AI 輔助開發基礎課程 (2026)

簡報人：賴志皓

對象：BE PTE 工程師

目標：建立 AI 協作思維，掌握內部工具「AI 小旺」

今日學習地圖

從觀念建立到實戰自動化

1. 現狀與資安

為什麼需要小旺？資安紅線在哪裡？

2. 基礎工具

Markdown 與 HedgeDoc 實戰。

3. 心態與風險

理解 SE/TE 差異與對抗 AI 幻覺。

4. 提示詞工程核心

掌握 Context 與系統提示詞。

5. 實戰演練

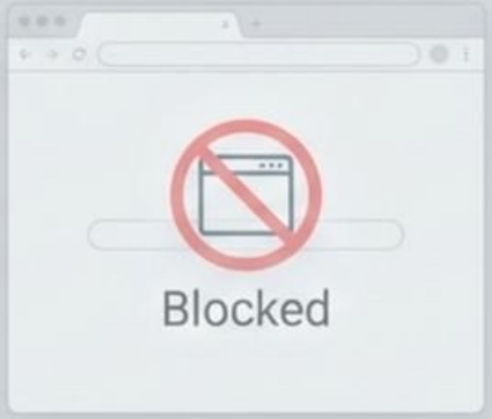
小旺 Coding 模式與進階開發工作流。

現狀與挑戰

填補官方 UI 的功能缺口

官方限制 (Official Web UI)

- 無法操作單則對話 (刪/改/重跑)
- 無法讀取專案目錄
- 無法分析圖片



小旺優勢 (AI Little Wang)

- Coding 模式專屬優化
- 多模態整合
- 專為工程師設計的操作流程

資安考量：限制外部 AI 服務，僅能使用公司內部 GPT-5 API.

資安意識：如何安全地餵資料？

在保護公司資產的前提下使用 AI

核心原則：API 視同外部服務

雖然工具是內部開發，但底層串接 OpenAI API，資料會離開公司內網。

脫敏原則 (De-identification)

- **關鍵字替換**：將專案代號改為 `Project_A`，將良率數字改為變數。
- **核心 IP 保護**：禁止上傳完整架構文件，僅擷取需除錯的程式邏輯片段。
- **Input 過濾**：所有的 Prompt 必須經過脫敏處理才可發送。

基礎工具：Markdown 與 HedgeDoc

溝通與紀錄的標準格式

數位轉型行動

Drop: Word / PowerPoint
(用於過程紀錄)

Adopt: Markdown / HedgeDoc

Benefit: 無縫銜接 AI 協作流程

小旺匯出實戰

小旺對話 → 整串匯出功能 → HedgeDoc 專業報告

AI 友善格式：Markdown 是純文字結構，比 Word/Excel/PPT 更容易被 AI 解析與精準處理。

Markdown 基礎語法

工程師必備的標記語言

Input (Source)

```
# 標題層級

- 清單項目 A
- 清單項目 B

**粗體重點**
```cpp
int main() {
    return 0;
}
```
```

Output (Render)

標題層級

- 清單項目 A
- 清單項目 B

粗體重點

```cpp
int main() {
    return 0;
}
```

進階應用：支援 Mermaid 語法，可直接繪製流程圖。

HedgeDoc 實戰演練

打造團隊的技術知識庫

即時協作

多人同時編輯，適合會議記錄與技術討論。

內網安全

架設於 BE 內部環境，資料不外流。

Publish 模式

一鍵生成唯讀網頁連結，方便跨部門分享。

自動化工作流

1. 小旺對話完成 >> 2. 點擊「整串匯出」 3. 貼上 HedgeDoc >> 4. 調整標題 -> 完成專業技術報告

TE 開發與 AI 的磨合

理解硬體限制與軟體思維的差異

SE 軟體思維 (Cloud/OS)

- **訓練資料**：Modern C++ (C++11/14/17/20)
- **資源**：動態配置，豐富的 STL 支援
- **目標**：架構彈性與可維護性

TE 硬體思維 (ATE/DUT)

- **訓練資料偏差**：機台環境常限制於 C++98 或特定編譯器。
- **資源管理**：禁用 STL，需手動管理記憶體，避免動態配置。
- **核心 KPI**：TTR (測試時間) 與 Yield (良率) 是絕對命脈。

AI 不是萬能的

在 ATE 環境下的風險控制

警惕：AI 幻覺 (Hallucination)

