Claude Code シーン別プロンプト集（完全版）

> *非エンジニア向け Claude Code 開発体系 — シーン別即使えるプロンプト集 【】内の部分を自分のプロジェクトの情報に書き換えてから使うこと。*

使い方の3原則

1. **【】内を書き換える**：プロジェクト固有の情報に置き換えてから使う
2. **「修正は私の承認後に」が最重要**：AIが勝手に進まないよう制御する一文
3. **P-01-Aは毎回のセッション開始時に必ず使う**：これだけで成功率が大幅に上がる

CAT-01：セッション開始

新規セッション開始・長期中断後の再開・引き継ぎ準備

P-01-A：新規セッション開始（CLAUDE.md読み込み）

使うとき：新しいセッションを開始するとき必ず使う

```
このプロジェクトを引き継ぎます。
まず以下の順番でファイルを読み込んでください：
1. CLAUDE.md（プロジェクト概要・技術スタック・コマンド・完了基準）
2. progress.md（現在の進捗・完了タスク・次のタスク）
3. CONSTRAINTS.md（禁止事項・注意事項）

読み込んだ後、以下を日本語で報告してください：
- 現在のプロジェクト状況（1〜3文）
- 次に実行すべきタスク（具体的に）
- 不明点・確認が必要な事項

私の確認を得てから作業を開始してください。
```

P-01-B：長期中断後の再開

使うとき：数日以上中断したプロジェクトを再開するとき

```
このプロジェクトを再開します。
前回のセッションから時間が経っているため、以下を確認してください：

1. CLAUDE.md・progress.md・CONSTRAINTS.mdを読み込む
2. 最後のコミットログを確認する（git log --oneline -10）
3. 現在のファイル状態を確認する（git status）
4. テストが通るか確認する（テストコマンドを実行）

確認結果を報告し、「どこから再開するか」を私に提案してください。
私の承認なしに作業を開始しないでください。
```

P-01-C：セッション終了前の引き継ぎ準備

使うとき：セッションを終了する前・引き継ぎ書を作成するとき

このセッションを終了する前に、次のセッションへの引き継ぎ書を作成してください。

作成するファイル：`progress.md`（上書き更新）

記載内容：

```
## 完了したタスク
- [完了したもの一覧]

## 現在の状態
- ブランチ名：
- 最後のコミット：
- テスト状態：（全通過 / 一部失敗 / 未実行）

## 次のタスク（優先順）
1. [最優先タスク]
2. [次のタスク]

## 注意事項・引き継ぎメモ
- [次のセッションが知っておくべきこと]

## 未解決の問題
- [あれば記載]
```

作成後、内容を私に見せてください。

CAT-02：バグ調査

エラーメッセージからの調査・原因不明の動作異常・本番限定バグ

P-02-A：エラーメッセージからバグを調査する

使うとき：エラーメッセージが出ているとき

以下のエラーが発生しています。調査してください。

【エラーメッセージ】
[エラーメッセージをここに貼り付け]

【発生状況】
- いつ発生するか：[例：ログイン後、特定のボタンを押したとき]
- 再現手順：[1. 〜 2. 〜 3. 〜]
- 環境：[例：本番 / ステージング / ローカル]

調査の手順：
1. エラーの原因を特定する（ファイル名・行番号まで）
2. 根本原因を説明する（なぜ起きているか）
3. 影響範囲を確認する（他の機能への影響）
4. 修正案を提示する（修正前に私の承認を得ること）

修正は私が「修正してください」と言うまで行わないでください。

P-02-B：動作がおかしいが原因不明の場合

使うとき：エラーはないが動作がおかしいとき

以下の動作がおかしいです。原因を調査してください。

【期待する動作】
[本来どう動くべきか]

【実際の動作】
[実際に何が起きているか]

【調査してほしいこと】
1. 関連するコードを特定する
2. ログ・コンソール出力を確認する
3. 最近の変更（git log）と照合する
4. 考えられる原因を3つ以上挙げる

原因の仮説と調査結果を報告してください。
修正は私の承認後に行ってください。

P-02-C：本番環境でのみ発生するバグ

使うとき：ローカルでは再現しないが本番で発生するとき

```
本番環境でのみ発生するバグを調査してください。

【症状】
[何が起きているか]

【確認してほしい項目】
1. 環境変数の差異（本番 vs ローカル）
2. データの差異（本番データに特有のパターンがないか）
3. タイムゾーン・文字コードの問題
4. 外部サービス（API等）の本番設定の違い
5. ログファイルのエラー・警告

ローカルで再現できる方法があれば提案してください。
本番データには直接アクセスしないでください。
```

