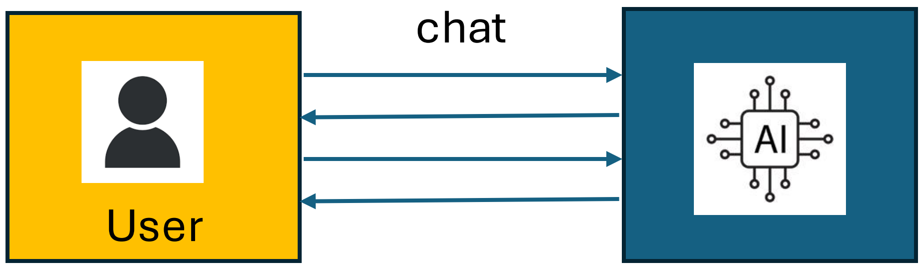
Tutorial on Building AI Scientist Agents with Model Context Protocol

Xiao-Liang Qi and Chen Nie @Path Integral

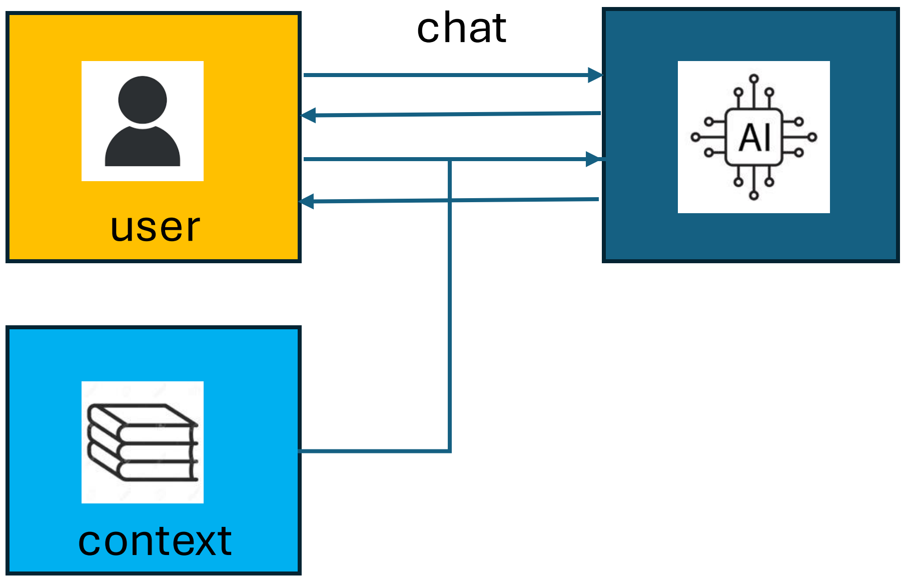
Part I: Background

From LLM to agents

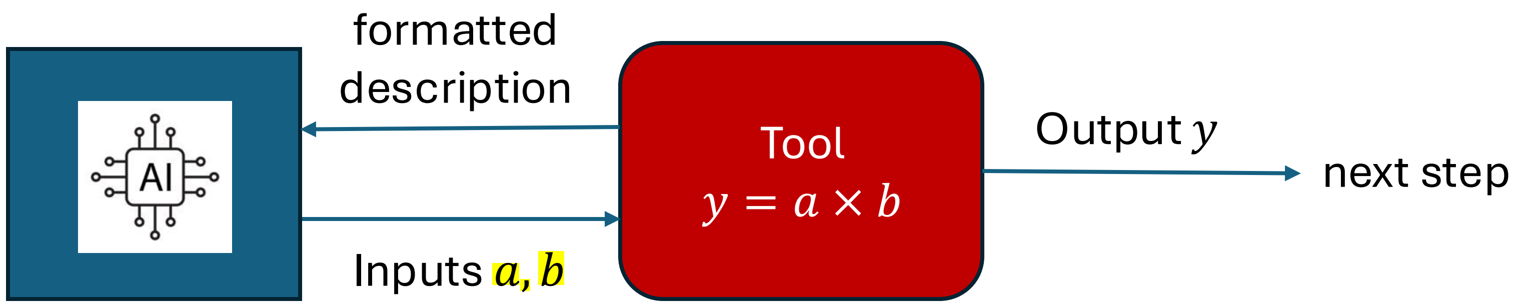
Model

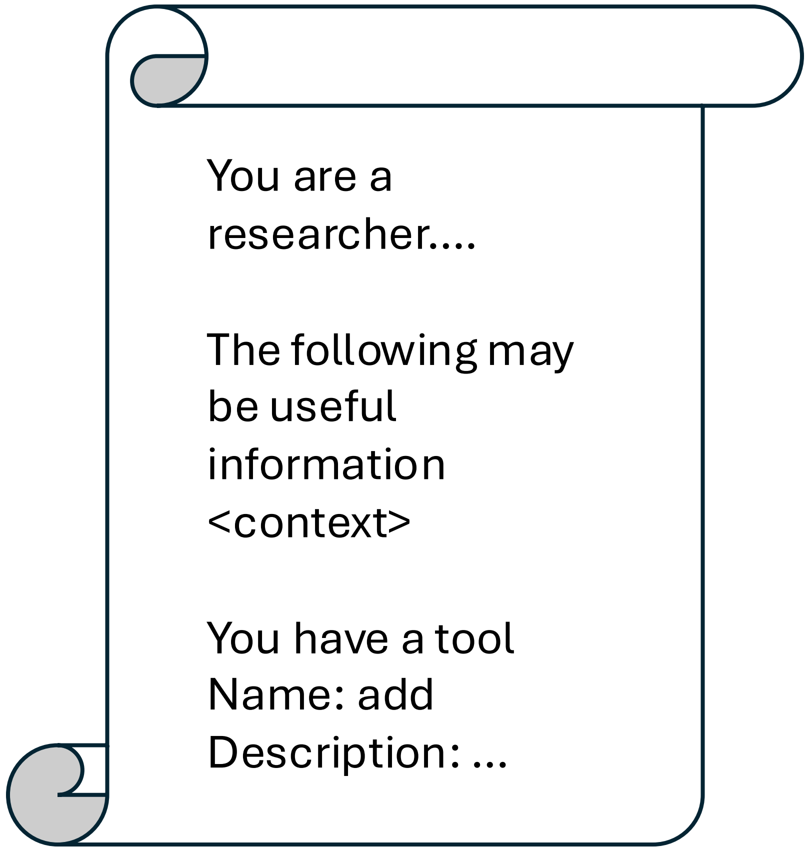


Model with Additional Context



Model with Tool Use




You are a researcher....

The following may be useful information <context>

You have a tool
Name: add
Description: ...

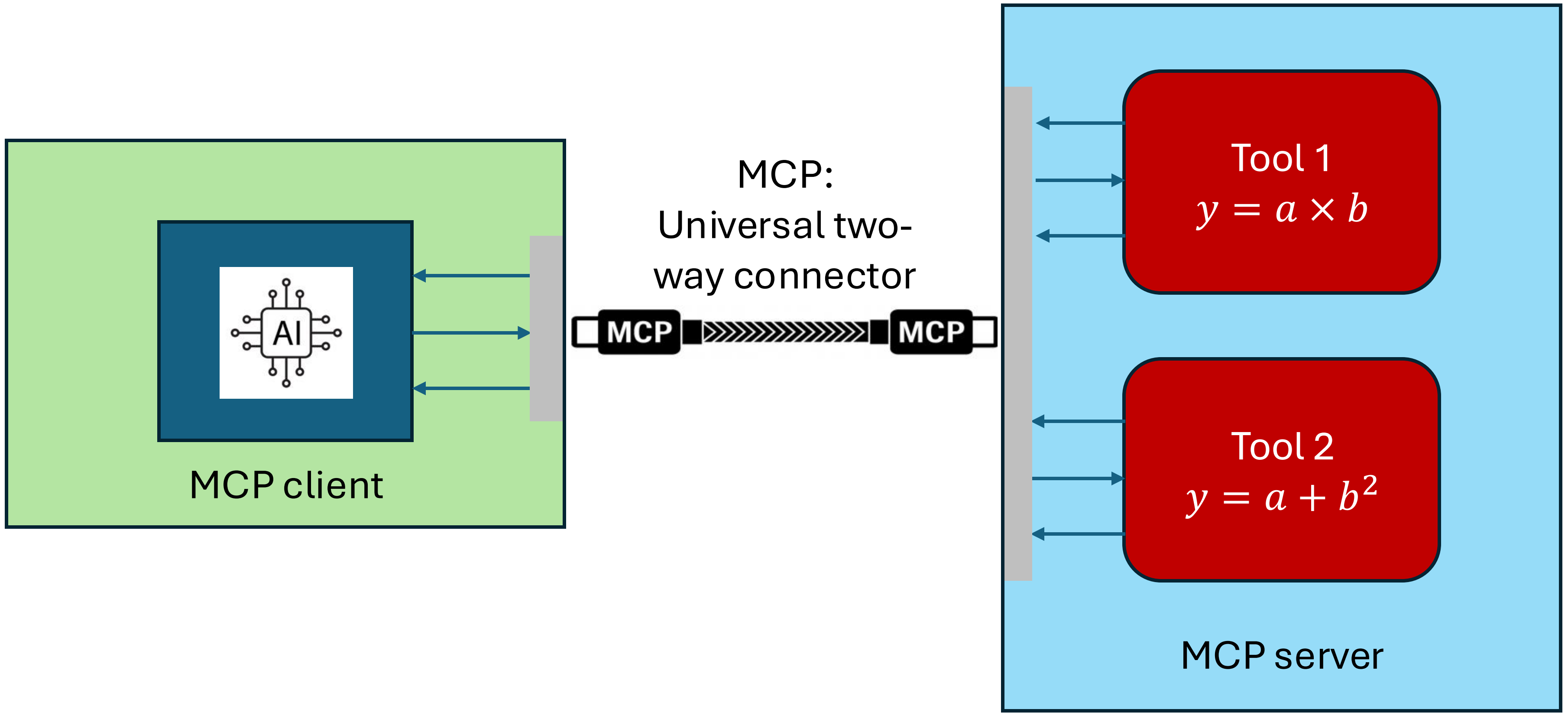
- Everything is a prompt for the model
- Tool use is achieved by context with specialized format
- Tools can be anything you need: functions carrying a computation, searching online, accessing a software

