MCP

A GUIDE ON

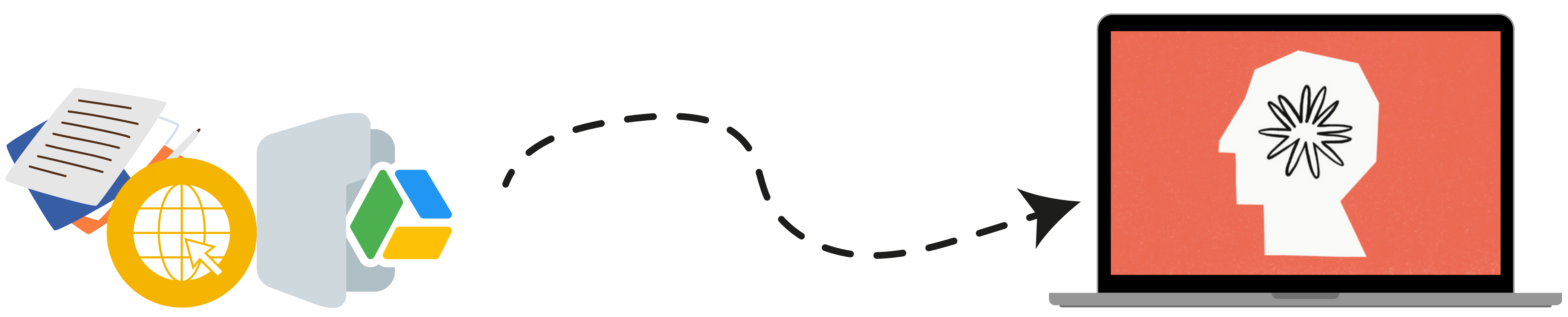
MODEL CONTEXT PROTOCOL (MCP)

