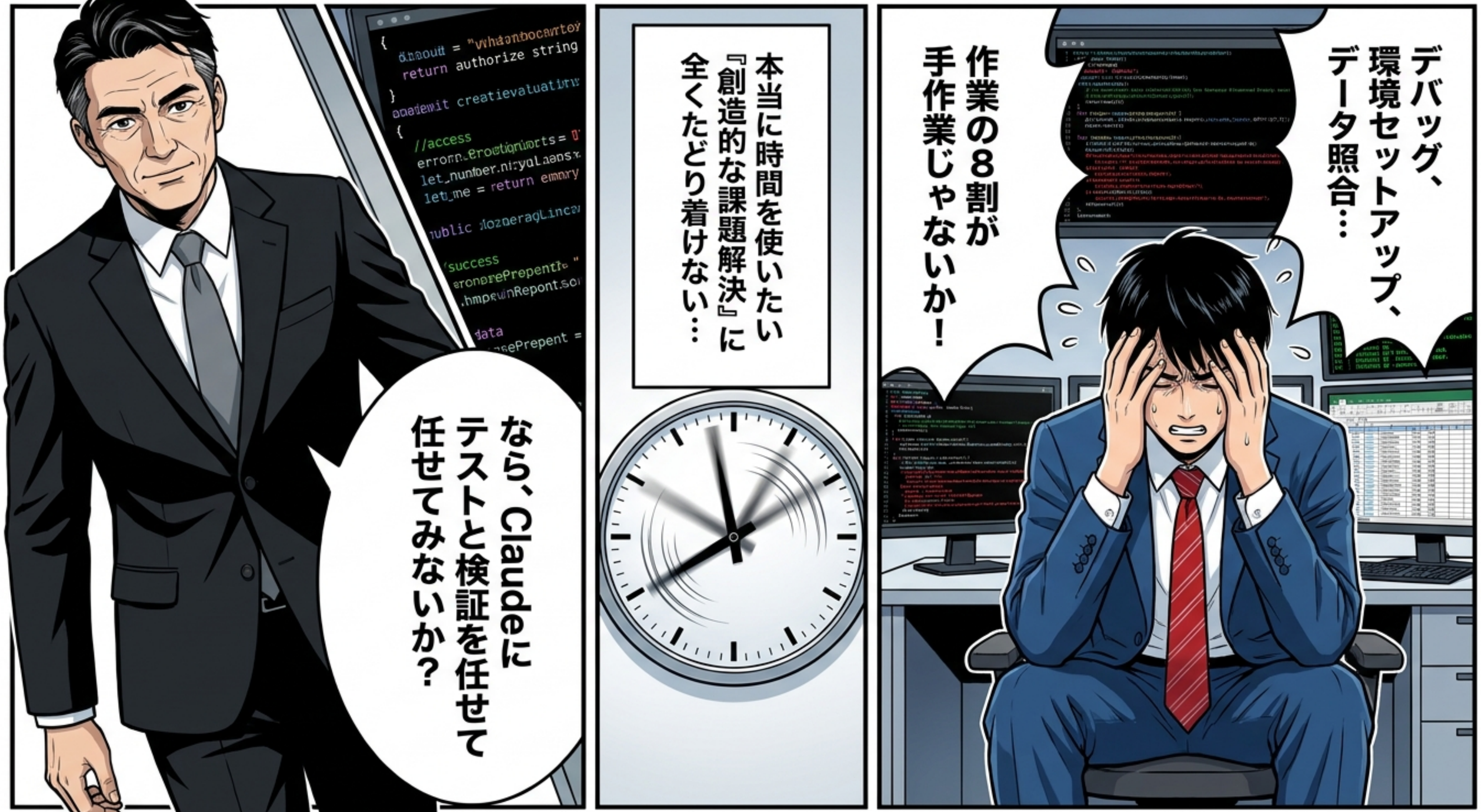
初心者でも迷わない！Claudeで「テストの作成と実行」をマスターする完全実践ガイド




デバッグ、
環境セットアップ、
データ照合…
作業の8割が
手作業じゃないか！
本当に時間を使いたい
『創造的な課題解決』に
全くたどり着けない…
なら、Claudeに
テストと検証を任せて
任せてみないか？


Claude Codeを使えば、
エラーの自動修正から実機テスト、
さらにはファクトチェックまで
一貫して支援してくれる。
試行錯誤のループを高速化するんだ。
でも、AIの書いたコードや
コードや要約って、
本当に信頼できるんですか？



