

GPT-6 Astra를 10배 더 잘 쓰는 법

초보자를 위한 실전 가이드 · 끝까지 말하기 · 복붙 프롬프트 · 쉬운 설정

오늘의 한 줄 의도를 파악해 허락된 범위에서 끝까지 진행해줘. 확인이 필요한 단계는 내가 검토할 결과를 먼저 만든 뒤 물어봐.

먼저 알아둘 것

이 책의 '10배'는 성능 보장 숫자가 아니라, 같은 일을 덜 고치고 덜 되문게 만드는 활용 목표다. 프롬프트는 AI에게 주는 요청문이다. 어려운 명령어 없이 평소 말로 시작하면 된다. 예시는 그대로 붙여 넣고 대괄호 안만 내 업무로 바꾼다.

기준 자료: OpenAI 공식 GPT-6 Astra 안내 및 스칼·프롬프트 점검 글. 아래 한국어 요청문은 일상 업무에 맞게 풀어 쓴 활용 예시다.

이 문서의 사용법

1. 처음이라면 2장의 시작 문장을 대화창에 붙여 넣는다.
2. 3장에서 자꾸 멈추는 이유와 요청을 고치는 방법을 익힌다.
3. 4장에서 내 업무에 맞는 템플릿 하나를 골라 사용한다.
4. 프로그램에 AI를 연결하는 사람만 5장의 API 설정을 읽는다.
5. 7장의 비교표와 9장의 30분 실습으로 실제 효과를 확인한다.

1. GPT-6 Astra, 무엇이 달라졌나

Astra는 문서 작성, 자료 조사, 코딩처럼 여러 단계를 거치는 작업에 강한 모델이다. 다만 결과를 크게 바꿀 선택은 질문할 수 있다. 일을 맡길 때 완성할 결과와 스스로 결정해도 되는 범위를 함께 적으면 도움이 된다. [1]

사용 상황	Astra에 맡길 일	시작할 때 정할 것
글쓰기·정리	읽고 바로 쓸 초안	독자, 분량, 꼭 넣을 내용
조사·비교	근거 있는 선택안	비교 대상, 기준, 최신성
코딩·화면 제작	동작하는 수정 결과	수정 범위와 완료 기준
앱·브라우저 작업	연결된 도구로 실행	접근 권한과 승인할 단계

새 기능과 변화를 쉬운 말로

- 비동기 도구 호출 — 도구가 일하는 동안 다른 독립적인 작업을 이어갈 수 있다. 앱의 지원이 필요하다.
- 작업 중 방향 수정 — 진행 도중 추가 요청을 반영할 수 있다. API에서는 이를 위한 연결 방식이 있다.
- 대화 중 추론 조절 — 어려운 구간에 더 생각하게 하고, 쉬운 후속 작업에서는 줄일 수 있다.
- 강해진 지시 준수 — 예전 스킬이나 작업 규칙도 더 민감하게 따를 수 있어 충돌을 정리하면 좋다.
- 비용은 업무당 비교 — 일부 평가에서는 출력 토큰이 줄어 예상 비용이 낮았다. 모든 업무의 절약을 보장하지는 않는다.
- 답변 길이는 별도 지정 — 토큰 효율과 화면에 보이는 글 길이는 다르다. 짧게 원하면 분량을 요청한다. [1]

2. 60초 빠른 시작

지금 하려는 일	이렇게 요청하기	완료 기준 붙이기
회의록 정리	결과와 할 일 정리	담당·기한 없으면 미정
보고서 작성	팀장용 1쪽 초안	근거와 제안 포함
상품·서비스 비교	같은 기준으로 비교	출처와 확인 시점 포함
메일 초안 작성	정중하게 5문장	제목 포함, 발송 전 확인
코드 오류 수정	원인을 찾고 수정	관련 동작 확인까지
아직 목표가 모호함	가능한 방향 2개 제안	중요한 선택 1개만 질문

복사할 시작 문장 내 의도와 대화 맥락을 파악해 허락된 범위에서 끝까지 진행해줘. 사소한 선택은 판단해서 처리하고, 승인할 단계가 있으면 검토 가능한 결과부터 만들어줘.

어디에 넣고, 어떻게 쓰나요?