Tools in Scientific Research

Computational Methods	Experimental Equipments	Information Resource	Derivation and Documents
Monte Carlo	Monte Carlo	search	mathematica
Diagonalization	Diagonalization	database	overleaf
DFT	DFT	arxiv	ppt
…	…	…	…

Model-Context-Protocol (MCP)

A “USB connector” between agents and tools



What can AI do in scientific research

Automation of routine works

Reduction of communication barrier

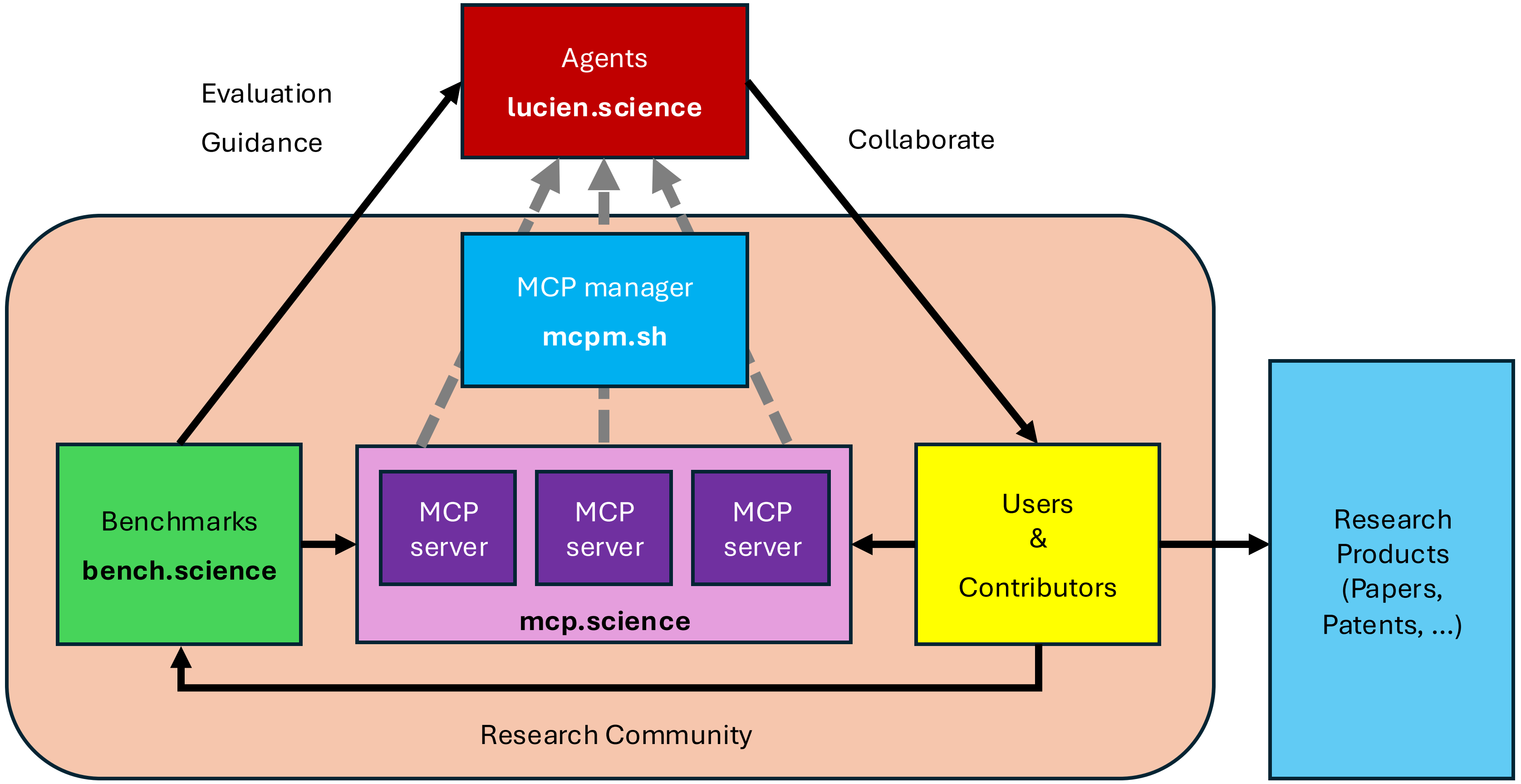
Enable more efficient knowhow sharing

Rennovating scientific publication

This procedure requires all researchers to teach the AI with their expertise

Build the AI Scientist with Research Community

Path Integral's slogan: Integrate all Paths to Knowledge



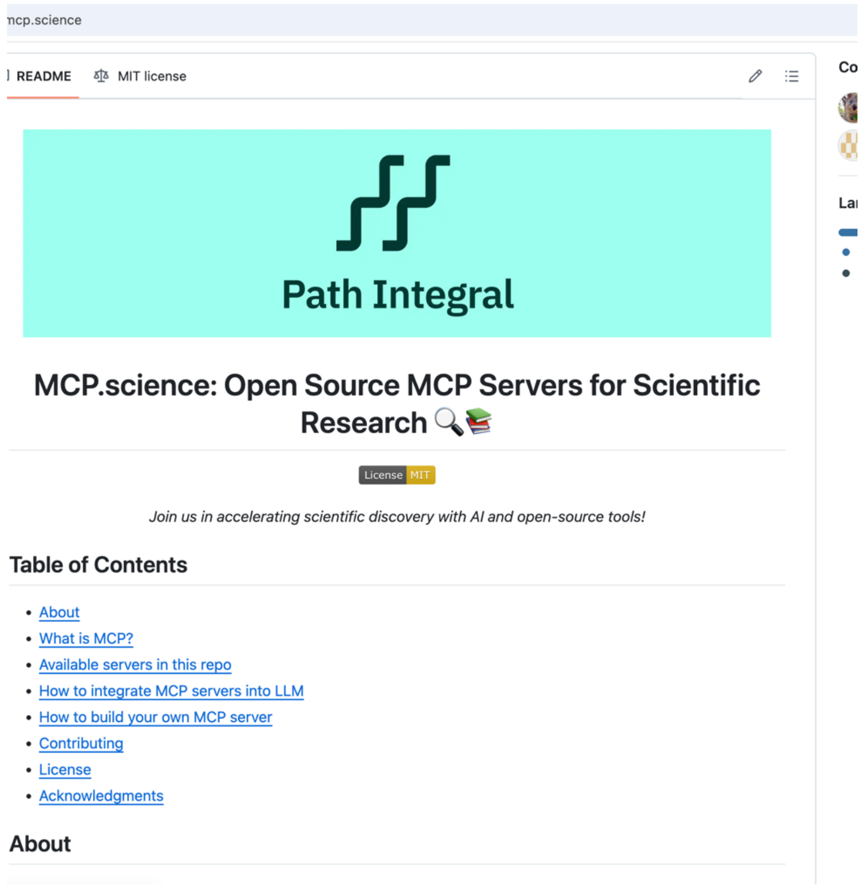
Path Integral Institute

mcp.science

- An open source repo for science mcp servers
- A more efficient way to share knowledge and knowhow
- A hub for discovering, installing and maintaining a wide range of tools for scientific research
- mcp.science servers can be connected easily by a command such as

```
uvx mcp-science python-code-execution
```

```
uvx mcp-science mathematica-check
```



MCP.science: Open Source MCP Servers for Scientific Research 🔍📚

Join us in accelerating scientific discovery with AI and open-source tools!

