Introduction to MCP

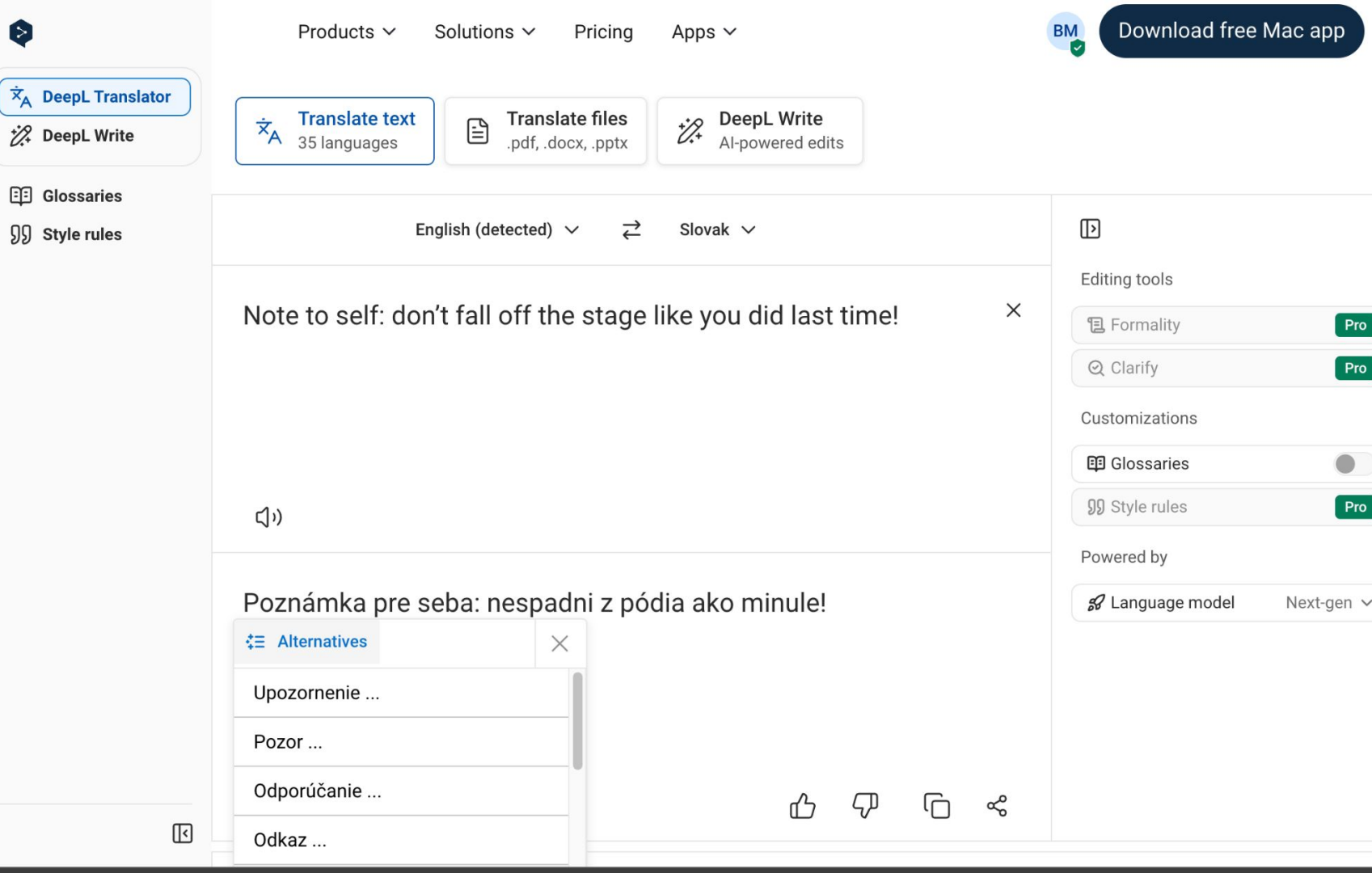
DeepL

Ben Morss

Developer Evangelist

(and Xoogler)