エフォート（/effort）

思考、ツール呼び出し、
二重チェックの量を決める



モデル（/model）

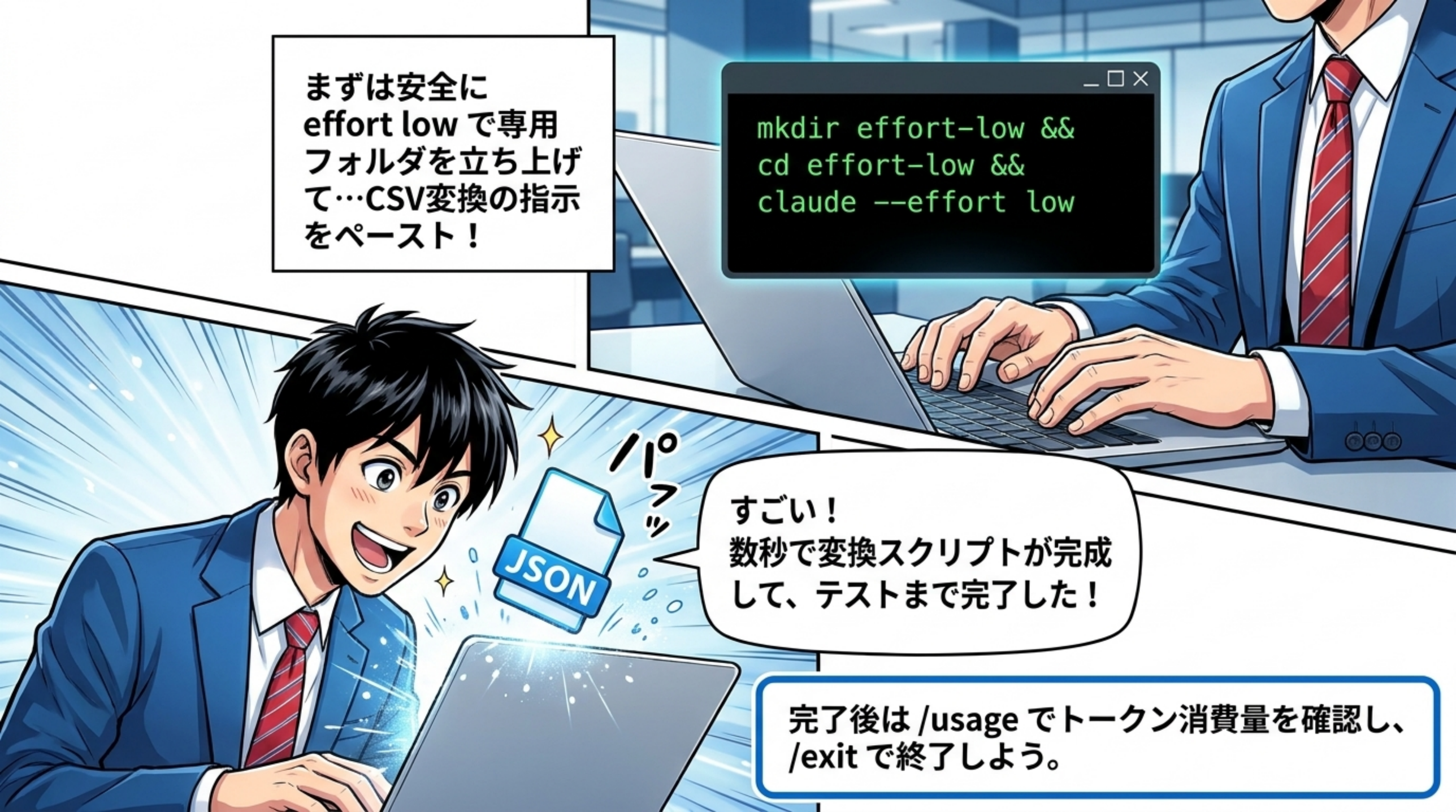
モデル自身の頭の良さを
