TRAINING

生成AIエバンジェリスト育成研修

Day3 AI駆動開発とガバナンス

ギークス株式会社 御中 / 2026年8月26日

Givery

本資料の二次転載禁止について

禁止事項

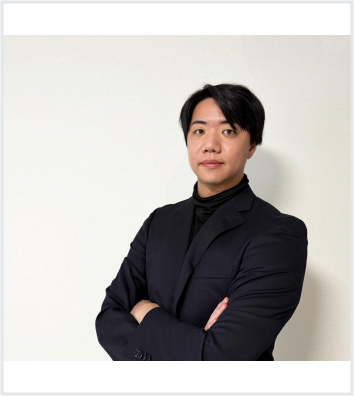
本資料は、株式会社ギブリー（Givery, Inc.）が実施する研修のために作成した著作物です。著作権は株式会社ギブリーに帰属します。

研修受講者ご本人の学習目的の利用以外での複製・転載・再配布、第三者への提供、SNSやWebサイト等での公開、社内外での二次利用を禁止します。

ご不明な点は、講師までお問い合わせください。

講師紹介

生成AI研修の設計と実施を担当



人的資本インテリジェンス部門 講師
生成AIエバンジェリスト

安田 光喜

Mitsuki Yasuda

プロフィール

公立はこだて未来大学 大学院（メディアデザイン領域）を卒業し、街づくり会社での複合商業ビルの立ち上げにおける情報インフラ整備やデザイン全般業務の責任として関与。その後デザイン事務所を立ち上げ、飲食店を中心としたデザイン業務や美術館展示のアートコンテンツ開発などを行う。また、地域ブランディングにも注力し、市主催の企画運営業務を行なった。2018年10月『小中学生向けのプログラミング教育』に注目し、函館市で初めての小中学生向けプログラミングスクール『D-SCHOOL北海道』を立ち上げて運営。

得意領域

・生成AIをはじめとした世界最先端トレンド
・生成AIを用いた新規事業の開発、運営、補助
・アプリケーション開発/自動化プログラム環境構築
・教育（大学講師、研修、スクール運営など）

実績

・生成AI研修
・大手企業、外資、教育機関(小中高大）でのITトレンド講演
・自治体と連携した地域教育実績
・上場企業社員対象のリスキリング支援

保有資格

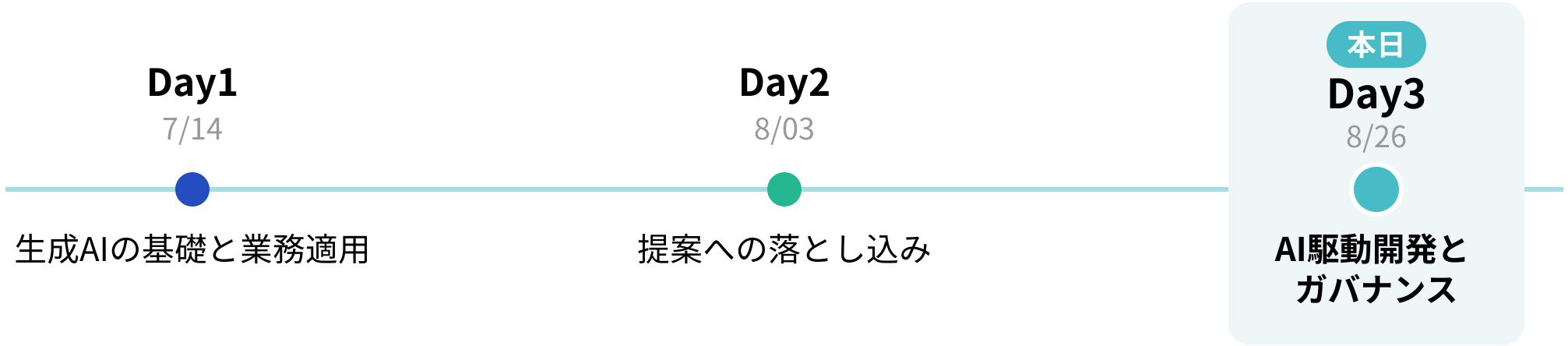
・全国統一生成AI活用技能試験 1級
・生成AIパスポート
・G検定

本日の進め方

分からない箇所はその場でご質問ください。
手が止まったまま進むほうが困ります。

Day1からDay3までの流れ

Day3で獲得する開発現場の言葉



本日のゴール

開発現場のAI活用を自分の言葉で説明できる状態

01

AI駆動開発の手法と用語の説明が可能

生成AIやAI駆動開発で使われる用語を、自分の言葉で言い換えられます。

02

ツールの違いを機能で判断可能

Copilot型・Chat型・Agent型の違いを、機能の観点で比較できます。

03

セキュリティの勘所を押さえた提案が可能

入力リスクと出力リスクを踏まえ、現場で使える提案ができます。

本日のタイムテーブル

後半は手を動かす時間とロープレ

内容	形式	所要
オープニング	座学	**[15min]**
AI駆動開発の全体像と営業転換	座学	**[55min]**
休憩	—	[10min]
セキュリティ・ガバナンス	座学	**[30min]**
理解度確認テスト	テスト	**[10min]**
休憩	—	[10min]
簡易プロトタイプ開発	ハンズオン	**[100min]**
休憩	—	[10min]
ロールプレイング	ワーク	**[40min]**
振り返りと質疑	座学	**[20min]**

※本日の研修時間は合計300分（5時間）です。休憩30分を含みます。

補足

用語の確認① 生成AIの土台

本日のスライドに出てくる基本の6語

生成AI	文章や画像を新しく作り出すAIの総称
プロンプト	AIへ渡す指示文。条件と出力の形式まで書く
コンテキスト	今回の作業でAIが読める情報のまとまり
モデル	AIの本体。版が上がると性能と価格が変化
トークン	AIが文字を数える単位。料金と上限の基準
ハルシネーション	もっともらしい誤りをAIが出す現象

同じ用語集を教材サイトの下部に置いています。当日わからない言葉が出たら、そちらを開いてください。

補足

用語の確認② 開発の進め方

開発の進め方を指す6語

AI駆動開発	AIに任せる前提で組み直した開発の進め方
仕様駆動開発	仕様書を先に成果物として書く進め方
テスト駆動開発	先にテストを書き、通るコードを書く進め方
プロトタイプ	方向性を確かめるために作る動く試作品
リファクタリング	動きを変えずにコードを整理し直す作業
BPMN	業務の流れを決まった記号で描く図の書き方

同じ用語集を教材サイトの下部に置いています。当日わからない言葉が出たら、そちらを開いてください。

補足

用語の確認③ ツールと運用

ツールと運用まわりの6語

エージェント	手順を自分で決めて作業まで進めるAI
IDE	コードを書くための統合開発環境
ハーネス	AIに渡すルール・情報・手順の一式
AgentRules	AIへ常に効かせる決めごとを書くファイル
シャドーAI	会社が把握しないまま使われている生成AI
ガードレール	使ってよい範囲を先に決めておく仕組み

同じ用語集を教材サイトの下部に置いています。当日わからない言葉が出たら、そちらを開いてください。

CHAPTER 01

AIが変えたもの

業務の前提が「AIありき」に置き換わった数年

010

この章では、企業が公表した数値と開発現場の変化を確認します。
効率の話だけでなく、信頼が追いついていない部分もあわせて見ていきます。

AI前提へ移った業務の効率

業務の標準速度がAI利用を織り込んだ水準へ移動

業務で生成AIを使う日本企業

86.4%

前年度調査の55.2%から急伸しました。

活用方針を定めた企業

個人の生成AI利用経験

生成AIを使う企業が例外ではなく、多数派になった段階です。
個人の利用経験率は58.8%まで伸びましたが、中国の93.6%とは開きがあります。
利用が先に広がり、活用方針の整備が後から追いついている形です。

出所: 総務省 令和8年版情報通信白書（2026年7月公開、企業515件・個人1,236件）

企業が公表した削減効果

効果は時間と件数という測れる単位で公表

企業	用途	公表された効果	公表時期
パナソニック コネクト	社内AIアシスタント	年間78.8万時間を削減（総労働時間の3.4%）	2026年7月
LINEヤフー	人事総務領域	月1,600時間以上の削減を見込み	2026年2月
GMO インターネットグループ	全社の業務全般	月43.2万時間を削減（1人あたり65.1時間）	2026年6月
サイバーエージェント	開発の要件定義と調査	要件定義・開発の工数が3割減 仕様・影響調査の工数は約75%減	2026年7月
DeNA	コードレビュー	人のレビュー回数 7.2回 → 2.7回 コメント数 6.0件 → 1.9件	2026年1月

出所: 各社の公式発表と技術ブログ（2026年1月〜7月公開）。数値はいずれも各社の自社集計です。

解説

AI駆動開発とは

実装を任せて人は判断と確認へ回る進め方

設計・実装・テストのうちAIに任せられる工程を前提に、進め方そのものを組み直した開発のやり方

身近な言い方

下書きをAIに作らせて、人は判断と確認に時間を使う進め方です。手を動かす量が減る代わりに、指示の書き方と確認の目が効いてきます。

なぜ今よく聞くか

モデルの性能が上がり、まとまった量のコードを一度に書けるようになったためです。速さより先に、レビュー体制のほうが足りなくなります。

関連する言葉

仕様駆動開発　テスト駆動開発　エージェント

AI駆動開発を前提とした開発手法

進め方は「どこまで先に決めるか」で分岐

手法	進め方	向く場面	代表的な実装
バイブコーディング	生成物を読まずに 指示と試行を繰り返す	使い捨ての小さな道具	チャットで一気に生成
仕様駆動開発	要件・設計・タスクを 先に固定してから実装	仕様が固まる業務系	AWS Kiro / Spec Kit
テスト駆動開発	失敗するテストを先に 書かせ、通るまで直す	壊せない既存機能	Claude Code の二役分担
ドキュメント駆動	決めた理由を残し 次の依頼の前提にする	引き継ぎと長期保守	spec と RDR・ADR の分離
ペア型レビュー	AIが一次レビューし 人が最終判断を担う	レビュー待ちの滞留	Claude Code Actions によるPR自動レビュー

出所: 各手法の公式ドキュメントと各社の技術ブログ（Kiro / Spec Kit / Anthropic / メルカリ / DeNA）

人が手を動かす場所が後ろにずれる構図

従来の開発	工程	AI駆動開発
人が聞き取って文書化	**要件定義** 言語化	人が決め、AIが文書化
人が方式を決めて図に	**設計** 選択	AIの案から人が選ぶ
人がコードを書く	**実装** 委譲	AIが書き、人が方針を出す
人がテストを書いて実行	**テスト** 自動化	AIが書き、機械が合否判定
人が全行を読む	**レビュー** 絞り込み	AIが一次確認、人が最終判断

CHAPTER 02

AIと人間の業務切り分け

任せる範囲の判断軸と、仕様駆動・テスト駆動の考え方

016

この章では、AIに任せる範囲の決め方と、二つの開発手法を確認します。
あわせて、エージェントに文脈と規律を渡すハーネスの仕組みも扱います。

解説

AIが得意なこと・苦手なこと

量が多くて形が決まっている作業ほど、任せた効果が出ます

得意なこと

下書きを大量に出す
案を10個出す、文章の初稿を書く

形を変える
議事録を表にする、箇条書きを図にする

洗い出す
確認項目やテストのパターンを並べる

よくある形を作る
似た例が世の中に多い画面やコード

苦手なこと

正解が1つに決まらない判断
この案件を受けるかどうか

社内だけの事情
誰が決裁するか、過去の経緯

公開されていない最新情報
先週決まった社内ルール

責任を持つこと
出した内容の最終的な確認

得意な側を任せて、苦手な側を人が持つ。この分け方が、次のページの判断軸につながります。

やり直しの効きやすさと正解の一意性で振り分け

3つのゾーン

人主導
やり直しが高くつく判断

AI下書き＋人確認
案は機械、可否は人

AI主導
結果を機械が判定できる

> 迷ったときは、やり直しのコストが低いものから任せてください。

解説

仕様駆動開発とは

作るものの説明書をコードより先に置く進め方

何を作るかを先に文章で確定させ、実装はAIに任せて何度でも作り直せるようにする進め方

身近な言い方

提案書を先に固めてから制作に入るのと同じ順番です。文章を直せば作り直しがきくので、あとから戻る手戻りが減ります。

なぜ今よく聞くか

AIは指示があいまいだと毎回違うものを出します。仕様を文書にしておくと、同じ指示から同じ結果へ近づきます。

関連する言葉

Kiro　GitHub Spec Kit　要件定義

仕様駆動開発で作る文書

要件・設計・タスクの3文書をコードの前に確定

要件・設計・タスクの3文書を先に確定させ、
実装はAIに何度でも作らせる

文書を直せば、実装はやり直せます。

同じ文書から、別の実装を作らせることもできます。

> **代表的な実装＝ AWS Kiro / GitHub Spec Kit**

クイズ

クイズ1

Q. AI駆動開発で「仕様駆動開発」が解決しようとしている課題はどれか

A 生成の速度が遅い

B 利用料金が高い

C 依頼が曖昧で毎回違う出力になる

D 対応する言語が少ない

正解は次のページで確認します。まず自分で考えてみてください。

解説

クイズ1の解説

仕様駆動開発が消すのは依頼の曖昧さによる出力のばらつき

C **依頼が曖昧で毎回違う出力になる** 正解

要件・設計・タスクを先に成果物として固定し、同じ仕様から同じ実装を出させます。

A	生成の速度が遅い	速度はモデル側の性能で決まります。
B	利用料金が高い	料金はプランで決まり、書き方とは別の話です。
D	対応する言語が少ない	対応言語はツール側の実装範囲の話です。