- 한 번만 쓸 때 — 새 대화의 첫 줄에 시작 문장을 넣고, 아래에 실제 할 일을 적는다.
- 반복해서 쓸 때 — 이용 중인 앱의 맞춤 지침이나 프로젝트 지침이 있다면 공통 규칙만 저장한다.
- 이번 일의 정보 — 독자, 자료, 기한, 분량은 매 요청에 적는다. 저장된 지침과 충돌하면 이번 기준을 명시한다.
- 도구가 필요한 일 — 파일을 읽거나 앱을 수정하려면 해당 기능과 권한이 있어야 한다. 요청문만으로 연결되지는 않는다.
- 예시 — ‘아래 회의 메모를 팀원용으로 정리해줘. 결정 사항과 담당별 할 일을 포함하면 완료야.’

3. 끝까지 완성시키는 10가지 원칙

01 질문을 줄일 범위를 먼저 정한다

‘알아서 해줘’에 범위를 붙인다. 예: ‘제목·소제목·문장 순서는 판단해서 정해줘. 예산이나 제출 대상이 달라지면 물어봐.’ 질문을 없애기보다 필요한 질문만 남기는 방식이다.

02 ‘할 수 있어?’도 실행 요청이라고 적는다

예: ‘내가 할 수 있어, 도와줘, 하고 싶어라고 말하면 실제 작업을 부탁하는 뜻이야. 가능하다는 답이나 계획만 주지 말고 요청한 결과까지 만들어줘.’ 단순한 기능 질문이라면 그 의도를 따로 밝힌다.

03 승인은 결과를 보고 하게 만든다

메일이라면 제목·본문·첨부 후보까지 준비한 뒤 발송 여부를 묻게 한다. 예: ‘초안 작성은 바로 진행해줘. 발송 승인은 완성된 내용을 보여준 다음 받아줘.’ 승인 대기와 초안 작업을 분리하면 덜 멈춘다.

04 허락한 일과 확인할 일을 구체화한다

예: ‘자료 읽기와 복사본 편집은 진행해줘. 영구 삭제·결제·외부 발송은 먼저 확인해줘.’ 이미 허락한 단계는 되묻지 않게 한다. 필요한 자료나 핵심 결정이 없으면 해당 부분만 질문하게 한다.

05 파일 지시가 충돌하면 근거를 받는다

스킬은 반복 업무 설명서, **AGENTS.md**는 코드 작업 규칙 파일이다. 예: ‘스킬의 작업 지침과 내 명시적 요청이 충돌하면 내 요청을 우선해줘. 멈추면 파일명과 해당 문장을 알려줘.’ 서비스의 상위 규칙이나 실제 권한까지 바꾸는 지시는 아니다. [2]

06 글투는 원하는 예로 정한다

예: ‘한 문단에 생각 하나씩, 쉬운 말로 써줘. 표는 비교할 때, 목록은 항목을 나란히 볼 때만 써줘. 상투적인 결론 문구와 불필요한 X가 아니라 Y 식 표현은 줄여줘.’ 문서 서식이 정해져 있다면 그 형식을 우선한다.

07 검증은 변경한 만큼 한다

버튼 글자 하나를 바꿨다면 해당 화면부터 확인한다. 예: ‘작고 되돌릴 수 있는 변경에는 불필요한 새 테스트를 만들지 말고, 필요한 확인만 해줘. 통과한 검사는 새 문제 없으면 반복하지 마.’ 기능 변경과 오류 수정에는 관련 검증이 필요하다.

08 끝나는 모습을 눈에 보이게 적는다

‘자료 정리’보다 ‘비교표 1개와 추천 이유 3문장’이 명확하다. 예: ‘요청 항목을 모두 담고, 수치와 링크를 확인하고, 바로 쓸 수 있는 문서를 주면 완료야.’ 시간이나 분량 때문에 필수 항목을 임의로 빼지 않게 한다.

09 진행 중 수정은 바뀐 점부터 알린다

예: ‘기존 조사 내용은 활용하되 대상 독자를 신입사원으로 바꿔줘. 최종 목표는 1쪽 안내문이야.’ 중간 질문을 하더라도 기존 목표가 이어지는지 밝혀주면 작업 흐름을 유지하기 쉽다.

10 실제로 덜 고쳤는지 비교한다

같은 자료로 기존 요청과 새 요청을 각각 실행해본다. 완성도, 재질문 횟수, 내가 고친 시간부터 비교한다. 빠르게 끝났어도 필수 내용이 빠지면 개선으로 보지 않는다.

