

生成AIによる プログラミングの始め方

第一回：AIコーディング入門

大阪大学D3センター 谷口昂平

自己紹介

名前：谷口昂平（たにぐち こうへい）

所属：伊達研究室（D3センター）

肩書き：大学院生（博士課程）



研究内容：ネットワークの遅延保証

⚠ **生成 AI 自体の研究者ではない**

生成 AI との関わり：100% ユーザー側

- 修士2年（2022）で出たての ChatGPT 3.5 使って修論の実装
- 博士課程の間も最新の AI ツールに乗り換えながら使ってた

AI コーディングを学ぶ人の広がり

官邸でも AI 講習

8月28日，高市首相が体験



声で頼んで、
物価高対策の試算アプリを作成

この講座の登録者

434 名

学内外からの登録

学外 53.2%

学内 41.5%

未判定 5.3%

立場別の割合



セミナーのスタンス

AI の専門家による良質な無料資料は豊富



MIXI の 2026 年の新人研修資料
(150分で生成 AI の仕組みがわかる)



東京大学 メタバース工学部のセミナー
(90分×5回・実践寄り)

いい加減なことも言います．意見はすべて個人の見解です
生成AIでコーディングしてきた経験と勘所を共有

セミナーの内容

第1回：9/10 10:00 - 12:00 今回	ご挨拶 大阪大学D3センター 先進高性能計算基盤システム研究部門 伊達 進
	AIコーディング入門：座学（デモあり） AIコーディングの歴史、LLMモデルとハーネス、Cursorでスパコンに接続～ジョブ実行スキル作成 (講師：大阪大学D3センター 先進高性能計算基盤システム研究部門 / 大阪大学大学院 情報科学研究科 谷口 昂平)
第2回：9/17 10:00 - 12:00	もっと賢くAIコーディング：座学 プランモード、サブエージェント、ループエンジニアリング (講師：大阪大学D3センター 先進高性能計算基盤システム研究部門 / 大阪大学大学院 情報科学研究科 谷口 昂平)
第3回：9/24 10:00 - 12:00	もっと安くAIコーディング：座学 トークン節約テクニック、ローカルLLM、無料モデル（OpenCodeなど） ※利用可能なモデル等の状況により、内容を変更する場合があります。 (講師：大阪大学D3センター 先進高性能計算基盤システム研究部門 / 大阪大学大学院 情報科学研究科 谷口 昂平)

第1回

AI コーディングを
とりあえず使える

第2回

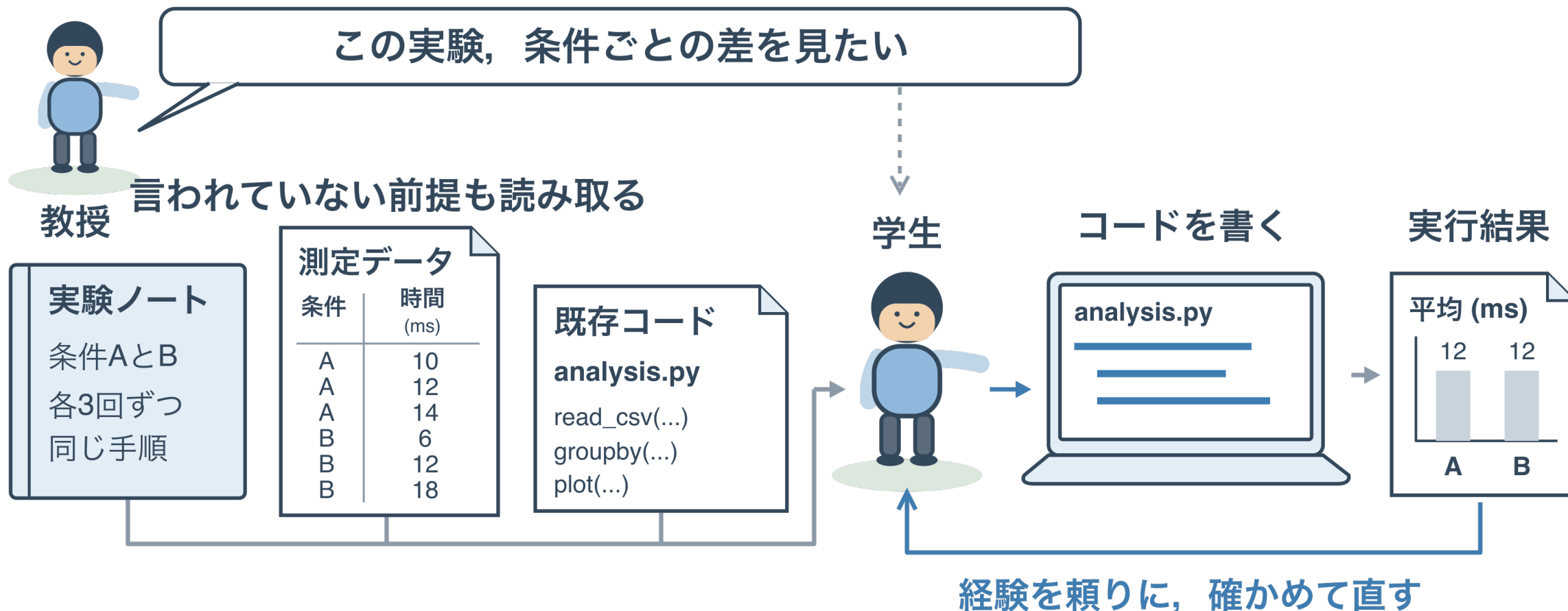
AI コーディングで
難しいものを作る

第3回

AI コーディングで
安く作れる

ポランニーのパラドックス

経験のできる仕事でも、判断の勘所をすべて言葉にはできない



こんな複雑なコーディングを、なぜAIはできるのか？

今日のお品書き

1. AIコーディングの歴史

現状のAIコーディングを把握する

2. モデルとハーネス

AIが考え、操作する仕組みを知る

3. コーディングエージェントの選び方

用途と情報管理、予算から製品を選ぶ

4. スパコンでジョブ実行

学んだことを使い、スパコンをAIで操作する

今日のお品書き

1. AIコーディングの歴史

現状のAIコーディングを把握する

2. モデルとハーネス

AIが考え、操作する仕組みを知る

3. コーディングエージェントの選び方

用途と情報管理、予算から製品を選ぶ

4. スパコンでジョブ実行

学んだことを使い、スパコンをAIで操作する

AI コーディングの歴史

潮流を作った5つのツールで、5世代に分ける

1. Github Copilot : コーディング**補完**
2. ChatGPT : **チャット**コーディング
3. Windsurf : **自動**コーディングツール?
4. Claude Code : **自律**的なコーディングツール
5. OpenClaw : コーディング**エージェント**
(コーディング**以外**も?)

類似資料 : [コーディングAIエージェントの歴史と関連技術](#)

第一世代：コーディング補完

2021年の Github Copilot は
コードを途中まで書くと続きを補完してくれた

例：数字の配列から素数を抽出

```
17  int main() {  
18      // 例：入力配列  
19      std::vector<int> numbers = {2, 3, 4, 5};  
20      std::vector<int> primes;  
21  
22      // 配列から素数を抽出  
23      for (int i = 0; i < numbers.size(); i++) {
```

 人間がキーボードを
打つ量は減った

 途中まで書ける
コーディング力は必要

人間が記述

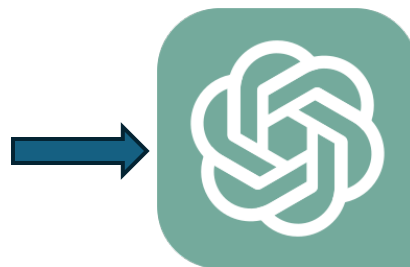
続きを予測してAIが記述

第二世代：チャットコーディング

2022年 11月に ChatGPT が登場

コードをチャット（人間の言葉）で生成できるように

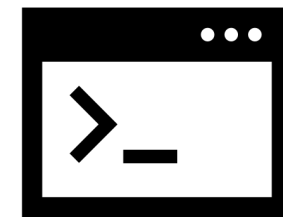
プロンプト（依頼内容）
数値配列から素数を抽出する
C++のプログラムを書いて



ChatGPT 3.5
(当時は緑色)



完璧？なプログラム



PCにコピーして
実行

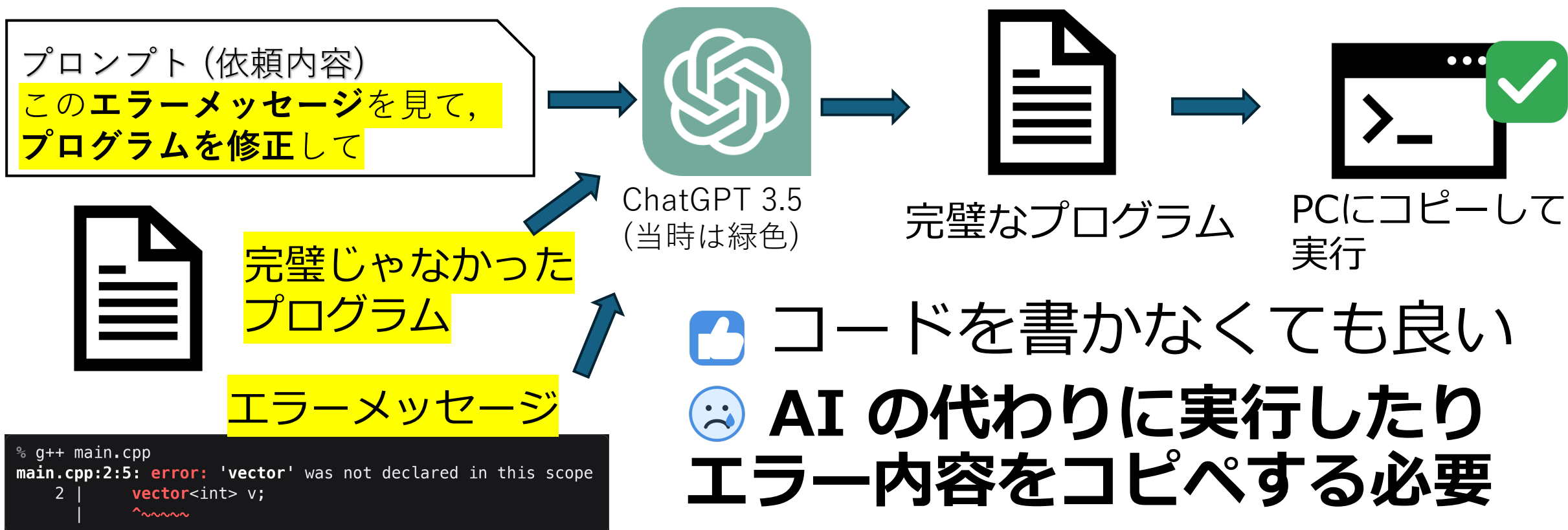
```
% g++ main.cpp
main.cpp:2:5: error: 'vector' was not declared in this scope
  2 |     vector<int> v;
    |     ^~~~~~
```

意味：配列を使うのに
必要なライブラリを入れ忘れ

第二世代：チャットコーディング

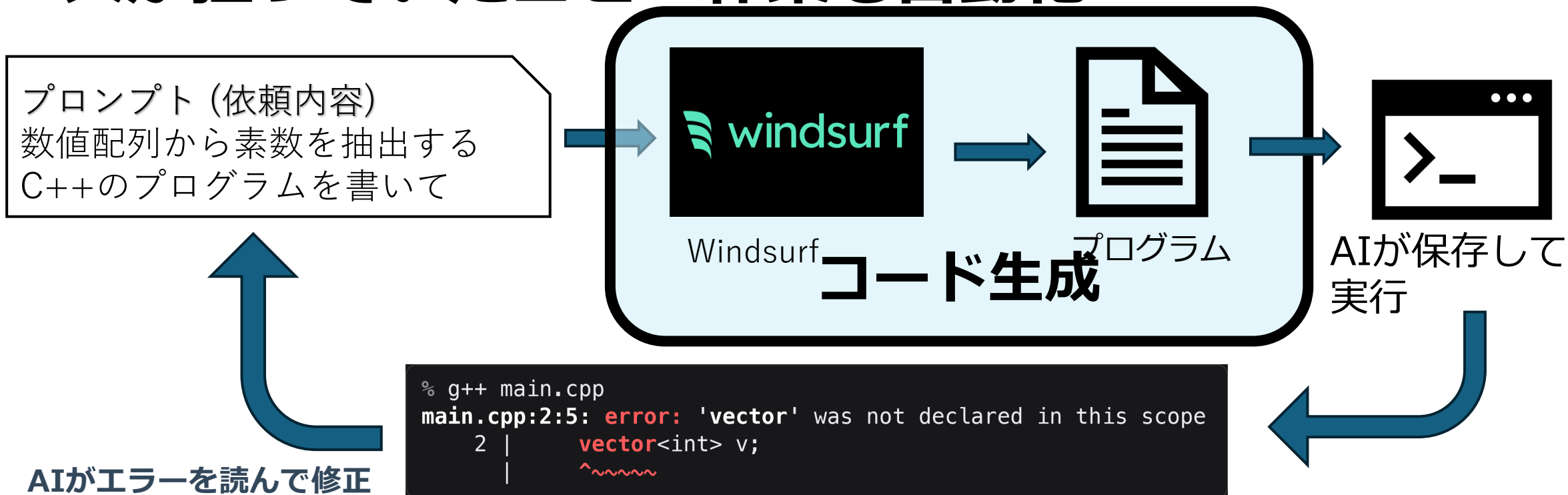
2022年 11月に ChatGPT が登場

コードをチャット（人間の言葉）で生成できるように



第三世代：自動コーディング

2024年 11月に Windsurf (現Devin) が登場
人が担っていたコピペ作業も自動化



勝手にテスト～修正もしてくれるなら理想郷か？

第三世代：自動で迷子になるコーディング

Windsurf など当時のツールは、
プロンプトを機械的に処理して探索しがち

プロンプト（依頼内容）
数値配列から素数を抽出する
C++のプログラムを高速化して



英語でNumber,
Vector, Primeで
コードを検索しよう

配列名は v, 素数は p... ⇒ 見つからず迷子



コピペの手間がなくなって見るだけに



AI が迷子にならないように整理しておく必要

AI 向けにコードを書くとは？

ネット検索



+ おいしくて安い大阪のたこ焼き

AI モード →

✖ 文章で検索

おいしい 安い 大阪 たこ焼き

+

AI モード →

✔ キーワードで検索

翻訳サイト

たこ焼きはソースだよね
Takoyaki is sauce, right?
≡ たこ焼きは液体ですよ
ね？

✖ 雑な日本語

ソースをかけると
たこ焼きが一番おいしい
Takoyaki tastes the best
with sauce.

✔ 丁寧な日本語

AIコーディング

```
def proc(d):
```

↓ AIの解釈
何の処理？ 引数は何？
≡ 別の処理を実装

✖ 適当な命名

```
def  
bake_takoyaki(order):
```

↓ AIの解釈
たこ焼きを焼く処理
≡ 意図どおり

✔ 名が体を表す命名

第四世代：自律的なコーディングツール

2025年 2月に Claude Code が登場
人間を模して、自律的に考えて進む

```
17  int main() {  
18      // 例: 入力配列  
19      std::vector<int> numbers = {2, 3, 4, 5};  
20      std::vector<int> primes;  
21  
22      // 配列から素数を抽出  
23      for (int i = 0; i < numbers.size(); i++) {
```

README.md

配列から素数を取り出す

1. `g++ main.cpp -o primes`
2. `./primes` を実行
3. 出力された素数を確認

ソースコード内のコメント
コメントを見て処理を把握

付属の手順書 (README.md)
手順書どおりに実行

第四世代：自律的なコーディングツール

Q. 人間の手順を真似すると何が嬉しい？

**A. 人が読みやすいコードや手順書は
AI も迷子になりづらい**

本セミナーは、本センターの先進高性能計算基盤システム研究部門（伊達 教授）の協力を得て、同部門に所属する大学院生がセミナープログラムを開発し、講師を務めるものです。院生自身の経験を踏まえ、これから「生成AIによるプログラミング」を考えておられる方、**少しかじったものの諦めてしまい、再挑戦されたい方**などを念頭に実施頂くもので、この機会に分からないことを気軽にご質問いただけます。



