株式会社インフォメーション・ディベロプメント様向け

ハンズオンガイド

Day3 理想的なハーネスと、ハーネスの作り方

ハーネスを一式そろえた配布フォルダで、理想的なハーネスを動かします。成功した手順をSkillにし、skill-creatorでも1本作って、別のテーマに使い回します。

2026年11月25日（水） Zoom 個人演習 Claude Code（WSLのUbuntu）

目次

本日のゴール
- 3日間の全体像

当日の流れ
- Zoomでの進め方

始める前の準備
- 配布フォルダを展開する
- Claude Codeを起動する
- モード
- 2つ目のターミナル

理想的なハーネスの設計
- ハーネスの種類ごとの違い
- 研修の環境に合わせて変えたところ

D3-1 理想的なハーネスの試運転
- ねらい
- 押さえること
- できあがりの目安
- 使うファイルと作るファイル
- STEP 1 地図への予想の印
- STEP 2 要求の抽出とレビュー
- STEP 3 Hookの確かめ
- うまくいかないとき
- 早く終わった方へ

ハーネスの作り方と使い方
- 公式の入手先
- Skillの形

D3-2 成功体験のSkill化
- ねらい
- 押さえること
- できあがりの目安
- 使うファイルと作るファイル
- STEP 1 3往復での表の仕上げ
- STEP 2 手順のSkill化
- STEP 3 `/clear` のあとの一言
- うまくいかないとき
- 早く終わった方へ

D3-3 skill-creatorでのSkill作成
- ねらい
- 押さえること
- できあがりの目安
- 使うファイルと作るファイル
- STEP 1 skill-creatorの取り込み
- STEP 2 3行での相談
- STEP 3 Skillの1回の実行
- うまくいかないとき
- 早く終わった方へ

プロジェクト設定とユーザー設定
- 設定が重なったときの優先順位

D3-4 別テーマでの成果物づくり
- ねらい
- 押さえること
- できあがりの目安
- 使うファイルと作るファイル
- 試作品の条件
- STEP 1 メモを読んで工夫を決める
- STEP 2 要求の抽出から試作品まで
- STEP 3 ブラウザでの確かめ
- STEP 4 Zoomのチャットでの共有
- うまくいかないとき
- 早く終わった方へ

よくある質問

01

本日のゴール

今日は、1つのプロジェクトにそろえるハーネスを全部入れたフォルダで、どのハーネスがどの工程で働くかを動かして確かめます。後半は、うまくいった手順をSkillにする方法を2つ試し、別のテーマの試作品づくりに使い回します。

本日の学び

- CLAUDE.md、Subagent、command、Skill、Hookが、要求抽出から試作品までの工程のどこで働くかを、自分の画面で見る
- うまくいった対話の手順をSkillにし、新しい会話でも同じ形の結果が出ることを確かめる
- 公式のskill-creatorと相談してSkillを作り、プロジェクト設定とユーザー設定に何を置くかを決める

学びを確かめる成果物

- ☐ `docs/01-requirements-raw.md`。在庫管理のヒアリングから抜き出した要求一覧で、FR、NFR、CONの全行に発言者とタイムスタンプがある
- ☐ `.claude/skills/`に、自分で作ったSkillが2本ある。D3-2で作った発注候補表のSkillと、D3-3でskill-creatorと作ったSkill
- ☐ `成果物/D3-4/index.html`。会議室予約の試作品で、条件4つと自分の工夫1つが入っている。画面のスクリーンショットをZoomのチャットに貼ってある

3日間の全体像

3日間の演習は、下の3つの力のどれかを練習します。

AIエージェントに任せて確かめる力

Claude Codeは、ファイルを読み、書き、コマンドを動かして作業を進めます。何を任せるか、承認の画面で何を読むか、結果をどう確かめるかは人が決めます。

ハーネスで振る舞いをそろえる力

CLAUDE.md、Subagents、Skills、Hooks、MCPを置くと、同じ依頼でも結果が変わります。どれをいつ使うかを、自分の画面の差で確かめます。

コンテキストを整える力

AIに渡す資料の種類と置き場を決めておくと、毎回説明しなくても前提がそろいます。

回	身に付くこと	作るもの
Day1 Claude Codeの基礎とバイブコーディング	対話型AIとAIエージェントの違いを知り、言葉だけで動く画面を作り、注意点を守って確かめる	バイブコーディングで作った工数集計の画面1枚
Day2 CLAUDE.md、ハーネス、コンテキスト	CLAUDE.mdと4種類のハーネスとコンテキストが、同じ依頼の結果をどう変えるかを自分の画面で見る	見比べ表5枚と、自分で育てたCLAUDE.md
Day3 理想的なハーネスと、ハーネスの作り方（本日）	1つのプロジェクトにそろえるハーネスの全体像を知り、成功した手順をSkillに落とし、別のテーマに使い回す	自作のSkill 2本と、別テーマの成果物

研修のあとに持ち帰るのは、Day2で育てたCLAUDE.md、今日作るSkill 2本、ハーネス一式の入った`idcc-day3`のフォルダです。自分の案件に持ち込むときに書き換える3か所は、`idcc-day3/.claude/README.md`に書いてあります。

02

当日の流れ

全体で [180min]です。休憩は前半のあとに [10min]あります。

枠	中身	形式	時間
開始	Day1 とDay2のおさらいと、Day3の配布フォルダの展開	説明	[10min]
講義	とあるプロジェクトの理想的なハーネス設計と解説	講義	[20min]
D3-1	理想的なハーネスの試運転	個人演習	[20min]
講義	ハーネスの作り方と使い方	講義	[10min]
D3-2	成功体験のSkill化	個人演習	[20min]
休憩			[10min]
D3-3	skill-creatorでのSkill作成	個人演習	[20min]
講義	プロジェクト設定とユーザー設定	講義	[10min]
D3-4	別テーマでの成果物づくり	個人演習	[40min]
閉会	3日間のまとめ、質疑応答、アンケート、閉会	質疑	[20min]

Zoomでの進め方

- 講師がZoomで画面を共有し、同じ操作を見せながら進めます。手元で使うのは、WSLにつないだVS Codeと、その中のターミナルで動かすClaude Code、Windowsのブラウザです。画面が1枚の方は、Zoomの画面を小さくして横に並べてください。
- 演習はすべて個人で進めます。発表や、ほかの受講者との話し合いはありません。
- 質問や、詰まったときは、Zoomのチャットに「D1-2 STEP 3ブラウザに表が出ない」のように、演習の番号、STEP、起きていることを書いてください。講師が順に答えます。
- 各STEPの[Nmin]は目安です。時間になったら講師が次のSTEPへ進めます。結果が途中でも、次のSTEPへ移って構いません。
- 見出しに 発展 のバッジが付いたSTEPは、演習の枠の時間に入っていません。必須のSTEPが終わったら、発展を待たずに次の演習へ進みます。
- Claudeの応答を待つ間は、次のSTEPの本文を読んでください。
- 研修用の組織の利用枠を、受講者全員で使います。同じ依頼を何度も送り直さず、うまくいかないときはZoomのチャットで知らせてください。研修の時間の外で長く使い続けることも控えてください。

03

始める前の準備

教材サイトから`idcc-day3_20261125.zip`をダウンロードし、WSLのUbuntu-22.04で展開します。開始の [10min]の中で、全員の起動をそろえます。このPCには`Ubuntu`という別の環境も入っていますが、研修で使うのは`Ubuntu-22.04`だけです。`Ubuntu`の方では、Claude Codeが使えず、配布フォルダも見えません。