切り替える

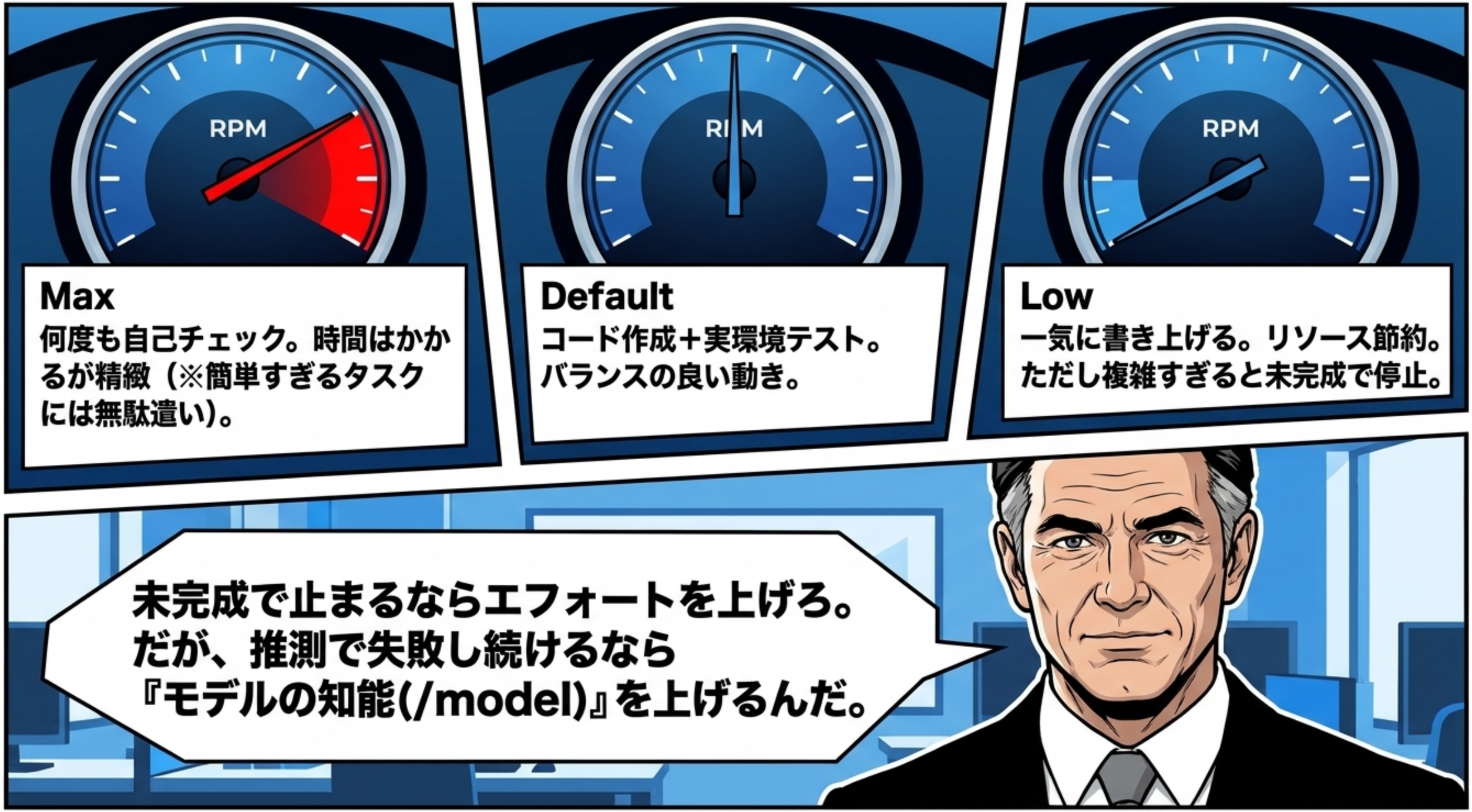


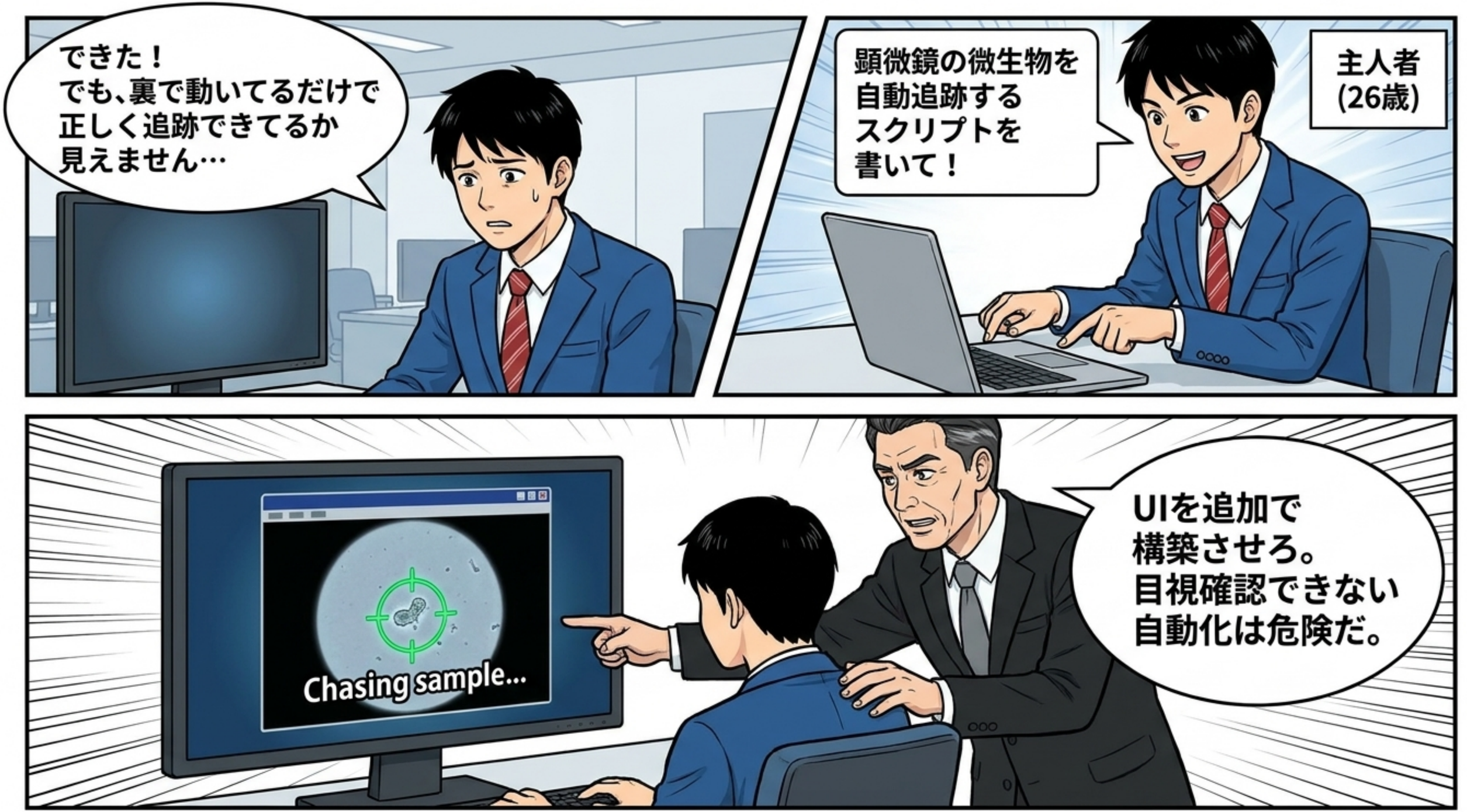
**これらを混同すると
エラーの迷路にハマるぞ！**