WORKING

HANDS-ON

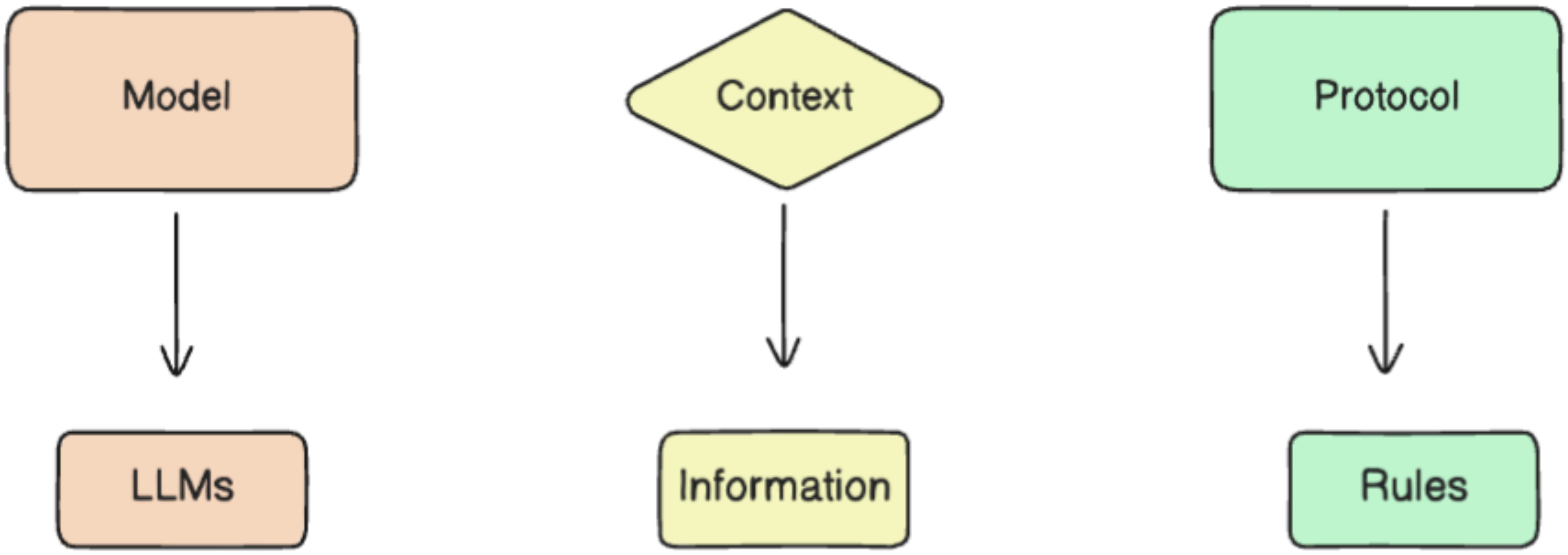
APPLICATIONS

& MORE



WHAT IS MCP?

It is a powerful open standard that was launched by Claude's parent company, Anthropic, in November 2024.



Model: The LLM (e.g., Claude, GPT-4) generates responses.

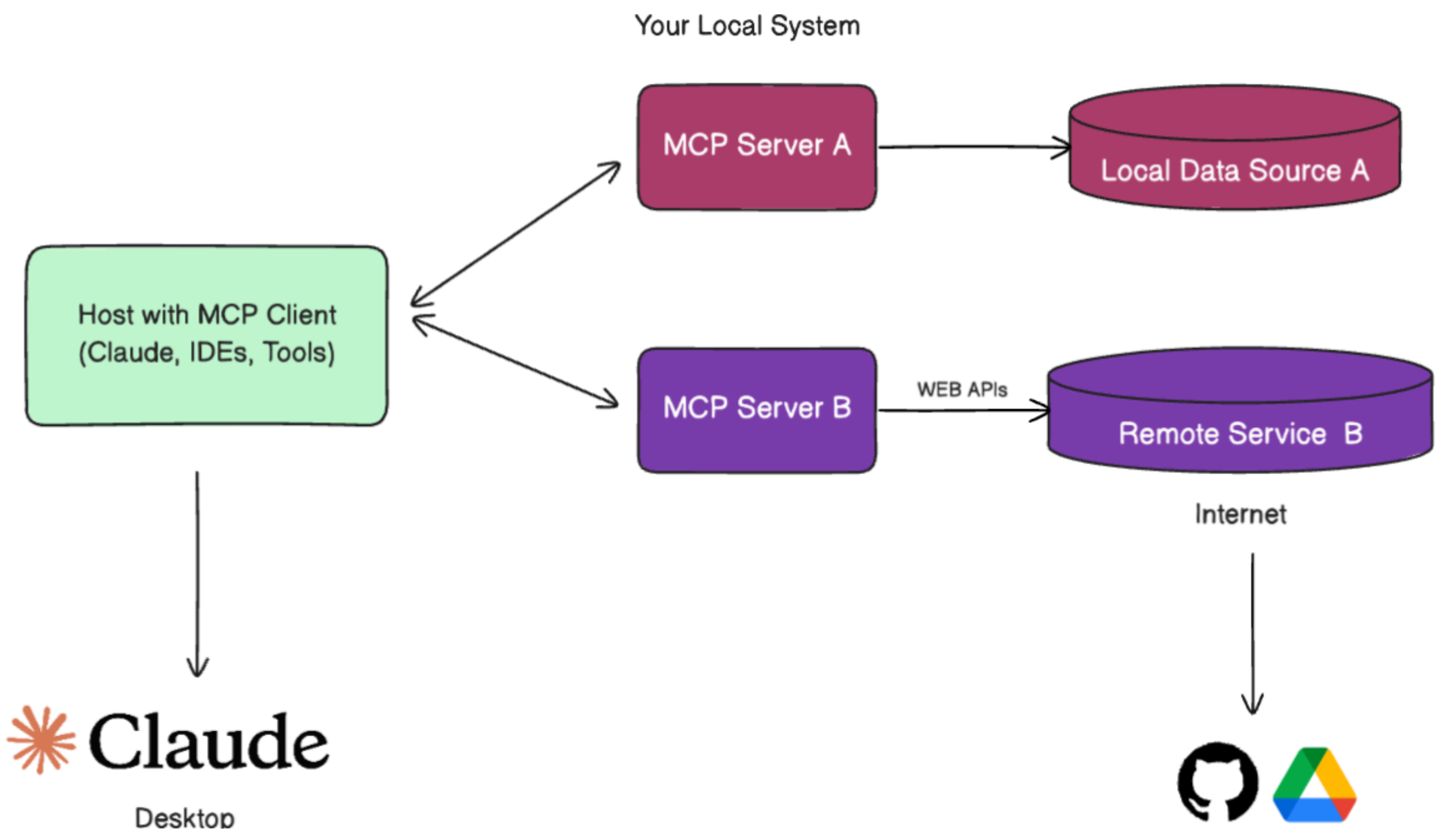
Context: Additional input (documents, PDFs, prompts, databases) for meaningful replies.