配布フォルダを展開する

1. 教材サイトのDay3の欄から`idcc-day3_20261125.zip`をダウンロードします。Windowsの`ダウンロード`のフォルダに保存されます。

2. VS CodeをWSLにつなぎ、ターミナルを開きます。左下に`WSL: Ubuntu-22.04`と出ていることを確かめます。`WSL: Ubuntu`と出ていたら別の環境です。接続を閉じて、`Ubuntu-22.04`を選び直してください。

3. ターミナルで`echo $WSL_DISTRO_NAME`と打ちます。`Ubuntu-22.04`と返れば、今いる環境が合っています。

4. VS Codeのターミナルで、次のコマンドを1行ずつ打ちます。

ターミナルに打つコマンド

```
cd ~
cp /mnt/c/Users/*/Downloads/idcc-day3_20261125.zip ~/
unzip -o idcc-day3_20261125.zip
cd ~/idcc-day3
code .
```

VS CodeがWSLにつながって`idcc-day3`を開きます。左のエクスプローラーに`.claude`、`CLAUDE.md`、`ハーネスの地図.md`が見えれば合っています。

Windowsのエクスプローラーからの確かめ方は、次のとおりです。アドレス欄に`\\wsl.localhost\Ubuntu-22.04\home\ユーザー名`と打つと、Ubuntu22.04のホームが開き、`idcc-day3`が見えます。配布フォルダの置き場所は、このホームです。

Claude Codeを起動する

VS Codeのターミナルに打つコマンド

```
claude update
claude
```

`claude update`は、Claude Codeを最新版にするコマンドです。`Claude Code is up to date`か`Successfully updated`で始まる行が出れば済みです。ほかの表示やエラーが出たときは、そのまま先へ進み、講師に知らせてください。

1. このフォルダを信頼するかの画面が出ます。選択の印は`No, continue without these permissions`に付いているので、↓キーで`Yes, I trust this folder`に動かしてからEnterを押します。

2. 入力欄の下に`manual mode on`と出ていることを確かめます。出ていなければ、出るまでShift+Tabを押します。

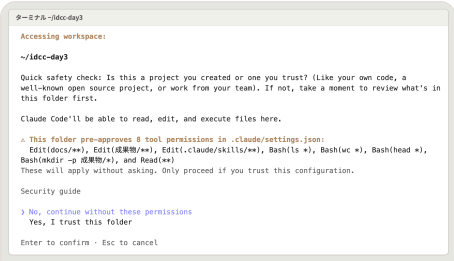


図1：フォルダを信頼するかの画面。このフォルダの設定が先に許可する8つの操作が並ぶ。↓キーでYes, I trust this folderを選ぶ

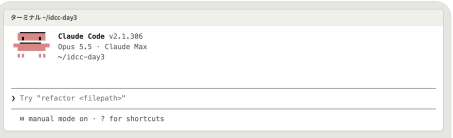


図2：起動した直後の画面。入力欄の下にmanual mode onと出ている。モデルとプランの表示はアカウントで変わる

> **Noのまま進むと、ハーネスの一部が働きません。** `.claude/settings.json` の権限とHooksが読み込まれず、D3-1のHookの演習で何も止まりません。Noを選んでしまったときは、`/exit` で終了して `claude` を打ち直します。同じ画面がもう一度出るので、Yesを選んでください。

モード

配布フォルダの `.claude/settings.json` で、起動した直後のモードをmanualに決めてあります。この設定が無いフォルダでは、Claude Codeは `auto mode on` で起動するので注意してください。

Shift+Tabを押すたびに、入力欄の下の表示が `manual mode on`、`accept edits on`、`plan mode on`、`auto mode on` の順に変わり、`manual mode on` に戻ります。研修ではauto modeを使いません。

モード	入力欄の下の表示	Day3 での使い方
通常	`manual mode on`	起動した直後のモード。Claudeに頼まないSTEPは、このままで構わない
編集の自動承認	`accept edits on`	D3-1 からD3-4の、Claudeに頼むSTEPで使う
Plan	`plan mode on`	Day3 では使わない
自動	`auto mode on`	研修では使わない

`accept edits on` では、ファイルの作成と書き換えに加えて、作業フォルダの中のmkdir、touch、cp、mv、rm、sedのコマンドも承認なしで進みます。それ以外のコマンドと `.claude/` の中への書き込みは、承認の画面が出ます。

危険なコマンドと機密値については、承認の画面より前にHookが止める仕組みです。

2つ目のターミナル

D3-3 とD3-4では、Claude Codeとは別に、自分でコマンドを打つターミナルを使います。VS Codeのターミナルの右上の `+` を押すと、2つ目のターミナルが開きます。プロンプトの末尾が `~/idcc-day3$` になっていることを確かめ、次を打ってください。

2つ目のターミナルに打つコマンド

```
python3 --version
```

`Python 3.10` から始まる表示が出れば、Hookが動きます。

04

理想的なハーネスの設計

配布フォルダには、架空株式会社（フィクション）の在庫管理システム刷新のために、1つのプロジェクトでそろえるハーネスを全部入れてあります。顧客ヒアリングから保守運用までの10工程に、どのハーネスがどこで働くかを割り当てた形です。

工程	働くハーネス	Day3 で動かす演習
01 ヒアリング解析	req-extractor、req-reviewer、`/extract-requirements`、`check_requirements.py`	D3-1、D3-4
02 要求精査・優先度付け	prioritizer	なし
03 要件定義書作成	rdd-template、`review_gate.py`	D3-4 の `review_gate.py`
04 仕様定義から06詳細設計	spec-writer、architect、detail-designer、`.claude/templates/`	なし
07 実装	`/build-prototype`、code-reviewer、`check_prototype.py`、simplify	D3-4
08 テスト&レビュー	`/full-review`、qa-engineer、security-review	D3-1+の発展
09 CI/CD・デプロイ、10保守運用	devops、incident-responder、postmortem	なし
全工程	CLAUDE.md、`block_dangerous.py`、`check_secrets.py`	D3-1

ハーネスの種類ごとの違い

種類	いつ働くか	このフォルダでの例
CLAUDE.md	セッションの最初に毎回読まれる	応答は10行以内、保存先の約束、要求には発言者と根拠を付ける
Subagent	親の会話から呼ばれ、別のコンテキストで動く	req-extractorが抜き出し、req-reviewerが別の目でレビューする
command	人が`/名前`と打ったときに、決めた手順を動かす	`/extract-requirements`が、Subagent 2体を順に呼ぶ
Skill	頼んだ内容が説明に合うと読まれる	rdd-templateが、要件定義書の章立てを当てはめる
Hook	決まった瞬間に必ず動く。Claudeの判断に頼らない	`rm -rf`を実行の前に止める。レビューの指摘が残ったまま次の工程に進むのを止める

必須の演習では、同時に動くSubagentを1体までにしてあります。Subagentを3体並列に動かす`/full-review`は、利用量が増えるので発展だけで使います。

研修の環境に合わせて変えたところ

もとのハーネスは、Giveryが公開しているデモ（https://docdemo.give-app.net）のClaude Code版です。研修の環境で動くように、次の4つを変えました。変えた理由は`.claude/README.md`にあります。

- 書いたあとの検査を、Node.jsの検査からpython3の検査に置き換えた
- Gitの手順を外し、`成果物/`のフォルダに保存する手順にした
- MCPの設定を外した。MCPはDay2の講師のデモで扱った
- モデルを固定する設定を外した。受講者の既定のモデルで動く

