1\_lab1.ipynb Notebook 檔案內容，以下是對應的\*\*功能規格說明（Functional Specification），摘要如下：

**📘 專案名稱：LangGraph 初探（基礎入門練習）**

**🔧 功能目的：**

本專案旨在透過 Python Notebook 的方式，引導使用者學會使用 [LangGraph](https://docs.langchain.com/langgraph/) 套件構建流程導向的應用邏輯，並熟悉以下技能：

1. 定義狀態物件（**State** object）以描述資料流程。
2. 使用 LangGraph 的 **Graph** Builder 來建立工作流程圖。
3. 建立節點函式（**Node**）作為流程中的處理單元。
4. 設定流程邊（**Edges**）以連接節點並定義邏輯分支。
5. **編譯**與執行整體流程圖，模擬資料狀態的轉移與處理。

**✨ 技術亮點：**

* **TypedDict** / **Pydantic BaseModel**：用於定義狀態資料結構。
* **Annotated**：用來註解狀態欄位，支援 LangGraph 的 reducer（狀態合併邏輯）。
* **Reducer 概念**：使用內建的 add\_messages reducer，自動將節點回傳結果合併至狀態中。
* **Node 函式**：每個節點是 Python 函式，負責資料處理與狀態更新。
* **Graph 結構**：透過連接邊與條件分支，模擬類似「狀態機」的工作流邏輯。
* **純 Python 函式導向**：即使不含 LLM，也可使用 LangGraph 進行資料流程建模。

**✅ 功能步驟：**

1. **定義 State 結構**  
   使用 **TypedDict** 或 **BaseModel** 定義初始狀態資料欄位。
2. **啟用 Graph Builder**  
   使用 StateGraph 建立狀態導向的流程圖元件。
3. **建立處理節點（Node）**  
   自定義一或多個 Python 處理函式，接收狀態並返回處理結果。
4. **建立流程邊（Edge）與分支邏輯**  
   使用 .add\_edge() 或 .add\_conditional\_edges() 實作循環、結束等邏輯分支。
5. **編譯與執行流程**  
   使用 .compile() 編譯流程圖，並以 .invoke() 觸發初始狀態執行流程。