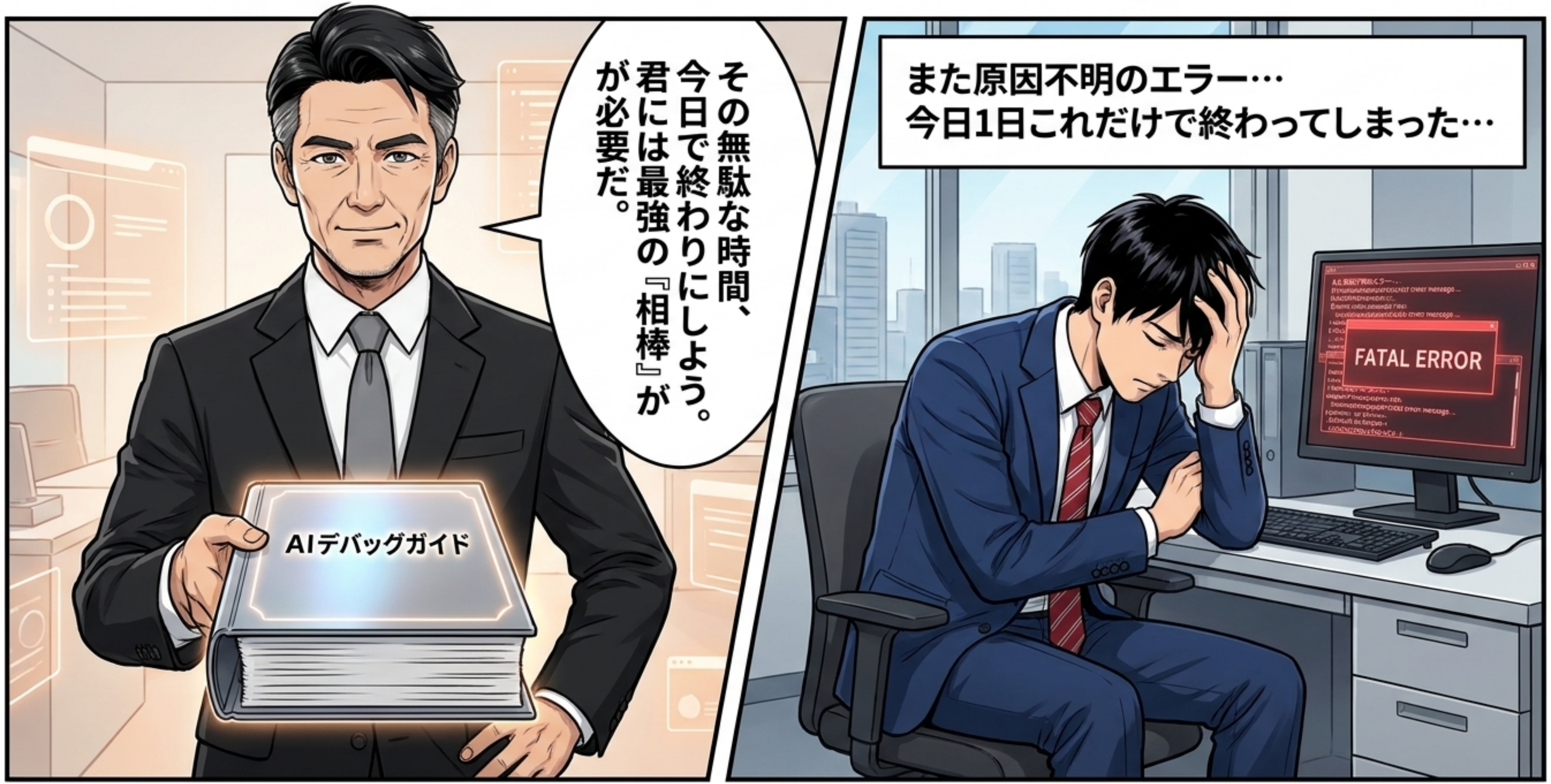
