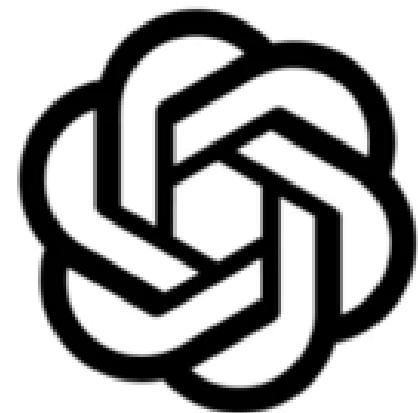


# **Introduction to LLM Agent & LangGraph**

Lorrin

# Back to ChatGPT-3.5 ...



ChatGPT-3

## 推理能力有限

雖擅長文本生成，但在複雜的數學運算及邏輯推理方面表現不佳。

## 知識時效性

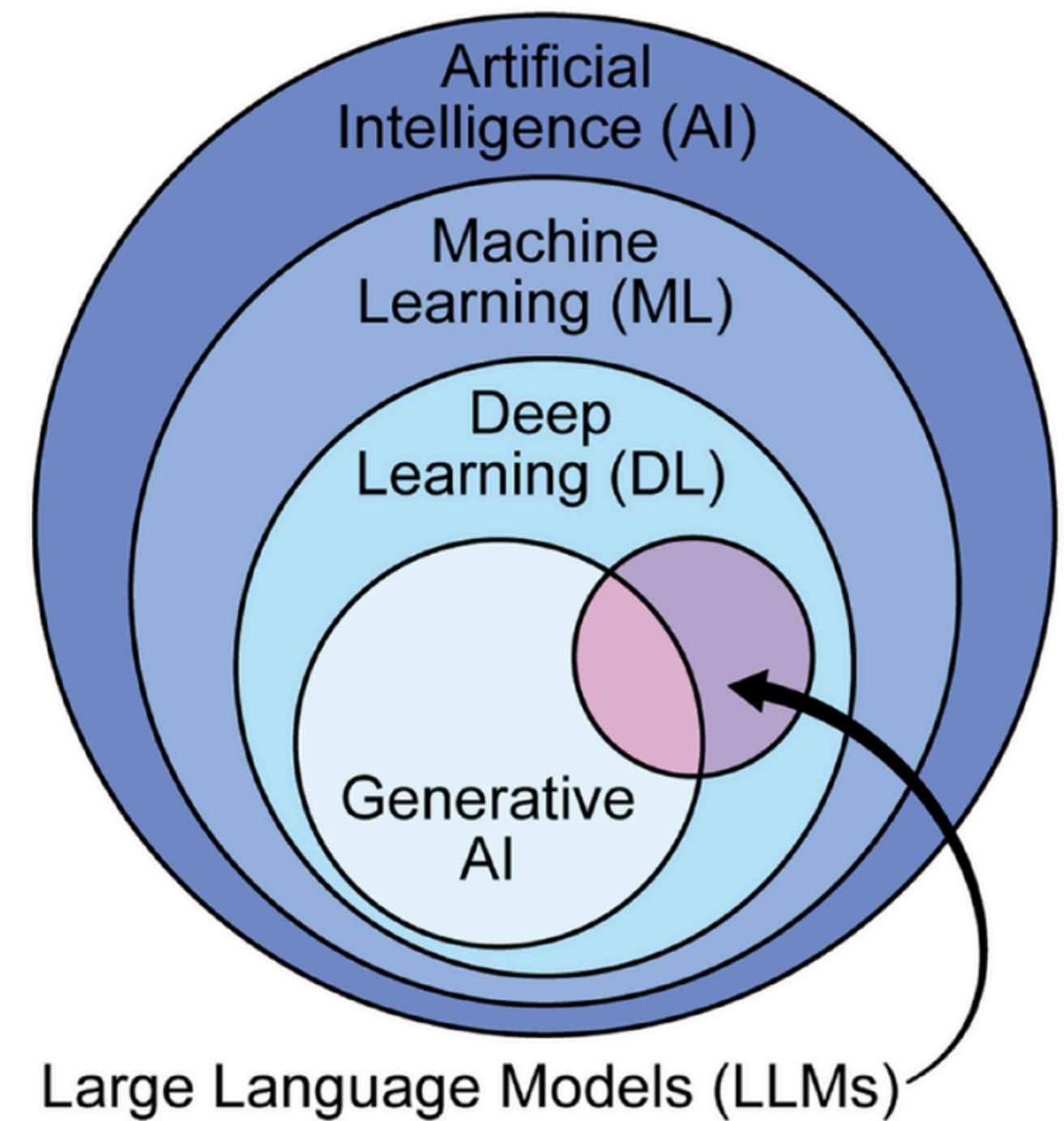
知識來自訓練數據，無法即時更新。

## 環境互動受限

雖擅長文本生成，但在複雜的數學運算及邏輯推理方面表現不佳。

# LLM Agent ? AI Agent ?

What's difference?



# What's LLM Agent ?

2024年12月出版

- ✓ 喜鵲是狗嗎? ➡➡➡