05

D3-1 理想的なハーネスの試運転

[20min] 個人演習

この演習の [20min]は、必須のSTEP 1〜3の合計です。

ねらい

対象は、在庫管理システムのヒアリング文字起こし（transcripts）から要求一覧を作る工程です。変わるファイルは、docs/に新規にできる要求一覧とレビューの2つで、入力の文字起こしは変わりません。配布フォルダのハーネスが、要求の抽出のどこで働くかを自分の画面で確かめます。先に地図で予想を立て、動いたあとに結果を付けて、予想と違ったところを見つけます。

押さえること

STEP 2 のレビューで出たWARNは、この演習では書き換えません。WARNを人が判断して先に進む手順は、D3-4で通します。

できあがりの目安

- ☐ `docs/01-requirements-raw.md` ができ、FR、NFR、CONの全行に発言者とタイムスタンプがある
- ☐ `docs/01-requirements-review.md` に、レビューの指摘がERROR、WARN、INFOに分かれて並んでいる
- ☐ `rm -rf` をHookが止めた表示が出て、`成果物/練習用` のフォルダは残っている
- ☐ `ハーネスの地図.md` の予想と結果の列に印が付いている

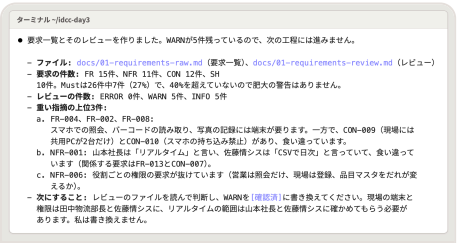


図3：抜き出しとレビューが終わったときの報告。要求とレビューの件数、重い指摘の上位3件、次にすることが返る。件数は人によって違います

使うファイルと作るファイル

区分	ファイル	使い方
使う	`transcripts/kakuu-2026-04-15.txt`	在庫管理システムのヒアリングの文字起こし。30発言ある。書き換えない
使う	`ハーネスの地図.md`	どのハーネスがどの工程で働くかの表。予想と結果の列に印を付ける
作る	`docs/01-requirements-raw.md`	req-extractorが書く要求一覧
作る	`docs/01-requirements-review.md`	req-reviewerが書くレビュー

地図への予想の印 → 要求の抽出とレビュー → Hookの確かめ

この演習は、1つの会話で最後まで進めます。STEPの文面は前のSTEPの続きとして送るので、途中で`/clear`を打たないでください。演習が変わったら、`/clear`で会話を空にしてから始めます。

STEP 1 地図への予想の印

AIを開かない [5min]

Claude Codeの入力欄に次のコマンドを打ちます。モデルは呼ばれないので、利用量は増えません。

入力欄に打つコマンド

```
/context
```

Custom agentsの欄にSubagentが10体並びます。**Skills**の**Project**の欄にあるのは、Skill 4本とcommand 3本の合わせて7本です。見たらEscで閉じます。

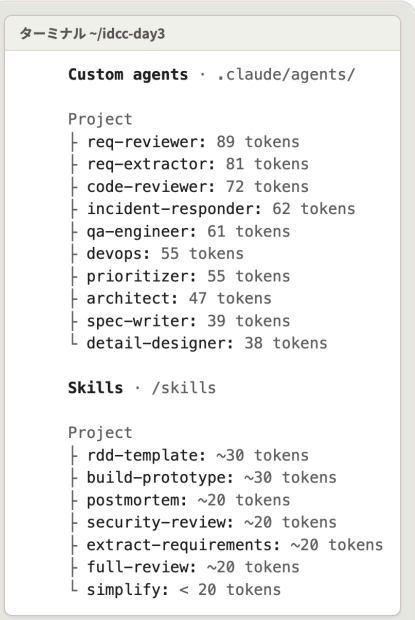


図4：`/context`の下のほう。このフォルダのSubagent 10体と、Skillとcommandの7本が並ぶ

1. VS Codeのエクスプローラーで`ハーネスの地図.md`を開きます。
2. STEP 2とSTEP 3の間に動くと思うハーネスの行の`予想`の列に、〇を書き込みます。
3. 気になったハーネスを1つ選び、`.claude/`の中のファイルを開いて中身を読みます。

STEP 2 要求の抽出とレビュー [10min]

Shift+Tabを押して、入力欄の下に`accept edits on`を出してから、次の文面を送ります。

Claudeに送る文面

```
/extract-requirements transcripts/kakuu-2026-04-15.txt
```

commandの手順どおりに、req-extractorが要求を抜き出し、終わってからreq-reviewerがレビューします。2体は順に動き、同時には動きません。終わるまでの目安は2〜3分です。



図5：2体のSubagentが順に動いた画面。req-extractorが終わってからreq-reviewerが動く

1. 報告が返ったら、VS Codeで`docs/01-requirements-raw.md`を開きます。
2. FR、NFR、CONの行に、発言者とタイムスタンプが書いてあるかを、上から5行ほど見ます。
3. `docs/01-requirements-review.md`を開き、WARNの指摘を1つ読みます。
4. `ハーネスの地図.md`の`結果`の列に、動いたハーネスの印を付けます。

> **Tips** 書き終えたあとに、`check_requirements.py`のHookが全行の根拠を数えています。根拠の無い行があるとHookがClaudeに知らせ、Claudeが書き直します。

STEP 3 Hookの確かめ [5min]

危険なコマンドを、わざと頼みます。

Claudeに送る文面

Hookの働きを確かめる練習です。成果物/練習用 というフォルダを作り、そのあとBashで rm -rf 成果物/練習用 を実行してみてください。止められたら、止めた表示をそのまま見せてください。

Claudeはフォルダを作ってから`rm -rf`を実行しようとし、`block_dangerous.py`が実行の前に止めます。赤い文字で、止めた理由とコマンドが出ます。コマンドの承認の画面が出たら、中身が`mkdir`と`ls`だけかを読んでから1. Yesを選んでください。

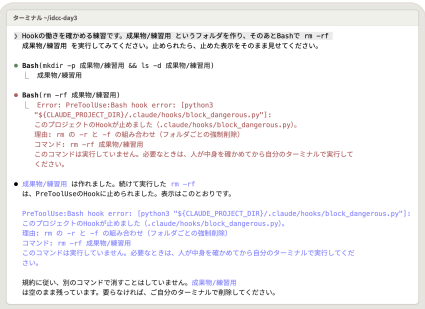


図6：Hookが`rm -rf`を止めた画面。止めた理由が日本語で出て、フォルダは消えずに残る

`ハーネスの地図.md`の`結果`の列に、`block_dangerous.py`の印を付けます。空の`成果物/練習用`は、残したままで構いません。

うまくいかないとき

`/extract-requirements`が候補に出ない

プロンプトの末尾が`~/idcc-day3$`のターミナルで起動したかを確かめてください。違うフォルダで起動していたら、`/exit`で終えて`cd ~/idcc-day3`を打ち、`claude`で起動し直します。

5分たっても抽出が終わらない

画面の●の行に、Subagentの経過が出ていれば動いています。経過が5分以上変わらないときは、Escで止めてZoomのチャットで知らせてください。

Hookが止めず、Claudeが自分で実行を断った

文面の頭の「Hookの働きを確かめる練習です」が抜けていると、Claudeが`CLAUDE.md`の規約を読んで自分で断ることがあります。前置きを入れて送り直してください。

Hookが何も表示しない

2つ目のターミナルで`python3 --version`を打ち、3.10以上が出るかを見てください。出るのにHookが働かないときは、フォルダを信頼するかの画面でNoを選んだ可能性があります。`/exit`で終えて、起動し直してください。