Kiro は requirements・design・tasks の3ファイル、Spec Kit は4段階のコマンドで仕様を固定します。

解説

テスト駆動開発とは

合格条件を先に書いてコードをあとから書く順番

動作の合格条件をテストとして先に書き、それが通る状態を作ってから中身を整える進め方

身近な言い方

検収条件を先に決めてから作業に入るのと同じです。できたかどうかを人の感覚ではなく、機械が判定します。

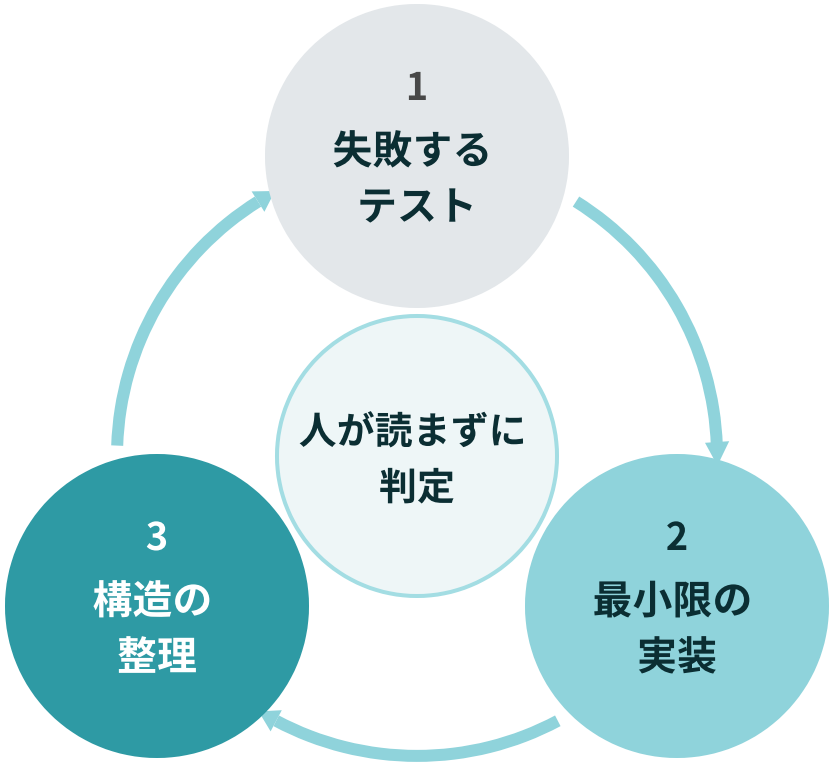
なぜ今よく聞くか

AIが書いたコードを人が全部読むのは現実的ではありません。テストがあると、確認の一部を機械へ預けられます。

関連する言葉

レッド　グリーン　リファクタ

失敗するテスト・最小限の実装・構造の整理という循環



循環の3段階

1. 失敗するテスト
期待する振る舞いを、通らない
テストとして先に書きます。

2. 最小限の実装
そのテストが通るところまでを
実装します。

3. 構造の整理
動く状態を保ったまま、重複や
名前を直します。

> この循環そのものが、AIの出力に
> 合否を返す仕組みになります。

クイズ

クイズ2

Q. AIにテストを先に書かせる進め方の利点として最も適切なものはどれか

A 出力の正しさを人が読まずに判定できる

B トークンの消費量が減る

C 生成の速度が上がる

D 対応できる言語が増える

正解は次のページで確認します。まず自分で考えてみてください。

解説

クイズ2の解説

読んで確かめる作業を合否が返る仕組みへ置き換え

A **出力の正しさを人が読まずに判定できる** 正解

テストが合否を返すため、生成物の全文を人が読まなくても判定できます。

B トークンの消費量が減る

テストの作成と実行を挟むぶん、消費はむしろ増えます。

C 生成の速度が上がる

1回の生成そのものが速くなるわけではありません。

D 対応できる言語が増える

対応言語はモデル側の性能で決まります。

短くなるのは生成の時間ではなく、確認と手戻りにかかっていた時間です。

解説

ハーネスとは

AIを業務に沿って動かすための情報と決めごとの一式

毎回口で言っていた前提・資料・手順をファイルにして、常にAIへ効かせておく仕組み

身近な言い方

新しく入った人へ渡す業務マニュアルと用語集、過去資料のセットに近いものです。渡す量が増えるほど、外し方が減ります。

なぜ今よく聞くか

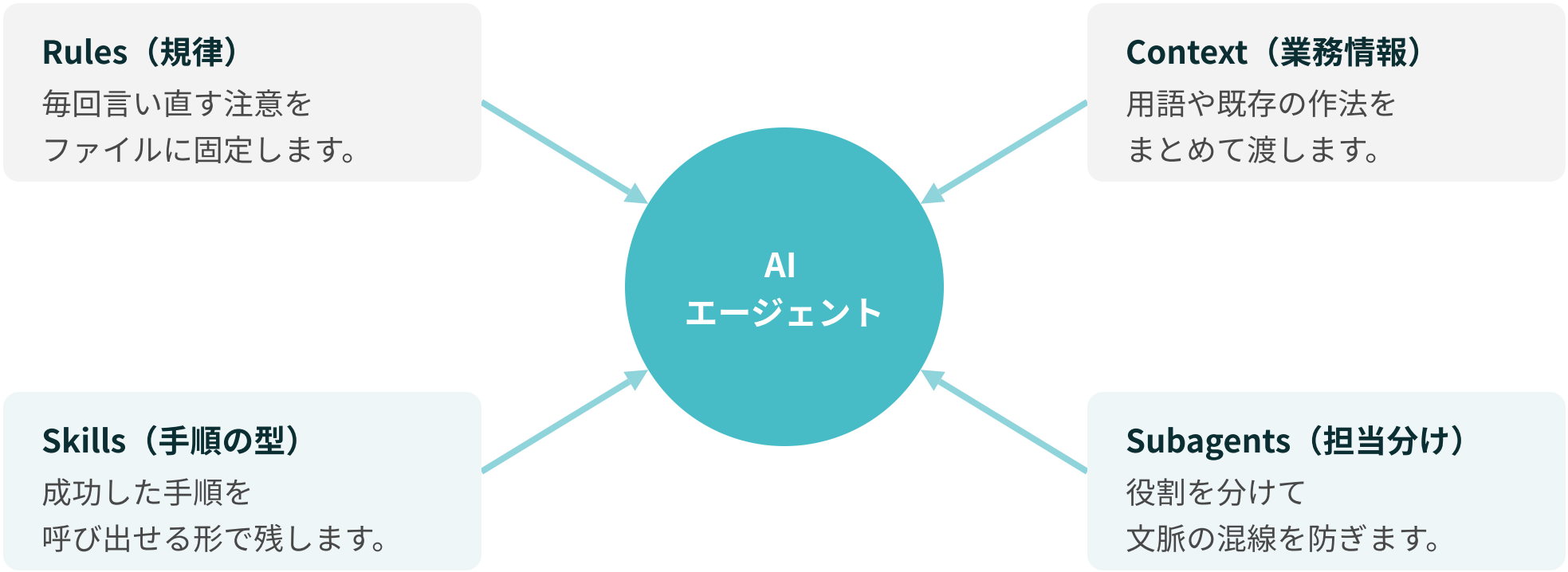
同じツールでも、渡してある情報の量で結果が変わります。差が出るのはモデルではなく準備のほうだと分かってきたためです。