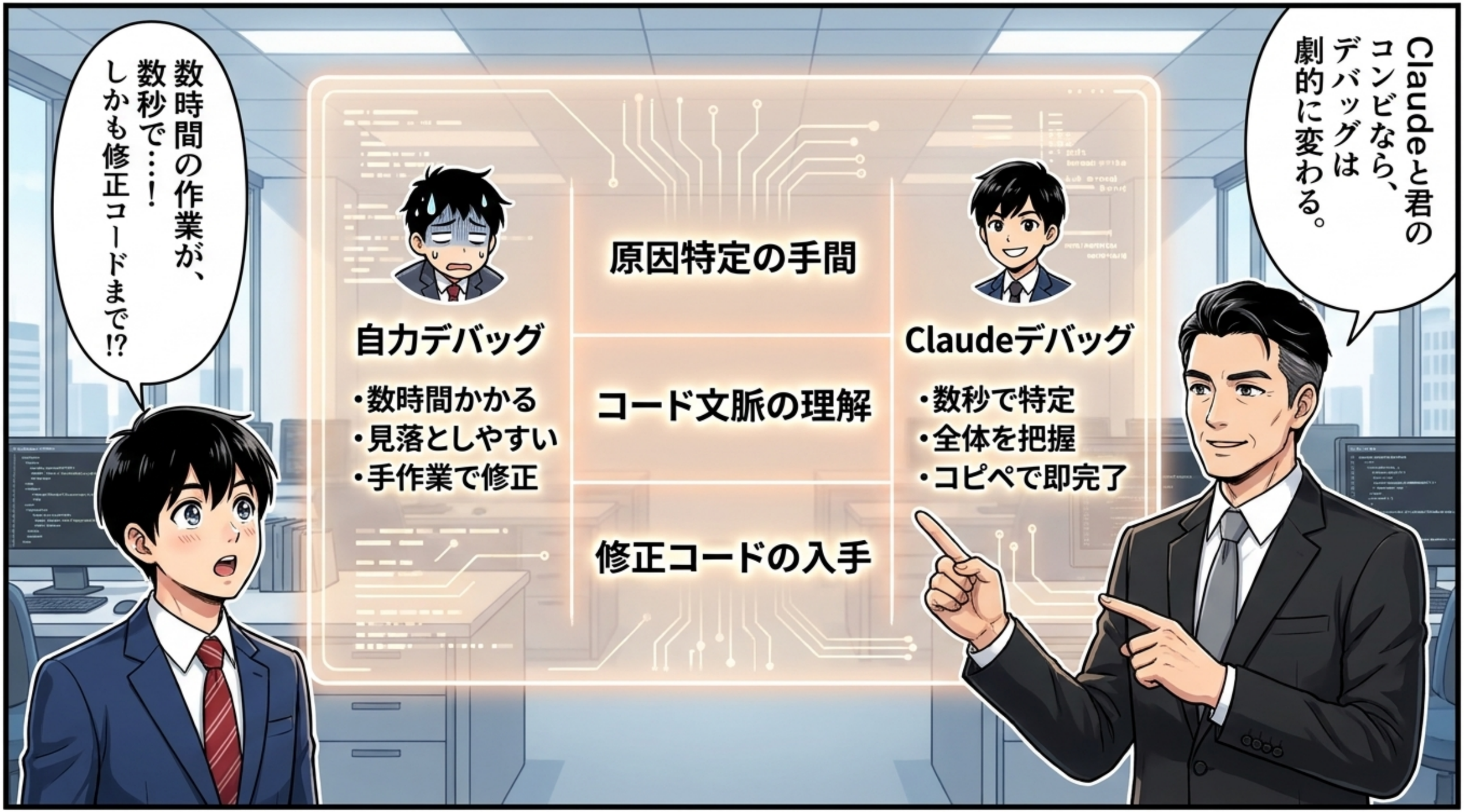
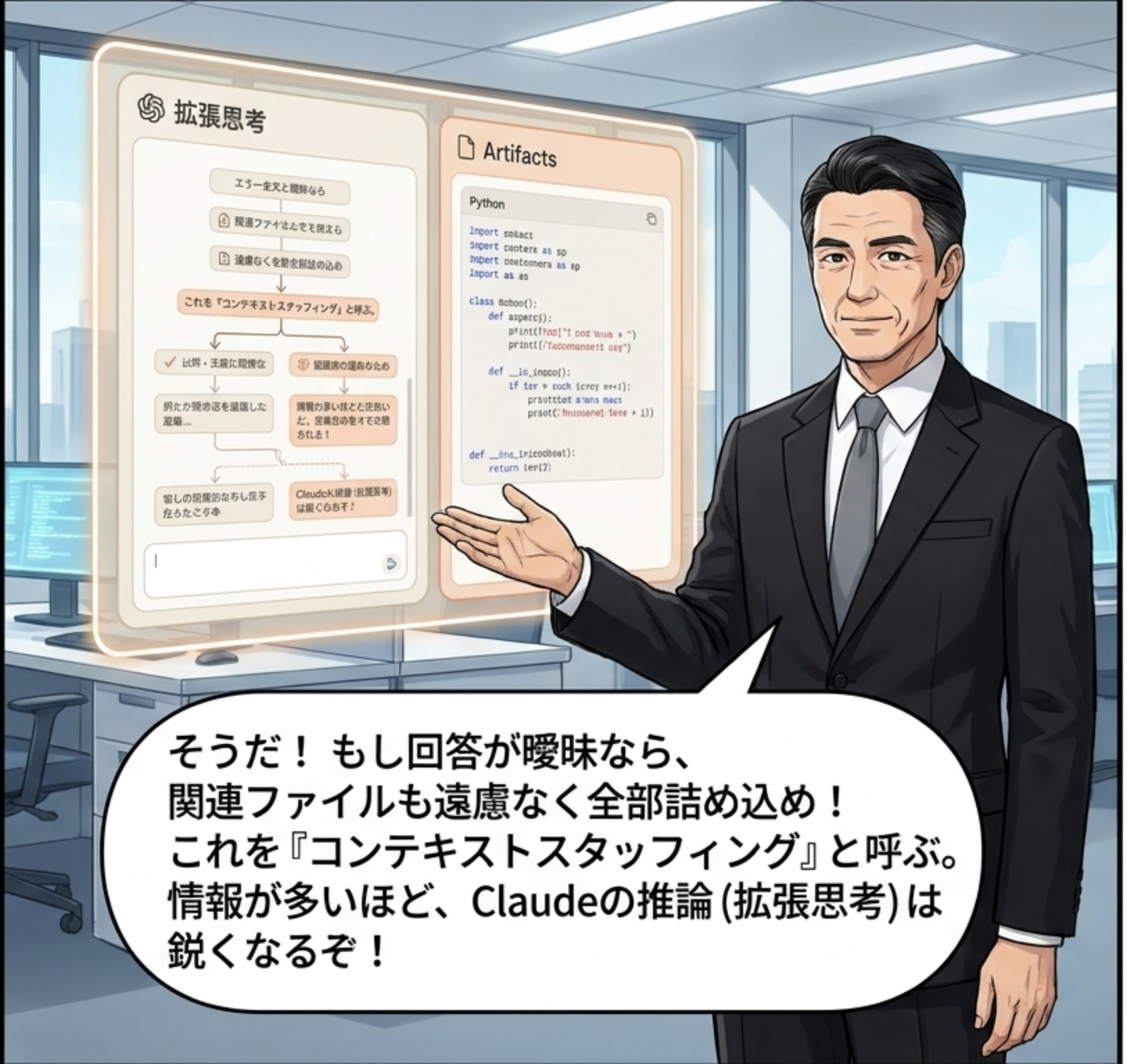
AIを最高のデバッグパートナーに！Claudeで始める完全ガイド