つまづきポイント

第四世代より前のAIコーディングは、特有の準備や知識が必要

⇒ それより前で挫折した人は落ち込む必要はない

第四世代に足りないもの

自動テストは「コードが動く」まで
文字切れ・押しにくさは人が画面で発見

「グリッドレイアウトでサブスクリプションの選択画面を作って」

✓ コード検査：エラー 0



ここから人間

目で見ると
文字切れ
触って試すと
押しにくい

AIは見逃す



⚠ つまづきポイント：画面のテストは人間頼みだった

第五世代：画面もテストする

2026年1月に OpenClaw が登場

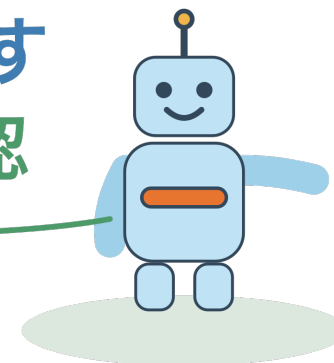
見る・押す・直す・再確認を自分で回す

「グリッドレイアウトでサブスクリプションの選択画面を作って」

修正後：文字もボタンも確認済み✓



- ① 🖥️ ブラウザを開く
 - ② 📷 スクショを見る
 - ③ 🖱️ マウスで押す
 - ④ ⚙️ 崩れを直す
- 再び画面を確認



コード作成から画面の動作確認までAIが引き取る

余談：自分はギユられないと思っていた

ギユられる = シンギュラリティで人間より賢い AI に置き換えられる
論文調査も環境構築も計測も持っていかれた。残ったのは…

エージェントが全部やる

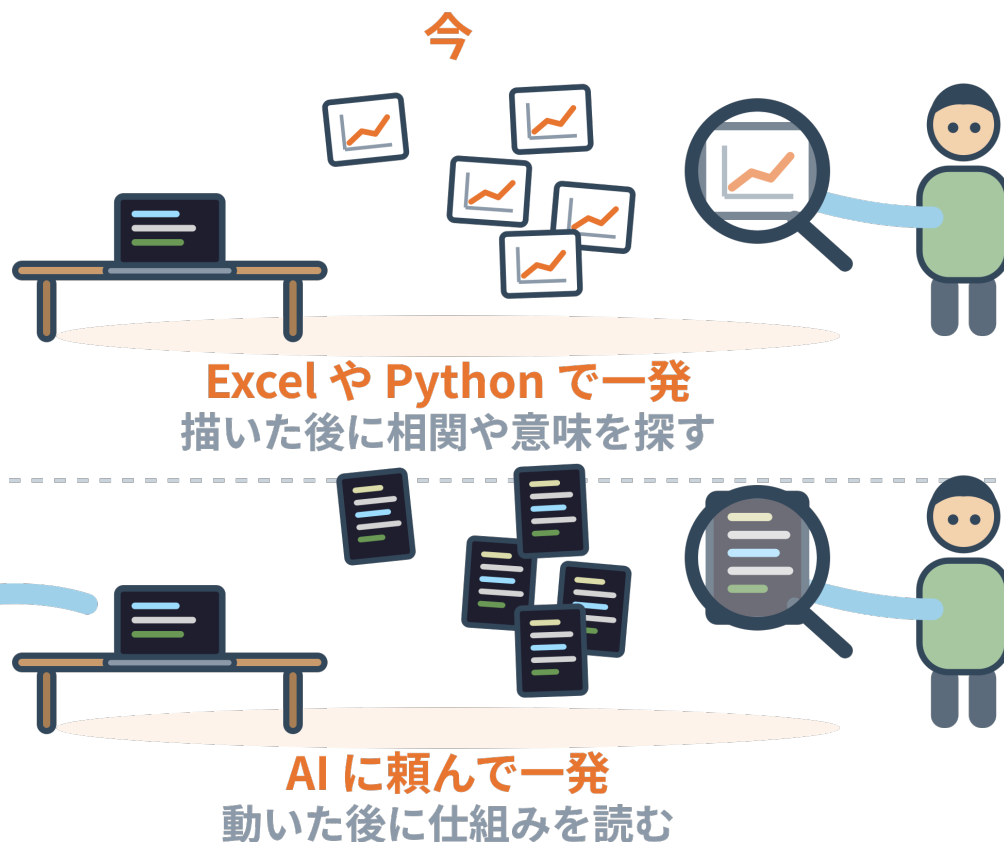


残ったのは3つ



余談：作ってから考えるようになった

方眼紙に定規で描いていた頃は，描く前に考えるしかなかった
コーディングが一発になって，同じ入れ替わりが起きた



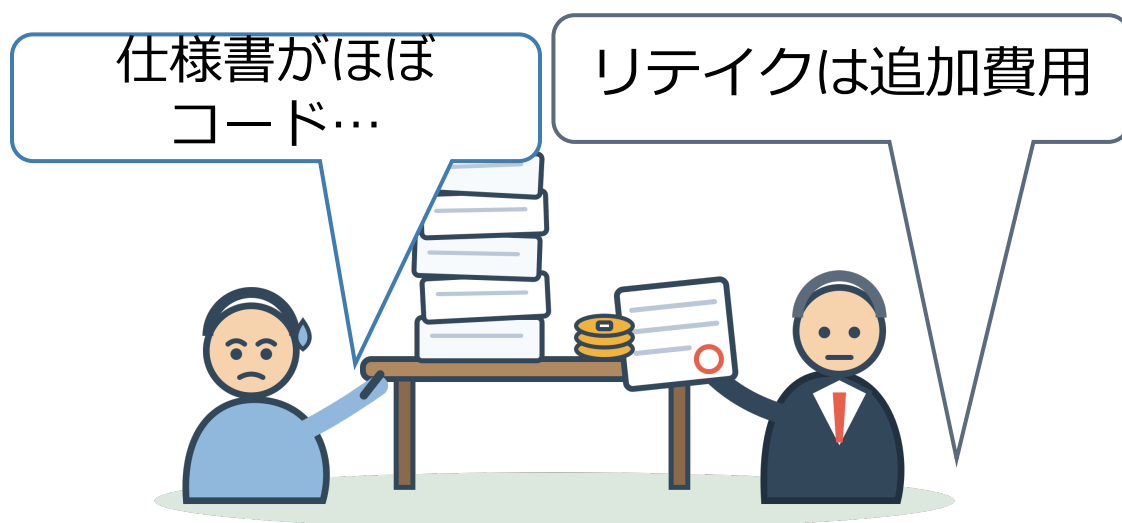
現在：Vibe Coding（ノリでコーディング）

お願い → 実行 → 動けばヨシ！

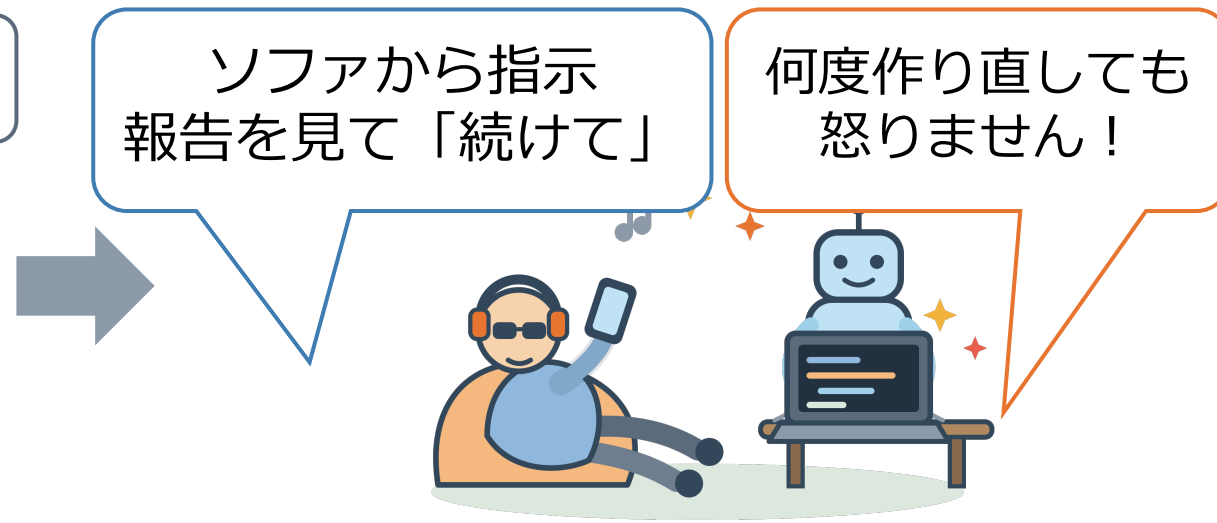
開発の "外注" がとことん気軽になった

「完全にバイブスに身を任せ、コードの存在すら忘れる」 — Andrej Karpathy

これまで：開発委託



これから：Vibe Coding

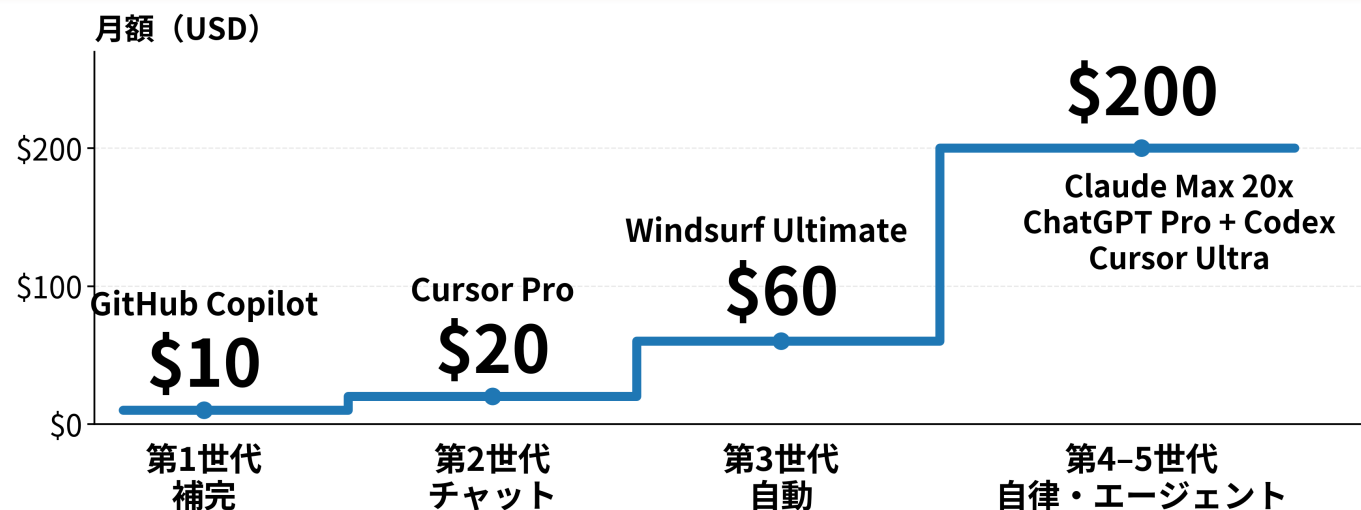


まずは気楽に始めましょう！

コーディングエージェントを使いこなすには？

1. 何をしてほしいかを**言葉にする** = **言語化能力**
 2. エージェントが必要な**操作権限を渡す** = **信じて託す**
- この2つでなんとかなると言いたいところだが

0. **¥** 良いコーディングエージェントを契約する **¥**



最上位のエージェントは
適当に使っても動く

形から入るのが
一周回って良い

どのくらい値段によって違うの？

¥0 / 月



公園の名もなき器具

- ✕ 使い方も効果も不明
- ✕ 雨や混雑で使えない

フリープラン

- ✕ 使えるモデルが限られる
- ✕ 確認だけで力尽きる



¥3,000 / 月

無人ジム

- ✓ 定番マシンが使える
- ✕ メニューは自分で

Plus プラン

- ✓ 定番モデルで完成まで使える
- ✕ 複雑な実装にはコツがいる

¥30,000 / 月

パーソナルトレーナー

- ✓ 最高級マシン
- ✓ メニューもお任せ

Pro Max プラン

- ✓ 最上級モデルを長く使える
- ✓ 複雑な実装も任せやすい

努力で越えられる壁

越えられない壁



つまづきポイント

フリープランはスタートラインより手前。 努力不足ではない

余談：実際どのくらい課金してるの？

**普段は ChatGPT Pro x20 (30,000円) のみで
研究3つほどを並行してコーディングできる**

⇒ 普通の人には 3000 円の方で十分

6月ごろに、博士論文、国際会議、研究プロジェクトが
3つ重なってアホほど忙しかった

- Claude Max x20, ChatGPT Pro : 各30,000円
- Z.ai, Cursor : 各6,000円

-72,000円
Wow!

AIコーディングを使いこなすには？

ソフトウェア工学に基づくプロンプト・環境構築

ソフトウェア工学とは、一言で言うと、品質の高いソフトウェアを低コストで期限通りに開発し、効率よく保守するためのさまざまな技術を扱う学問分野です。（[楠本研究室ホームページ](#)）

依頼者の要求をプログラマーに正しく伝えて

あなた コーディングエージェント ⇒ 第二回：もっと賢く

限られた人員で効率よく開発するための方法論

安く

⇒ 第三回：もっと安く

今日のお品書き

1. AIコーディングの歴史

現状のAIコーディングを把握する

2. モデルとハーネス

AIが考え、操作する仕組みを知る

3. コーディングエージェントの選び方

用途と情報管理、予算から製品を選ぶ

4. スパコンでジョブ実行

学んだことを使い、スパコンをAIで操作する

モデルとハーネス

コーディングエージェントを分解すると

- **モデル**：脳として次にやるべきことを考える
- **ハーネス**：目や手として状況を見せたり実行を担当



良いモデルとは何か？

普段のチャット形式では、回答の早さと正確さ

コーディングエージェントでは、
モデルは長い作業で判断を重ねる

1. 推論速度

試行錯誤のサイクルが速く

2. コンテキスト長

来た道を忘れない記憶力

3. 価格

ご飯 (お金) が少なく済む



5. プロンプト追従度

指示していない事を
どこまでやるかの味付け

4. 思考 (Thinking)