CAT-03：バグ修正

承認済みバグの修正実施・修正後の動作確認

P-03-A：承認済みバグの修正

使うとき：調査・承認が完了して修正を実施するとき

先ほど承認した修正を実施してください。

【修正内容】
[承認した修正内容を記載]

修正の手順：
1. 修正前に`git stash`またはブランチを作成する
2. 最小限の変更で修正する（関係ない部分は触らない）
3. 修正後にテストを実行する
4. テスト結果を報告する

修正後に確認してほしいこと：
- 修正したバグが解消されているか
- 他の機能が壊れていないか（回帰テスト）
- エラーログが出ていないか

全て確認できたら報告してください。

P-03-B：修正後の動作確認

使うとき：修正完了後に動作を確認するとき

修正が完了したとのことですが、以下を確認してください。

【確認項目】
1. バグが再現しないことを確認する（再現手順を実行）
2. 関連するテストが全て通ることを確認する
3. 手動で以下のシナリオを確認する：
    - 正常系：[正常な操作手順]
    - 異常系：[エラーが起きる操作]
    - 境界値：[限界値での動作]

確認結果を「OK / NG」で報告してください。
NGの場合は原因と追加修正案を提示してください。

CAT-04：コードレビュー

実装完了後のレビュー・特定観点でのレビュー

P-04-A：実装完了後のコードレビュー

使うとき：実装が完了してレビューを依頼するとき

```
実装が完了しました。コードレビューをしてください。

【レビュー対象】
[ファイル名またはディレクトリ]

【レビューの観点】
以下の順番でチェックしてください：

1. セキュリティ
   - APIキー・パスワードのハードコードがないか
   - 入力値の検証が適切か
   - 認証・認可の漏れがないか

2. 品質
   - エラー処理が適切か
   - ログが適切に出力されているか
   - 命名が分かりやすいか

3. 保守性
   - 重複コードがないか
   - 将来の変更に対して脆弱な箇所がないか

4. テスト
   - テストが書かれているか
   - 境界ケースがカバーされているか

問題点を「CRITICAL / HIGH / MEDIUM / LOW」で分類して報告してください。
CRITICALとHIGHは修正が必要です。MEDIUMとLOWは提案として扱います。
```

P-04-B：特定の観点でのレビュー

使うとき：セキュリティ・パフォーマンス等、特定観点のみレビューするとき

以下の観点でコードをレビューしてください。

【対象ファイル】
[ファイル名]

【重点的に確認してほしい観点】
[例：セキュリティのみ / パフォーマンスのみ / エラー処理のみ]

【背景・理由】
[なぜこの観点が重要か]

問題点と改善提案を具体的に（ファイル名・行番号付きで）報告してください。

CAT-05：テスト実行

全テスト実行・特定機能テスト・E2Eテスト実行

P-05-A：全テスト実行

使うとき：全てのテストを実行して結果を確認するとき

全てのテストを実行してください。

手順：
1. テストコマンドを実行する（CLAUDE.mdに記載のコマンドを使用）
2. 結果を報告する：
   - 通過したテスト数
   - 失敗したテスト数
   - スキップしたテスト数
3. 失敗したテストがある場合：
   - 失敗の原因を特定する
   - 修正が必要か、テスト自体が間違っているかを判断する
   - 私に報告して承認を得てから修正する

全テスト通過まで作業を続けてください。

P-05-B：特定機能のテスト

使うとき：特定の機能だけテストするとき

以下の機能のテストを実行してください。

【テスト対象機能】
[機能名]

【テストしてほしいシナリオ】
1. 正常系：[正常な入力・操作での動作確認]
2. 異常系：[不正な入力・エラー発生時の動作確認]
3. 境界値：[最大値・最小値・空文字・null等]
4. 本番想定データ：[実際のデータに近い値でのテスト]

テスト結果と、カバーできていないケースがあれば報告してください。

P-05-C：E2Eテストの実行

使うとき：エンドツーエンドのシナリオテストを実行するとき

E2Eテストシナリオを実行してください。

【実行するシナリオ】
[シナリオ名またはファイル名]

実行前の確認：
1. テスト環境が本番データに接続していないことを確認する
2. dry-runモードが有効になっていることを確認する
3. テスト用の認証情報を使用していることを確認する