좋은 요청문은 이 다섯 칸이면 충분하다

할 일: [무엇을 완성할지]
대상·용도: [누가 어디에 쓸지]
입력 자료: [파일·본문·링크]
진행 범위: [알아서 할 일 / 확인할 일]
완료 기준: [형식·분량·필수 항목]

예시
아래 메모로 신입사원 안내문을 완성해줘.
친절한 말투로 1쪽, 준비물과 첫날 순서를 포함해줘.
소재목과 순서는 알아서 정하고, 없는 장소·시간은 미정으로 써줘.

초보자 포인트 설명을 길게 쓸 필요는 없다. ‘무엇을 받으면 끝인지’와 ‘어디까지 말기인지’를 먼저 정한다. 꼭 필요한 정보가 빠졌다면 먼저 묻는 편이 더 빠를 수도 있다.

4. 바로 복사해 쓰는 프롬프트

A. 범용 작업 — 끝까지 맡기기

[할 일]을 [대상 독자]가 [사용 목적]에 바로 쓸 수 있게 완성해줘.
입력 자료: [자료]
결과 형식: [문서·표·파일 / 분량]
필수 항목: [꼭 넣을 내용]

내 의도를 파악해 허락된 범위에서 끝까지 진행해줘.
‘도와줘’, ‘할 수 있어?’도 실제 작업 요청으로 받아줘.
가능하다는 답이나 계획만 내놓고 끝내지 마.
사소한 구성은 판단해서 정하고, 사용한 가정은 짧게 밝혀줘.
확인이 필요한 단계: [예: 발송·결제]
그 전까지 가능한 준비를 끝내고 결과를 보여준 뒤 물어봐.
마지막에는 완성한 결과와 아직 확인이 필요한 점을 알려줘.

B. 분석·의사결정 — 같은 기준으로 비교

[선택지 A, B, C] 중 [목적]에 맞는 안을 비교해줘.
중요한 기준: [예산·시간·기능 등]
자료: [파일이나 링크]

같은 기준을 열로 둔 비교표 1개를 만들고,
추천 이유를 초보자도 이해할 3문장으로 적어줘.
공개된 사실과 네 판단을 구분해줘.
자료에 없는 가격·수치는 만들지 말고 확인 필요로 표시해줘.
기준의 비중은 합리적으로 제안해도 돼.
선택이 완전히 달라질 핵심 정보만 질문하고, 나머지 비교는 진행해줘.

C. 코딩·수정 — 필요한 검증까지

문제: [어떤 상황에서 무엇이 잘못되는지]
수정 범위: [파일·화면·기능]
완료 기준: [기대하는 동작]

관련 코드를 읽고 원인을 찾아 필요한 수정을 해줘.
계획만 제시하지 말고 수정과 관련 동작 확인까지 진행해줘.
작고 되돌릴 수 있는 변경에 불필요한 새 테스트를 만들지 마.
기능 변경·오류 수정에는 의미 있는 관련 검증을 수행해줘.
통과한 검사는 새 변경·실패·미해결 문제가 없으면 반복하지 마.
사용자 데이터 삭제나 배포가 필요하면 결과를 준비한 뒤 확인해줘.
최종 답에는 바뀐 동작, 확인 결과, 남은 문제를 적어줘.

D. 조사 — 근거를 확인한 결과 받기

[질문]을 공식 자료와 원문 중심으로 조사해줘.
활용 목적: [목적]
조사 범위: [기간·대상·지역]

필요한 검색과 자료 읽기는 진행해줘.
중요한 주장 옆에 실제로 확인한 출처 링크를 붙여줘.
최신 정보는 확인 시점을 적고, 출처가 다르면 차이를 설명해줘.
확인한 사실·추론·제안을 구분해줘.
자료가 부족한 부분은 만들지 말고 어떤 정보가 없는지 적어줘.
완료 결과: 핵심 발견 3개, 내 업무에 미치는 영향, 출처.

E. 앱·브라우저 — 승인은 마지막 단계에

작업: [예: 받은 자료로 고객 회신 메일 준비]
사용할 자료·앱: [대상]
완료 기준: [예: 제목·본문·첨부 후보가 있는 회신 초안]

자료 읽기, 정리, 초안 작성은 바로 진행해줘.
이미 허락한 범위는 같은 대화에서 다시 승인받지 마.
발송·게시·결제·영구 삭제는 실행 전에 확인해줘.
승인이 필요하다면 그 전까지 할 일을 끝내고 검토할 결과를 보여줘.
연결이나 권한이 없으면 가능한 준비부터 하고 필요한 조치만 알려줘.
막히면 같은 시도를 반복하지 말고 원인과 다음 방법을 설명해줘.

