

TRAINING

Cursor基礎講座

AI駆動開発の基本操作とモデルの使い分け

株式会社アイネス様 | 2026年10月23日（金） オンライン



本資料の二次転載禁止について

本資料の著作権は株式会社ギブリーに帰属します。受講者の学習を目的として提供するものです。

本資料の全部または一部を、当社の許諾なく複製、転載、配布、公衆送信することを禁じます。社内での共有をご希望の場合は、事務局を通じてご相談ください。

講師紹介



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

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。その後北海道のドラッグストア『サッポロドラッグストアー』からスクールの買収を受け、教育を専門とするグループ会社『株式会社シーラクス』にて技術開発責任者として運営。スクール設計・運営、20以上の自治体連携（イベントや出張教室運営）、大人向けプログラミングスクールメンターなどさまざまな『次世代IT教育』に取り組む。Dify公式資料の日本語翻訳や、SecondMeアンバサダーなど生成AI最新トレンドに注力。

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

メッセージ

本日は全力でサポートしていきます！
なにかご不明点などありましたら、講師陣になんでも気軽にご質問ください。

#ClaudeCode #Codex #Cursor #Antigravity
#全国統一生成AI活用技能試験S1
#生成AIパスポート #G検定

本日の流れ

ハンズオンが研修時間の7割

セッション	内容	形式	時間
S01 オリエンテーション	進め方と質問のしかた、 事前セットアップと利用状況の確認	講義	[10min]
S02 生成AIの動向と ガバナンス	コーディングエージェントの動向と事例、 入力情報の匿名化とシャドーAI、利用上限	講義	[45min]
S03 Cursorの基本操作	画面構成、在庫管理コードへの機能追加、 Cursor Rules	ハンズオン	[70min]
休憩			[10min]
S04 モデルの使い分けと レガシーコード改修	モデルの使い分けと利用状況、テストで守りながらの 書き直し、SQLとTerraformのデモ	ハンズオン	[95min]
S05 質疑応答と閉会	利用状況の再確認、質疑応答	講義	[10min]

本日のゴール

研修の終わりに、次の**3つ**ができる状態を目指します

1

Cursorの4つの操作

Tab補完、インライン編集、チャット、エージェントを作業に合わせて使い分けます。

2

モデルの選び方と利用状況

作業の重さでモデルを選び、使った量を自分で画面を開いて確かめます。

3

入力してよい情報の線引き

匿名化が要る情報と、入力してはいけない情報を見分けます。

持ち帰るもの

改修した在庫管理システムのコードと、自分で書いたRulesのファイルと、当日の利用状況の記録

進め方と質問のしかた

止まったら、画面の文字をそのまま講師に送ってください

1 講師が画面共有で操作を見せます



2 受講者が同じ操作を自分のCursorで行います



3 手順と完了の目安はハンズオンガイドにあります

チャットで送る質問の例

「演習1 STEP 2 :
Ctrl+Kを押しても
入力欄が出ません」

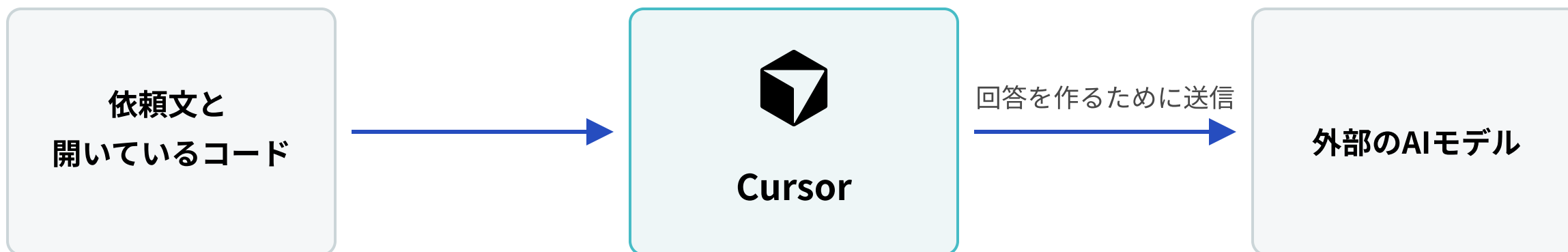
場所と、押したキーと、起きたことを書きます。
画面の文字は、そのまま貼り付けます。

ハンズオンガイドのURL

<https://1023cursor.give-app.net/handson>

演習で使うデータ

演習で使うのは配布フォルダの架空のデータだけ



業務のコードと顧客の情報は入力しないでください

ノースパイン商事株式会社は架空の企業です

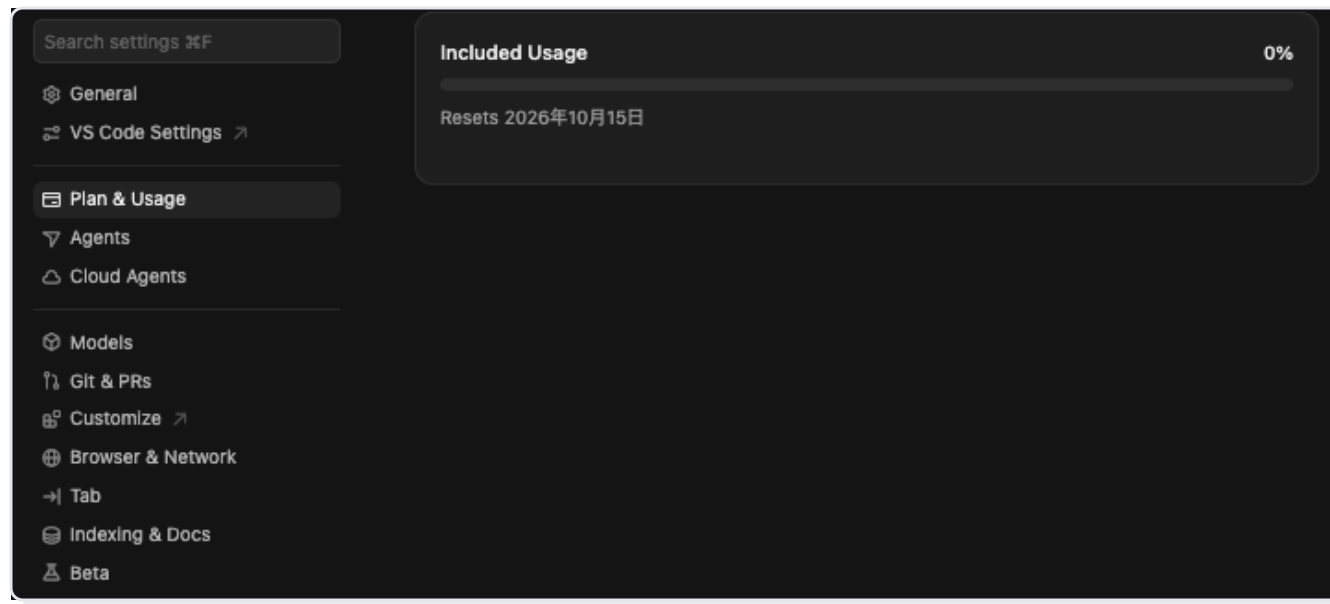
7つの項目がそろっているかを確認してください

- ☐ 会社のメールアドレスでCursorのアカウントを作った
- ☐ 研修事務局が案内した有料プランを用意した
- ☐ Azure Virtual DesktopにCursorを入れ、起動できた
- ☐ Cursorの左下に会社のメールアドレスが表示された
- ☐ エディタの画面で質問を送り、返事が表示された
- ☐ Plan & Usageで利用状況を確認された
- ☐ 配布フォルダを展開し、Cursorでinventory-appを開けた

そろっていない項目は、チャットで講師にお知らせください

Included Usageの割合をメモしてください

- 1 Ctrl+Shift+Jで
Cursor Settingsを開きます
- 2 Plan & Usageを押します
- 3 Included Usageの割合を
メモします



画面: Cursor SettingsのPlan & Usage
(2026年10月5日、無料プランで撮影)

研修の最後にもう一度見て、4時間で使った割合を比べます

02

生成AIの動向とガバナンス

コーディングエージェントの動向と事例、入力情報の扱い、利用上限
[45min]

コーディングAIの3つの段階

補完からファイルをまたいで作業するエージェントへ

補完

書きかけのコードの次の数行を、AIが提案します。

チャット

開発者がコードについて質問し、答えを読んで自分で貼ります。

エージェント

AIがファイルを読み、複数のファイルを変え、テストを流して確かめます。

Cursorで使う機能

Tab補完

インライン編集とチャット

エージェント

GitHub Copilotのcoding agentは、2025年9月25日に一般提供になりました。

出典: github.blog/changelog/2025-09-25-copilot-coding-agent-is-now-generally-available

主なコーディングエージェント

主なコーディングエージェントは**使う場所**で分かれる

				
製品	Cursor	Claude Code	Codex	GitHub Copilot
開発元	Anysphere	Anthropic	OpenAI	GitHub
使う場所	2026年8月からSpaceX傘下			
エディタ	エディタ	エディタの拡張		エディタ
ターミナル		ターミナル	ターミナル	
クラウド			クラウド	coding agent

※ 各ロゴは各社の商標です。

開発者のAI利用

使う人は**84%**に対して精度を信頼する人は**33%**

84%

AIツールを使用中か使用予定の開発者

33%

AIの出力の精度を信頼している開発者

不満の1位 「ほぼ正しいが少し違う」 **66%**

出典: Stack Overflow Developer Survey 2025 (survey.stackoverflow.co/2025/ai、2026年10月5日確認)

Cursorの成り立ち

CursorはVS Codeを元にしたAIエディタ

エディタ	VS Codeを元にしたエディタです。 VS Codeの拡張機能・テーマ・設定・キー割り当てを取り込みます。
モデル	独自のComposer 2.5とGrokに加え、他社のモデルも選べます。
会社	開発元のAnysphereは、2026年8月にSpaceXによる約600億ドルの買収が完了し、SpaceX AI部門に入りました。
補足	サインインの通信先は、accounts.spacex.aiやaccounts.x.aiへ移行中です。

出典: cursor.com/docs/configuration/migrations/vscode (2026年10月5日確認)
cursor.com/help/troubleshooting/sign-in-domains (2026年10月5日確認)
GIGAZINE (2026年8月16日) gigazine.net/gsc_news/en/20260816-spacex-officially-closes-cursor-acquisition

Cursorの直近の更新

半年でモデルの自動選択と並行作業の機能が増えた



出典: cursor.com/changelog (2026年10月5日確認)