AI 會以極度自信的語氣，說出完全錯誤的資訊。

TE 常見陷阱

- **虛構 API**：在 Advantest 環境寫出 Teradyne 的 API。
- **語法混用**：跨廠商語法張冠李戴。
- **忽略硬體特性**：忽略 `volatile` 關鍵字導致編譯器優化錯誤。

對策：驗證為王，交叉比對，保持懷疑。

AI 的語言天賦：為什麼它會張冠李戴？

理解訓練資料對 AI 能力的影響

Assembly (精通級)

Public Data

資料來源：x86/ARM 公開資料豐富。

能力：擅長解釋反組譯代碼與優化指令。

ATL (風險區)

Proprietary/NDA

資料來源：封閉性語法，公開資料極少 (NDA)。

風險：AI 極易產生 API 幻覺。

工程師對策：邏輯交給 AI，硬體控制 API 必須由人工提供範例 (Few-Shot)。

Context 污染：AI 為什麼會「歪樓」？

AI 的「即時記憶」與「理解邊界」

主動管理

- 不要試圖糾正嚴重錯誤。
- 直接刪除那則錯誤的回覆。
- 保持 Context 的純淨。

AI 的記憶是「盲目」的，它會把之前的錯誤
當成正確的背景資訊 (Context)。

Token：AI 的計量單位

為什麼小旺限制 600KB？

換算公式

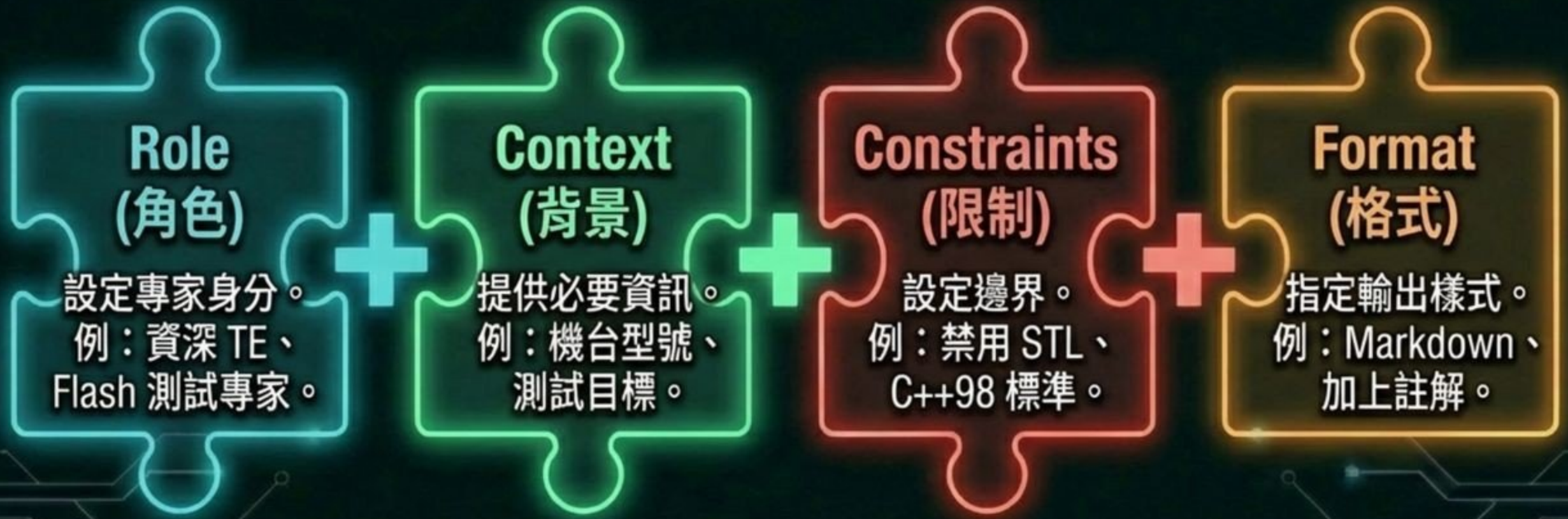
- 1000 Tokens ≈ 750 英文字
- 600KB 程式碼 ≈ 15~20 萬 Tokens

避免溢位 (Overflow)

600KB 的限制是為了確保 AI 在讀取代碼後，還有足夠的空間處理對話歷史與系統提示詞，避免 AI 出現「失憶」。

提示詞工程 (Prompt Engineering)

