

2026.03 俄羅斯方塊與自我介紹作業

在 AI Agent(智能體)深度融入開發流程的趨勢下, 命令列介面(CLI)AI 工具已成為開發者重度依賴的「終端內編程助手」。除了 Anthropic 的 **Claude Code** 之外, 美中兩國各自發展出許多功能強大且具備特色的主流 CLI 系統:

美國主流 CLI AI 系統

美國的 CLI 工具大多依賴強大的基底模型(如 Gemini、GPT-4o、Claude 3.5/3.7), 並且非常注重自動化沙盒執行、**Git** 工作流整合與開發者生態。

1. Claude Code (Anthropic)

- 特點: CLI 終端 Agent 的業界標竿, 具備極強的代碼推理與自主執行能力。
- 優勢: 深度整合 Git 工作流, 支援內建的 `/simplify` 與 `/batch` 指令, 能自主掃描專案、修改多個檔案、跑測試並修正錯誤, 語感最像真人且幻覺率最低。
- 適用場景: 複雜架構重構、端到端功能開發、高精準度的專案維護。

2. Antigravity (Google)

- 特點: 免費額度高、支援 1M+ Token 超長上下文。
- 優勢: 原生整合 GCP(Google Cloud), 適合用於大規模程式碼庫(Codebase)分析、文檔解讀與架構探索。
- 適用場景: 專案前期架構理解、大型檔案掃描。

3. Codex CLI / Copilot CLI (OpenAI / GitHub)

- 特點: 強調系統層級的安全性與效能。
- 優勢: Codex CLI 底層採用 Rust 重寫, 執行速度極快, 具備內核級沙盒環境, 非常適合自動化 CI/CD 流水線與腳本執行。Copilot CLI 則完美契合 GitHub 生態圈。
- 適用場景: 自動化測試、CI/CD 腳本編寫、輕量化 Terminal 操作。

3. Aider (開源界熱門代表)

- 特點: Git 原生的配對程式設計師 (Pair Programmer)。
- 優勢: 可以靈活切換 OpenAI、Claude、DeepSeek 等多種 API。它能在你完成程式碼修改後, 自動生成非常精準的 Git Commit 訊息並提交, 在開發者社群中擁有極高評價。
- 適用場景: 靈活切換模型的重度 Git 開發者。

中國主流 CLI AI 系統

中國的 CLI 系統主要基於本土頂尖的開源/商業大模型 (如 Qwen3-Coder、DeepSeek、Kimi 等), 並且特別針對中文語境、海量網路搜尋以及國內網路與 API 存取環境進行優化。

1. Qoder CLI (阿里灵码)

- 特點: 被譽為國內「Claude Code 強力平替」。
- 優勢: 預設接入 Qwen3-Coder、DeepSeek-V3.1 等頂尖國產模型, 具備本土化搜尋 (BrowseComp-ZH)、多模態圖片識別, 且能自動化執行環境配置、資料集下載與代碼修復。
- 適用場景: 國內環境下的全自動化 Terminal Agent。

