



基於 Anthropic
工程師實戰架構

Agent Skills 完全攻略

從零打造最強 AI 夥伴

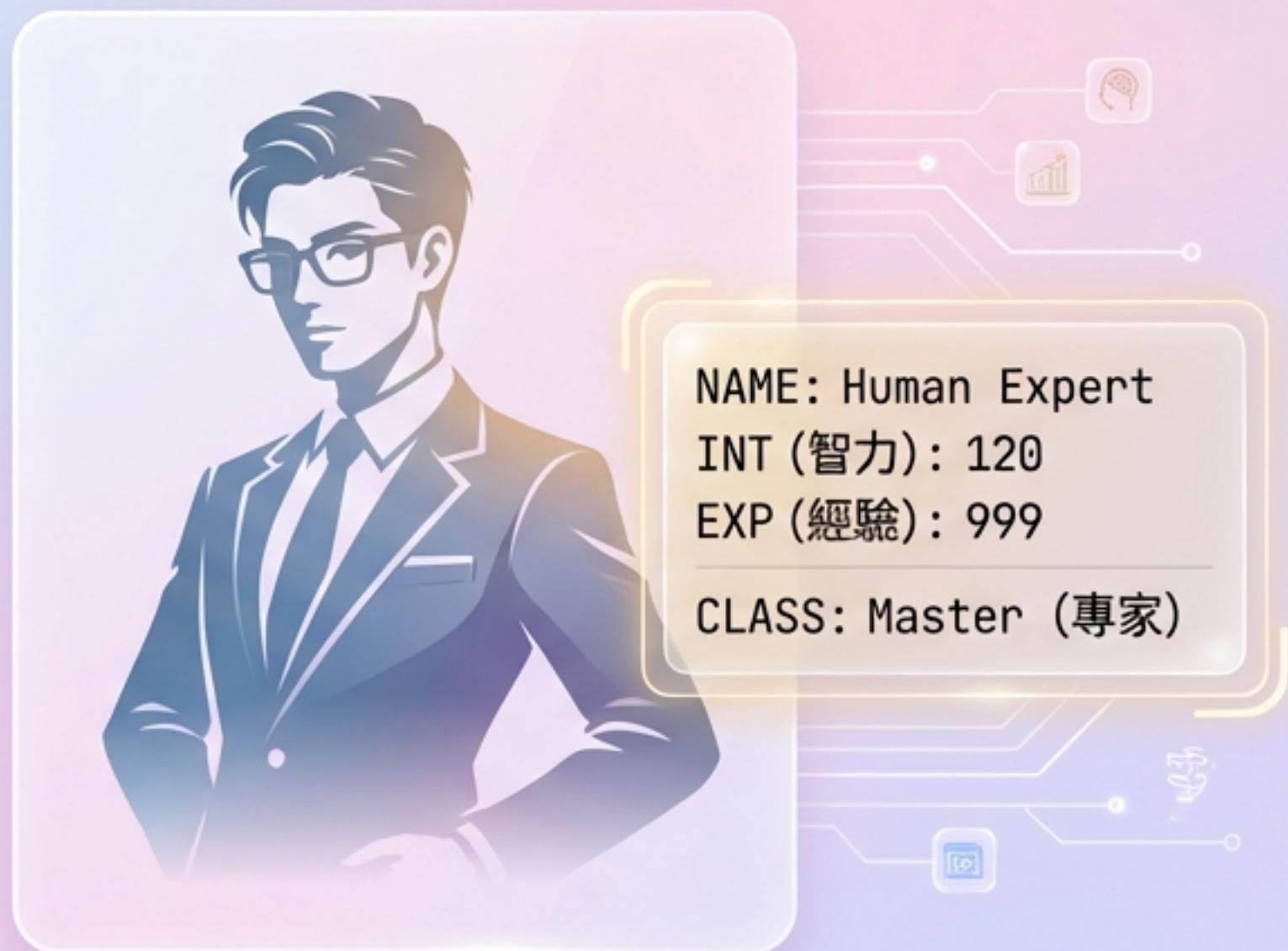


為什麼你不需要新的 Agent，而是需要更好的 Skills？

你的 Agent 是一個 IQ 300 的數學天才，但今天是它第一天上班



VS.



