LLM Aucoder

Github: https://github.com/CrowHatter/Local\_LLM\_Coder

本系統由兩大部分組成：

1. **MCP (Multi-Component Processing) Server** ── mcp.py
2. **多代理人工作流 (5-Agent Workflow Client)** ── client\_orchestrator.py

### MCP Server (mcp.py)

MCP 採用 FastMCP 作為核心框架，並定義三個實用工具（tool functions），用於知識搜尋、程式寫入、與程式執行等功能。

#### Tools 實作

#### 1. RAG\_Search

* 功能：從 utils/ 目錄中所有文字檔案建構嵌入式資料庫（RAG），並以語意搜尋找到最相關的片段。
* 運作流程：
  + 使用 SentenceTransformer 向量化資料段落。
  + 使用餘弦相似度計算查詢與段落間的關聯性。
  + 取前三名相關結果返回。

#### 2. WriteTemp

* 功能：將 LLM 產生的 Python 程式碼寫入 temp.py 檔案中。
* 運作流程：簡單地將接收到的程式碼寫入硬碟上，供之後執行使用。

#### 3. ExecTemp

* 功能：以指定的 conda 環境（autocoder）執行 temp.py，並捕捉輸出或錯誤訊息寫入 temp.log。
* 回傳值：包含執行是否成功、輸出內容（log）、與狀態標記。

### Agents 設計說明

Client 端負責 orchestrate 全部流程，從需求開始，自動執行以下五個 Agent：

### 1. RetrieverAgent

* 功能：呼叫 MCP 的 RAG\_Search 工具，回傳與需求語句語意最接近的文件片段。
* 技術重點：透過語意搜尋實作知識擷取（RAG pipeline 的第一步）。

### 2. PlannerAgent

* 功能：根據使用者需求與檢索上下文，由 LLM 規劃出一份英文的「實作計畫」。
* 輸出格式：純文字計畫，不包含任何程式碼。
* 技術重點：設計 prompt 引導 LLM 像架構師般擬定可行規劃。

### 3. CoderAgent

* 功能：輸入實作計畫，生成對應 Python 程式碼。
* 技術重點：
  + Prompt 限制只回傳一段 fenced code block。
  + 使用 regex 萃取程式碼（python ... 區塊）。

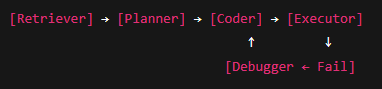
### 4. ExecutorAgent

* 功能：
  + 寫入 temp.py 並執行它。
  + 檢查輸出是否與實作計畫符合。
* 技術亮點：
  + 結合 WriteTemp + ExecTemp。
  + 使用 LLM 判斷 log 是否符合預期（模擬自動化 judge）。

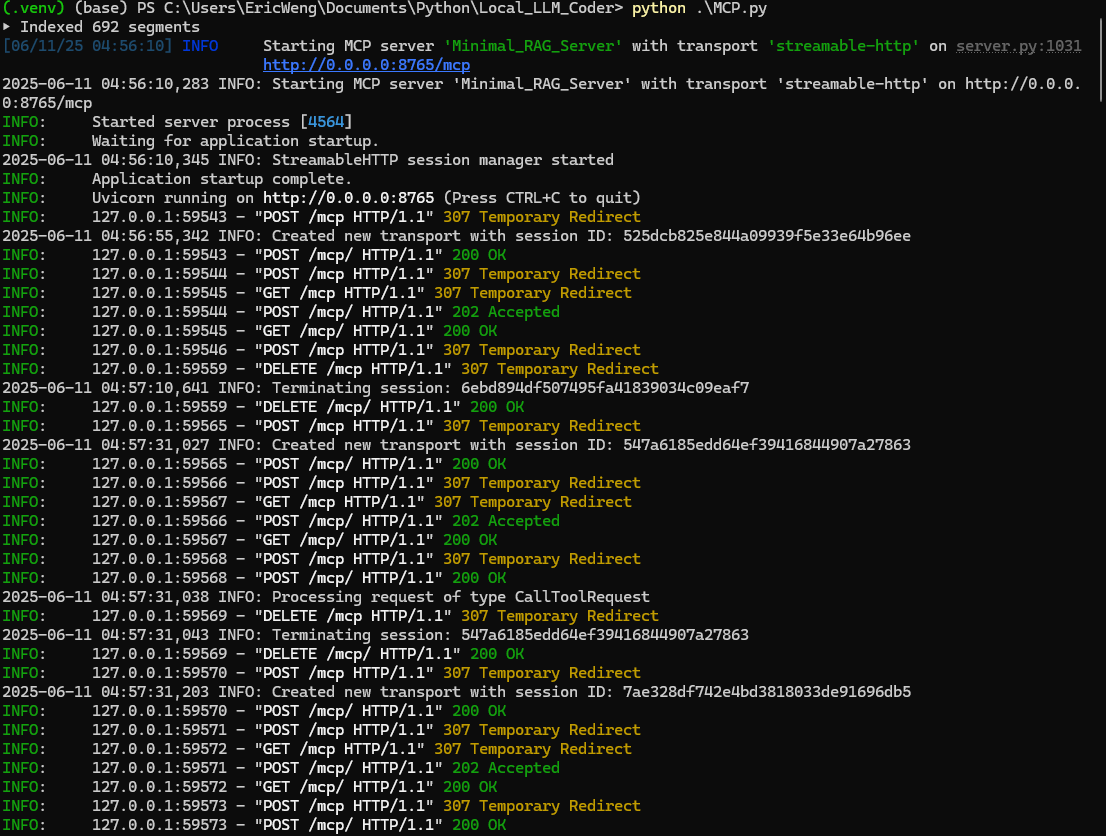
### 5. DebuggerAgent

* 功能：若執行失敗，請 LLM 根據 log 與原始計畫回饋，生成新計畫。
* 目的：實作自我修復與強化迴圈（Self-improvement loop）。

#### Agent 流程總覽



#### Demo



🔹 prompt: write a quick sort example and print the sorted result  
