

あなたもできる Codex・ Claude Code

何ができる？どっちを使う？ 一歩上のAI活用法

仕事での使い分けがわかる

安田 大介

本書を読む前に

本書は、CodexとClaude Codeのどちらが常に優れているかを決めるものではありません。仕事の種類、利用環境、確認方法、安全性から、自分に合う入口を選ぶための実用書です。

本文中のケーススタディは、同一条件で性能を競わせた比較実験の結果ではありません。各製品の公式情報を基に、同じ仕事へ取り組む場合の進め方と、人間が確認すべき点を整理したものです。

AIサービスの画面、機能、料金、対象プラン、利用上限は変更される可能性があります。契約、課金、秘密情報の送信、ファイル削除、サーバー公開などを行う前に、必ず各社の最新の公式情報と自分の契約画面を確認してください。

本書に掲載した操作例や指示例を利用するときは、元データのバックアップを取り、対象範囲と権限を確認し、重要な結果を人間が検証してください。本書の情報のみを根拠として、法律、会計、医療、セキュリティその他の専門判断を行わないでください。

Codex、ChatGPTおよびOpenAIはOpenAIの商標または名称です。Claude、Claude CodeおよびAnthropicはAnthropicの商標または名称です。その他の会社名、製品名、サービス名は各社の商標または登録商標です。本書はOpenAIまたはAnthropicによる公式出版物ではありません。

情報確認日：2026年8月28日

目次

本書を読む前に	2
はじめに	25
似て見える二つの道具を、勝ち負けではなく「仕事」で選ぶ	25
この本が比較実験をしない理由	27
非プログラマーにも関係がある理由	28
CodexとClaude Codeを理解するための前提	29
本書の読み方	30
最後に責任を持つのは人間	31
第1章 チャットAIとAI開発エージェントは何が違う？	33
答えるAIから、作業を進めるAIへ	33
チャットAIが得意なこと	35
AI開発エージェントが加えるもの	36
基本は「調べる・動く・確かめる」の循環	37
1 調べる	37
2 動く	37
3 確かめる	38
「AIが完了と言った」と「仕事が完了した」は違う	38
AIが見られる範囲は、最初から無限ではない	40
ローカル、クラウド、リモートを分けて考える	42
最低限知っておきたい七つの言葉	42

ファイル	42
フォルダ	43
コマンド	43
ターミナル	43
コードベース	43
テスト	43
Git	43
最初は「変更しない調査」から頼める	44
チャットAIとAI開発エージェントを使い分ける	45
この章のまとめ	45
 第2章 Codexとは何か	47
Codexは「コードを答えるAI」だけではない	47
「製品」と「モデル」を分ける	49
主な入口は一つではない	50
ChatGPTデスクトップアプリ内のCodex	51
Web・リモートからの利用	51
Codex CLI	51
IDE拡張	52
Codexクラウド	52
Codexが進める仕事の流れ	53
プロジェクトという考え方	54
ファイル以外の道具を使う場合	55
権限と承認は、邪魔ではなく境界線	56
非プログラマーが最初に試すなら	57

仕事から入口を選ぶ



製品名より先に、仕事の条件を見る

図01 仕事からCodexとClaude Codeを選ぶ——仕事、環境、権限を確認し、最後は人間が成果を確かめる。

図01を見てください。両者は勝敗で分けるのではなく、仕事への入り方で捉えると理解しやすくなります。

たとえば、「会社案内の文章を考えたい」という仕事と、「会社のWebサイト一式を調べて、表示崩れを直し、テストまで行いたい」という仕事は、同じAI活用でも性質が違います。前者は対話だけで進められるかもしれませんが。後者では、複数のファイルを読み、関係を理解し、変更し、実際に動かして確かめる必要があります。

また、同じWebサイト修正でも、普段使うデスクトップ画面から取り組みたい人と、ターミナルやIDEと呼ばれる開発画面で取り組みたい人では、使いやすい入口が違います。会社のデータを扱うなら、便利さだけでなく、どのファイルをAIが読めるのか、どのコマンドを実行できるのか、外部サービスへ何を送るのかも確認しなければなりません。