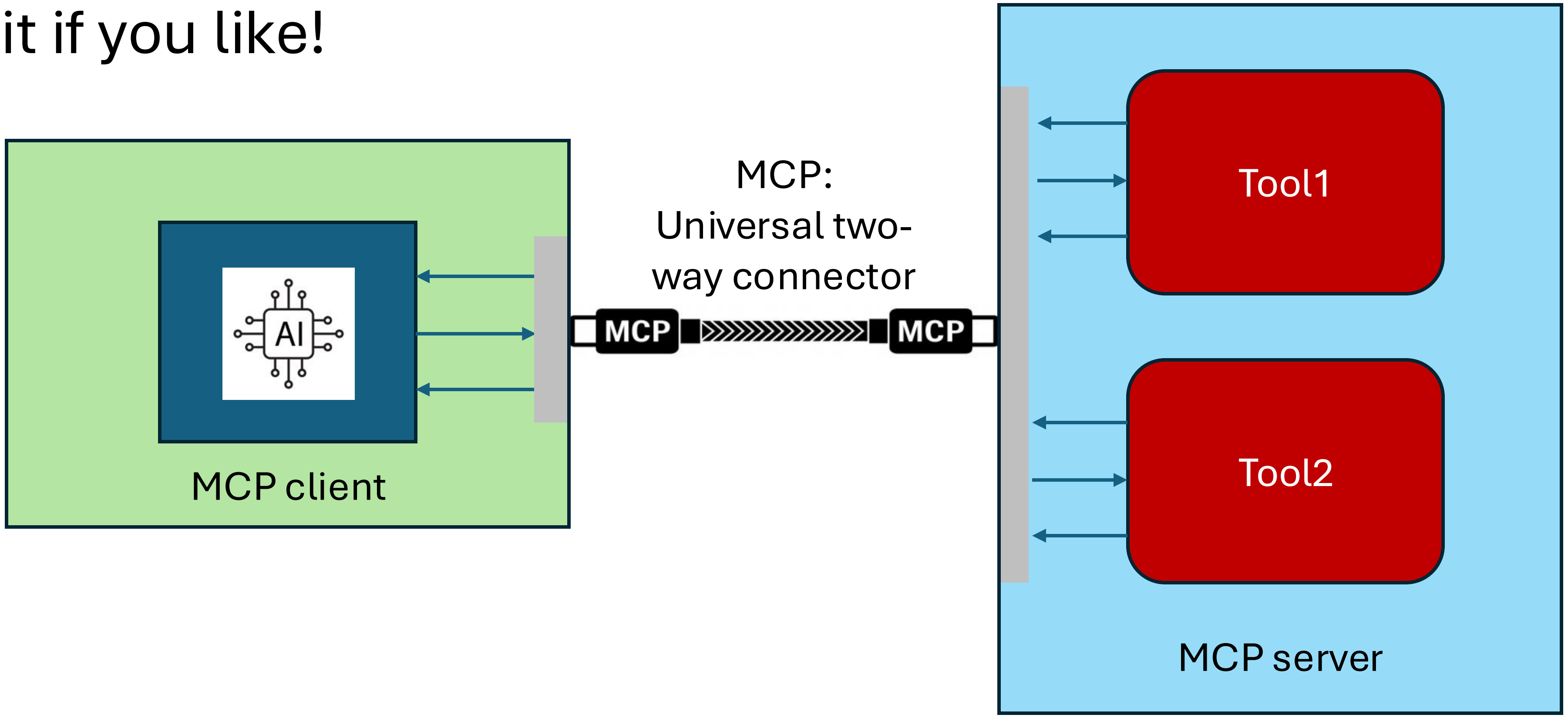
Table of Contents

About

Part II: Build MCP Servers

Concrete Steps to connect your research tool with AI

- 1. Environment Setup
- 2. Develop MCP servers
- 3. Connecting MCP servers to MCP clients. Ask the agent to start working
- 4. Share it if you like!



Environment Setup

- 1. Install uv
  Mac: `curl -LsSf https://astral.sh/uv/install.sh | sh`
- Windows: `powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"`
- 2. Set up your server folder

  `cd` to the folder you want to add your mcp server, such as `cd /Users/your_user_name/Documents/example_server`
- and run the command

```
uv init
uv venv
source .venv/bin/activate
uv add mcp 'mcp[cli]'
```

mac

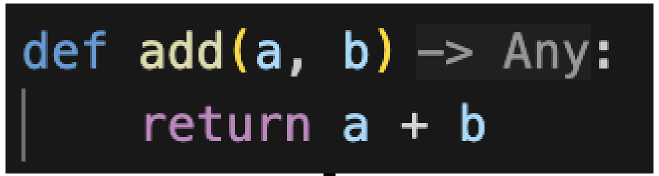
```
uv init
uv venv
.venv\Scripts\activate uv add
mcp 'mcp[cli]'
```

windows

More Detailed Guide

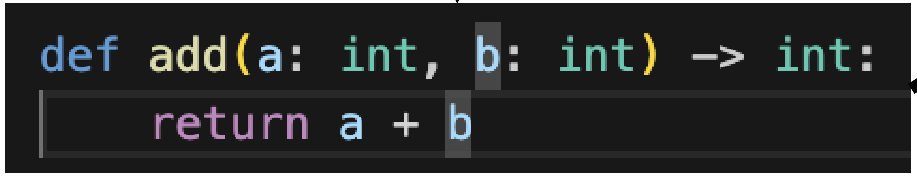
Building an MCP server: Simplest example

- Simplest example tool

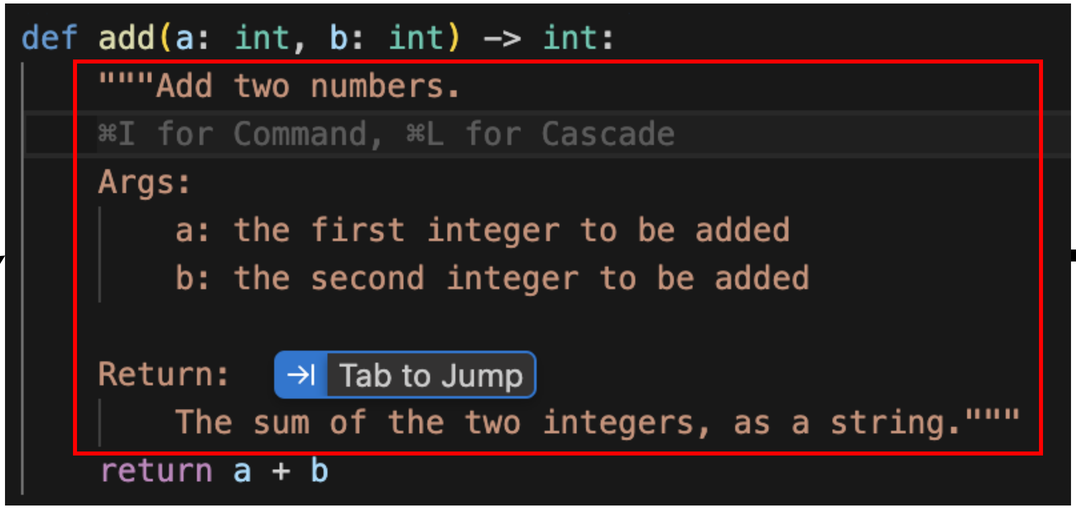


```
def add(a, b) -> Any:
    return a + b
```

add type hint



```
def add(a: int, b: int) -> int:
    return a + b
```



```
def add(a: int, b: int) -> int:
    """Add two numbers.
    ⌘I for Command, ⌘L for Cascade
    Args:
        a: the first integer to be added
        b: the second integer to be added

    Return:     Tab to Jump
        The sum of the two integers, as a string."""
    return a + b
```