- ✗ 我還剩幾天假? ➡➡➡



# What's LLM Agent ?

2024年12月出版

✓ 喜鵲是狗嗎? ➤➤➤

✗ 我還剩幾天假? ➤➤➤

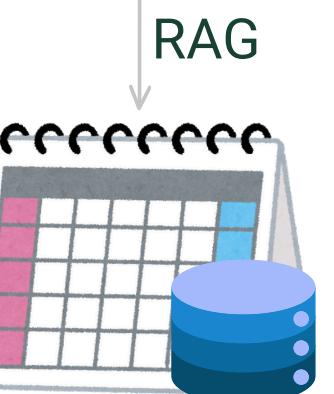


✓ 喜鵲是狗嗎? ➤➤➤

✓ 我還剩幾天假? ➤➤➤

✗ 今天天氣如何? ➤➤➤

2024年12月出版



# What's LLM Agent ?



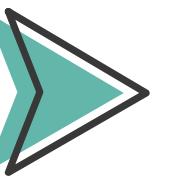
# What's LLM Agent ?

簡單來說，LLM Agent 就像一個更主動、更有彈性的 AI 助理。  
它不只是被動地遵循指令，而是**能夠自己思考、規劃、使用工具，**  
並從錯誤中學習，來完成你交辦的、甚至是比較複雜的任務。