Protocol: Rules enabling the model to access and use structured context.

WHAT CAN WE DO WITH MCP?

MCP streamlines communication between the **AI clients** and **data servers**, enabling the system to access real-time data from various sources without custom code or manual uploads.

- **Client:** The interface for LLM responses (e.g., Claude's app, IDE-like cursor, custom chatbots, etc).
- **Server:** The data source storing context (e.g., Google Drive, GitHub, database, local files, etc).



GROWING POPULARITY OF MCP

Clients

Claude

Continue

CURSOR

WINDSURF AI

Spin.ai

Servers

GitHub

Google Maps

Google Drive

slack

Spotify®

MCP: HANDS ON

Tasks:

1. Create a Custom Greeting
2. Count the number of files on our desktop
3. Save a chat with Claude to our local system
4. Ask Claude questions using a local PDF document

To perform these 4 tasks, we are working with :

Client: Claude Desktop App

Server: Local files like documents and PDFs that are available on the local system

MCP: SET UP REQUIRMENTS

Step 1: Create a Python Environment

- With Conda type, *conda create ~n mcp python==3.11*

Step 2 : Install the MCP Library

- pip install mcp

Step 3 : Download Claude's Desktop App

- Download the app from https://claude.ai/download
- Works with free plan as well but may hit rate limits while experimenting

Step 4 : Set Up the Configuration File

- On the Desktop app, go to Claude Settings.
- Then go to the Developer Section.
- Click on "Edit Configuration"
- Look for or create "claude_desktop_config.json"

