

## CHAPTER 2

# AGENT SKILLS

HORAZON

ANTIGRAVITY

# PART 1: 基礎概念 (THE WHY)

在了解 Skill 之前，我們必須先理解 AI 是如何「思考」與「運作」的。

## 1. PROMPT ENGINEERING (提示工程)

- AI 不是魔法水晶球，它更像是一個聽話但需要精確指令的實習生。
- **Prompt (提示)**：你給 AI 的指令。
- **Output (輸出)**：AI 紿你的回應。
- **品質關鍵**：垃圾進，垃圾出 (Garbage In, Garbage Out)。指令越明確，結果越好。

# CONTEXT WINDOW (上下文視窗)

## AI 的「短期記憶」

- AI 記不住所有的事情。它只能看到你「現在」餵給它的資訊。
- 這個「可見範圍」就是 Context Window。

## 為什麼不把整座圖書館塞進去？

1. Token 限制與費用：越多字越貴，且有長度上限。
2. 雜訊 (Noise)：資訊太多會讓 AI 分心，抓不到重點。
3. 遺忘 (Forgetting)：根據 "Lost in the Middle" 現象，太長的文件中間容易被忽略。

| 解決方案：AI 需要一種方式，在「需要時」才去獲取特定的知識。

# CONTEXT ENGINEERING (上下文工程)

## 定義

不僅僅是「寫好提示詞 (Prompt Engineering)」，而是系統化地設計 AI 接收的資訊環境。

## 核心概念

- **提供背景**：讓 AI 知道它是誰、任務是什麼 (System Instructions)。
- **注入知識**：自動提供相關文件、程式碼片段 (RAG)。
- **設定規則**：明確規範什麼能做、什麼不能做 (User Rules)。

Prompt 是戰術 (單次提問技巧)，Context 是戰略 (整體資訊佈局)。

# SYSTEM INSTRUCTIONS (系統指令)

## 賦予 AI 「角色」

- 在對話開始前，我們會給 AI 一個 Persona (人設)。
- 例如：「你是一個資深的 Unity 工程師，請用繁體中文回答...」

## AGENT 運作核心

- 一般 LLM：看完指令 -> 回答文字。
- Agent：看完指令 -> 思考 (Reasoning) -> 使用工具 (Use Tools) -> 得到結果 -> 回答文字。

| Skill 就是我們塞給 Agent 的強力工具箱。

# PART 2: 什麼是 AGENT SKILL?

## 定義

- Agent Skill 是一包預先定義好的專門知識或標準作業程序 (SOP)。
- 它是一組指令與檔案，教導 Agent 如何處理特定的任務。

## 類比

- 瀏覽器 (Browser) → 擴充套件 (Extension)
- 遊戲角色 (Player) → 技能書 (Skill Book)
- 工程師 (Developer) → API 手冊 (Docs)

# 發展歷史 (HISTORY)

時間點	事件	影響
2025/10	Anthropic 推出 Agent Skills	功能公測 (Beta)：首度展示 Agent Skill，能執行特殊任務。
2025/12	Anthropic 發布 Agent Skills	協議開源 (Open Source)：正式發布，定義標準化介面。
2026/01	Antigravity 引入 Skills	Google 團隊將此概念整合進 VS Code Agent，讓開發者能自定義 AI 的能力。

## 為什麼這很重要？

- 過去：等原廠更新功能。
- 現在：自己寫 Skill，讓 AI 學會你團隊專屬的 Workflow。

# PART 3: 如何建立與獲取 SKILL? (THE HOW)

我們有三種途徑來獲得 Skill：

1. Create (自己寫)：量身打造專屬工具。
2. Discover (找現成)：使用社群貢獻的 Skills。
3. Generate (叫 AI 寫)：用魔法打敗魔法。

# 途徑一：自己寫 (CREATE)

一個 Skill 通常包含為了達成任務所需的 Prompt 與 Scripts。

## 檔案結構

在 `.agent/skills/` 目錄下建立資料夾：

```
.agent/
  └── skills/
    └── translate/      <-- Skill 名稱
      └── SKILL.md     <-- 核心定義檔
```

## SKILL.MD 格式

- Frontmatter (YAML)：定義名稱、描述。
- Instructions (Markdown)：教 AI 怎麼做。

# 實作範例：翻譯助手 (TRANSLATE SKILL)

此 Skill 讓 AI 能精準可控地進行多語言翻譯。

```
---  
name: translate  
description: 專門負責中英翻譯的技能
```

```
# Translate Skill
```

當使用者要求「翻譯」或「Translate」時，請遵守以下規則：

1. \*\*用語精確\*\*：使用台灣繁體中文 (zh-TW) 慣用語。
2. \*\*保留專有名詞\*\*：如 "Unity", "Component" 不需翻譯。
3. \*\*格式保留\*\*：不要破壞原始 Markdown 格式。

# 途徑二：找現成 (DISCOVER)

你不一定需要從頭開始造輪子。

## 資源來源

1. **Claude MCP / Skills** : Anthropic 官方或社群維護的列表。<https://skills.sh/>
2. **Git Hub Open Source** : 許多開發者公開分享他們的 Prompt。
3. **Team Shared** : 資深工程師寫好 Skill，透過 Git 分享給全團隊使用 (如 Coding Style 檢查)。

# 途徑三：叫 AI 寫 (GENERATE)

這是最快的方法！直接命令 Antigravity 幫你產生。  
(推薦安裝 <https://skills.sh/> 的 skill-creator)

## 範例指令

"Hey Agent, 我想要一個翻譯成中文的 Skill。請幫我建立一個 `translate_cn` skill，當我輸入英文時，自動幫我翻譯成流暢的繁體中文。"

## AI 的動作

1. AI 讀取你的需求。
2. AI 自動在 `.agent/skills/translate_cn/` 建立 `SKILL.md`。
3. AI 自動填入詳細的翻譯規則與 Prompt。

你就獲得了一個新的 Skill！

# 進階結構：不僅僅是 PROMPT

複雜的任務需要更強大的支援。

## 1. SCRIPTS (腳本)

- **用途**：讓 AI 能執行程式碼。
- **例子**：`scripts/translate.py` (呼叫 Google Translate API)、`scripts/verify_build.sh` (檢查建置狀態)。

## 2. RESOURCES (資源/參考資料)

- **用途**：提供 AI 額外的知識庫 (RAG) 或範本。
- **例子**：`resources/style_guide.md` (Coding Style)、`resources/api_docs.json` (API 文件)。

```
.agent/skills/super_skill/  
  └── SKILL.md           <-- 指令  
  └── scripts/  
      └── data_process.py  <-- 工具程式
```

# 結語：賦能 (EMPOWERMENT)

Agent Skill 將 AI 從「通用的聊天機器人」轉變為「專精的領域專家」。

- **累積知識**：把團隊的 Know-how 寫成 Skill。
- **自動化**：把重複的 SOP 寫成 Skill。
- **擴充性**：隨著專案成長，讓你的 AI 隊友也一起升級。