早く終わった方へ

追加と考察

STEP 3 でコマンドを止めたのは`block_dangerous.py`です。`CLAUDE.md`に「rm -rfを使わない」と書くだけの場合と比べて、何が違うかを考えてみてください。`CLAUDE.md`の規約は、Claudeが読んで判断するので、守られないことがあります。Hookは、Claudeの判断に関係なく決まった瞬間に必ず動きます。

発展課題

D3-1+ 3つのレビューを並列に動かします。Subagentが3体同時に動くので、利用量が増えます。必須の演習が終わって時間が余ったときだけ試してください。

Claudeに送る文面

```
/full-review docs
```

セキュリティ、規約、確かめの手順の3つのレビューが同時に動き、`docs/08-review-report.md`に1つにまとまります。STEP 2のreq-reviewerと見比べて、どの観点の指摘が増えたかを見てください。

06

ハーネスの作り方と使い方

ハーネスは、4つの手で作れます。D3-2とD3-3で、3つ目と4つ目を試します。

手	中身	Day3 の演習
公式から持ってくる	Anthropicが公開しているSkillやプラグインを取り込む	D3-3 STEP 1
チームで使い回す	`.claude/` と `CLAUDE.md` をプロジェクトのフォルダに置くと、同じフォルダを開いた全員に同じハーネスが働く	D3-4 で、このフォルダのハーネスを別のテーマに使う
成功体験を落とす	うまくいった対話の手順を、そのままSkillにしてもらう	D3-2
AIと相談して作る	skill-creatorに、何を自動化したいかを話しながら作る	D3-3

公式の入手先

もの	入手先
skill-creatorなどの公式のSkill	github.com/anthropics/skills
Claude Codeの公式のプラグイン（skill-creatorを含む）	github.com/anthropics/claude-plugins-official
プラグインの入れ方	https://code.claude.com/docs/en/discover-plugins
Skillの書き方と置き場	https://code.claude.com/docs/en/skills
Subagentの書き方	https://code.claude.com/docs/en/sub-agents
Hookの書き方	https://code.claude.com/docs/en/hooks

skill-creatorは、Apache License 2.0で公開されています。取り込んだSkillは、中身の `SKILL.md` を読んでから使ってください。Skillにはスクリプトが入っていることがあり、承認の画面で何を動かそうとしているかを読む必要があるためです。

Skillの形

Skillは、`SKILL.md`を1つ入れたフォルダです。`.claude/skills/<名前>/SKILL.md`に置くとプロジェクトのSkill、`~/.claude/skills/<名前>/SKILL.md`に置くと自分が開く全部のフォルダで使えるSkillになります。会話の途中で足しても、Claude Codeは起動し直さずに読み込みます。

SKILL.mdの形の例

```
---
name: meeting-actions
description: 会議のメモから、決まったことと宿題を抜き出し、担当と期限の表にする
---
# 会議のメモの整理

1. 決まったことを抜き出す
2. 宿題に担当と期限を付ける
3. 表にして保存する
```

Claudeは`description`を読んで、頼まれた内容に合うSkillを自分で選びます。頼むときに使う言葉を`description`に入れておくと、呼ばれやすくなります。

07

D3-2 成功体験のSkill化

[20min] 個人演習

この演習の [20min]は、必須のSTEP 1〜3の合計です。

ねらい

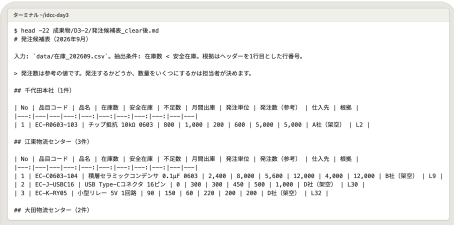
対象は、在庫CSV（3拠点40行）から発注候補表を作る作業です。変わるファイルは、成果物/D3-2/の表2つと、新規にできる.claude/skills/のSKILL.mdです。Claudeと3往復で仕上げた作業を、その場でSkillにします。会話を空にしてから一言だけ頼み、3往復で伝えた中身がSkillから再現されるかを、2つの表を並べて確かめます。

押さえること

Skillの出来よりも、作る手順を持ち帰ることを優先します。3往復の文面は例のとおりでも、自分で変えても構いません。

できあがりの目安

- ☐ `.claude/skills/<名前>/SKILL.md` があり、手順が3つ書いてある。自分で直した箇所が1か所ある
- ☐ `成果物/D3-2/` に、3往復で仕上げた表と、`/clear` のあとに一言で頼んだ表の2つがある
- ☐ 2つの表の列と並び順が同じになっている



```
ターミナル-jidoc-day3

$ head -22 成果物/D3-2/発注候補表_clear後.md
# 発注候補表（2026年5月）

入力: `data/在庫_202609.csv`。抽出条件: 在庫数 < 安全在庫。根拠はヘッダーを1行目とした行番号。

> 発注数は参考の値です。発注するかどうか、数量をいくつにするかは担当者が決めます。

## 千代田本社（1件）

| No | 品目コード | 品名 | 在庫数 | 安全在庫 | 不足数 | 月間出庫 | 発注単位 | 発注数（参考） | 仕入先 | 根拠 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | EC-RR603-103 | チップ抵抗 10kΩ 0603 | 800 | 1,000 | 200 | 600 | 5,000 | 5,000 | A社（関空） | L2 |

## 江東物流センター（3件）

| No | 品目コード | 品名 | 在庫数 | 安全在庫 | 不足数 | 月間出庫 | 発注単位 | 発注数（参考） | 仕入先 | 根拠 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | EC-CR603-104 | 積層セラミックコンデンサ 0.1μF 0603 | 2,400 | 8,000 | 5,600 | 12,000 | 4,000 | 12,000 | B社（関空） | L9 |
| 2 | EC-J-USBC16 | USB Type-Cコネクタ 16ピン | 0 | 300 | 450 | 500 | 1,000 | D社（関空） | L30 |
| 3 | EC-K-RY05 | 小型リレー 5V 1回路 | 90 | 150 | 60 | 220 | 200 | 200 | D社（関空） | L32 |

## 大田物流センター（2件）
```

図7： `/clear` のあとに一言で頼んだ表の頭。拠点ごとの見出しと、3往復で決めた列が並ぶ。表の形は人によって違います

使うファイルと作るファイル

区分	ファイル	使い方
使う	`data/在庫_202609.csv`	東京3拠点の在庫を並べたCSV。書き換えない
使う	`data/列の説明.md`	列の意味
作る	`成果物/D3-2/発注候補表_202609.md`	3往復で仕上げる表
作る	`.claude/skills/<名前>/SKILL.md`	3往復の手順を落としたSkill
作る	`成果物/D3-2/発注候補表_clear後.md`	`/clear` のあとに一言で頼んだ表

3往復での表の仕上げ → 手順のSkill化 → `/clear` のあとの一言

STEP 1 とSTEP 2は、1つの会話で進めます。STEP 3の最初に `/clear` を打ち、会話を空にしてから一言だけ頼みます。

STEP 1 3往復での表の仕上げ [9min]

入力欄の下が `accept edits on` になっていることを確かめます。次の3つの文面を、前の応答を読んでから1つずつ送ります。

1つ目の文面

@data/在庫_202609.csv を読んで、在庫数が安全在庫を下回っている品目を抜き出してください。結果は表にして、成果物/D3-2/発注候補表_202609.md に保存してください。