F. 화면·웹페이지 — 완성 모습까지 확인

[제품·화면]을 [사용자]가 [행동]할 수 있게 구현해줘.
참고 디자인: [링크·이미지·기존 화면]
유지할 것: [색·글꼴·배치·기능]

기존 디자인을 확인하고 요청한 범위의 화면을 완성해줘.
모바일과 데스크톱에서 읽기 쉬운 크기와 간격을 사용해줘.
버튼·입력·로딩·오류 상태 등 실제 사용 흐름을 확인해줘.
완료 조건은 코드 작성, 실행, 화면 확인, 발견한 문제 수정까지야.
참고 이미지의 문구나 데이터는 이번 내용에 맞게 바꿔줘.
공개 배포는 완성 화면을 보여준 뒤 확인받아줘.

G. 문서·요약 — 쉬운 글과 원본 서식

[자료]를 [초보 독자]를 위한 [문서]로 다시 써줘.
원본의 제목 체계, 표, 강조 박스, 글꼴과 색을 유지해줘.

어려운 말은 처음 나올 때 쉬운 뜻과 예를 붙여줘.
한 문단에 생각 하나씩 쓰고, 핵심 내용을 먼저 말해줘.
표는 비교 정보, 목록은 순서나 나란한 항목에만 써줘.
‘결론적으로’, ‘주목할 점은’ 같은 상투어는 줄여줘.
불필요한 대비 문장과 반복 요약은 덧붙이지 마.
가상의 위험만으로 면책 문구나 체크리스트를 추가하지 마.
꼭 필요한 사실·조건·예외는 남기고 완성 문서를 제공해줘.

H. 멈춤 원인·최종 품질 점검

현재 요청한 결과를 기준으로 남은 일을 확인해줘.
빠진 항목과 잘못된 수치는 가능한 범위에서 바로 고쳐줘.
이미 통과한 검사를 이유 없이 반복하지 마.

스킬 지침과 내 명시적 요청이 충돌하면 내 요청을 우선해줘.
파일 지시 때문에 멈추면 파일명·링크·관련 문장을 알려줘.
그 문장이 실제 승인 요구인지 네 해석인지 구분해줘.
단, 서비스의 상위 규칙과 실제 권한 제한은 지켜줘.
아직 하지 않은 작업을 완료했다고 표현하지 마.
최종 결과와 해결되지 않은 항목만 간결하게 알려줘.

5. API 설정 레시피 · 선택 읽기

기본: 프로그램에서 Astra 호출하기

```
import OpenAI from "openai";
const openai = new OpenAI();

const response = await openai.responses.create({
  model: "gpt-6-astra",
  reasoning: { effort: "medium" },
  text: { verbosity: "medium" },
  input: "신입사원용 회의 준비 안내문을 5문장으로 완성해줘."
});

console.log(response.output_text);
```

API는 내 프로그램이 AI에 요청을 보내는 연결 방식이다. 위 코드는 Node.js와 openai 패키지, OPENAI_API_KEY 환경변수가 준비된 서버 환경용 예시다. medium은 이 책의 비교 시작점이다. 대화창만 사용한다면 이 장은 건너뛰어도 된다. [3]

장기 작업: 앞선 대화를 이어 요청하기

```
// 위 기본 예시의 response를 이어서 사용
const next = await openai.responses.create({
  model: "gpt-6-astra",
  previous_response_id: response.id,
  reasoning: { effort: "medium" },
  input: "같은 독자와 목적을 유지하고 준비물 예시를 추가해줘."
});

console.log(next.output_text);
```

- **previous_response_id** — 앞선 응답의 ID를 넣어 대화 맥락을 이어간다.
- 전제가 바뀌면 — ‘대상을 신입에서 팀장으로 변경’처럼 새 기준을 분명히 보낸다.
- 추론 상태 — 대화 연결과 추론 재사용은 구분한다. 더 세밀한 상태·저장 설정은 Reasoning 가이드를 확인한다. [4]

품질 우선: 생각할 양을 비교하기

```
const review = await openai.responses.create({
  model: "gpt-6-astra",
  reasoning: { effort: "high" },
  input: "이 계획에서 서로 충돌하는 조건을 찾아 설명해줘: ..."
});

console.log(review.output_text);
```

effort는 답변 전에 생각하는 정도이고, **verbosity**는 답변의 자세함이다. 더 깊게 생각하게 해도 글을 길게 쓸 필요는 없다. **low·medium·high**를 같은 업무로 비교한다. **Pro mode**는 고난도 작업의 선택 기능이며 자세한 조건은 공식 가이드를 확인한다. [4]

