プロンプトエンジニアリングのポイント

プロンプトの抽象度とトレードオフ

抽象的なプロンプト

- **作成負荷**：小さい（短く簡単）
- **AI委譲範囲**：広い（AIが多くを判断）
- **精度**：ばらつきが生じる
- **使いどころ**：探索的な作業、方向性を決める段階

具体的なプロンプト

- **作成負荷**：大きい（詳細な記述が必要）
- **AI委譲範囲**：小さい（人間が多くを指定）
- **精度**：向上する
- **使いどころ**：明確な要件、本番コード、重要な修正

最適なバランスを見極める

- ユースケースに応じて抽象と具体のバランスを調整（万能なレベルは存在しない）
- **コストパフォーマンス**が最も良いポイントを探す（プロンプト作成時間 vs 出力精度のトレードオフ）
- タスクの性質・重要度・時間制約を考慮

レビューとの兼ね合い

- AI駆動開発では「レビューアの負担が重くなる」という課題がある
- レビューには必ず「観点」があるため、それを適切にプロンプトに反映させることが重要
- Clineの場合は、/clinerulesでレビューの観点（設計標準やコーディング規約）を定義することができる

プロンプトの抽象度・具体度に応じた分類（当研修固有）

Simple（最も抽象的）

- **特徴**：短く簡潔な指示
- **長さ**：最大でも20行程度
- **メリット**：素早く指示できる
- **デメリット**：出力のばらつきが大きい
- **適用場面**：探索・試作・アイデア出し

Just（バランス重視）

- **特徴**：必要十分な情報
- **長さ**：20行〜100行程度
- **メリット**：作成負荷と精度のバランスが良い
- **デメリット**：状況判断が必要
- **適用場面**：通常の開発タスクの大半

Much（最も具体的）

- **特徴**：詳細な仕様・制約・例を含む
- **長さ**：数百行
- **メリット**：高精度な出力
- **デメリット**：プロンプト作成に時間がかかる
- **適用場面**：本番実装、複雑な要件、品質重視

本研修におけるユースケース

- **berry-books**：Jakarta EEによるWebアプリケーション（オンライン書店）
  - レッスン1：小規模改善
  - レッスン2：ガイドラインへの準拠チェック
  - レッスン3：不具合修正
  - レッスン4：機能拡張
  - レッスン5：単体テスト生成
- **berry-books-frontend**：ReactによるSPA（オンライン書店の管理者画面）
  - レッスン6：仕様書からのスクラッチ開発（React）
- **struts-to-jsf-person**：StrutsからJSFへのマイグレーション
  - レッスン7：フレームワークのマイグレーション（リライト）
- **accounting_glue**：PySpark/GlueによるETL処理（会計EPRへのバッチデータ連携）
  - レッスン8：仕様書からのスクラッチ開発（PySpark/Glue）