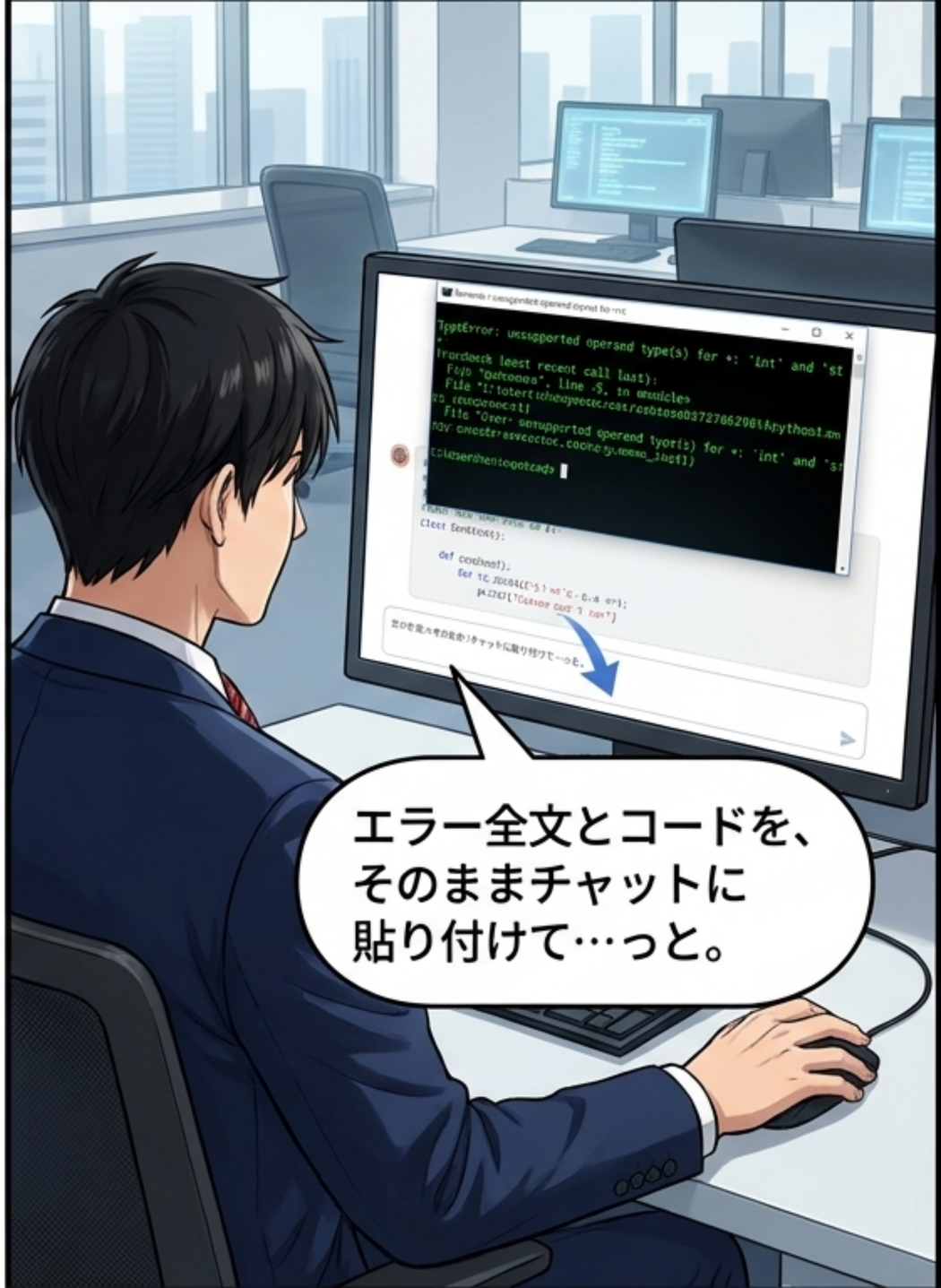



80%
開発・研究者の
時間の約80%が、
デバッグなどの非本質的な
トラブルシューティングに
奪われている
(※Anthropic研究事例)
任人前の記者
(26歳)
エラー文を検索して、
記事を読み比べて…
本当にやりたい開発に
全然進めない！
Claude：解決策提案
(Solution Proposal)
エラー画面への
恐怖は捨てろ！
Claudeがいれば、
数秒で原因を特定し、
君は本来の
『クリエイティブな開発』に
全力を注げるようになる！