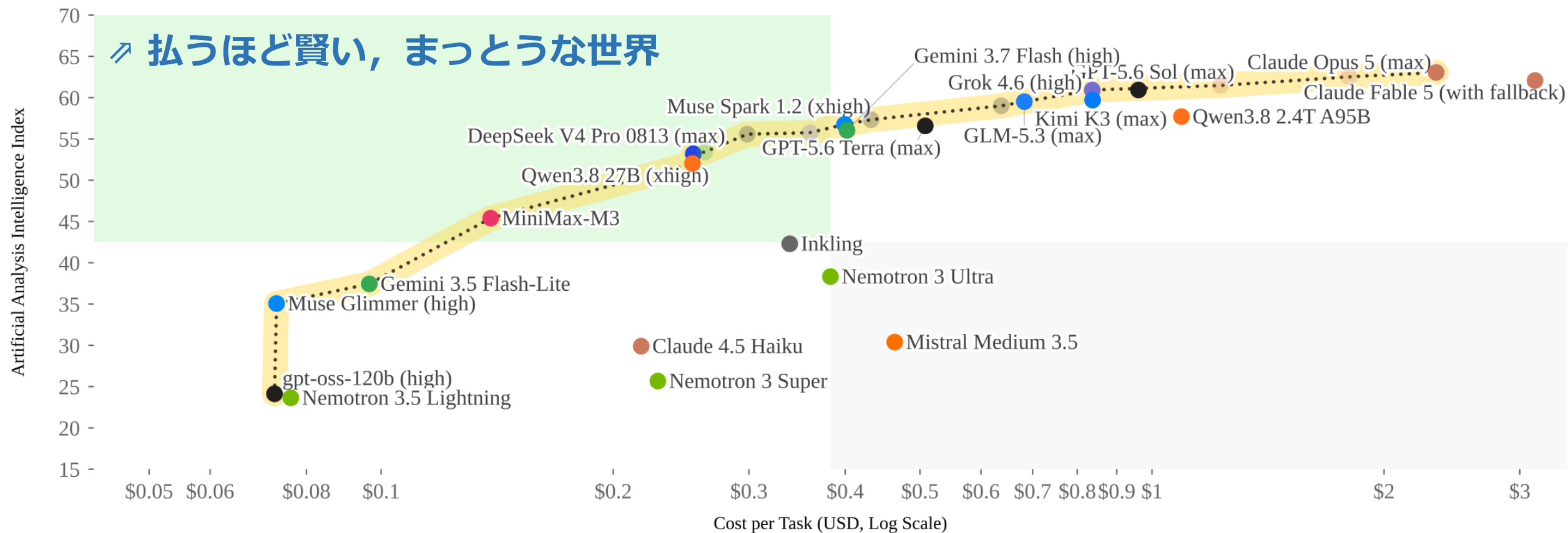
迷子になってないか
振り返れる力

コスト vs 性能

賢いモデルほど高い ⇨ 各価格帯に "コスパ番長" の居場所 があった

--- マーカーがパレート線 = その値段で一番賢いモデルたち

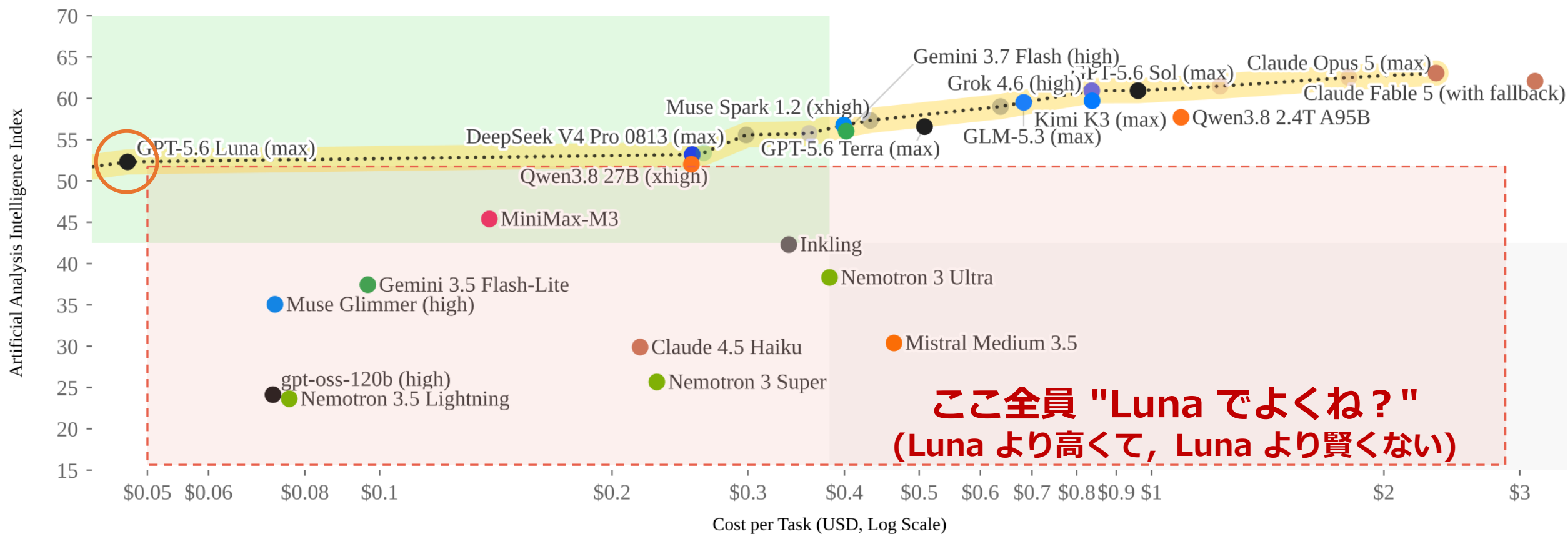
✧ 払うほど賢い, まっとうな世界



コスト vs 性能

7/30 Luna 80%オフ ⇒ 左半分が真横に. **中堅の存在理由を破壊**

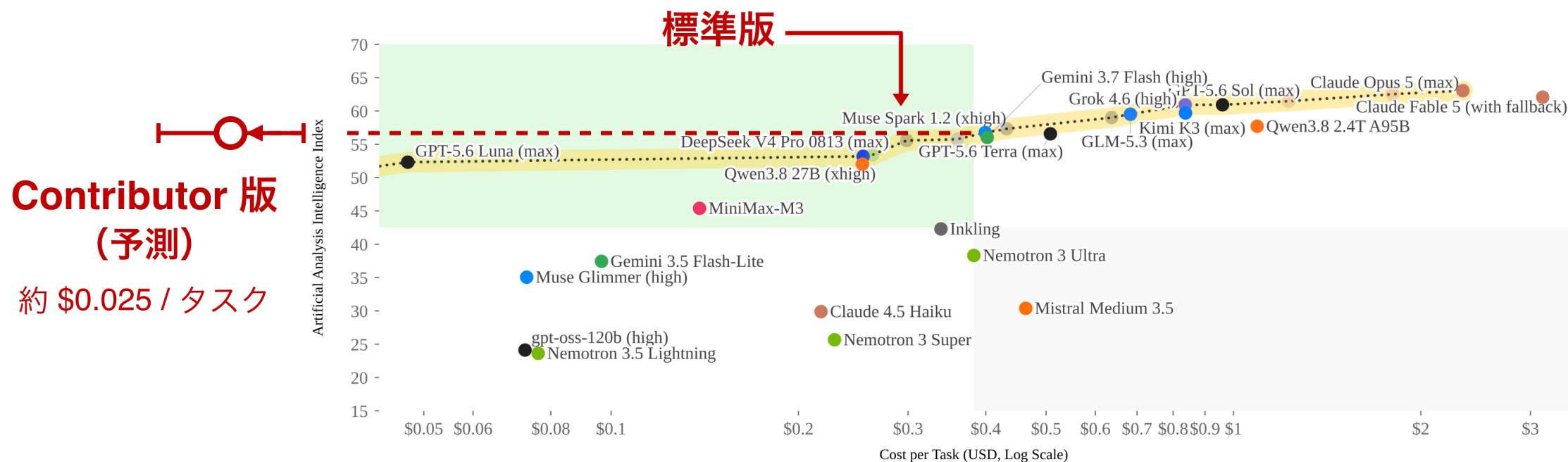
--- マーカーがパレート線 = その値段で一番賢いモデルたち



コスト vs 性能

Muse Spark の Contributor 版：安さの条件は入出力の学習利用

出力100万トークン 標準版 \$4.25 → Contributor 版 **\$0.20** **約1/20**



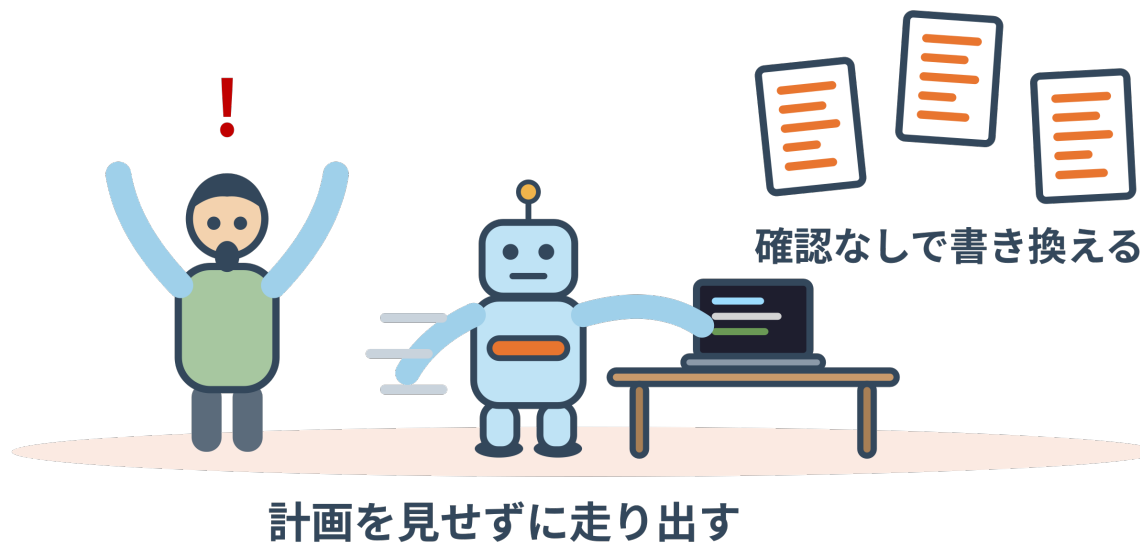
つまづきポイント

未公開データを出す前に，学習に使われない設定かを確かめる

エフォート：どれだけ考えさせるか

どのモデルを使うかとは別に，考える量を段階で指定する
low だと計画を見せずに走り， **max** だと調べ続けて終わらない

段が低いまま



段を上げすぎ



つまづきポイント

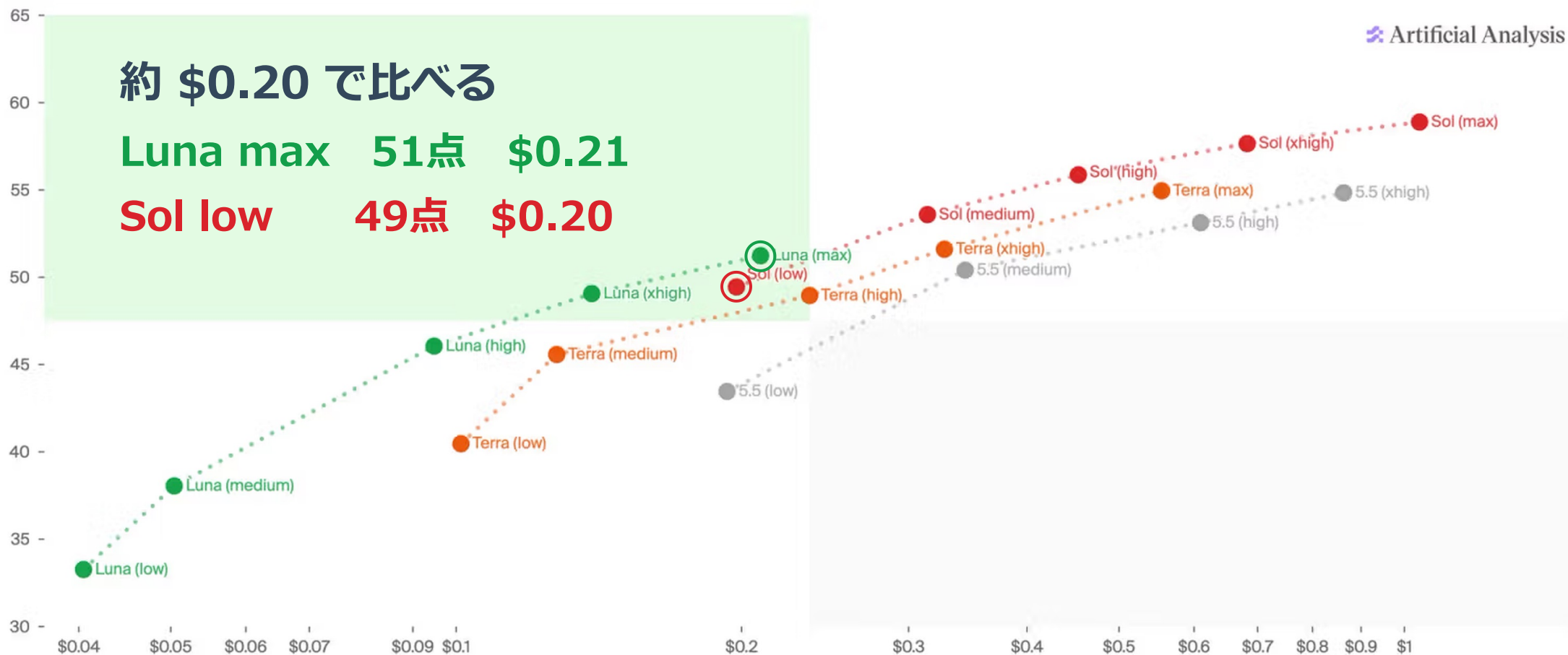
いいモデルをけちって low で使うと，安いモデルに負ける

エフォートとモデル、どちらを上げる？

ほぼ同額でも、Luna max が Sol low を上回る

性能 (Intelligence Index)

横軸：1タスクの費用 (ドル・対数)



エフォートとモデル、どちらを上げる？

費用を約5倍ずつ増やすと、得点は +18点、次は +8点

性能 (Intelligence Index)

横軸：1タスクの費用 (ドル・対数)



