

> **알아둘 점**
> 스킬은 업무 지침이지 앱 연결 그 자체가 아닙니다. 파일·메일·서비스 권한은 별도로 필요합니다.

**Codex에서 쓰려면**

anthropics/skills의 skill-creator를 Codex용 스킬로 설치할 수 있습니다. 도구 호출·평가 환경 차이는 실제 실행으로 확인하세요.

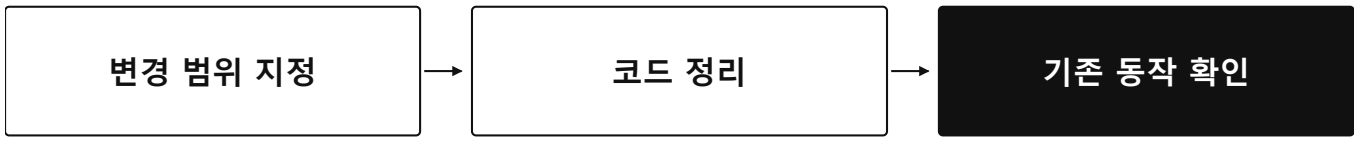
**공식 저장소** github.com/anthropics/skills/tree/main/skills/skill-creator

**06 · 동작은 유지하고 코드를 정리하기**

# Code Simplifier

최근 수정한 코드의 중복·불필요한 중첩·불명확한 이름을 정리해 읽기 쉽고 유지보수하기 좋게 만듭니다.

Apache-2.0 | 이 가이드의 설치 경로는 Claude Code 전용




| 사용 전 | 기대 변화 |
| --- | --- |
| **동작하지만 복잡한 코드** | **같은 동작·명확한 구조** |

## 설치

아래 명령은 Claude Code 대화창에서 한 줄씩 실행합니다.

```
/plugin install code-simplifier@claude-plugins-official
```

## 첫 요청은 이렇게

> code-simplifier로 방금 수정한 파일만 정리해줘. 기능과 외부 동작은 바꾸지 말고, 정리 전후 테스트 결과와 미확인 부분을 알려줘.

## 작동 확인

변경 범위를 제한하고, 리팩터링 전후 기존 기능·테스트 결과가 유지되는지 봅니다.

> **알아둘 점**
> 자동 정리도 코드 변경입니다. 커밋이나 백업 후 실행하고 테스트를 확인하세요. 짧게 만드는 것보다 읽기 쉬운 코드가 우선입니다.

**Codex에서 쓰려면**

Claude Code 플러그인·에이전트 설치법을 Codex 공통 명령으로 소개하지 않습니다. 이 문서는 Codex에서의 동일 동작을 보장하지 않습니다.

**공식 저장소** github.com/anthropics/claude-plugins-official/tree/main/plugins/code-simplifier

**07 · 완료 조건까지 수정·검증 반복하기**

# Ralph Loop

같은 작업 지시를 반복하며 이전 결과를 개선하도록 돕습니다. 테스트처럼 완료 여부가 명확한 작업에 적합합니다.

Apache-2.0 | 이 가이드의 설치 경로는 Claude Code 전용




| 사용 전 | 기대 변화 |
| --- | --- |
| **한 번 수정하고 완료 선언** | **지정한 한도 안에서 반복 검증** |

## 설치

아래 명령은 Claude Code 대화창에서 한 줄씩 실행합니다.

```
/plugin install ralph-loop@claude-plugins-official
```

## 한도를 둔 첫 실행 예시

/ralph-loop "샘플 프로젝트의 실패한 단위 테스트만 수정해. 기능을 추가하지 말고, 전체 테스트가 통과하면 <promise>COMPLETE</promise>를 출력해." --completion-promise "COMPLETE" --max-iterations 5

## 작동 확인

최대 반복 횟수와 완료 조건을 먼저 지정합니다. 결과는 실제 테스트 로그로 확인합니다.

> **알아둘 점**
> 모델 사용량이 반복마다 늘어납니다. 무제한으로 켜지 마세요. 중단 명령은 /cancel-ralph입니다. Windows에서는 Git Bash 등 공식 안내의 실행 환경을 확인하세요.

**Codex에서 쓰려면**

이 가이드의 Ralph Loop는 Claude Code의 Stop 훅 기반 플러그인입니다. Codex에 같은 명령을 그대로 적용하지 않습니다.

**공식 저장소** github.com/anthropics/claude-plugins-official/tree/main/plugins/ralph-loop

**마지막 확인**

# 모두 켜기보다,
# 단계를 나누세요.

7개를 한꺼번에 작동시키는 것이 목표가 아닙니다. 계획·구현·검증·정리를 구분하면 중복 작업과 불필요한 반복을 줄일 수 있습니다.

01 **큰 틀은 Superpowers**

요구사항을 정리하고 구현 계획에 동의한 뒤 시작합니다. 이미 검증된 계획이 있다면 과정을 다시 길게 반복할 필요는 없습니다.

02 **구현은 Ponytail · 화면은 Frontend Design**

기존 기능을 우선 활용하고 변경을 작게 유지합니다. 화면 작업일 때만 디자인 원칙을 함께 적용합니다.

03 **필요할 때만 Ralph Loop · 마무리는 Code Simplifier**

테스트 통과처럼 명확한 목표가 있는 작업에만 반복 한도를 둡니다. 정리 후에는 테스트를 다시 확인하고, Superpowers의 반복 절차와 무작정 중첩하지 않습니다.

04 **전 과정은 Caveman · 재사용은 Skill Creator**

핵심 보고는 짧게 받되 오류·위험·미확인은 남깁니다. 자주 반복되는 작업만 규칙을 정리해 스킬로 만듭니다.

## 설치 후 확인표

- [ ] 선택한 도구의 공식 저장소·라이선스·호환 환경을 확인했다.
- [ ] 설치 목록에 보이는 것뿐 아니라 활성화 여부도 확인했다.
- [ ] 작은 샘플 작업에서 해당 스킬·에이전트가 사용되는지 확인했다.
- [ ] 코드 변경 전후 테스트와 실제 결과를 확인했다.
- [ ] 실패할 때 중단할 조건과 사람이 승인할 작업을 정했다.

### 반드시 사람이 승인할 것

유료 API·결제, 배포, 외부 게시·메시지 전송, 삭제와 권한 변경. 설치가 이런 작업의 자동 승인을 뜻하지 않습니다.

### 문제가 생겼다면

/plugin의 Installed에서 해당 도구를 비활성화하고 새 세션으로 확인하세요. Ralph Loop가 실행 중이면 /cancel-ralph로 중단합니다.

**문서 기준**

2026.10.27 · 공식 README·라이선스와 로컬 설치·활성화 기록을 기준으로 정리했습니다. 독자 PC에서 7개를 새로 설치·실행 검증한 결과는 아닙니다. 명령·버전·제공 방식은 변경될 수 있어 설치 시 각 페이지의 공식 저장소를 다시 확인하세요.

Claude Code 플러그인 공식 안내 Codex용 스킬 설치 도구

더 많은 AI 활용법 · 신영선의 AI탐구 @ai.yeongseon