国内の開発現場の事例

期間とレビュー回数の短縮を公表する国内企業

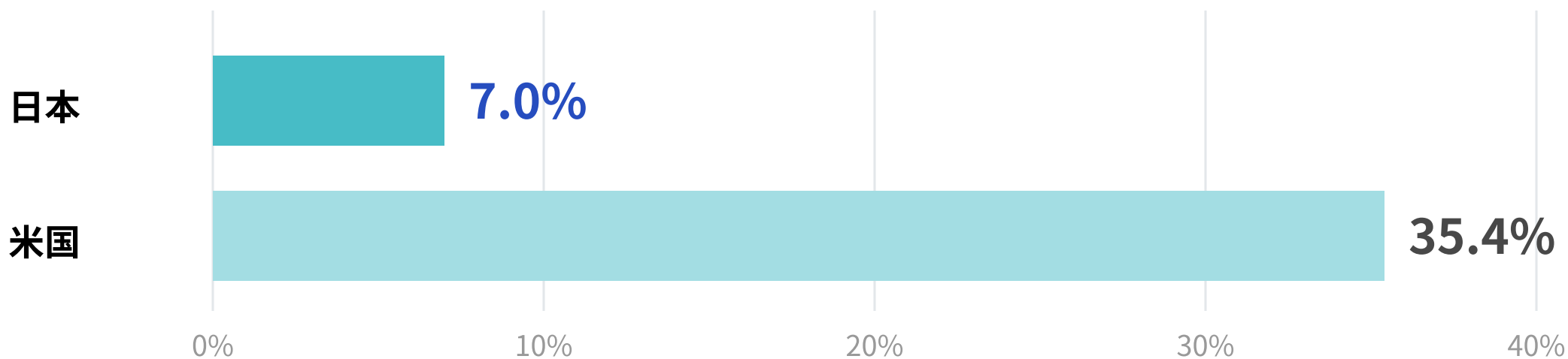
会社	取り組み	公表された数字
楽天	Claude Codeを使った開発	機能を出すまでの期間を 24営業日から5日 に短縮
freee	Claude Codeを中心にしたAI駆動開発	月に 600～700名 が利用し、利用の 約80% がClaude Code
DeNA	AIを前提にした開発の進め方	1件のプルリクエストあたりのレビュー回数が 7.2回から2.7回 に減少
サイバーエージェント	AIコーディングツールの全社導入	約2,000人 が利用

各社が公表した数字です。使っている道具は、Cursorとは限りません。

出典: claude.com/customers/rakuten、developers.freee.co.jp/entry/ai-driven-development-2025-report、engineering.dena.com/blog/2026/01/ai-driven-develop、developers.cyberagent.co.jp/blog/archives/64334

国内の利用経験

コード生成AIの利用経験は日本が**7.0%**で米国が**35.4%**

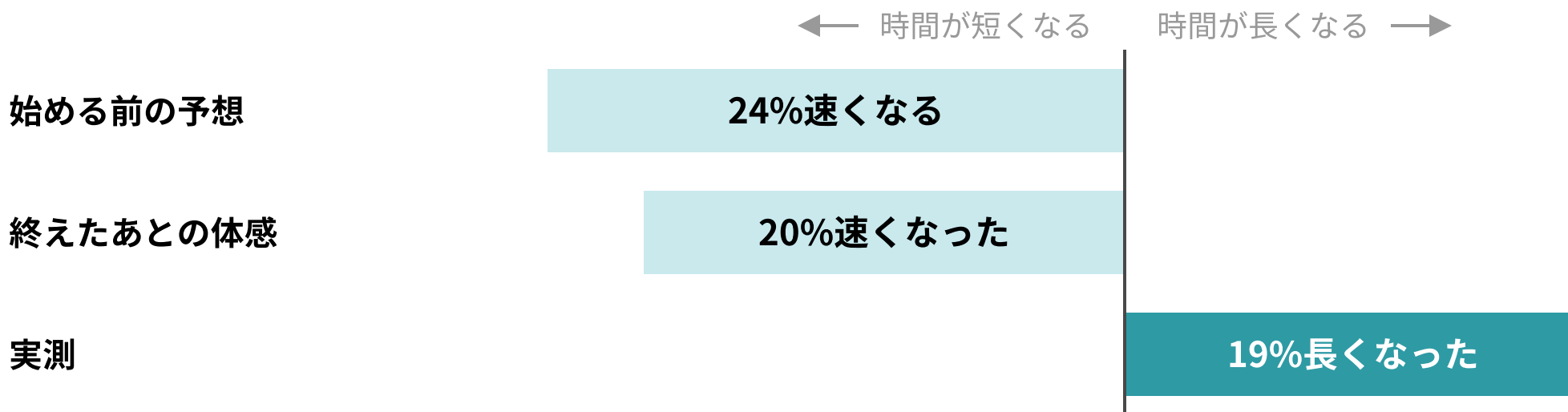


国内では、コード生成AIを業務で使い始める人がまだ少数の段階です。

出典: 総務省『令和8年版 情報通信白書』 soumu.go.jp/johotsusintokei/whitepaper/ja/r08/pdf/n1110000.pdf

生産性の数字の読み方

速くなったという体感と実測の時間は一致しない



経験のある開発者16人が、慣れたリポジトリの246件の課題で、2025年2～6月に試した結果です。
研修でも、利用状況と時間を自分で記録して確かめます。

出典: METR (2025年7月10日) metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study

2025年のMETRの実験で、AIを使った経験者の作業時間はとなりましたか。

A 約半分になった

B 約2割短くなった

C 約2割長くなった

D 変わらなかった

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。

正解はC。体感では速くなったと答えたのに約2割長くなった

C

約2割長くなった

16人の開発者が246件の課題で試し、AIを使うと作業時間が19%長くなりました。

A

約半分になった

半分になったという結果ではありません。

B

約2割短くなった

約2割短くなったのは開発者の体感で、実測とは逆です。

D

変わらなかった

差は19%あり、変わらなかったわけではありません。

出典: METR 「Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity」
metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study (2025年7月10日)

入力情報の3つの区分

入力する前に、情報を3つの区分に分けて考えます

区分	例	扱い
そのまま入力してよい	公開されている情報、研修の架空のデータ	そのまま入力する
匿名化してから入力する	顧客名や担当者名を含むログ、業務データの集計表	置き換えや丸めのあとで入力する
入力しない	パスワード、接続文字列、APIキー、個人の評価	匿名化しても入力しない

業務では、社内の規程に従って線引きしてください。この表は研修での目安です。

匿名化の手段

匿名化は置き換え・丸め・削除・ダミー化の4つ

手段	例	向いている情報
置き換え	取引先名を A社 に、担当者名を 担当者1 に置き換える	社名、人名
丸め	3,214,500円を 約300万円 に丸める	金額、日付
削除	メールアドレス、電話番号、図面番号を消す	連絡先、識別番号
ダミー化	配布フォルダと同じ方法で、同じ形の架空のデータを作る	テストのデータ

匿名化の例

顧客名と担当者は置き換え・電話番号は削除・金額は丸め

匿名化の前

[ERROR] 注文の登録に失敗
customer=ノースパイン商事
person=北野 太郎
tel=011-000-0000
amount=3214500



匿名化の後

[ERROR] 注文の登録に失敗
customer=**A社** 置き換え
person=**担当者1** 置き換え
tel= **(削除)** 削除
amount=**約300万円** 丸め

ノースパイン商事と北野 太郎は、どちらも架空の名前です。

入力が原因の情報漏えい

入力が原因で生成AIの利用を禁止した事例

Samsung

2023年春

Samsungのエンジニアが、ソースコード・テスト手順・会議の文字起こしの3件を、社外のチャットAIに入力しました。

2023年5月

社員による生成AIツールの利用を禁止しました。

2023年

3月

4月

5月

6月

国内

2023年6月2日

個人情報保護委員会が、生成AIサービスの利用について注意喚起しました。

入力した月は報道で3月と4月に分かれるため、春と書いています。

出典: TechCrunch (2023年5月2日) techcrunch.com/2023/05/02/samsung-bans-use-of-generative-ai-tools

個人情報保護委員会 (2023年6月2日) ppc.go.jp/files/pdf/230602_kouhou_houdou.pdf

シャドーAIの広がり

許可されていないAIを業務で使う人は8割超

許可されていないAIを業務で使う人

80%超

UpGuardが2025年に1,500人を調べた結果です。

IBMの報告（2025年7月30日）

侵害を受けた組織の**5社に1社**が、
原因にシャドーAIを挙げました。

シャドーAIの利用が多い組織は、侵害の
費用が平均**67万ドル**高い結果でした。

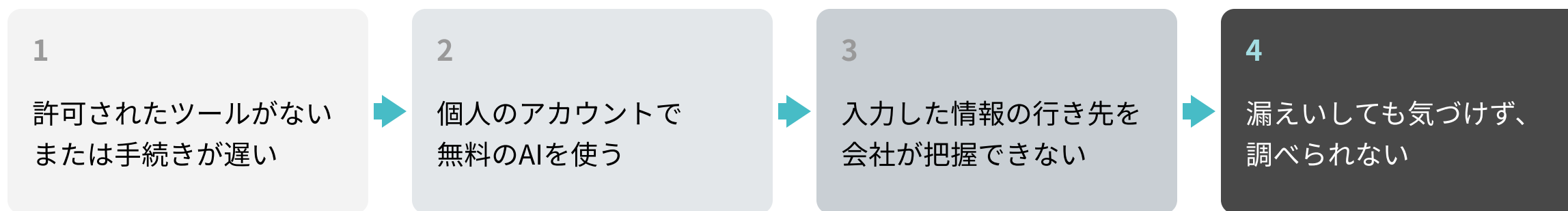
シャドーAIは、会社が許可していないAIを社員が業務で使うことです。

出典: Cybersecurity Dive（UpGuardの調査）cybersecuritydive.com/news/shadow-ai-employee-trust-upguard/805280

IBM（2025年7月30日）newsroom.ibm.com/2025-07-30-ibm-report-13-of-organizations-reported-breaches

シャドーAIが引き起こすこと

会社が把握できない入力漏えいしても調べられない



3位

IPAは2026年1月29日に、情報セキュリティ10大脅威2026を公表しました。
AIの利用に関するリスクが、組織向けの3位に初めて入っています。

出典: ipa.go.jp/pressrelease/2025/press20260129.html (2026年10月5日確認)

Cursorのプライバシーモード

プライバシーモードならコードは学習に使われない

プラン

オンにする人と場所

個人プラン

利用者



Cursor Settings



オンにする

Teams

管理者



ダッシュボード



全メンバーに強制できる

ヘルプには、全メンバーで既定でオンと書かれています。

設定の場所と既定値は、公式文書の間で記述が分かれています。

業務で使う前に、社内の担当部署と確認してください。

出典: cursor.com/help/security-and-privacy/privacy (2026年10月5日確認)

cursor.com/docs/enterprise/privacy-and-data-governance (2026年10月5日確認)

利用上限で起きること

上限に達すると作業が止まるか想定外の請求になる

上限で止まる

Cursorの個人プラン

利用枠を使い切ったあとも使い続けるには、利用者がon-demand usageを自分で有効にする必要があります。有効にしていないと、月末の急ぎの作業でも使えなくなります。

想定外の請求になる

設定した上限

250ドル

請求された額

→ **約1万ドル**

2026年5月、クラウドのAIサービスで起きた事例を、The Registerが報じました。

対策

利用状況を定期的に見る

重い作業だけ上位のモデルにする

on-demandの上限額を決めておく

出典: cursor.com/help/models-and-usage/usage-limits (2026年10月5日確認)

The Register (2026年5月18日) theregister.com/ai-ml/2026/05/18/surprise-ai-bills

Cursorのプランと利用量

プランと席の種類によって含まれる**利用量**が変わる

区分	プラン	月額	利用量と条件
個人	Hobby	無料	Agentの回数に制限
	Pro	20ドル	Tab補完は無制限
	Pro+	60ドル	Proの 3倍
	Ultra	200ドル	Proの 20倍
Teams	Standard	40ドル	価格は1人あたり
	Premium	120ドル	Standardの 5倍

価格は税抜きの月額です。

出典: cursor.com/pricing (2026年10月5日確認)