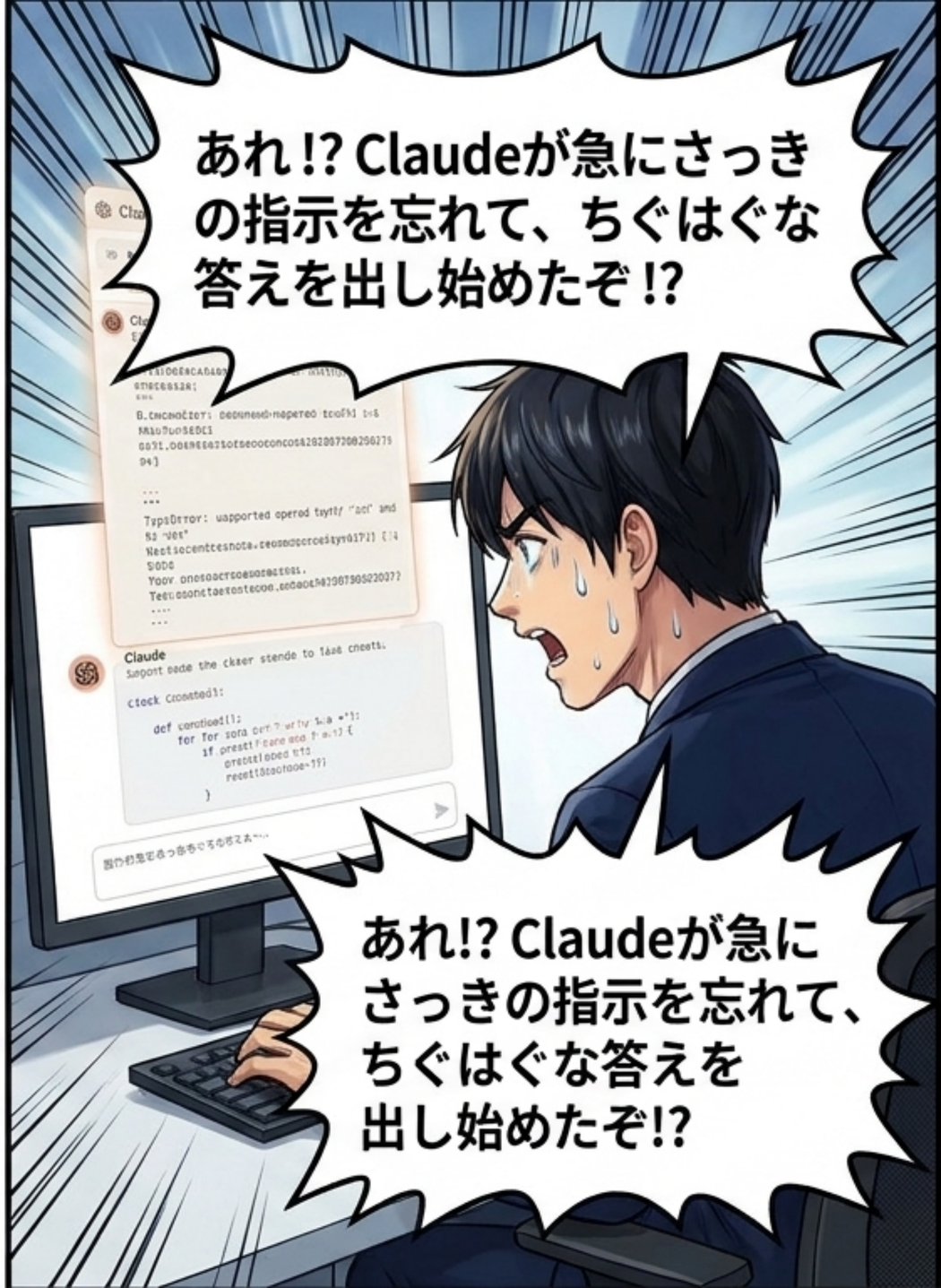

数時間の作業が、
数秒で…！
しかも修正コードまで!?
Claudeと君のコンビなら、
デバッグは劇的に変わる。
原因特定の手間
自力デバッグ
・数時間かかる
・見落としやすい
・手作業で修正
コード文脈の理解
Claudeデバッグ
・数秒で特定
・全体を把握
・コピペで即完了
修正コードの入手