つまり、道具だけを比べても、自分に合う答えは出ません。仕事、環境、権限を一緒に考えて、初めて選べるのです。

本書の比較を読む順序

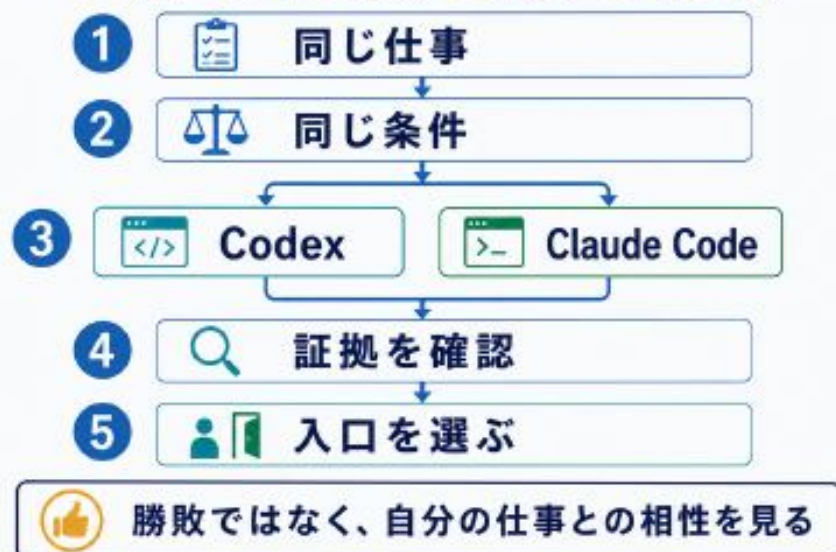


図01-2 本書の比較を読む順序——同じ仕事と条件から出発し、証拠を確認して一方または組み合わせを選ぶ。

図01-2は、本書の読み方そのものを示しています。各事例では、まず仕事と条件をそろえます。次に、CodexとClaude Codeで考えられる作業経路を同じ完成条件で見ます。最後に、成果物や確認記録を見て、その仕事に合う入口を判断します。

ここでいう「組み合わせる」は、二つの製品へ同時に同じ変更を任せることではありません。たとえば、一方で既存プロジェクトを調査し、その調査結果と未確認事項をもう一方へ引き継ぐように、役割と責任の境界を明確にした使い方です。

非プログラマーにも関係がある理由

「開発」という言葉を聞くと、黒い画面に難しい文字を入力するプログラマーの仕事を思い浮かべるかもしれません。しかし、AI開発エージェントが扱う対象は、プログラムだけではありません。