DeepL


Products
Solutions
Pricing
Apps
BM
Download free Mac app
DeepL Translator
DeepL Write
Glossaries
Style rules
Translate text
35 languages
Translate files
.pdf, .docx, .pptx
DeepL Write
AI-powered edits
English (detected)
Slovak
Note to self: don’t fall off the stage like you did last time!
Poznámka pre seba: nespadni z pódia ako minule!
Alternatives
Upozornenie ...
Pozor ...
Odporúčanie ...
Odkaz ...
Editing tools
Formality
Pro
Clarify
Pro
Customizations
Glossaries
Style rules
Pro
Powered by
Language model
Next-gen

DeepL Agent

New task
Artifacts
Workflows
All tasks

Recent
Workflow Creation Tutorial
Interactive Prompting Tutoria
GDPR EN-IT Glossary Creatic
GDPR EN-IT Glossary Creatic
Create GDPR English-Italian
Convert VTT to German via
Create blog article on machi
Audit DeepL blog localizatior
Research Agentic AI for Loca
Research Best Storytelling W
Research machine translatio

Contact us

M Morana

What do you want to achieve?

Compare the English (EN) and Italian (IT) version of the EU GDPR regulation at https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng . Then create an English-Italian glossary of the most relevant terms in the EU regulation and upload it to DeepL with the name “GDPR glossary”.

How to Use | Examples

Better Prompting
Learn how to write clear, specific prompts that produce better results

Workflows
Discover how to create and use workflows for complex tasks

Artifacts
Understand how artifacts store and organize your work results

README MIT license

deepl-mcp-server

npm v0.1.3-beta.0 license MIT smithery.ai ▲ 10

A Model Context Protocol (MCP) server that provides translation capabilities using the DeepL API.

Features

- Translate text between numerous languages
- Rephrase text using DeepL's capabilities
- Access to all DeepL API languages and features
- Automatic language detection
- Formality control for supported languages

Installation

You can install this using npm:

```
npm install deepl-mcp-server
```

Or you can clone this repository and install dependencies:

```
git clone https://github.com/DeepLcom/deepl-mcp-server.git
```

No packages published
Publish your first package

Contributors 4


akash-joshi Akash Joshi


morsssss Ben Morss


leoncheng57 Leon Cheng


smithery-ai[bot]

Languages

JavaScript 96.0% Dockerfile 4.0%

Suggested workflows

Based on your tech stack


Deno Configure

Test your Deno project


Datadog Synthetics Configure

Run Datadog Synthetic tests within your GitHub Actions workflow

MCP is eating the world —and it's here to stay

Young-jin Park
Software Engineer

Despite the hype, Model Context Protocol (MCP) isn't magic or revolutionary. But, it's simple, well-timed, and well-executed. At Stainless, we're betting it's here to stay.

What you'll need: **an AI client**

We'll be using **Claude Desktop**

- You can also use Windsurf, Cursor, etc.
- Long clients list: https://modelcontextprotocol.io/clients