2. Qwen Code CLI (阿里巴巴)

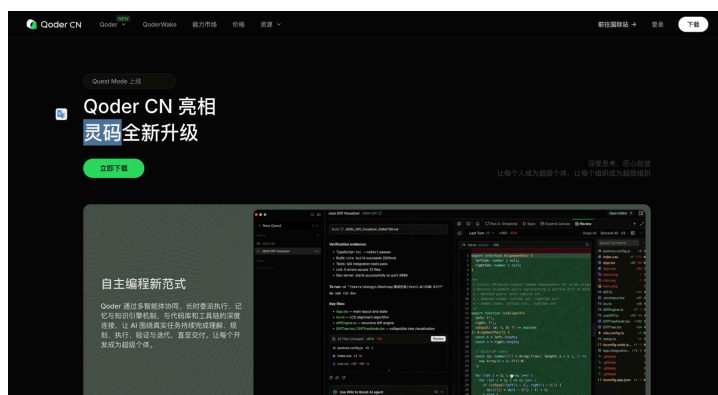
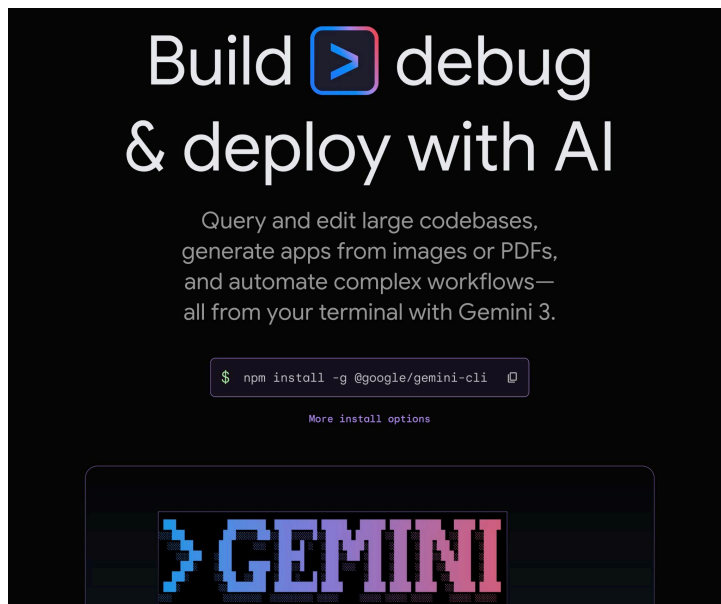
- 特點: 基於 Qwen (通義千問) 開源生態。
- 優勢: 極度輕量化, 且在多語言 (中英雙語) 語法與程式碼生成上表現非常精準。支持地端的 Qwen-Coder 模型私有化部署, 資料安全性高。
- 適用場景: 企業私有部署、中英雙語開發環境。

3. Kimi Code CLI / OpenCode (月之暗面 / 社群開源)

- 特點: 基於 Kimi 龐大的長文本理解能力, 或透過 OpenCode 框架調用。
- 優勢: 處理長篇 API 文件、分析複雜業務邏輯時表現優異, 且對中文指令的意圖理解非常精確。
- 適用場景: 重度中文需求、複雜業務邏輯拆解。

作業一

請選擇以上任一系統，不知如何選擇，請用你最熟悉的如 Gemini / Qoder，在系統內輸入，“幫我做一個俄羅斯方塊”，然後試試看成果。可能會出現相關訊息，就是確定執行。”



如果順利完成，可以嘗試不同的視覺風格。（此作業為毓修老師提供）

完成後請問 AI 如何發表到網路上，讓其他人可以使用，你會得到幾個選項，如 Github Pages 或是 Netlify Drop，按照 AI 建議一步一步完成後，請將成果貼出來，讓大家使用。如下所示，唐老師大概花的五分鐘吧。

<https://statuesque-cascaron-c4240f.netlify.app/>

作業二

請準備三個關於自己的文檔，可以是 pdf 檔，可以是 CV，可以是 Portfolio。請使用您熟悉的 AI 工具，和系統聊天，嘗試建立自我介紹的聊天機器人 chatbot，讓大家可以詢問你相關的問題。沒有標準答案，但是希望要能分享在 wechat 上，讓同學可以問你的 chatbot 問題，讓大家可以認識你。

例如這是我用 10 分鐘用 Antigravity 做的簡單 Chatbot

<https://tang-cv-chatbot.vercel.app/>

唐

唐玄輝 HSIEN-HUI TANG
國立臺灣科技大學 設計系主任兼教授
設計創新思考研究室 (DITL) 總監

Hsien-Hui Tang is a Design Professor at Taiwan Tech, who enjoys both research and practice of AI-assisted product, service, and experience innovation, conducting research projects, publishing papers, executing design projects, and hosting industry trainings.

22 年
設計教育與產學實踐

20+ 座
國際頂尖設計大獎

100+ 場
企業創新培訓與工作坊

亞洲首度
CHI SDC & SDN 金獎

2026 最新前瞻演講

唐玄輝教授 AI 數位特使
National Taiwan University of Science and Technology - DITL

唐

您好！我是國立臺灣科技大學設計系主任兼教授、設計創新思考研究室 (DITL) 總監 唐玄輝教授 (Prof. Hsien-Hui Tang) 的 AI 數位特使。

我能代表唐老師向您介紹他在體驗戰略、服務創新、AI 應用的 22 年實踐與教育成果：

- 累計 20+ 座國際頂尖大獎 (包含亞洲首度奪得 ACM CHI SDC 金獎與 SDN 服務設計金獎紀錄)
- 頂尖企業校友網絡 (遍佈 Google、微軟、台積電、阿里巴巴等頂尖機構)
- 百場企業內訓與產學合作 (涵蓋 P&G、富邦銀行、大潤發、公部門公共服務改革等)
- 2026 前瞻觀點 (《人工智慧浪潮中 創新的改變及 設計的消逝與重生》)