業務用アカウントに切り替えず、**講師に知らせてください**

1

講師にチャットで知らせる

2

残りの演習は講師の画面で見る

3

完成版のコードを受け取る

匿名化せずにCursorへ入力してよいものはどれですか。

A 研修で配布した架空の企業のコード

B 顧客名が入ったエラーログ

C 本番データベースの接続文字列

D 社員の評価コメント

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。

正解はA。架空のデータは匿名化せずに使える

A

研修で配布した架空の企業のコード

配布フォルダのコードとデータは、すべて架空の企業のものです。

B

顧客名が入ったエラーログ

顧客名を置き換えてから入力します。

C

本番データベースの接続文字列

接続文字列は、匿名化しても入力しません。

D

社員の評価コメント

個人の評価は入力しません。

研修での使用量の見込み

4時間の演習はAPIの料金に換算して**5ドル程度**の見込み

約**5**ドル

講師の見込み

研修の最初と最後にPlan & Usageの割合を記録し、実際の量を自分で確かめます。

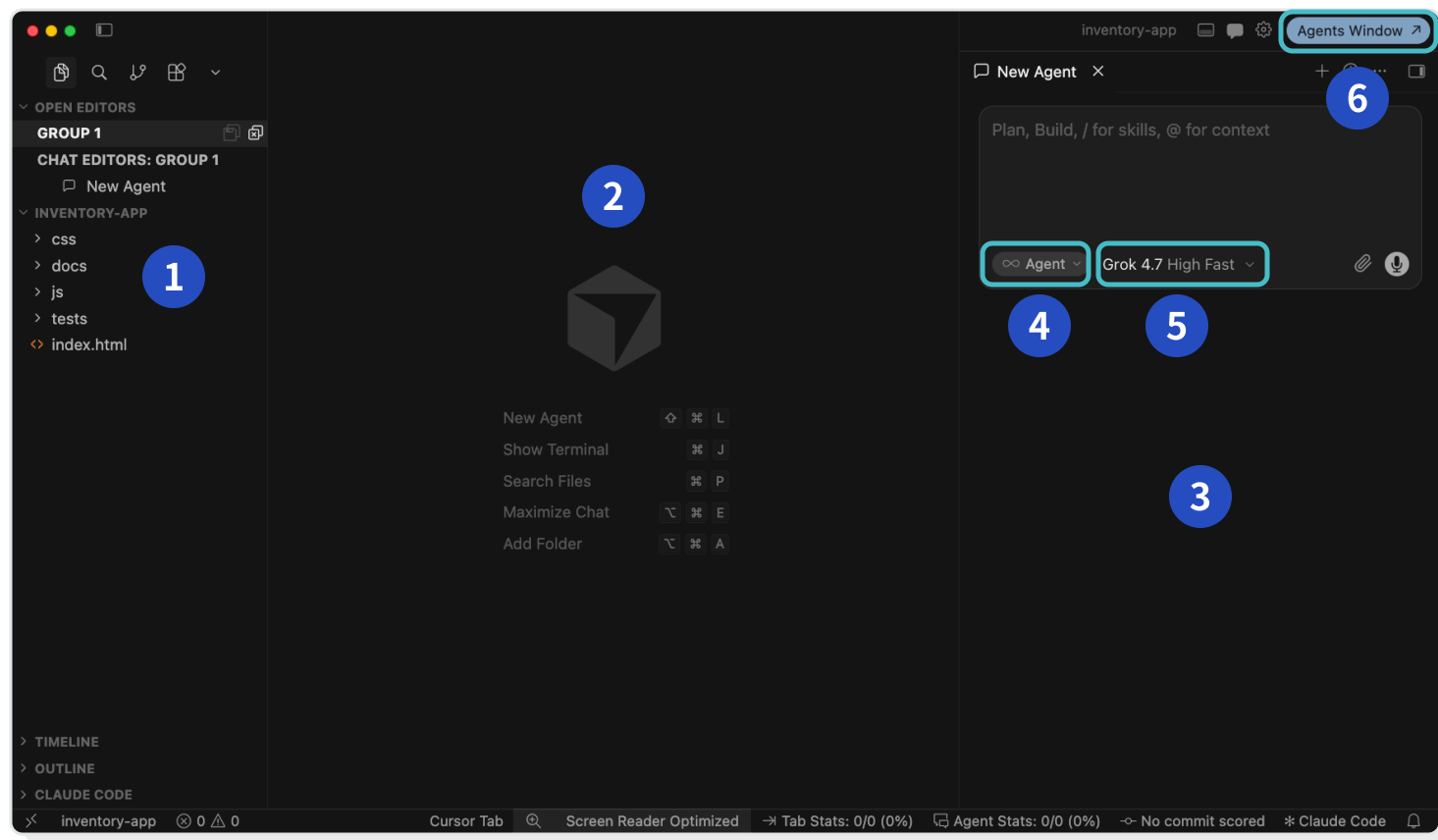
03

Cursorの基本操作

画面構成と4つの操作、在庫管理コードへの機能追加、Cursor Rules
[70min]

エディタの画面

演習はすべてエディタの画面で行います



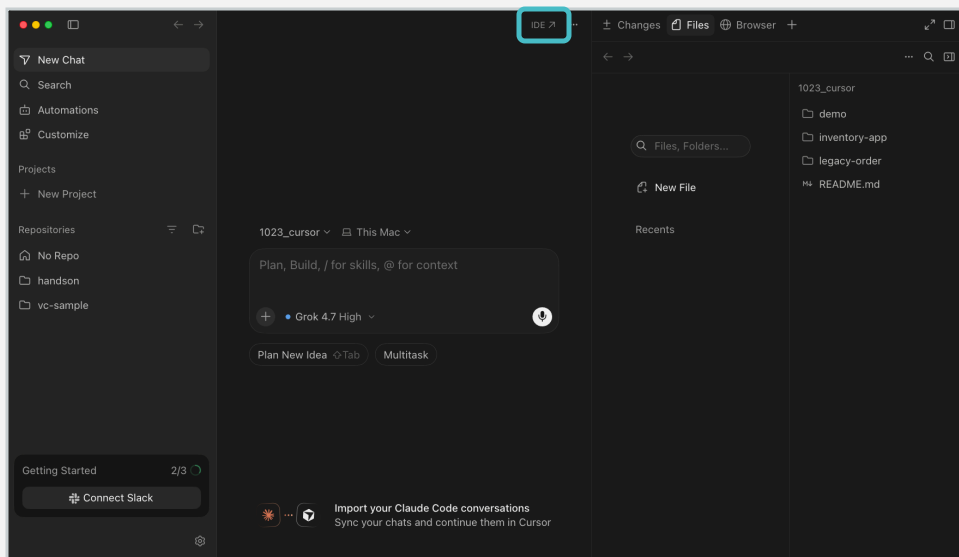
- 1 **ファイルの一覧**
開いたフォルダの中身が並びます
- 2 **エディタ**
コードを開いて読み書きする場所です
- 3 **サイドパネル**
Ctrl+Lで開き、依頼を送ります
- 4 **モードの欄**
Agentなど4つから選ぶ欄です
- 5 **モデルの欄**
回答を作るモデルを選びます
- 6 **Agents Windowボタン**
複数の依頼を並行して進める画面です

画面: Cursor 3.23.12のエディタ (macOSで撮影。Windowsでは⌘がCtrlになります)

2つの画面

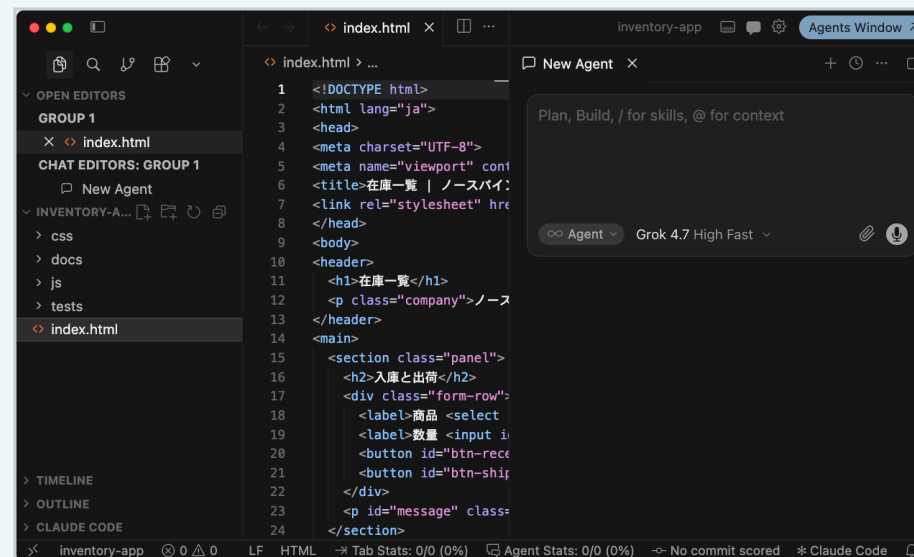
Agents Windowが開いたら、エディタへ切り替えます

Agents Window



複数のエージェントに並行して依頼する画面です。
講師のデモで、使い方を紹介します。

エディタ (IDE)



コードを開いて読み、4つの操作を使う画面です。
研修の演習は、すべてこの画面で行います。

Agents Windowの上部にある**IDE**を押すと、エディタに切り替わります。

4つの操作

作業の大きさに4つの操作を使い分けます

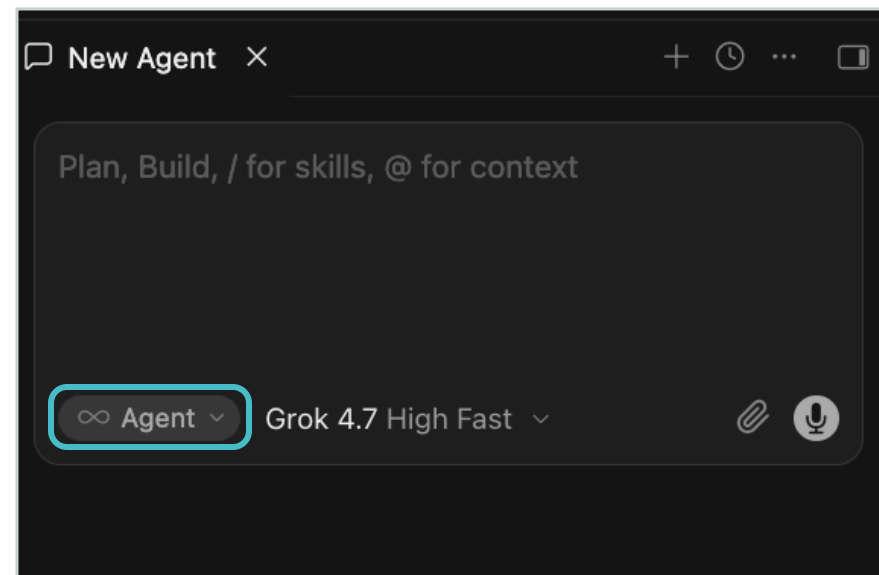
範囲が広がる

操作	呼び出し方	向いている作業	Rulesの適用
Tab補完	書きかけのコードに出る薄い文字をTabキーで確定	次の数行を書く	適用されない
インライン編集	コードを選んでCtrl+K	選んだ範囲だけを直す	適用されない
チャット	Ctrl+Lでサイドパネルを開き、Askで質問	コードの意味を聞く	適用される
エージェント	サイドパネルのAgentで依頼	複数のファイルにまたがる変更	適用される