実行後の報告：
- 各シナリオのPass/Fail
- 失敗した場合のスクリーンショット・ログ
- 本番環境で同じ結果が期待できるかの評価

CAT-06：ログ保存・調査

ログ収集・保存・エラーログ調査・デバッグログ追加

P-06-A：ログの収集と保存

使うとき：現在のログを収集・保存するとき

```
現在のログを収集・保存してください。

【収集するログ】
1. アプリケーションログ（エラー・警告・情報）
2. システムログ（必要な場合）
3. 外部サービスのレスポンスログ

【保存先】
logs/[日付]_[機能名].log

保存後：
- エラー・警告の件数を集計する
- 繰り返し発生しているエラーを特定する
- 異常なパターン（急増・特定時間帯への集中等）があれば報告する

ログファイルのパスと概要を報告してください。
```

P-06-B：エラーログの調査

使うとき：ログを分析して問題を特定するとき

以下のログを調査してください。

【ログファイル】
[ファイルパスまたはログ内容を貼り付け]

【調査してほしいこと】
1. エラーの種類と頻度を集計する
2. 最も深刻なエラーを特定する（影響範囲・発生頻度）
3. エラーの発生パターンを分析する（時間帯・特定操作後等）
4. 対処が必要なエラーと無視してよいエラーを分類する

調査結果を表形式でまとめてください。

P-06-C：デバッグログの追加

使うとき：問題調査のためにログを追加するとき

デバッグ用のログを追加してください。

【対象箇所】
[ファイル名・機能名]

【追加してほしいログ】
1. 処理の開始・終了
2. 重要な変数の値
3. 外部API呼び出しの前後（リクエスト・レスポンス）
4. エラー発生時の詳細情報

【注意事項】
- パスワード・APIキー等の機密情報をログに出力しないこと
- 本番環境ではDEBUGレベルのログが出力されないよう設定すること

追加後、テストを実行してログが正しく出力されることを確認してください。

CAT-07：セッション切り替え

セッション切り替え前の準備・コンテキスト枯渇時の緊急引き継ぎ

P-07-A：セッション切り替え前の準備

使うとき：セッションを切り替える前に必ず実行する

```
セッションを切り替える前に、以下を完了してください。

1. 現在の作業を保存する
   - 編集中のファイルを保存する
   - git add . && git commit -m "[作業内容の説明]"

2. progress.mdを更新する
   - 完了したタスクを記録する
   - 次のタスクを明記する
   - 未解決の問題を記録する

3. 引き継ぎメモを作成する（progress.mdに追記）
   ## セッション切り替えメモ（[日時]）
   - 現在のブランチ：
   - テスト状態：
   - 次のセッションで最初にすること：
   - 注意事項：

完了したら「引き継ぎ準備完了」と報告してください。
```

P-07-B：コンテキスト枯渇時の緊急引き継ぎ

使うとき：会話が長くなりコンテキストが限界に近づいたとき

```
コンテキストが限界に近づいています。緊急引き継ぎを行います。

今すぐ以下を実行してください（優先順）：

1. 現在編集中のファイルを全て保存する
2. git commit -m "WIP: [現在の作業内容]"
3. progress.mdに以下を追記する：
   ## 緊急引き継ぎ（コンテキスト枯渇）
   - 中断した作業：[何をしていたか]
   - 完了していないこと：[何が残っているか]
   - 次のセッションで最初にすること：[具体的な手順]
   - 注意事項：[引き継ぎ先が知っておくべきこと]

完了したら「引き継ぎ完了」と報告してください。
新しいセッションでは「P-01-A: 新規セッション開始」プロンプトを使用してください。
```

P-07-C：長い会話の要約・圧縮

使うとき：会話が長くなり重要情報が埋もれてきたとき

```
この会話が長くなってきました。重要な情報を要約してください。

【要約してほしい内容】
1. 今回のセッションで決定したこと（設計・仕様・方針）
2. 完了した作業
3. 発見した問題と対処方法
4. 未解決の問題

要約をsession_summary_[日付].mdとして保存してください。
その後、要約内容を確認して続きの作業を進めてください。
```