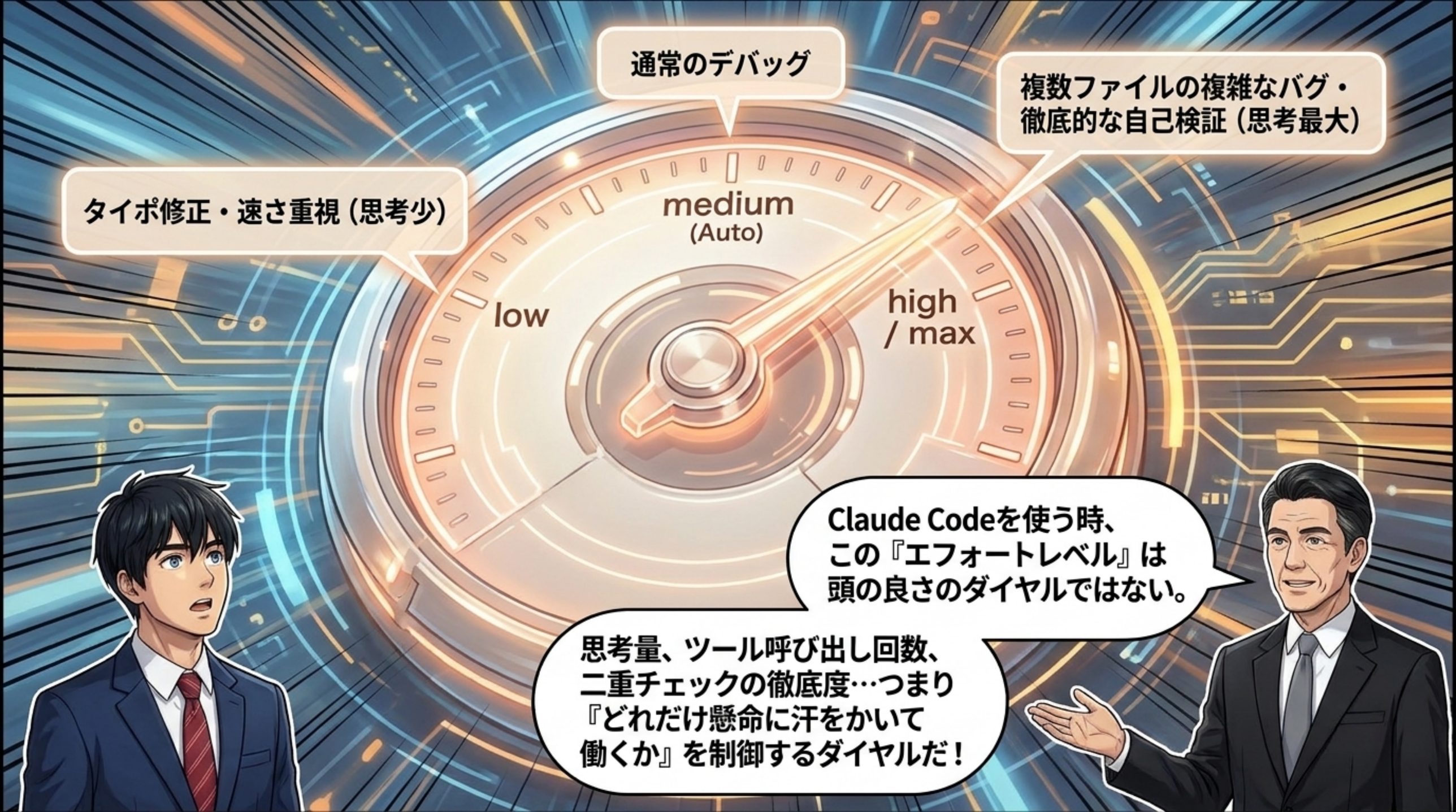

パラメトリックメモリ
(AIの脳内記憶)
コンテキスト
(目の前のオープンブック)
脳内の記憶(パラメトリックメモリ)だけで
推測させるとハルシネーション(嘘)が起きる!
目の前の教科書(コンテキスト)を
しっかり渡すことが重要だ。