Rulesはチームのコードの書き方を書いたファイルで、演習2で作ります。

モードはAgent・Ask・Plan・Debugの4つ

モード	Cursorが行うこと
Agent	ファイルを読んで書き換え、 コマンドの実行まで進めます。
Ask	コードを読んで答えるだけで、 ファイルは変えません。
Plan	作業の計画を先に作り、 確かめてから進めます。
Debug	不具合の原因を調べるモードです。



画面: 枠で囲んだ部分がモードの欄

Shift+Tabでモードを順に切り替え、
Ctrl+.でモードの一覧を開きます。

出典: cursor.com/help/customization/keyboard-shortcuts (2026年10月5日確認)

研修で使うキー

Windowsのキーで示します。macOSではCtrlが⌘になります。

Tab

補完の確定

Ctrl

+

/

モデルの切り替え

Ctrl

+

→

補完を単語ごとに確定

Shift

+

Tab

モードの切り替え

Ctrl

+

K

インライン編集

Ctrl

+

Shift

+

J

Cursor Settings

Ctrl

+

L

サイドパネルの開閉

Ctrl

+

Shift

+

P

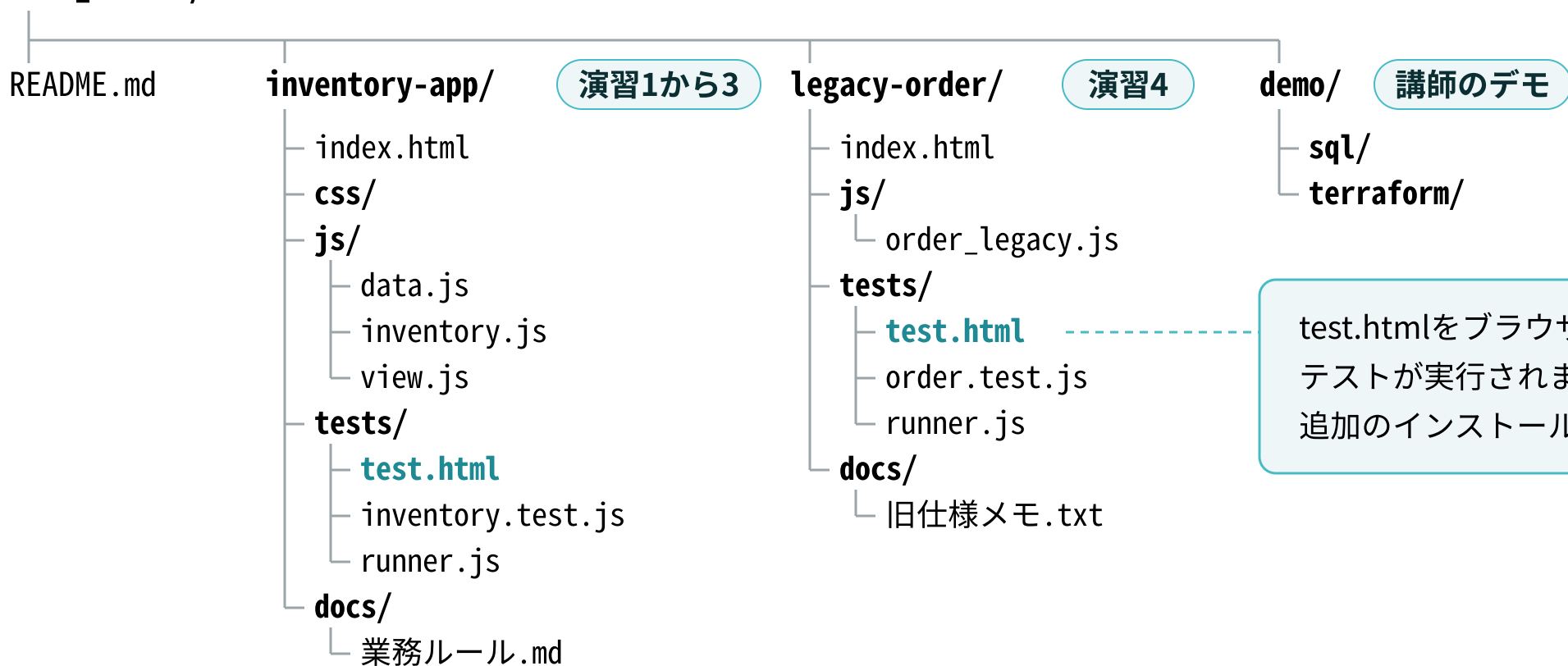
コマンドパレット

出典: cursor.com/help/customization/keyboard-shortcuts (2026年10月5日確認)

配布フォルダの構成

演習4だけlegacy-orderのフォルダを開き直します

1023_cursor/



test.htmlをブラウザで開くと、
テストが実行されます。
追加のインストールは要りません。

題材の在庫管理システム

架空のノースパイン商事が使う在庫管理システム

在庫一覧

ノースパイン商事株式会社 物流センター（架空の企業です）

入庫と出荷

商品 NP-1001 A4コピー用紙 500枚 数量 1 入庫 出荷

在庫

商品コード	商品名	分類	在庫数	引当数	利用可能数	発注点	状態
NP-1001	A4コピー用紙 500枚	用紙	120	30	90	40	
NP-1002	A3コピー用紙 500枚	用紙	22	10	12	15	要発注
NP-2001	ゲルインクボールペン 黒 10本	筆記具	64	12	52	20	
NP-2002	油性マーカー 赤 10本	筆記具	9	0	9	10	要発注
NP-3001	フラットファイル A4 10冊	ファイル	80	40	40	25	
NP-3002	リングファイル A4 背幅40mm	ファイル	35	8	27	30	要発注
NP-4001	トナーカートリッジ 黒	OA消耗品	6	3	3	5	要発注
NP-4002	インクカートリッジ 4色セット	OA消耗品	30	5	25	8	
NP-5001	ダンボール箱 100サイズ 10枚	梱包資材	50	20	30	12	
NP-5002	OPPテープ 透明 6巻	梱包資材	16	0	16	18	要発注

分類ごとの在庫金額

- 用紙：2品目、在庫金額 78,720円
- 筆記具：2品目、在庫金額 84,720円
- ファイル：2品目、在庫金額 66,700円
- OA消耗品：2品目、在庫金額 196,800円
- 梱包資材：2品目、在庫金額 90,840円

- 1

入庫と出荷

商品と数量を選び、入庫か出荷のボタンを押します。
- 2

在庫の一覧

10商品の在庫数・引当数・利用可能数・発注点が並びます。
発注が必要な商品の行には「要発注」と表示されます。
- 3

分類ごとの在庫金額

用紙や筆記具など5つの分類ごとに、品目数と金額を出します。

index.htmlをブラウザで開くと動きます。
教材サイトの演習アプリの節からも開けます。

画面: 配布時のinventory-app/index.html

在庫管理の業務ルール

発注は在庫数ではなく**利用可能数**で判定



- 出荷できるのは引当数までです。引当数を超える出荷は、エラーにして止めます。
- 利用可能数が発注点以下になったら、その商品は発注が必要です。

出典: 配布フォルダのinventory-app/docs/業務ルール.md

4つの操作を1回ずつ使い、3つの機能を足します

[35min]

STEP 1

引当の関数

Tab補完

[7min]

引当数を増やす関数
allocateを足します

STEP 2

出荷の上限チェック

インライン編集

[7min]

引当数を超える出荷を
例外で止めます

STEP 3

業務ルールの確認

チャット

[6min]

発注の判定のしかたを
質問して確かめます

STEP 4

保管棚の追加

エージェント

[15min]

保管棚の列と絞り込みを
3つのファイルに足します

手順と依頼文は、ハンズオンガイドの演習1にあります。依頼文はガイドからコピーしてください。

コメントと1行目は自分で打ち、続きを読んで確定します

[7min]

- 1 js/inventory.jsを開き、
ship関数のすぐ上の空行に、
右の白い文字の2行を打ちます。
- 2 薄い文字で出た続きを、
業務ルールの「引当」と比べます。
- 3 合っていればTabで確定し、
違えばEscで消して自分で書きます。
- 4 ファイルの末尾にあるreturnの
一覧に、allocateを足します。

```
}  
// 受注した数を引き当てる。利用可能数を超えては引き当てられない  
function allocate(stocks, code, qty) {  
    ... (ここに薄い文字の提案が出ます)  
  
    // 引当済みの商品を出荷する。倉庫の数と引当数の両方が減る  
    function ship(stocks, code, qty) {
```

白い2行が自分で打つ部分で、灰色の行は配布時からあるコードです。

docs/業務ルール.mdの「引当」

受注したら引当をする。引当数が増え、利用可能数が減る

コメントに**上限まで**書くと、提案が業務ルールに近づきます

短いコメント 引き当てる数と上限が書かれていません。

```
// 引当  
function allocate(stocks, code, qty) {
```

何を、どこまでを書いたコメント 対象は受注した数で、上限は利用可能数です。

```
// 受注した数を引き当てる。利用可能数を超えては引き当てられない  
function allocate(stocks, code, qty) {
```

- Tab補完は、カーソルの前後にあるコードとコメントから続きを推測します。

ship関数を選んでCtrl+Kで直します

[7min]

- 1 js/inventory.jsのship関数の全体を選びます。
- 2 Ctrl+Kを押し、表示された入力欄に右の依頼文を貼り付けて送信します。
- 3 差分を読み、エラーの文言が日本語になっているかを確認してから採用します。

依頼文

出荷数が引当数を超える場合は、出荷せずに例外を投げてください。エラーの文言は「商品コード NP-1001 は引当数 3 を超えて出荷できません」の形にしてください。

採用したあとの確かめ方

index.htmlでNP-1001を数量31で出荷すると、日本語のエラーが表示され、在庫は変わりません。

NP-1001の初期データは引当数30なので、31は上限を超えます。

先に**自分で数えて**から、チャットに聞きます

[6min]

AIを使わない

1

docs/業務ルール.mdの
発注点の節を読みます。

AIを使わない

2

index.htmlで発注が必要な
商品の数を数えます。

チャット

3

Ctrl+Lを押し、
下の質問文を送ります。

AIを使わない

4

Cursorの答えと
自分の答えを比べます。

質問文

@docs/業務ルール.md と @js/inventory.js を読んで教えてください。

発注が必要かどうかは、どの数をどの値と比べて判定していますか。

初期データで発注が必要な商品のコードを、理由と一緒に全部挙げてください。

- @に続けてファイルの場所を書くと、Cursorはそのファイルを読んでから答えます。

発注が必要な商品は、**NP-3002**を含む5件です

商品コード	在庫数	引当数	利用可能数	発注点
NP-1002	22	10	12	15
NP-2002	9	0	9	10
NP-3002	35	8	27	30
NP-4001	6	3	3	5
NP-5002	16	0	16	18

在庫数35だけを見ると、
発注点30を上回ります。
引当数8を引いた利用可能数
27は、発注点30を下回ります。

判定は、利用可能数が発注点以下かどうかで行います。利用可能数は在庫数から引当数を引いた数です。

送る前に、変わるはずのファイルを3行で書きます

依頼の前に書く3行メモ

1	data.jsの10商品にlocation
2	inventory.jsにfilterByZone
3	view.jsとindex.htmlに保管棚の列と選ぶ欄