図02 回答から成果物へーチャットAIでは回答後の反映を人が行い、開発エージェントでは調査・編集・テストを一続きで進める。

図02を見てください。人間がすべての作業を引き取る形から、AIへ作業工程も任せる形へ変化しています。



図02-2 AI開発エージェントの基本循環ー調べる、作業する、確かめるを証拠とともに繰り返す。

- 新しい変更は動くが、以前の機能への影響は調べていない。

大切なのは、完了報告を疑うのではなく、「何を根拠に完了としたか」を確かめることです。

よい報告には、少なくとも三つの情報があります。

- 1 何を変更したか。
- 2 どの方法で確認したか。
- 3 何が未確認か。

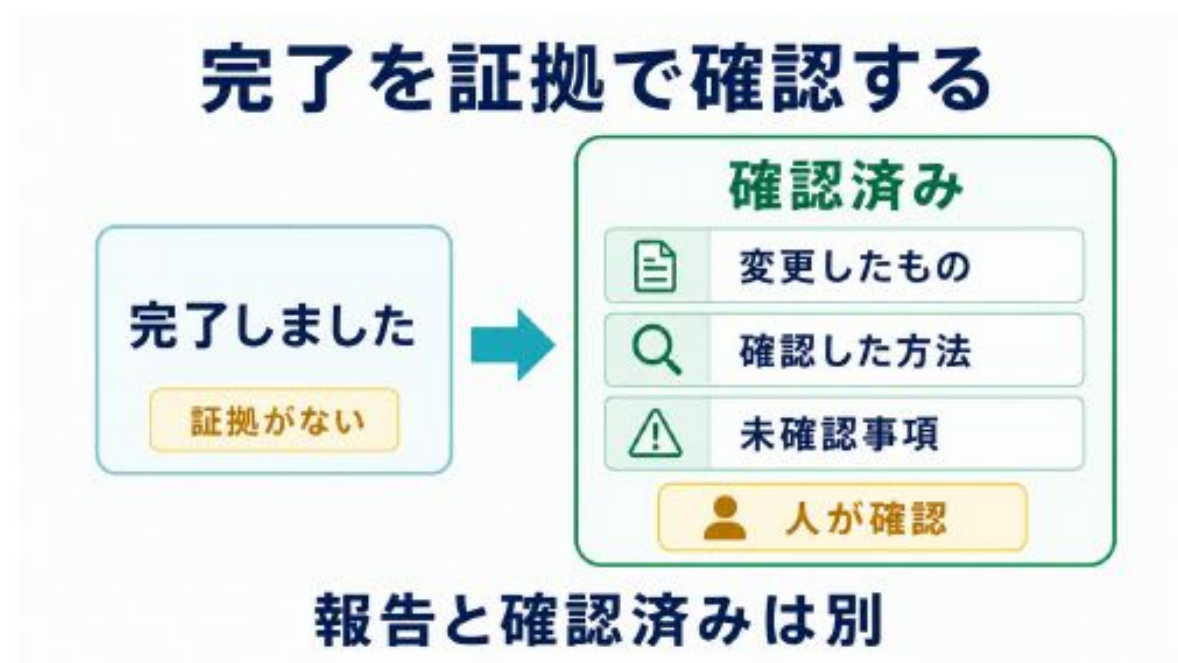


図02-4 完了を証拠で確認する——変更したもの、確認した方法、未確認事項を分け、人間が完成条件と照合する。

図02-4の左側には、根拠のない「完了しました」だけがあります。右側には、変更したファイル、テストやブラウザでの確認、まだ残っている未確認事項があります。右側の情報がそろって初めて、依頼者は自分の完成条件と照合できます。

権限は、多ければよいわけではありません。仕事に必要な範囲だけを渡す方が、誤操作の影響を小さくできます。

作業範囲と承認の境界

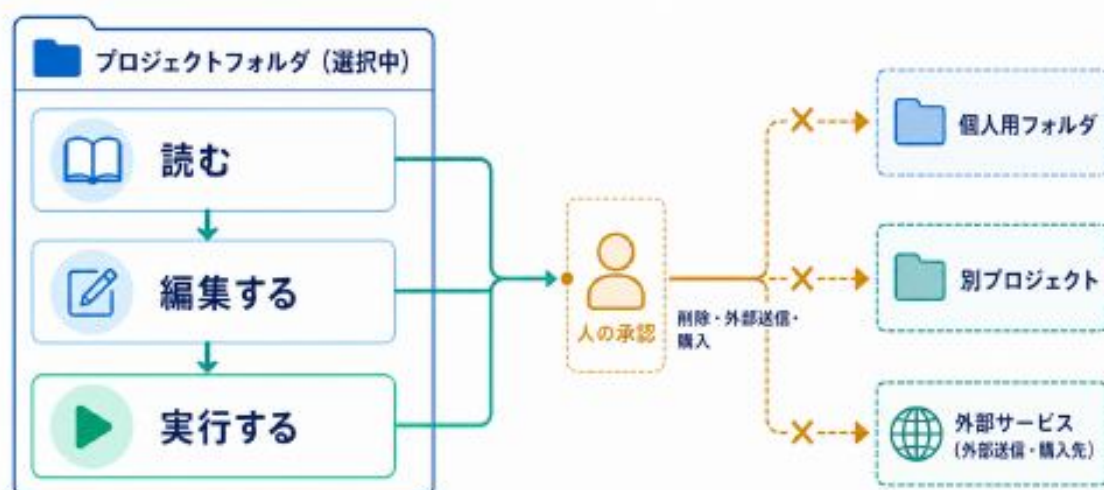


図02-3 作業範囲と承認の境界——読む、編集する、実行するは別の権限であり、範囲外や外部操作には人の判断を置く。

図02-3では、許可されたプロジェクトだけが明るい作業範囲になっています。AI開発エージェントを使うことは、パソコン全体を一括して渡すことではありません。対象フォルダを限定し、必要な操作だけを許可し、影響の大きい操作の前で止める設計ができます。