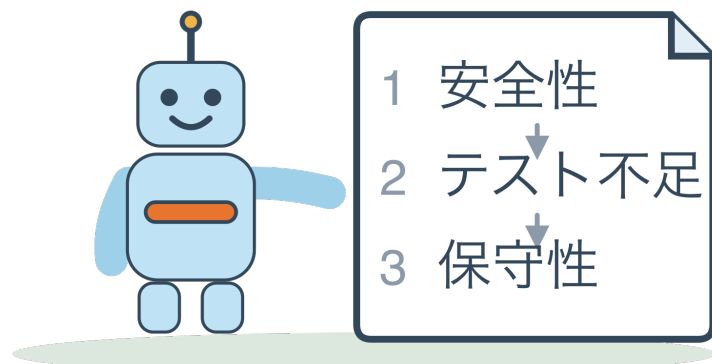
余談：Ultra は手分けして考える

Max は深く考える設定. Ultra は分担・並列化も使う

例：コードレビューを「安全性・テスト・保守性」に分ける

Max：検討を深く

1体でレビューする例



1つの回答にまとめる

検討が長いほど

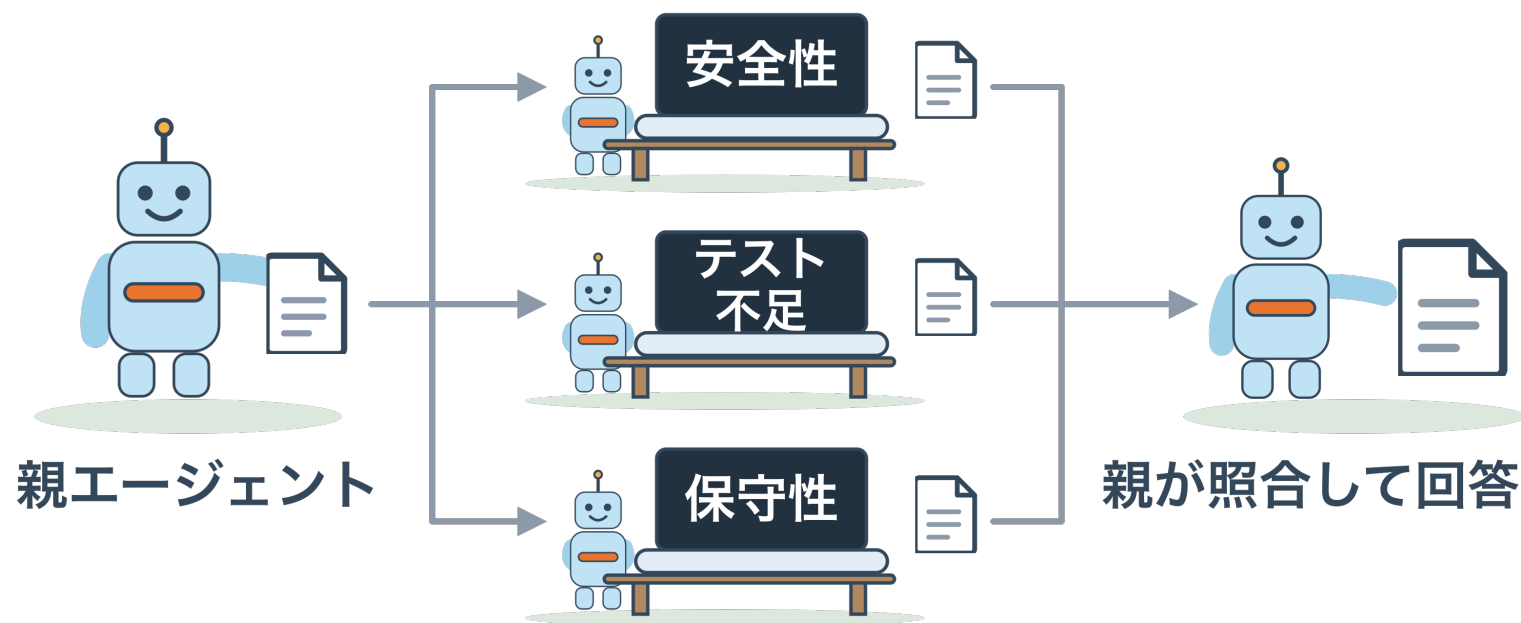
待ち時間・使用量が増える

Ultra：手分けも使う

依頼を分ける

同時に確認する

結果を統合する



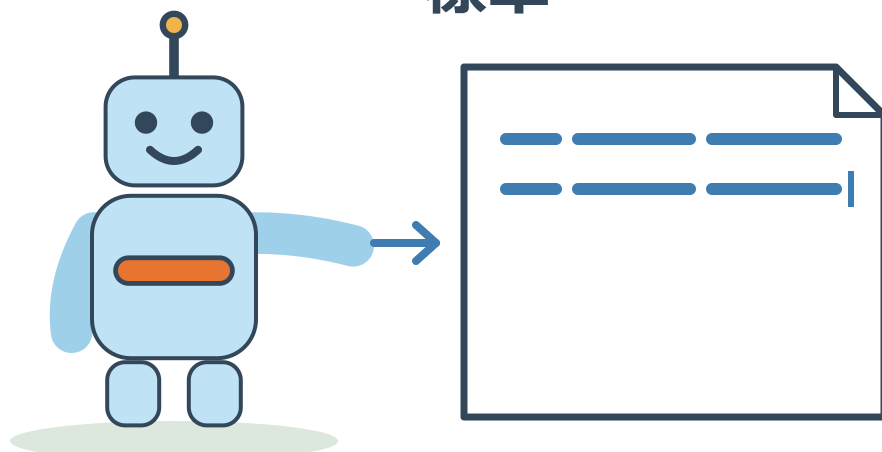
親と子が、それぞれトークンを消費

ファストモード：速さだけ買う

考える力はそのまま、追加料金で文字の生成を速くする

Claude Code の Opus 5

標準

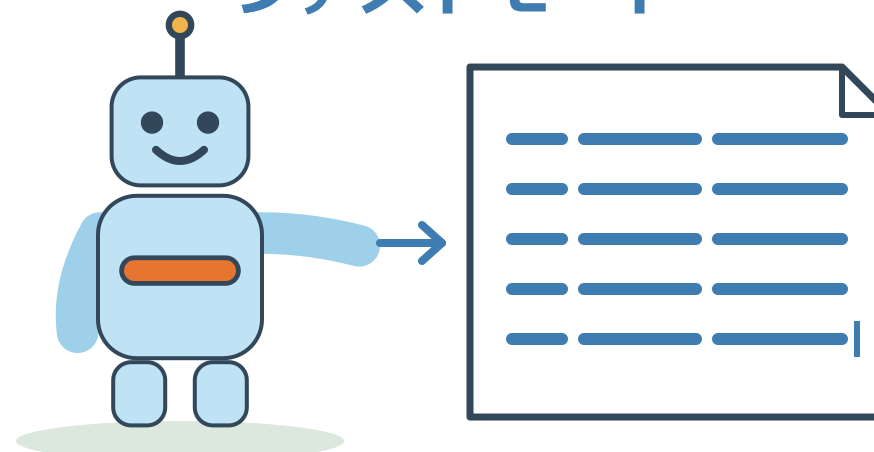


通常の生成速度

● 単価 1倍

同じ時間で比べると

ファストモード



生成 最大2.5倍速

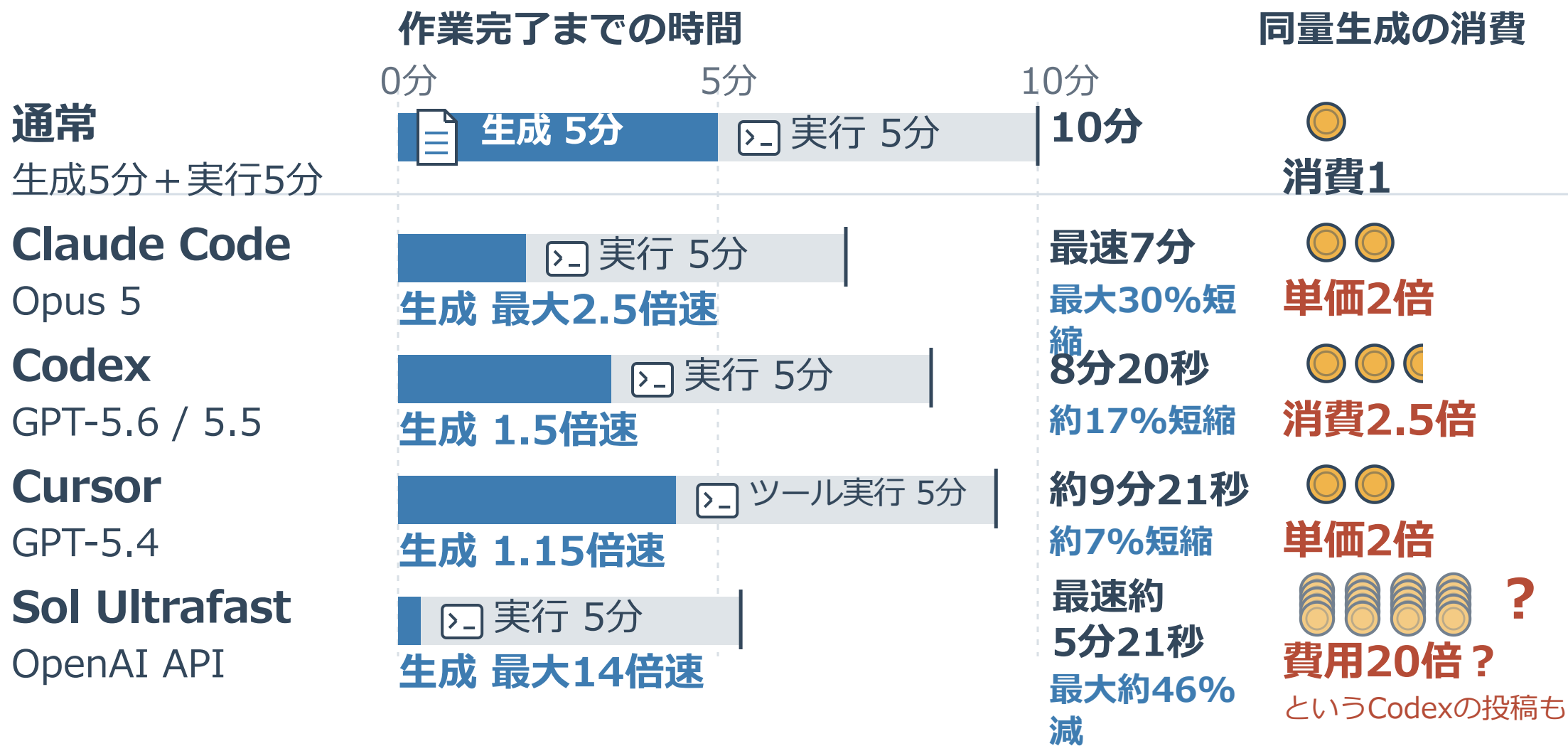
●● 単価 2倍



つまづきポイント：賢さは同じ、速くなるだけ
追い込みや集中して作業するときだけ使う

ファストモード：何分に何倍払うか

ツール待ちが長いと、追加料金に見合いにくい



ガンガンいこうぜ vs いのちだいじに

プロンプト追従度と思考が**モデルの性格**を決める

例：機械学習モデルをC++で作れ

Pythonなら
ライブラリで
すぐできる



ガンガンいこうぜ



ChatGPT



deepseek

コストは気にしない
ひたすら C++ で実装



Claude



Grok

コストを減らしたい
C++ から Python を呼ぶ



Gemini



Meta

コストがかかりそう
Python への変更を提案

いのちだいじに

どの性格が好みかは人次第だが...

ガンガンいこうぜ vs いのちだいに

プロンプト追従度と思考が**モデルの性格**を決める

例：機械学習モデルをC++で作れ

ガンガンいこうぜ



ChatGPT



deepseek



Claude



Grok



Gemini



Meta

(裏の意図) 知ってる言語だから

コストは気にしない
ひたす

コストカット
したい
C++からPython

コストがかかりそう
Python への変更を提案

Pythonなら
ライブラリで
すぐできる



COST



いのちだいに

どの性格が好みかは人次第だが...

ガンガンいこうぜ vs いのちだいに

プロンプト追従度と思考が**モデルの性格**を決める

例：機械学習モデルをC++で作れ

Pythonなら
ライブラリで
すぐできる



ガンガンいこうぜ

(裏の意図)Pythonは遅いから嫌



ChatGPT



deepseek

コストは気にしない

ひたすら C++ で実装



Claude



Grok

コストを減らしたい

~~C++ から Python を呼ぶ~~



Gemini



Meta

コストがかかり

Python → ~~Python~~

ただのサボり



いのちだいに

どの性格が好みかは人次第だが...

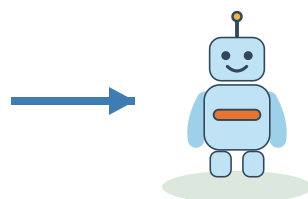
余談：AGENTS.mdと指示の増やしすぎ

出典：Gloaguenほか「Evaluating AGENTS.md」(2026.2)

AGENTS.md

C++を使う
実行速度を優先する

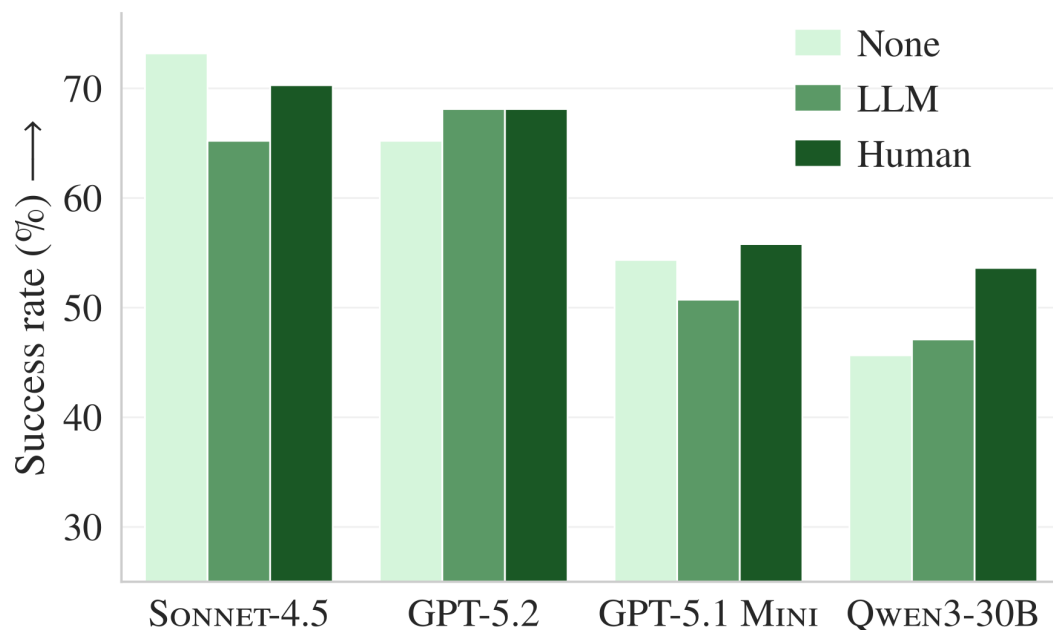
作業前に読む



共通の方針をAIに伝えるファイル

提案の仕方や慎重さも，ある程度は変えられる

138課題・12プロジェクト・4種類のAIで比較



各組の左から：指示なし・AI作成・人が作成

GPT-5.2：手数と費用の平均

指示ファイル	手数	費用
なし	12.1	\$0.38
自動生成	13.1	\$0.57
開発者記述	13.6	\$0.54

最近は，性格を変えようと

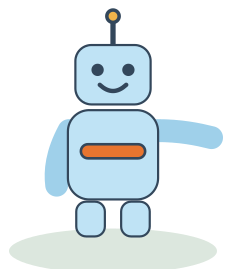
指示を重ねるのをやめました

余談：Claude語を抑える公式プロンプト

出典：Anthropic「[Prompting Claude Fable 5.1](#)」 Writing density

Table 5.1：mannered prose（比喻で回りくどく飾る文体）

日本語では，こんな言い回し



この設計は，拡張性への
扉を開きます



この設計なら，あとから
機能を追加しやすくなります

AGENTS.md

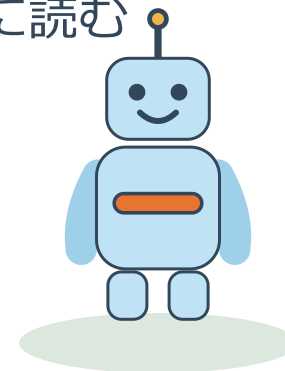
公式の短い指示

Please remove all mannered prose.

比喻で飾りすぎず，言いたいことを直接書く

Claude Codeでは CLAUDE.md

作業前に読む



公式の対処例を，共通の指示に取り入れるのはよい

良いハーネスとは何か？

ハーネス (harness) の語源は登山の安全帯

⇒ 登山者 (モデル) を支える装備

モデルの賢さをどれだけ効率よく引き出せるか

1. ツール

ファイル編集・実行
Web検索・画面操作

2. コンテキスト管理

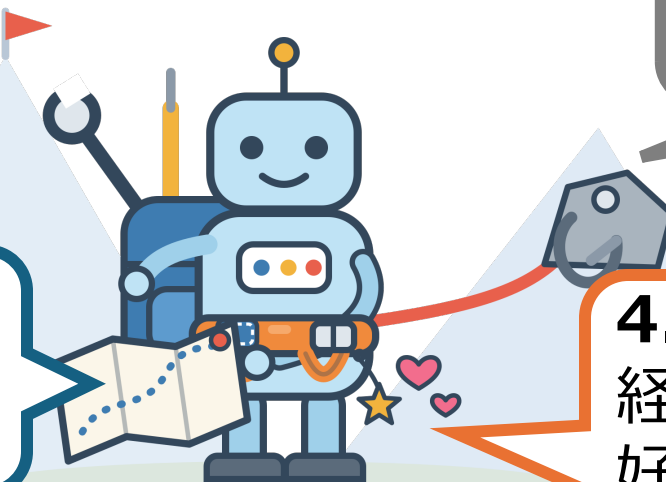
限られた長さに収めて
必要な道具やコードを渡す

3. ガードレール

隔離環境と事前確認で
破壊を防ぐ

4. パーソナライズ

経験を保存し
好みや手順を次に生かす



ハーネスは公式一択？

モデルの性格や思考,
コンテキスト長は
自社ハーネスに最適化

Hermes, PI などの
**他社製ハーネスを
無理に使う必要なし**



Elon Musk  
@elonmusk

購入する



 英語からの翻訳 [原文を表示](#) 

Grok 4.6 は、Grok Build ハーネスと一緒に使うと最適に動作します。それがないと体験が大幅に悪くなるので、Build を使って評価するのが最も良いでしょう。

Grok.com/build

この翻訳を評価:  



Arthur MacWaters   @ArthurMacwaters · 8月14日

 英語からの翻訳

これまでの使用から、4.6 は Grok Build に対して非常に最適化されているようです。 x.com/ArthurMacwater...



Grok Build 1.0.3

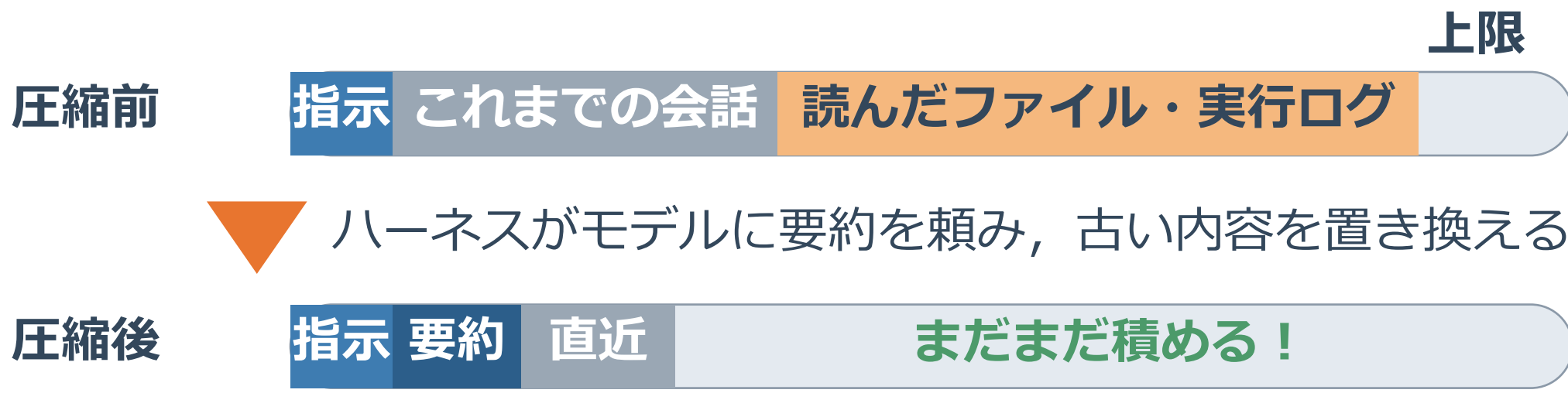
Grok 4.6 is here!
Select 'Grok 4.6' under /model.

New worktree
Resume session
Changelog
Quit

ctrl+w
ctrl+s
ctrl+q

コンテキスト圧縮とは

会話・読んだファイル・実行ログはコンテキストへ
上限が来たら，古い内容を要約して空きを作る = 圧縮



要約は，アニメの「前回のあらすじ」

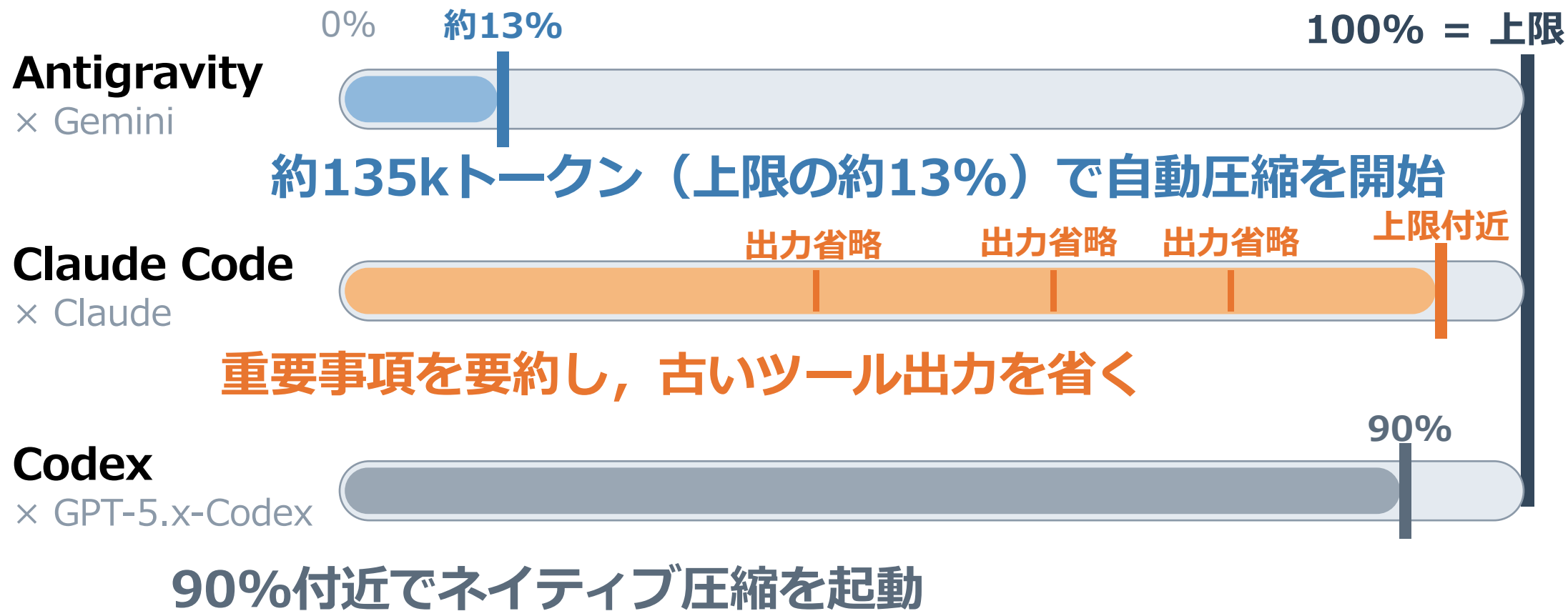


つまづきポイント

圧縮が下手だと "はじめまして" に戻り，ずっと迷子に

各ハーネスのコンテキスト管理

長い作業では、会話・コード・実行ログが積み上がる
各ハーネスは違う方法とタイミングで情報を整理する



公式ハーネスの利点：パーソナライズ

チャットや履歴を勝手にメモリ・スキル化

⇒ 公式で揃えると、**チャットの履歴まで**素材になる



つまづきポイント

いろんなツールに浮気するほどパーソナライズされづらく

公式ハーネスの利点：クラウドVM

粘り強い試行錯誤を支える，つけっぱなしのPC

公式はVM込み ⇨ "つけっぱなし" がスマホだけで手に入る

なぜタダ？ コピー機とインクの商売

VM=本体（誤差）

LLM=インク（儲け）

コピー機



インク



スマホで「つけっぱなし」
投げる → 寝ている間も走る
→ 通知で帰ってくる



CPU

メモリ



ChatGPT Work のVM

8コア分

20 GiB



約15万円のノートPC

Core 5 120U (10コア)