您可以點擊下方的精選問題，或隨時在輸入框中提問！

精選探索問題

請介紹唐玄輝老師的核心專業領域與背景 →

唐老師曾獲哪些國際大獎？指導團隊的歷史紀錄是什麼？ →

學生畢業後的出路與指導特色如何？ →

唐老師進行過哪些企業培訓與產學合作？ →

詢問唐老師的研究領域、國際大獎、學生出路或 2026 AI 設計觀點...

6/588 ↑

PROF. HSIEN-HUI TANG - DESIGN INNOVATION & THINKING LAB (DITL) - AI AMBASSADOR

這兩個作業的重點不是測試你的 Coding 能力，而是測試你是否知道如何運用 AI 協助建立資訊系統，跟 Vibe Coding 能力無關，而是 AI 素養，所有思考服務體驗創新的同仁都應該要具備。

幾件事情建議

1. 如果遇到問題就問 AI，他會教你一步一步執行。講不清楚就用截圖傳給 AI，也是可以。
2. 可以改成中文顯示。
3. 建立一個新的檔案匣，只允許他在這個檔案匣內工作，不能給所有檔案匣的權限。
4. 目前還不需要用到 API 和帳號密碼，所以不要給 AI。
5. 如果覺得設計不好，就跟 AI 說好醜。
6. 如果用 github vercel cloudflare 發表你的作品，需要學一下，不會的話一樣截圖問 AI。

≠

2026.03 自我介紹作業

要完成的任務

(階段一) 請準備三個關於自己的 pdf 檔, 可以是 CV, 可以是 Portfolio。請使用您熟悉的 AI 工具, 運用 vibe coding 的理念, 建立自我介紹的 RAG 系統, 讓大家認識你。建議是用編碼型 IDE 配合上通用型 AI, 會最省時間與減少問題。

作業說明分為三個階段說明, 請紀錄做作業的時間, 以及看到第幾階段的說明, 只看階段一, 少於兩個小時就完成的, 是具備良好的 Vibe Coding 能力, 以及目前國際數位環境能力。