「読めるなら変更もできる」「編集できるなら外部へ送れる」と考えないことが大切です。見た目が一つの画面でも、読み取り、書き込み、コマンド実行、ネット接続、外部送信は別の操作です。仕事を頼む前に、今回必要なのはどこまでかを決めます。

Codexで成果物を作る

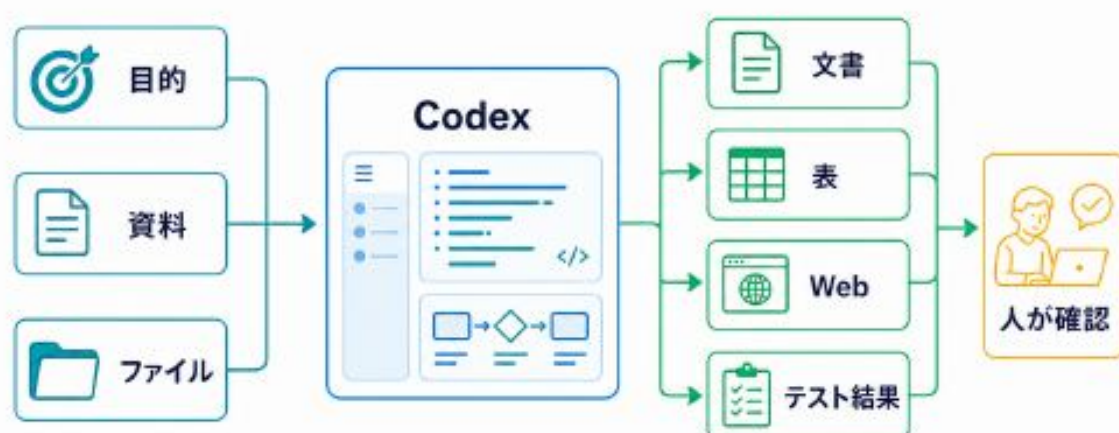


図03 Codexで成果物を作る——目的と材料を受け取り、確認できる成果物へつなげる。

図03を見てください。Codexは特定の一形式だけでなく、複数の成果物と作業工程を扱えます。

Codexの四つの入口



図03-2 Codexの四つの入口——対象、実行場所、結果の保存先を入口ごとに確認する。

モデルとCodexの作業環境は別



図03-3 モデルとCodexの作業環境——考えるモデルと、道具・権限・実行を含む環境を分ける。

確認すべきなのは、次の三点です。

- どの作業環境を使っているか。
- その環境で、どの道具を使えるか。
- どの範囲まで操作を許可しているか。

主な入口は一つではない

Codexの入口を、非プログラマー向けに整理してみましょう。



図03-4 プロジェクトから確認まで——指示、変更、テストを追い、外部操作の前で人が判断する。

ファイル以外の道具を使う場合

Codexの仕事は、ローカルファイルだけに閉じるとは限りません。利用環境によっては、Web検索、ブラウザ操作、画像生成、外部サービスとの接続などを使える場合があります。

これにより、「公式情報を調べ、内容を整理し、資料を作る」「Webサイトを開き、表示を確認する」「デザイン案に使う画像を作る」といった、複数種類の仕事を一つの流れにできます。

一方、道具が増えるほど、確認事項も増えます。

- Web情報は公式か、更新日はいつか。
- 外部サービスへ秘密情報を送っていないか。
- 画像や文章の利用条件に問題はないか。
- 実在する顧客や第三者へ送信しようとしていないか。



図04 Claude Codeで成果物を作る——プロジェクトを読み、変更、コマンド、差分、テストを一つの流れで扱う。

図04を見てください。Claude Codeはコード、ターミナル、Git、テストなどの開発作業を一つの流れとして扱います。



図04-2 入口と実行環境を分ける——同じ画面に見えても、コードが動く場所を確認する。

ただし、対象プラン、利用上限、提供条件は変わる可能性があります。本書では固定料金を暗記するのではなく、利用開始前に公式の料金・契約画面で、「Claudeのチャット」と「Claude Code」のどちらが含まれるかを確認することを勧めます。