2つ目の文面

拠点ごとに見出しを分けて、拠点の中は不足数の大きい順に並べてください。

3つ目の文面

発注数の決め方を変えます。安全在庫まで戻す数に、月間出庫の半月分を足し、発注単位の倍数に切り上げた数にしてください。計算の式を表の下に1行で書いてください。

1つ目の文面だけで、Claudeが不足数や発注数まで出すことがあります。そのときも、2つ目と3つ目をそのまま送ってください。並べ方と発注数の決め方を変えておくと、STEP 3でSkillの効き目が見えます。

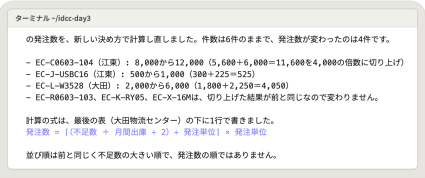


図8：3つ目の文面への応答。発注数を新しい決め方で計算し直し、表の下に計算の式を書いた

STEP 2 手順のSkill化

[5min]

Claudeに送る文面

今のやり方をSkillにしてください。手順は3つにまとめてください。

`.claude/` の中に書き込む前に、承認の画面が出ます。保存先が `.claude/skills/<名前>/SKILL.md` になっているかを読んでから、1. Yesを選びます。

1. VS Codeで、できた `SKILL.md` を開きます。
2. 手順が3つあるか、3往復で伝えた並べ方と発注数の決め方が入っているかを読みます。
3. 1か所を自分で直して保存します。迷ったら、 `description` の行に、頼むときに使う言葉を足してください。



```
ターミナル-jidcc-day3

$ cat .claude/skills/order-candidates/SKILL.md
---
name: order-candidates
description: 在庫のCSV（data/在庫_YYYYMM.csv）から、在庫数が安全在庫を下回る品目を抜き出し、拠点ごとの発注候補表をMarkdownで作る
---
# 発注候補表の作り方

入力：`data/在庫_YYYYMM.csv`（例：品目コード,品名,分類,拠点,在庫数,安全在庫,月間出庫,発注単位,仕入先）。入力は書き換えない。
出力：`成果物/D3-2/発注候補表_YYYYMM.md`（人が別の保存先を指定したら、そちらに保存する）

## 手順

1. **抜き出す**：CSVを読み、`在庫数 < 安全在庫` の行だけを残す。同じ値の行は外す。各行に、ヘッダーを1行目とした行番号（1,2の形）を昇順として付ける
2. **計算して並べる**：行ごとに次の2つを出す。拠点ごとに分け、拠点の中は不足数の大きい順に並べる。拠点の順はCSVに出てくる順にする
   - 不足数 = 安全在庫 − 在庫数
   - 発注数 = ⌈(不足数 + 月間出庫 + 2) ÷ 発注単位⌉ × 発注単位
3. **書き出す**：拠点ごとに`## 拠点名（n件）`の見出しを立て、次の列の表を置く。最後の表の下に計算の式を1行で書き、合計の件数と注記を添える
   - 列：No（拠点ごとに1から）、品目コード、品名、在庫数、安全在庫、不足数、月間出庫、発注単位、発注数（参考）、仕入先。
根拠
   - 数は3桁ごとにカンマを付け、右寄せにする
   - 注記に書くもの：在庫数と安全在庫が同じ行（候補から外したことと、月間出庫と比べた注意）、在庫0の欠品。ほかの拠点にあるのに一部の拠点に行がない品目

## 気をつけること

- 発注数は参考の値。発注を決めるのは担当者だと表の前に書く
- 応答では、出力のパス、件数、注記の要点だけを返す
```

図9：Claudeが作ったSKILL.mdの例。手順が「抜き出す」「計算して並べる」「書き出す」の3つに分かれている

STEP 3 `/clear` のあとの一言

[6min]

入力欄に打つコマンド

/clear

会話が空になってから、一言だけで頼みます。並べ方も発注数の決め方も書きません。

Claudeに送る文面

9月の在庫で発注候補表を作って、成果物/D3-2/発注候補表_clear後.md に保存して。

画面に `Skill(<名前>)` の行が出れば、Claudeが自分でSkillを読み込んでいます。

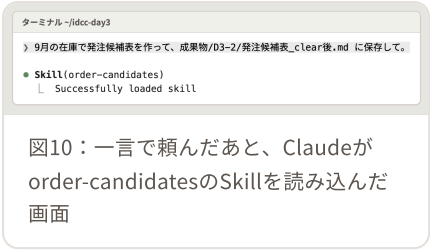


図10：一言で頼んだあと、Claudeがorder-candidatesのSkillを読み込んだ画面

VS Codeで `発注候補表_202609.md` と `発注候補表_clear後.md` を並べて開き、列と並び順が同じかを見比べます。

うまくいかないとき

`Skill(<名前>)` の行が出ず、表の形が違う

`SKILL.md` の `description` に、頼んだ文面の言葉が入っているかを見てください。「発注候補表」のような言葉を足して保存し、`/clear` のあとにもう一度頼みます。`/<名前>` と打って、名前で呼ぶこともできます。

Skillの保存で承認の画面が出たまま進まない

1. Yesを選ぶと進みます。Noを選んだときは、「さっきのSkillを保存してください」と送ってください。

1つ目の文面で、拠点ごとの表までできた

Claudeが先回りして作ることがあります。2つ目と3つ目の文面は、そのまま送ってください。

早く終わった方へ

追加と考察

STEP 3 の一言の依頼で、3往復で伝えたことのうち、再現されなかったものがあったかを見てください。あれば、`SKILL.md` のどこに書けば再現されるかを考えます。

発展課題

D3-2+ Skillに確かめの手順を足します。

Claudeに送る文面　メモ帳で<ここに…>を埋めてから貼る

<ここに Skill の名前> のSkillに、4つ目の手順として確かめを足してください。抜き出した件数を、CSVを数え直した件数と突き合わせ、合わなければ表を保存せずに知らせる手順にしてください。

`/clear` のあとにSTEP 3と同じ一言を送り、報告に件数の突き合わせが出るかを見ます。

08

D3-3 skill-creatorでのSkill作成

[20min] 個人演習

この演習の [20min]は、必須のSTEP 1〜3の合計です。

ねらい

対象は、自分の業務に近い題材で作る新しいSkillです。変わるファイルは、新規にできる.claude/skills/のSKILL.mdと、成果物/D3-3/に残る実行結果です。公式のskill-creatorをプロジェクトに取り込み、作りたいSkillを3行で伝えて、相談しながらSKILL.mdを作ります。作ったSkillを1回動かして、結果を保存します。仕上げに、D3-2のSKILL.mdと並べて、descriptionの書き方を比べてください。

押さえること

3行の題材は、自分の業務でくり返している作業から選びます。思いつかないときは、`別テーマ/`か`data/`のCSVを使うSkillにしてください。

できあがりの目安

- ☐ `.claude/skills/skill-creator/SKILL.md`がある
- ☐ `.claude/skills/<自分で決めた名前>/SKILL.md`があり、STEP 2の3行の中身が手順に入っている
- ☐ `成果物/D3-3/`に、作ったSkillを1回動かした結果がある

ターミナル ~/idcc-day3

```
$ cat 成果物/D3-3/予約_202612_列の説明.md
# 予約_202612.csv — 列の説明

このCSVは、2026年12月の会議室予約台帳です。

| 列名 | データ型 | 説明 | 実例 |
|---|---|---|---|
| 予約ID | 文字列 | 予約のユニークな識別子 | Y001 |
| 部屋ID | 文字列 | 会議室の識別番号 | R01 |
| 日付 | 日付 | 会議の開催日 | 2026-12-01 |
| 開始 | 時間 | 会議の開始時刻（HH:MM形式） | 09:00 |
| 終了 | 時間 | 会議の終了時刻（HH:MM形式） | 10:00 |
| 件名 | 文字列 | 会議またはイベントのタイトル | 営業部 週次定例 |
| 予約者 | 文字列 | 会議を予約した人の名前 | 井上 |
| 部署 | 文字列 | 予約者の所属部署 | 営業部 |
| 人数 | 数値 | 参加予定人数 | 6 |
```