# LLM Agent 核心能力

## 規劃 (Planning)

將一個宏觀、複雜的目標拆解成一系列具體、可執行的子任務或步驟的能力。



例如：使用者說「幫我規劃一趟東京五日自由行」

1. 拆解成每日行程規劃 → 景點推薦 → 路線安排
2. 串接地圖、氣象、預算等工具輔助決策
3. 結合記憶與工具，逐步產出完整方案

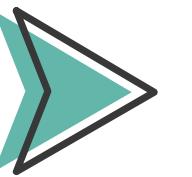
# LLM Agent 核心能力

## 規劃 (Planning)

將一個宏觀、複雜的目標拆解成一系列具體、可執行的子任務或步驟的能力。

## 記憶 (Memory)

在執行任務的過程中，能夠儲存、檢索和利用相關資訊的能力。



包含：

1. 短期記憶（當前對話內容）
2. 長期記憶（個人偏好、歷史行為）

# LLM Agent 核心能力

## 規劃 (Planning)

將一個宏觀、複雜的目標拆解成一系列具體、可執行的子任務或步驟的能力。

## 記憶 (Memory)

在執行任務的過程中，能夠儲存、檢索和利用相關資訊的能力。

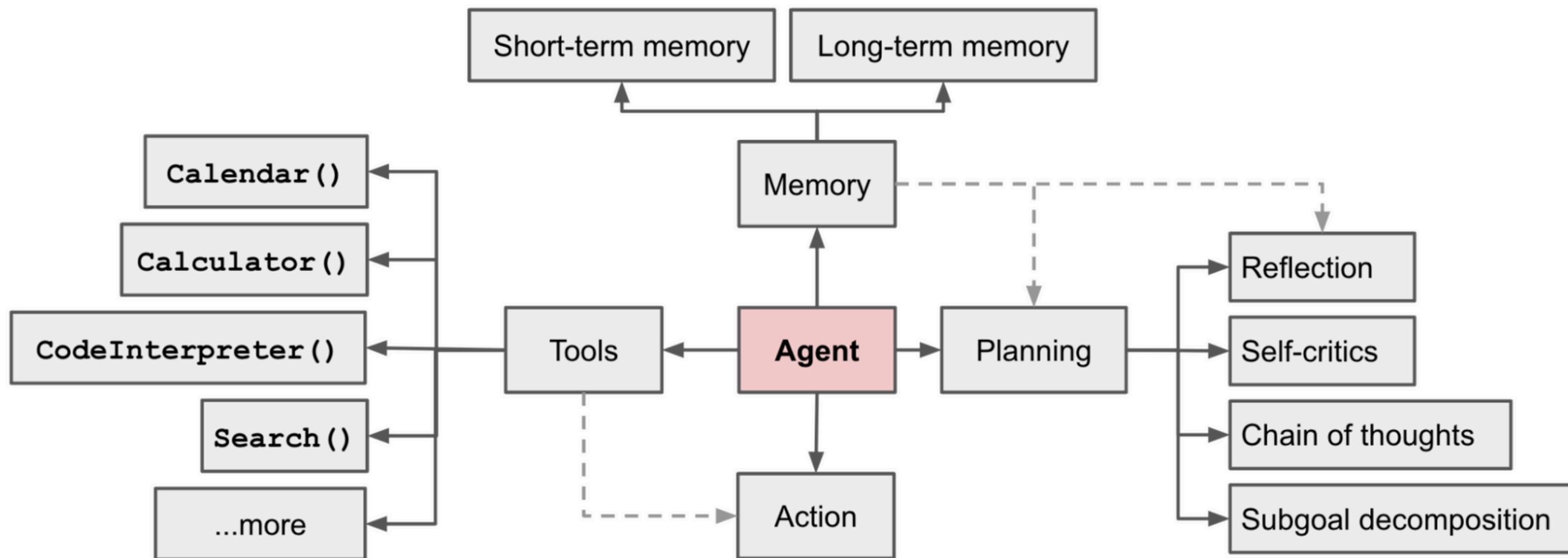
## 工具使用 (Tool Use)