依頼文の要約

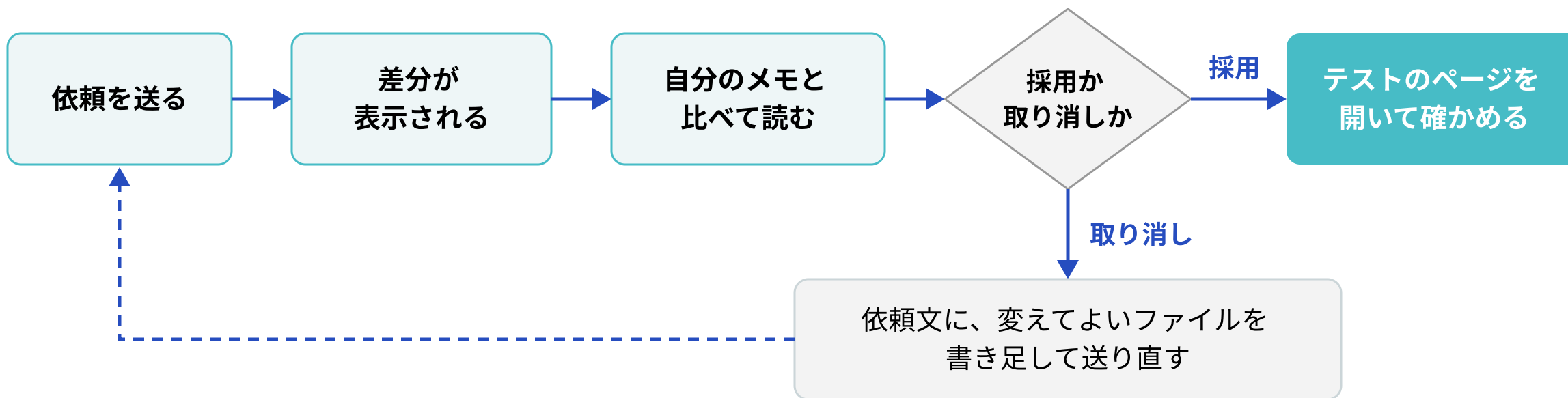
- 1 **js/data.js**
10商品すべてにA-01の形のlocation
- 2 **js/inventory.js**
棚の列の英字で絞り込むfilterByZone
- 3 **在庫の一覧**
保管棚の列と、棚の列を選ぶ欄
- 4 **tests/inventory.test.js**
filterByZoneのテスト1件

差分が表示されたら、このメモと見比べます。

依頼文の全文は、ハンズオンガイドの演習1 STEP 4からコピーしてください。

差分を読んでから採用

読まずに採用すると、**変わった場所**を説明できなくなります



メモになかったファイルが変わっていたら、採用する前に理由をチャットで聞きます。

演習1のできあがり

保管棚の列と、棚で絞り込む欄が増えます

在庫

保管棚 A ▼

商品コード	商品名	分類	保管棚	在庫数	引当数	利用可能数	発注点	状態
NP-1001	A4コピー用紙 500枚	用紙	A-01	120	30	90	40	
NP-1002	A3コピー用紙 500枚	用紙	A-02	22	10	12	15	要発注
NP-5001	ダンボール箱 100サイズ 10枚	梱包資材	A-05	50	20	30	12	
NP-5002	OPPテープ 透明 6巻	梱包資材	A-06	16	0	16	18	要発注

在庫管理のテスト結果

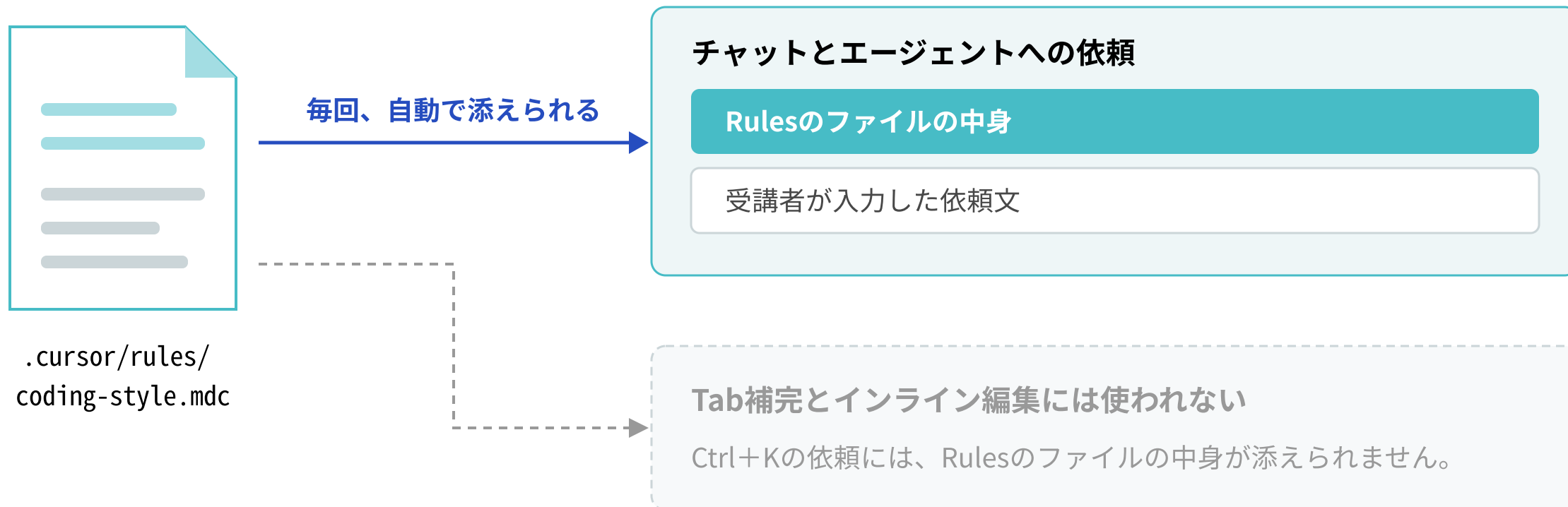
このページをブラウザで開くと、テストがすべて実行されます。コードを直したら、ページを再読み込みしてください。

13件中 13件 合格

合格 利用可能数は在庫数から引当数を引いた数

- ☐ テストが13件中13件合格になる
- ☐ 引当数を超える出荷が日本語のエラーで止まる
- ☐ 保管棚でCを選ぶとトナーとインクの2行だけになる

Rulesは毎回の依頼に自動で添えるチームの決まりごと



出典: cursor.com/docs/rules (2026年10月5日確認)

Rulesの4つの適用タイプ

使われるタイミングは先頭の書き方で決まる

名前	先頭に書くこと	使われるとき
Always Apply	<code>alwaysApply: true</code>	毎回の会話 演習2で使う
Apply Intelligently	descriptionだけ	エージェントが関係すると判断したとき
Apply to Specific Files	globs	一致するファイルが文脈にあるとき
Apply Manually	descriptionもglobsも書かない	チャットで@ルール名と書いたとき

先頭とは、ファイルの1行目の---から次の---までの部分です。

出典: cursor.com/docs/rules (2026年10月5日確認)

Rulesの置き場所

種類	置き場所	使われる範囲
Project Rules	.cursor/rules/の.mdc	そのプロジェクト
User Rules	Customize > Rules	自分の全プロジェクト
Team Rules	Cursorのダッシュボード	TeamsとEnterpriseのチーム全体
AGENTS.md	プロジェクト直下とサブフォルダ	そのプロジェクト frontmatterなしのMarkdownで書く

.cursor/rules/に置いた、拡張子が.mdのファイルは読まれません。演習2では.mdcで作ります。

出典: cursor.com/docs/rules (2026年10月5日確認)

Rulesを置く前と後で、同じ依頼の結果を比べます

1	STEP 1 自分のチームの規約の書き出し レビューでよく指摘する書き方を3つメモに書く	AIを使わない	[5min]
2	STEP 2 Rulesなしでの依頼 差分は採用せずに取り消す	エージェント	[5min]
3	STEP 3 Rulesのファイルの作成 /create-ruleに続けて規約を送る	エージェント	[8min]
4	STEP 4 同じ依頼での見比べ 差分を採用し、tests/test.htmlで全件合格を確認する	エージェント	[7min]

STEP 2とSTEP 4で送る依頼文

js/inventory.js に、間違えて在庫した数を取り消す関数を追加してください。

入力欄に/create-ruleと打ち、規約を続けて送ります

依頼文

/create-rule

在庫管理コードの書き方を、常に適用されるRulesにしてください。

- 関数名は「動詞＋名詞」のcamelCaseにする
- 数量を表す引数と変数はqtyとする
- 在庫の配列を直接書き換えず、mapやfilterで新しい配列を返す
- エラーは日本語のメッセージで例外を投げ、先頭に「商品コード〈コード〉は」を付ける
- 関数を足したり変えたりしたら、tests/inventory.test.js にテストを1件以上足す

• STEP 1で書いた自分の規約

点線の行に、STEP 1で書いた自分の規約も足して送ります。

送ると、エージェントが.cursor/rules/に.mdcのファイルを作ります。

できあがるRulesのファイル

先頭の**alwaysApply: true**と自分の規約を確かめます

 .cursor/rules/coding-style.mdc

```
---
description: ノースパイン商事の在庫管理コードの書き方
alwaysApply: true          ← Always Applyの指定
---

# 在庫管理コードの書き方
- 関数名は「動詞＋名詞」のcamelCaseにする（例: cancelReceipt、listReorderItems）
- 数量を表す引数と変数はqtyとし、quantityやnumとは書かない
- 在庫の配列を直接書き換ええない。mapやfilterで新しい配列を返す
- エラーは日本語のメッセージで例外を投げ、先頭に「商品コード <コード> は」を付ける
- コメントは日本語で、処理の理由だけを書く          ← STEP 1で足した自分の規約の例
- 関数を足したり変えたりしたら、tests/inventory.test.jsにテストを1件以上足す
```

上のファイルは一例で、エージェントが作る文面は受講者ごとに少しずつ変わります。

守られやすいRulesの書き方

判定できる書き方の規約ほど守られる

守られにくい書き方

わかりやすい名前にする

テストをちゃんと書く

エラーを丁寧に



守られやすい書き方

関数名は「動詞＋名詞」のcamelCaseにする

関数を足したらtests/inventory.test.jsに1件以上足す

日本語で、先頭に「商品コード〈コード〉は」を付ける

Rulesに書いたのに守られなかった規約は、判定できる書き方に直してもう一度試してください。

Cursor Rulesが使われないのはどれですか。

A エージェントへの依頼

B チャットでの質問

C /create-ruleのあとの依頼

D インライン編集 (Ctrl+K)

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。

正解はD。Rulesはインライン編集には使われない

D

インライン編集 (Ctrl+K)

Rulesが使われるのはチャットとエージェントだけで、Tab補完とインライン編集には使われません。

A

エージェントへの依頼

エージェントへの依頼なので、Rulesが使われます。

B

チャットでの質問

チャットへの質問なので、Rulesが使われます。

C

/create-ruleのあとの依頼

チャットとエージェントへの依頼なので、Rulesが使われます。

出典: cursor.com/docs/rules (2026年10月5日確認)

休憩

[10min]

再開後は、S04の「モデルの使い分けとレガシーコード改修」から始めます。

04

モデルの使い分けと レガシーコード改修

利用状況の確認、テストで守りながらの書き直し、SQLとTerraformのデモ
[95min]