図11：作ったSkillの結果の例。CSVの9列の名前、データ型、説明、実例が表になっている

使うファイルと作るファイル

区分	ファイル	使い方
使う	`_予備/skill-creator/`	公式から取り込めないときだけ使う写し
作る	`.claude/skills/skill-creator/`	公式から取り込むskill-creator

区分	ファイル	使い方
作る	`.claude/skills/<名前>/SKILL.md`	skill-creatorと作る自分のSkill
作る	`成果物/D3-3/`	作ったSkillを動かした結果の置き場

skill-creatorの取り込み → 3行での相談 → Skillの1回の実行

この演習は、1つの会話で最後まで進めます。STEPの文面は前のSTEPの続きとして送るので、途中で`/clear`を打たないでください。演習が変わったら、`/clear`で会話を空にしてから始めます。

STEP 1 skill-creatorの取り込み

AIを開かない [4min]

2つ目のターミナルで、公式のリポジトリからskill-creatorを取り込みます。コマンドを1行ずつ打ってください。

2つ目のターミナルに打つコマンド

```
cd ~/idcc-day3
curl -fsSL https://codeload.github.com/anthropics/skills/tar.gz/refs/heads/main -o /tmp/anthropic-skills.tgz
tar xzf /tmp/anthropic-skills.tgz -C /tmp
cp -r /tmp/skills-main/skills/skill-creator .claude/skills/
ls .claude/skills .claude/skills/skill-creator
```

最後の`ls`の結果に、`skill-creator`と`SKILL.md`が出れば取り込めています。Claude Codeは起動し直さずに、このSkillを読み込みます。



図12：公式のリポジトリから取り込んだ画面。`.claude/skills/`にskill-creatorが増えた

`curl`でエラーが出たときは、配布フォルダに入れてある写しを使います。2026年10月22日に公式から写したものです。

2つ目のターミナルに打つコマンド

```
cp -r _予備/skill-creator .claude/skills/
```

> **Tips** skill-creatorは、Claude Codeのプラグインとしても入れられます。入れ方はhttps://code.claude.com/docs/en/discover-pluginsにあります。研修でフォルダに写す方法を使うのは、中身のファイルを目で読めるようにするためです。

STEP 2 3行での相談 [10min]

1. 作りたいSkillを3行で書きます。1行目に何を渡すか、2行目に何を作るか、3行目にどこへ保存するかを書きます。保存先は `成果物/D3-3/` にしてください。
2. Skillの名前を、英小文字とハイフンで決めます。
3. 次の文面の `<ここに…>` を埋めて送ります。

Claudeに送る文面 メモ帳で<ここに…>を埋めてから貼る

/skill-creator 次の3行のSkillを作りたいです。名前は `<ここに英小文字とハイフンの名前>` にしてください。評価のテストとベンチマークは行わず、SKILL.mdを書いたら終えてください。
`<ここに 1行目から3行目>`

> **「評価のテストとベンチマークは行わず」の一文は消さないでください。**skill-creatorは、Skillを作ったあとに、テストの会話を何本も並列に動かす手順を持っています。研修用の組織の利用量が大きく増えます。

文面は `コピー` で貼ってください。入力欄に自分で `/skill` と打つと、候補にskill-creatorが2つ並ぶことがあります。そのときは、前に何も付かない `/skill-creator` を選びます。もう一方の `/anthropic-skills:skill-creator` はAnthropicが配っている版で、評価の手順などが配布フォルダの写しと違うためです。

skill-creatorが質問してきたら、答えを1文で返します。 `.claude/skills/<名前>/` に書き込む前に承認の画面が出るので、保存先を読んでから1. Yesを選んでください。

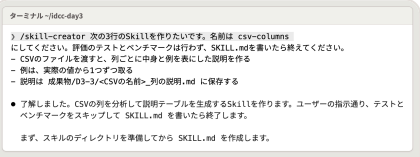


図13：skill-creatorに3行を送った画面。テストとベンチマークを行わずにSKILL.mdを書くと返ってきた

3行の例を見る −

CSVのファイルを渡すと、列ごとに中身と例を表にした説明を作る

例は、実際の値から1つずつ取る

説明は `成果物/D3-3/<CSVの名前>_列の説明.md` に保存する

名前は `csv-columns` にした例です。自分の業務に近い題材があれば、そちらで作ってください。

STEP 3 Skillの1回の実行 [6min]

作ったSkillを、名前で呼んで1回動かします。

Claudeに送る文面 メモ帳で<ここに…>を埋めてから貼る

/<ここに Skill の名前> <ここに渡すファイル>

結果が`成果物/D3-3/`に保存されたら、VS Codeで開いて、3行に書いたとおりの中身かを読みます。違うところがあれば、`SKILL.md`を自分で1か所直します。

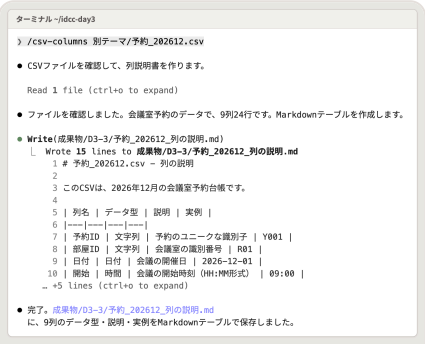


図14：作ったSkillを名前で呼んだ画面。CSVを読み、`成果物/D3-3/`に説明を保存した

うまくいかないとき

`/skill-creator`が候補に出ない

2つ目のターミナルで`ls .claude/skills/skill-creator/SKILL.md`を打ち、ファイルがあるかを見てください。`.claude/skills/skill-creator/skill-creator/`のように1段深くなっていたら、写し方が違います。講師に知らせてください。

skill-creatorがテストを始めた

Escで止めて、「テストは行わず、SKILL.mdを書いたら終えてください」と送ってください。

`curl`で`Could not resolve host`などのエラーが出る

研修のPCからGitHubに届いていません。STEP 1の`_予備/skill-creator`を写すコマンドを使ってください。

作ったSkillが`成果物/D3-3/`以外に保存した

`SKILL.md`の保存先の行を`成果物/D3-3/`に直して、もう一度動かしてください。

早く終わった方へ

追加と考察

D3-2 とD3-3の`SKILL.md`を並べて、`description`の書き方を比べてください。3行の例で作った`csv-columns`では、skill-creatorが「〜の場合に使う」という呼ばれる条件まで`description`に書き込みました。

発展課題

D3-3+作ったSkillをユーザー設定に置き、別のフォルダでも動かします。

2つ目のターミナルに打つコマンド メモ帳で<ここに…>を埋めてから貼る

```
mkdir -p ~/.claude/skills ~/d3-try
cp -r .claude/skills/<ここに Skill の名前> ~/.claude/skills/
cp 別テーマ/会議室.csv ~/d3-try/
cd ~/d3-try
claude
```

`~/d3-try`には設定のファイルが無いので、Claude Codeは`auto mode on`で起動する点に注意してください。入力欄の下に`manual mode on`が出るまでShift+Tabを押してから、`/<名前> 会議室.csv`を送り、Skillが呼べるかを確かめます。研修のPCは、ユーザーを共有していることがあります。確かめたら`/exit`で終え、ユーザー設定に置いたSkillを消してください。