- 現況：Agent 極度聰明，推理能力超強，但它不懂你的公司歷史，也不懂行業潛規則。
- 問題：就像把報稅交給一個沒經驗的天才，你會放心嗎？
- 解法：你不需要更聰明的模型，你需要的是一位「專家」。Skills 就是讓 Agent 穿上裝備，累積經驗值。

計算機史的啟示：硬體再強，沒有 App 也無法工作



- CPU (Model)：潛力巨大，但單獨存在用處有限。
- OS (Agent)：負責編排資源，讓處理器發揮價值。
- App (Skills)：將領域知識編碼為可執行的軟體。沒有安裝 Skills 的 Agent，就像沒有 App 的手機。

Skills 是什麼？本質就是一個「資料夾」



- 定義：組織好的檔案集合，打包了可組合的程序性知識。
- 三大特性：
 1. 程序性：不是「知道什麼」，而是「知道怎麼做」(SOP 轉化為程式碼)。
 2. 按需載入：平時安靜躺在硬碟，不佔用記憶體 (Context Window)。
 3. 可組合：像樂高一樣，一個 Skill 可以呼叫另一個 Skill。

現代 Agent 的三角戰術架構



MCP 讓你拿得到資料，Skills 讓你知道怎麼用資料。

Scripts as Tools : 告別模糊的 Function Calling

Key takeaway at bottom: Python腳本的 Docstring 比自然語言更嚴謹，且只有需要時才載入。

傳統 Function Calling



模糊的定義，無法修改，容易出錯。

Scripts as Tools



程式碼即文件，邏輯嚴謹，可自我修正。

漸進式載入：如何塞下上千個 Skills？



- 階段一：只看目錄。Agent 就像查圖書館索引，只讀取 SKILL.md。
- 階段二：載入主程式。確認需要後，才讀取 main.py。
- 階段三：按需讀取細節。執行中需要資料才進一步載入。

讓有限的 Context Window 發揮最大效益。

企業內部 Skills (Enterprise)

第三方夥伴 Skills (Partner)

基礎 Skills (Foundation)



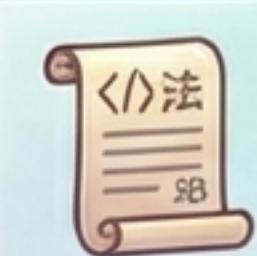
Skills 的生態系金字塔

- 基礎層：通用能力，任何領域都用得到。
- 邊緣層：由外部服務商建構，深度整合平台。
- 企業層：最具競爭力的資產。將你公司獨有的流程轉化為程式碼。

動手做：打造你的第一個 Skill (以台灣營業稅為例)

Crafting Bench

Ingredients



Business Logic
(稅法)



Python



Template



Result



taiwan-vat-skill

```
VAT_RATE = 0.05 # 台灣一般稅率
def calculate_vat(amount):
    return amount * VAT_RATE
```



- 情境：財務部每季需要計算 5% 營業稅。
- 結構：SKILL.md (定義時機) + calculator.py (計算公式) + validators.py (格式驗證)。
- 成果：將重複性的人工運算，封裝成標準工具。

SKILL.md：決定 Agent 是否能找到裝備的關鍵

- 這是漸進式載入的唯一入口。
- 寫作公式：[動詞] + [具體任務] + [適用情境] + [不適用情境]。

Bad Description

Name: Tax Tool

Desc: Helps with tax.