캐싱: 반복되는 앞부분을 재사용하기

```
// 반복 요청에서 공통 설명을 앞에 두는 예시
const request = {
  model: "gpt-6-astra",
  reasoning: { effort: "medium" },
  input: [
    {
      role: "developer",
      content: "고객 안내문은 짧고 친절하게 작성해."
    },
    {
      role: "user",
      content: "배송이 하루 지연된다는 안내문을 작성해줘."
    }
  ]
};
// 실제로 긴 공통 입력이 반복될 때 캐시를 검토
const result = await openai.responses.create(request);
```

- 캐시는 같은 입력의 앞부분을 다시 활용하는 기능이다. 개인용 대화에서 직접 설정할 필요는 없다.
- 공통 정책·설명 앞, 매번 달라지는 질문은 뒤에 둔다.
- 짧은 예시만으로 캐시 적중이나 비용 절약이 보장되지는 않는다. 반복되는 실제 입력으로 확인한다.
- 개발자는 캐시 읽기뿐 아니라 쓰기 비용도 함께 확인한다. 캐시 경계·유효시간·요금은 공식 문서의 현재 조건을 따른다. [5]

이전 API에서 옮길 때 꼭 확인할 것

전환 주의 **Astra**는 reasoning effort의 none을 지원하지 않는다. 기존 none·minimal은 low부터 비교한다. 도구 호출은 Responses API를 사용한다. temperature·top_p·top_logprobs는 제거한다. [1]

```
{
  "model": "gpt-6-astra",
  "reasoning": { "effort": "low" },
  "input": "이 문장을 쉬운 말로 바꿔줘: ..."
}

// Responses 요청 본문 예시
// Chat Completions의 reasoning_effort와 구분
```

6. 기능을 언제 쓰고 언제 피할까

기능·방식	쓰면 좋은 경우	먼저 확인할 것
대화 이어가기	같은 문서 연속 수정	낡은 전제가 남지 않았나
깊게 생각하기	복잡한 조건·중요 검토	추가 시간이 가치 있나
캐시 재사용	긴 공통 설명 반복	실제로 반복되는 입력인가
도구 결과 묶기	많은 자료 필터·집계	필수 근거가 남는가
여러 에이전트	서로 독립적인 조사	중복 없이 합칠 수 있나
작업 중 방향 수정	독자·조건이 변경됨	유지·변경할 점이 명확한가

도구 작업을 묶어도 되는지 판단하기

- 도구는 검색·파일 읽기처럼 AI가 쓰는 기능이다. 먼저 필요한 도구가 연결돼 있는지 확인한다.
- 여러 자료를 읽어 같은 항목만 뽑는 일이라면 결과를 한 번에 모으는 방식이 유용하다.
- 앞 결과를 보고 다음 행동을 결정해야 하거나 승인이 끼어 있으면 순서대로 진행한다.

- 요약하면서 출처나 중요한 예외가 사라지지 않도록 결과에 남길 항목을 정한다.

예: ‘제품별 가격·기능·출처만 모아 비교해줘. 가입이나 구매는 하지 마.’ 프로그램으로 결과를 줄여 처리하는 방식은 **Programmatic Tool Calling**이라고 한다. 일반 사용자는 원하는 결과만 설명해도 된다. [6]

여러 AI에게 나눠 맡겨도 되는지 판단하기

- 에이전트는 맡은 일을 도구와 함께 수행하는 AI다. 여러 에이전트 사용은 해당 환경의 지원이 필요하다.
- 예: ‘A는 기능, B는 사용자 안내 자료를 조사하고 마지막에 같은 기준의 표로 합쳐줘.’
- 각자 맡을 범위가 겹치지 않게 정하고, 최종 결과를 합칠 역할을 지정한다.
- 짧은 문장 하나를 고치는 일에는 나눠 맡길 이점이 작다. 독립 작업이 많을 때만 검토한다. [7]

7. 내 업무에서 좋아졌는지 확인하기

먼저 자주 하는 일 3개를 고른다. 같은 자료에 기존 요청과 새 요청을 각각 써보고 아래 항목을 비교한다. 정밀한 점수보다 ‘내가 다시 손볼 일이 줄었는가’를 먼저 본다. 수치는 직접 측정해 채운다.