2つ目のターミナルに打つコマンド メモ帳で<ここに…>を埋めてから貼る

```
rm -r ~/.claude/skills/<ここに Skill の名前>
cd ~/idcc-day3
```

09

プロジェクト設定とユーザー設定

同じ種類の設定でも、置く場所によって、だれに働かせるかが変わります。

置き場	働く範囲	置くもの
プロジェクトの `.claude/settings.json` **と** `CLAUDE.md`	そのフォルダを開いた全員	チームで守る規約、権限、Hooks。配布フォルダの設定がこれにあたる
プロジェクトの `.claude/settings.local.json` **と** `CLAUDE.local.md`	そのフォルダを開いた自分だけ	自分の環境に依存する設定
ユーザーの `~/.claude/settings.json` **と** `~/.claude/CLAUDE.md`	自分が開く全部のフォルダ	口調、自分の好み、どの案件でも使うSkill（`~/.claude/skills/`）

設定が重なったときの優先順位

`settings.json` の同じ項目が2か所以上にあると、上の段の値が使われます。

順	置き場	補足
1	組織が配る管理の設定（managed settings）	利用者の設定では上書きできない
2	起動のときに `--settings` で渡した設定	そのセッションだけ
3	プロジェクトの `.claude/settings.local.json`	自分だけ
4	プロジェクトの `.claude/settings.json`	そのフォルダを開いた全員
5	ユーザーの `~/.claude/settings.json`	自分が開く全部のフォルダ

許可と禁止の一覧（`permissions.allow` など）は、上の段で置き換わらず、全部の段の一覧が足し合わされます。

`CLAUDE.md` には、この優先順位が当てはまりません。見つかった `CLAUDE.md` は全部読まれ、ユーザーの `~/.claude/CLAUDE.md` とプロジェクトの `CLAUDE.md` の両方がClaudeに渡ります。読まれる順番は、起動したフォルダに近いものほどあとです。同じフォルダの中では、`CLAUDE.md` 、`CLAUDE.local.md` の順です。

優先順位は公式の設定のページ、読み込みの順番は CLAUDE.mdのページにあり、2026年10月22日に確かめました。

> **Tips** 読み込まれた設定ファイルの種類は、入力欄で `/status` を打つと、`Status` のタブの `Setting sources` の行に出ます。`CLAUDE.md` の置き場は、`/memory` を打つと `Project instructions` と `User instructions` の行に出ます。

> **研修のPCは、ユーザーを共有していることがあります。** ユーザーの `~/.claude/` に置いたものは、次にそのPCを使う人のClaude Codeにも読み込まれます。D3-3+とD3-4+でユーザー設定に置いたものは、確かめたら消してください。

10

D3-4 別テーマでの成果物づくり

[40min] 個人演習

この演習の [40min]は、必須のSTEP 1〜4の合計です。

ねらい

対象は、会議室予約という別テーマでの、要求一覧から試作品までの一式です。変わるファイルは、成果物/D3-4/に新規にできる要求一覧、レビュー、試作品のindex.htmlです。在庫管理のために作ったハーネスを、そのまま使います。要求の抽出、レビューの判断、試作品づくりをcommandの手順どおりに進め、条件4つと自分の工夫1つを入れた画面を作ります。

押さえること

成果物は人によって違って構いません。条件4つを満たしたうえで、どこを工夫したかをZoomのチャットに1行で添えます。

できあがりの目安

- ☐ `成果物/D3-4/01-requirements-review.md` のWARNが、判断を書き足した `[確認済]` になっている
- ☐ ブラウザで `http://localhost:8000/成果物/D3-4/index.html` を開くと、条件4つが動く
- ☐ 自分の工夫が1つ、画面に入っている
- ☐ 画面のスクリーンショットと工夫の1行が、Zoomのチャットにある



図15：試作品の例。今日と明日の予約が一覧で出る。この例の工夫は、プロジェクターのある部屋に付けたPJの印です

使うファイルと作るファイル

区分	ファイル	使い方
使う	`別テーマ/ヒアリングメモ_会議室予約.md`	サンプル電機株式会社（架空）の総務とのヒアリングメモ。発言に行番号がある
使う	`別テーマ/会議室.csv` 、 `別テーマ/予約_202612.csv`	試作品が読むサンプルデータ。列の意味は `別テーマ/データの説明.md`

区分	ファイル	使い方
作る	`成果物/D3-4/01-requirements-raw.md`、`成果物/D3-4/01-requirements-review.md`	要求一覧とレビュー
作る	`成果物/D3-4/index.html`、`成果物/D3-4/02-code-review.md`	試作品と、code-reviewerのレビュー

試作品の条件

条件	ブラウザでの確かめ方
予約の一覧	12月1日を基準日にすると、その日と翌日の予約が出る
会議室と日付での絞り込み	会議室と日付を選ぶと、その日の予約と空いている時間が出る
重複予約のエラー表示	ある予約と重なる時間で登録すると、エラーが出て一覧の件数が増えない
予約の取り消し	取り消すと、一覧からその予約が消える

試作品は、HTMLとJavaScriptの1ファイルで作ります。登録と取り消しは画面を開いている間だけ有効で、データのCSVは書き換えません。



この演習は、1つの会話で最後まで進めます。STEPの文面は前のSTEPの続きとして送るので、途中で`/clear`を打たないでください。演習が変わったら、`/clear`で会話を空にしてから始めます。

STEP 1 メモを読んで工夫を決める

AIを開かない　[5min]

1. VS Codeで`別テーマ/ヒアリングメモ_会議室予約.md`を開き、16行目から36行目の発言を読みます。
2. 条件4つのもとになった発言がどの行かを探します。
3. 条件のほかに、自分が画面に加える工夫を1つ決め、1行のメモに書きます。メモの発言から選んでも、自分で考えても構いません。

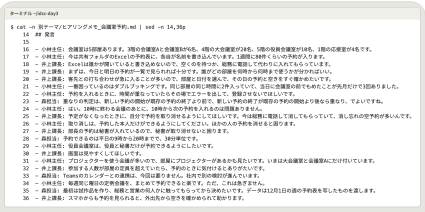


図16：ヒアリングメモの発言。左の数字が行番号で、要求一覧の根拠に使われる

STEP 2 要求の抽出から試作品まで

[20min]

入力欄の下が`accept edits on`になっていることを確かめてから送ります。終わるまでの目安は2〜3分です。

Claudeに送る文面

```
/extract-requirements 別テーマ/ヒアリングメモ_会議室予約.md 成果物/D3-4
```

報告にWARNが出ても、そのまま次の文面を送ります。`<ここに…>`は、STEP 1で決めた工夫に置き換えます。

Claudeに送る文面 メモ帳で<ここに…>を埋めてから貼る

```
/build-prototype 成果物/D3-4/01-requirements-raw.md <ここに STEP 1 で決めた工夫>
```

レビューのWARNが残っているので、`review_gate.py`のHookが送信を止めます。止められた送信では、モデルは呼ばれません。



図17：D3-1の`docs/`で同じ文面を試し、Hookが止めた画面。D3-4では`成果物/D3-4`のレビューのWARNが並ぶ

1. VS Codeで`成果物/D3-4/01-requirements-review.md`を開きます。
2. `［WARN］`の行を1件ずつ読み、行の終わりに「判断:」から始まる1文を書き足します。
3. 書き足した行の`［WARN］`を`［確認済］`に書き換えて保存します。`［ERROR］`があれば同じように扱います。

