

lab_answer

April 3, 2026

1 Lab 13 — Building Your Own MCP Server (ANSWERS)

Build an MCP server with tools and resources using FastMCP.

```
[ ]: # Setup
from dotenv import load_dotenv
load_dotenv()

class NotebookMCP:
    def __init__(self, name):
        self.name = name
        self._tools = {}
        self._resources = {}

    def tool(self):
        def decorator(func):
            self._tools[func.__name__] = func
            return func
        return decorator

    def resource(self, uri_pattern):
        def decorator(func):
            self._resources[uri_pattern] = func
            return func
        return decorator

mcp = NotebookMCP("lab-server")
print(f"Notebook MCP server '{mcp.name}' ready.")
```

1.1 Exercise 1 — Minimal Server with Echo Tool

```
[ ]: @mcp.tool()
def echo(message: str) -> str:
    """Echo a message back to the caller.

    Args:
```

```

        message: The message to echo
    """
    return f"Echo: {message}"

print(echo(message="Hello MCP!"))
print(echo(message="Testing 1 2 3"))

```

1.2 Exercise 2 — Add Notes Tools

```

[ ]: notes: dict[str, str] = {}

@mcp.tool()
def add_note(title: str, content: str) -> str:
    """Add a new note to the collection.

    Args:
        title: Short title for the note
        content: Full text content of the note
    """
    notes[title] = content
    return f>Note '{title}' saved ({len(content)} chars)"

@mcp.tool()
def list_notes() -> str:
    """List all saved notes with their titles."""
    if not notes:
        return "No notes saved yet."
    return "\n".join(f"- {title}" for title in sorted(notes.keys()))

# Test
print(add_note(title="Python", content="A versatile programming language"))
print(add_note(title="MCP", content="Model Context Protocol for LLM tools"))
print(add_note(title="FastMCP", content="High-level framework for MCP servers"))
print()
print(list_notes())

```

1.3 Exercise 3 — Add Notes Resource

```

[ ]: @mcp.resource("notes://all")
def get_all_notes() -> str:
    """All notes formatted as text."""
    if not notes:
        return "No notes saved."

```

```

    parts = [f"## {title}\n{content}" for title, content in sorted(notes.
↪items())]
    return "\n\n".join(parts)

@mcp.resource("notes://{title}")
def get_note_by_title(title: str) -> str:
    """A specific note by its title."""
    if title not in notes:
        return f"Note '{title}' not found."
    return notes[title]

# Test
print("=== All Notes ===")
print(get_all_notes())
print()
print("=== Single Note ===")
print(get_note_by_title(title="Python"))
print()
print("=== Missing Note ===")
print(get_note_by_title(title="Nonexistent"))

```

```

[ ]: # Summary of registered tools and resources
print("Registered tools:", list(mcp._tools.keys()))
print("Registered resources:", list(mcp._resources.keys()))

```