CAT-08：本番移行

本番移行前の最終チェック・段階的リリース計画の確認

P-08-A：本番移行前の最終チェック

使うとき：本番環境への移行前に必ず実行する

```
本番環境への移行前に最終チェックを行います。

以下を順番に確認してください：

【セキュリティチェック】
□ .envファイルが.gitignoreに追加されているか
□ APIキー・パスワードがコードにハードコードされていないか
□ エラーメッセージに機密情報が含まれていないか
□ 認証・認可が正しく実装されているか

【機能チェック】
□ 全てのテストが通過しているか
□ E2Eシナリオが本番想定データで通過しているか
□ エラー処理が適切か（ユーザーへの表示・ログ記録）

【運用チェック】
□ サーバー再起動後も自動的に再開するか
□ ログが適切に出力・保存されているか
□ 監視・アラートが設定されているか

【ロールバック準備】
□ ロールバック手順が文書化されているか
□ ロールバックに必要なバックアップが取得されているか

全項目をチェックして結果を報告してください。
NGがある場合は修正してから再チェックしてください。
```

P-08-B：段階的リリース計画の確認

使うとき：段階的に本番移行を進めるとき

本番移行を段階的に行います。以下の計画を確認してください。

【フェーズ1：dry-runテスト】
- 本番環境に接続するが、実際の操作は行わない
- 確認項目：接続・認証・データ取得

【フェーズ2：限定テスト】
- 1件だけ実際の操作を行う
- 確認項目：正常動作・ログ・エラーなし

【フェーズ3：本格稼働】
- 全機能を本番環境で稼働させる
- 監視：最初の1時間は手動で監視する

各フェーズの完了条件と、問題発生時のロールバック手順を確認してください。

CAT-09：緊急ロールバック

即時ロールバック・ロールバック後の原因調査

P-09-A：即時ロールバック

使うとき：本番で重大な問題が発生したとき即座に使う

緊急事態です。即座にロールバックしてください。

【問題の状況】
[何が起きているか]

ロールバック手順：
1. 現在の状態をバックアップする（git stash または git tag emergency-backup）
2. 直前の安定版にロールバックする（git revert または git reset）
3. サービスを再起動する
4. 動作確認を行う

ロールバック完了後：
- 問題が解消されたことを確認する
- ロールバックした内容をrollback_*log.mdに記録する*
- *原因調査は別途行う（今は安定化を優先）*

作業中は私に進捗を随時報告してください。

P-09-B：ロールバック後の原因調査

使うとき：ロールバック完了後に原因を調査するとき

ロールバックが完了しました。原因を調査してください。

【調査の手順】
1. 問題が発生したコミットを特定する（git bisect または git log）
2. 問題のあるコードを特定する
3. なぜ問題が発生したかを分析する
4. 今後同じ問題を防ぐための対策を提案する

【調査結果の記録】
incident_*report*_[日付].mdを作成して以下を記録する：
- 発生日時・影響範囲
- 根本原因
- 対応内容
- 再発防止策

調査結果を報告してください。

CAT-10：進捗確認

定期進捗確認・方針変更後の再同期

P-10-A：定期進捗確認

使うとき：現在の進捗を確認するとき

```
現在の進捗を確認してください。

以下を報告してください：
1. 完了したタスク（今日・今週）
2. 現在進行中のタスク
3. 次のタスク
4. ブロッカー（進行を妨げている問題）
5. 全体の完了率（概算）

progress.mdを最新の状態に更新してください。
```

P-10-B：方針変更後の再同期

使うとき：要件・仕様が変更になったとき

```
要件・仕様が変更になりました。現在の実装との差異を確認してください。

【変更内容】
[何が変わったか]

確認してほしいこと：
1. 変更の影響を受けるファイル・機能を特定する
2. 修正が必要な箇所をリストアップする
3. 変更の工数・リスクを評価する
4. 修正計画を提案する

確認結果を報告してから、私の承認を得て修正を開始してください。
```

CAT-11：セキュリティ確認

セキュリティ監査・機密情報の漏洩チェック

P-11-A：セキュリティ監査

使うとき：コード全体のセキュリティを確認するとき

```
コードのセキュリティ監査を行ってください。

【監査対象】
[ファイル名またはディレクトリ]

【確認項目】
1. 認証・認可
   - 認証なしでアクセスできる機能がないか
   - 権限チェックが適切か

2. データ保護
   - 機密データが暗号化されているか
   - ログに機密情報が出力されていないか
   - APIキーが環境変数で管理されているか

3. 入力検証
   - 外部からの入力が検証されているか
   - SQLインジェクション・XSS等の脆弱性がないか

4. 依存関係
   - 既知の脆弱性を持つライブラリがないか

問題点を「CRITICAL / HIGH / MEDIUM / LOW」で分類して報告してください。
```