関連する言葉

AgentRules　コンテキスト　Skill　Subagents

ハーネスの全体像

エージェントの精度は渡した情報と規律の量で決定



ハーネスはAIを乗りこなすための装具一式です。渡す情報と守らせる規律の総称として使います。

CHAPTER 03

ツールの使い分けと事例

置き場所で決まる得意領域と、公表されている効果

029

生成AIツールは、PC内のファイルを直接操作できるかどうかで3つに分かれます。
今日使う2つの役割分担と、企業が公表している事例をあわせて確認します。

生成AIツールは「PC内のファイルを直接操作できるか」で3パターン

置き場所で任せられる業務の範囲が判断可能

① ブラウザ完結

Gemini / ChatGPT

できること
調査・要約・文章の下書き
添付ファイルの読み取り

できないこと
PC内のファイルを
直接書き換えること

② エディタ統合

GitHub Copilot / VS Code

できること
エディタで開いた範囲の
生成と修正

できないこと
エディタの外にある
作業の代行

③ 実行環境ごと

Antigravity / Claude Code

できること
ファイル編集と
コマンド実行、差分提示

できないこと
起動フォルダの外への
承認なしの書き込み

※ 各ロゴは各社の商標です。

解説

今日2つのツールを使う理由

ゼロから作る場面と、手元のファイルを直す場面で道具が変わります

Gemini

ブラウザで使う

- 何もないところから1本つくるのが速い
- できあがったファイルは手元に落とす
- PC内のファイルを直接は触れない

Antigravity

PCにいれて使う

- 手元のファイルをそのまま開いて直せる
- 直した内容をその場で保存できる
- 何もない状態から始めるのは不向き

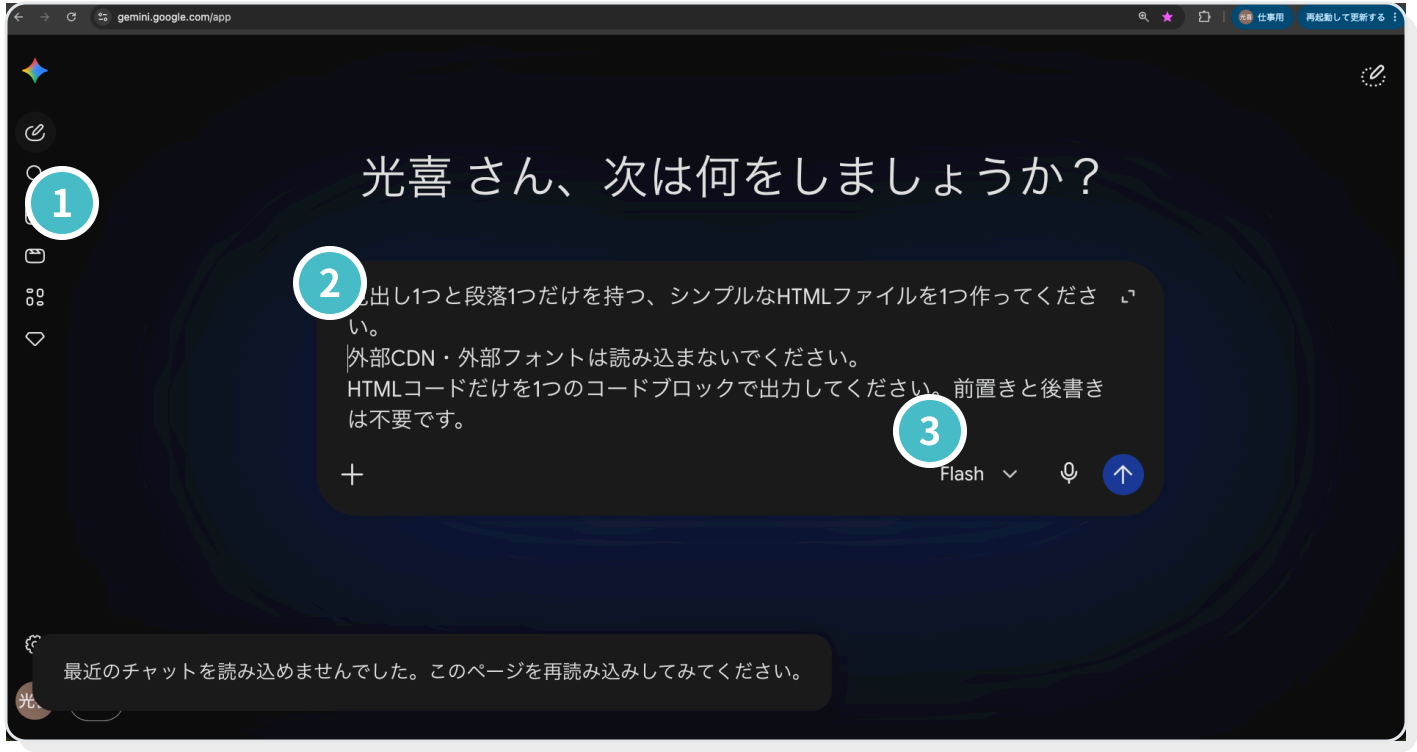
今日の流れ

Geminiで **つくる** → 手元に **落とす** → Antigravityで **直す** → ブラウザで **たしかめる**

コードを書く場面はありません。どちらのツールにも、日本語の文章で頼みます。

Geminiの画面構成

左のレール・中央の入力欄・モデル選択の3か所



1 **左のレール**

新しいチャットの開始、
過去の履歴、Gem を開く場所

2 **中央の入力欄**

指示の入力と、左下の「＋」
からのファイル添付

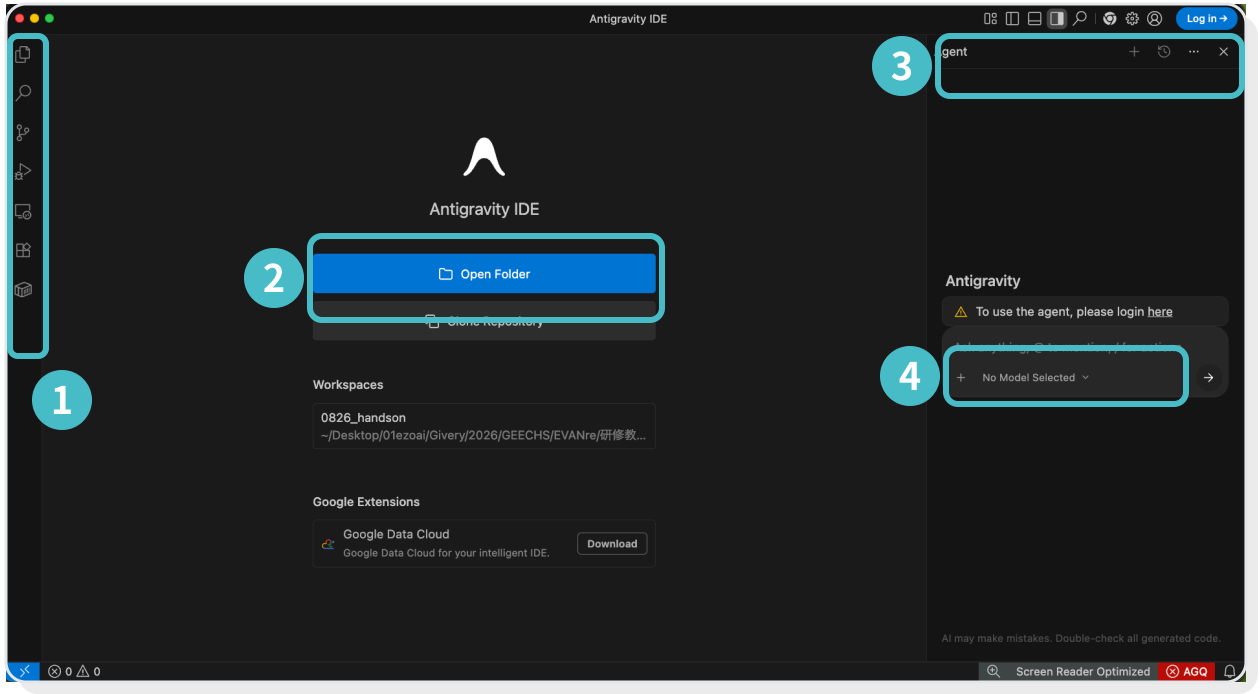
3 **モデル選択**

Flash と Pro の切り替え
無料プランでは回数制限つき

出所: Gemini アプリ ヘルプセンター / 公式プラン紹介ページ（2026年8月3日取得）

Antigravityの画面構成

書く場所のEditorと任せる場所のAgentパネル



1. **アクティビティバー**
   ファイル・検索・拡張機能の切替
2. **中央のEditor領域**
   フォルダを開く入口と編集画面
3. **右のAgentパネル**
   依頼を書いて渡す場所
4. **モデル選択**
   Gemini・Claudeなど6モデル

> Agent Manager は複数のエージェントをまとめて動かすダッシュボードで、Editor と切り替えて使います。

出所: Google Cloud 公式ブログ / Google Antigravity 公式ブログ・公式ドキュメント（2026年8月3日取得）

クイズ

クイズ3

Q. ブラウザ版のGeminiにできないことはどれか

A 貼り付けた長文の要約

B 引用付きの調査レポートの作成

C HTMLやコードの生成

D PC内のファイルを直接書き換える

正解は次のページで確認します。まず自分で考えてみてください。

解説

クイズ3の解説

境界は生成物を手元のファイルへ直接書き戻せない点

D **PC内のファイルを直接書き換える** 正解

ブラウザの中で完結するため、手元のファイルを開いて上書きする操作は行えません。

A	貼り付けた長文の要約	添付したファイルの要約も含めて行えます。
B	引用付きの調査レポート	Deep Research が計画・横断・生成の3段で進めます。
C	HTMLやコードの生成	生成はできます。受け取りはコピーかダウンロードです。

手元のファイルを直接触らせたい作業は、AntigravityのようにPC上で動くツールへ渡します。

Geminiの活用事例

効果を利用率と削減時間という測れる形で公表

企業	使い方	公表された効果	公開時期
船井総研グループ	Google Workspace with Gemini を約1,500名へ導入	検討開始から4か月で利用率97%	2025-08
星野リゾート	施設の魅力造成の業務で企画・文章作成・校正	作業時間を約30%削減校正は30分から5分程度	2025-09
ローム	Google Workspace with Gemini を900名以上へ展開	1人あたり月31時間の作業短縮を見込み	2026-06
ソラスト	全職員3万人以上をWorkspace と Gemini へ移行	ベテランの知見を全社で共有する基盤	2025-12
MEDITECH（米国）	会議の議事録とメール・文書の作成	従業員1人あたり週7時間の削減	2025-03

出所: Google Workspace ブログ（各社の導入事例、2025年3月～2026年6月公開）

失敗事例③ 情報流出と詐欺

入力と本人確認の2経路で被害が発生

経路① 生成AIへの入力

社内資料や個人データ → 生成AIの入力欄 → 応答以外での取り扱い

同意のない個人データの入力は、法違反となる可能性があります。

経路② なりすましの成立

公開された動画と音声 → AIで合成した映像と音声 → 会議での送金指示

映像と音声が本物に見えるため、会議での確認だけでは防げません。

確認された実例

サムスン電子（2023年5月）
社内情報の入力を受け、利用を全面禁止しました。
社内調査では約65%がリスクを認識しています。

個人情報保護委員会（2023年6月）
個人情報保護法第147条に基づく注意喚起です。
同意のない個人データの入力が対象となります。

アラップ 香港拠点（2024年）
AI で偽装したCFOとのビデオ会議を信じ、
2億香港ドルを15回に分けて送金しました。

IPA「情報セキュリティ10大脅威 2026」で、AIの利用をめぐるサイバーリスクが組織向け3位に初選出されました。

出所: TechCrunch（2023-05-02）／個人情報保護委員会（2023-06-02）／IPA（2026-01-29）
アラップの社名は Fortune Europe（2024-05-17）、送金額は The Register（2024-02-05）

クイズ

クイズ4

Q. AIの出力をそのまま社外向け資料に使う前に、必ず人が行う作業はどれか

A 文体を社内の口語に直す

B 数値と固有名詞を原典で照合する

C 分量を半分に要約する

D 同じ質問をもう一度AIに投げ直す

正解は次のページで確認します。まず自分で考えてみてください。

解説

クイズ4の解説

文体を直しても誤った数値と名前はそのまま残存

B **数値と固有名詞を原典で照合する** 正解

AIは出典を持たないまま断定形で書けるため、数字と名前は人が原典で確かめます。

A	文体を社内の口語に直す	読みやすさの調整であり、事実の誤りは残ります。
C	分量を半分に要約する	短くしても、誤った数値や名前はそのまま残ります。
D	同じ質問をもう一度AIに投げ直す	同じ誤りが返ることがあり、確認の代わりになりません。

照合に使った資料名を本文の脇に残しておくと、後から出所を聞かれたときに答えられます。

CHAPTER 04

セキュリティ・ガバナンス

事故の起き方と、決めておく範囲

040

この章では、実際に起きた事故の経路と、組織として決めておく範囲を確認します。
あわせて、生成コードに固有のリスクと、生成の前後で守らせる仕掛けを扱います。

入力リスクと出力リスク

危ないのは渡すものと受け取ったものの扱い

入力リスク　何を渡すか

機密を含む貼り付け
顧客名や単価、未公開の仕様を
そのまま入力すると社外へ出ます。

個人データの入力
本人の同意なく個人データを含む
入力は、法違反となりえます。

保存と学習の設定
入力が残るかは契約とプランで
変わります。使う前に確認します。

出力リスク　何を鵜呑みにするか

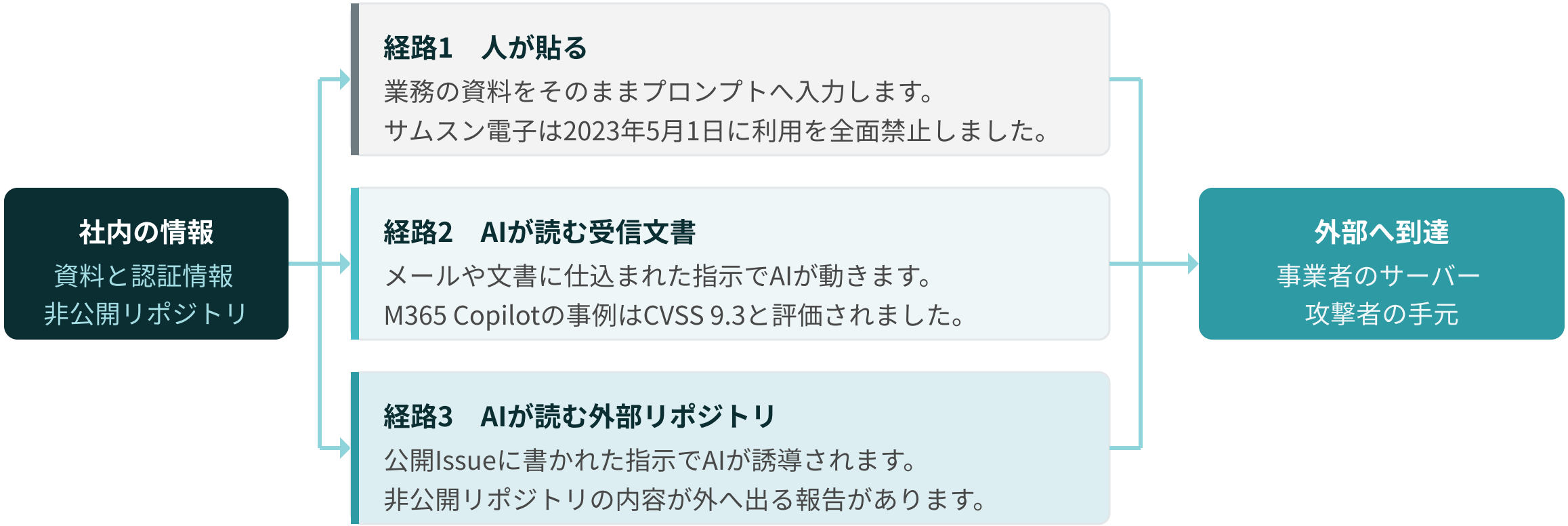
存在しない事実の引用
架空の判例6件を引用した書面で、
弁護士2名と事務所が制裁されました。

誤案内と責任の所在
チャットボットの誤案内について、
航空会社に支払いが命じられました。

生成コードの欠陥
AI生成コードの45%に、深刻度の
高い欠陥が確認されています。

出所（入力側）: TechCrunch 2023-05-02／個人情報保護委員会 2023-06-02
出所（出力側）: Mata v. Avianca 2023-06-22／Moffatt v. Air Canada 2024-02-14
Veracode 2025 GenAI Code Security Report 2025-07-30

漏れる経路は人が貼るとAIが読むの2系統

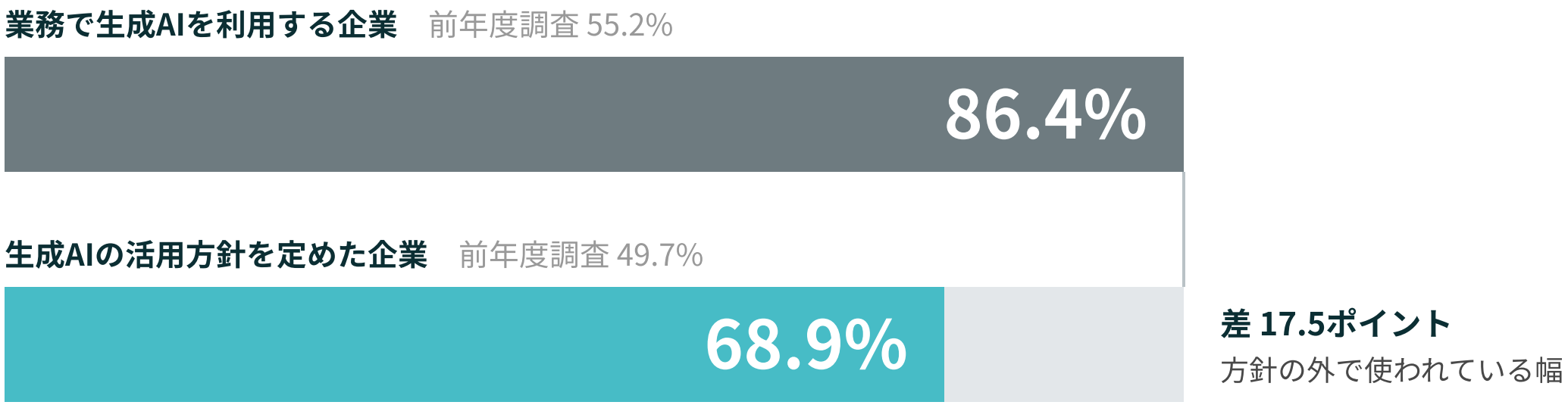


経路1は人の判断で止まりますが、経路2と経路3はAIが自分で読むため、権限の絞り込みと監視で止めます。

出所: TechCrunch 2023-05-02／MITRE CVE-2025-32711 2025-06-11／Invariant Labs 2025-05-26

ルールの外で使われる領域

使う企業86.4%と方針を決めた企業68.9%の差



差 17.5ポイント
方針の外で使われている幅

86.4%と68.9%の差は、方針が決まる前に現場で使われている範囲にあたります。
見えない利用を減らす近道は、禁止の追加ではなく、使ってよい範囲の明文化です。
IPAの情報セキュリティ10大脅威2026では、AI利用のリスクが組織向け3位に初選出されました。