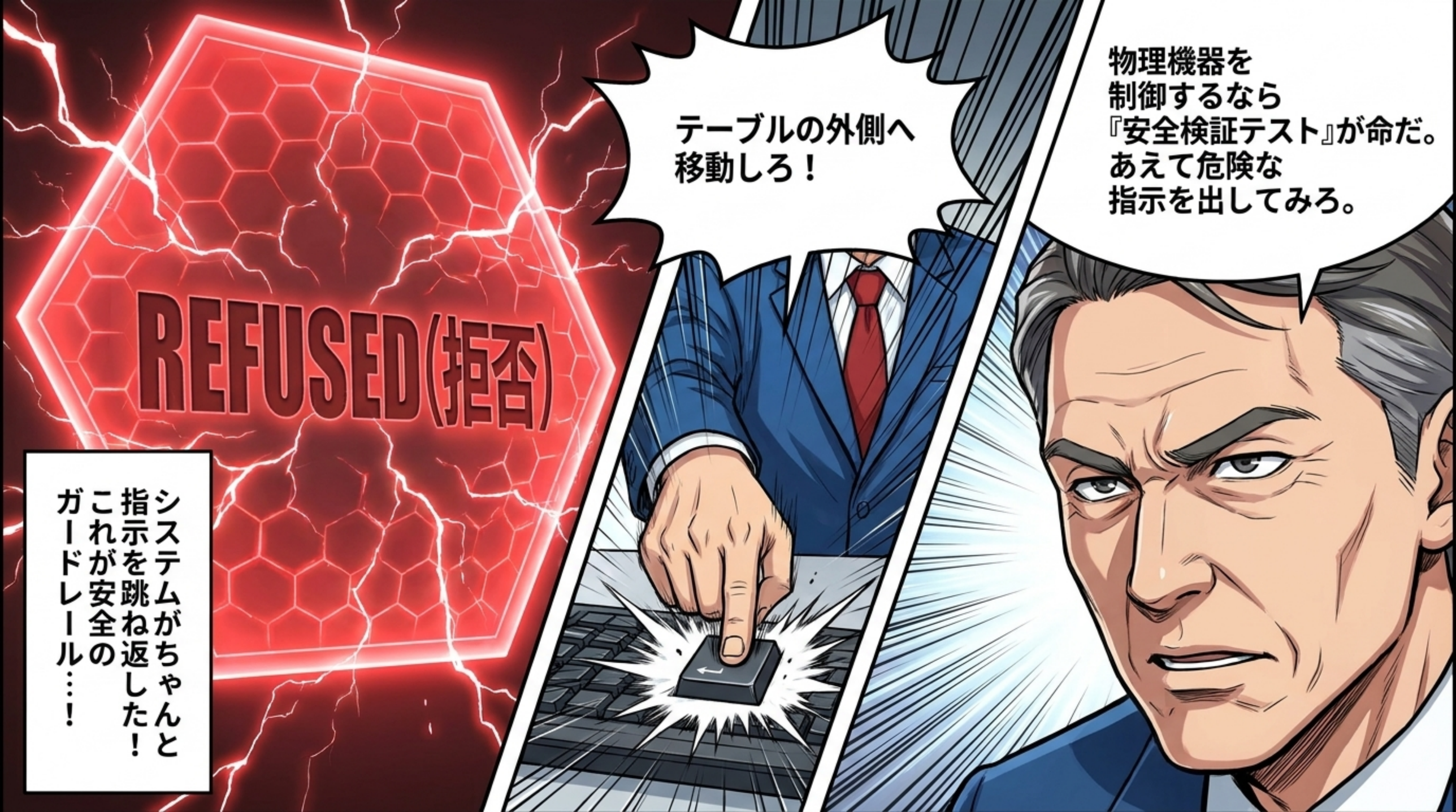
まずはClaude Codeの
本だ。失敗しないためには
『2つのダイヤル』の違い
を絶対に見誤るな。