Claude Codeの基本的な働き方

Claude Codeの公式説明では、仕事の流れが「状況を集める」「行動する」「結果を検証する」という循環で示されています。



図04-3 Claude Codeの作業循環——状況、行動、検証を繰り返し、人は途中で止めたり方向を変えたりできる。

たとえば、「会員登録後のメールが送られない」という問題を頼んだ場合、利用できる範囲で次のように進めます。

- 1 エラーメッセージや期待する動作を確認する。
- 2 関係するファイルや設定を検索する。
- 3 複数のファイルを読み、処理のつながりを理解する。



図04-4 プロジェクト指示と拡張—CLAUDE.md、Skills、MCP、Hooksは役割を分け、人が内容を確認して更新する。

拡張機能は、目的から選ぶ

Claude Codeには、繰り返し使う作業をまとめるスキル、外部のデータや道具を接続するMCP、特定の出来事の前後にコマンドを動かすフックなどの拡張方法があります。また、複数のエージェントやセッションを使って仕事を分ける仕組みもあります。

これらは便利ですが、名前を覚えることが目的ではありません。

たとえば、毎回同じ確認手順を使うならスキル化を考える。社内の課題管理システムから情報を読むならMCPなどの接続方法を検討する。ファイル編集後に必ず整形コマンドを動かしたいならフックを検討する、という順序です。

最初からすべてを導入すると、どの仕組みが何をしたのか分かりにくくなります。小さな手順で結果を確認し、再利用する価値があるものだけを仕組みにします。



図05 仕事の終点で二択——自分で見る成果物ならCodex、担当者が確認するコード差分ならClaude Codeを検討する。

図05を見てください。最初に製品名を決めるのではなく、仕事の終点を決めます。欲しいものがPowerPoint、Excel、PDF、画像、Web案などの成果物なのか。それとも、既存システムへ反映できるコード差分とテスト結果なのか。この違いが最初の分かれ目です。

この場面はCodexが向く

場面1 複数の資料を、一つの成果物へまとめたい

たとえば、会議メモ、売上表、社内資料、公式サイトの情報を集め、説明資料へまとめる仕事です。

この仕事では、プログラムのコードだけでなく、文章、表、ブラウザ、スライドなどを行き来します。完成条件も「コードが動くこと」ではなく、「内容が正しく、読みやすい資料として確認できること」です。

たとえば、「このフォルダを調べ、問題点と改善案を文書にしてください」「このExcelを変更せず、集計方法だけ提案してください」という依頼です。調査結果を成果物として受け取り、そのあと実装するかを判断できます。



図05-2 Codexが向く非エンジニアの仕事——資料、表、公式情報、Web案を横断し、成果物を画面で確認する。

Codexが向く目印は、次の三つです。

- 完成品が、資料、表、画像、PDF、Web案など複数の形式にまたがる。
- ブラウザや外部の道具も使いながら進めたい。
- 非エンジニア自身が、完成した成果物を画面で確認したい。

Claude Codeには、CLAUDE.md、権限ルール、Skills、MCP、Hooksなど、開発手順をプロジェクトへ組み込む仕組みがあります。たとえば、編集後に決まった検査を行う、禁止されたファイルを変更させない、既存のレビュー手順へ結果を渡す、といった設計です。

この場面では、非エンジニアがすべてのコマンドを理解するより、開発担当者が用意した手順の中で依頼し、差分と確認結果を担当者へ渡すことが重要です。

場面3 成果物より「変更差分」が納品物になる

仕事の終点がPowerPointやPDFではなく、既存リポジトリへ取り込める変更差分である場合です。

このとき確認するのは、見た目だけではありません。どのファイルが変わったか、既存の設計と合っているか、テストで何を確認したか、本番へ影響する操作が残っていないかを見ます。