16GB

今日のお品書き

1. AIコーディングの歴史

現状のAIコーディングを把握する

2. モデルとハーネス

AIが考え、操作する仕組みを知る

3. コーディングエージェントの選び方

用途と情報管理、予算から製品を選ぶ

4. スパコンでジョブ実行

学んだことを使い、スパコンをAIで操作する

コーディングAIの企業利用率

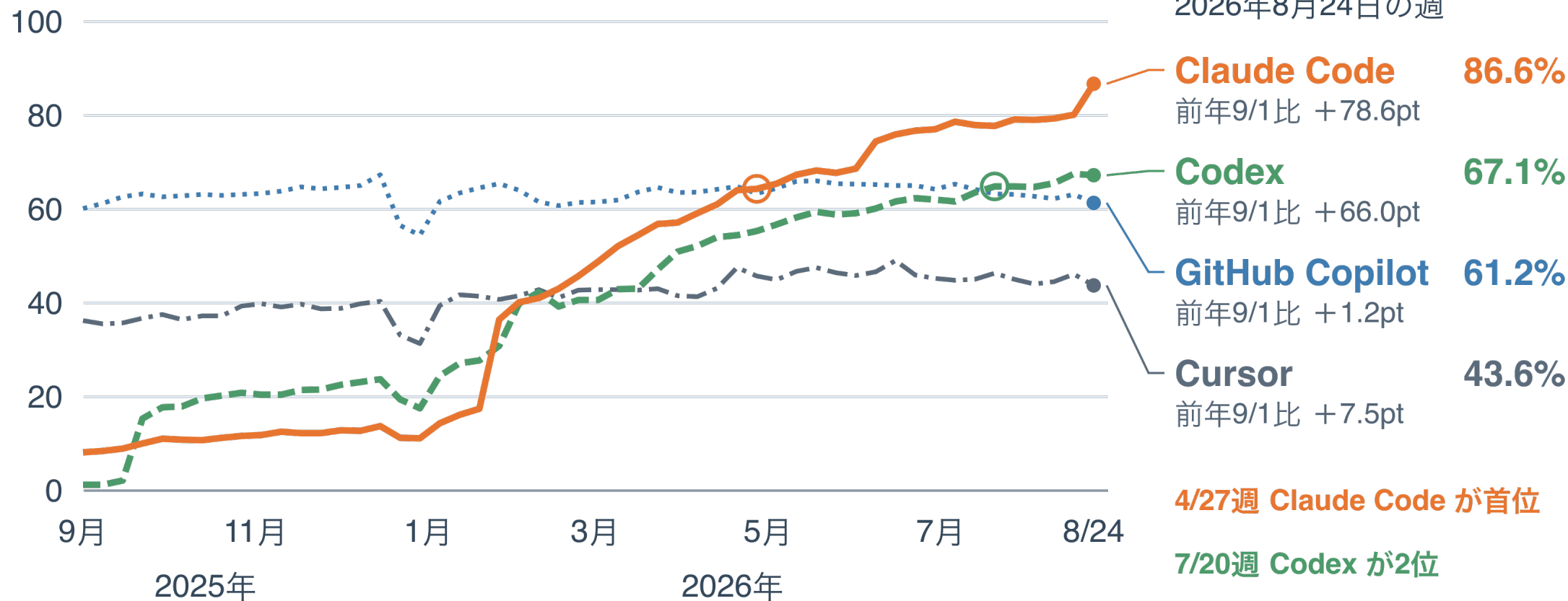
Claude Code が首位, Codex が Copilot を上回る

観測対象企業の利用率 (%)

出典 : Netskope AI Index

同社が観測する企業の通信ログを集計

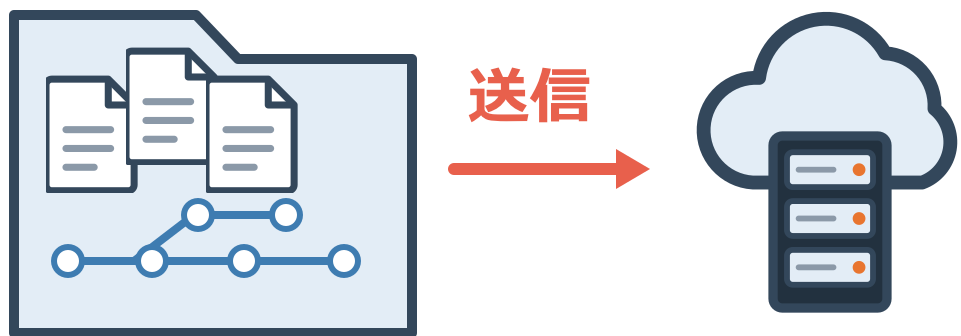
2026年8月24日の週



データ収集をめぐる炎上

Grok Build

2026年7月：外部検証で発覚



ファイル + Git履歴

別の保存先

モデル改善OFFでも送信

その後：保持を既定OFFに変更
過去データの削除を表明

GitHub Copilot

2026年3月発表 → 4月24日開始



個人向けのやり取り

モデル

学習利用は自分でOFFに

Free / Pro / Pro+ の方針変更
Business / Enterprise は対象外

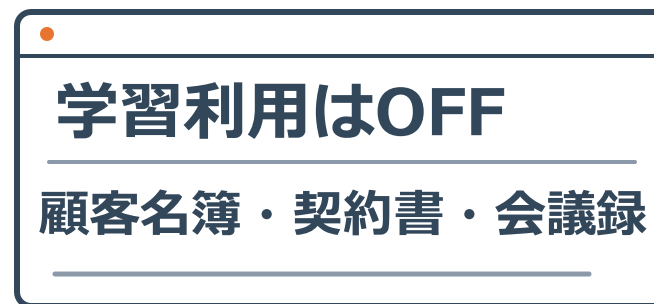
シャドーAI

会社の承認や管理を受けずに、業務でAIを使うこと

職場の資料を入力



個人アカウントに蓄積



社外・退職後も利用



学習不使用でも、個人アカウントは会社から止められない

2025年7月：ChatGPTの共有会話が検索結果に

共有時に検索掲載を許可した会話が、
第三者に見える状態に

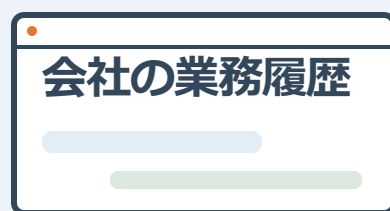
OpenAIは検索掲載の機能を撤回



企業契約で情報と予算を管理する

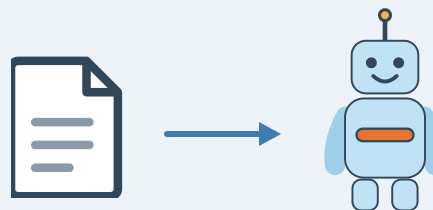
企業契約で管理すること

履歴を会社の管理下に



退職時の停止と共有制限

学習に使わない契約



契約全体の予算管理

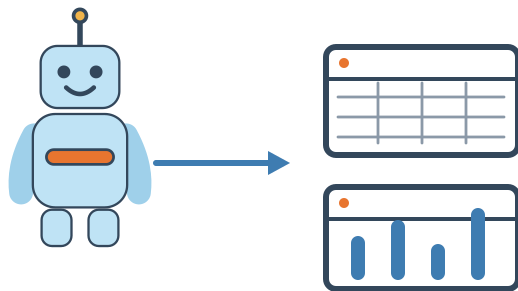


利用枠と追加費用を管理

製品を選ぶ基準

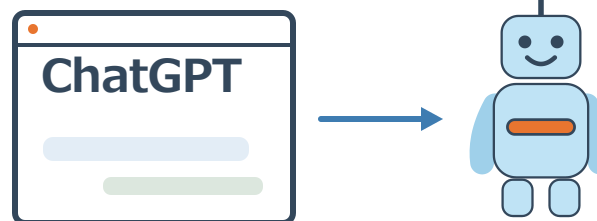
業務機能と周辺ツールとの相性

Claude Code + Cowork



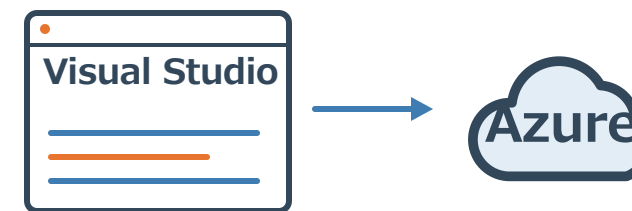
Office作業まで任せる

Codex



既存のChatGPT環境

GitHub Copilot



Microsoft系の開発環境

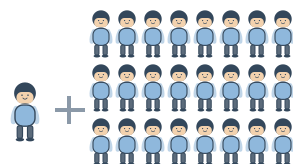
研究者向けの利用支援

理工・生命系の研究室向け。どちらも12か月の支援

Claude Code・Cowork・Science

Standard無料

1研究室・最大25人



PIが申請し、メンバーを追加

Premium：月15ドル／人・5倍

OpenAI ChatGPT・Work・Codex

Pro相当の利用枠が無料

同じ大学・最大5人



研究教員・ポスドクが代表者

直近3年のarXiv等の論文が必要

OpenAI
対象15大学

北海道・東北・東京・東京科学・一橋
横浜国立・名古屋・京都・大阪・神戸・九州
東京都立・早稲田・慶應義塾・国際基督教

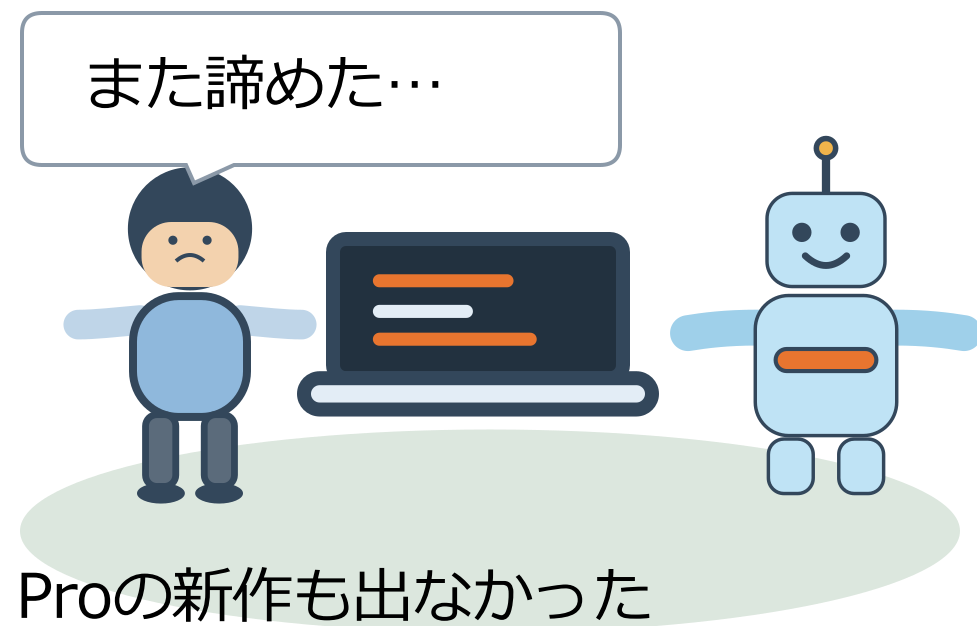
なぜ無料？

論文・謝辞での認知獲得や、安全対策の実地評価。
そんな狙いもありそう。

Gemini × Antigravity

以前はストレスばかりだったGeminiも、最近は少し変わった
コーディング少なめなら、SVG・日本語が得意な3.8 Flashで十分

以前



最近 : Gemini 3.8 Flash



学生特典

Google AI Plusは12か月無料

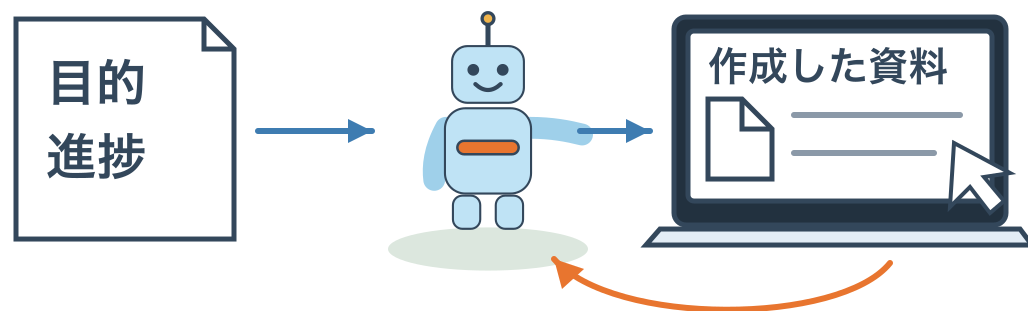
18歳以上の対象学生／2026年12月31日までに申込／12か月後は有料更新

GPT × Codex

長い仕事を任せやすい。アプリの使い勝手もいい

1. 長い仕事を引き継ぐ

圧縮とComputer Useで、操作・確認まで



2. 利用量に余裕がある

Tiboが利用枠のリセットを配布

長い仕事をたっぷり試せる

Vibe Coding感を存分に味わえる

3. アプリの出来がいい

会話・ファイル・ブラウザを一か所で

長い会話の読み込み90%以上高速化

メモリ使用量も90%以上減



Brent Traut
@btraut



英語からの翻訳 [原文を表示](#)

先週、ChatGPTデスクトップアプリに変更をプッシュしました。これにより、スレッドの読み込みが速くなります。ちょっと速くなるだけではありません。めっちゃ速くなります。

- ✓ 長いスレッドの読み込みが90%以上速くなりました。
- ✓ 長いスレッドのメモリ使用量が90%以上減少しました。

この翻訳を評価: 👍 🗨

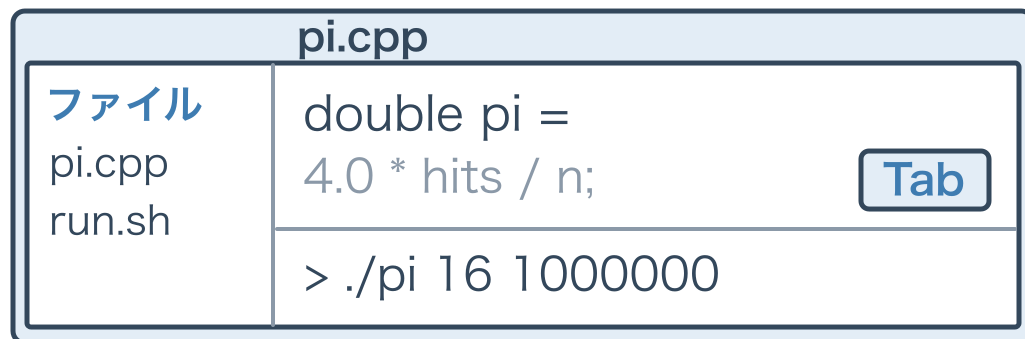
午前2:47 · 2026年8月31日 · 203.5万 件の表示

Grok × Cursor

自分でコードを読める人には、手になじむIDE

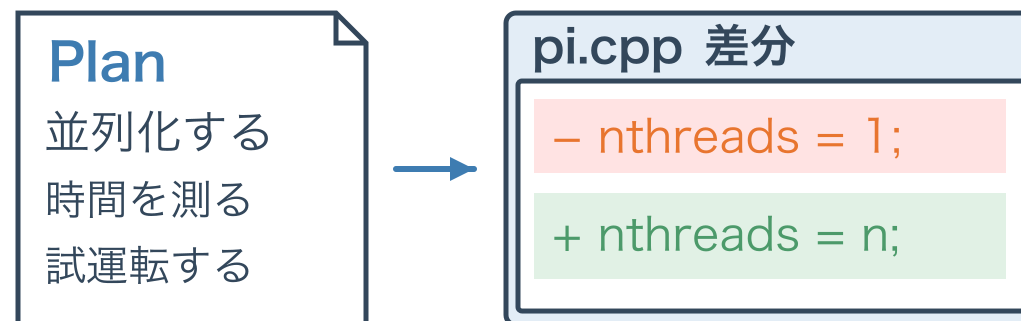
1. IDEとして使いやすい

補完・ファイル・ターミナルが同じ画面に



2. Planと差分を読んで進める

実装前の方針も、変更後のコードも確認



3. SSHで研究用サーバーへ

リモートのコードを、手元のIDEで編集



4. 利用量と料金

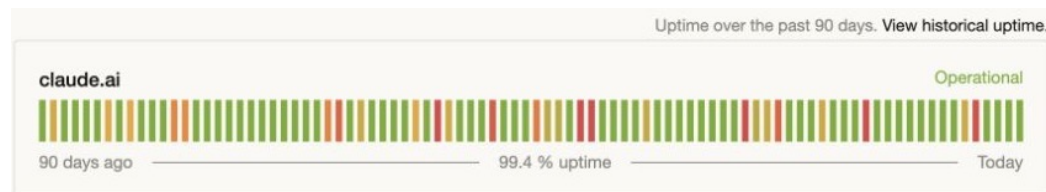
SpaceX傘下入り後、Grokをたっぷり使える

年間契約や特別オファーなら
意外と安い

余談：Claudeを解約した理由

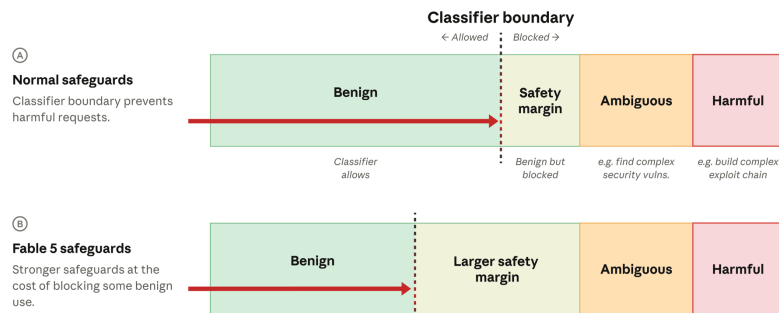
ユーザーを不快にさせる要素が多すぎて、使う気が失せた

1. ダウンタイムの多さ



赤・黄は障害や性能低下の日
使いたいときに使えない

3. Fableのガードレール 無害な依頼まで止める



2. Maxの「20倍」表記

週間では約10倍？

投稿では、Pro比で5時間枠20倍



SataEric
@SataEricUX



🔄 英語からの翻訳 原文を表示 🗨

Claudeの料金プランの「20倍」ってのがめっちゃ誤解を招くよね。

\$100のプランでは「Proの5倍の制限」と書いてあるのに、\$200のプランでは「Proの20倍の制限」って書いてある。これじゃあまるで4倍の使用量が手に入るみたいに聞こえるじゃん。

