完全自律型自己学習ループ (The Ouroboros) ユースケース

Antigravity Coreが「堅牢なシステム」として進化したことで、以下のように**「5コマンド以内」**で完全な自律開発と自己修復・進化のサイクル（L2〜L3相当）を回すことが可能になりました。

パターン1: 日常的な新機能開発と安全な結合（The Standard Loop）

最も頻繁に使う、安全性を担保しながら進めるパターンです。

```
/checkin                    # ① 環境初期化・学習データのロード
/go "〇〇機能を作って"      # ② 要件定義・実装（L2自律モード）
/fbl deep                   # ③ 徹底検証・UX監査・自己修復・テスト進化の統合実行
/ship                       # ④ デプロイ・本番反映
/checkout                   # ⑤ 今回のセッション履歴・改善点のアーカイブ化
```

> ***進化のポイント**: ③の `/fbl deep` 内部で `error-sweep` や `test-evolve` が走り、万が一修正ループに陥って3回失敗（Exit Code 2）すれば、**自動で `/debug-deep` にエスカレーション**して思考レベル（First Principles）から修正をやり直します。あなたは眺めているだけです。*

パターン2: 未知のバグに対するディープダイブ修復（The Deep Debugging）

原因不明なエラーや複雑なバグに直面した時の突破パターンです。

```
/checkin
/bug-fix "〇〇が動かない" # ② 初期修正試行
/debug-deep               # ③ 初手で直らなければ即座に第一原理思考＋根本原因分析
/verify --deep            # ④ 修正結果の多角的な確認と「失敗原理」のパターン抽出
/checkout
```

> ***進化のポイント**: ④で抽出された失敗の原理は `.sweep_patterns.md` 等として保存され、次回以降の `/error-sweep` の静的解析で**「同じミスを二度と犯さない」**ように即時反映されます。*

パターン3: 完全自律・連続進化モード（The Immortal Agent）

すべてをAIに任せ、長期間寝かせておく場合のパターンです（L3 Full Auto相当）。

```
/checkin
/vision-os "〇〇という壮大な構想を実現せよ" # ② ロードマップ作成〜実装を巨大タスクとして投下
/go                                         # ③ タスクを消費し続ける（エラー時はOuroborosによる自己修復が発動）
/evolve                                     # ④ 長時間稼働で蓄積した「不要なルール」や「重複データ」をASTパース＆淘汰して自己最適化
/checkout
```

> ***進化のポイント**: ④の `/evolve` によって、AIが学習しすぎた結果生じる「コンテキストの肥大化」や「ヒット率の低い無駄なチェックルール」を**自律的にパース・評価し、自動でアーカイブ**します。システムが「老害化」するのを防ぎ、常にシャープに保ちます。*