bit.ly/mcp-deck-datacamp

bit.ly/mcp-code

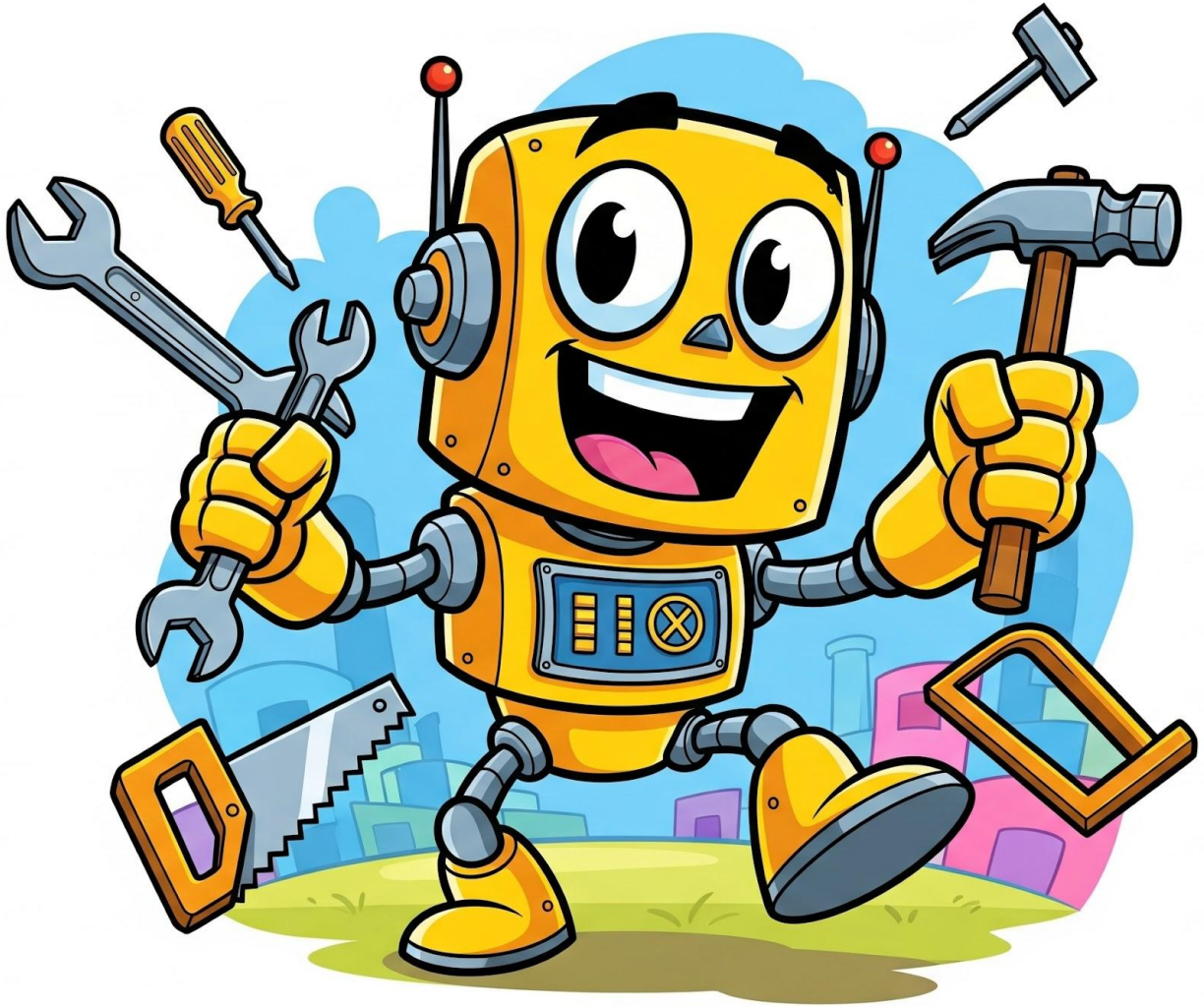
bit.ly/mcp-post


What is MCP?
Using MCP as a client
Building your MCP server


What is MCP?
Using MCP as a client
Building your MCP server

What is MCP?

- A new standard from Anthropic that gives AI clients to outside tools and resources
- uses JSON-RPC
- inspired by Language Server Protocol



What is MCP?

model

context

protocol

What’s an AI client?