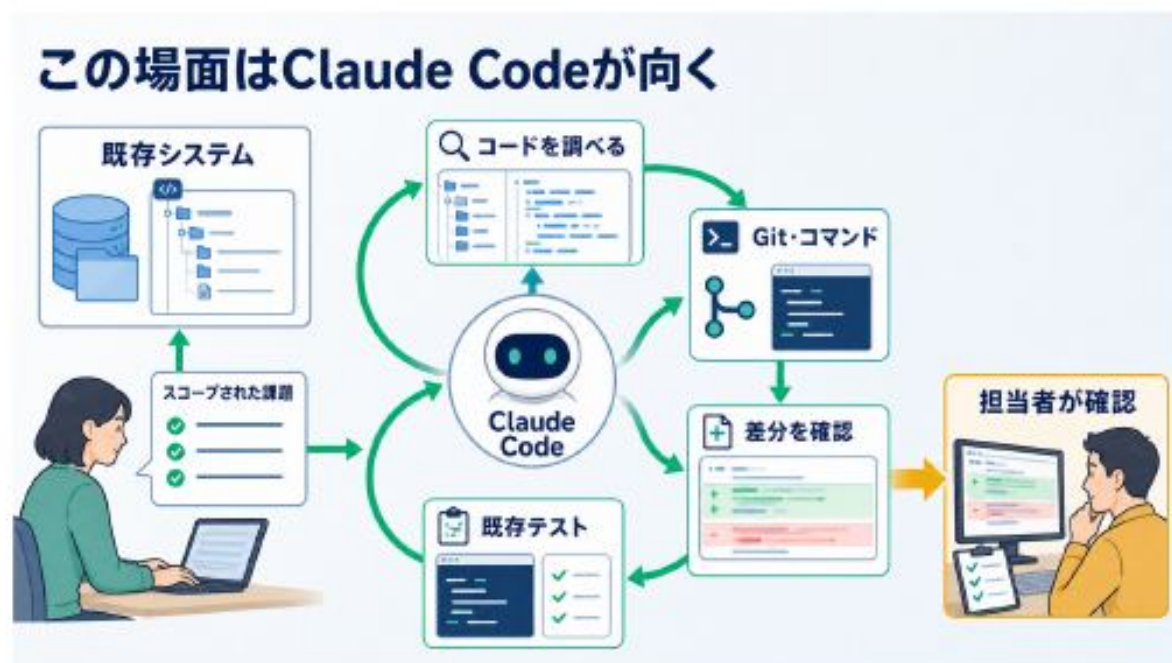


図05-3 Claude Codeが向く非エンジニアの依頼——既存システムを調べ、Git、コマンド、差分、テストを通して担当者へ渡す。



図05-4 どちらを選んでも人が止める——公開、送信、削除、購入、本番変更は、製品に関係なく人の承認を置く。

「成果物ができた」と「外部へ反映した」は別の完了状態です。非エンジニアは、AIの作業報告に、作成済み、確認済み、公開済みのどこまで含まれるかを確認めます。

非エンジニア向け五つの判断質問

迷ったときは、次の順番で答えてください。

質問1 最終成果物は何か

資料、表、画像、PDF、Web案なら、まずCodexを検討します。既存システムへ入れるコード差分なら、Claude Codeを検討します。

質問2 既存のリポジトリが中心か

Gitで管理された既存システムがあり、その中を調べて直す仕事なら、Claude Codeの開発手順が合いやすくなります。

第5章 製品名を決める前に、仕事を七つに分ける

「どちらを使うか」の前に、一枚書く

第4章では、非エンジニア向けの入口を二つに分けました。

- 文書、表、画像、Webページなど、自分が成果物を見て判断する仕事はCodexから考える。
- 既存システム、Git、コマンド、テストなど、開発者が変更差分を確認する仕事はClaude Codeから考える。