Good Description

Name: Taiwan Tax Filing

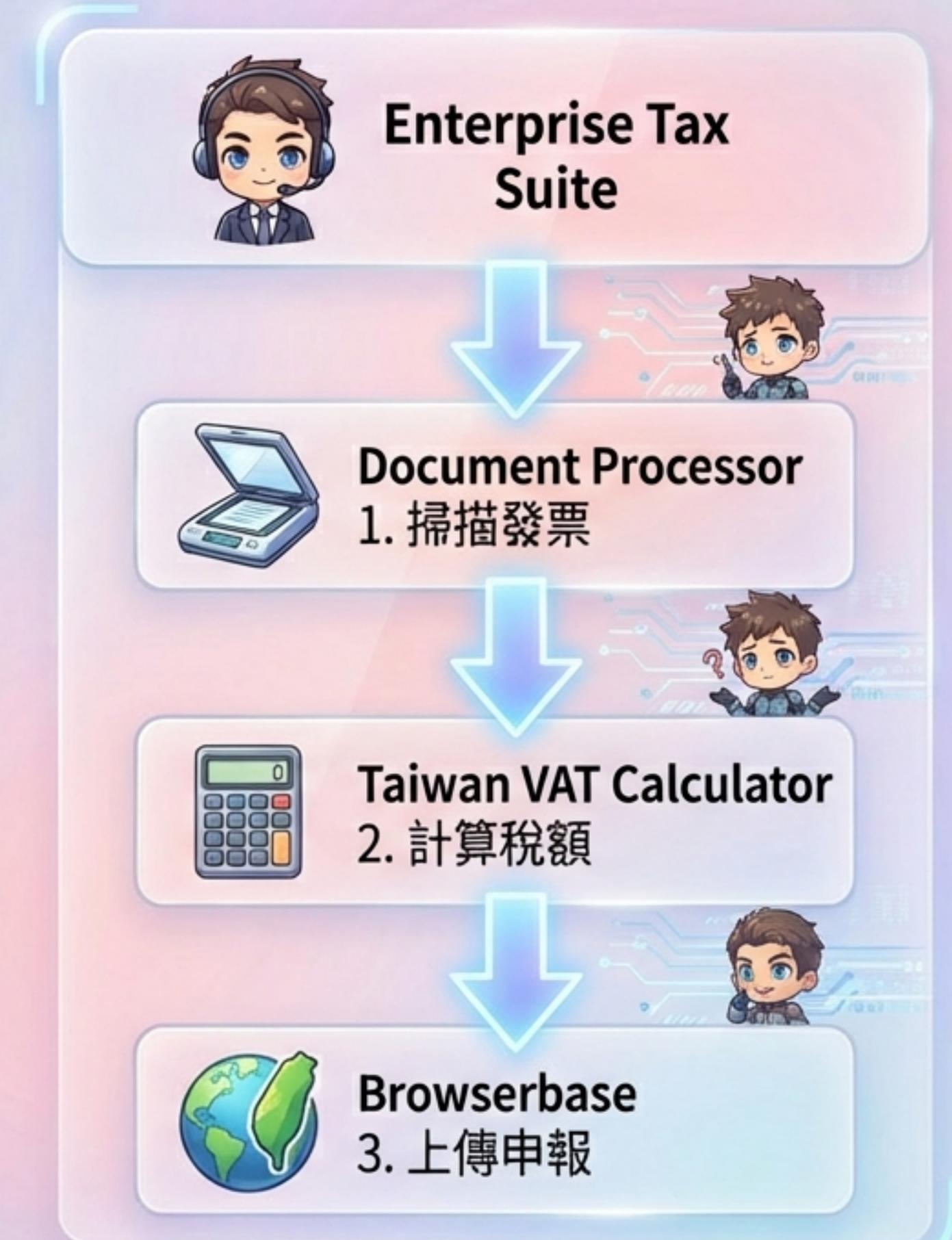
Desc: 處理台灣企業所得稅申報，支援統一發票核對，適用每年 5 月申報季。

Check

A cartoon illustration of the same boy from the first section. He is now smiling and holding a magnifying glass over a document that has a green checkmark on it. The word "Check" is written in large, green, bubbly letters above him.

