# Lesson 7: Setting Up MCP Servers with Claude Code

## Learning Objectives

By the end of this lesson, you will be able to: - Understand what Claude Code is and how to install it - Create a simple MCP server in Python - Configure and expose MCP servers to Claude Code - Use MCP server functionality within Claude Code

## Introduction

In previous lessons, we've learned how to set up MCP servers as Python scripts that leverage the Model Context Protocol framework. We've also learned how to create Python clients that connect to these servers. In this lesson, we'll take it a step further by learning how to integrate MCP servers with Claude Code, a powerful AI coding assistant that runs in your terminal.

## What is Claude Code?

Claude Code is an AI-powered coding tool that lives in your terminal, created by Anthropic (the same company behind Claude Sonnet and Claude Opus). It's designed to help developers write code through natural language commands, making development more intuitive and efficient.

### Key Features:

- Terminal-based AI coding assistant
- Natural language code generation
- Integration with MCP servers
- Permission-based system for computer interactions

## Installing Claude Code

To get started with Claude Code, you'll need to install it using npm:

```
npm install -g @anthropic/claude-code
```

**Note:** Make sure you have npm installed on your machine before running this command.

Once installed, you can run Claude Code by simply typing:

```
claude
```

## Creating an MCP Server for Claude Code

Let's create a simple MCP server that Claude Code can use. We'll build a toy example that returns a list of favorite foods.

### Step 1: Set Up Your Project

First, create a new directory and set up a UV project:

```
mkdir mcp-server-claude
cd mcp-server-claude
uv init
uv venv
source .venv/bin/activate  # On Windows: .venv\Scripts\activate
```

### Step 2: Install Dependencies

Add the required MCP packages:

```
uv add mcp
```

### Step 3: Create the MCP Server

Create a file named `claude_mcp_server.py`:

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

# Create an instance of the MCP server
mcp = FastMCP(name="lucas-top-three-foods")

@mcp.tool
def print_lucas_top_three_foods():
    """
    Returns Lucas's top three favorite foods.
    This is a simple demonstration function.
    """
    return "Pancakes, Pizza, and Ribs"

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

### Step 4: Test Your Server

Before integrating with Claude Code, test that your server runs without errors:

```
uv run claude_mcp_server.py
```

If no errors appear, your server is ready!

## Integrating the MCP Server with Claude Code

Now comes the exciting part - making your MCP server available to Claude Code.

**Step 1: Find Your Python Executable Path**

You need the path to the Python executable in your virtual environment:

```
which python  # On Mac/Linux
# On Windows PowerShell: where python
```

**Step 2: Get the Full Path to Your Server Script**

```
realpath claude_mcp_server.py  # On Mac/Linux
# On Windows: Use the full path
```

**Step 3: Add the Server to Claude Code**

Use the `claude mcp add` command with the following format:

```
claude mcp add "lucas-top-three-foods" /path/to/venv/python /path/to/claude_mcp_server.py
```

Example:

```
claude mcp add "lucas-top-three-foods" /Users/yourname/project/.venv/bin/python /Users/yourn
```

You should see a confirmation message that the STDIO MCP server was added successfully.

## Using Your MCP Server in Claude Code

Once your server is added, you can use it within Claude Code:

1. Open Claude Code by typing `claude`
2. Type `/mcp` to see available MCP servers
3. You should see your "lucas-top-three-foods" server listed as connected
4. Ask Claude to use your tool:

   ```
   What are Lucas's top three foods?
   ```

Claude will request permission to use the MCP tool (the first time only), and then execute your function, returning the result.

## Project-Scoped MCP Servers

By default, MCP servers are available globally in Claude Code. However, you can restrict an MCP server to a specific project by using the `-s project` flag:

```
claude mcp add -s project "lucas-top-three-foods" /path/to/venv/python /path/to/claude_mcp_s
```

This creates a `.mcp.json` file in your current directory that contains the server configuration:

```json
{
  "mcpServers": {
    "lucas-top-three-foods": {
      "type": "stdio",
      "command": "/path/to/venv/python",
      "args": ["/path/to/claude_mcp_server.py"],
      "env": {}
    }
  }
}
```

## Best Practices

1. **Virtual Environments**: Always use virtual environments to isolate your MCP server dependencies
2. **Secure Paths**: Use absolute paths to your Python executable and server script
3. **Testing**: Test your server independently before integrating with Claude Code
4. **Permissions**: Be mindful of the permissions you grant to Claude Code when it requests to use your tools
5. **Scoping**: Use project-scoped servers when the functionality is specific to a particular project

## Summary

In this lesson, we've learned how to: - Install and use Claude Code as an AI coding assistant - Create a simple MCP server using FastMCP - Configure and add MCP servers to Claude Code - Use MCP server functionality within Claude Code - Implement project-scoped MCP servers

This integration opens up powerful possibilities for extending Claude Code's capabilities with custom tools and functionalities. In the next lesson, we'll explore how to set up MCP servers with other development tools like Cursor.

## Practice Exercise

Try creating your own MCP server with a different function, such as: - A tool that returns the current date and time - A function that performs a simple calculation - A utility that formats text in different ways

Then, integrate it with Claude Code and test it out!

## Additional Resources

- Claude Code Documentation
- Model Context Protocol Specification

- FastMCP Framework Documentation