③ 性能向上：ワークフロー.py 性能比較レポート（Sample）

1. 基本情報

- 受講者氏名：田中　太郎
- 作成日：2025年10月1日
- 対象工程：③ 性能向上（モデル変更・最適化・性能比較）
- 比較対象モデル：
  - モデルA：Claude Sonnet 3.5（汎用型・高速処理）
  - モデルB：Claude Sonnet 4.0（高精度・複雑処理）
  - モデルC：GPT-4.1（バランス型・安定稼働）
  - モデルD：GPT-5（次世代型・高度AI機能）

---

2. 実験概要

- 実験目的：4つのAIモデル統合アーキテクチャにおける性能特性の定量評価と最適モデル選定指針の策定
- 比較観点（例：処理速度、精度、リソース使用量）：
  - 処理速度：3件データ処理の総実行時間（秒）
  - 処理精度：登録成功率とコード取得正確性（%）
  - メモリ効率：最大メモリ使用量とガベージコレクション効果
  - エラー回復率：TimeoutException等の自動復旧成功率
  - コード品質：可読性・保守性・拡張性の定性評価
- 使用データセット：data.xlsx（3件のサンプル経費データ）
  - 行1：タイトル「交際費」、種別「その他」、金額5030、備考「先月分」
  - 行2：タイトル「宿泊費」、種別「立替」、金額7590、備考「ホテル代」
  - 行3：タイトル「交際費」、種別「仮払」、金額10980、備考「今月分」
- 実行環境（OS / CPU / GPU / メモリ / ツール / バージョン）：
  - OS：Windows 11 Pro（64bit）
  - CPU：Intel Core i7-12700F（3.6GHz）
  - GPU：なし（CPU処理）
  - メモリ：32GB DDR4
  - ツール：Python 3.10.11、VS Code、仮想環境(.venv)
  - Selenium：4.10.0、webdriver-manager：4.0.0
- 実行回数：各モデル5回実行（平均値算出）
- 計測方法：time.time()による実行時間測定、psutil.Process()によるメモリ監視
- 許容基準値：処理成功率98%以上、3件処理15秒以内、メモリ使用量1GB以下

---

3. 実験結果

比較条件	成功率（%）	正確性（%）	処理時間（秒）	備考
Claude Sonnet 3.5	100.0	100.0	7.25	最高速度・軽量実装
Claude Sonnet 4.0	100.0	100.0	10.50	高精度・堅牢設計
GPT-4.1	98.5	100.0	9.76	バランス型・安定動作
GPT-5	100.0	100.0	9.00	次世代・高機能実装

詳細性能指標

モデル	メモリ使用量(MB)	ガベージ回収数	エラー回復率(%)	実装特徴
Claude Sonnet 3.5	485	127	95.0	@retry_on_staleデコレータ
Claude Sonnet 4.0	520	98	98.5	ExcelOptimizedOperations
GPT-4.1	502	115	92.0	safe_click/safe_input関数
GPT-5	545	89	100.0	dataclasses設定管理

---

4. 考察

- 性能差の要因：
  - Claude Sonnet 3.5：軽量実装による高速処理、メモリ効率最適化
  - Claude Sonnet 4.0：堅牢なエラーハンドリングによる高い回復率、品質重視設計
  - GPT-4.1：標準的な実装による安定性、シンプルな構造で保守性確保
  - GPT-5：dataclasses活用による構造化設計、最新技術による高機能性
- 改善が見られた点：
  - 全モデルで統一アーキテクチャ（@dataclass、robust_automation）による品質向上
  - WebDriverManager活用による環境構築の自動化と互換性問題の解消
  - メモリ最適化（float32型変換）による37.5%のメモリ削減効果確認
- 劣化が見られた点：
  - GPT-4.1で若干の成功率低下（98.5%）、ネットワーク系例外処理の改善要
  - 高機能モデル（GPT-5、Claude Sonnet 4.0）でのメモリ使用量増加傾向
- 想定外の結果や原因：
  - Claude Sonnet 3.5の予想を上回る高速性能（設計値7秒台を達成）
  - 全モデルでコード取得正確性100%達成（6桁数字の正規表現抽出が効果的）

---

5. 結論

- 最適モデルの選定理由：
  - **高速処理用途**：Claude Sonnet 3.5（7.25秒、メモリ485MB）
  - **高精度・重要処理用途**：Claude Sonnet 4.0（回復率98.5%、堅牢設計）
  - **日常業務・バランス型**：GPT-4.1（安定性重視、シンプル構造）
  - **先進機能・実験用途**：GPT-5（構造化設計、拡張性最高）
- 今後の改善案：
  - アンサンブル予測：4モデルの結果統合による精度・信頼性向上
  - 動的モデル選択：データ量・処理要求に応じた最適モデル自動選択機能
  - パフォーマンス監視：リアルタイム性能監視とボトルネック自動検出
- 他工程への適用可能性：
  - マルチAIモデル統合パターンの他自動化プロジェクトへの適用
  - 業務要件に応じたモデル選択指針の標準化・テンプレート化
  - エンタープライズ環境での大規模展開における性能予測基準として活用

---

6. 再現手順

1. Python仮想環境(.venv)の構築とrequirements.txtからの依存関係インストール
2. data.xlsxファイルの準備（3件のサンプルデータ配置）
3. 各AIモデルスクリプトの順次実行（python .¥py¥経費登録_{model}.py）
4. expense_automation.logでの実行時間・メモリ使用量確認
5. data.xlsxでの登録コード書き戻し結果検証（563696、563697、563698）

---

7. 添付・参考資料

- 添付ファイル名：
  - AIモデル比較分析.html（詳細性能レポート）
  - 各モデル実行ログ：expense_automation_{model}.log
  - 検証結果データ：data_result_{model}.xlsx
  - 性能測定スクリプト：performance_benchmark.py（カスタム測定ツール）
- 参考URLや文献：
  - プロジェクト計画書.md（マルチAIモデルアーキテクチャ設計）
  - 開発ガイドライン（統一アーキテクチャ実装指針）
  - Selenium Performance Best Practices
  - Python Memory Optimization Techniques