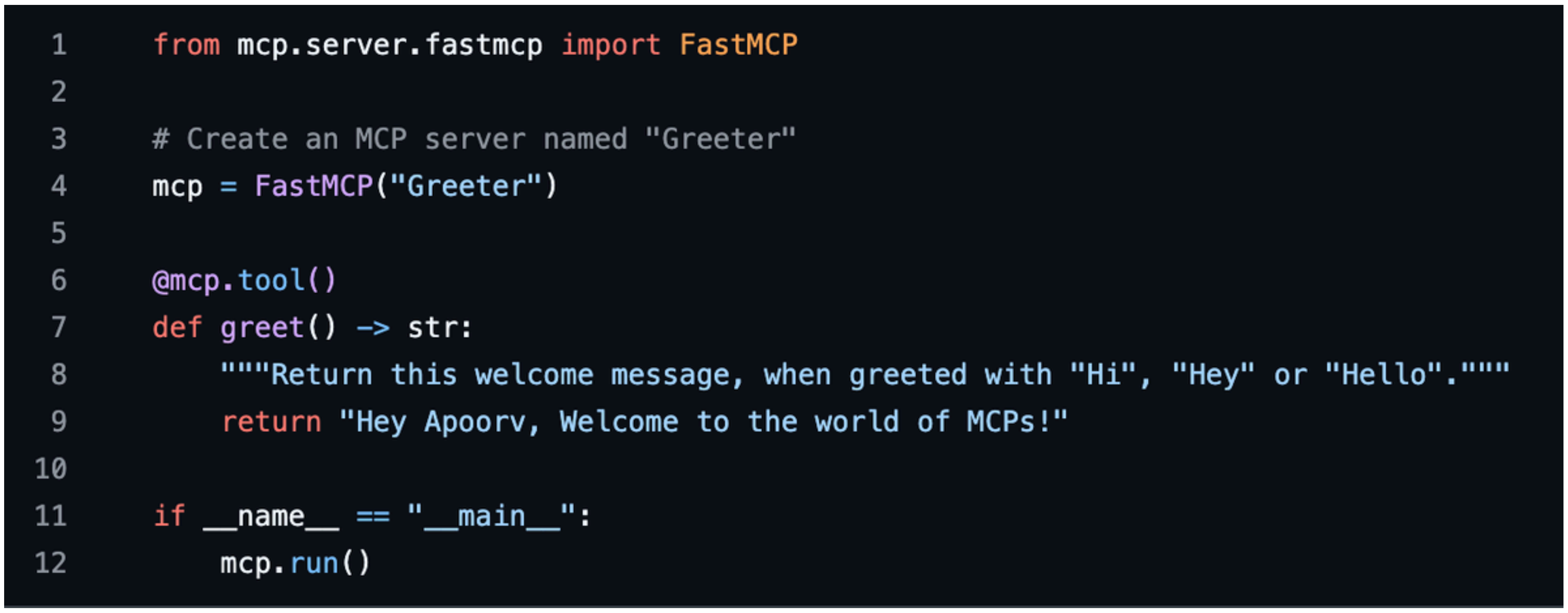
MCP

TASK 1 : CUSTOM GREETING

Step 1: Write the code in a new Python file

- Name the file: greeter.py

Code:



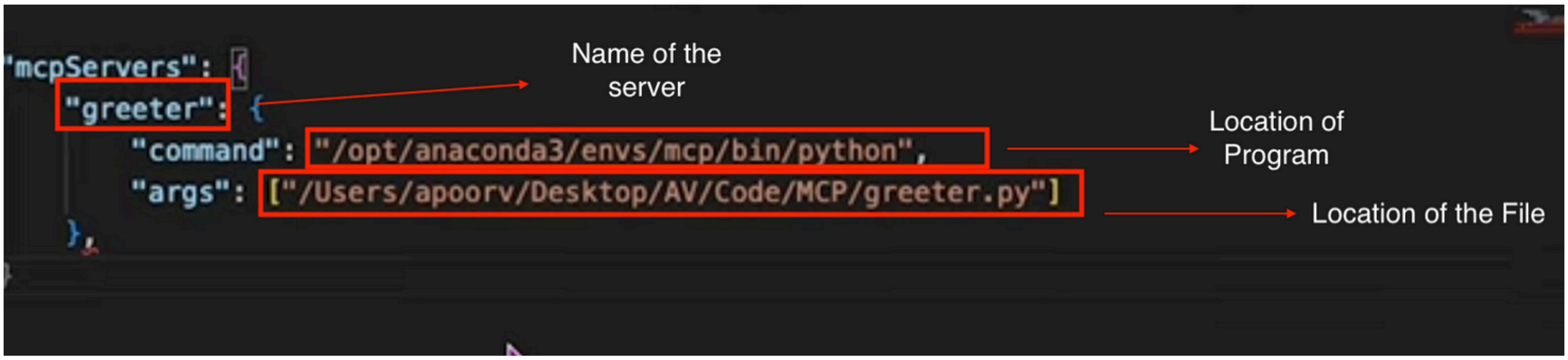
```python
from mcp.server.fastmcp import FastMCP

# Create an MCP server named "Greeter"
mcp = FastMCP("Greeter")

@mcp.tool()
def greet() -> str:
    """Return this welcome message, when greeted with "Hi", "Hey" or "Hello"."""
    return "Hey Apoorv, Welcome to the world of MCPs!"

if __name__ == "__main__":
    mcp.run()
```

- An MCP server listens for greetings ("Hi", "Hey", "Hello") and responds with a personalized message.
- The server is created simply using FastMCP and the tool is added using @mcp.tool() decorator.
- Outputs can differ since the LLM decides the final response.