可組合性：像樂高一樣 堆疊你的專業

- **單一職責原則：**每個 Skill 只做一件事。
- **層級呼叫：**企業邏輯 Skill → 呼叫通用 Skill。
- **優勢：**當稅法改變，只需更新 Calculator Skill，不用重寫整個流程。



企業實戰：將「死」的文件轉化為「活」的 Skills



Complaint SOP v3.2 (PDF)



handle_complaint.py



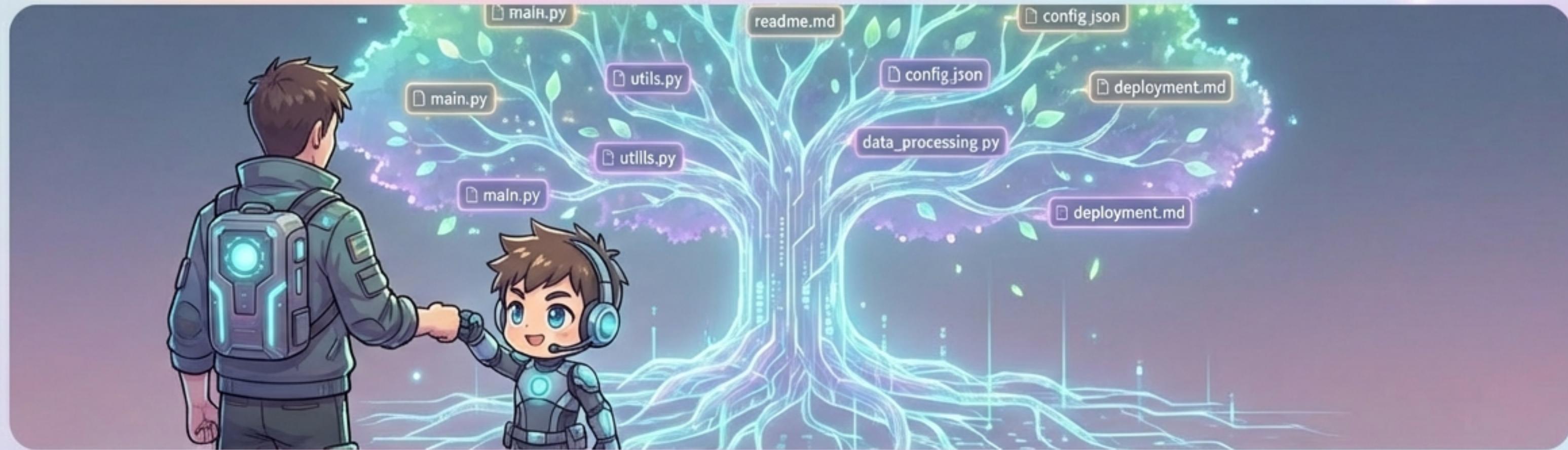
- **過去**：SOP 躺在 Google Drive 裡，只有新人入職看一次。
- **現在**：SOP 變成 Python 腳本，每次客訴發生時自動執行。
- **範例**：客訴分級 SOP → 自動判斷 A/B/C 級並設定 SLA 截止時間。

像管理軟體一樣管理 Skills

- **版本控制 (Git)**：追蹤每一次知識的演進 (v1.0 -> v2.0)。
- **自動化測試 (CI/CD)**：確保 Agent 能在正確時機載入正確 Skill。
- **品質柵欄**：任何 Skill 修改都必須通過測試才能 Merge。



終極願景：人機共創的活體知識庫



- **雙向優化**：人類建立框架，Agent 執行並發現邊界案例。
- **自我修正**：Agent 可以修改腳本以適應新情境，人類審核變更。
- **複利效應**：知識不再是靜態的，而是隨著每次任務執行而不斷變強。

Skill = Folder + Python + Context



SKILL.md is
the Map (地圖)



Scripts are
the Tools(工具)



Git is the
History (歷史)



快速參考卡：Agent 育成心法

- 別重造 Agent：用 Skills 疊加能力。
- 程式碼即文件：用 Python 寫 SOP。
- 漸進式載入：只給 Agent 當下需要的資訊。



「複利的本質，是疊加，不是重來。」