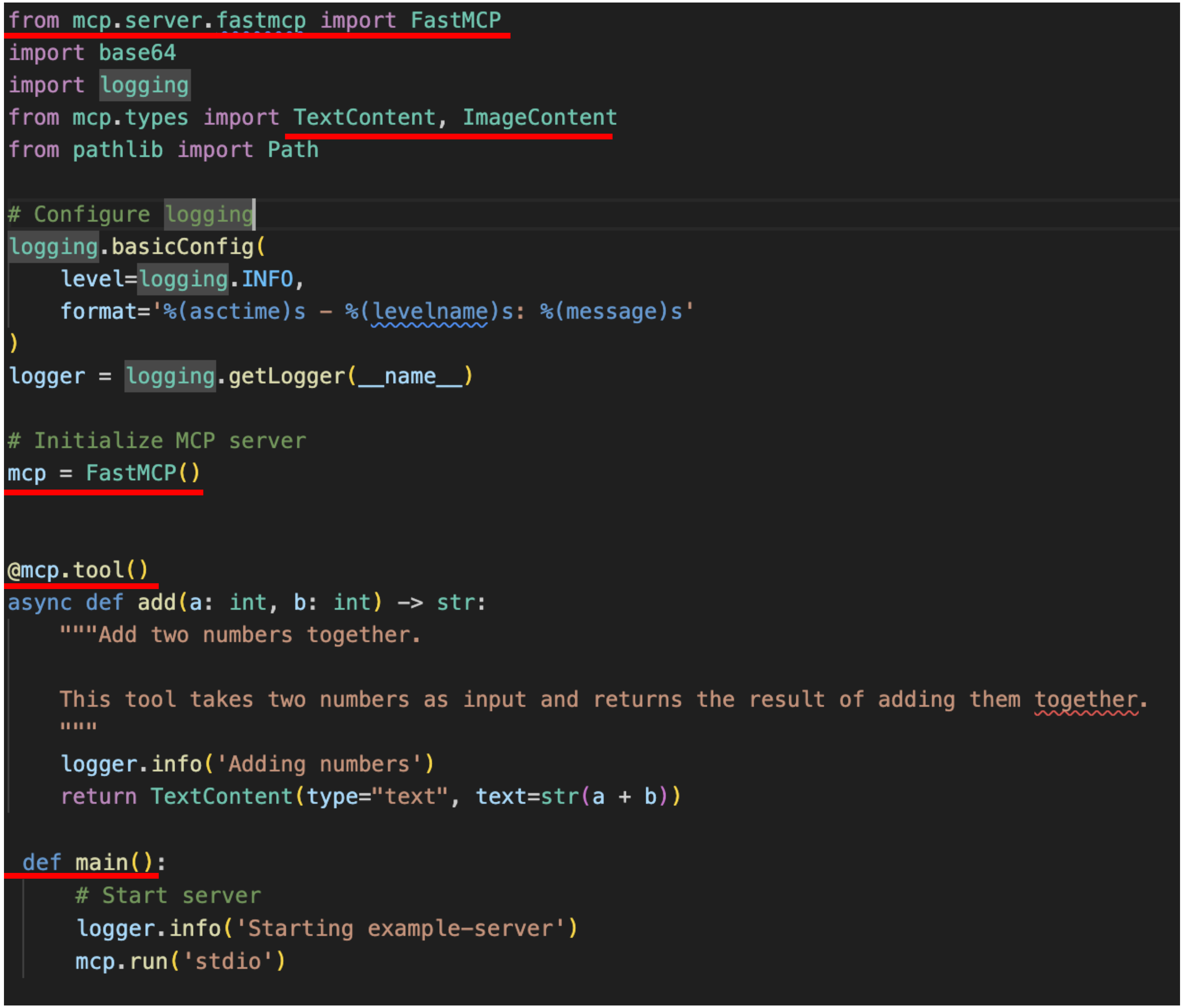
add doc string

Step-by-step Guide
Example Server

Specify output format

Import mcp sdk and add decorators

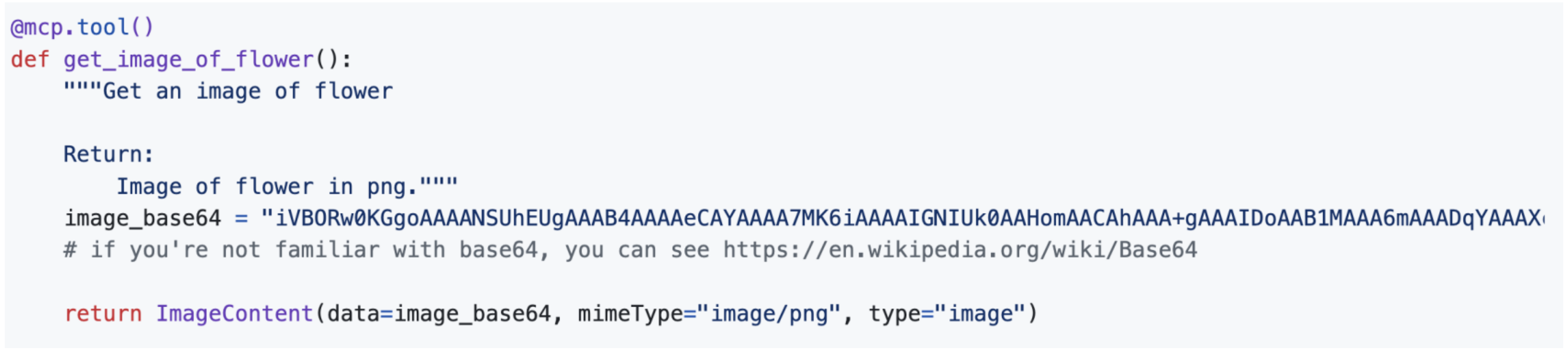
Define the main entry point



```
from mcp.server.fastmcp import FastMCP
import base64
import logging
from mcp.types import TextContent, ImageContent
from pathlib import Path

# Configure logging
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s: %(message)s'
)
logger = logging.getLogger(__name__)

# Initialize MCP server
mcp = FastMCP()


@mcp.tool()
async def add(a: int, b: int) -> str:
    """Add two numbers together.

    This tool takes two numbers as input and returns the result of adding them together.
    """
    logger.info('Adding numbers')
    return TextContent(type="text", text=str(a + b))

 def main():
    # Start server
    logger.info('Starting example-server')
    mcp.run('stdio')
```

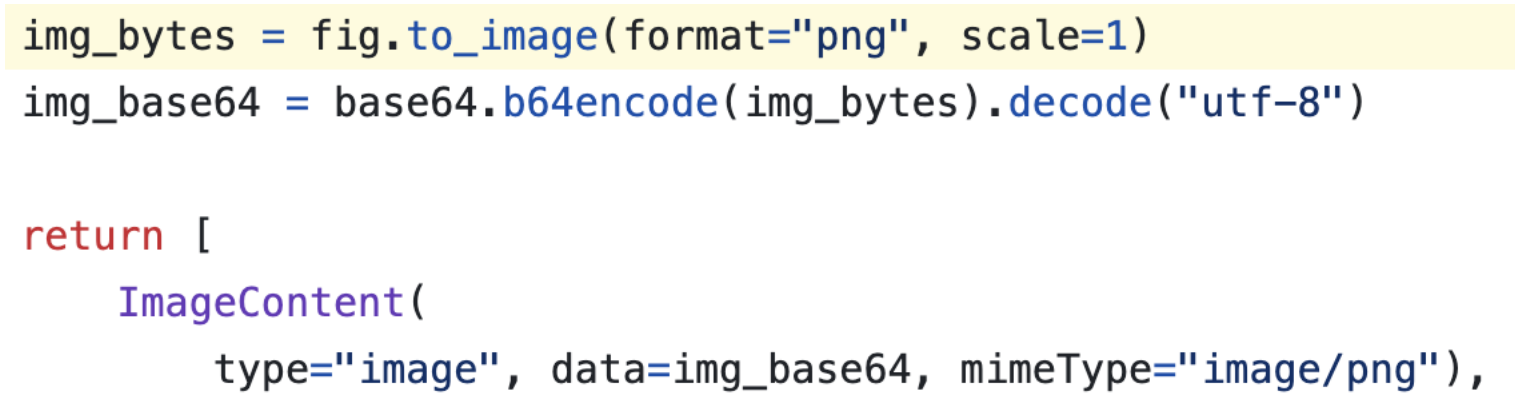
Useful tips: Image handling

- Images needs to be returned as `ImageContent` in the base64 format



```python
@mcp.tool()
def get_image_of_flower():
    """Get an image of flower

    Return:
        Image of flower in png."""
    image_base64 = "iVBORw0KGgoAAAANSUhEUgAAAB4AAAAeCAYAAAA7MK6iAAAAIGNIUk0AAHomAACAhAAA+gAAAIDoAAB1MAAA6mAAADqYAAAX
    # if you're not familiar with base64, you can see https://en.wikipedia.org/wiki/Base64

    return ImageContent(data=image_base64, mimeType="image/png", type="image")
```

- Check here for an example of converting an image to base64



```python
img_bytes = fig.to_image(format="png", scale=1)
img_base64 = base64.b64encode(img_bytes).decode("utf-8")

return [
    ImageContent(
        type="image", data=img_base64, mimeType="image/png"),
```

Useful tips: Running a subprocess

- Running a command to control another app, or a python code that requires a different environment
- If you need to control another app, command line is usually the simplest way
- Python has subprocess library which can be used to run any command
- Example:

```
import subprocess
subprocess.run([’ls’])
```

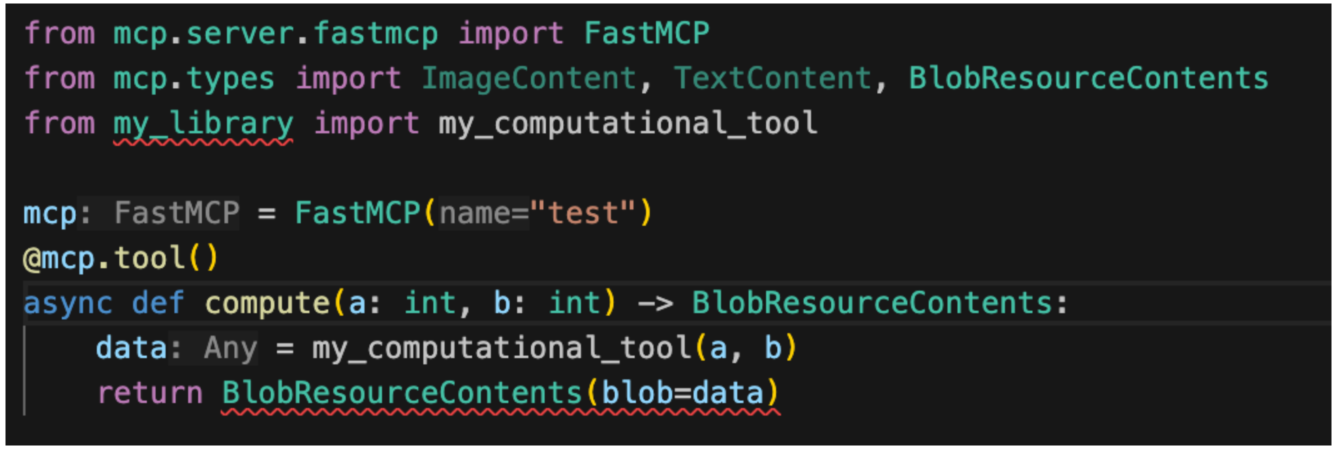
- Click here to see an example of mcp.science server running mathematica through subprocess
- Using subprocess could be complicated. Ask AI or our tech support team if you need this and meet a problem!