能夠判斷何時需要藉助外部工具來獲取資訊或執行動作，並能有效地調用這些工具、理解其返回結果的能力。

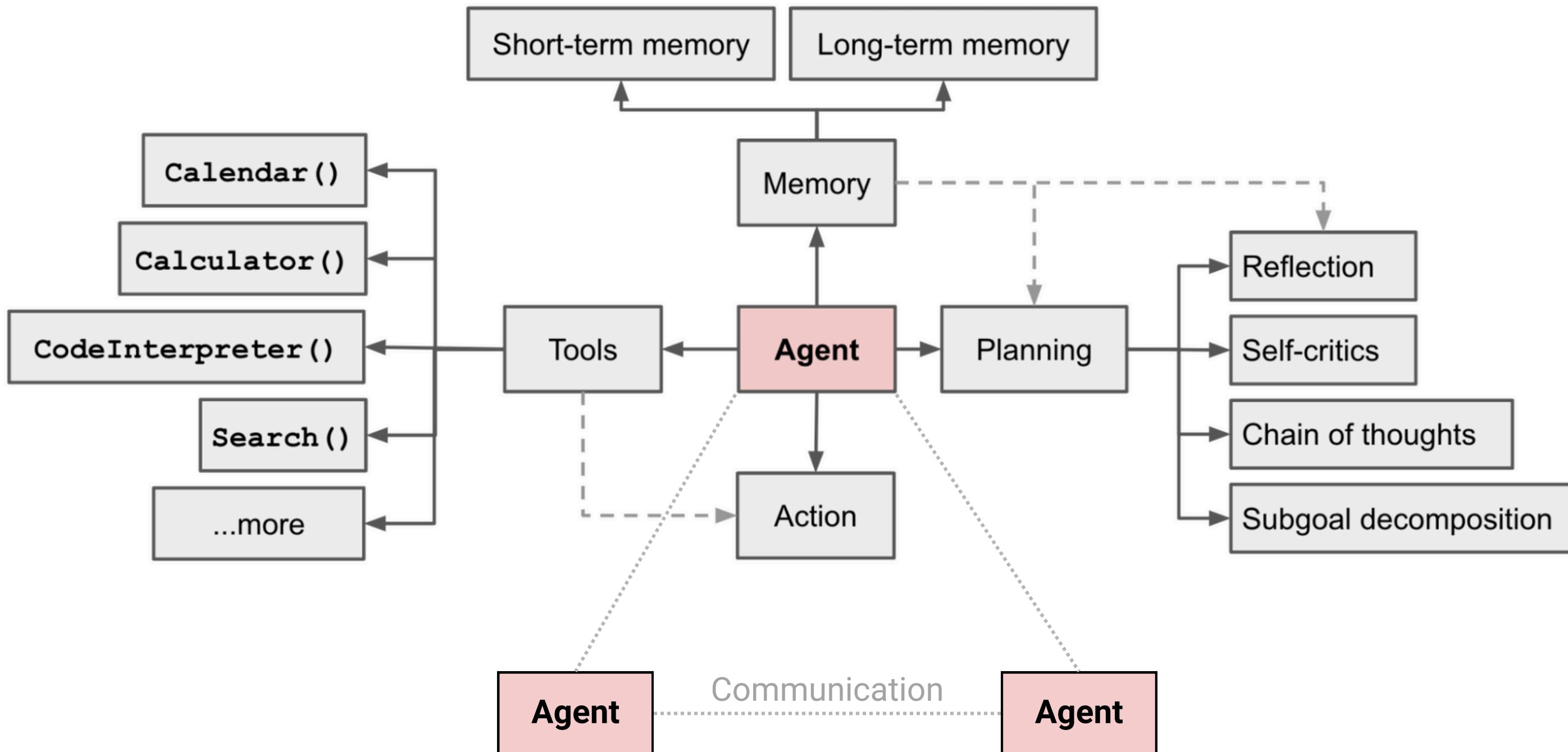


例如：遇到查匯率的問題，它會自動呼叫 API 查詢而不是直接亂掰答案  
工具可以是：Google 搜尋、資料庫查詢、計算機、甚至觸發內部系統流程

# LLM Agent 架構



# Multi Agent 架構



# Chatbot vs LLM Chatbot vs LLM Agent

	傳統 Chatbot	LLM Chatbot	LLM Agent
回應方式	基於規則/關鍵字匹配	基於大型語言模型的自然語言理解	同上，但能進一步執行任務/決策
彈性與理解力	低，無法應對變化語句	高，可理解多樣表達	更高，理解上下文並 <b>做多步推理</b>
記憶能力	無記憶，回合制互動	可短暫記住對話上下文	<b>支援長期記憶</b> ，可記住用戶偏好、任務歷史等
工具使用	無	無（單純對話）	<b>可調用工具</b> （API、資料庫、外部系統）
流程控制能力	無條件判斷，流程固定	無明確流程控制	有狀態、有流程圖（如 LangGraph）控制複雜任務



LangGraph

Studio

# What's LangGraph ?

LangGraph 是一個專門用來建立有**狀態、多代理應用程式**的 Python 函式庫，  
它就像是幫你的 AI 代理團隊畫了一張工作流程圖。

它建立在大家比較熟悉的 **LangChain 基礎之上**，但特別強化了處理多個 LLM 或代理需要協同工作，  
並且需要記住「之前發生什麼事」（也就是狀態管理）的場景。