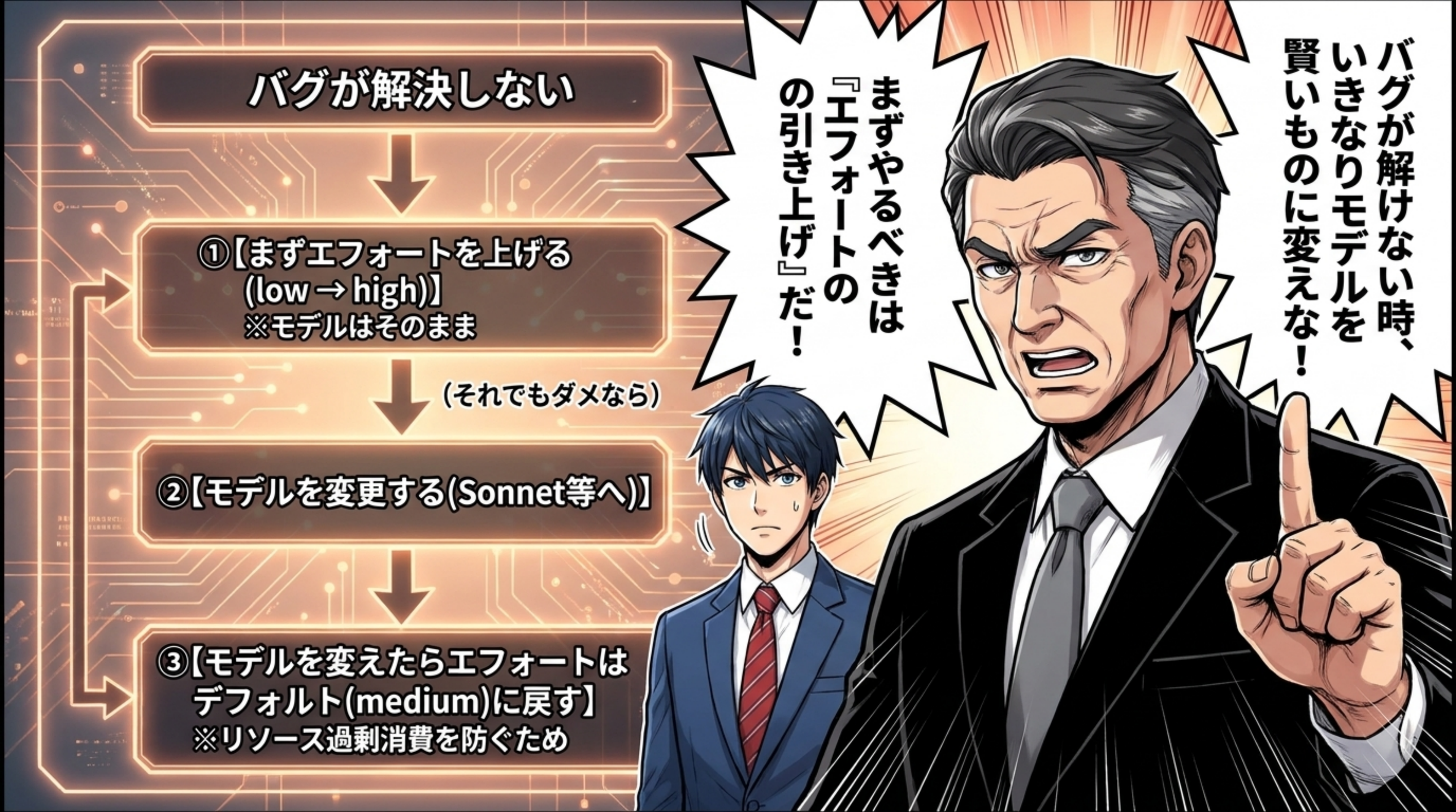

拡張思考
Artifacts
Python
そうだ！ もし回答が曖昧なら、
関連ファイルも遠慮なく全部詰め込め！
これを『コンテキストスタッフィング』と呼ぶ。
情報が多いほど、Claudeの推論 (拡張思考) は
鋭くなるぞ！