出所: 総務省 令和8年版情報通信白書 2026-07／IPA 情報セキュリティ10大脅威 2026 2026-01-29

生成AIガバナンスの構成要素

外のルールを現場の設定まで落とす3階層

第1層　法とガイドライン
守るべき最低限の枠を国が示します。

第2層　社内規程
使ってよい範囲と承認を決めます。

第3層　現場の運用
設定と記録で規程を回します。
日々の点検もここに入ります。

日本

AI事業者ガイドライン
総務省と経済産業省の連名で策定されています。
最新は第1.2版で2026年3月31日公表です。

AI新法
正式名は人工知能関連技術の研究開発及び
活用の推進に関する法律です。
2025年6月4日公布、同年9月1日に全面施行です。

人工知能基本計画
2026年7月14日に閣議決定されました。

EU

EU AI Act
発効は2024年8月1日、主要適用は2026年8月2日です。

順番は入れ替えられません。国の枠組みを読んでから社内規程を書き、最後に設定へ落とします。

出所: 総務省／内閣府／European Commission の各公式ページ

エンジニアに特有のリスク

コードだから起きる問題は権利・秘密・ズレの3種

OWASP Top 10 for LLM Applications 2025 の該当項目

01 生成コードの類似性とライセンス

Copilotは約150文字の周辺コードを公開コードと照合します。
一致時はライセンス種別が表示され、遮断か採用かを選べます。

LLM03
Supply Chain

02 機密データの取り扱い

入力した内容は事業者へ送られ、設定によっては保存されます。
受信文書や公開Issueに仕込まれた指示でAIが動く例も出ています。

LLM02
Sensitive Information Disclosure

03 仕様とコードのズレ

実装だけが進むと、仕様書とコードは別物になっていきます。
AI支援を使った群ほど、自分のコードを安全だと誤信しました。

LLM06
Excessive Agency

AI生成コードの45%にOWASP Top 10級の欠陥が確認されています。動作の確認だけでは足りません。

出所: GitHub Blog 2023-08-03／Veracode 2025 GenAI Code Security Report
OWASP Top 10 for LLM Applications 2025／Perry et al. arXiv:2211.03622

安全に回すための2層

守らせる仕掛けは生成前と生成直後の2か所

1. **ルールの読み込み**
   規約と禁止事項をファイルで渡します。
2. **生成**
   依頼文どおりにAIが書きます。
3. **セルフチェック**
   テストと差分をAI自身に点検させます。
4. **人の確認**
   通ったものだけを読み、反映を決めます。

第1層　生成前の規律
毎回言い直さずに済みます。

速いのはこの工程だけ

第2層　生成直後の検査
合否が返る検査を渡します。

人が見る位置
ここだけは自動化しません。

ルールは文脈であって強制ではありません。守らせたい処理はHooksのような仕掛けで固定します。
IT技術者572名の調査では、95.5%が今後も人間によるコードレビューは必要と答えています。
出所: Anthropic Claude Code 公式ドキュメント／レバレジーズ 2026-07-15

口頭の前提をリポジトリのルールファイルへ固定

ルール未整備

- 入力してよい情報の線引きが人ごとに違います。
- 命名規則や例外処理の作法を毎回伝えています。
- 同じ指摘をレビューで繰り返しています。

ルール整備済み

- 顧客データや資格情報の扱いを明記しています。
- 命名規則・例外処理・依存追加の可否が固定です。
- 指摘はルールへの追記で次回から効きます。

セキュリティ要件とコーディング規約の置き場所

AGENTS.md	リポジトリのルートに1枚。Codex や Cursor など23ツールが読み込み
CLAUDE.md	4階層を連結して読み込み。1ファイル200行以内が公式の目安
.github/copilot-instructions.md	そのリポジトリの全リクエストに適用
.cursor/rules/*.mdc	description・globs・alwaysApply の組み合わせで4方式

出所: AGENTS.md 公式サイト／Anthropic Claude Code 公式ドキュメント／GitHub Docs／Cursor 公式ドキュメント

クイズ

クイズ5

Q. 秘密情報の混入チェックを、ルールファイルに書くだけでは徹底できない理由は

A 文字数の上限が小さいため

B 読めるツールが限られるため

C 文脈にすぎず強制力がないため

D 起動時にしか読まれないため

正解は次のページで確認します。まず自分で考えてみてください。

解説

クイズ5の解説

抜けが許されない検査だけHooksへ寄せる切り分け

C **文脈にすぎず強制力がないため** 正解

ルールファイルは読ませる文脈にとどまります。必ず実行させたい検査は Hooks に書きます。

A	文字数の上限が小さいため	200行は分量の目安で、強制力の有無とは別です。
B	読めるツールが限られるため	読み込めるツールは増えており、主因ではありません。
D	起動時にしか読まれないため	読み込む回数を増やしても、実行の保証にはなりません。

秘密情報の検出やテスト実行のように外せない処理は、settings.json の Hooks に寄せておきます。

ここまでの要点

人が読んで信じる運用から
検査で担保する運用へ

1. 入力してよい情報の線引きを、ルールファイルに書いて全員で共有します。
2. 生成コードは脆弱性が混ざる前提で、機械の検査を通してから採用します。
3. 必ず実行させたい処理はルールではなく Hooks に置き、毎回動かします。

検証ではAI生成コードの45%がOWASP Top 10級のセキュリティ欠陥を含んでいました。
出所: Veracode 2025 GenAI Code Security Report（2025年7月30日）

補足

理解度確認テストの案内

ここまでの講義範囲から20問

出題数	20問
配点	100点満点
所要時間	[10min]
実施方法	Googleフォーム
回収する情報	氏名のみ
出題範囲	ここまでの講義

回答の進め方

1. 配布するURLをブラウザで開きます。
2. 氏名を入力してから解答します。
3. 見直しは残り時間の範囲で行います。
4. 送信後に正解と解説を確認します。

出題はこれまでの講義内容からです。持ち込みの資料はスライドと配布物に限ります。

CHAPTER 05

簡易プロトタイプ開発

議事録を整理するアプリを、AIとの往復で組み立てる工程

052

Geminiで最初の1本を作り、Antigravityで機能を足していきます。
コードを書く場面はありません。依頼文と動作確認だけで進めます。

解説

BPMNとは

誰が何をするかを決まった記号で描く書き方

開始・作業・分岐・終了の記号と担当ごとのレーンで、業務の流れを表す国際規格の図

身近な言い方

業務フロー図の書き方を、記号のレベルまで統一したものです。書き手が変わっても、同じ意味で読めます。

これから出てくる場面

今日つくるアプリは、議事録からこの図を自動で描きます。図の形が決まっているからこそ、機械が組み立てられます。

関連する言葉

レーン　ゲートウェイ　タスク

解説

HTMLファイル1枚で動くしくみ

ダブルクリックで開ける1つのファイル

ファイルと拡張子

`.html`	ブラウザが開く。画面と動きが入っています
`.txt`	メモ帳が開く。文字だけのファイル
`.png`	画像。写真や図

後ろに付く文字で、開くソフトが決まります。

置き場所

- 0826_handson
  配布ZIPを解凍したフォルダ
  - output
    自分が作ったものを置く場所
    - **meeting-organizer.html**
      今日つくるファイル
  - answer
    うまくいかないときの完成版

サーバーもデータベースも使いません

ファイルをダブルクリックすると、ブラウザが開いて動きます。インターネットにつながっていなくても使えます。
入力した内容は、そのブラウザの中に残ります。他の人からは見えません。

ファイルが見つからないときは、Antigravity の左側にフォルダの中身が出ているかを確かめてください。

実演

今日つくるもの

議事録を貼ると担当者別のタスクボードができるアプリ

画面は完成版のタスクボード部分です。

1 議事録の貼り付け
入力欄にそのまま貼るだけ

2 3分類への整理
タスク・決定事項・懸念に仕分け

3 担当者別のカラム
名前ごとに列を分けて表示

4 優先度・期日・状態
カードに付く3つの属性

> 保存はブラウザの中で完結します。
> サーバーもデータベースも使いません。

ワーク

演習の進め方

Geminiで作ってAntigravityで直す7ステップ

STEP 1 [20min]

解説とデモ

生成物
講師の実演を見ます

STEP 2 [25min]

Geminiで生成

生成物
meeting-organizer.html の初版

STEP 3 [15min]

取り込みと操作練習

生成物
Antigravityで開いた作業フォルダ

STEP 4 [20min]

動作確認と修正

生成物
表示と動きを直した版

STEP 5 [20min]

機能追加

生成物
機能を1つ足した版

STEP 6 [40min]

ロールプレイング

生成物
聞き取った改修要望と振り返りシート

STEP 7 [8min]

振り返り

生成物
足りない機能のメモ

合計 [148min]。STEP5 は機能が1つ足せた時点で完了とします。手が止まったら講師を呼んでください。

解説

依頼文の型

依頼文に入れる3つの要素で結果が安定

01 **誰が何に使うか**
会議の議事録を整理して、部内に共有するために使います

02 **条件**
HTMLファイル1つで動くこと。外部のサービスは使わないこと

03 **出す形**
コードをそのまま貼れる形で、1ファイルにまとめて出してください

直すときの頼み方

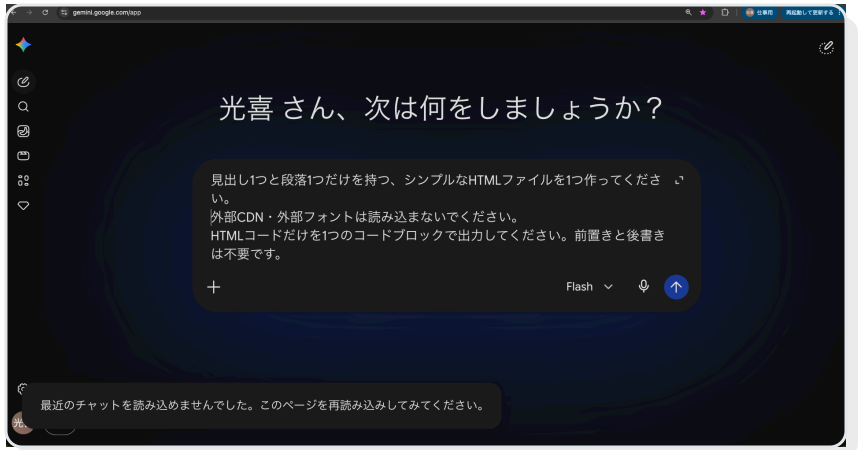
伝わりにくい書き方
うまく動きません。直してください。

伝わる書き方
コピーのボタンを押しても、何も起きません。
押したら決定事項の文字がコピーされる、
という状態にしてください。

ワーク

Geminiでプロンプト作成・送信・修正

動くものを先に出してから直す進め方



Geminiの入力欄に依頼文を貼って送信します。

1. **プロンプトの作成**
   つくるもの・入力・出力・制約の4点を明記します。

2. **送信と生成物の受け取り**
   HTMLが1枚返ってくるので、作業フォルダに保存します。

3. **足りない点の修正**
   直したい箇所を1つに絞って伝えます。
   まとめて頼むと原因が追えなくなります。

依頼文の型

```
# つくるもの：議事録を貼るとタスクボードになる1枚のHTML
# 入力：会議の議事録テキスト（担当者名と期日を含む）
# 出力：index.html を1ファイルだけ。外部ライブラリは使わない
# 制約：ブラウザだけで動く。データはブラウザ内に保存する
```