TASK 1 : CUSTOM GREETING

Step 2: Update the configuration file

- Name of our server
- Location of the Python program
- Location of the Python file that has to be executed.



```
'mcpServers": {
    "greeter": {
        "command": "/opt/anaconda3/envs/mcp/bin/python",
        "args": ["/Users/apoorv/Desktop/AV/Code/MCP/greeter.py"]
    },
```

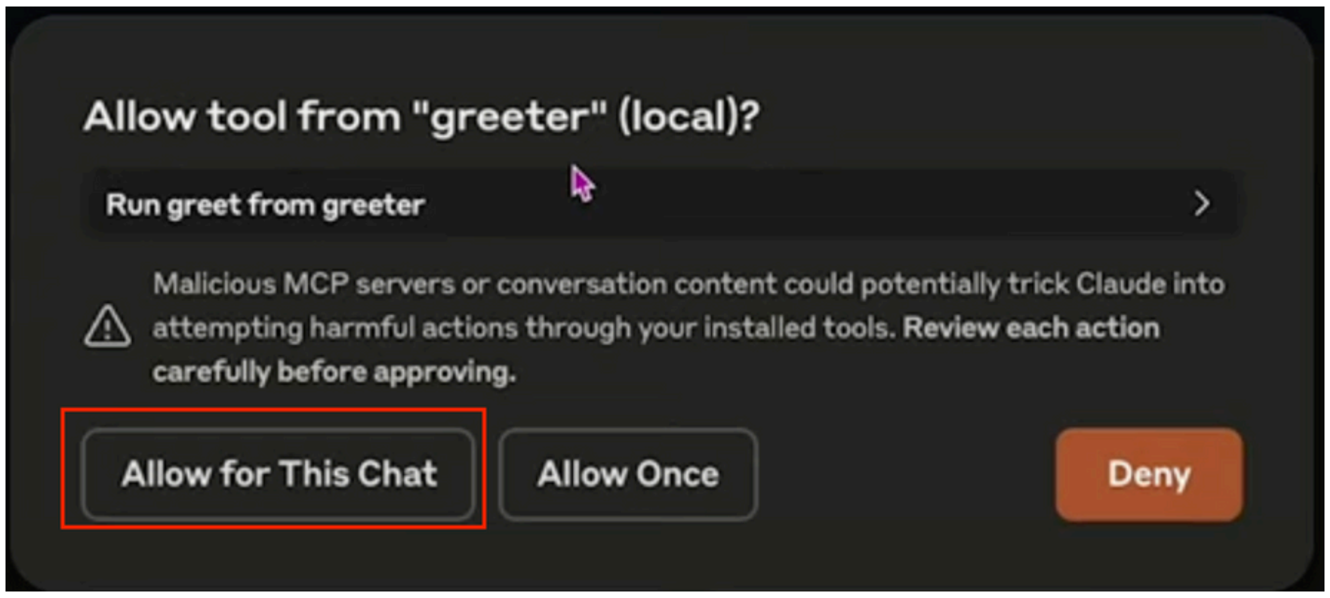
Step 3: Restart Your Claude Desktop App

- Any error in the code gets highlighted on this step.
- If no error is found, the tool gets registered within the desktop app
- Perform this step every time the code is changed or updated..

TASK 1 : CUSTOM GREETING

Step 4: Test the tool

- Just prompt a “Hi” to Claude and when asked, permit the LLM to use the tool.

