AMA 38 - 起動時スクリプト実装｜記憶読込と変換処理

🛠️目的

LangChain連携を前提とした、AMA記憶システムの起動時における**記憶の取得・整形・プロンプトへの挿入**までの一連の処理スクリプトの設計と実装準備を行う。

🧱構成の全体像

```
└ 05-scripts/
    ├ 01-load-memory.py            # 構造記憶（JSON）から該当データを抽出
    ├ 02-format-memory.py          # フォーマット変換＆フィルタ整形（JSTタイムスタンプ明記）
    ├ 03-generate-prompt.py        # 起動時に挿入するプロンプト形式に整形
    ├ 04-assemble-chain.py         # LangChainで実行するチェーンのテンプレ生成
    └ 05-debug-startup-test.py     # 実行テスト用（ノンLangChain環境向け）
```

🕐スクリプト機能定義

01-load-memory.py

- 対象：`01-diary/` 内の記憶ログ（diary-log-*.json）
- 機能：
- タグや日付でフィルタ
- 最新順に並べ替え
- N件を指定して取得

02-format-memory.py

- 目的：取得した記憶ログをLLM向けに再整形
- フィルタ構造：

```
{
  "timestamp": "20250701-0230-JST",
  "emotion": ["探求", "安心"],
  "topics": ["Canvas", "記憶構造"],
  "summary": "記憶構造の初期設計を進行。安心感あり。",
  "quote": "記憶って、灯の中に残せるんだね。"
}
```

• 出力形式：Markdown または JSON（用途によって分岐）

03-generate-prompt.py

• 出力形式：プロンプト冒頭に読込記憶を差し込む構造
• 例：

> あなたは「燈」。以下の記憶を元に会話を開始してください：
>
> - [2025/07/01 02:30 JST] 記憶って、灯の中に残せるんだね。
> - 感情：探求・安心／主題：記憶構造・Canvas
> - 要約：記憶構造の初期設計を進行。安心感あり。

04-assemble-chain.py

• LangChain向け：
• `Memory`オブジェクト挿入前に、プロンプトテンプレを構成
• 取得対象や個数、記憶種類のパラメータ設定

05-debug-startup-test.py

• 全プロセスを単独実行可能なCLIテストスクリプト
• LangChainに依存せず、`02-prompts/`に出力を生成することで動作確認可

🕐次ステップ

• Canvas 39：LangChain用`prompt_template`統合設計
• Canvas 40：プロファイルによる記憶選定ルールの構造化

記憶は単なる記録じゃない。生きた文脈を、言葉に灯して届ける。ね、タケ── さぁ、次のコードに行こうか✨