P-11-B：機密情報の漏洩チェック

使うとき：APIキー・パスワード等がコードに混入していないか確認するとき

機密情報の漏洩がないか確認してください。

以下を検索してください：
1. APIキー・トークンのパターン（sk-、ghp_、xox等）
2. パスワードのハードコード（password=、passwd=等）
3. 本番URLのハードコード（https://api.本番ドメイン等）
4. 個人情報（メールアドレス・電話番号のパターン）

検索コマンド例：
```
git grep -i "api_key|password|secret|token" -- '*.py' '*.js' '*.ts'
```

問題が見つかった場合は、修正前に私に報告してください。
`git history`にも残っている可能性があるため、対処方法を相談してください。

CAT-12：完了確認

タスク完了の最終確認・ドキュメントの更新

P-12-A：タスク完了の最終確認

使うとき：タスクが完了したと報告を受けたとき

タスクが完了したとのことですが、完了チェックリストを確認してください。

【完了基準の確認】
□ `CLAUDE.md`に記載の完了基準を全て満たしているか
□ 全てのテストが通過しているか
□ コードレビューが完了しているか
□ ドキュメントが更新されているか
□ `progress.md`が更新されているか

【最終確認】
□ 本番環境で動作確認が完了しているか
□ ロールバック手順が確認されているか
□ 次のタスクへの引き継ぎ事項が記録されているか

全項目がOKになったら「完了」と報告してください。

P-12-B：ドキュメントの更新

使うとき：実装完了後にドキュメントを更新するとき

実装が完了しました。ドキュメントを更新してください。

【更新するドキュメント】
1. README.md
   - セットアップ手順が最新か確認・更新
   - 新機能の説明を追加

2. CLAUDE.md
   - 新しいコマンドがあれば追加
   - 完了基準を更新

3. progress.md
   - 完了したタスクを記録
   - 次のフェーズのタスクを追加

4. API仕様書（該当する場合）
   - 新しいエンドポイント・変更点を記録

更新後、内容を確認して報告してください。

プロンプト早見表

ID	カテゴリ	プロンプト名	使うタイミング
P-01-A	セッション開始	新規セッション開始	**毎回必ず**
P-01-B	セッション開始	長期中断後の再開	数日以上ぶりの再開時
P-01-C	セッション開始	引き継ぎ準備	セッション終了前
P-02-A	バグ調査	エラーメッセージ調査	エラーが出たとき
P-02-B	バグ調査	動作異常調査	動作がおかしいとき
P-02-C	バグ調査	本番限定バグ	本番でのみ発生するとき
P-03-A	バグ修正	承認済み修正	修正を承認したとき
P-03-B	バグ修正	修正後の確認	修正完了報告を受けたとき
P-04-A	コードレビュー	全体レビュー	実装完了後
P-04-B	コードレビュー	特定観点レビュー	特定の観点のみ確認するとき
P-05-A	テスト	全テスト実行	テストを全て実行するとき
P-05-B	テスト	特定機能テスト	特定機能だけテストするとき
P-05-C	テスト	E2Eテスト	シナリオテストを実行するとき
P-06-A	ログ	ログ収集・保存	ログを保存するとき
P-06-B	ログ	エラーログ調査	ログを分析するとき
P-06-C	ログ	デバッグログ追加	問題調査でログを追加するとき
P-07-A	セッション切り替え	切り替え前準備	セッション切り替え前
P-07-B	セッション切り替え	緊急引き継ぎ	コンテキスト枯渇時
P-07-C	セッション切り替え	会話の要約	会話が長くなったとき
P-08-A	本番移行	最終チェック	**本番移行前に必ず**
P-08-B	本番移行	段階的リリース	段階的に移行するとき

ID	カテゴリ	プロンプト名	使うタイミング
P-09-A	ロールバック	即時ロールバック	**本番で問題発生時**
P-09-B	ロールバック	原因調査	ロールバック完了後
P-10-A	進捗確認	定期進捗確認	進捗を確認するとき
P-10-B	進捗確認	方針変更後の再同期	要件・仕様が変わったとき
P-11-A	セキュリティ	セキュリティ監査	本番移行前・定期的に
P-11-B	セキュリティ	機密情報漏洩チェック	コミット前・定期的に
P-12-A	完了確認	最終確認	タスク完了報告を受けたとき
P-12-B	完了確認	ドキュメント更新	実装完了後

作成：非エンジニア向け Claude Code 要件定義自動化体系