2つの利用枠

Proの利用枠はCursorのモデル用と他社のモデル用の2つ



Cursor Models

Cursorが提供するモデル

Composer 2.5

Grok 4.7

Grok 4.6

Grok 4.5

Other Models

他社のモデル

使った量は、APIの価格で数えます。
演習3では、ComposerとGrok以外の
モデルを1つ選んで比べます。

どちらの枠も、毎月の請求日にリセットされます

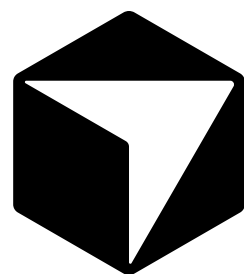
上限に達したあとは、on-demand usageを有効にすると、APIの価格で使い続けられます。
個人プランでは、利用者が自分で有効にします。Tab補完は、Proでは無制限です。

出典: cursor.com/docs/models-and-pricing、cursor.com/pricing (2026年10月5日確認)

※ 各ロゴは各社の商標です。

当日の基本のモデル

当日の基本はCursor独自のComposer 2.5



Composer 2.5

作り手	Cursorが自社で作ったモデルです
-----	--------------------

既定	高速版のFastが選ばれます
----	----------------

利用枠	Cursor Modelsの枠から使われます
-----	------------------------

出典: cursor.com/docs/models/cursor-composer-2-5 (2026年10月5日確認)

※ 各ロゴは各社の商標です。

モデルの選び方

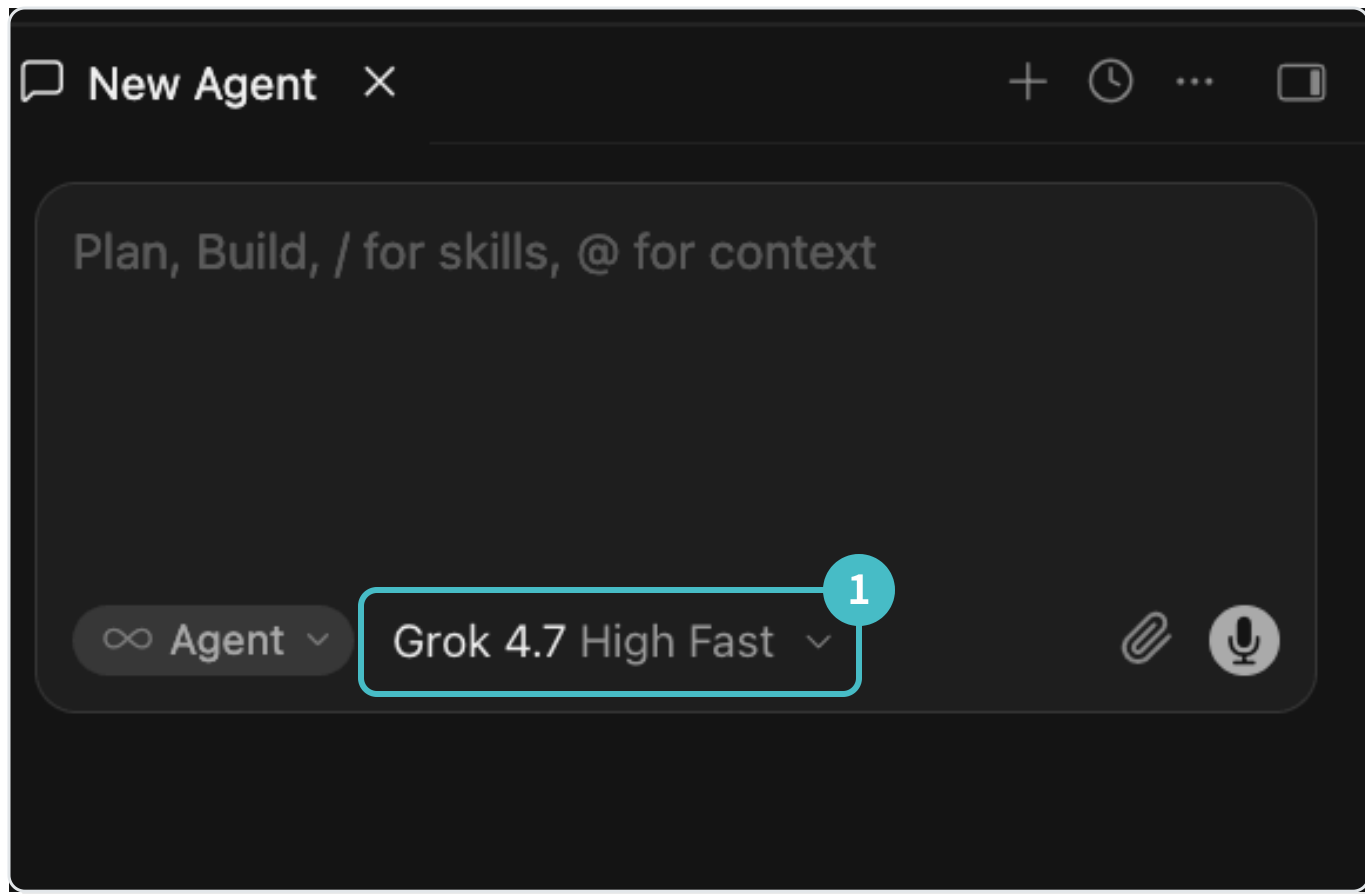
モデルを選ぶ基準は作業の重さと誤りへの気づきやすさ

軽いモデル    上位のモデル

	作業が軽い	作業が重い
誤りに気づける	軽いモデル（Composer 2.5） ・ ファイルの検索 ・ 関数の場所の調査 ・ 定型の書き換え	まず軽いモデル 結果を見て、足りなければ 上位のモデルに切り替えます
誤りに気づきにくい	上位のモデル＋別の依頼でレビュー 書かせた依頼とは別の依頼で、 誤りがないかを点検させます	上位のモデル ・ 設計 ・ レガシーコードの仕様の読み取り ・ 複数ファイルの変更

演習3では、調べ物を軽いモデルのサブエージェントに任せ、設計を上位のモデルで行います。

モデルの切り替え方



画面: Cursor 3.23.12のサイドパネル (macOSで撮影)

1 モデルの欄を押す

入力欄の下にあるモデルの欄を押し、一覧から選びます

2 Ctrl+/で切り替える

Ctrl+/を押すたびに、モデルが順に切り替わります

3 既定のモデルを決める

Cursor Settings > Modelsで、既定のモデルを選びます

Cursor Settingsは、
Ctrl+Shift+Jで開きます

同じ依頼を2つのモデルで比べ、調べ物は軽いモデルに任せます

- | | | | |
|---|--|---------|---------|
| 1 | STEP 1 2つのモデルでの同じ依頼
Composer 2.5と別のモデルに同じ依頼を送り、3点で比べます | チャット | [10min] |
| 2 | STEP 2 調べ物用のサブエージェント
調べ物を軽いモデルで行う設定を、.cursor/agents/に保存します | ファイルを作る | [8min] |
| 3 | STEP 3 調べ物と設計の分担
/code-searcherで調べ物を任せ、設計は上位のモデルで書きます | エージェント | [8min] |
| 4 | STEP 4 利用状況の見比べ
Plan & Usageの割合と、明細のトークン数を書き写します | AIを使わない | [4min] |

目安は[30min]です。手順と依頼文は、ハンズオンガイドの演習3にあります。

Composer 2.5と、ComposerとGrok以外のモデルで比べます

依頼文

発注が必要な商品について、仕入先に送る発注書の下書きを作る機能を設計してください。

コードはまだ書かず、関数の名前と引数と戻り値、発注する数の決め方を、箇条書き10行以内で教えてください。

1 モデルの欄でComposer 2.5を選んで送る

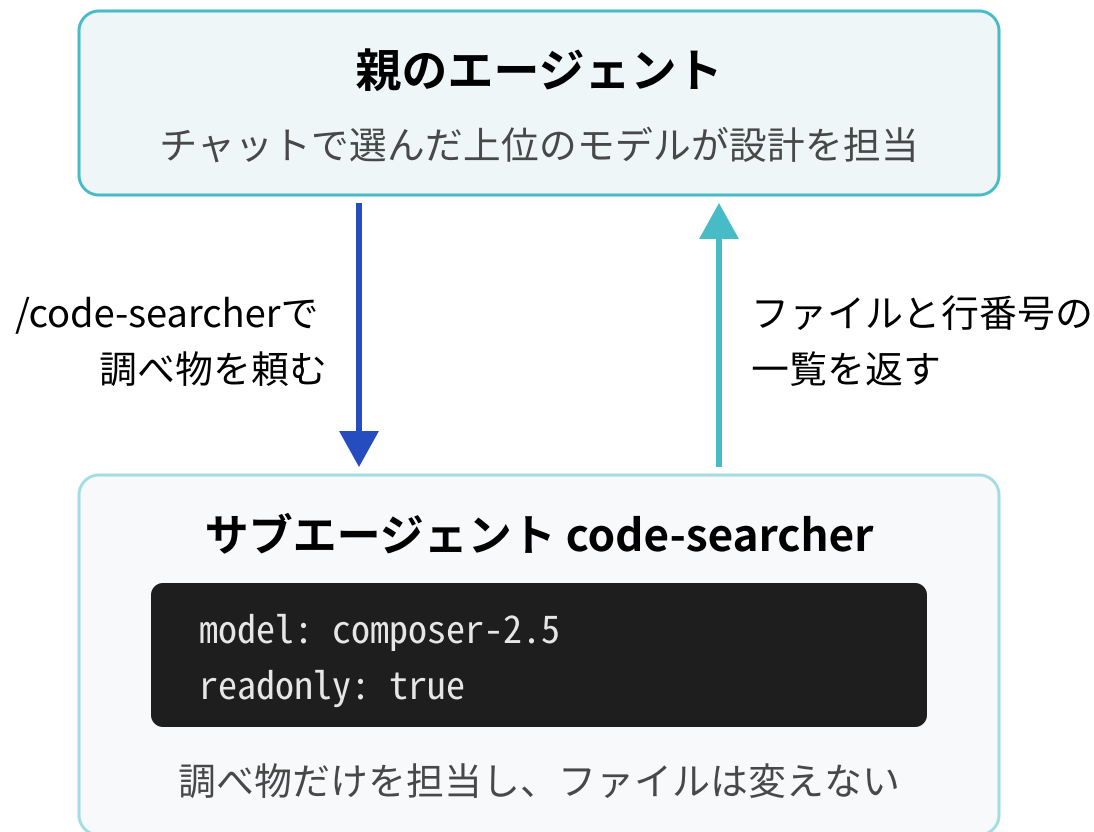


2 新しいチャットで、もう1つのモデルに送る

比べる点	Composer 2.5	もう1つのモデル
返ってくるまでの時間		
発注する数の決め方の具体さ		
10行に収まったか		

空欄の2列は、2つの答えを見比べて自分のメモに書いてください。

作業ごとにモデルを変える公式の方法はサブエージェント



Rulesでは指定できない

Rulesの先頭に書けるのは、descriptionとglobsとalwaysApplyの3つです。モデルを指定する項目はありません。

モデルは、.cursor/agents/のファイルの先頭に書きます。

出典: cursor.com/docs/subagents、cursor.com/docs/rules (2026年10月5日確認)

inventory-appの.cursor/agents/に保存してください

```
.cursor/agents/code-searcher.md
```

```
---
```

```
name: code-searcher
```

```
description: コードベースの検索専用。関数やファイルの場所、変数の使われ方を調べるときに使う。
```

```
コードは書き換えない。
```

```
model: composer-2.5
```