実演

ダウンロード・配置・Antigravityでの起動

フォルダごと開くのが前提

1 保存場所の指定

ダウンロードフォルダから、
作業用のフォルダへ移します。

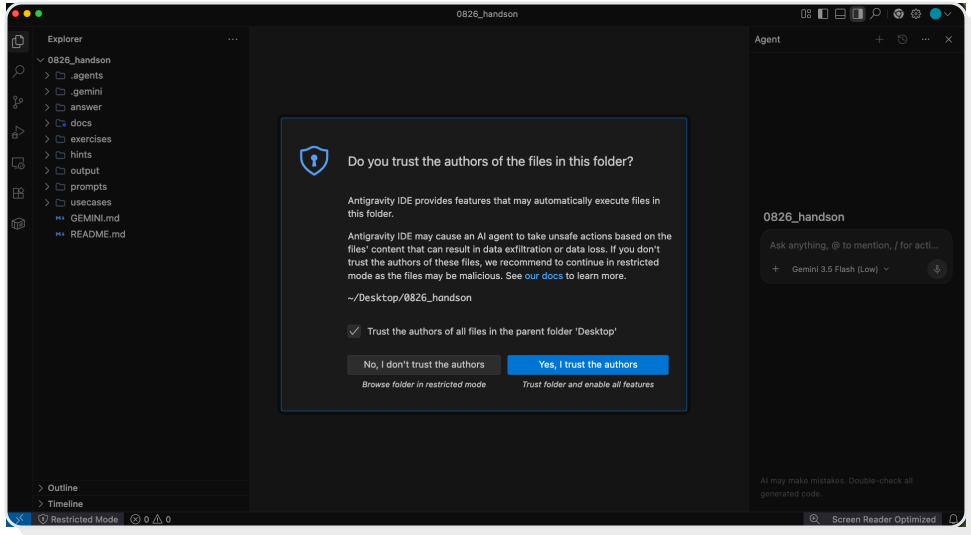
2 Open Folder の実行

Antigravityのメニューから
File > Open Folder を選びます。

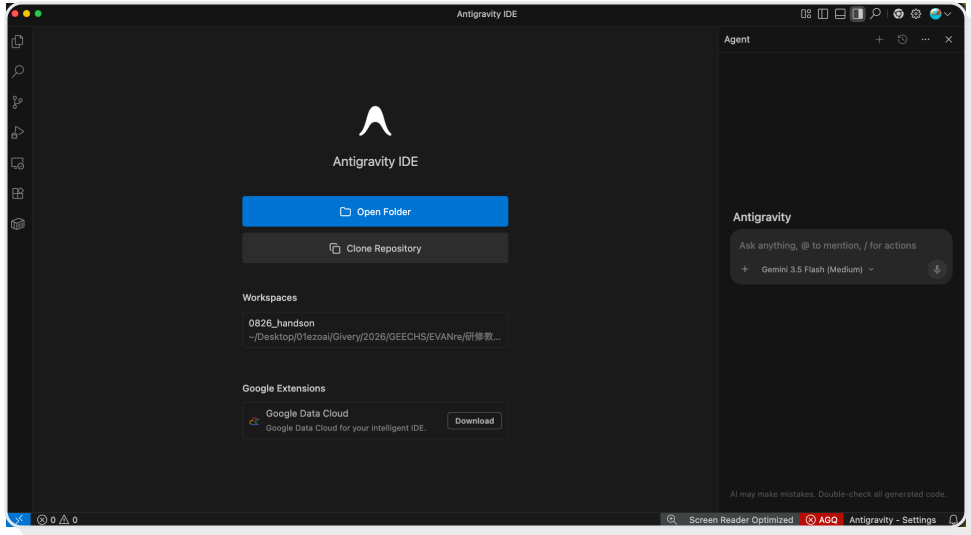
3 信頼ダイアログの確認

自分で作ったフォルダなので、
信頼する側を選びます。

Open Folder 直後の信頼ダイアログ



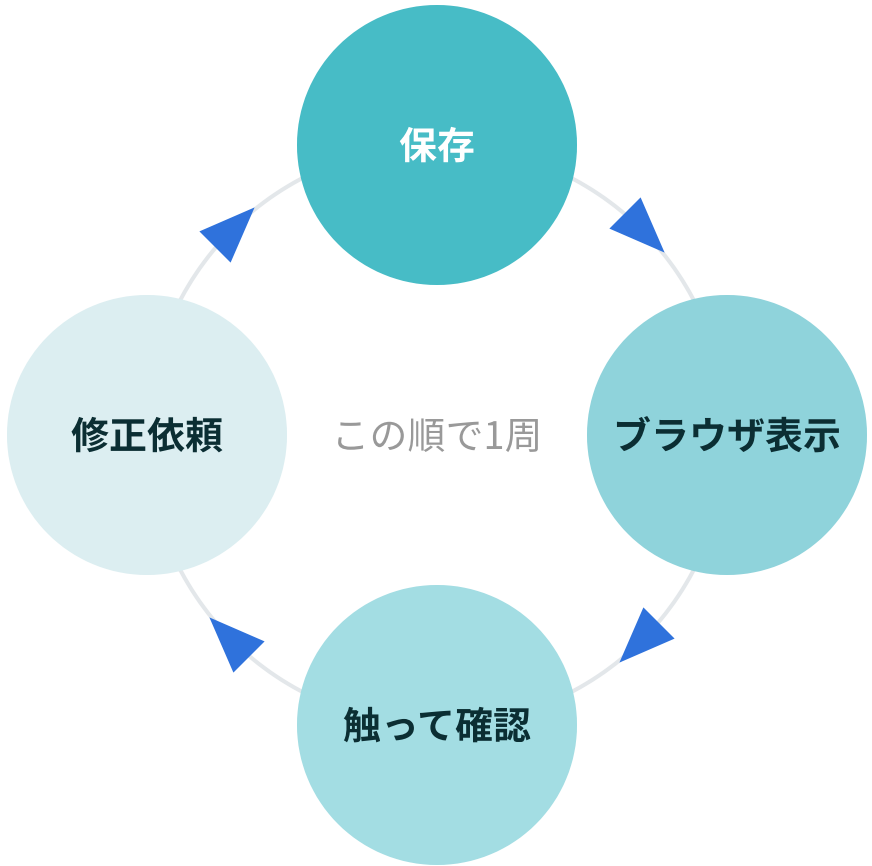
フォルダを開いた直後の画面



ワーク

ブラウザで動作確認と修正

実際に触ってから直す順番



ブラウザで開いた画面を、実際に触って確かめます。

壊れたときの戻し方

直す前のファイルを別名でコピーしておきます。
うまく動いた時点で 01.html のように残します。
戻したいときは、そのファイルを開き直します。

補足

画面の確かめ方

保存してからブラウザを更新する順番

01 **Antigravityで直す**
依頼した内容が反映されるのを待ちます。

02 **保存する**
Windows は Ctrl + S、Mac は Command + S。保存しないと画面は変わりません。

03 **ブラウザに切り替える**
アプリを開いてあるタブに移ります。閉じてしまった場合は、ファイルをもう一度ダブルクリックします。

04 **更新する**
Windows は F5、Mac は Command + R。ここで直した内容が画面に出ます。

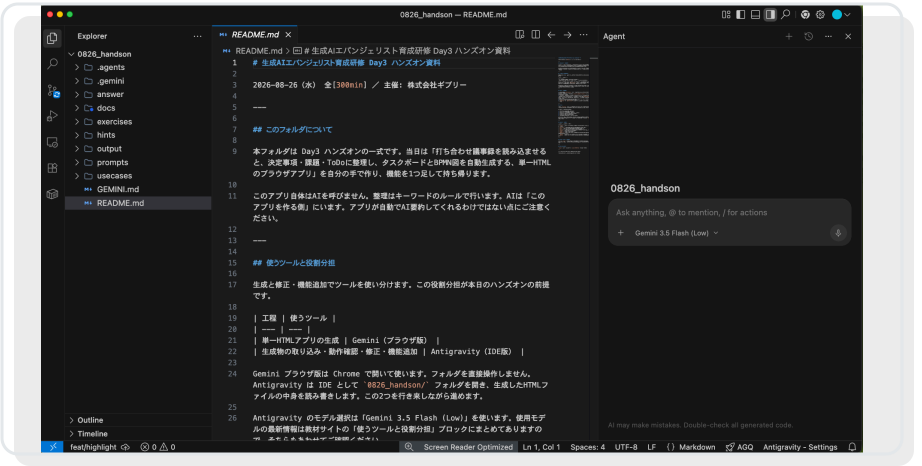
ライブプレビューの拡張機能について
Antigravity の中に画面を出す拡張機能もありますが、本日は使いません。入れると再起動が必要になるためです。

ワーク

Antigravityで機能追加と見た目の変更

1つ足して動作を確認してから次の依頼

要素	書き方の例
症状	担当者が未設定のカードが 左端に寄ってしまう
期待する動き	未設定のカードは最後の列に まとめて表示したい
該当箇所	index.html のうち、カードを 並べ替えている部分



エージェントに依頼文を送り、
返ってきた差分を確認します。

1回1変更の理由

まとめて頼むと、どの依頼で壊れたのかが分からなくなります。
1つ足すたびにブラウザで確認しておけば、直前の状態まで戻せます。

ワーク

成果物

この場のゴールは改修要望の聞き取りまで

01 企業から聞き取った改修要望

何のために使うのかとセットで、振り返りシートに書き出します。

02 図と表記の読みやすさ

BPMN図とカンバン表記が、相手に見せて伝わるかを確かめます。

03 実装するもの1つ

聞き取った要望から1つ選び、ロープレのあとに実装します。

自身で作成・改良した議事録オーガナイザーを用いてヒアリングを行い、BPMN図やカンバン表記が
理解しやすい・または改良点が具体的に見えることが目的です。
全体で [40min]。2名のペアをつくり、役を交代しながら2ラウンド回します。

ワーク

ロープレのポイント

クライアント役が6つの観点で評価します

1	**聞き取り**	要望をそのまま受けず、誰がいつ何のために使うのかを確かめられたか
2	**深掘り**	要望の裏にある困りごとまで聞けたか
3	**説明**	画面を見せながら、相手に分かる言葉で話せたか
4	**プレゼン**	行ったこと（何をしたか）をわかりやすく伝えられているか
5	**プレゼン**	意図・背景を明確に伝えられたか
6	**区切り**	完成版ではないことと、次にやることを伝えて終われたか

評価を書くのはクライアント役です。ヒアリング役は聞き取った要望と、実装するものを書きます。
シートは output フォルダの「ロープレ振り返りシート」。ラウンドごとにシートが分かれています。

ワーク

場面の設定

前回の相談を形にして見せに行く3回目の打ち合わせ

前の2回　業務を棚卸しし、タスクの一覧とBPMN図を手作業で作りました。
「毎回手でやるのは無理」という一言を受けて、今日はその形を持ってきています。

クライアント役

立場

製造業の情報システム部
DX推進の担当者
エンジニアではありません

この場ですること

タスクの洗い出しが手作業
BPMN図を描くのに半日
図を描ける人が2名だけ

ヒアリング役

立場

ギークス様のDX営業担当
エンジニアではありません
前の2回を担当した本人です

この場ですること

自分で作ったアプリを見せる
改修要望を聞き取る
完成版ではないと伝える

30分の打ち合わせのうち、画面を見せてから要望を受けるまでの [5min] を切り出した設定です。

ワーク

ロールプレイングの流れ

実演と準備をはさんで2ラウンド回します

説明 全体	進め方とポイントの確認。ここでハンドアウトを開きます。	**[5min]**
実演 全体	聞き取った要望をどう実装するかを、講師が画面で見せます。	**[5min]**
準備 グループ	役割の分担を決めて、自分の役のハンドアウトを読みます。	**[3min]**
ラウンド1・2 グループ	1ラウンドは打ち合わせ [5min] と振り返り [5min]。役を交代します。	**[20min]**
実装 各自	聞き取った要望から1つ選んで直します。終わらなければ持ち帰りです。	**[7min]**

合図はすべて講師が出します。残り時間もお伝えします。

ワーク

役割の交代

2名のペアがラウンドごとに役を入れ替えます

ラウンド	使うハンドアウト	A さん	B さん
ラウンド1	C-1／H-1 図の見せ方への要望	A ヒアリング役	B クライアント役
ラウンド2	C-2／H-2 タスクの追いかけ方と実データ	A クライアント役	B ヒアリング役

1ラウンドは打ち合わせ [5min]、振り返り [5min] です。合図は講師が出します。
3名の組は、1名がレビュー役に回ります。ラウンドごとに交代してください。
振り返りシートはラウンドごとに分かれています。該当するシートを開いてください。

ワーク

進行のルール

進行を止めないための5つの決めごと

1. **ハンドアウトは自分の役の分だけ開きます**
   クライアント役とヒアリング役では、書いてある内容が違います。

2. **クライアント役は自分の言葉で相談します**
   ハンドアウトを読み上げず、困っている本人として話してください。

3. **ヒアリング役はこの場でアプリを直しません**
   聞き取りに集中します。直すのはロープレが終わってからです。

4. **ヒアリング役は最後に必ず区切りを伝えます**
   完成版ではないことと、次に何をするかを言ってから終わります。

5. **1ラウンドは [5min] で必ず止めます**
   話の途中でも打ち切ります。終わらないこと自体が持ち帰る材料です。

ハンドアウトは配布ZIPの roleplay フォルダにあります。自分の役のファイルだけ開いてください。

ワーク

ラウンド1

要望を聞き取って次にやることを決めます

使うハンドアウト

- C-1 図の見せ方を相談する
- H-1 プロトタイプを見せる

このラウンドの役割

- A ヒアリング役
- B クライアント役

画面を見せて、前の2回の話と合っているかを確かめる

要望が出たら確かめること

誰がいつ見るのか	どの場面で、どなたが見るものになりますか
何が困っているのか	いまはそれを、どうやって回されていますか

打ち合わせ [5min]

要望が出たら、その場で直さずに聞き取ります。

振り返り [5min]

クライアント役が記入

ワーク

ラウンド2

要望を聞き取って次にやることを決めます

使うハンドアウト

- C-2 タスクの追いかけ方を相談する
- H-2 要望を掘り下げる

このラウンドの役割

- A クライアント役
- B ヒアリング役

要望を聞き、実際の議事録を受け取って扱えるかを確かめる

要望が出たら確かめること

いまはどうしているか	いまは、その宿題をどうやって追いかけていますか
どこまで必要か	全員分が見えればよいですか、自分の分だけですか

打ち合わせ [5min]

実データが分類されなくても直しません。原因を言葉にして持ち帰ります。

振り返り [5min]

クライアント役が記入

ワーク

聞き取った要望の実装

2ラウンドで出た要望から1つ選んで直します

01 選ぶ
ヒアリング役として書いた要望から、いちばん効きそうなものを1つ選びます。

02 依頼する
Antigravity に1文で頼みます。まとめて頼まず、1つだけにします。

03 確かめる
保存してから、ブラウザを更新します。この順番でないと画面は変わりません。

04 残す
終わらなければ持ち帰りです。どこまでやったかをシートに書いておきます。

[7min] で終わらなくて構いません。持ち帰って続きをやるところまでが、この演習です。
手が止まったら、講師を呼んでください。

ワーク

ロープレの振り返り

ロープレで見えた3点の洗い出し

01 詰まった場面

要望を聞き取るとき、どこで手が止まったかを挙げます。

02 足りなかった機能

聞き取った要望のうち、まだ直せていないものを挙げます。

03 明日の一手

自分の担当業務で次に試すことを、1つだけ決めます。

組の中で [8min]。1人 [2min] で話し、残りの時間で持ち帰るものを整理します。
出てきた案は output フォルダの振り返りシートに書き戻してください。

今日の持ち帰り

明日の商談で使う1文と開発で試す1手

AIの出力を良くするのは指示の量ではなく、
合否が返る確かめ方の有無

テスト・ビルド・画面での確認など、機械が合否を返す仕組みを先に渡します。
渡せない作業は、人が確かめる時間を工程の中に残しておきます。

出所: Anthropic Claude Code 公式ドキュメント Best practices（2026年8月時点）

商談で使う1文

仕様を先に固めるほど、AIの出力は安定します。

開発で試す1手

不具合を再現するテストを先に書かせます。

質問

質疑

その場の疑問とあとで確かめる一次情報

質問の出し方

気になったことは、挙手でもチャットでも構いません。
案件で使えるかどうかの相談は、状況を添えていただけると答えやすくなります。
その場で答えを持たないものは、確認先の一次情報をお伝えします。