階段一：懂 vibe coding 的人, 我們提供 要完成的任務, 就是目前這一段。

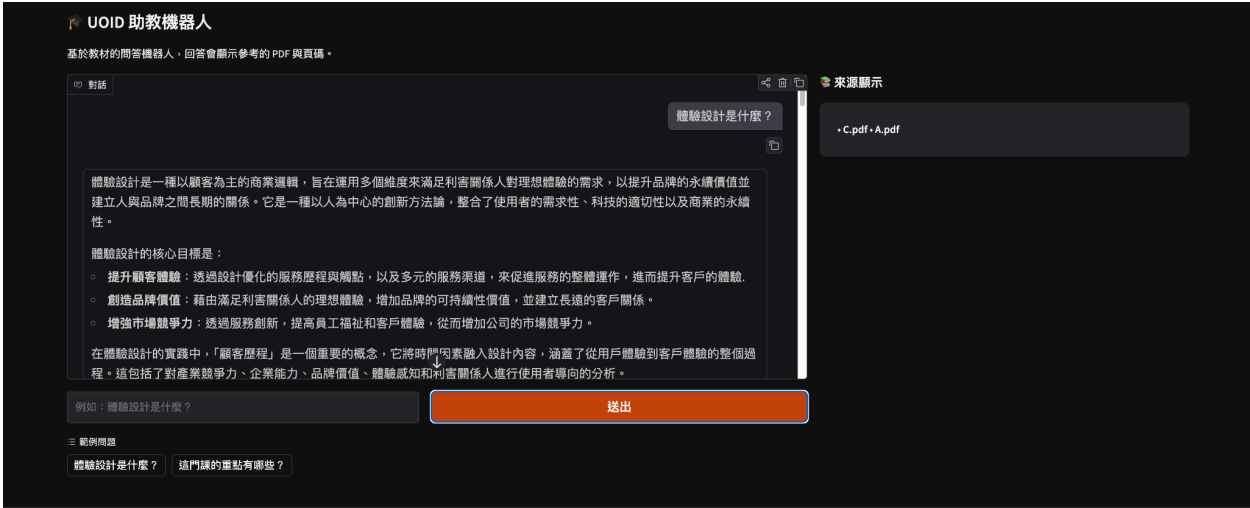
階段二：積極嘗試vibe coding 的人, 我們提供 資訊服務與框架的說明

階段三：剛起步, 需要一步一步教學的人, 我們提供 **UOID** 助教機器人 — 完整操作教材

目前作業說明是以 Google 體系為主, 過程中可以使用 cursor 搭配 Gemini, 但是也可以用其他的工具。作業目的是了解大家的 AI 素養, 以及建立思考 AI 賦能的服務體驗創新。

作業的重點不是測試你的 Coding 能力, 而是測試你是否知道如何運用 AI 協助建立資訊系統, 所以跟 Coding 能力 無關。

作業完成後, 請將 RAG 連結貼到最後的成果頁。



AI 工具介紹

進入 2026 年, AI 工具的版圖已經從單純的「對話框」演變成「代理程式 (Agents)」的全面對決。目前市場有兩類產品:「全能大腦 (通用型)」與「自動工廠 (IDE 型)」。

一、通用型 AI: 你的「數位大腦」

這些工具的核心競爭力在於模型推理能力與生態系整合。

工具	2026 年的定位與強項	適合場景
Claude (Anthropic)	代碼與邏輯的王者。憑藉著 Claude 3.7/4 系列, 它依然是工程師的首選。其 /simplify 與 /batch 展現了它對開發流程的深度理解, 語感最像真人, 幻覺率最低。	複雜架構設計、重構、高品質文案、Vibe Coding。
Gemini (Google)	長文本與多模態霸主。支援 200 萬甚至更高的 Token 窗口。它能「一口氣吃掉」你整個專案的影片演示、所有文件與歷史記錄, 並精準定位	大規模專案分析、多模態輸入 (看圖/片寫 code)、Google Workspace 深度整合。

	問題。	
ChatGPT (OpenAI)	全能通才。GPT-5/5.1 在邏輯推理和數學能力上極強，且「Deep Research」模式能幫你做極深度的技術調研，不再只是淺層回覆。	技術調研、疑難雜症排查、跨領域創意發想。
DeepSeek (深度求索)	極致性價比與開源先鋒。2026 年的 DeepSeek-V3/V4 在程式碼能力上已能硬剛 GPT-5，且價格極低，是許多開發者串接 API 的首選。	大規模自動化任務 (搭配 /batch)、預算有限的高頻率開發。

二、編碼型 IDE：你的「自動化工廠」

如果說通用 AI 是大腦，這類 IDE 就是「會寫 Code 的手」。

1. Cursor：目前最成熟的「輔助駕駛」

Cursor 依然是 AI-native IDE 的標竿。它的優勢在於 "Flow" (心流)：

- **Composer** 模式：能在幾秒鐘內根據你的描述修改多個檔案。
- 預測性編輯：在你還沒打完字前，它就已經預測了你下一步的重構路徑。
- 甜蜜點：適合那些「想要掌控細節，但想跳過重複勞動」的開發者。

2. Antigravity (Google)：新崛起的「Agent 經理人」

這是 Google 在 2025 年底推出的「Agent-first」開發平台，徹底顛覆了 IDE 的邏輯：

- 從「輔助」變「委派」：在 Antigravity 中，你更像是一個管理者。你可以開啟「Manager View」，同時派發三個 Agent：一個去修 Bug、一個去寫單元測試、一個去跑瀏覽器自動化測試。
- **Artifacts** 驗證：它不只是給你代碼，還會產出截圖、錄影，證明它跑過測試且成功了。
- 甜蜜點：適合 **Greenfield** (新開發專案) 或需要大規模異步作業的場景。

3. Trae：來自 ByteDance 的強大對手

Trae 是目前 Cursor 最強的免費替代品 (在 2026 年依然維持極佳的免費額度)：

- 速度極快：針對程式碼生成做了極致優化，延遲感比 Cursor 更低。
- 零門檻遷移：因為是 VS Code 深度定製版，所有擴展插件無縫接軌。

三、總結：你該如何組合使用？

目前的 AI 開發趨勢已經從單打獨鬥變成了「工具鏈組合」：

1. 想換感覺、重構老程式碼：用 **Claude Code** 跑 `/simplify`。
2. 要寫一個全新的、規模龐大的功能：打開 **Antigravity**，派出一組 Agent 去並行作業。
3. 在既有專案中快速修補：在 **Cursor** 裡用 Tab 和 Composer 快速切入。
4. 遇到完全沒碰過的技術領域：先在 **ChatGPT** 跑 Deep Research，再把結論餵給 IDE。