ただし、「Excelを整理したい」「ホームページを直したい」だけでは、まだ仕事の形が分かりません。そこで製品を開く前に、依頼を七つの欄へ分けます。

依頼前の仕事整理シート	
1 成果物	2 新規か変更か
3 対象範囲	4 一度か繰り返しか
5 作業場所	6 危険な操作 ⚠
7 完成条件	

製品を選ぶ前に、仕事を分ける

図20 依頼前の仕事整理シート——七つの欄を埋めてから、作業の入口を選ぶ。

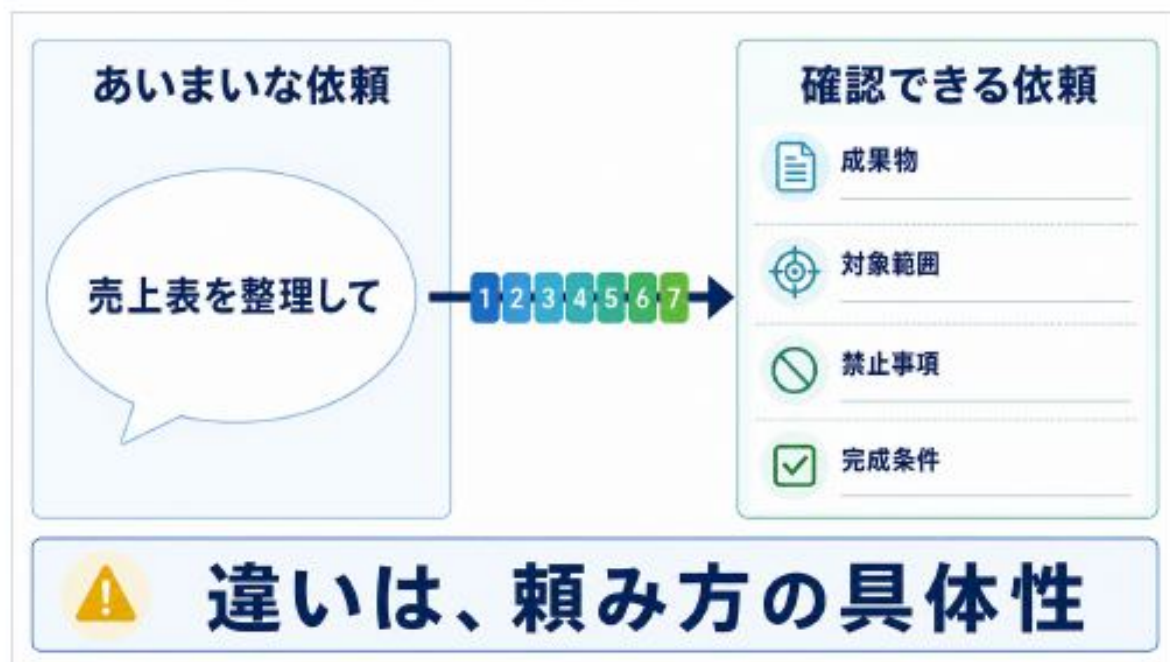


図21 あいまいな依頼から確認できる依頼へ——成果物、対象範囲、禁止事項、完成条件を言葉にする。

七つの欄を埋める目的は、長い指示を書くことではありません。少なくとも、次の四点を相手とそろえることです。

- 最後に何を受け取るのか。
- どこまで触ってよいのか。
- 何をしてはいけないのか。
- 何を見れば終わったと分かるのか。

ここが決まると、第4章の二択も具体的になります。たとえば、加工済みの表を自分が目視確認するならCodex側の入口が分かりやすい場面です。一方、既存サイトの複数ファイルを直し、テスト結果とGitの差分を開発者が確認するならClaude Code側の入口が分かりやすい場面です。

製品名だけを比べるより、仕事の形から選ぶほうが迷いません。

- 元ファイルの上書きや削除。
- 個人情報や秘密情報の外部送信。
- 顧客や社員へのメッセージ送信。
- WebサイトやSNSへの公開。
- 購入や有料生成。
- 本番サーバーや本番データの変更。

危険度が高いほど、対象を小さくし、途中で人が承認する場所を増やします。「たぶん行わないだろう」と期待するのではなく、「行わない」と明記します。



図22 元データを残す記入例——元ファイルは変更せず、コピーへ作業する条件を先に決める。

たとえば表計算ファイルの整理なら、次の一枚で十分です。



作っただけでは、公開完了ではない

図23 作成・確認・公開は別の状態——公開や送信へ進む前に、人間の承認を置く。

AIがファイルを作っただけなら、作成済みです。人が内容を見たら確認済みです。外部URLで新しい内容を確認できたとき、初めて公開済みと言えます。

公開、送信、削除、購入、本番変更は、CodexでもClaude Codeでも人の承認を置きます。完了報告には、成果物の場所、確認した内容、未確認の内容を分けて書かせます。