エラー全文とコードを、
そのままチャットに
貼り付けて…っと。

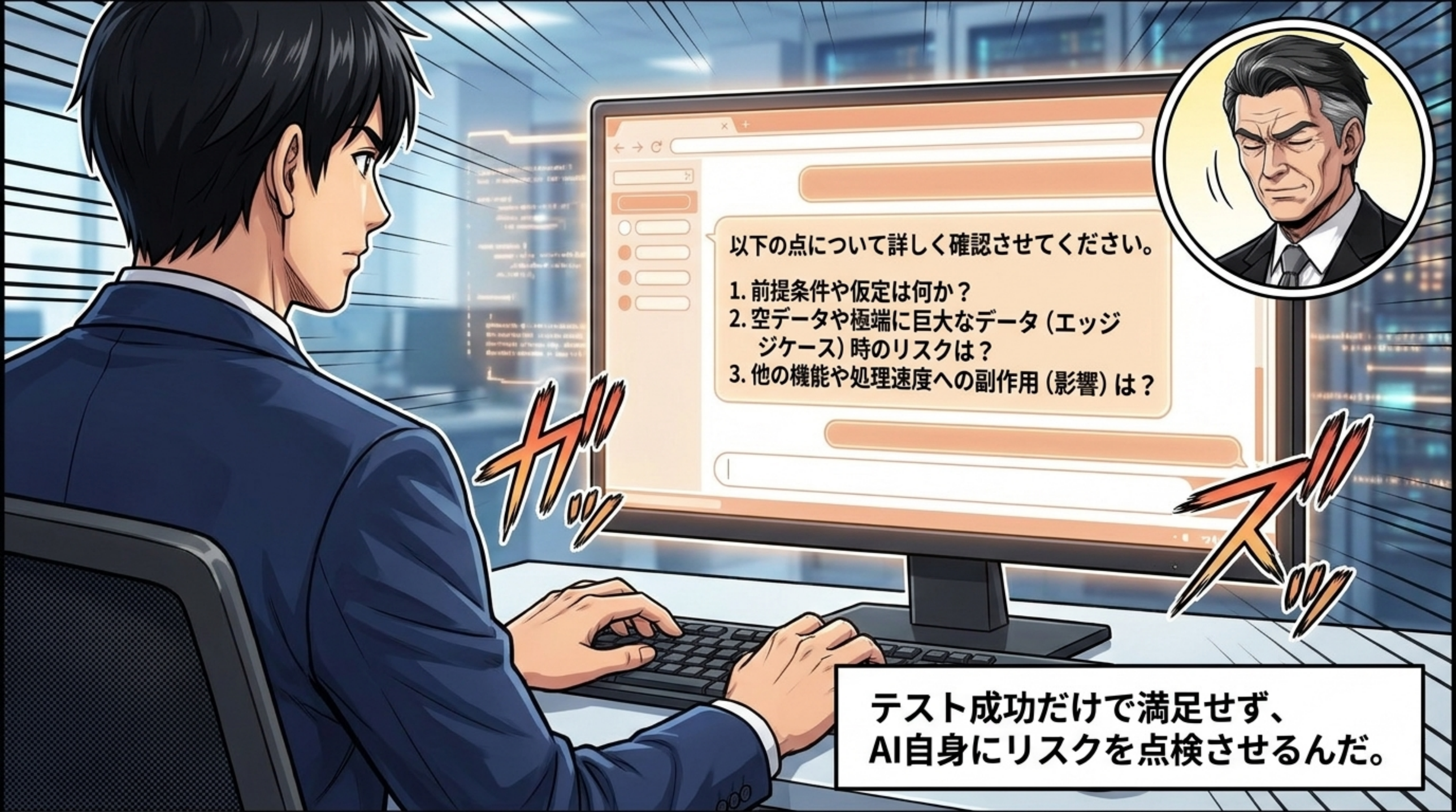

会話が長すぎると、AIは過去の情報を圧縮(コンパクション)して詳細を忘れてしまう。潔く『新しいチャット』を立ち上げ直すのがプロの鉄則だ！
New Chat
フレッシュなセッションに、最新のコード状況だけを整理して再入力しよう。


あれ!? Claudeが急にさっきの指示を忘れて、ちぐはぐな答えを出し始めたぞ!?
あれ!? Claudeが急にさっきの指示を忘れて、ちぐはぐな答えを出し始めたぞ!?


通常のデバッグ
タイポ修正・速さ重視（思考少）
複数ファイルの複雑なバグ・
徹底的な自己検証（思考最大）
low
medium
(Auto)
high
/ max
Claude Codeを使う時、
この『エフォートレベル』は
頭の良さのダイヤルではない。
思考量、ツール呼び出し回数、
二重チェックの徹底度…つまり
『どれだけ懸命に汗をかいて
働くか』を制御するダイヤルだ！


バグが解決しない
①【まずエフォートを上げる
(low → high)】
※モデルはそのまま
(それでもダメなら)
②【モデルを変更する(Sonnet等へ)】
③【モデルを変えたらエフォートは
デフォルト(medium)に戻す】
※リソース過剰消費を防ぐため
バグが解けない時、
いきなりモデルを
賢いものに変えな！
まずやるべきは
『エフォートの
の引き上げ』だ！


4Dフレームワーク
Delegation:委任
Description:
指示
Discernment:
見極め
Diligence:責任
待て！
動いたからといって盲信するな！
品質と挙動の最終責任 (Diligence) は、
AIではなく『君 (ビルダー)』にある。
見極める力 (Discernment) を持て！
よし！テスト通った！
早速本番環境に
コミットだ！
TERMINAL
PASSED


以下の点について詳しく確認させてください。
1. 前提条件や仮定は何か？
2. 空データや極端に巨大なデータ（エッジケース）時のリスクは？
3. 他の機能や処理速度への副作用（影響）は？
ガッ
ス
テスト成功だけで満足せず、
AI自身にリスクを点検させるんだ。

CLAUDE.md（プロジェクト記憶）

過去の不具合と再発防止ルール：
CSVのNull値は例外を出さず補完すること

プロジェクトフォルダに置けば、
Claudeが次回から
自動で読み込む！

AIデューデリジェンス声明

使用ツール：Claude Code
レビュー内容：ファクトチェック実施
最終責任者：[自分の名前]

透明性を保ち、
チームの信頼を得るための
5つの必須要素



これでもう、同じバグ
は二度と繰り返さ
ないですね！

さあ、次は君の番だ。
まずは今抱えているエラー
一行、Claudeにコピペする
『最初の一歩』から始めよう！

暗い画面と格闘した時間は終わりだ。
デバッグは今や、AIと一緒に解き明かす
『面白いパズル』になった！



- [x] エラーログをコピーする
- [x] Claude.aiを開く
- [x] 「原因と修正案を教えて」と送信する！