持ち帰り用の一次情報

AGENTS.md 公式サイト	https://agents.md/
Claude Code メモリ管理	https://code.claude.com/docs/en/memory
Kiro 仕様駆動開発	https://kiro.dev/docs/specs/
GitHub Spec Kit	https://github.com/github/spec-kit
Antigravity のルール	https://antigravity.google/docs/rules-workflows
Model Context Protocol	https://modelcontextprotocol.io/

APPENDIX

参考資料

当日は投影しない、持ち帰り用のページ

075

研修時間の都合で本編から外したページをまとめています。仕様駆動開発とテスト駆動開発の詳細、ハーネスの構成要素、ツールの操作手順、企業事例の深掘りが入っています。
社内で共有するときは、この章の図と出典をそのままお使いいただけます。

人が書く比率より確かめる比率が上昇

自社にマージされるコードの80%超をClaudeが記述

Anthropic が自社の開発体制について2026年5月に公表した数値です。

2年 → 3か月
Chromeチームの開発期間

8倍
1人が1日にマージするコード量（2024年比）

利用の広がりに信頼が追いついていない現状

84%
AIツールを利用または利用予定

32.7%
出力の正確さを信頼

45.2%
デバッグに想定以上の時間

出所: Anthropic「Recursive self-improvement」（2026年5月） / Alphabet 2026年第2四半期 CEOメッセージ（2026年7月）
出所: Stack Overflow 開発者調査2025（2025年12月公開、AIに関する設問）

開発の進化段階

補完から対話を経て自律実行へ移った3段階

01　補完
エディタ内で候補を提示

02　対話
チャットで指示して生成

03　自律
計画から実行までを委任

エディタ内の候補提示
次の数行を提案する範囲

人が主導する前提
設計も記述も人の担当

2025年2月2日
「バイブコーディング」の初出

2025年11月6日
Collins英語辞典の年間語に選出

2025年9月25日
Copilot coding agent 一般提供

2025年11月18日
Google Antigravity 発表

Martin Fowler は、生成コードを読まない進め方をバイブコーディングと定義し、使い捨てのソフトに限るよう警告しています。

出所: Wikipedia「Vibe coding」（Karpathy の投稿と Collins Dictionary の発表を記載）
GitHub Changelog / Google Antigravity 公式ブログ / martinfowler.com bliki: Vibe Coding

大量・定型・下書き・言い換えは機械側が優位

大量の文章の要約
長さが増えても処理速度が落ちません。

網羅的な洗い出し
抜けの発見は、量を出せるほど有利です。

定型文書の下書き
型の決まった文書ほど手間が消えます。

既知パターンの実装
前例の多いコードほど精度が上がります。

形式の変換
表・箇条書き・コードへ機械的に移せます。

テストケースの列挙
条件の組み合わせを機械的に並べられます。

共通する条件は、正解が一つに決まり、やり直しの効く作業であることです。

人間の得意なタスク

決める・責任を取る・現場の事情を持ち込むのは人の担当

目的の設定
何のためにやるのかは、依頼した側が決めます。

関係者の合意形成
立場の違う人をそろえる仕事です。

優先順位の決定
何を先にやるかは、期限と体制で変わります。

例外処理
前例のない事象は、その場の判断が要ります。

妥当性の判断
出てきた案が現場で通るかを見極めます。

最終責任
出した成果の責任はAIには渡せません。

共通する条件は、正解が一つに決まらず、外したときの影響が大きいことです。

仕様駆動開発の流れ

先に固定する3文書と差し替え可能な実装

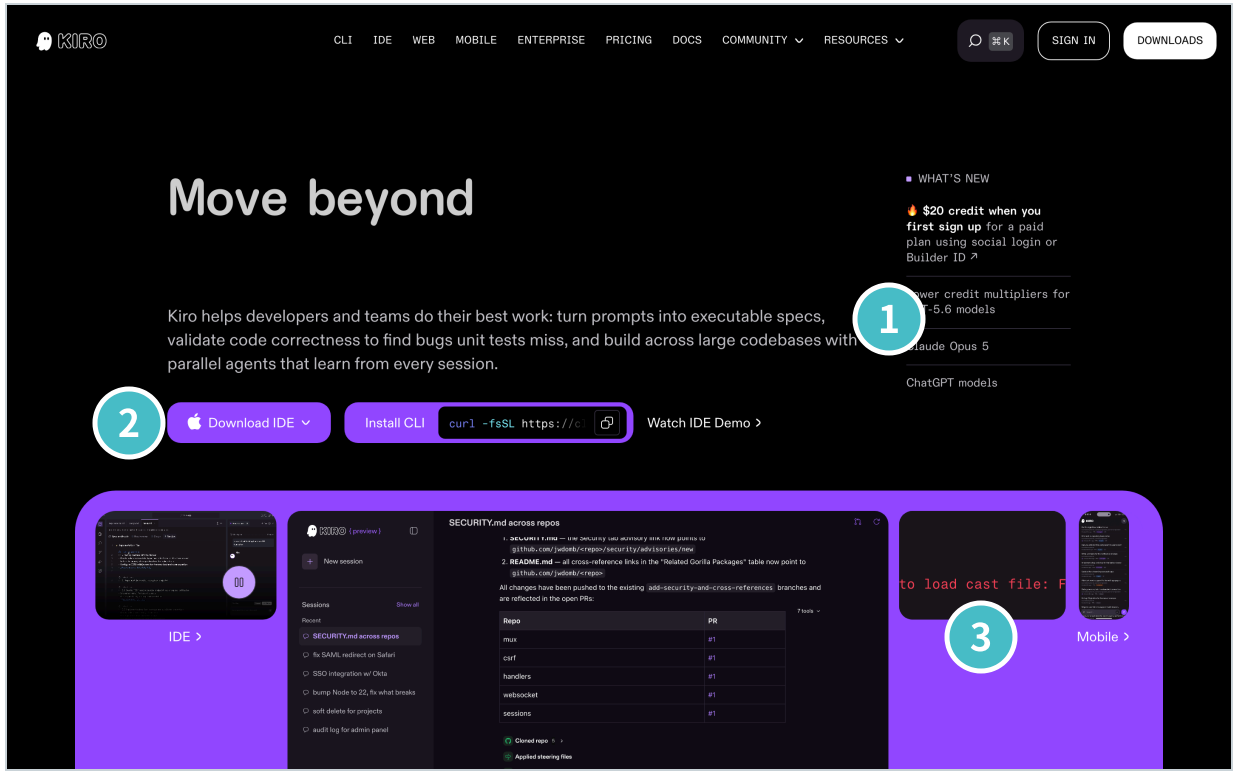
前の3工程が承認の対象
要件・設計・タスクの区切りで内容を人が読み、通ってから次の工程へ進みます。

3つの文書を残す意味
実装を捨てても3文書が残るため、作り直しの指示は文書の修正だけで済みます。

各ノードの下段が生成物です。ファイル名はツールごとに異なり、次のページで実物を確認します。

Kiro式の3ファイル

requirements・design・tasks の3枚で仕様を固定



出所: Kiro 公式サイト・公式ドキュメント（kiro.dev）2026年8月時点

1. プロンプトを実行可能な仕様へ
2. IDE版とCLI版の2つの入口
3. IDE・CLI・Mobileの実画面

specの3ファイル

`.kiro/specs/{機能名}/`

requirements.md
EARS記法で条件と振る舞いを書きます。
`WHEN 条件 THE SYSTEM SHALL 振る舞い`

design.md
アーキテクチャと設計方針を残します。

tasks.md
順序付きの実装タスクを並べます。

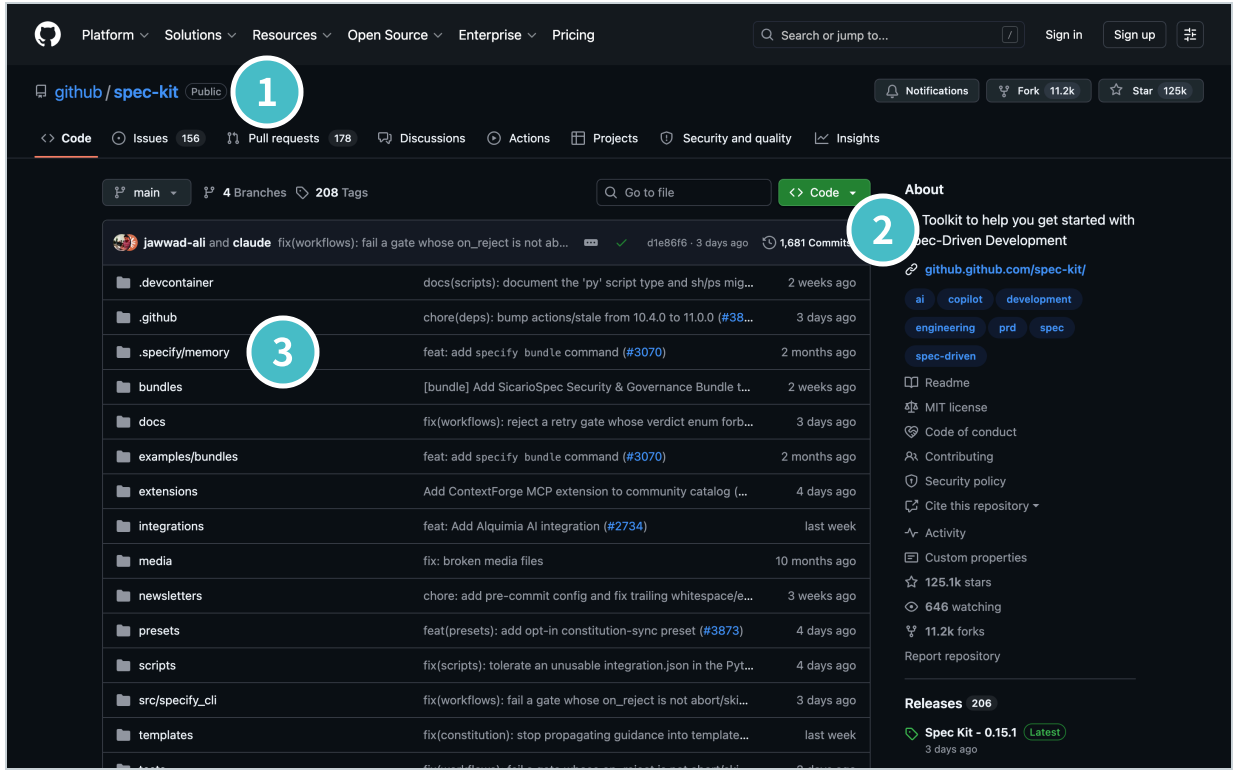
GitHub Spec Kit のコマンド体系

仕様から実装までを決まった順番のコマンドで進行

1. GitHub公式のOSSリポジトリ
2. 仕様駆動開発のツールキット
3. .specify/memory に開発原則

主要なコマンド

/speckit.constitution	開発原則の定義
/speckit.specify	仕様の作成
/speckit.plan	技術設計と計画
/speckit.tasks	タスク分解
/speckit.implement	実装の実行



Copilot・Claude・Codex など30以上に対応します。

任意コマンドとして /speckit.clarify と /speckit.analyze があります。

出所: spec-kit 公式README（2026年8月時点）

IDE同梱と既存ツールへの後付けという違い

比較項目	Kiro（AWS）	Spec Kit（GitHub）
公開	2025年11月17日に一般提供	2025年9月2日にOSS公開
形態	仕様駆動開発IDE本体を導入	既存エージェントへ後付け
主な成果物	`requirements.md / design.md / tasks.md`	`spec.md / plan.md / tasks.md`
常設の前提	`.kiro/steering/` `product.md tech.md structure.md`	`.specify/memory/` `constitution.md`
動かす場所	Kiro本体（IDE版・CLI版）	Copilot・Claude・Codex など30以上
費用の起点	無料50クレジット、Pro は月20ドル	OSSとして無償公開

出所: kiro.dev 公式ドキュメント・料金ページ / spec-kit 公式README・公式ドキュメント（2026年8月時点）

TDDの手順

先にテストを書かせてAIの出力に合否判定を付与

	STEP 1 振る舞いの明確化	STEP 2 テスト作成の依頼	STEP 3 実装の依頼	STEP 4 通るまでの修正
人がやること	期待する結果の記述 入出力の例示	観点の過不足の確認 追加観点の指示	変更範囲の指定 触らせない箇所の指定	落ちた原因の確認 直す方針の判断
AIがやること	曖昧な点の質問 前提条件の確認	テストコードの作成 境界値の洗い出し	テストを通す実装 差分の説明	失敗ログの解読 修正案の提示

Anthropicの公式ガイドは、不具合を再現する失敗テストを先に書く進め方を推奨しています。

出所: Anthropic Claude Code 公式ドキュメント（Best practices）

AIとTDDの相性

出力の良し悪しを人が読まずに判定できる仕組み

テスト無しの生成

- 要件を満たすかは人が読んで判断します。
- 量が増えると確認が追いつかなくなります。
- 直した箇所の副作用に気づけません。

確認の負荷が人に残る構図

テスト有りの生成

- 期待する振る舞いが実行できる形で残ります。
- 合否が機械的に返り、AIが自分で直せます。
- 変更後も同じテストで確認できます。

確認をAIに返せる構図

> **「テスト駆動開発はAIの暴走を防ぐガードレールになる」**
>
> 和田卓人氏 ／ 出所: Agile Journey（ユーザベース） 2025年8月29日