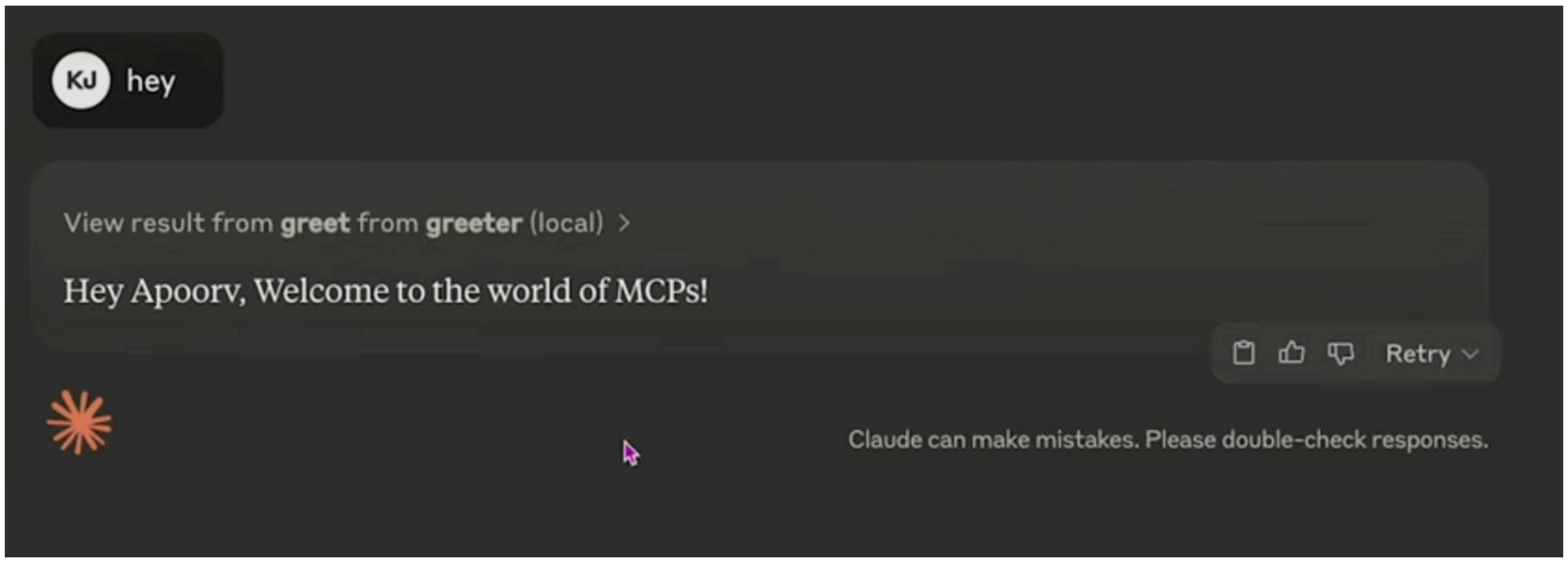


Allow tool from "greeter" (local)?

Run greet from greeter

Malicious MCP servers or conversation content could potentially trick Claude into attempting harmful actions through your installed tools. **Review each action carefully before approving.**

Output:



KJ hey

View result from **greet** from **greeter** (local)

Hey Apoorv, Welcome to the world of MCPs!

Claude can make mistakes. Please double-check responses.

MCP

MCP: APPLICATIONS

Multi-Step Projects: MCP enables AI to coordinate tasks across platforms, like event planning, without complex integrations.

Real-World Awareness: MCP helps AI interact with smart environments and IoT devices, making it more proactive.

Collaborating Agents: MCP allows multiple AI agents to collaborate and share information without direct integrations.

Personal AI Assistants: MCP enables personalized assistants to securely access personal data without exposing it to third parties.

Enhanced Customer Support: With MCP, AI provides context-aware customer service by accessing real-time data, improving efficiency.

MCP

HEAD TO THE : BLOG/VIDEO FOR TASKS 2,3 &4

CLICK HERE

Blog

Youtube

GitHub