- any app which embeds an LLM
- you could build your own with Langchain etc





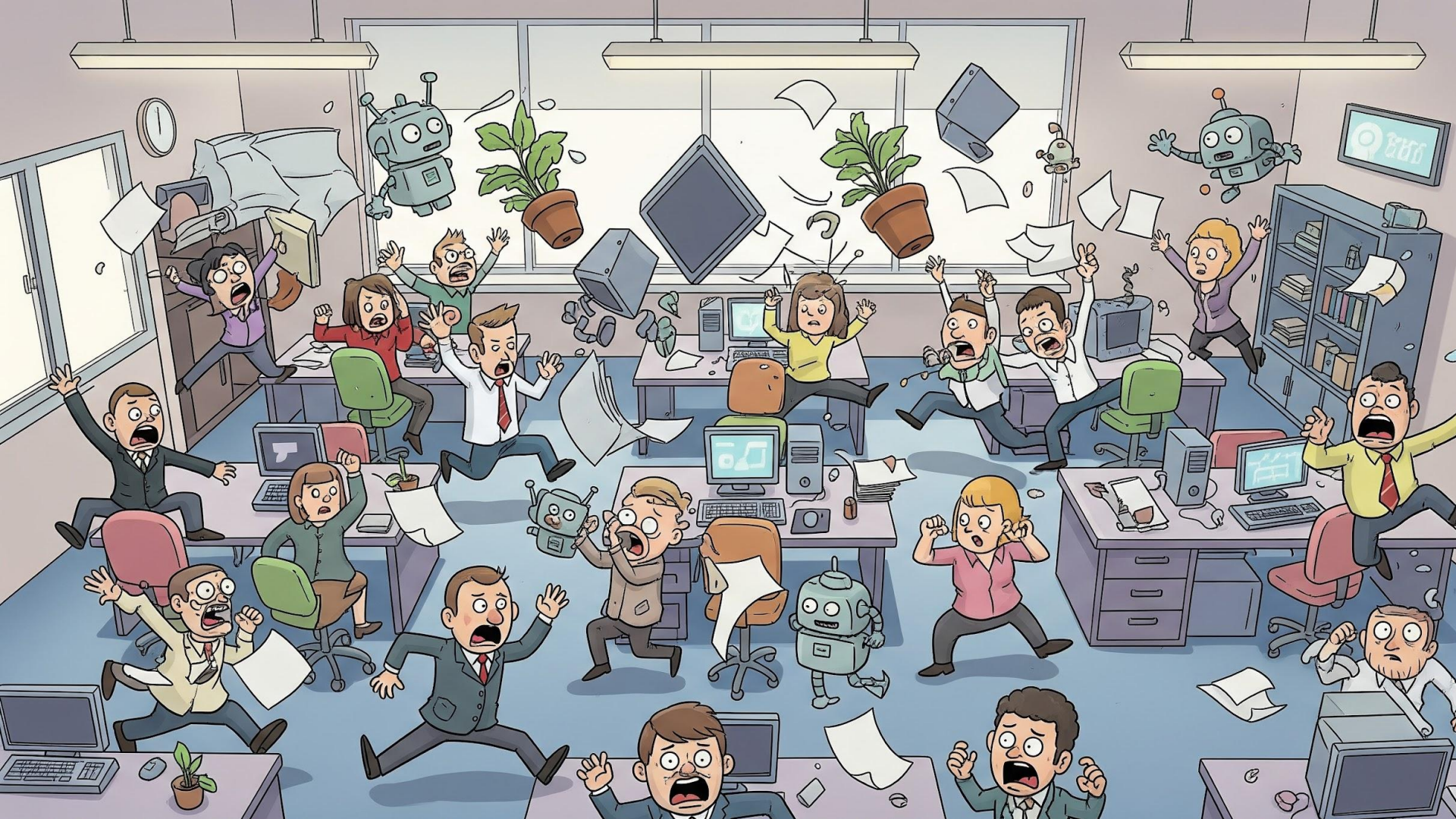
Can you write me some Python code that uses the DeepL API to translate "The rain in Spain falls mainly in the plain" into Spanish, making it more casual?