# LangGraph 核心元件

## 圖形(Graph)

就像是一張流程圖或地圖，是整個 AI Agent 執行邏輯的主架構。

## 節點 (Nodes)

圖上的每一個點，你可以把它想像成一個「工作站」。

這個工作站可以是一個 AI 代理、一個 LangChain 鏈、一個普通的 Python 函數。

有的節點負責接收用戶指令，有的負責查資料庫，有的負責呼叫 LLM。

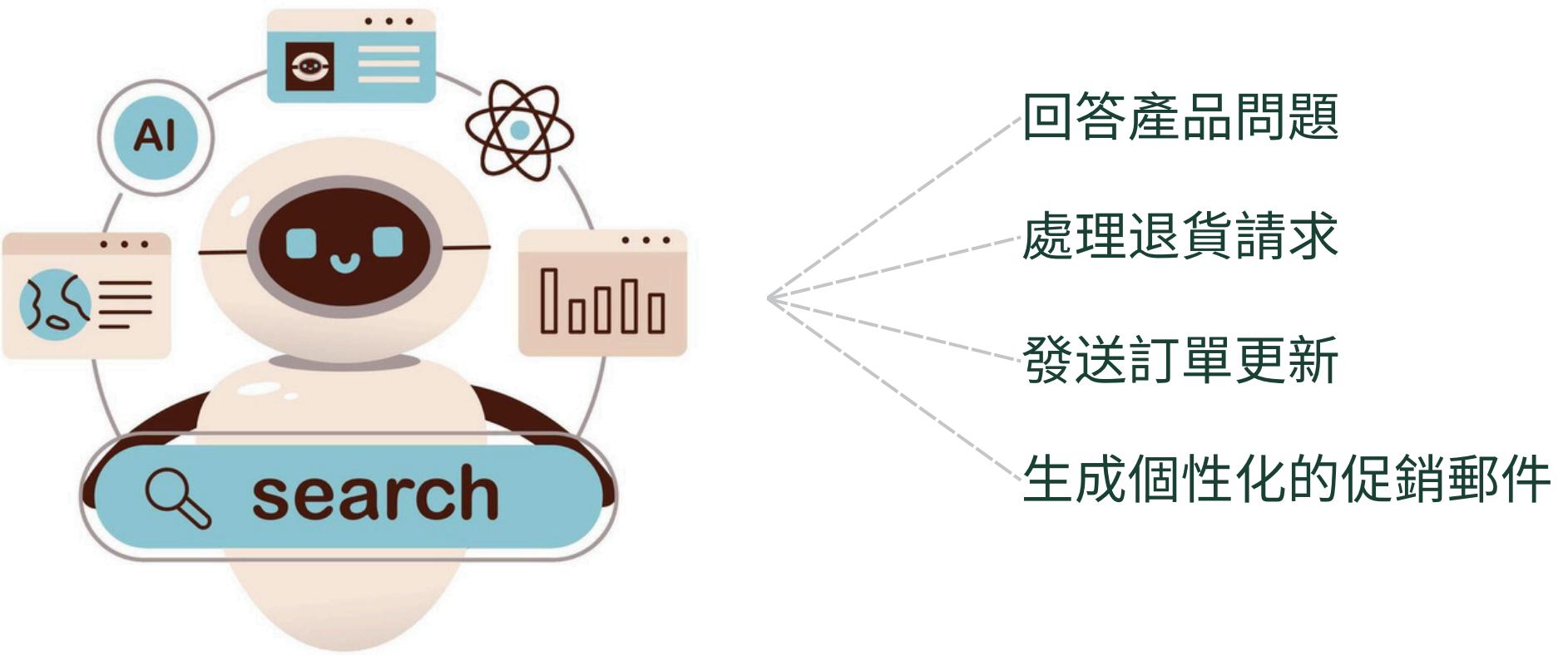
## 邊 (Edges)