Useful tips: Saving files

- Your tool may generate some file or data that is useful later, but LLM does not need to see
- For example, large amount of computational data.
- Save it and tell the LLM the uri (similar to url) such as

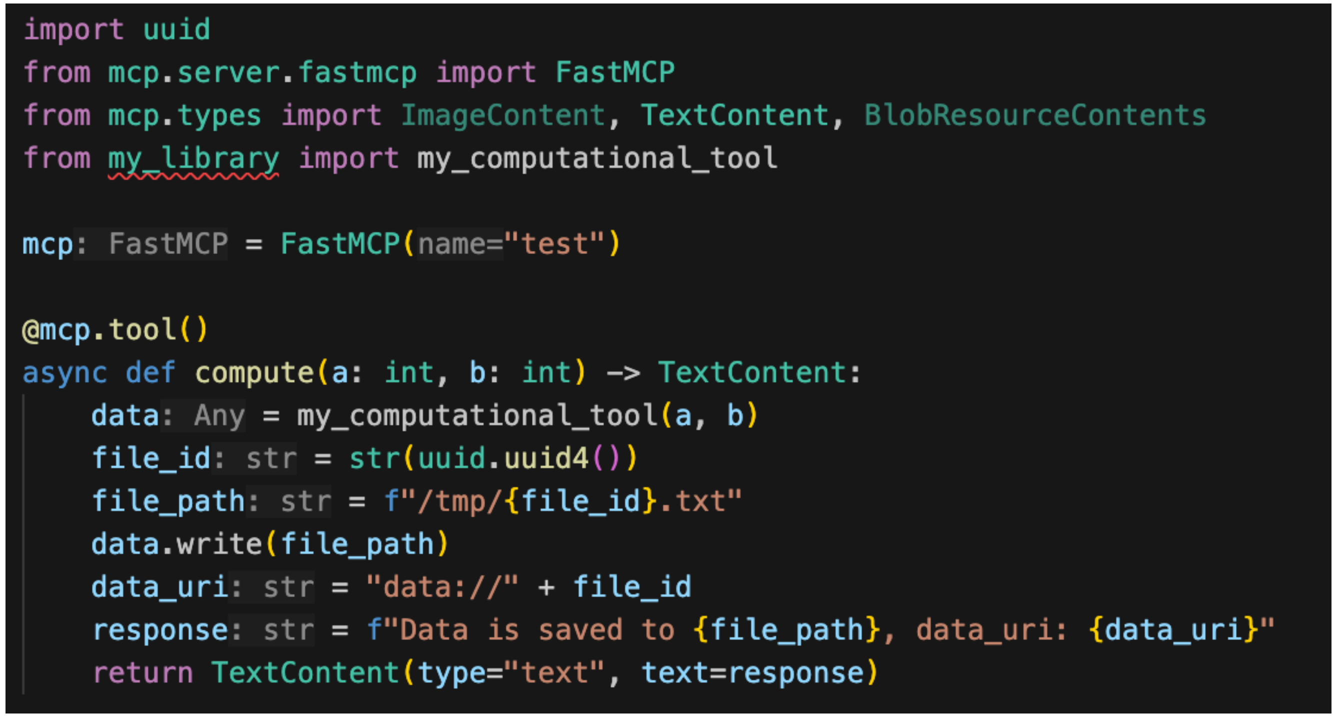
  data://0d4f9b17-88c5-4700-9e35-258a1eb4cbf7

  structure://4700-88c5-9e35-258a1eb4cbf7-0d4f9b17
- Use the uri to retrieve the file later



```python
from mcp.server.fastmcp import FastMCP
from mcp.types import ImageContent, TextContent, BlobResourceContents
from my_library import my_computational_tool

mcp: FastMCP = FastMCP(name="test")
@mcp.tool()
async def compute(a: int, b: int) -> BlobResourceContents:
    data: Any = my_computational_tool(a, b)
    return BlobResourceContents(blob=data)
```

better version



```python
import uuid
from mcp.server.fastmcp import FastMCP
from mcp.types import ImageContent, TextContent, BlobResourceContents
from my_library import my_computational_tool

mcp: FastMCP = FastMCP(name="test")

@mcp.tool()
async def compute(a: int, b: int) -> TextContent:
    data: Any = my_computational_tool(a, b)
    file_id: str = str(uuid.uuid4())
    file_path: str = f"/tmp/{file_id}.txt"
    data.write(file_path)
    data_uri: str = "data://" + file_id
    response: str = f"Data is saved to {file_path}, data_uri: {data_uri}"
    return TextContent(type="text", text=response)
```

MCP servers: local test run

- Run your server locally

```
python main.py
```

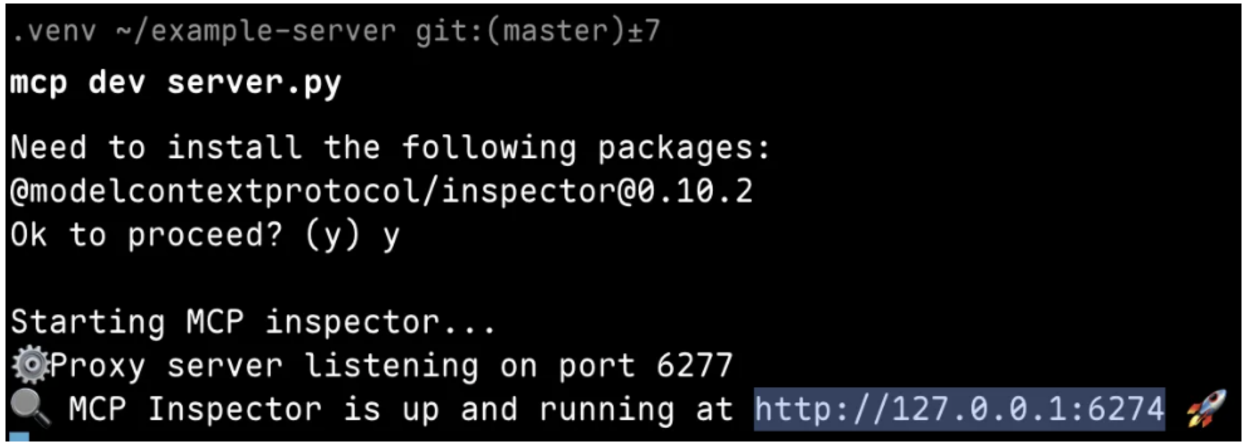
- You should see something like

```
2025-04-01 09:58:42,666 - INFO - your_new_server - Starting your-new-server
```

- Use the dev tool

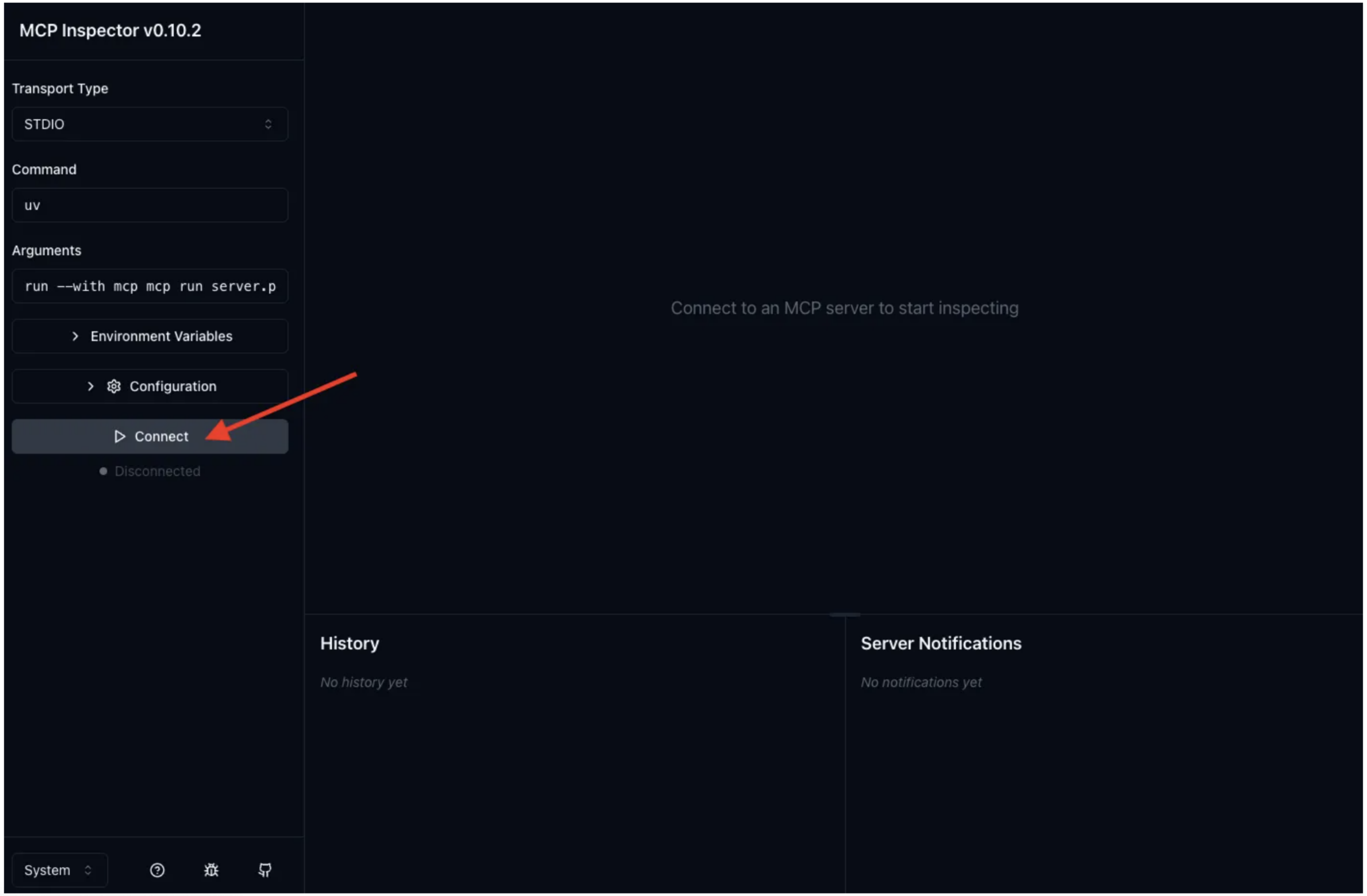
```
mcp dev main.py
```

- This will give you a local url. Click it to open a MCP Inspector



```
.venv ~/example-server git:(master)±7
mcp dev server.py
Need to install the following packages:
@modelcontextprotocol/inspector@0.10.2
Ok to proceed? (y) y

Starting MCP inspector...
⚙️Proxy server listening on port 6277
🔍 MCP Inspector is up and running at http://127.0.0.1:6274 🚀
```