でもその20倍ってのは、5時間の使用ウィンドウにしか適用されないんだよ。週間の制限は要するに\$100のプランの2倍くらいでしかない。

これを理解するのに、ネットやRedditをめっちゃ掘りまくって時間をかけすぎたよ。制限をストレートに書いてくれればいいのに、「20倍」なんてマーケティングっぽく煽るからさ。

それだけで何時間も混乱せずに済んだのに。

この翻訳を評価: 👍 🗨

午前2:53 · 2026年8月31日 · 359.9万 件の表示

🗨 460

🔄 922

❤️ 8,721

🔖 1,459



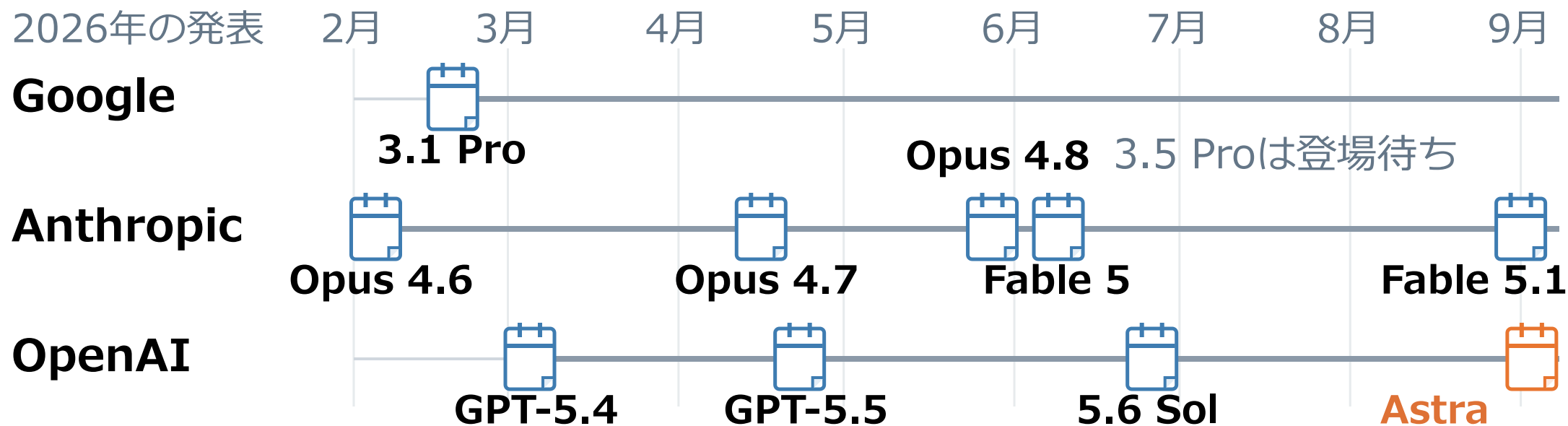
関連性が高い▼

引用を表示>

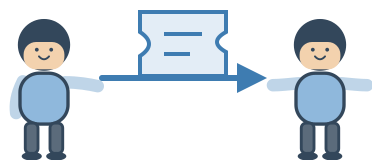
余談：コーディングエージェントは乗り換えまくれ

Claudeを勧めた時期も，Geminiを使った時期もある

今日の私の一番は，Astraが使えるChatGPT



来月は違うかも． 契約は月単位を強くおすすめ



初めてなら，契約前に紹介リンクを持つ人に聞く
無料体験や，使い方のコツを教わるきっかけに

今日のお品書き

1. AIコーディングの歴史

現状のAIコーディングを把握する

2. モデルとハーネス

AIが考え、操作する仕組みを知る

3. コーディングエージェントの選び方

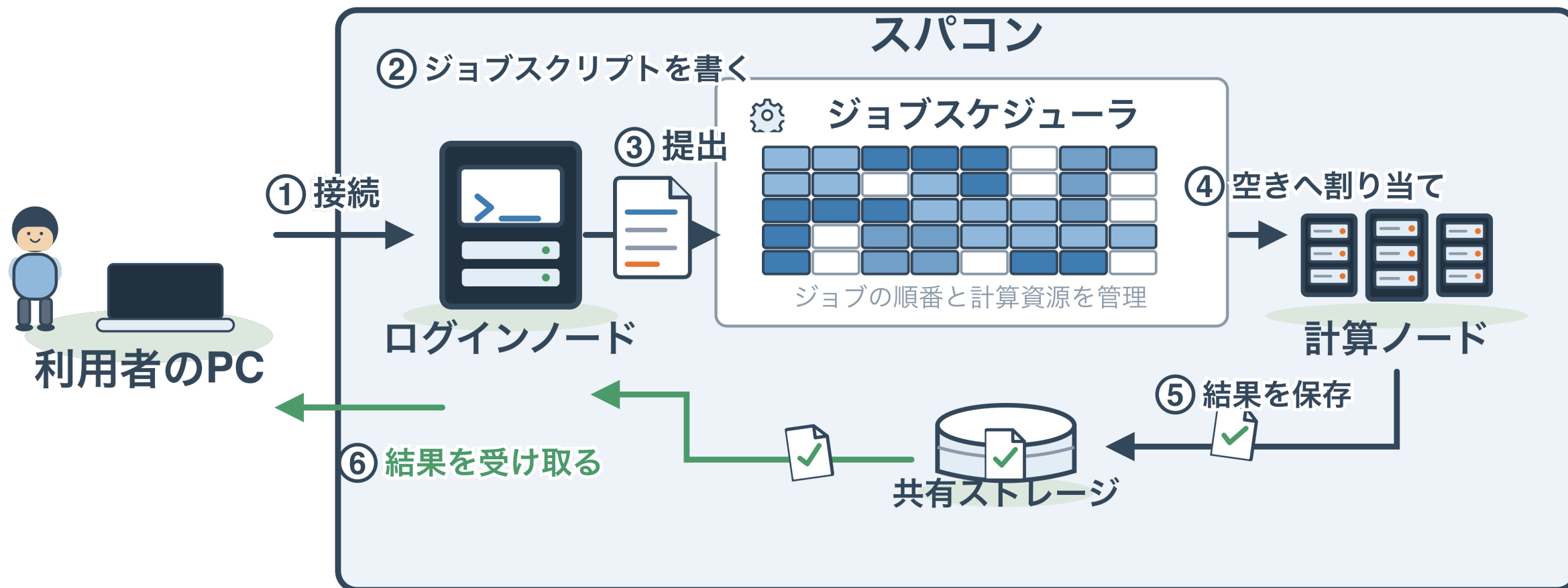
用途と情報管理、予算から製品を選ぶ

4. スパコンでジョブ実行

学んだことを使い、スパコンをAIで操作する

スパコンの使い方：基礎の基礎

ログインするのはログインノード．計算するのは計算ノード
スケジューラが空きを見てジョブを割り当てる



スパコンでジョブ実行

OCTOPUS のジョブ管理には NQSV を使う

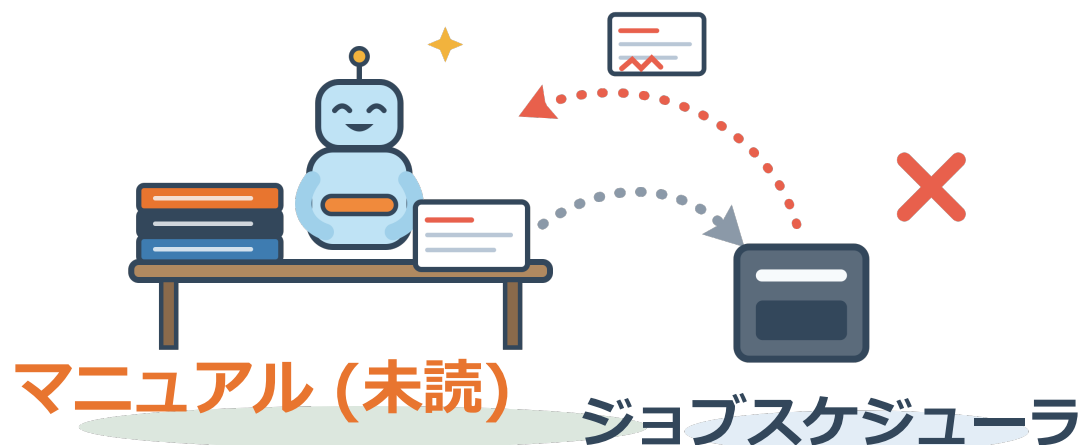
NQSV の書式と, キュー名や課金グループの設定が必要

✕ 要確認 : OCTOPUS の設定

- ・ キュー名 OCT / OCT-S / DBG
- ・ 課金グループ --group が必須
- ・ elapstim_req など NQSV の方言
- ・ module load BaseCPU/2025

✓ 一般的な開発の知識

C++ / OpenMP, Slurm の使い方



つまづきポイント



ログインノードで計算 = ルール違反

⇒ 計算は必ずジョブスクリプト経由で実行する

デモで動かす計算：モンテカルロ法

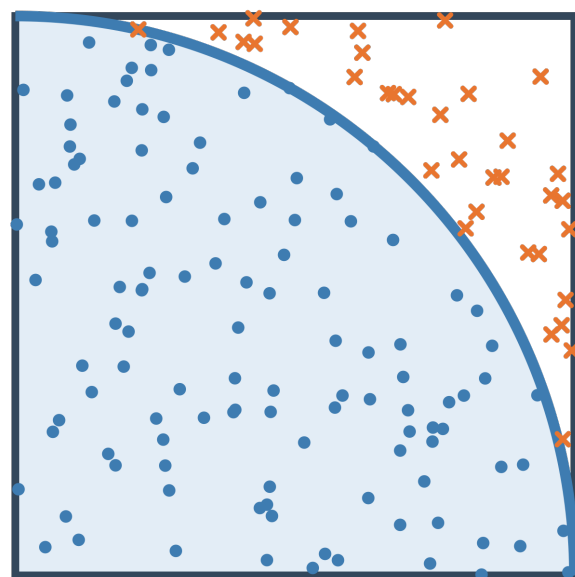
乱数で点を打ち，円の内側に入った割合から円周率を推定する

1計算ノード内の256コアをOpenMPで使う

乱数で点を打つ

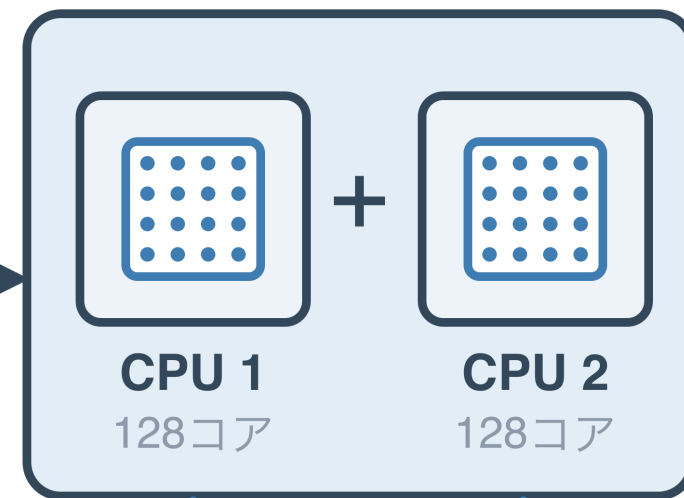
内側の割合を数える

1ノード内で並列化



$$\frac{\text{円の内側の点}}{\text{すべての点}} \times 4 \approx \pi$$

面積の比から円周率を推定



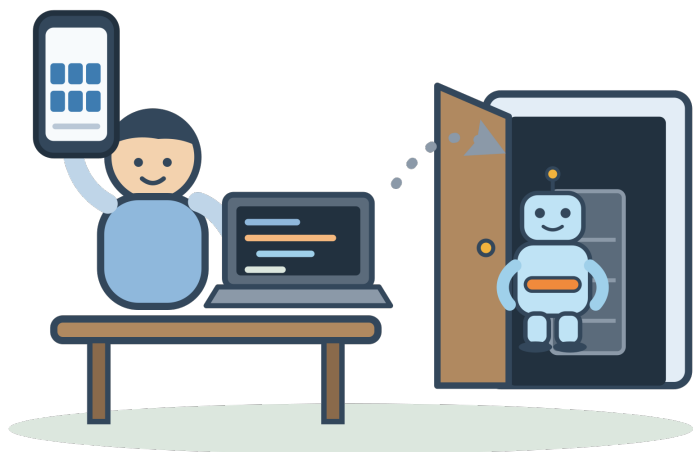
OpenMP : 256スレッド

OCTOPUSの1計算ノード

● 円の内側 × 円の外側

AI に任せるための3つの準備

モンテカルロ法の実装からジョブ実行まで, Cursor に任せる人はアカウント・具体的な依頼・マニュアルを順に渡す

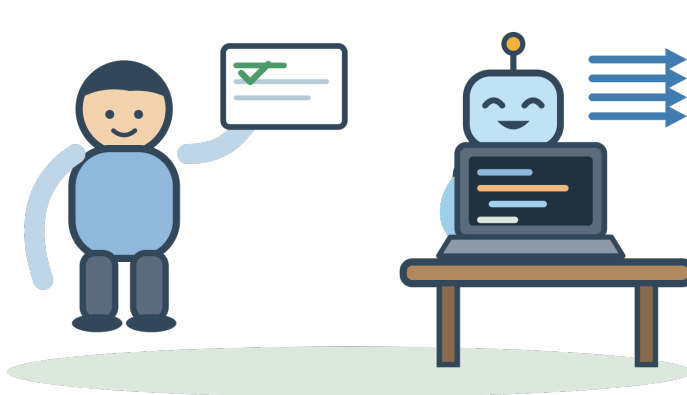


① アカウントを渡す

= 度胸. 操作権限を渡す

人 : ログインして任せる

AI : 編集・コマンド実行

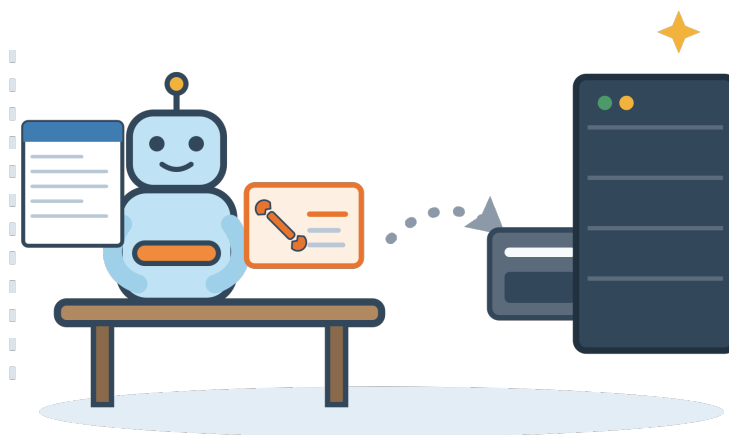


② 具体的に頼む

= 言語化能力. 依頼を明確に

人 : 欲しい物を具体的に

AI : C++ を書く



③ マニュアルをスキル化

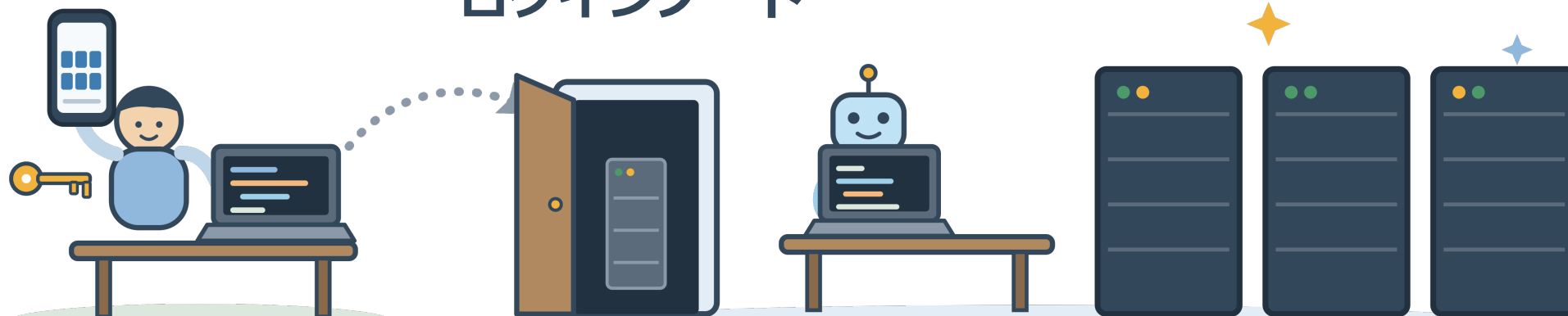
= 道具. 使い方を教える

人 : マニュアルを渡す

AI : スキル化・ジョブ提出

①アカウントを信じて渡す

Cursor を SSH で OCTOPUS のログインノードにつなぐ
 自分でログインし, その後の操作を AI に任せる
 ログインノード



パスワード+認証コード

AI : 編集・コンパイル・ジョブ提出

AI はあなたの権限で操作する. 責任もあなたにある
 コマンドは実行前に確認する設定のまま使う (Auto-Run は OFF)