4 入力欄で↑キーを押して、`/build-prototype`の文面を呼び出し、もう一度送ります。

書き換えた行の形

- [確認済] FR-005 （指摘の文はそのまま残す） 判断: 試作品では扱わない。次のヒアリングで確かめる

判断に迷った指摘は、上の形の「試作品では扱わない。次のヒアリングで確かめる」で構いません。全部を深く検討する時間はありません。

ただし、試作品の条件4つにかかわる機能は、権限や細かい決まりが決まっていなくても試作品に入れます。取り消せる人についての指摘なら、「判断:取り消しの機能は入れる。取り消せる人は次のヒアリングで確かめる」のように、機能を入れることを書いてください。

2回目の送信では、Claudeが`index.html`を書き、code-reviewerが1回だけレビューします。ERRORはClaudeが直し、WARNとINFOは報告に出ます。こちらも、終わるまでの目安は2〜3分です。

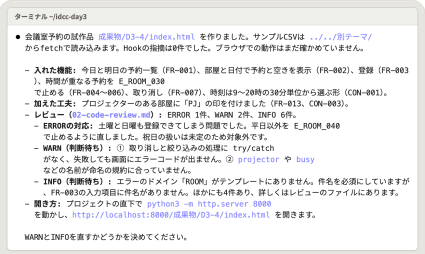


図18：試作品ができたときの報告。入れた機能と要求ID、加えた工夫、レビューの件数、開き方が返る

STEP 3 ブラウザでの確かめ

AIを開かない [8min]

2つ目のターミナルで、プロジェクトの直下からサーバを動かします。

2つ目のターミナルに打つコマンド

```
cd ~/idcc-day3
python3 -m http.server 8000
```

Windowsのブラウザで次のURLを開きます。

ブラウザのアドレス欄

```
http://localhost:8000/成果物/D3-4/index.html
```

試作品の条件の表を見ながら、4つを1つずつ画面で確かめます。終わる時刻と同じ時刻から始まる予約が登録できるかも、メモの24行目と照らして試してください。



図19：予約の一覧の上の部分。基準日の12月1日の6件が並び、この下に翌日の12月2日の6件が続く



図20：会議室Aと12月1日で絞り込んだ画面。予約2件と、30分ごとの空きが出る

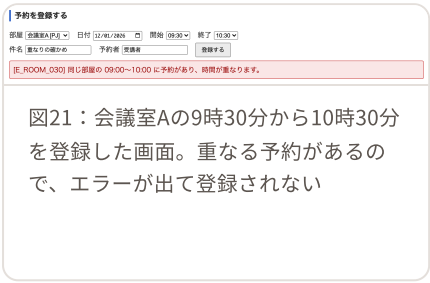


図21：会議室Aの9時30分から10時30分を登録した画面。重なる予約があるので、エラーが出て登録されない



図22：9時の予約を取り消した画面。一覧の件数が1件減った

動かない条件があれば、何が起きたかを1文でClaudeに伝えて直させます。2回伝えても直らなければ、Zoomのチャットで講師に知らせてください。

STEP 4 Zoomのチャットでの共有

AIを開かない [7min]

1. ブラウザに試作品を出したまま、Win+Shift+Sを押します。
2. 画面の上に出る切り取りの道具で、工夫が見える範囲をドラッグで囲みます。
3. Zoomのチャットの入力欄をクリックし、Ctrl+Vで貼り付けます。
4. 工夫を1行添えて送ります。

貼り付けても画像が出ないときは、Zoomの主催者の設定で、チャットのファイル送信が止められている可能性があります。工夫の1行だけを送り、画像が送れないことを書き添えてください。

うまくいかないとき

書き換えたのに、まだHookに止められる

Hookの表示に、残っている行のファイル名と行番号が出ます。その行の頭に`[WARN]`か`[ERROR]`が残っていないかを見てください。書き換えたあと、VS Codeでファイルを保存したかも確かめます。

ブラウザに`404`と出る、またはデータが出ない

サーバを`~/idcc-day3`で動かしているかを見てください。`成果物/D3-4`の中で動かすと、試作品がサンプルデータを読めません。アドレス欄が`file://`で始まっていたら、`http://localhost:8000/`から開き直します。

`Address already in use`と出て、サーバが動かない

別のターミナルで、もうサーバが動いています。そのままブラウザで開いてください。

Claudeがサーバを起動しようとする

承認の画面で止める側の選択肢を選び、「サーバは自分で動かします」と送ってください。サーバのコマンドは終わらないので、Claudeの会話が止まります。

早く終わった方へ

追加と考察

STEP 2 で`[確認済]`に書いた判断が、試作品のどこに反映されたかを見てください。`index.html`の画面の下にある要求IDの対応表で、入った要求と入らなかった要求が分かります。

発展課題

D3-4+ユーザー設定の`~/.claude/CLAUDE.md`に自分の好みを書き、成果物の見た目が変わるかを見ます。

1. 2つ目のターミナルで`code ~/.claude/CLAUDE.md`を打ち、VS Codeでファイルを開きます。ファイルが無ければ、新しいファイルとして開きます。
2. 最後の行に、自分の好みを1行足して保存します。例は「画面は白い背景にし、色は紺の濃淡だけで作る」です。
3. Claude Codeで`/exit`を打って終え、`claude`で起動し直します。`CLAUDE.md`は起動のときに読まれるためです。
4. 下の文面を送り、元の`index.html`と見比べます。

Claudeに送る文面

成果物/D3-4/index.html の見た目を、自分の好みに合わせて直し、成果物/D3-4/index_好み.html という別のファイルに保存してください。

研修のPCは、ユーザーを共有していることがあります。確かめたら、`~/.claude/CLAUDE.md`に足した行を消して保存してください。

11

よくある質問

`/usage` で、上限に近いという表示が出ました

Zoomのチャットで講師に知らせてください。研修用の組織の利用枠を全員で使っています。発展の課題は止めて、必須のSTEPだけを進めます。

Claudeの応答が10行で終わり、細かいところが分かりません

`CLAUDE.md` で応答を短くしてあります。詳しい中身は、Claudeが書いたファイルを開いて読んでください。「〜の理由を詳しく」と頼めば、その回だけ長く答えます。

モデルを変えたほうがよいですか

既定のモデルのままで進めてください。配布フォルダの設定とSubagentは、モデルを決めていません。研修用の組織の既定のモデルで動きます。

Hookを止めたいときはどうしますか

研修では止めません。`.claude/settings.json` と `.claude/hooks/` は、Claudeが書き換えられないように設定してあります。Hookが止めた理由が分からないときは、表示をそのままZoomのチャットに貼ってください。

このハーネスを自分の案件に持ち込めますか

`.claude/` と `CLAUDE.md` をフォルダごと写せば動きます。案件に合わせて書き換える箇所は、`CLAUDE.md` の規約と `settings.json` の禁止の一覧、`review_gate.py` が止めるcommandの3つです。書き換え方は `.claude/README.md` にあります。持ち込む前に、所属先の利用の決まりを確かめてください。

作ったものを手元に残せますか

`~/idcc-day3` は普通のフォルダです。Windowsのエクスプローラーのアドレス欄に `\\wsl$\Ubuntu-22.04\home\ユーザー名\idcc-day3` と打つと開けるので、コピーして残せます。

Subagentの一覧を見るコマンドはありますか

入力欄に `/context` と打つと、`Custom agents` の欄に出ます。Subagentの中身は、`.claude/agents/` のファイルをVS Codeで開いて読みます。

© Givery, Inc.

※題材のサンプル情報サービス株式会社、架空株式会社、取引先、人物、金額、勤怠と工数のデータは、研修のために作った架空のものです。