More Detailed Guide



MCP Inspector v0.10.2

Transport Type

STDIO

Command

```
uv
```

Arguments

```
run --with mcp mcp run server.p
```

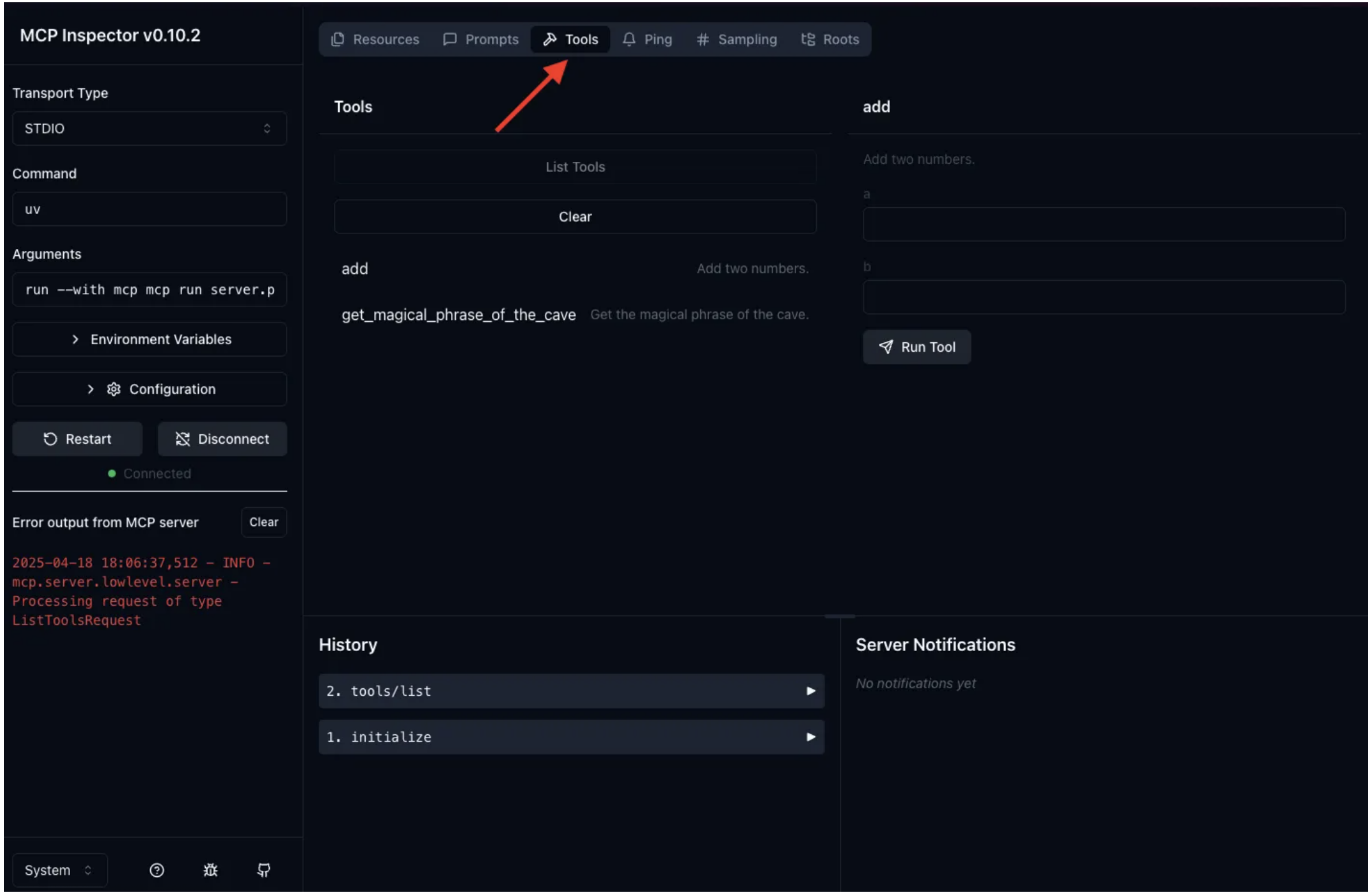
Connect to an MCP server to start inspecting

History

No history yet

Server Notifications

No notifications yet



Transport Type: STDIO

Command: uv

Arguments: run --with mcp mcp run server.p

Environment Variables

Configuration

Connected

Error output from MCP server

```
2025-04-18 18:06:37,512 - INFO -
mcp.server.lowlevel.server -
Processing request of type
ListToolsRequest
```

Tools

add — Add two numbers.

get_magical_phrase_of_the_cave — Get the magical phrase of the cave.

add

Add two numbers.

a

b

History

```
2. tools/list
1. initialize
```

Server Notifications

No notifications yet

Part III: Connecting MCP Servers to AI Agents

Two types of MCP server transport

- streamable HTTP / SSE
  - usually hosted as a web service
  - best suited for accessing cloud resources & tools
  - more complexity to build and maintain
- stdio (standard I/O)
  - MCP servers running as a process on local computer
  - usually started by MCP clients through a command (uvx, npx, docker)
  - best for accessing local resources & tools
    - local lab equipments
    - compute clusters accessible only within a private network
    - local code, file & data
  - less complexity to build and minimal work to maintain

Connect MCP servers with MCP client

- Lucien by Path Integral
  - Download Link and Detailed instructions
  - Configure MCP
  - Canvas + Jupyter + Knowledge Base
- 5ire & Lambda
  - claim your $400 Lambda credits
  - Install 5ire https://5ire.app/ and configure Lambda with llama-4
- Claude desktop
  - Download from Anthropic
  - Configure MCP through JSON editor
  - free tier with token rate limit

More Detailed Setup Instruction of These Agents

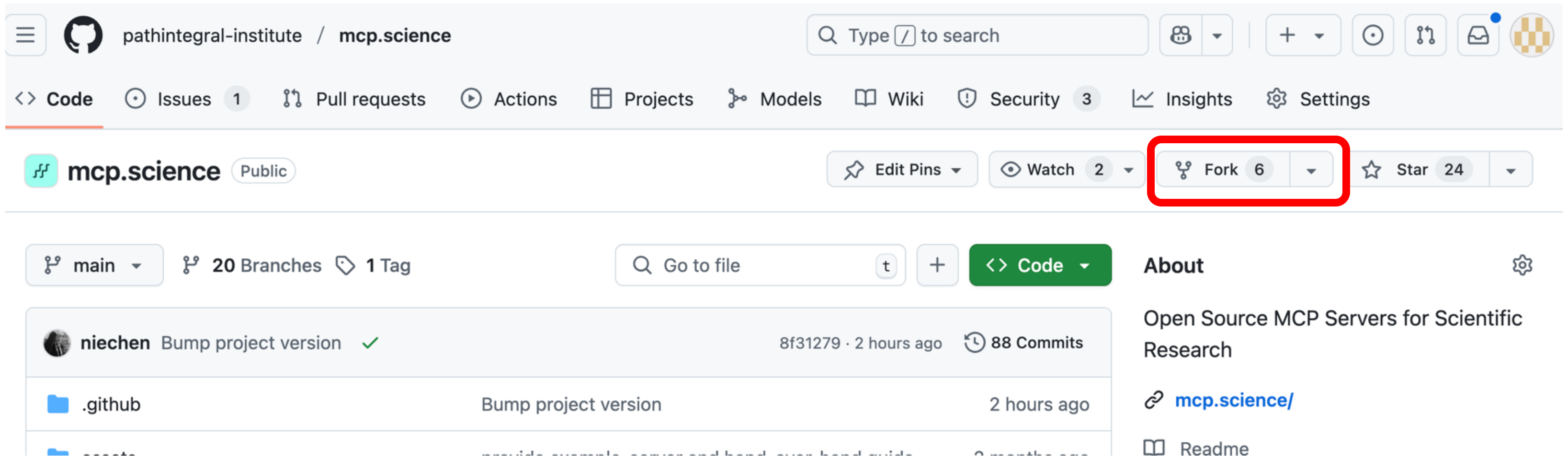
MCP local development pitfall

- Testing local MCP, remember to include the absolute path of the server entry file and define the UV_PROJECT environment variable. For example:

- Server type: Stdio
- Server name: example
- Command: uv
- Arguments: run /local/path/to/mcp.science/servers/example-server/src/example_server/main.py
- Environment Variables: UV_PROJECT=/local/path/to/mcp.science/servers/example-server/