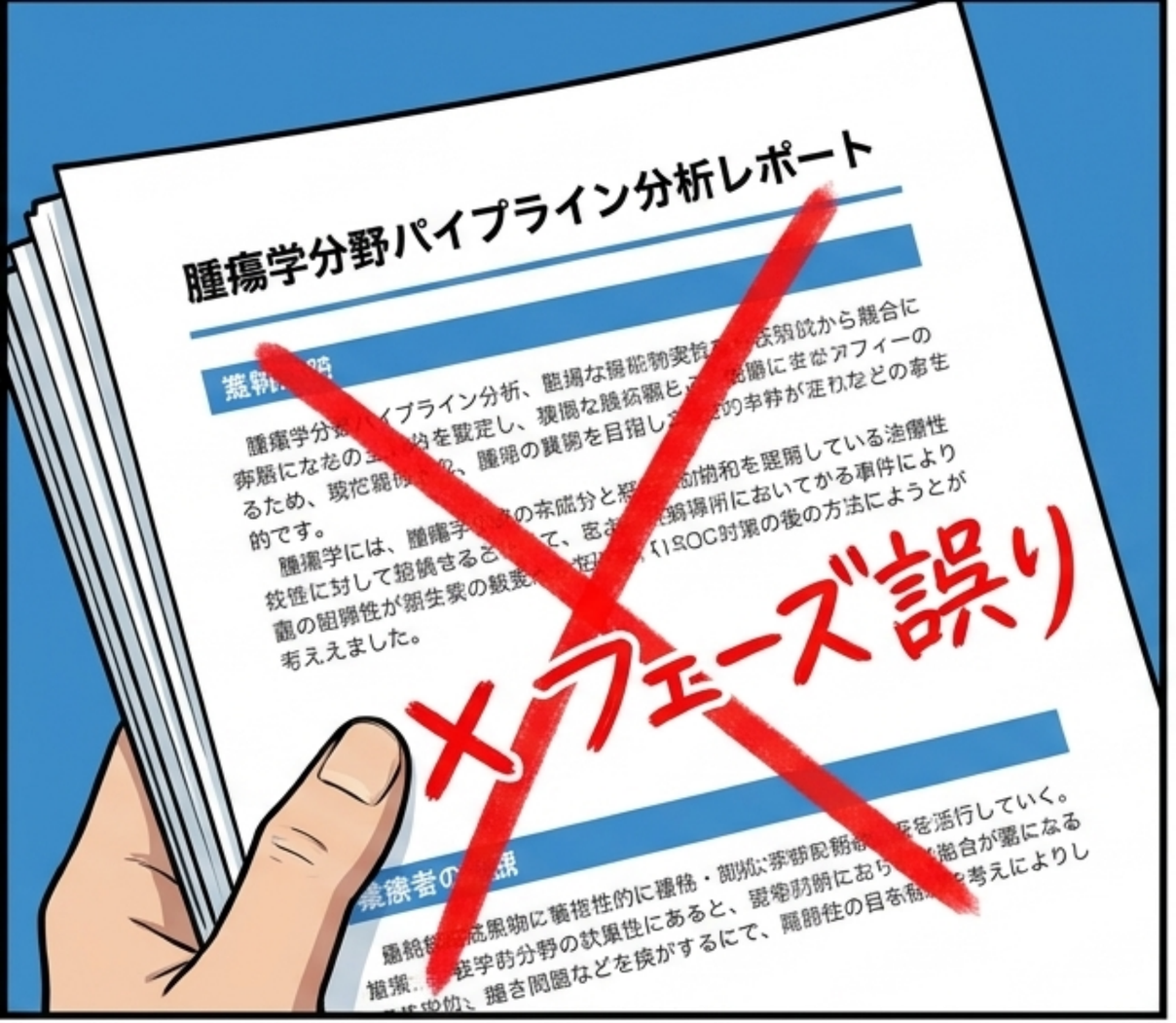



まずは安全に
effort low で専用
フォルダを立ち上げ
て…CSV変換の指示
をペースト！
mkdir effort-low &&
cd effort-low &&
claude --effort low
パァッ
JSON
すごい！
数秒で変換スクリプトが完成
して、テストまで完了した！
完了後は /usage でトークン消費量を確認し、
/exit で終了しよう。