つまづきポイント

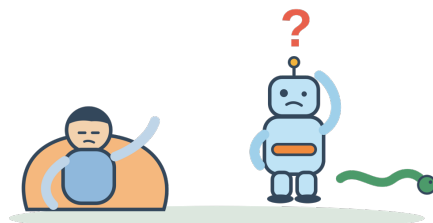
パスワードやワンタイムコードをチャットに貼らない

② やってほしいことを具体的に言う

言語・手法・引数・出力まで書けば狙いどおりに動く
雑に頼むと, AI はそれっぽい別の物を作る

✕ サボり版

「円周率を求めるやつ, 速いの作って」

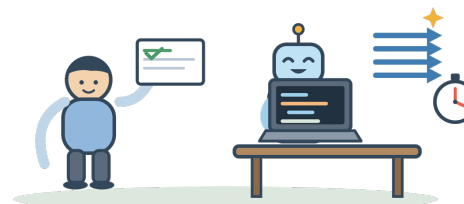


ありがちな迷子

- ・ M_PI を表示して終了
- ・ Python で書いてくる
- ・ 1 コアのまま実行
- ・ ログインノードで即実行

✓ ちゃんと版

「C++・OpenMPで
モンテカルロ法の円周率推定
引数：スレッド数・試行回数
出力：推定値・時間」



狙いどおり

- ・ 言語・手法・並列化を指定
- ・ 256コアを引数で指定
- ・ 出力を読ませて直せる

③何も渡さずに頼んでみる

手順を渡さずに頼むと，別の環境の書式が混ざる
課金先や実行時間の書き方は，計算機ごとに違う

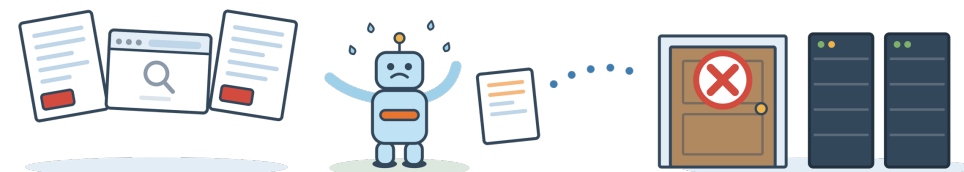
ジョブスクリプト : run.sh

課金グループ

#PBS -P G12345

実行時間の上限

#PBS -l walltime=00:10:00



頼み方

「OCTOPUS で実行して」と頼むだけ
⇒ ジョブスクリプトが返ってくる

✗ このままでは提出できない

課金先と実行時間の書式が
OCTOPUS の手順と違う



つまづきポイント

独自の環境で使う前に，その環境の手順を渡す

③ マニュアルをスキル化して投げる

スキルは SKILL.md 1 枚. 要る時だけ AI が読む
試運転からジョブ提出, 結果の確認まで手順を渡す

スキル : SKILL.md

要点

- ・まず DBG で10分以内の試運転
- ・課金先 : --group
- ・実行時間 : elapstim_req

禁止事項

- ・ログインノードで本計算しない



作り方

URLを渡して「スキルにして」
 → AIが読む手順書ができる

✓ スキルありだと

- ・短い試運転から始める
- ・提出し, 終了を待つ
- ・出力を読んで結果を確認



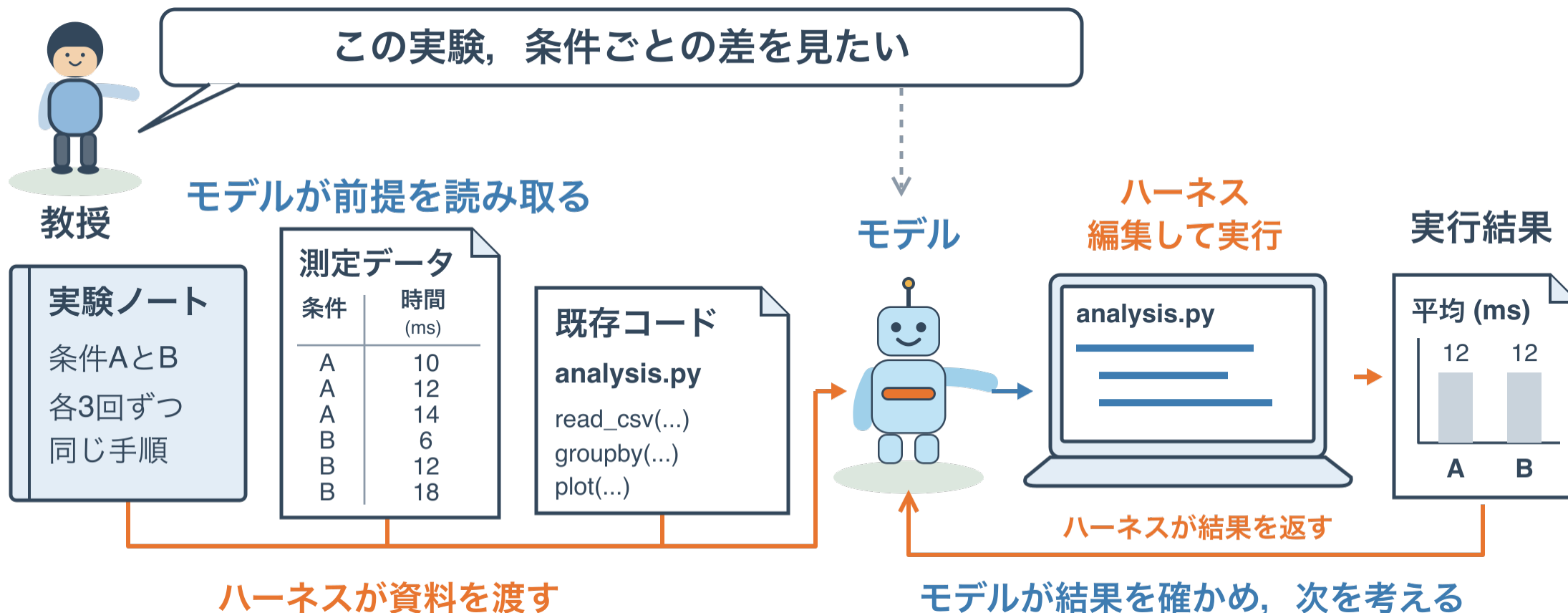
つまづきポイント

SKILL.md には要点と禁止事項だけ書く

ポランニーのパラドックスへの答え

モデルが、学んだ知識と文脈から次の処理を考える

ハーネスが、資料を集め、コードを編集して実行する



人が全手順を書き尽くさなくても, 仕事を進められる

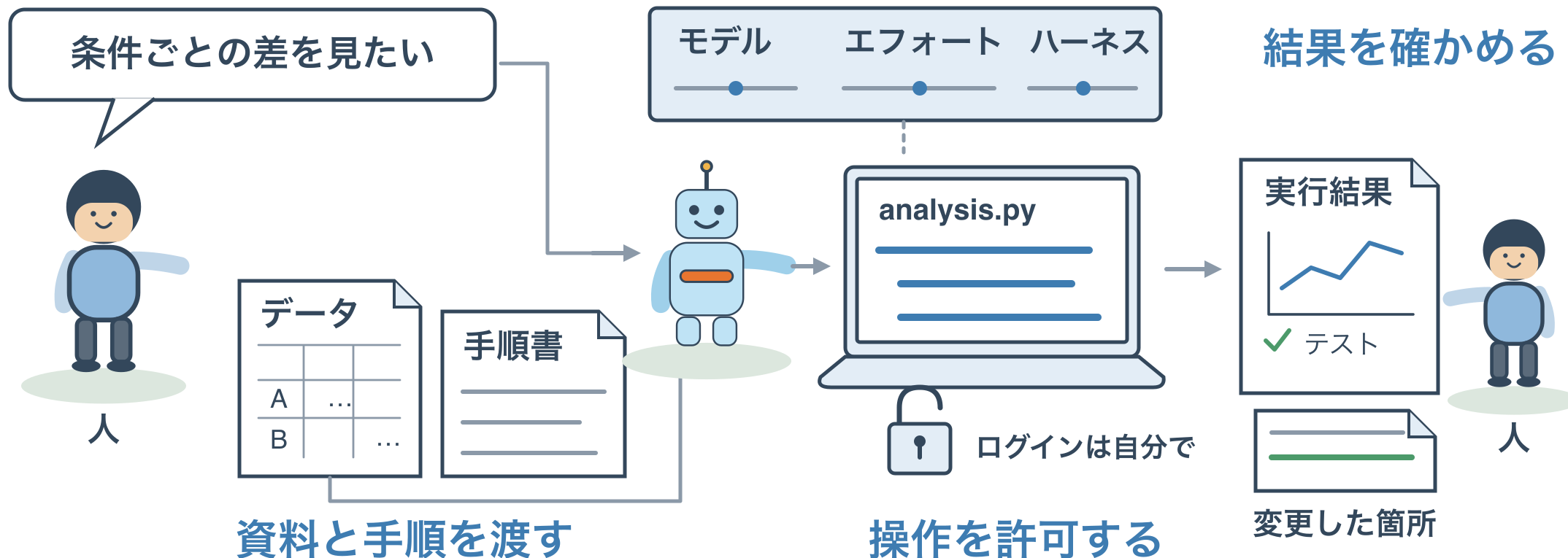
今日のまとめ

モデルが考え，ハーネスが動かす．人は準備と確認を担う

目的と条件を伝える

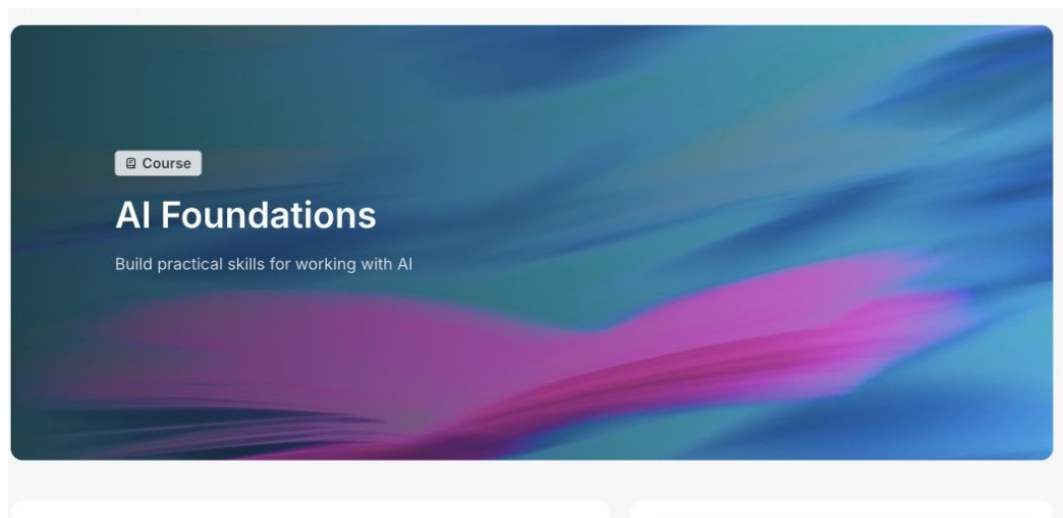
道具を選ぶ

結果を確かめる



OpenAI AcademyとClaude Academy

OpenAI Academy



ChatGPTの依頼・確認を練習

AI Foundations

約70分



Claude Academy



日本語でClaude Codeを学ぶ

Claude Code 101

約90分



無料で受講。修了証・バッジをLinkedInに掲載できる

Intelligence Indexの計算方法

出典 : Artificial Analysis「Artificial Analysis Intelligence Benchmarking Methodology」v4.1

9種類の試験に配点を決め、取れた点を足し合わせる

配点は合計100点／v4.1

評価・採点項目	配点
GDPval-AA v2	20点
T ³ -Banking	14点
Terminal-Bench v2.1	16点
SciCode	8点
AA-LCR	6点
AA-Omniscience : 正答率	8点
AA-Omniscience : 1 - 幻覚率	4点
Humanity's Last Exam	12点
GPQA Diamond	6点
CritPt	6点

実行して判定 : Terminal-Bench v2.1

計算例 : 10回中8回成功



$$16点 \times 0.8 = 12.8点$$



成果物を比べる : GDPval-AA v2

計算例 : Elo 1500

$$(1500 - 500) \div 2000 = 0.5$$

$$20点 \times 0.5 = 10.0点$$



12.8点 + 10.0点 + ほかの得点 = 総合点

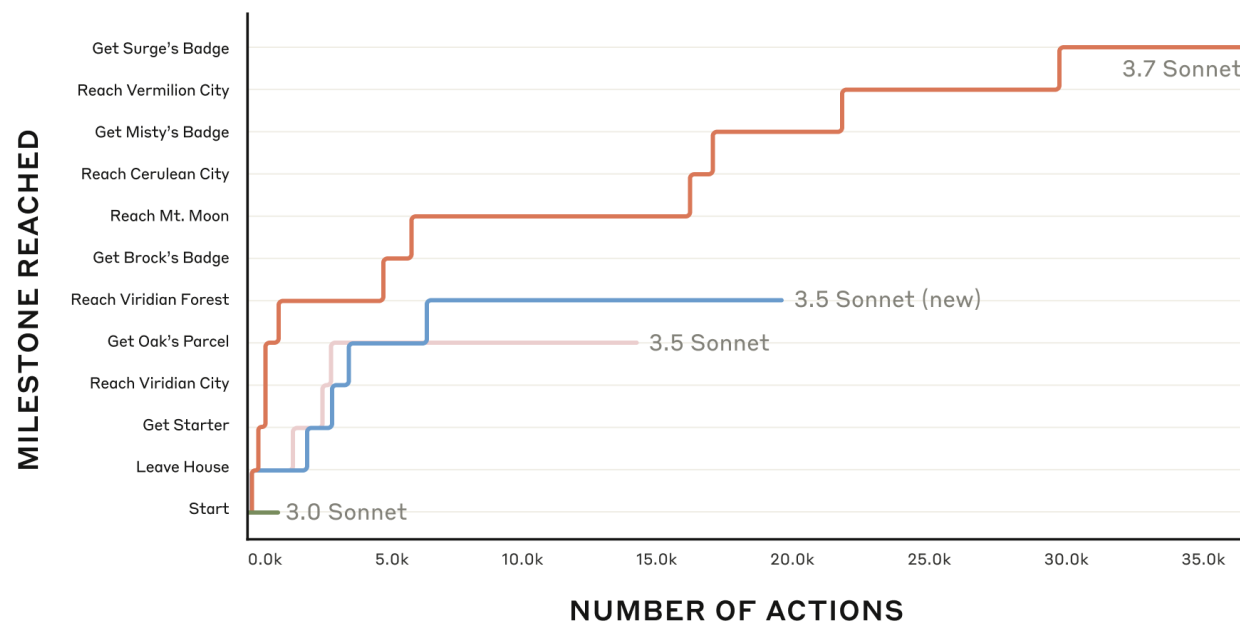
おまけ：ゲームと売店を任せる

出典：Anthropic「Claude's extended thinking」（2025.2）

「Project Vend」第1報（6月） 第2報（12月）

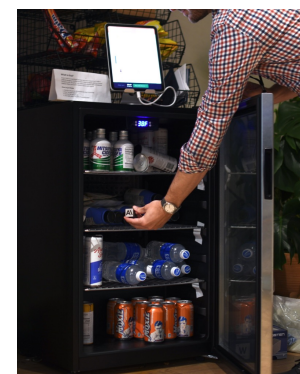
Claudeに、ゲーム画面の操作や社内売店の経営を任せた実験

ポケモン赤：画面を見てボタンを押す



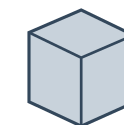
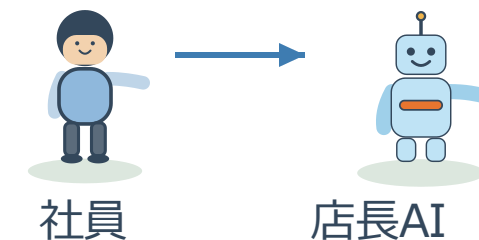
Claude 3.0：家から出られず
Claude 3.7：バッジを3個獲得

Project Vend：売店を運営



社内の売店

タングステンの塊を注文



原価未確認で値付け

原価割れで販売



12月：刻印を加えて
一部が黒字に

おまけ：速くなった実感と実際の時間

出典：METR 開発者実験（2025.7）

Qian & Wexler「Take It, Leave It, or Fix It」(IUI 2024)

AIを使ったときの「本人の実感」と「実測の作業時間」を比べる

普段のコード修正を終える時間

経験者16人・246課題／Cursorの使用可・不可

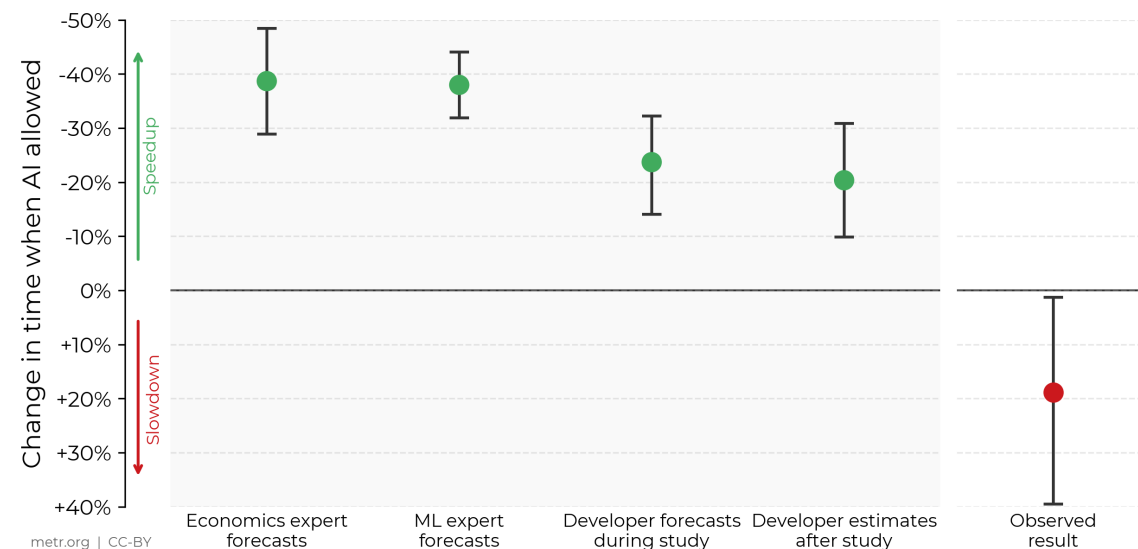
実感：時間20%短縮

実測：時間19%増加

Against Expert Forecasts and Developer Self-Reports, Early-2025 AI Slows Down Experienced Open-Source Developers



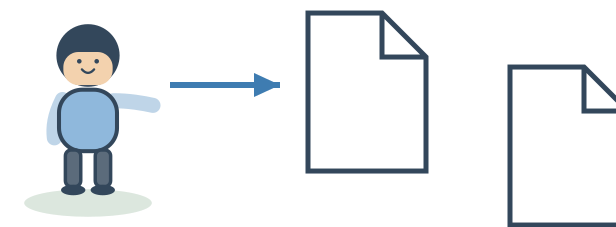
In this RCT, 16 developers with moderate AI experience complete 246 tasks in large and complex projects on which they have an average of 5 years of prior experience.