●----- このサブエージェントだけを軽いモデルで動かす指定

```
readonly: true
```

●----- ファイルを書き換えない指定

```
---
```

```
あなたは在庫管理システムのコードを調べる担当です。
```

```
依頼されたら、次の順で答えてください。
```

1. 関係するファイルと関数を、ファイルのパスと行番号つきで一覧にする
2. それぞれが何をしているかを1行で書く
3. 依頼の答えに必要な事実だけを書き、設計の提案はしない

保存したら、STEP 3で依頼の先頭に/code-searcherと書いて呼び出します。

先頭の/code-searcherで、調べ物だけを任せます

準備

チャットのモデルは、STEP 1で選んだComposer以外のモデルのままにしてください

依頼文

/code-searcher 在庫数が引当数を変えているところをすべて洗い出してください。

そのあと、その結果をもとに、返品を受けたときに在庫を戻す関数の設計を5行以内で書いてください。

コードはまだ書かないでください。

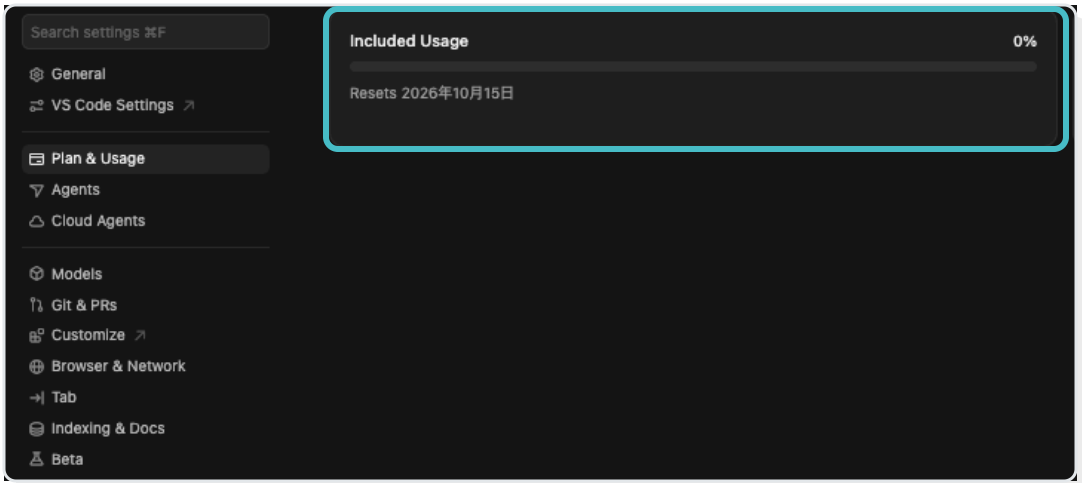
確かめること

- ☐ チャットに出るサブエージェントのカードに、composer-2.5と表示されます
- ☐ 違うモデル名が出たら、プランで使えないモデルを指定していないかを見直します

利用状況を見る2つの画面

割合は**アプリ**で、モデル別のトークン数は**ブラウザ**で確かめます

アプリ Cursor Settings > Plan & Usage



無料プランで撮影

Ctrl+Shift+JでCursor Settingsを開きます。
枠で囲んだIncluded Usageに割合が表示されます。

ブラウザ cursor.com/dashboard/usage

項目	分かること
Total tokens	期間内に使ったトークンの合計
Included	プランの利用枠で使った分
On-demand	枠を超えて従量で使った分
モデル別の集計	モデルごとの使用量
Export CSV	使用量の明細をCSVで保存

依頼ごとの費用は、Spendingの画面で見られます。

出典: cursor.com/help/models-and-usage/usage-limits (2026年10月5日確認)

AutoとCursor Router

モデルの欄でAutoを選ぶと、Cursor Routerが依頼ごとにモデルを選びます。
選び方の基準は、2026年7月22日の更新でOptimize Forの3つから決められるようになりました。

Optimize For



使えるプラン

2026年10月時点では、TeamsとEnterpriseのプランで使えます。
個人プランへは後日提供すると、公式ヘルプに書かれています。

出典: cursor.com/docs/cursor-router (2026年10月5日確認)

演習4の題材

2012年から継ぎ足してきた受注金額の計算

受注金額の計算

ノースパイン商事株式会社（架空の企業）の、古い受注金額計算の画面です。

会員ランク	一般 ▼	地域	本州・四国・九州 ▼
-------	------	----	------------

NP-1001	数量	3
NP-2001	数量	2
	数量	

計算する

小計	3,840円
値引	0円
送料	1,100円
消費税	384円
合計	5,324円

js/order_legacy.js

```
/*
 * 受注金額計算モジュール
 * 作成 2012/10（システム部）
 * 2014/04 消費税8%対応
 * 2016/07 会員ランク対応
 * 2019/10 送料改定
 * ※ノースパイン商事株式会社（架空）の演習用コードです
 */
```

会員ランクと地域と明細を入れて「計算する」を押すと、小計・値引・送料・消費税・合計が出ます。

演習4では、この計算をしているcalc関数を、テストで動きを固定してから書き直します。

画面: 配布時のlegacy-order/index.html

古いコードの特徴

1つの関数に値引・送料・消費税がすべて入っている

js/order_legacy.js

抜粋、左は行番号

```
10 var SHOHIN = {  
23 var zaiko = {  
28 var flg = 0;  
32 function calc(o) {  
33     var shokei = 0;  
36     var nebiki = 0;  
37     var soryo = 0;  
38     var zeï = 0;  
80     nebiki = Math.floor(shokei * 0.1);  
82     nebiki = Math.floor(shokei * 0.05);  
105     if (shokei >= 10000) {  
108         soryo = 800;
```

1 グローバル変数が3つ

SHOHIN、zaiko、flg
calc関数の外から書き換えられます

2 ローマ字の変数名

shokei・nebiki・soryo・zeï
小計・値引・送料・消費税のローマ字

3 直接書かれた数値

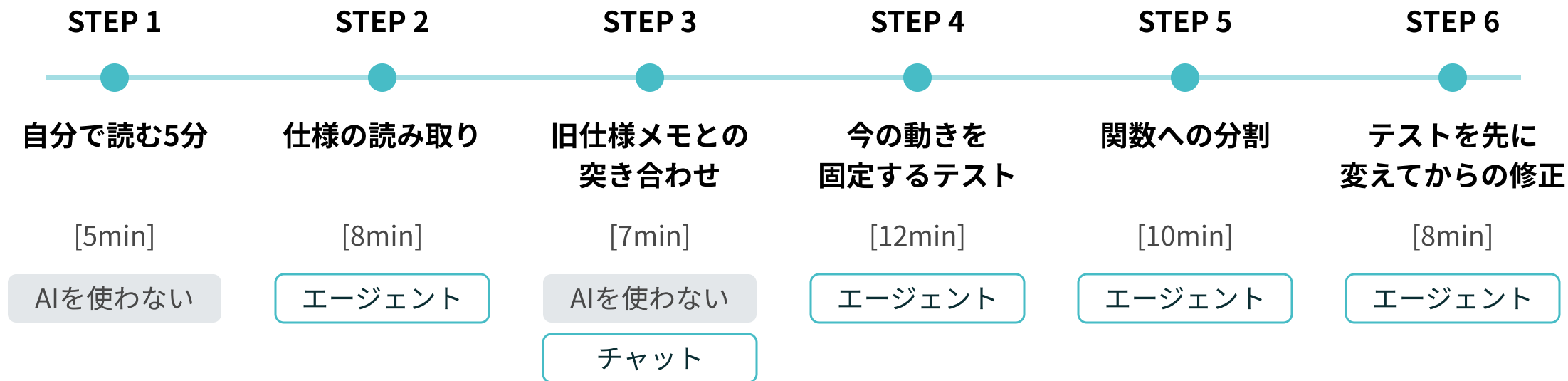
0.1、0.05、10000、800
0.1が値引率と消費税率の両方に出ます

旧仕様メモ（2019/10）と食い違う箇所が、このコードのどこかにあります。

今の動きをテストで固定してから書き直します

コードと旧仕様メモを読む

テストで守りながら書き直す



目安は[50min]です。Cursorでlegacy-orderのフォルダを開き直してから始めます。
STEP 3では、旧仕様メモに自分で印を付けてから、チャットに聞きます。

メモを先に書き、エージェントの読み取りと比べます

STEP 1 自分で読む

[5min]

AIを使わない

js/order_legacy.jsのcalc関数を、上から自分で読みます。値引と送料と消費税の計算を1行ずつメモに書いてください。読み切れないところは「不明」と書いて構いません。

値引

送料

消費税

不明

STEP 2 仕様の読み取り

[8min]

エージェント

依頼文の要約

- 1 docs/仕様_読み取り.mdに書き出す
- 2 入力・エラー・値引・送料・消費税・合計の6つの見出しに分ける
- 3 コードから読み取れる事実だけを書き、コメントとの食い違いは注記する
- 4 全体でA4の1枚に収まる長さにする

自分が「不明」にしたところが、どう説明されたかを確認めます。

自分で印を付けてから、チャットに聞きます

docs/旧仕様メモ.txt (抜粋)

■送料

- ・ 本州・四国・九州 … 800円
- ・ 北海道 … 1,200円
- ・ 沖縄・離島 … 1,800円
- ・ 値引後の金額に消費税を足した額が
11,000円以上なら送料無料
(沖縄・離島は対象外)
(重量の加算の2行は省略)

■消費税

- ・ 値引後の金額の10%
- ・ 1円未満は切り捨て (会員ランクに関係なく同じ)

質問文

@docs/旧仕様メモ.txt と
@js/order_legacy.js を比べて、計算の結果が
変わる食い違いを挙げてください。
コメントだけの食い違いは除いてください。
それぞれ、食い違いが出る注文の例を
1つずつ、会員ランクと地域と明細で
示してください。

進め方

- 1 docs/旧仕様メモ.txtを自分で読む
- 2 仕様_読み取り.mdと食い違っていきそうな箇所に印を付ける
- 3 チャットに質問文を送る

計算の結果が変わる食い違いは2つ

食い違いと注文の例	旧仕様メモ	今のコード
1 送料無料の判定 GOLD会員が本州で NP-4001とNP-2001を1個ずつ	値引後の税込金額で判定 送料800円	値引前の小計で判定 送料0円
2 消費税の端数 GOLD会員がNP-3002を1個	常に切り捨て 消費税37円	GOLDだけ四捨五入 消費税38円