AgentRules

毎回言い直す注意事項をファイルに書いて自動で適用

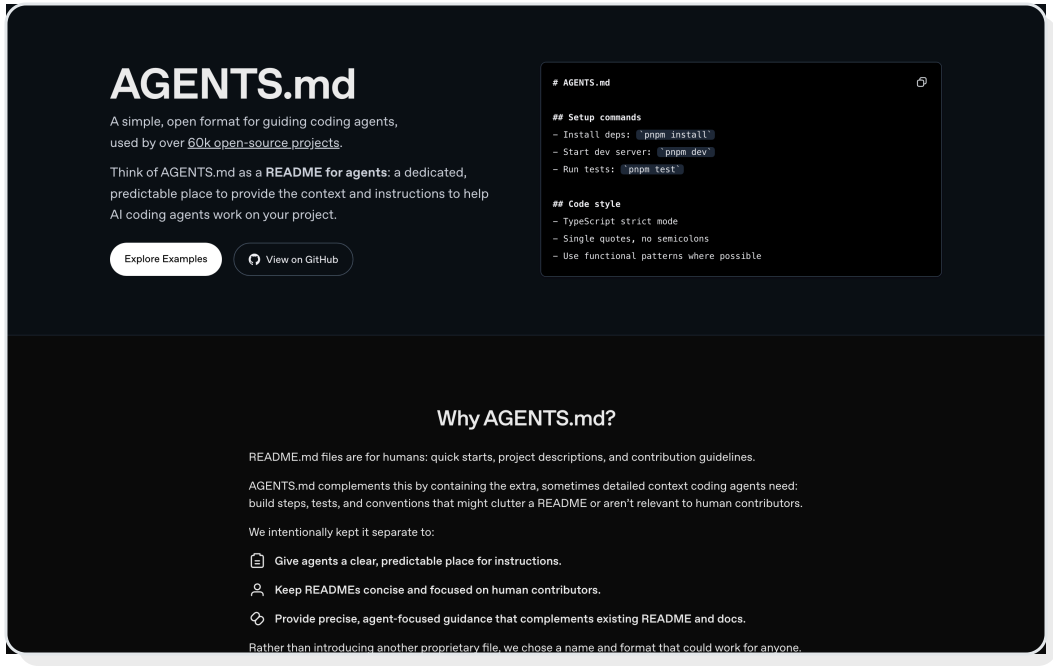
毎回プロンプトに書く運用

- 注意事項を人が毎回思い出して入力します。
- 書き忘れた回だけ品質が落ちます。

ルールファイルに書く運用

共通形式	AGENTS.md
Claude	CLAUDE.md
Copilot	.github/copilot-instructions.md
Cursor	.cursor/rules/*.mdc

Claude Code は @AGENTS.md と書いて取り込みます。



AGENTS.mdはリポジトリのルートに置く
1枚のMarkdownです。23ツールが読み込みます。

出所: agents.md 公式サイト

成果にたどり着くための情報がそろっているかが分かれ目

AIが外す典型	渡しておくもの
社内用語の取り違え 略語や通称をそのまま別物として扱います。	**用語集** 略語と正式名称の対応表を渡します。
既存の作法の無視 その場限りの書き方で実装します。	**該当箇所のコード** 似た機能の実装例を指し示します。
決定済みの案の蒸し返し 見送ったはずの案を再び出します。	**議事録と決定事項** 見送った案とその理由も残します。

渡す情報は多いほど良いわけではありません。過剰な情報はかえって精度を下げます。

出所: LayerX エンジニアブログ

Skill

成功した手順を書き留めて次から同じ品質を再現

うまくいったやり取り
その場では成功しても、
次に残りません。

→

手順としての書き出し
入力・手順・完了条件を
文章にして残します。

→

次回からの呼び出し
同じ手順を同じ品質で
再現できます。

Skillファイルの置き場所と中身

`.claude/skills/<名前>/SKILL.md` プロジェクト単位で置く場合
`~/.claude/skills/<名前>/SKILL.md` ユーザー共通で置く場合

入力
何を受け取るか

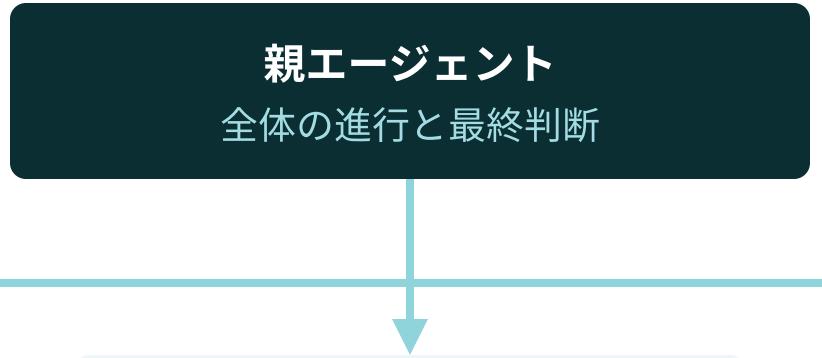
手順
どの順で何をするか

完了条件
何がそろえば終わりか

出所: Anthropic Claude Code 公式ドキュメント（Skills）

Subagents

役割ごとに担当を分けた構成で精度が安定



調査担当

権限　読み取りのみ

返すもの　要点の要約

レビュー担当

権限　差分の閲覧

返すもの　指摘の一覧

テスト担当

権限　テストの実行

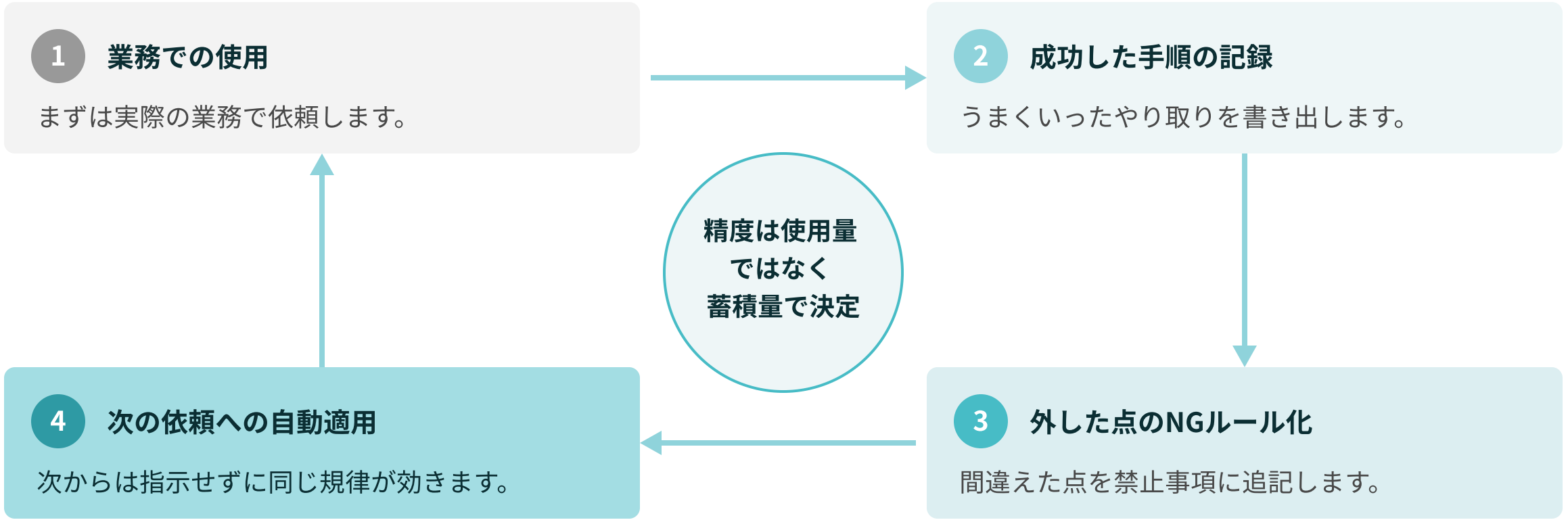
返すもの　失敗した箇所

> 子は独立した文脈で動き、親には要約だけを返すため、本筋の文脈が汚れません。

出所: Anthropic Claude Code 公式ドキュメント（Subagents）

AIツールの精度を上げる回し方

使い続けて成功をルール化した分だけ精度が上がる仕組み



ルールは足すだけでなく、効かなくなったものを削る運用も必要です。

コンテキストの重要性

自由度が高いほど渡した情報の欠けが外れに直結

> **その成果にたどり着くために必要な情報が、全部そろっているか**

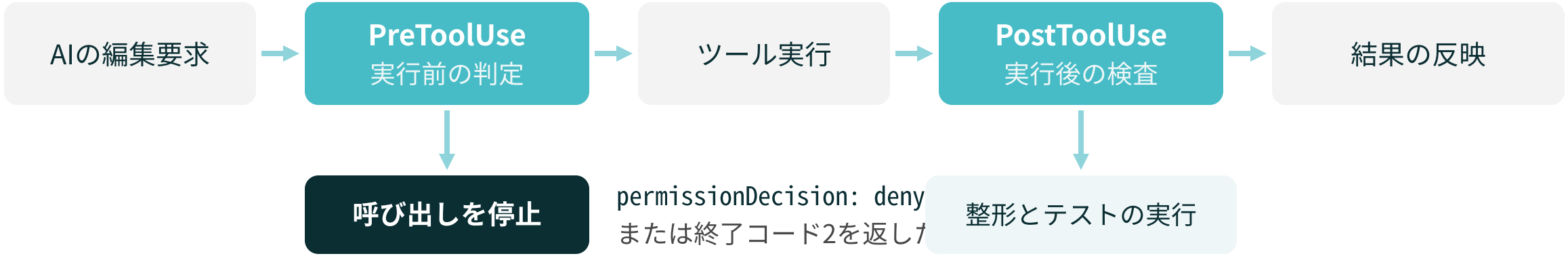
欠けやすい情報

1	**前提**	誰のための作業か、いつまでに必要か、何をもって完了とするか
2	**制約**	使ってよい環境、外に出してはいけない情報、触らない範囲
3	**既存の作法**	これまでの書き方、社内での呼び方、前回そう決めた理由

出所: Anthropic Engineering ブログ（2025年9月29日・コンテキストエンジニアリングの定義）

生成直後のセルフチェック機構

必ず実行させたい処理をHooksで強制



settings.json の記述例

```
{ "hooks": {
"PreToolUse": [ {
"matcher": "Write|Edit",
"hooks": [ { "type": "command",
"command": ".claude/deny_secret.sh" } ]
} ] } }
```

出所: Anthropic Claude Code 公式ドキュメント（Hooks / Memory）

決定論的に止まる理由

Write と Edit の直前に検査が走ります。
終了コード2を返すと編集は実行されません。
AIの判断ではなく設定で毎回動きます。

Geminiブラウザ版と他ツールの比較

ブラウザ版の強みは調査と下書きで弱みはファイル操作

ツール	形態	ファイルの扱い	得意	費用の起点
Gemini（ブラウザ）	ブラウザ・アプリ	添付して読ませる 1回10ファイル・100MBまで	Deep Research Canvas、Gem	無料は0円 Plus 月725円〜
ChatGPT（ブラウザ）	ブラウザ・アプリ	添付して読ませる PCへの書き戻しは不可	文章の下書き アイデア出し	無料は0ドル Go は月8ドル
Claude（ブラウザ）	ブラウザ・アプリ	添付して読ませる PCへの書き戻しは不可	長文の読解 文章の整理	無料プランあり Pro は月20ドル
GitHub Copilot Chat	エディタ内の チャット	エディタで開いている ファイルを参照	コードの補完 コードの説明	Free は0ドル Pro は月10ドル

※ 2026年8月3日時点。料金と提供条件は変動します。

出所: Gemini／OpenAI／Claude／GitHub Copilot の各公式プランページ

Antigravityと他ツールの比較

エージェント型ツールの中で広い無料枠が特徴

ツール	提供元	形態	選べるモデル	無料枠	向く使い方
Antigravity	Google	デスクトップアプリ CLI・IDE・SDK	Gemini・Claude・ gpt-oss-120b の6種	個人向け 0ドル 週単位の制限	エージェントに まとめて任せる
Claude Code	Anthropic	ターミナル中心 IDE拡張・ブラウザ	Anthropic のモデル	無料枠なし Pro は月20ドル	既存コードの 読解と改修
OpenAI Codex	OpenAI	CLI・IDE拡張 Web・クラウド	OpenAI のモデル	Free は0ドル Go は月8ドル	クラウド側で まとめて処理
Cursor	Cursor	AI前提のエディタ	複数社のモデル	Hobby は無料 Pro は月20ドル	エディタ操作を AIへ寄せる
GitHub Copilot	GitHub	VS Code などの エディタ拡張	複数社のモデル	Free は0ドル 補完2,000回まで	日常の補完と コードの説明

※ 2026年8月3日時点。選べるモデルと無料枠はプランによって異なります。
出所: 各社公式のダウンロード・料金・ドキュメントページ

調査のGeminiと実装のAntigravityで分かれる役割

Gemini

ブラウザで完結する調査と下書き

無料プランでも Deep Research・Canvas・Gem を利用可能
1プロンプトあたり10ファイル・1件100MBまで添付
Deep Research は調査計画・ウェブの自動横断・レポート生成の3段構成

Antigravity

Google

手元のファイルを触る実装と検証

個人向けプランは月0ドル、レート制限は週単位で回復
Gemini・Claude・gpt-oss-120b の6モデルから選択
作業結果はタスクリスト・実装計画・スクリーンショットで確認

