Hugging Face MCP Course - Unit 1 Notes

Hugging Face MCP (Model Context Protocol) - Unit 1: Introduction to MCP

Overview:

Model Context Protocol (MCP) is a lightweight protocol that allows large language models (LLMs) to access tools and resources via standard interfaces. MCP aims to provide structured, interpretable, and consistent interaction between AI models and external environments.

Key Concepts:

1. What is MCP?

- MCP allows tools and functions to be accessed by models through a consistent interface.
- It creates a contract for interaction between tools and models using JSON-based context.

2. Core Components:

a. Tool:

- A Python function decorated with @mcp.tool()
- Exposed as callable via the MCP server.

b. Resource:

- A dynamic endpoint that serves data.
- Defined using @mcp.resource("resource_id").

c. Prompt:

- Structured prompt format using @mcp.prompt()
- Helps in defining reusable prompt templates for consistent interaction.

3. Example Server Setup:

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

mcp = FastMCP("Weather Service")

@mcp.tool()
def get_weather(location: str) -> str:
    return f"Weather in {location}: Sunny, 72°F"

@mcp.resource("weather://{location}")
def weather_resource(location: str) -> str:
    return f"Weather data for {location}: Sunny, 72°F"

@mcp.prompt()
def weather_report(location: str) -> str:
    return f"You are a weather reporter. Weather report for {location}?"

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

4. Benefits:

- Easily expose Python functions to models.

- Centralized context management.

- Decouples logic from interface.

- Integrates with local or remote models.

5. MCP Server:

- FastMCP is a lightweight local MCP server.

- You define tools/resources/prompts and run the server.

- It listens to model requests and returns results via JSON.

6. ToolCollection:

- Group tools and use them within the MCP client.

- Helpful in building a context of multiple tools to a model.

7. Transport Types:

- stdio: Communicates over standard input/output (used in local CLI-based tools).

- sse: Server-Sent Events for remote streaming interaction.

Conclusion:

Unit 1 introduces the core idea of exposing tools to language models using the MCP framework. You build simple MCP servers with tools, resources, and prompts and understand how structured context enhances model reasoning.