コメントの「消費税（8%）」は、計算に影響しない古いコメントです。

受注金額の計算

ノースバイン商事株式会社（架空の企業）の、古い受注金額計算の画面です。

会員ランク **GOLD** 地域 **本州・西国・九州**

NP-4001	数量 1
NP-2001	数量 1
	数量

計算する

小計	11,000円
値引	1,100円
送料	0円
消費税	990円
合計	10,890円

今のコードは送料0円

受注金額の計算

ノースバイン商事株式会社（架空の企業）の、古い受注金額計算の画面です。

会員ランク **GOLD** 地域 **本州・西国・九州**

NP-4001	数量 1
NP-2001	数量 1
	数量

計算する

小計	11,000円
値引	1,100円
送料	800円
消費税	990円
合計	11,690円

直したあとは送料800円

書き直す前に**今の結果を正解**にしたテストを書きます

STEP 4

① 今の動きを固定

今のコードの結果を、そのまま正解にしたテストを書きます。旧仕様メモとの食い違いも、今のまま書きます。



STEP 5

② 書き直す

テストが全件合格することを確かめてから、コードを書き直します。

③ 全件合格なら動きは同じ

書き直したあとも全件合格なら、外から見た動きは変わっていません。

取り消して2へ

不合格が出た

不合格が出たとき

書き直して動きが変わった証拠なので、差分を取り消してやり直します。

test.htmlが11件中11件合格になれば完了

依頼文

tests/order.test.js に、calc 関数の今の動きを固定するテストを10件足してください。

- 今のコードの結果を正解として書く。旧仕様メモと食い違う箇所も、今の結果のまま書く
- 一般、SILVER、GOLDの値引、4つの地域の送料、送料無料、重量の加算、在庫不足、登録のない商品コードを1件以上ずつ含める
- テスト名は、何を確かめるかが読める日本語にする

受注金額の計算のテスト結果

このページをブラウザで開くと、テストがすべて実行されます。コードを直したら、ページを再読み込みしてください。

1件中 1件 合格

合格 明細がないときは合計0円でメッセージを返す

画面: 始める前の
test.html (1件中1件合格)

tests/test.htmlをブラウザで開き、11件中11件合格になることを確かめてください。

合格しないテストがあれば、テストとコードのどちらが正しいかをチャットで聞いて直します。

計算の結果は1円も変えずに分けます

分けたあとの形

calc

入力と戻り値の形は同じ

値引を計算する関数

送料を計算する関数

消費税を計算する関数

名前の付いた定数

依頼文

js/order_legacy.js の calc 関数を、値引、送料、消費税をそれぞれ計算する関数に分けてください。

- calc の入力と戻り値の形は変えない
(index.html から呼ばれているため)
- 計算の結果は1円も変えない。旧仕様メモとの食い違いも、この段階では直さない
- 割引率や送料のような数値は、名前の付いた定数にまとめる
- 終わったら tests/test.html の結果がどうなるかを教えてください

tests/test.htmlを開き直し、11件中11件合格のままか確かめます。不合格が出たら差分を取り消してやり直します。

不合格のテストを1件足してから、コードを直させます

依頼文

旧仕様メモでは、送料無料の判定は値引後の税込金額が11,000円以上かどうかで行います。

- 1 GOLD会員、本州、NP-4001とNP-2001を1個ずつの注文で、送料800円、合計11,690円を正解とするテストをtests/order.test.js に足してください
- 2 今の時点で、そのテストだけが不合格になることを教えてください
- 3 そのあとで、送料無料の判定を旧仕様メモどおりに直してください

受注金額の計算のテスト結果

このページをブラウザで開くと、テストがすべて実行されます。コードを直したら、ページを再読み込みしてください。

12件中 11件 合格

合格	明細がないときは合計0円でメッセージを返す
合格	一般会員、本州、値引なし
合格	一般会員は10点以上で3%の値引
合格	SILVER会員は5%の値引
合格	GOLD会員は10%の値引
合格	北海道の送料は1200円
合格	離島は金額が大きくても送料1800円
合格	本州で税込11000円以上なら送料無料
合格	5kgを超えた分は1kgごとに100円を足す
合格	在庫が足りないときは0円でメッセージを返す
合格	登録されていない商品コードは0円でメッセージを返す

不合格 送料無料の判定は値引後の税込金額で行う（旧仕様メモ通り）
期待値 [11000,1100,800,990,11690] に対して実際は [11000,1100,0,990,10890]

画面: 送料のテストを1件足した直後のtest.html。
その1件だけが不合格です。

足す前
11件中11件合格



テストを足した直後
12件中11件合格



直したあと
12件中12件合格

演習4のできあがり

12件中12件合格で、数値の直書きがなければ完成です

- ☐ tests/test.htmlが「12件中 12件 合格」になる
- ☐ calc関数の中に、割引率や送料の数値が直接書かれていない
- ☐ 画面のindex.htmlで、GOLD会員の11,000円の注文の送料が800円になる

受注金額の計算

ノースバイン商事株式会社（架空の企業）の、古い受注金額計算の画面です。

会員ランク GOLD 地域 本州・四国・九州

NP-4001	数量	1
NP-2001	数量	1
	数量	

計算する

小計	11,000円
値引	1,100円
送料	800円
消費税	990円
合計	11,690円

画面: 直したあとの送料800円

発展

消費税の端数も、STEP 6と同じ順で直します。
直す前に、旧仕様メモとコードのどちらを
正とするかを誰が決めるのかを、メモに1行書いてください。

表定義を読ませてから、集計のSQLを書かせます

demo/sql/schema.sqlの表と列（型は省略）

products	code, name, category, unit_price, reorder_point
customers	customer_id, name, member_rank, region
orders	order_id, customer_id, ordered_at, discount, shipping_fee, tax, total
order_lines	order_id, line_no, product_code, quantity, unit_price

依頼文

- 1 demo/sql/schema.sql を読んで、2026年9月の分類ごとの売上金額と数量を、売上金額の大きい順に出すSQLを書いてください。値引の前の金額で集計してください。
- 2 いま書いたSQLについて、会員ランクごとの内訳も出るように直してください。
- 3 いま書いたSQLが遅くなる場合に、どの列に索引を張ればよいかを、理由と一緒に3つまで挙げてください。

講師がエージェントに1から順に送ります。

列名と索引の理由が表定義に沿っているかを見ます

依頼	見せること	見るところ
依頼1	schema.sqlを読ませ、分類ごとの月次売上のSQLを書かせる	表定義を読ませると、列名を推測で書かなくなるか
依頼2と3	会員ランクの内訳を足し、索引の候補を理由つきで3つ挙げさせる	索引の候補の理由が、表定義の事実に基づいているか

確かめ方 SQLに出てくる列名を、schema.sqlの4つの表の列と1つずつ照らし合わせます。

依頼文の例は、配布フォルダのdemo/sql/依頼文の例.mdにあります。

生成したコードは**実行せず**、レビューまで見せます

demo/terraform/main.tf (抜粋)

```
# このファイルは実行しない。

provider "azurerm" {
  features {}
}

resource "azurerm_resource_group" "inventory" {
  name      = "rg-np-inventory-demo"
  location  = "japaneast"
}
```

見るところ

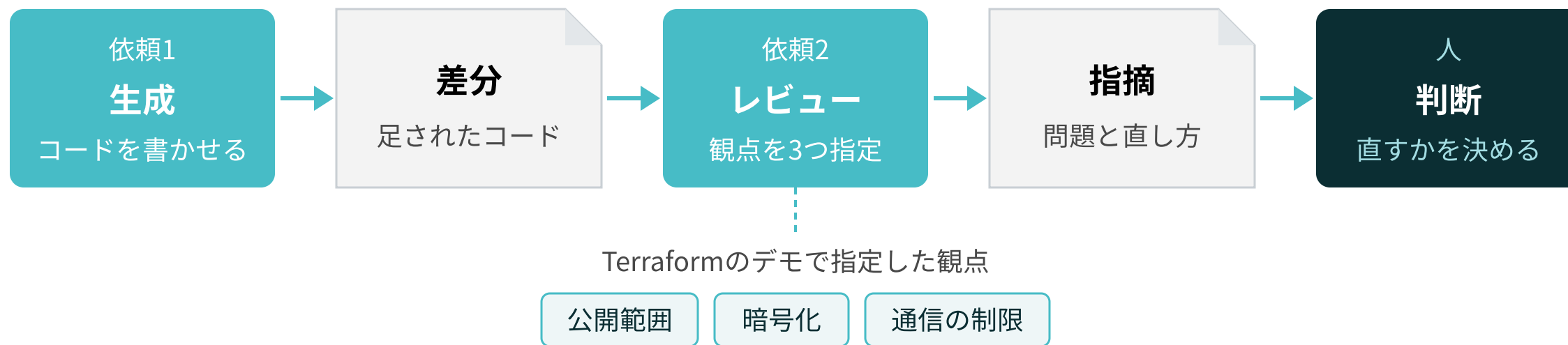
既存のリソースグループの名前に合わせられるか

依頼文

- 1 demo/terraform/main.tf に、在庫管理画面を置くためのストレージアカウントと静的サイトの設定を足してください。名前は既存のリソースグループに合わせてください。
- 2 いま足したコードを、セキュリティの観点でレビューしてください。公開範囲、暗号化、通信の制限の3点で、問題があれば直し方も書いてください。

2つ目の依頼で、足したコードを点検させます。

生成とレビューは別の依頼に分けます



同じ依頼の中で書かせて点検までさせるより、
観点を決めた別の依頼にしたほうが、見落としに気づきやすくなります。

作業ごとにモデルを指定する、Cursorの公式の方法はどれですか。

- A Rulesのfrontmatterにmodelを書く
- B サブエージェントのファイルにmodelを書く
- C AGENTS.mdにモデル名を書く
- D 依頼文の先頭にモデル名を書く

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。

正解はB。モデルの指定はサブエージェントのmodel

B

サブエージェントのファイルにmodelを書く

.cursor/agents/のファイルの先頭に、model: composer-2.5のように書きます。

A

Rulesのfrontmatterにmodelを書く

Rulesの先頭に書けるのはdescription、 globs、 alwaysApplyの3つだけです。

C

AGENTS.mdにモデル名を書く

AGENTS.mdはfrontmatterなしの指示書で、モデルは指定できません。

D

依頼文の先頭にモデル名を書く

依頼文に書いても、使われるモデルはモデルの欄で選んだものです。

出典: cursor.com/docs/subagents、cursor.com/docs/rules (2026年10月5日確認)

05

質疑応答と閉会

利用状況の再確認と質疑応答
[10min]

研修の最初にメモした割合との差を計算してください

終了時の割合

—

開始時の割合

=

4時間で使った割合

1

Ctrl+Shift+Jで
Cursor Settingsを開きます

2

Plan & Usageを押します

3

Included Usageの割合を
読み取ります

講師は、4時間の演習をAPIの料金に換算して5ドル程度と見込んでいます

持ち帰るもの

成果物は配布フォルダの中に残ります

成果物	場所
保管棚を足した在庫管理のコード	inventory-app
自分で書いたRules	inventory-app/.cursor/rules
調べ物用のサブエージェント	inventory-app/.cursor/agents/code-searcher.md
書き直した受注金額の計算とテスト	legacy-order
演習のメモと利用状況の記録	自分のメモ

職場で使う前に確かめること

業務のコードを入れる前に、社内の担当部署に確認します



研修で使ったアカウントとプランを業務に使ってよいか

社内の規程と研修事務局の案内に従います



プライバシーモードの設定

設定の場所と既定値は、公式文書の間で記述が分かれています



社内で許可されたモデル

使ってよいモデルの一覧を社内で確かめます

どれもここで決めず、社内の担当部署に確認してください

資料	URL
Rules	https://cursor.com/docs/rules
サブエージェント	https://cursor.com/docs/subagents
モデルと料金	https://cursor.com/docs/models-and-pricing
キーボードショートカット	https://cursor.com/help/customization/keyboard-shortcuts
教材サイト	https://1023cursor.give-app.net

日本語のドキュメントはcursor.com/ja/で読めます。

気になったことを何でもチャットへどうぞ

質疑応答

質問の例

業務のどの作業から使い始めるとよいですか

Teamsに切り替えると何が変わりますか



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