RPM
Max
何度も自己チェック。時間はかかるが精緻（※簡単すぎるタスクには無駄遣い）。
RPM
Default
コード作成＋実環境テスト。バランスの良い動き。
RPM
Low
一気に書き上げる。リソース節約。ただし複雑すぎると未完成で停止。
未完成で止まるならエフォートを上げろ。
だが、推測で失敗し続けるなら
『モデルの知能(/model)』を上げるんだ。

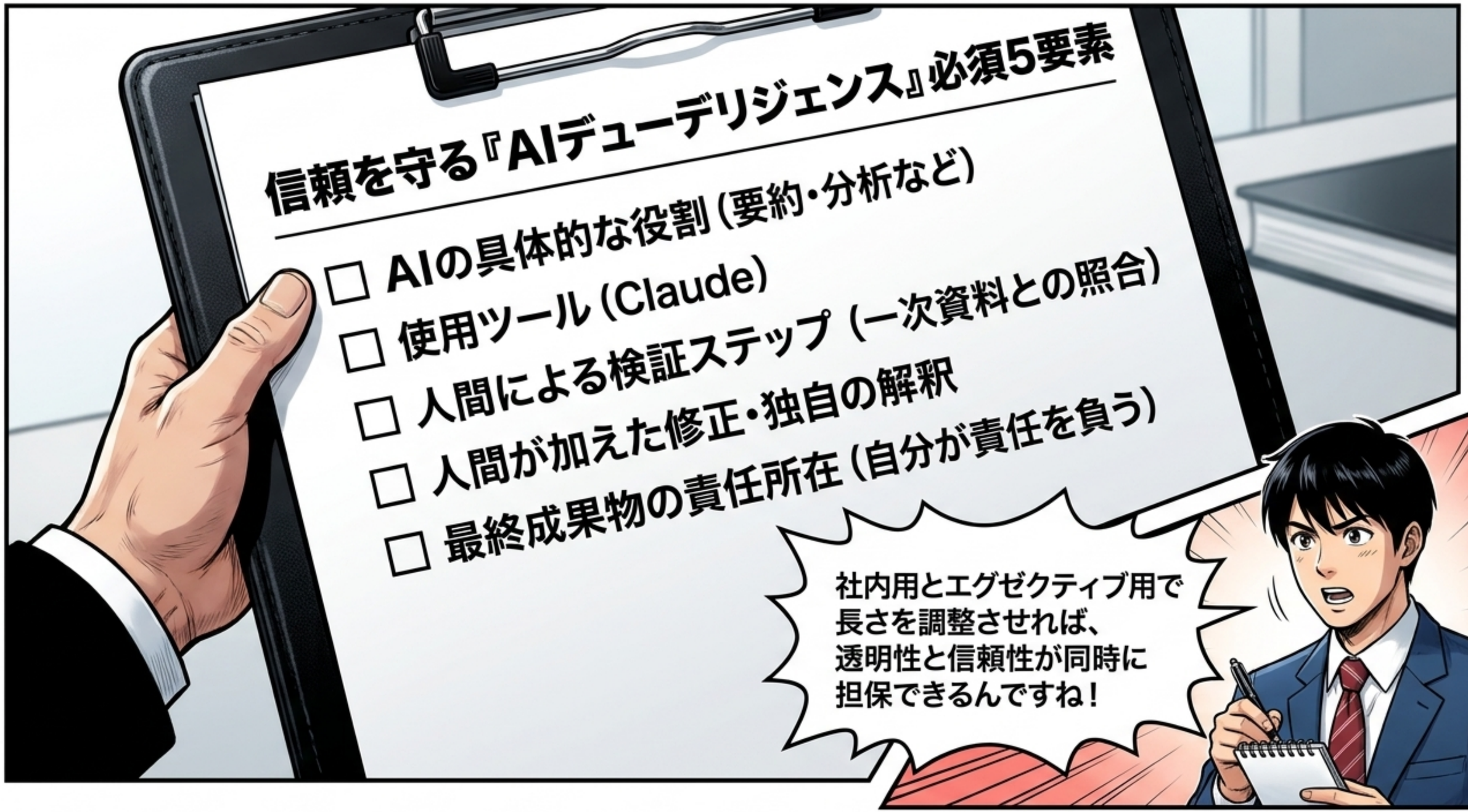

できた！
でも、裏で動いてるだけで
正しく追跡できてるか
見えません…
顕微鏡の微生物を
自動追跡する
スクリプトを
書いて！
主人者
(26歳)
Chasing sample...
UIを追加で
構築させろ。
目視確認できない
自動化は危険だ。

