DAY – 11

On Day 11, we were introduced to a powerful tool in the AI ecosystem — Notebook LM. Developed by Google, Notebook LM is designed to help users engage with their own documents using natural language prompts. This tool acts as a personalized AI research assistant, enabling context-aware interactions directly from uploaded files such as PDFs, Docs, and more.

1. WHAT IS NOTEBOOK LM?

Notebook LM (Language Model) is an AI-powered notebook interface where users can upload content (e.g., notes, PDFs, articles) and interact with it through intelligent queries. It functions like a research assistant that understands, summarizes, and responds based on the content you provide — not the general web.

2. HOW DOES NOTEBOOK LM WORK?

We learned the basic workflow and architecture of Notebook LM:

- Users upload content such as PDFs or Google Docs.
- The tool automatically scans and generates context, extracting key points and relevant summaries.
- Users can then ask questions or request summaries — and the responses are grounded specifically in the uploaded content.
- It uses contextual awareness to ensure the answers are accurate and relevant.

The model does not hallucinate or pull random facts — it strictly works within the uploaded content.

3. PURPOSE OF USING NOTEBOOK LM

We discussed various real-world applications and use cases:

- **For students:** To summarize long chapters, generate study notes, and quiz themselves.
- **For researchers:** To analyze papers, extract conclusions, or compare theories.
- **For professionals:** To automate documentation review or policy analysis.
- **For creators:** To transform written material into structured insights and outputs like audio or scripts.

Overall, it acts as a **knowledge assistant**, making content consumption faster, easier, and more interactive.

DAY – 13

On Day 13, we focused on the foundational yet highly practical concept of functions and function calling, especially in the context of integrating them with LLMs and external APIs. This session was aimed at understanding how to modularize code using Python functions and then extend this logic for real-time, dynamic applications using APIs.

1. WHAT IS A FUNCTION?

We began by revisiting the basics of Python, specifically the definition and purpose of functions. A function is a reusable block of code designed to perform a specific task. Functions help in:

- Keeping code modular and clean
- Avoiding repetition
- Separating logic for better readability and testing

2. WHAT IS FUNCTION CALLING?

We then discussed function calling, which refers to the process of invoking a function in your code. Function calling becomes especially important when:

- Executing specific tasks based on user input
- Managing multiple features within a program
- Integrating AI and API-based interactions in real-time applications

3. SIMPLE FUNCTION IMPLEMENTATION: name()

Our first coding activity involved creating and calling a simple function named name() that:

- Took no input
- Returned a hardcoded name (e.g., “Hello, I am Gursimran”)

This exercise helped us understand how basic functions are defined, invoked, and returned in Python.

4. MULTIPLE FUNCTION CALLING – INTERACTIVE SYSTEM

We expanded the project by designing multiple functions to simulate a basic personal assistant interface:

a. Name()

- Returns the name of the assistant/user.

b. Quotes()

- Returns a randomly selected motivational quote.

c. Health_tips()

- Provides daily health tips such as hydration reminders, breathing exercises, and nutrition suggestions.

d. Weather()

- Uses a Weather API key to fetch real-time weather updates based on location.
- The function made live API requests, parsed JSON responses, and displayed:
    - Temperature
    - Weather description
    - Humidity or wind speed

This demonstrated how functions can be linked to external APIs for real-time, useful outputs.

5. FUNCTION CALLING FOR CRICKET LIVE SCORES

As part of the practical task, we were assigned to implement a live cricket score tracker using:

- criAPI API key
- live_score() function that:
    - Fetched live match data
    - Displayed teams, score, overs, wickets, and status
- Optional use of additional functions like:
    - match_summary() for general overview
    - player_stats() for top performer information

We integrated these APIs using Python's requests module, structured the responses using json, and created a real-time sports update tool.

This assignment helped us understand:

- How to handle API keys securely
- How to parse and present live data
- How to build user-facing functions with live updates

CONCLUSION

Day 13 was a blend of Python fundamentals and real-time application development. By implementing and calling multiple functions — including those interacting with APIs — we learned how to make our applications dynamic, user-friendly, and modular.

This hands-on session laid the groundwork for building AI-enabled utilities and microservices, such as personal assistants, weather dashboards, and live sports trackers, powered by cleanly defined Python functions and real-time data.