出所: Gemini 公式プラン紹介ページ / Antigravity 公式ブログ・料金・モデルページ（2026年8月3日取得）

押さえる操作は モデル選択・添付・Deep Research・Gem の4つ

- **モデル選択**
  無料プランでは Gemini 3.6 Flash と、
  制限つきの 3.1 Pro を選べます。
- **ファイル添付**
  1プロンプトあたり最大10ファイル、
  動画は1本2GB、他は100MBまでです。
- **Deep Research**
  調査計画の作成、ウェブの自動横断、
  レポート生成の3段で進みます。
- **Gemの作成**
  左メニューの「Gemを探す」から作り、
  役割・やること・背景・出力形式を書きます。

> プレビューを使っただけではGemは保存されません。
> 必ず保存ボタンを押してください。

出所: Gemini アプリ ヘルプセンター（2026年8月）

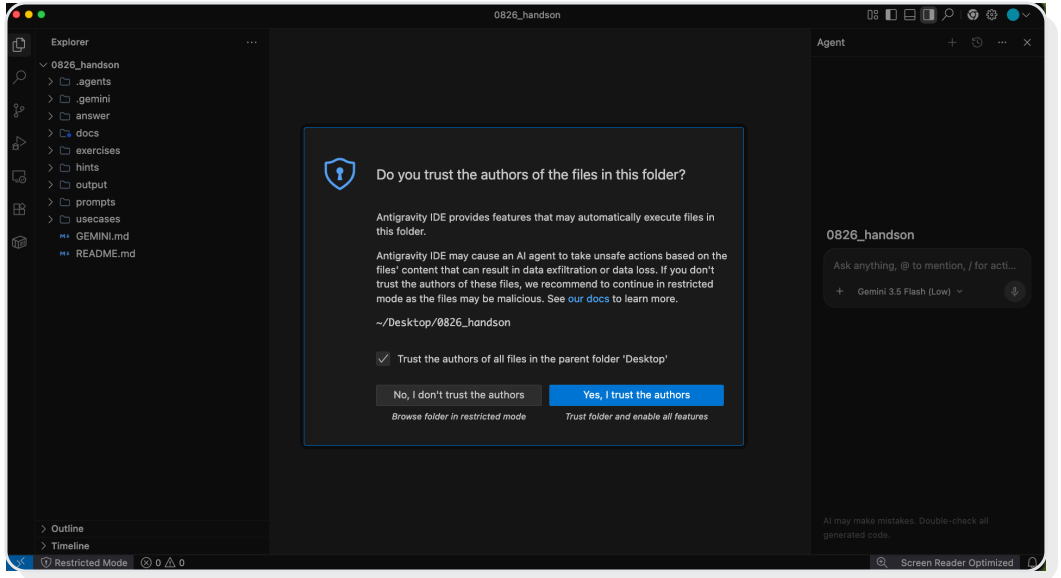
1 コードの受け取り

2 ブラウザで確認

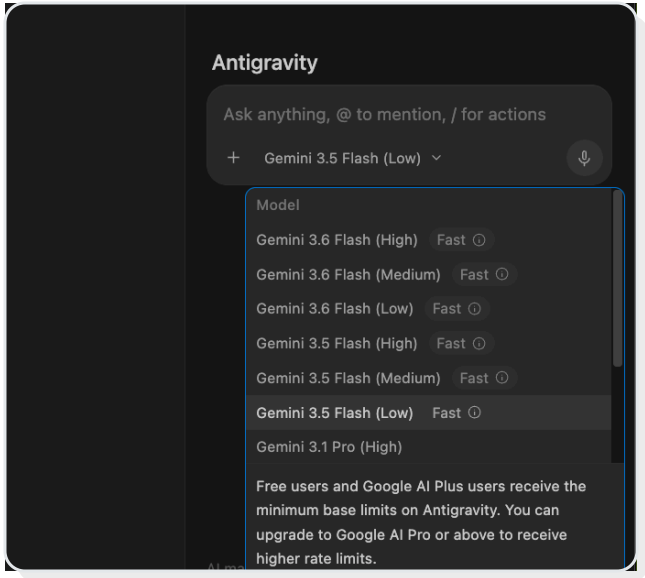
Antigravityの基本操作

フォルダを開いてモデルを選ぶまでが依頼前の準備

1. **フォルダを開く**
   Open Folder
2. **モデルを選ぶ**
   6モデルから
3. **依頼を送る**
   入力欄から
4. **差分を確認**
   1行ずつ承認



手順1 フォルダを開いた直後の信頼確認



手順2 モデル選択のドロップダウン

出所: Google Antigravity 公式ドキュメント（2026年8月）

活用事例の読み方（ローム）

効果が出たのは特定業務への当て込み

当初の予定

1年

実績

1週間

同じ導入で公表された数字

展開規模 900名以上

1人あたり月31時間の作業短縮見込み

導入したもの Google Workspace with Gemini

LSI開発部門では、1年かけて実施予定だったコード解析・改善プロジェクトが、
Gemini の活用により1週間で完了しました。展開そのものは900名以上の規模です。

この数字の読み方

- 対象は全社ではなく、LSI開発部門の単一プロジェクトに絞られています。
- 効果の単位が時間なので、自社の業務に置き換えて見積りができます。
- 出所はGoogleの公式ブログで、社名と数値が当事者名義で公開されています。

出所: Google Workspace ブログ（ローム、2026年6月公開）

AIコーディングの企業事例

各社の公表値は工数・生成比率・削減時間という測れる単位

企業	使い方	公表された効果
サイバーエージェント	Claude Code を全社で利用 利用者は約2,000人	96プロダクトで要件定義・開発の 工数3割減、調査工数は約75%減
メルカリ	社員の95%がAIツールを 日常的に利用	コードの約70%をAIが生成 開発スピードは前年比64%向上
DeNA	2018年製のレガシーAPIを Devin中心で移行	Perl約6,000行をGo約10,000行へ Devin80%・Claude Code15%・人5%
ZOZO	AI駆動開発を2つの スラッシュコマンドに集約	Jiraチケットから設計書と 進捗表を生成する流れを標準化
GMO インターネットグループ	全社の利用状況と 削減時間を調査	削減時間は月43.2万時間 1人あたり約65.1時間

出所: CyberAgent Developers Blog / メルカリエンジニアリング / DeNA Engineering Blog
ZOZO TECH BLOG / GMOインターネットグループ公式リリース（2025年12月〜2026年7月公開）

レビュー・テスト・ドキュメントの事例

人が全部読む前提を外して一次判定をAIへ寄せる構図

現場の課題	打ち手と結果
レビューの観点が 人によって変わる	**メルカリ（メルペイQAチーム）** Claude Code Skills の /review-pr と /improve-review-pr を運用。修正PRから観点を追記
3人体制でアラートを 追い切れない	**ZOZO（情報セキュリティ部）** 3人体制のSOCで、Tier1相当の アラートトリアージをClaude Codeのエージェントで全自動化
仕様レビューに 2週間〜1か月かかる	**DeNA（SDKチーム）** Claude Codeで仕様書生成とプロトタイプ確認を回し、 レビューのサイクルは1〜2日へ短縮

出所: メルカリエンジニアリング / ZOZO TECH BLOG / DeNA Engineering Blog（2026年6〜7月公開）

引用とファクトチェックの手順

数値を載せる条件は一次情報の本文での確認

1	**出典URLの参照**	検索結果の要約で済ませず、記事本文をブラウザで開きます。
2	**本文での数値照合**	書こうとしている数字が、本文に載っているかを目で確かめます。
3	**一次情報の判別**	発表元の公式ブログ・リリースか、まとめ記事かを見分けます。
4	**未確認値の除外**	一次情報で確認できない数字は、そのページから外します。

実例: 一度落として、公式ブログで拾い直した数値

「利用率95%・AI生成コード70%・開発速度+64%」は、まとめ記事しか見つからず一度落とし、
公式エンジニアリングブログで本文を確認できた段階で採用しました。出所: メルカリエンジニアリング

失敗事例① コストの暴発

上限を設定していても**請求が止まらない経路**が存在

Gemini API の課金バグ

1,000万円超

日本の開発者に発生した誤請求です。
1日あたり約150万円ずつ増え、
APIキーを全削除しても継続しました。

出所: Google AI Developers Forum（2025-09-02）、The Register（2026-05-18）

上限設定が効かない3経路

1 **APIキーの侵害**

上限250ドルを設定していた利用者に1万ドルが請求され、
別の利用者はAPIキー侵害で12万7,000ドルでした。

2 **監視の外を通る課金経路**

AWS Bedrock は AWS Marketplace 経由で課金され、
コスト異常検知が働かず3万〜3万8,000ドルの請求です。

3 **提供側の不具合**

Gemini API のバグではAPIキーを全て削除しても
請求が増え続け、上限設定では止められませんでした。

失敗事例② 出力の未検証

AIの誤りは使った側の責任として処理

出来事	経緯	結果
Mata v. Avianca 米連邦地裁 2023-06-22	ChatGPT が生成した架空の判例6件を、 実在するものとして裁判所へ提出しました。	弁護士2名と事務所に連帯で 5,000ドルの制裁金
デロイト豪州 豪州政府向け 2025-10	生成AIによる架空の引用を含む報告書を 豪州政府へ納品しました。	契約額29万ドルのうち 一部を返金
Moffatt v. Air Canada BC州 CRT 2024-02-14	チャットボットの誤案内を信じた利用者が、 割引運賃を受けられませんでした。	812.02カナダドルの 支払命令

> エア・カナダの「チャットボットは別人格」という主張は退けられ、サイト上の情報は自社の責任と判断されました。

出所: Mata v. Avianca（S.D.N.Y. 2023-06-22）／Moffatt v. Air Canada, 2024 BCCRT 149／Fortune（2025-10-07）

ワーク

生成AI活用事例の調査Gem作成

引用付きの調査を毎回同じ手順で回すGem

目的	Geminiでの事例調査を毎回同じ品質にするため、指示をGemとして固定します。 キーワードを入れ替えるだけで、同じ手順の調査が回せる状態を目指します。
生成物	「生成AI活用事例リサーチ」という名前のGem 1本。手順欄に役割・調査範囲・ 出力形式・引用の付け方・確度の書き分けの5点を書き込みます。
OK基準	キーワードを1語入れて実行し、出典URL付きの事例が3件以上返ること。 数値には必ず出所と公開時期が付いていること。
実施	研修後に各自で。目安はGemの作成と動作確認をあわせて20分程度です。

保存を押すまでGemは残らない仕様

プレビューで試しただけでは保存されません。出所: Google Gemini アプリ ヘルプセンター

ワーク

Gemの作成手順

Gemを新規作成して手順欄を書き保存で完成

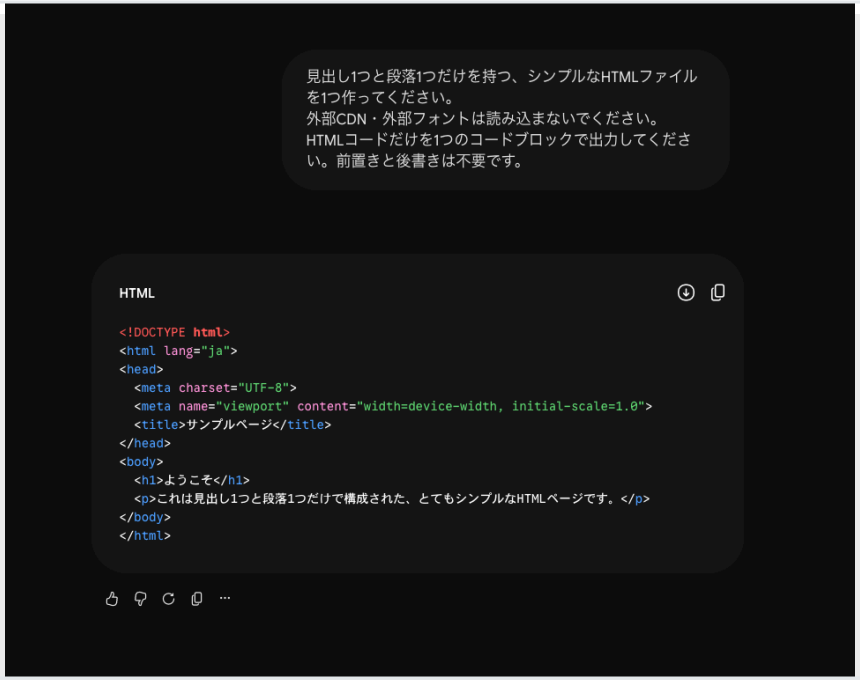
1. **新しいGemの作成**
   左メニューの「Gemを探す」から新しいGemを選びます。

2. **名前の設定**
   後から探せる名前を付けます。用途がわかる語を入れます。

3. **手順欄の記述**
   役割・調査範囲・出力形式・引用の付け方・
   確度の書き分けの5点を書きます。

4. **ナレッジの追加**
   参考にさせたい資料があれば添付します（任意）。

5. **プレビューと保存**
   右側で試し、「保存」を押して確定させます。



Gemの設定画面。名前と手順を入力します。

出所: Gemini アプリ ヘルプセンター

Gem用プロンプトへの書き換え

雑な一文から役割と出力形式を持つ指示へ

書き換え前

```
生成AIの活用事例をいくつか教えて
```

この指示の弱点

- 出所が付かない
- 企業名と数値の対応が崩れる
- 毎回ちがう形式で返る

書き換え後

```
# 役割
生成AI導入事例のリサーチャー
# 調査範囲
公式ブログ・プレスリリースなど一次情報
# 出力形式
企業名 / 使い方 / 数値 / 出典URL / 公開時期
# 引用の付け方
数値には必ず出典URLと公開時期を付ける
# 確度の書き分け
一次情報は確認済み、それ以外は未確認と明記
```

手順欄はあとから編集できます。運用しながら、外した観点や欲しい列を足していきます。

株式会社ギブリー（Givery, Inc.）

〒150-0036 東京都渋谷区南平台町15-13 帝都渋谷ビル8F