Part IV: Contribute to mcp.science

fork mcp.science



niechen Bump project version ✓ 8f31279 · 2 hours ago 88 Commits

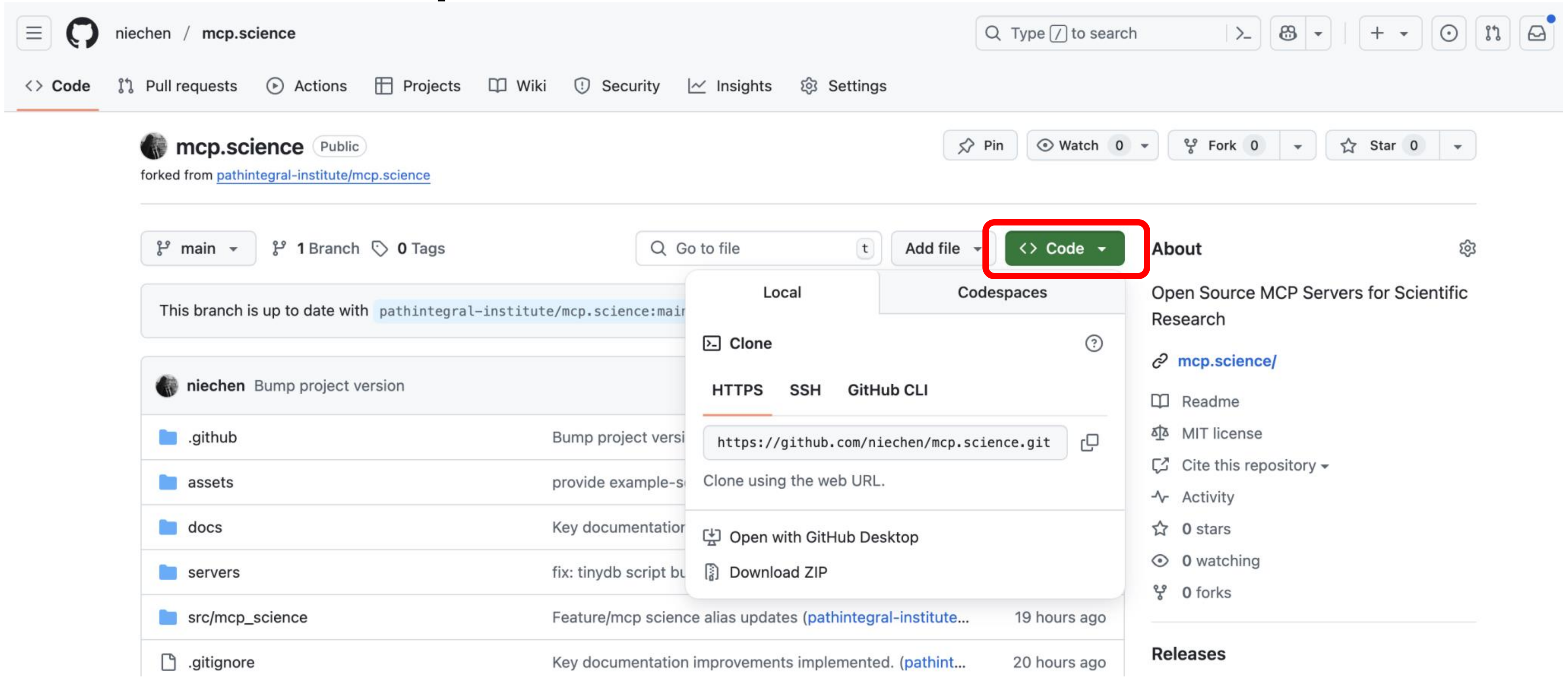
.github	Bump project version	2 hours ago

Open Source MCP Servers for Scientific Research

mcp.science/

Readme

Clone the repo to local



forked from pathintegral-institute/mcp.science

This branch is up to date with pathintegral-institute/mcp.science:main

niechen Bump project version

.github	Bump project versi	
assets	provide example-s	
docs	Key documentatior	
servers	fix: tinydb script bu	
src/mcp_science	Feature/mcp science alias updates (pathintegral-institute...	19 hours ago
.gitignore	Key documentation improvements implemented. (pathint...	20 hours ago

```
https://github.com/niechen/mcp.science.git
```

Clone using the web URL.

Open Source MCP Servers for Scientific Research

mcp.science/

Detailed Instruction on creating a github account and clone a repo for Windows and Mac

More Detailed Instruction on Contributing

Development Environment Setup

- create a new server directory under `servers/`
- When you are ready to submit your contribution, create a PR
- Once you create a PR on mcp.science, your server can be connected with AI agents conveniently by the command

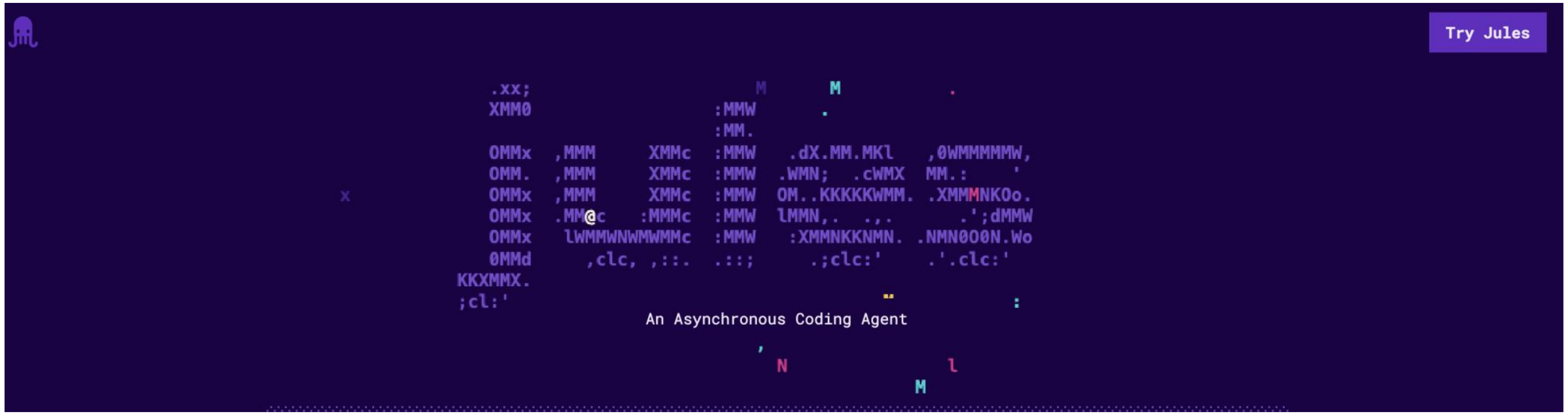
```
uvx mcp-science <server_name> -b <branch_name>
```

- When the PR is merged, your mcp server can be accessed by the simple command

```
uvx mcp-science <server_name>
```

Develop using jules.google

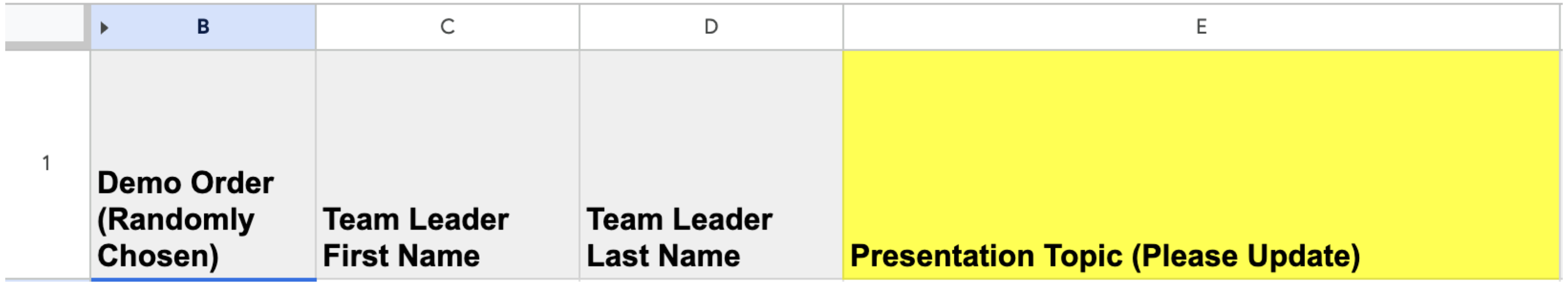
- Free to use
- Good for well defined tasks with clear requirements
- Integrates nicely with github development flow
- https://jules.google