連接這些工作站的箭頭，就是「邊」。它定義了工作的順序和方向，告訴你資料或指令該往哪裡走。

## 狀態(State)

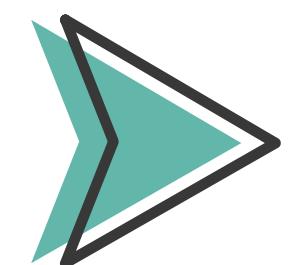
是整個流程中「攜帶的資訊」，每個節點(nodes)都可以讀取或更新這個狀態。

**So, why LangGraph? Why not just LangChain?**



一個簡單的客戶查詢可能演變成一系列複雜的操作：

1. 檢查訂單狀態
2. 如果訂單延遲，生成道歉信
3. 同時啟動退款流程
4. 最後，發送一個個性化的折扣碼來挽留客戶



### LangChain

採線性流程，難以應對條件跳轉與狀態維護

### LangGraph

擅長處理複雜、多步驟、具交互性的工作流

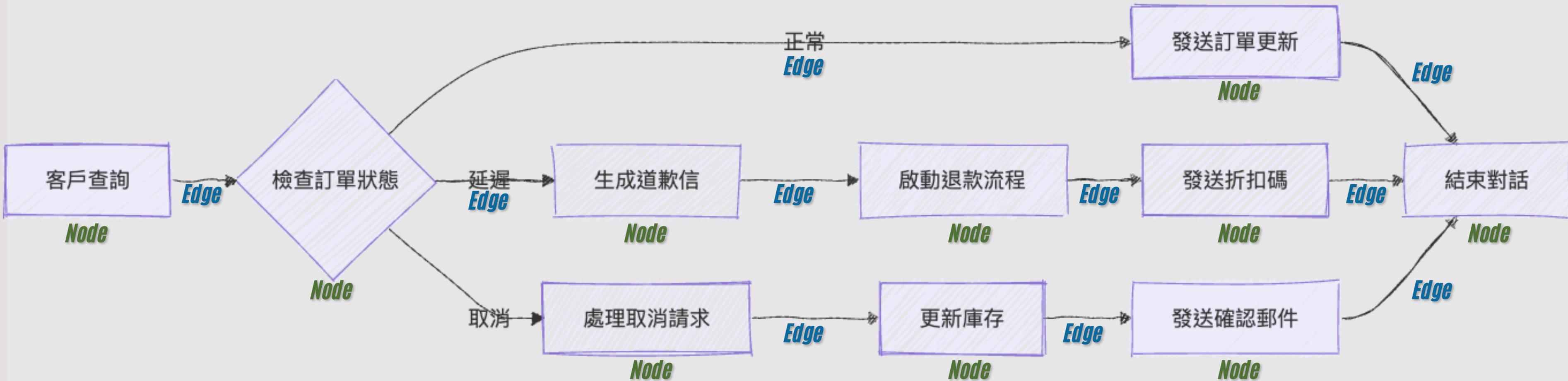
**LangGraph** 將你的整個工作流程視覺化為一個圖形 (Graph):

- 每個任務 (如檢查訂單、生成郵件、處理退款) 都是圖中的一個節點 (Node)
- 節點之間的連接邊 (Edge) 定義了任務的執行順序和條件
- 整個系統共享一個狀態 (State)，使得信息可以在不同任務間無縫傳遞



支援條件分支、迴圈與任務並行，  
能更靈活打造更智慧且高回應性的 AI 系統

**Graph**



The **state** is updated at each **Node**

# LangChain vs LangGraph

	<b>LangChain</b> 	<b>LangGraph</b> 
工作流結構	主要是 <b>線性</b> 的，適合順序執行的任務	基於圖的結構，適合複雜、 <b>非線性</b> 的工作流
靈活性	在處理條件分支和循環時較為受限	可以 <b>輕鬆定義條件路徑</b> 、循環和並行任務
狀態管理	在跨多個步驟維護狀態時可能遇到困難	提供了一個 <b>共享的狀態系統</b> ，方便在整個工作流中傳遞信息
可視化	工作流程可能不太直觀	將工作流 <b>視覺化為圖</b> ，使複雜系統更易於理解和設計
使用場景	適合相對簡單、直接的AI任務	適合需要複雜決策和多路徑執行的高級AI系統

LangGraph 並不是要取代 LangChain，而是對其進行補充和擴展。  
在許多情況下，你可能會發現**將兩者結合使用是最佳選擇**。

**1**

曉瑛

博丞

Buck

Mickey

瑄佑

**2**

Dorbe

Dicy

Roy

志偉

佳樺

奇緣

**3**

郁雯

芳均

Miles

Ken

Peggy

**4**

巧君

Ian

立晉

韻如

浩軒

**5**

小嫿

Abner

彥廷

詩涵

育安

依儒

課程	主題	日期	負責團隊
R1	Agent 基礎介紹	May 7, 2025	主辦團隊
R2	記憶機制 Memory	May 21, 2025	
R3	外部工具 Tool	Jun 4, 2025	
R4	規劃功能 Planning	Jun 18, 2025	
R5	現成工具實戰	Jul 2, 2025	
R6	評估指標	Jul 16, 2025	
	專題提案	Jul 23, 2025	
	成果發表	TBD	