そのまま使える依頼準備票

次の七行をコピーし、空欄を埋めてください。

成果物：新規か変更か：対象範囲：一度か繰り返しか：作業場所
：危険な操作・禁止事項：完成条件：

七行を書いたあと、第4章の二択へ戻ります。

第6章 どこで動かすか——ローカル・クラウド・リモート

ブラウザだからクラウド、ではない

ブラウザやスマートフォンから見ているからクラウド。デスクトップアプリだから自分のパソコン。画面だけでこのように判断すると、仕事が実際に動く場所を取り違えます。

最初に確認するのは、次の三点です。



図24 画面だけでは判断しない——元ファイル、実行場所、結果の保存先を分けて確認する。

- 1 元ファイルはどこにあるか。
- 2 ファイル操作やコマンドはどこで実行されるか。
- 3 変更結果はどこへ保存されるか。



図25 ローカルとクラウドの違い——自分のPCで動くか、分離された別環境で動くか。

クラウドが候補になるのは、次のような場面です。

- GitHubなどに保存されたプロジェクトを扱う。
- 自分のパソコンを閉じてでも作業を続けたい。
- 独立した複数の仕事を分けて進めたい。
- ローカル環境を直接変更せずに調査や試作をしたい。

重要なのは、クラウドが自分のパソコンの完全な複製ではないことです。

パソコン内だけにある未保存の変更、特別なソフト、社内ネットワーク、ローカル設定は、クラウド側にはない場合があります。GitHubから取得する方式なら、保存や反映をしていないファイルは渡りません。

クラウドへ仕事を渡す前に、次を確認します。

- 1 どのリポジトリやファイルを使うか。



図26 リモートでは接続先を確認する——同じスマートフォン画面でも、処理先は自分のPCまたはクラウドになり得る。

たとえば、スマートフォンから自分のパソコン上の作業へ指示する場合、画面はスマートフォンでも、ファイルとコマンドは自分のパソコン側にあります。

一方、スマートフォンからクラウドの作業を確認する場合、画面は同じでも実行場所はクラウドです。

公式資料で確認できる具体例は次のとおりです。

- Codex Remoteは、接続したパソコン上でタスクを実行し、スマートフォンから開始、指示、承認、結果確認を行います。
- Claude Code Remote Controlは、ブラウザやモバイル画面から、動作中のローカルセッションへ接続します。
- Claude Code on the webは、Anthropicのクラウド環境で実行されます。



図27 作業場所を選ぶ三つの質問——製品名ではなく、ファイル、継続条件、操作画面から選ぶ。

自分のパソコンにしかないものを使う

ローカルを候補にします。

例は、手元のExcel、編集中の画像、社内専用ソフト、まだ外部へ保存していないWebサイトです。CodexでもClaude Codeでも、ローカルで扱える入口を選びます。

自分のパソコンを閉じてでも続けたい

クラウドを候補にします。

ただし、必要なファイルや設定がクラウド側に用意されていることが前提です。「パソコンにしかない未保存ファイルを、クラウドが自動的に知っている」とは考えません。

外出先など、別の端末から確認したい

リモート操作を候補にします。

第7章 どこまで任せるか——権限と承認

「許可」を一つの言葉で考えない

AI開発エージェントは、質問へ答えるだけではありません。ファイルを読み、書き換え、コマンドを動かし、外部サービスへ接続できます。

だからといって、使える機能を最初からすべて許可する必要はありません。仕事に必要な操作だけを許し、それより影響の大きい操作では止めます。

権限は五段階で考える



必要な段階までだけ許可する

図28 権限は五段階で考える——読む、編集、実行、外部接続、外部へ影響する操作を分ける。

本書では、権限を次の五段階に分けます。

- 1 読む。

許可する前に、四つ確認

目的: 何をする?	☒
対象: どこを変える?	☒
通信: 外部へ送る?	☒
回復: 元に戻せる?	☒

⚠

? 説明を求める ✓ 内容を確認して許可

分からないまま押さない

図29 許可する前の四つの確認——目的、対象、外部通信、元へ戻す方法が分からなければ説明を求める。

- 1 目的——何をする操作か。
- 2 対象——どのファイルやデータを変えるか。
- 3 通信——外部へ接続し、何かを送るか。
- 4 回復——失敗した場合に元へ戻せるか。

分からない場合は、実行させる前に尋ねます。

この操作の目的、変更される対象、外部通信の有無、失敗した場合の影響と戻し方を、非エンジニア向けに説明してください。まだ実行しないでください。

説明を読んでも対象や影響が分からなければ、許可しません。

本番公開を三つの承認に分ける



図24-3 内容、公開先と対象版、実行許可を別々に確認し、一つの「OK」へまとめない。

公開前チェック

本番公開前の八項目



図24-4 対象、変更範囲、検査、確認用環境、設定、データ、戻し方、確認担当をそろえてから本番操作へ進む。