優質 Prompt 公式結構



AI 的靈魂設定

系統提示詞 (System Prompt)

System Prompt Configuration

```
You are an expert Test Engineer.
Rule 1: No Hallucination.
Rule 2: Only use provided API examples.
```

全域設定

System Prompt 是持續影響整場對話的邏輯與風格設定。

ATL 專用規範範本

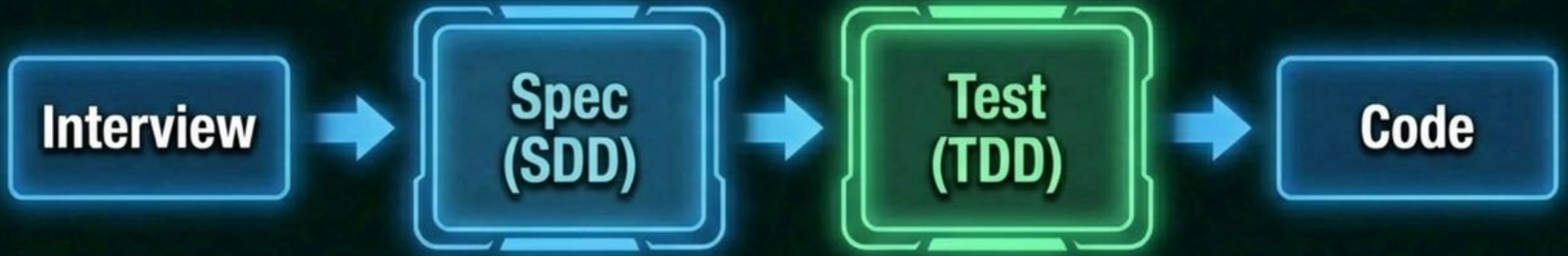
- 強制設定：「No Hallucination」。
- 行為準則：「針對機台控制 API，只能使用我提供的範例 (Few-Shot)，禁止自行創造」。

TE 常用範本

- ATE 審查標準
- Flash 規格實作規範
- 老舊編譯器相容性要求

規格驅動與測試驅動開發

對抗幻覺的最佳防禦



SDD (Spec-Driven Development)

讓 AI 先幫你整理 Datasheet 規格與邏輯，確認無誤後，再根據該規格生成程式碼。

TDD (Test-Driven Development)

針對純邏輯（如 BBM 演算法），先要求 AI 撰寫單元測試 (Unit Test)，再生成實作代碼。

與 AI 溝通的進階武器

讓 AI 輸出更精準、更有邏輯

1. 反向提問

讓 AI 採訪你，協助釐清模糊的需求描述。

2. 思維鏈 (Chain of Thought)

指令：「請一步步思考」，強制 AI 寫出推理過程，降低邏輯錯誤。

3. 範例引導 (Few-Shot)

給予 1-2 個正確的代碼範例讓 AI 模仿風格與語法。

4. 任務拆解

將複雜的大型任務拆解為數個小步驟逐一執行。

AI 小旺：Coding 模式實戰

從讀取到「一鍵套用」的完整閉環

硬體安全審查：涉及 `WriteRegister` 等寫入操作，Apply 前務必人工確認時序。

實戰案例分享

直接複製，立即上手 (TE 專用 Prompt 範本)

【重構】

將 Magic Numbers 轉換為具名常數，提升可讀性。

Copy

【驗證】

使用 Python 撰寫 C++ 演算法的對照組 (Oracle) 進行邏輯驗證。

Copy

【文件】

讀取 Header 檔，自動生成標準 API 文件。

Copy

【規格】

將 Datasheet 描述文字直接轉換為 C 語言結構或函式。

Copy

【分析】

自動化處理 STDF/CSV 數據，生成 Shmoo Plot 分析腳本。

Copy

結語：主導 AI 協作

建立以人為核心的開發閉環

角色轉型

從 Coder 轉型為 Architect 與 Reviewer

核心競爭力

- 領域知識 (Domain Knowledge)
- 問題定義能力
- 嚴格的審查與驗證 (Verify)

安全底線

- 工具是內部的，但 API 是外部的。

脫敏 (De-identification) 是不可逾越的紅線。

下一步

- 善用 AI 小旺與 HedgeDoc，開始你的高效開發之旅。