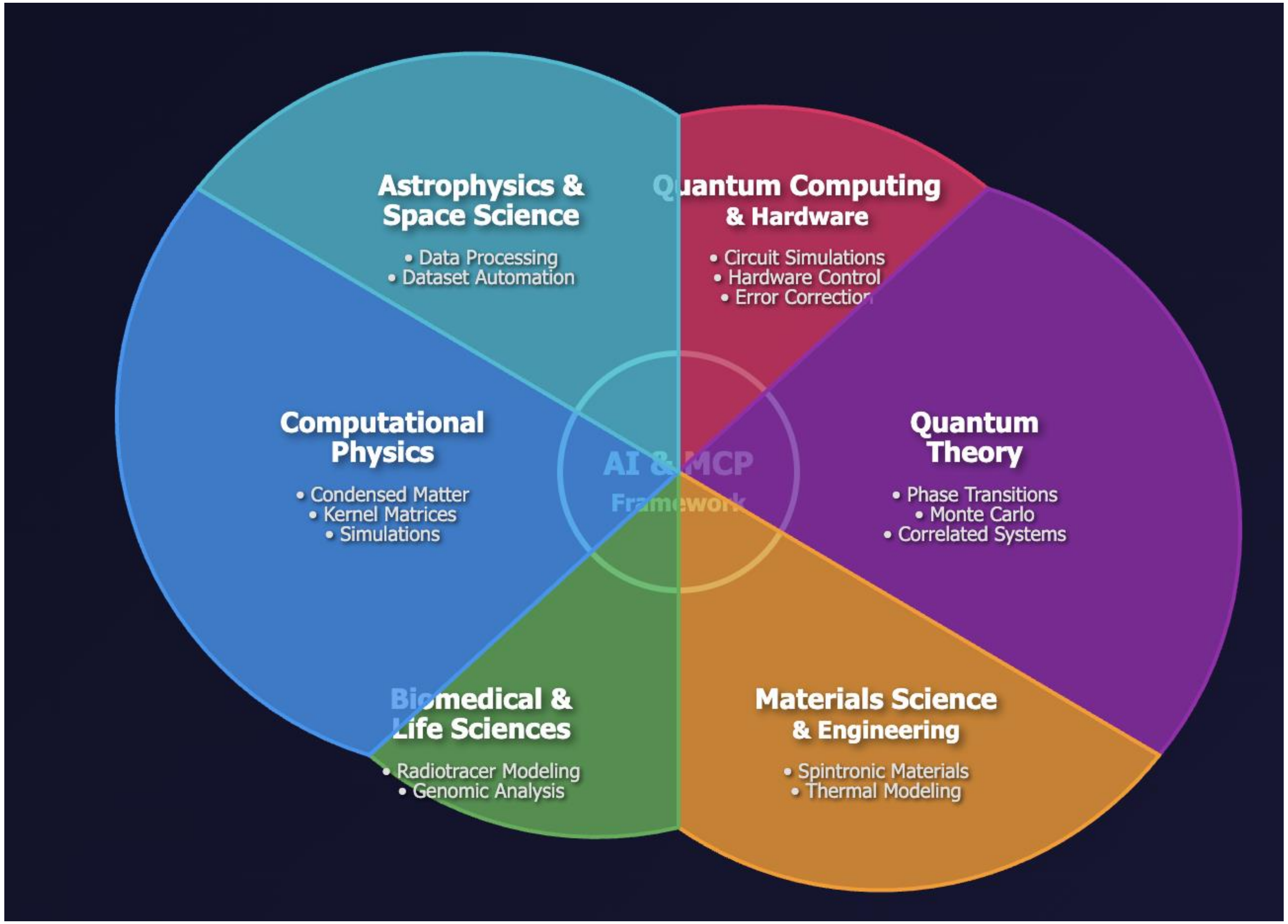
Part V: Logistics

Logistics: team formation and demo

- You are free to reorganize your team. Join, split, pivot…Just remember to update the Worksheet
- Please remember **specify your presentation topic** in the worksheet
- 5min demo. Order will be randomly chosen by 5pm Saturday.



	B	C	D	E
1	Demo Order (Randomly Chosen)	Team Leader First Name	Team Leader Last Name	Presentation Topic (Please Update)


Astrophysics & Space Science
• Data Processing
• Dataset Automation
Quantum Computing & Hardware
• Circuit Simulations
• Hardware Control
• Error Correction
Computational Physics
• Condensed Matter
• Kernel Matrices
• Simulations
AI & MCP Framework
Quantum Theory
• Phase Transitions
• Monte Carlo
• Correlated Systems
Biomedical & Life Sciences
• Radiotracer Modeling
• Genomic Analysis
Materials Science & Engineering
• Spintronic Materials
• Thermal Modeling

Logistics: Demo and Awards

- Awards:
  - First Prize: $4,000 in LambdaAI credits
  - Second Prize: $2,500 in LambdaAI credits
  - Third Prize: $1,500 in LambdaAI credits
- Selected by Amir Safavi-Naeini (QFARM), Wentao Jiang (Lambda), Xiao-Liang Qi (PII)
- Award selection criteria:
  - Scientific Impact & Relevance
  - Technical Excellence
  - Innovation & Creativity
  - Demonstration & Presentation
- Suggestions:
  - Choose a concrete goal
  - Present to the general audience
  - Stay on time
  - Try your best but don’t worry if you cannot finish in two days!

Tech Support Team

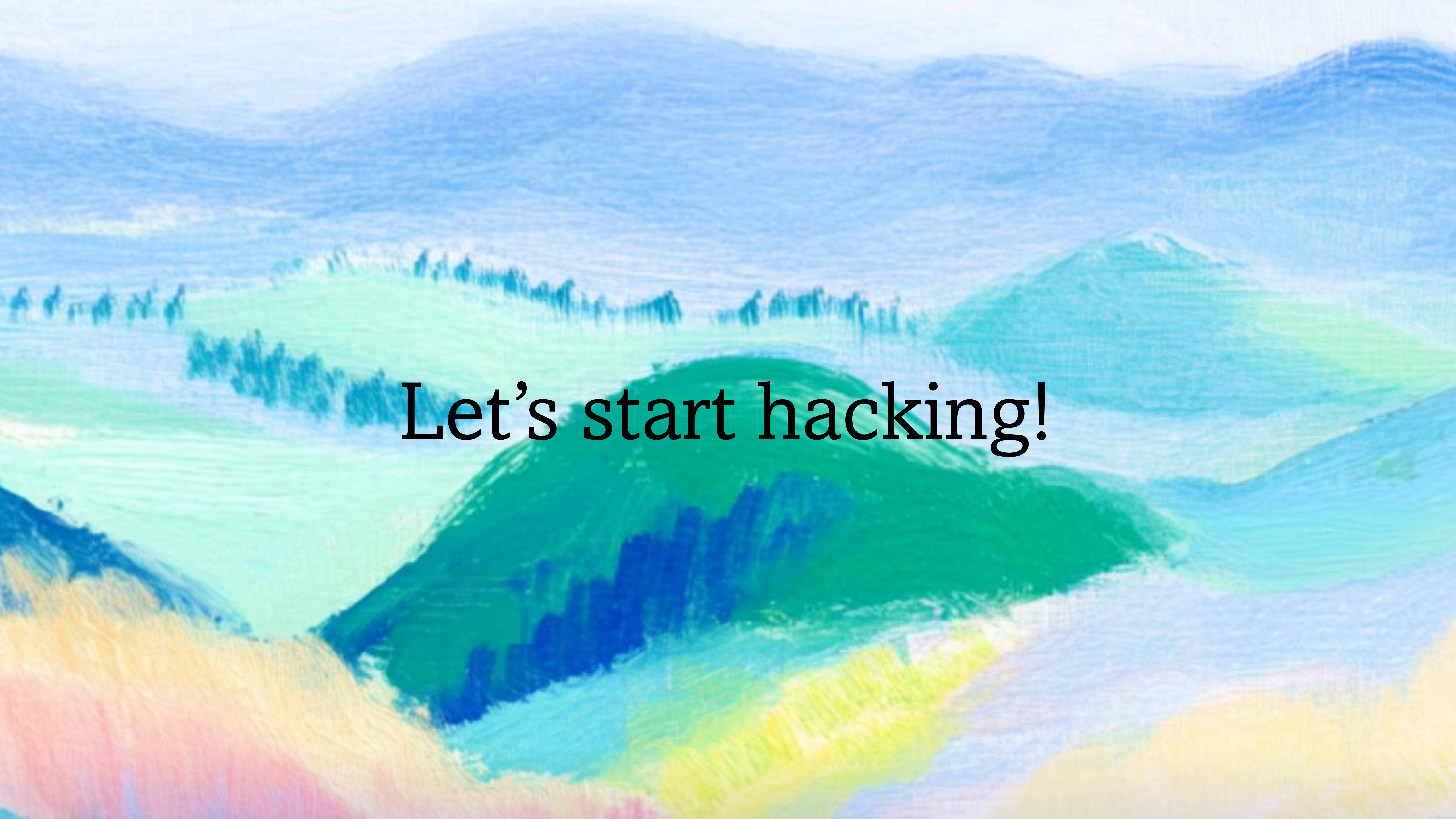
Name	Affiliation
Hsin Yen Chung	Head of Engineering, Cheehoo
Ju Huo	Technical Lead Manager at Google DeepMind
Wentao Jiang	Special Project Engineer at Lambda AI
Wei-Cheng Kuo	Co-founder and CPO, Cheehoo
Chen Nie	CEO of Path Integral Technology
Xiaoliang Qi	Stanford / Path Integral Institute
Guanhang Wu	Founder/CEO of Stealth Mode Startup
Joe Zhang	Senior Staff Software Engineer at Affirm, Inc.
Ethan Zou	Principle PM at GoodLeap

Useful Resources

- Official Introduction to MCP
  https://modelcontextprotocol.io/introduction
- mcp.science step-by-step guide
  - Building MCP server
  - Integrating MCP with LLM agents
  - Launch mcp.science existing servers
  - Using github and contributing your MCP server to mcp.science
- A tutorial course on MCP
- Installation of AI agents for the Hackathon

Sponsored by Lambda ShouCheng Zhang foundation

Follow Path Integral updates at pathintegral.xyz, Lucien_Science



Let’s start hacking!