以自我介紹這個作業，建議是用編碼型 IDE 配合上通用型 AI，會最省時間與減少問題。一句話總結：

2026 年的開發者不再比誰「寫得快」，而是比誰「**Vibe** 傳達得準」以及「會挑工具幹活」。

Vibe Coding 氛圍編程的說明

這是一個在 2025 年末到 2026 年初徹底爆紅的術語，最早由 AI 大神 Andrej Karpathy 帶起風向。所謂的「Vibe Coding」，直譯就是「憑感覺寫程式」。但這不是在貶低開發者，而是在描述一種全新的開發範式**。

1. 什麼是 Vibe Coding？

過去寫程式是「語法導向」(Syntax-first)，你要精通指標、閉包、非同步語法；現在的 Vibe Coding 是「意圖導向」(Intent-first)。

- 核心邏輯：你不再一行一行敲代碼，而是透過 **Natural Language** (自然語言) 向 AI 描述你想要的「感覺」和「功能」。
- 開發者的角色：你從「打字員」變成了「產品經理」兼「資深架構師」。你負責對齊 AI 的邏輯、審核結果，並確保整體的 **Vibe** (氛圍/走向) 是正確的。

2. Vibe Coding 的三大支柱

- **Agentic Tools** (代理工具)：像 **Claude Code**、**Cursor** 或 **Windsurf**。它們不只是補完程式碼，而是能自己開檔案、改路徑、跑測試。
- **Prompting as Coding**：你寫的 Prompt 就是你的「原始碼」。如果跑出來不對，你不是去改那一行 `if`，而是去調整你的 Prompt，讓 AI 重新理解。
- 快速迭代：因為生成速度極快，你可以五分鐘內做出三個版本，測試哪一個「感覺」對了。

3. Vibe Coding 的優缺點 (很現實的)

優點	缺點(潛在危機)
極速開發：一個人能做完一個團隊的量。	技術債堆積：如果你不看 AI 寫了什麼，你會擁有一堆「能動但沒人看得懂」的代碼。
低門檻：專注於邏輯而非瑣碎語法。	除錯地獄：當 Vibe 跑偏了，而你又不曉得底層語法時，你根本不知道怎麼修。
減少疲勞：不再需要為了少一個括號煩惱半小時。	依賴性：如果 AI 斷線或模型改版，你的開發能力會瞬間減半。

 結語：這是一個「品味」勝過「體力」的時代

現在的開發者，「讀代碼」的能力變得比「寫代碼」更重要。Vibe Coding 的甜蜜點在於：你有足夠的技術底蘊去判斷 AI 給你的東西是不是垃圾，但你不再親自動手去搬那些磚頭。

「我不寫程式了，我只負責傳遞 **Vibe**。」——這就是 2026 年開發者的浪漫。

RAG (Retrieval-Augmented Generation, 檢索增強生成) 的說明。

簡單來說，RAG 就是讓 AI「先翻書，再回答」。

1. RAG 的直觀比喻：考試模式

- 純 LLM (像 GPT-4 或 Claude 3.5/3.7)：像是一個博學多聞但不能翻書的學生。他憑記憶回答問題，如果記憶模糊或沒學過 (例如你昨晚才寫好的私有程式碼)，他就會開始「一本正經地胡說八道」(幻覺)。
- RAG 模式：像是給這個學生「開卷考試」。當你問問題時，系統會先去你的資料庫 (或是你的專案資料夾) 裡找出相關的章節，把這些資料貼在考卷旁邊，讓 AI 參考這些正確資訊後再寫出答案。

2. RAG 的三個標準動作

- 檢索 (Retrieval): 當你問「這段 Login 邏輯在哪？」時, 系統會去掃描你的整個專案, 找出最像答案的程式碼片段。
- 增強 (Augmentation): 把找出來的程式碼片段, 跟你的問題「打包」在一起, 丟給 Claude。
- 生成 (Generation): Claude 看著這些證據, 回答:「喔! 你的 Login 邏輯在 src/auth/service.ts 的第 42 行。」

3. 為什麼我們現在這麼依賴 RAG?