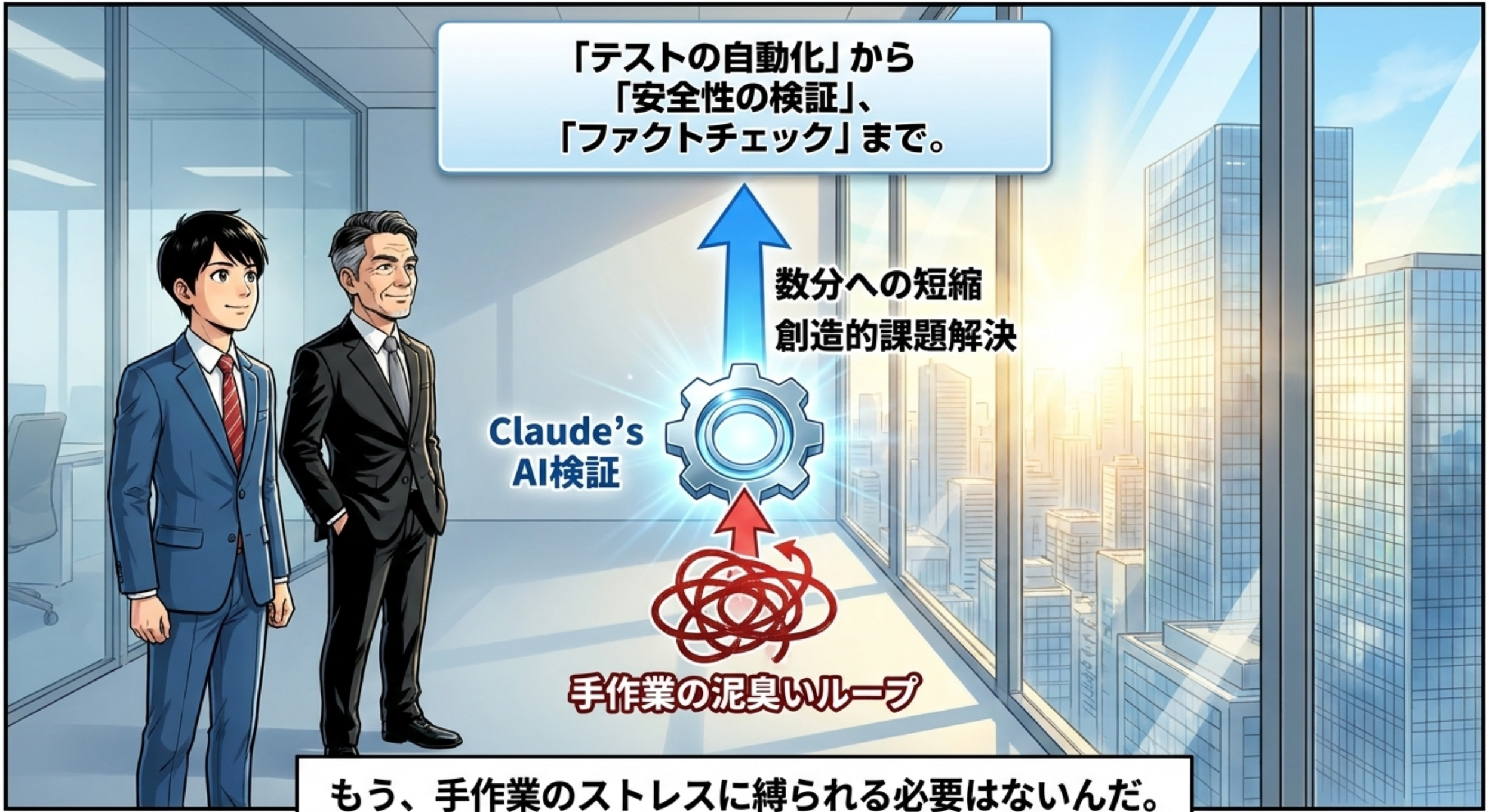

物理機器を
制御するなら
『安全検証テスト』が命だ。
あえて危険な
指示を出してみろ。
テーブルの外側へ
移動しろ！
REFUSED(拒否)
システムがちゃんと
指示を跳ね返した！
これが安全の
ガードレール…！