비교 항목	확인할 질문	기록할 것
업무 완료	바로 쓸 결과가 나왔나?	완료 / 미완료
누락	필수 항목이 모두 있나?	빠진 항목 수
정확성	사실·수치·링크가 맞나?	수정한 오류 수
재질문	불필요하게 멈췄나?	추가 답변 횟수
내 수정량	얼마나 다시 손봤나?	수정 시간·문장 수
걸린 시간	완성까지 얼마나 걸렸나?	전체 소요 시간
비용·사용량	절약 효과가 있었나?	API 업무당 비용 등

추천 비교 순서

1. 같은 입력 자료와 원하는 결과를 정하고 기존 답변을 저장한다.
2. 모델을 **Astra**로 바꿨다면 먼저 이전과 같은 요청문으로 결과를 본다.
3. ‘끝까지 진행’과 자율 진행 범위를 추가하고 재질문이 줄었는지 본다.
4. 승인할 일을 분리하고, 승인 전 검토할 결과가 만들어지는지 본다.
5. 글투와 분량을 지정한 뒤 내 수정 시간이 줄었는지 확인한다.
6. 코딩 작업이라면 필요한 검증을 유지하면서 중복 검사가 줄었는지 본다.
7. **API** 사용자만 **effort**를 한 단계 바꿔 품질과 시간을 함께 비교한다.
8. 좋았던 요청문을 저장한다. 여러 조건을 한꺼번에 바꾸면 개선 원인을 알기 어렵다.

8. 흔한 실패 패턴과 처방

겪는 문제	가능한 원인	이렇게 고치기
‘네, 가능합니다’로 끝남	요청의 실행 의도가 모호	‘지금 결과까지 완성해줘’
계속 ‘진행할까요?’	허락된 범위가 불명확	자율 진행·승인 단계 분리
초안도 없이 승인 질문	승인 시점이 너무 이름	검토할 결과부터 준비 요청
파일을 읽고 멈춤	스칼·규칙의 지시 충돌	파일명과 해당 문장 확인
딱딱한 표·목록이 많음	원하는 문체가 미지정	문단·표 사용 기준 제시
사소한 수정도 오래 걸림	불필요한 검사까지 반복	변경에 맞는 검증만 요청
짧아졌지만 내용이 빠짐	분량만 정하고 필수항목 누락	완료 기준에 필수내용 명시
API 전환 뒤 오류	이전 모델 설정을 그대로 사용	none·도구 호출·매개변수 점검

9. 30분 적용 체크리스트

- □ 처음 5분: 자주 하는 업무 하나와 입력 자료를 준비한다.
- □ 누가 어디에 쓸 결과인지 한 문장으로 적는다.
- □ 다음 5분: 형식·분량·필수 항목을 정한다.
- □ 2장의 시작 문장을 요청 맨 위에 넣는다.
- □ 알아서 결정할 일과 확인할 일을 나눈다.
- □ 4장에서 해당 업무의 템플릿 하나를 골라 대괄호를 채운다.
- □ 다음 10분: 실행하고, 필요하면 바뀐 조건을 짧게 추가한다.
- □ 멈췄다면 핵심 정보 부족인지 파일 지시 때문인지 확인한다.
- □ 마지막 10분: 필수 항목과 사실·수치·링크를 점검한다.
- □ 재질문 횟수와 내 수정 시간을 기존 방식과 비교한다.
- □ 잘 작동한 공통 규칙만 저장하고 이번 업무의 임시 조건은 뺀다.
- □ 다음 업무에는 한 가지씩 바꾸며 내게 맞는 요청문으로 다듬는다.

10. 공식 자료 · 더 알아보기

- [\[1\] GPT-6 Astra 모델 안내](#) — 행동 특성, 자율 진행, 전환 시 확인할 설정
- [\[2\] 스킬과 프롬프트 다시 점검하기](#) — AGENTS.md, 불필요한 절차, 완료 기준
- [\[3\] Responses API](#) — 프로그램에서 요청하고 결과를 받는 기본 방식
- [\[4\] Reasoning 가이드](#) — 생각할 양, 추론 상태, Pro mode
- [\[5\] Prompt caching](#) — 반복 입력 재사용과 비용 확인
- [\[6\] Programmatic Tool Calling](#) — 많은 도구 결과를 코드로 정리
- [\[7\] Responses multi-agent](#) — 여러 에이전트의 역할 분담
- [\[8\] Async tool calling](#) — 도구 실행을 기다리는 동안 독립 작업 진행

오늘 바로 해보기 업무 하나를 고르고 ‘허락된 범위에서 끝까지 진행해줘’를 붙여보자.
완성할 결과와 확인할 단계를 함께 적으면 다음 요청을 훨씬 구체적으로 만들 수 있다.