這解決了 AI 的三大痛點:

- 知識過時: AI 的訓練資料是有截止日期的 (例如到 2025 年某月)。有了 RAG, 它可以讀取「今天」剛發布的新聞或文件。
- 私有資料: 你的公司專案、個人筆記, AI 沒讀過。透過 RAG, 你可以讓 AI 在不重新訓練的情況下「認識」這些資料。
- 減少幻覺: 因為 AI 被要求「根據我提供的參考資料回答」, 它比較不敢亂編。



總結

RAG = 搜尋引擎 (Google) + 生成式 AI (Claude/GPT)。它是目前企業應用 AI 最主流的方式, 因為「微調 (Fine-tuning) 模型」太貴且太慢, 而「建立 RAG 系統」既便宜、快速, 又能確保資料的即時性。

AI Literacy (AI 素養)

如果說 **Vibe Coding** 是你的開發風格, 那麼 **AI Literacy (AI 素養)** 就是你在 2026 年保住飯碗並持續進化的「內功心法」。在 2026 年的語境下, AI Literacy 不再只是「會用 ChatGPT 打字」, 而是演變成了「如何與非人類智慧協作」的系統化能力。我們可以把這套素養拆解成四個維度:

1. 從「對話者」轉向「調度員」(Agentic Literacy)

這是 2026 年最重要的分水嶺。

- 過去: 你問 AI 一個問題, 它給你一個答案 (Reactive)。
- 現在: 你給 AI 一個目標 (Goal), 它自己規劃路徑、調用工具、執行任務並回報結果 (Proactive)。
- 素養體現: 你是否理解什麼時候該跑同步任務? 你是否知道如何拆解一個複雜需求, 好讓 AI Agent (如 Claude Code 或 Antigravity) 不會在執行中迷路?

2. 批判性驗證 (The "Verify" Loop)

當 AI 變得越來越強大，人類最核心的素養變成了「質疑的能力」。

- 幻覺識別：AI 可能會寫出一個看起來很完美，但其實包含安全漏洞或效能陷阱的函式。
- 驗證流程：一個具備 AI 素養的人會建立一套「Ask -> Verify -> Refine -> Decide」的循環。AI 產出後，你是否有能力透過單元測試、靜態分析或邏輯推演來「審核」它？
- 警覺性：知道 AI 在哪些領域 (如極尖端的演算法或全新的 API) 容易一本正經地胡說八道。

3. 技術底層的直覺 (Technical Intuition)

你不需要會寫 LLM 的訓練代碼，但你必須直觀地理解它的運作邏輯：

- **Context Window** (上下文窗口)：知道為什麼對話太長 AI 會變笨，並學會手動「清理」上下文。
- **RAG** 意識：知道 AI 回答不準是因為它「沒讀過這份文件」，並懂得如何餵給它正確的知識庫。
- 模型差異化：知道什麼時候該請 **Claude** 出馬 (邏輯重構)、什麼時候該叫 **Gemini** (超長代碼掃描)、什麼時候用 **DeepSeek** (高頻率測試生成)。

4. 數據與倫理素養 (Governance & Privacy)

這關乎職業操守與安全。

- 數據隱私：哪些敏感的客户數據或金鑰 (Secret Keys) 絕對不能餵給 AI？
- 版權意識：理解 AI 生成內容的法律屬性，以及如何避免程式碼中的 IP 侵權風險。

總結：AI Literacy 的三個階段

階段	狀態	特徵
Level 1: 使用者	AI as a Chatbot	偶爾問問問題，把 AI 當成加強版 Google。
Level 2: 協作者	AI as a Copilot	在 IDE 裡結對編程，能熟練使用 Prompt 解決日常任務。
Level 3: 編舞家	AI as a Workforce	這就我們期望的狀態。能調度多個 Agent，進行系統化重構。

 實戰小建議

要提升 AI Literacy, 最快的方法不是讀書, 而是「觀察 AI 的失敗」。

下次當 AI 報錯或出來的結果不如預期時, 不要急著生氣, 試著分析:

1. 是我的 **Context** 給不夠?
2. 是我的 **Prompt** 語意有歧義?
3. 還是這個任務超出了目前 **LLM** 的推理邊界?

這種分析過程, 就是你累積「AI 素養」的過程 !!