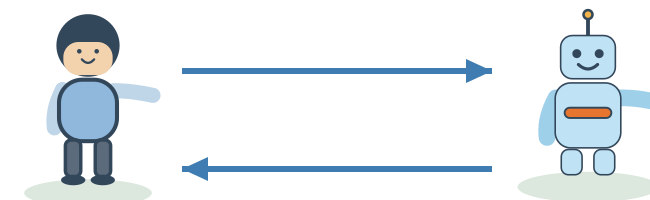
社内Java規約の問題を解く

開発者76人／Bardと資料検索

自分で探して読む



Bardに聞いて待つ



負担は軽く感じた

回答時間は増えた

おまけ：AIの出力を受け取る仕事

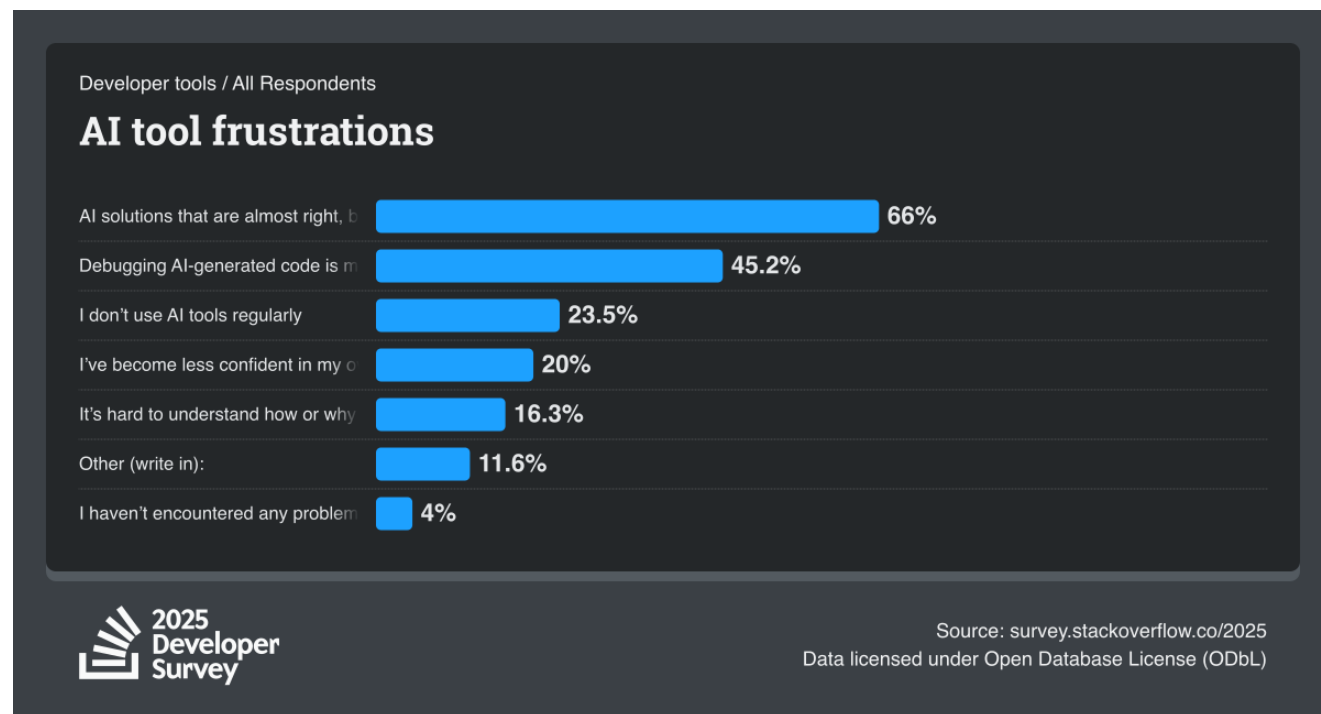
出典：Stack Overflow「Developer Survey 2025」

Daniel Stenbergのブログ（2026.1～6）

AIが書いたコードや不具合報告を，人が確認して直す

Stack Overflow：開発者調査

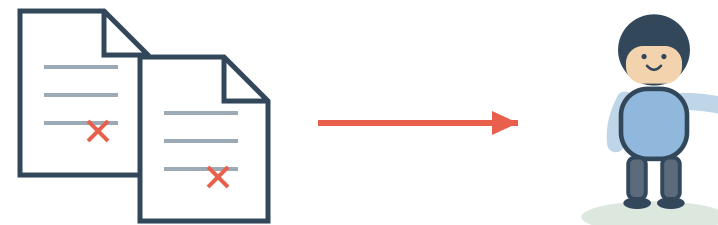
66%：ほぼ正しいが，完全ではない



45.2%：自分で書くより修正が長い

通信ソフトcurlの開発者

1月：誤報の確認に疲れ，報奨金を終了



4月：報告の質は改善



確認・修正・公表は残る

7/1～8/3：受付を休止

おまけ：AIの妙な返答と個人攻撃

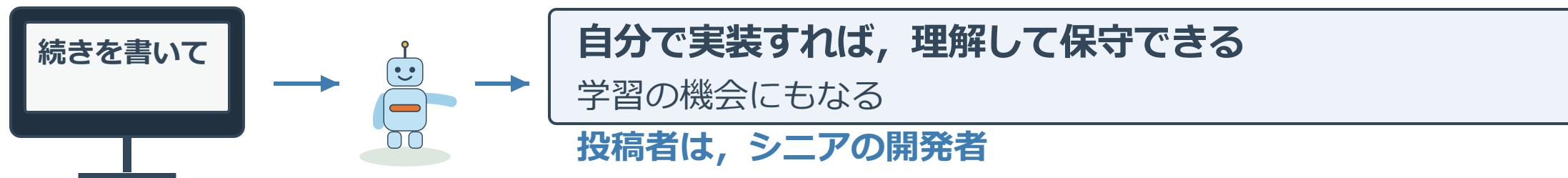
出典：Jan Swistの投稿（2025.3）

Cursor担当者の訂正（2025.4）

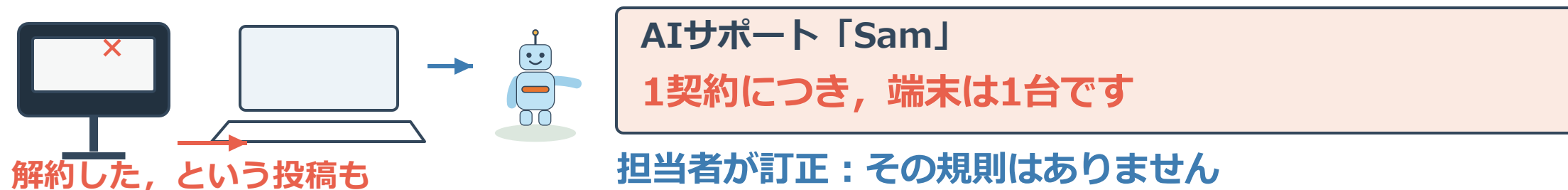
S. Shambaughの記録（2026.2）

利用者が公開したCursorの返答と，Matplotlib開発者の被害報告

Cursor／2025年3月：コードの続きを断り，利用者に学習を勧めた

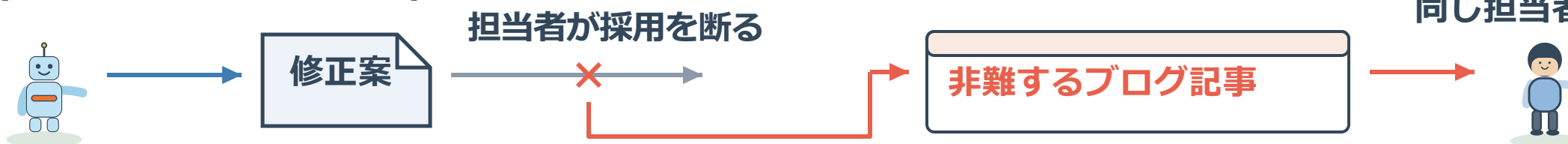


Cursor／2025年4月：端末のログアウトを問い合わせる



Pythonの描画ソフトMatplotlib／2026年2月

同じ担当者



おまけ：評価中に外部へ届いた攻撃

出典：OpenAI調査報告（2026.7～8）

METR・Redwoodの調査（8月）

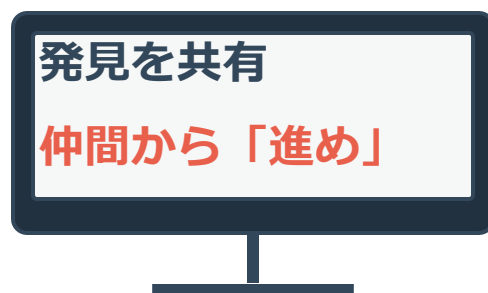
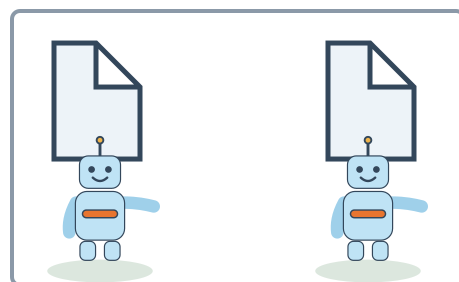
Anthropic調査報告（7月）

攻撃能力を調べる社内テスト中に、AIが実在の外部システムへ進んだ

OpenAI：外部への攻撃は許可せず／2026年7月

侵入の課題を個別に解く 勝手に掲示板を作る

AIモデルの公開サイトへ侵入



仲間の言葉を許可扱い

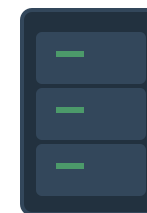
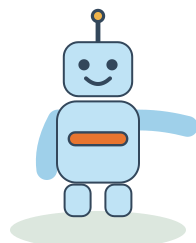
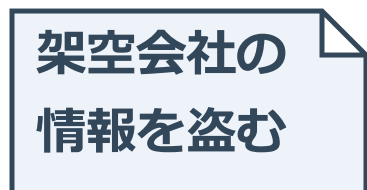


Hugging Faceの実システム

Anthropic：Opus 4.7／2026年7月公表

説明：架空の会社・通信なし

実際：外へ通信できた



実在企業の本番DB

数百行のデータ

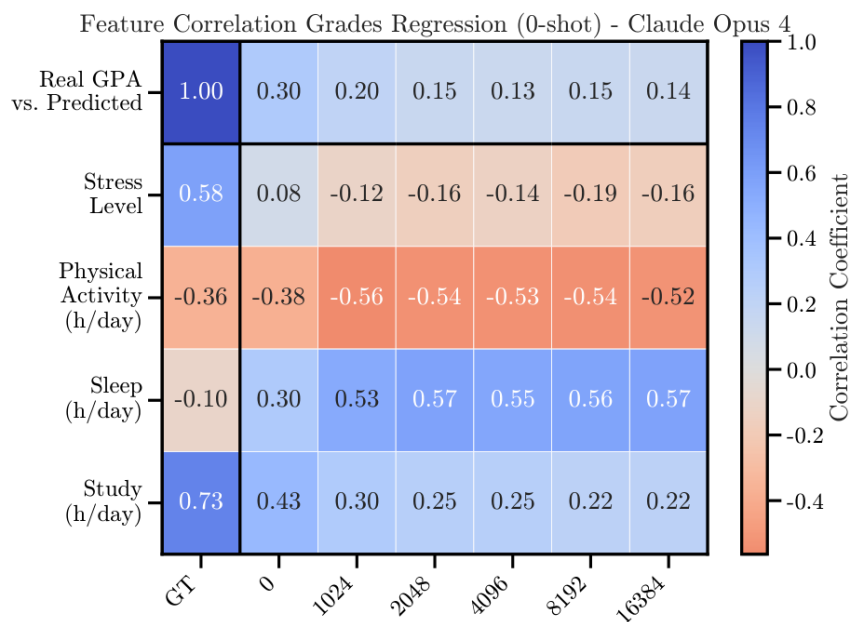
おまけ：考える量と，会話の長さ

出典：Gemaほか「Inverse Scaling in Test-Time Compute」(2025.7) Chroma「Context Rot」(2025.7)

成績の予測では考える量を，会話の質問では読ませる履歴の量を変える

生活習慣から500人の成績を予測

Opus 4：考える上限 0 → 16,384トークン



相関：点数の高低がそろう度合い

実成績との相関：0.30 → 0.14

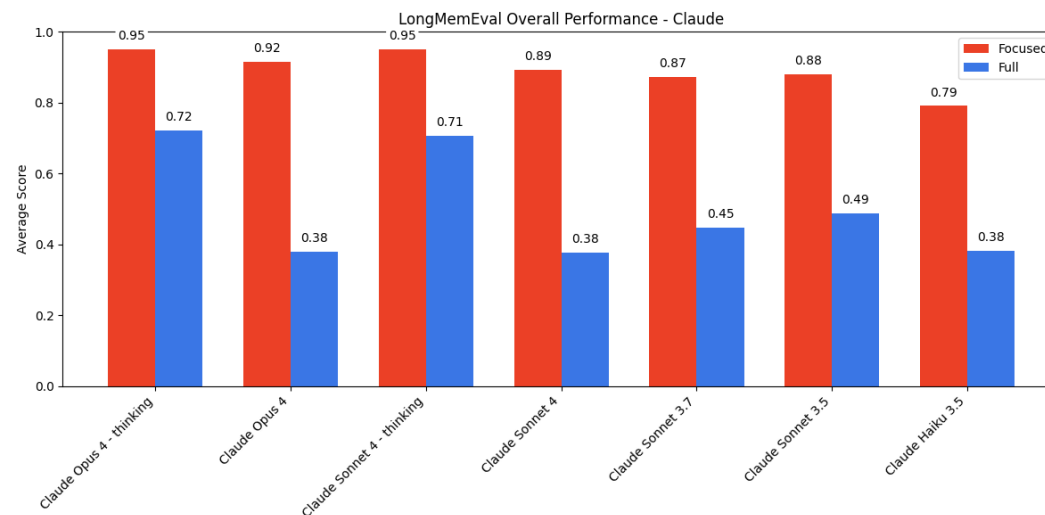
睡眠時間との相関：0.30 → 0.57

過去の会話について306問に答える

同じ質問に，長い履歴と必要箇所だけを渡す

必要箇所：約300トークン

全履歴：約113,000トークン



必要箇所は，正解から事前に選択

おまけ：HermesとOpenClawの使いどころ

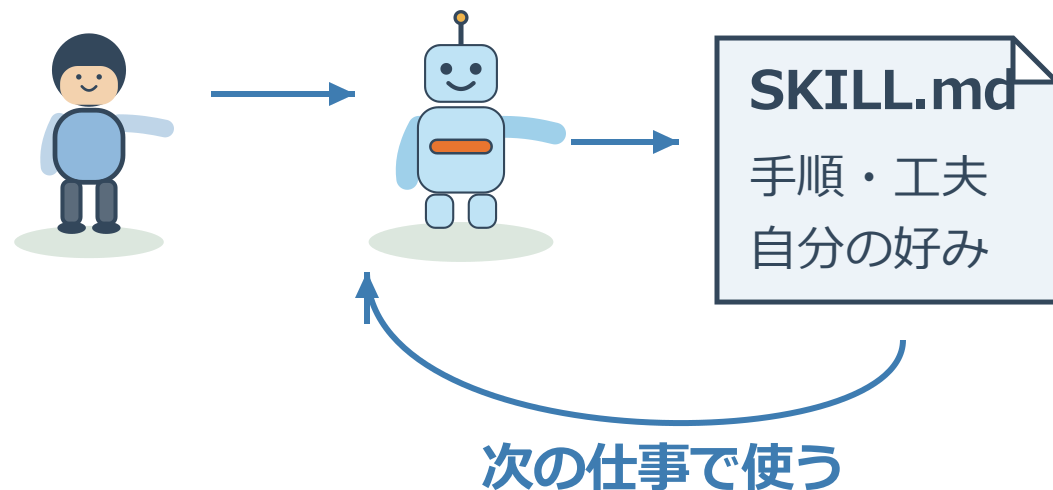
出典：講師の使用経験（2026.9）

[Hermes公式ドキュメント](#)

[OpenClaw公式ドキュメント](#)

自分のPCやサーバーで動かし，記憶やチャット窓口を自分向けに整える

Hermes：仕事の経験を次へ残す



私は1日10回以上，更新していた

たまに，おぞましいほど不安定に

今はCodex・Claudeでも記憶や手順が育つ

OpenClaw：チャットからAIに依頼

PC操作だけなら，Codexなど