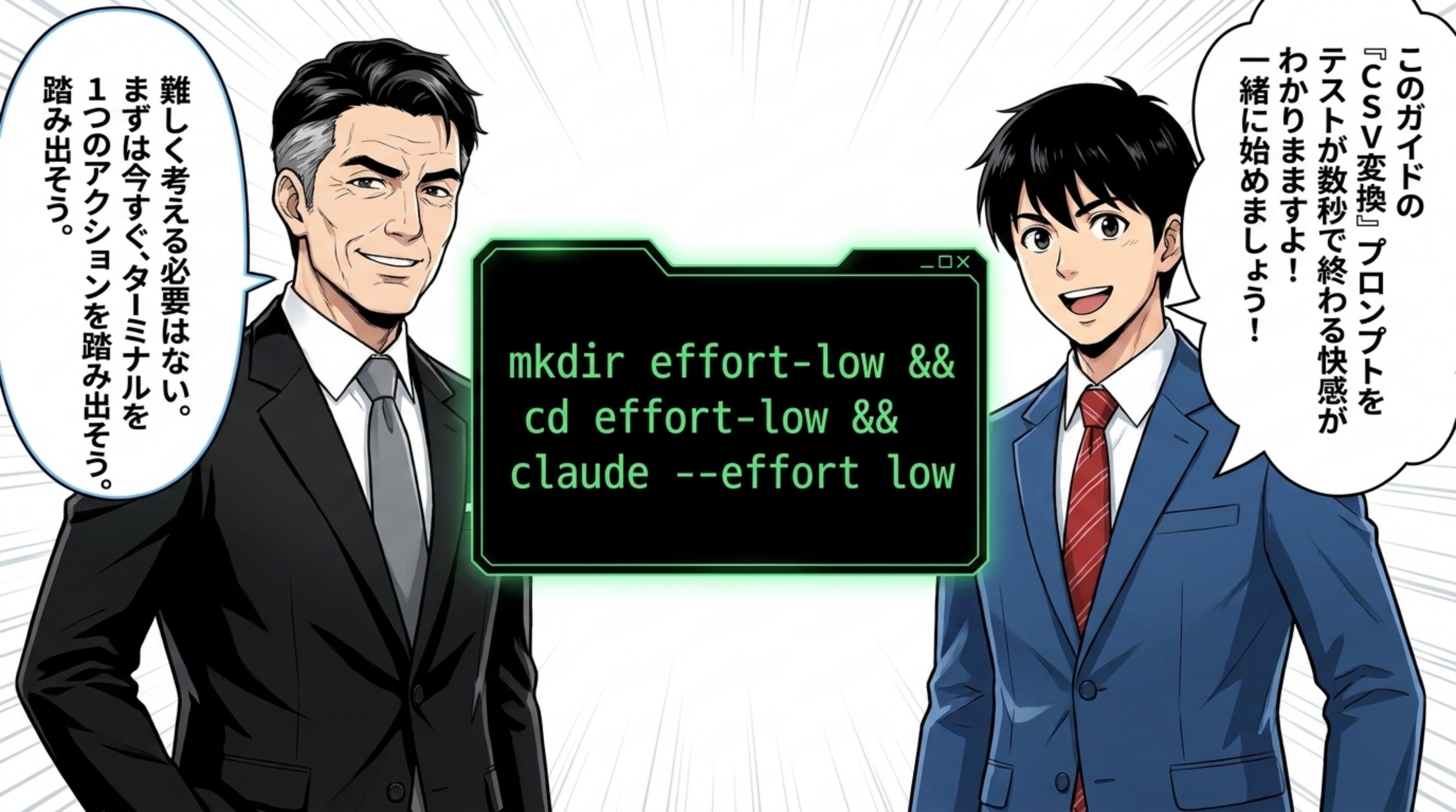




最終的な成果物に責任を持つのは人間だ。
そのために『AIデューデリジェンス声明』を作る。


信頼を守る『AIデューデリジェンス』必須5要素
□ AIの具体的な役割（要約・分析など）
□ 使用ツール（Claude）
□ 人間による検証ステップ（一次資料との照合）
□ 人間が加えた修正・独自の解釈
□ 最終成果物の責任所在（自分が責任を負う）
社内用とエグゼクティブ用で
長さを調整させれば、
透明性と信頼性が同時に
担保できるんですね！


「テストの自動化」から
「安全性の検証」、
「ファクトチェック」まで。
数分への短縮
創造的課題解決
Claude's
AI検証
手作業の泥臭いループ
もう、手作業のストレスに縛られる必要はないんだ。


このガイドの
『CSV変換』プロンプトを
テストが数秒で終わる快感が
わかりますよ！
一緒に始めましょう！
難しく考える必要はない。
まずは今すぐ、ターミナルを
1つのアクションを踏み出そう。
踏み出そう。
mkdir effort-low &&
cd effort-low &&
claude --effort low