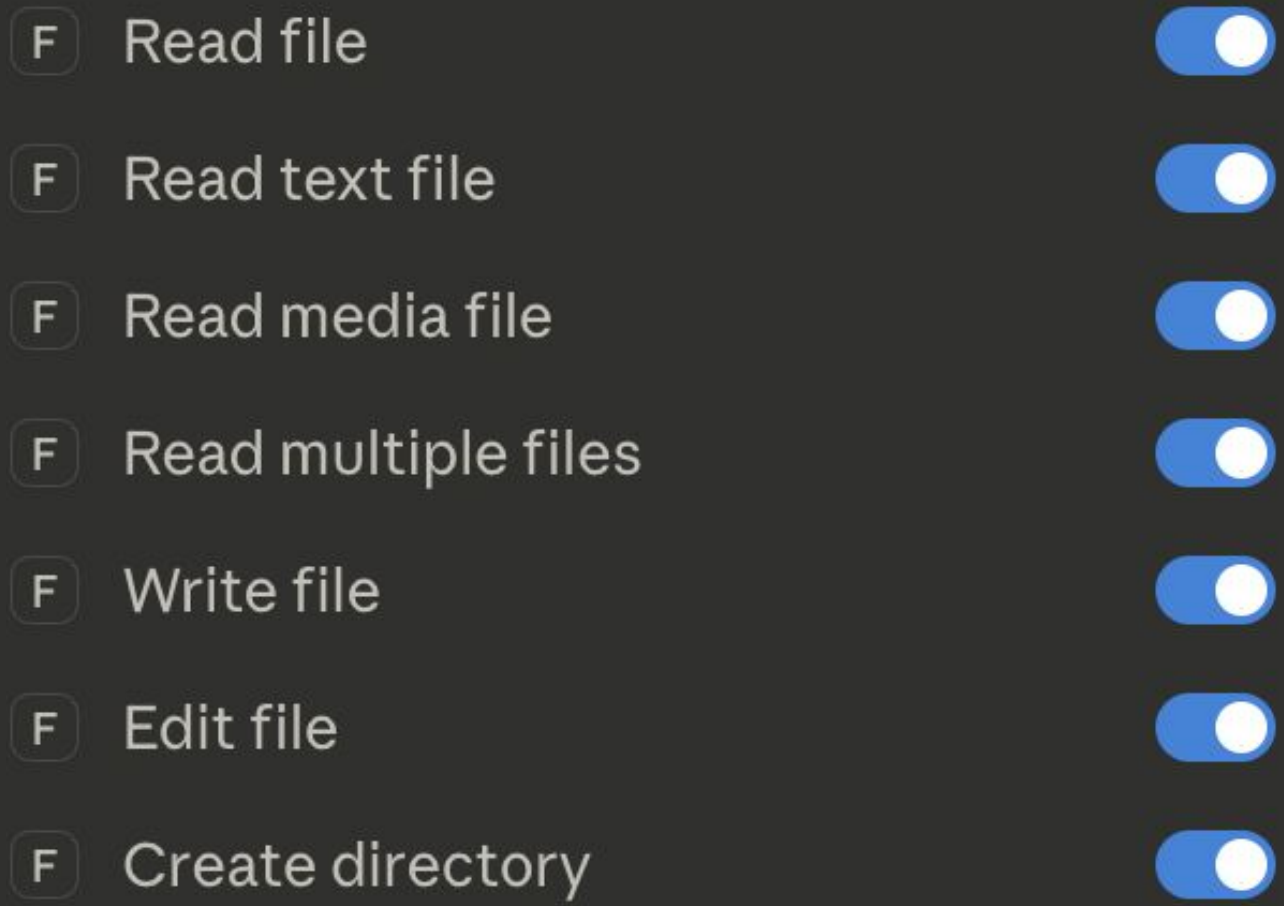
Context — none >

Here's Python code to translate that text into Spanish with casual formality using the DeepL API:

Why use an MCP?

```python
import deepl
import os

# Initialize the DeepL client
# Make sure to set your DEEPL_API_KEY environment variable
api_key = os.getenv('DEEPL_API_KEY')
if not api_key:
    raise ValueError("DEEPL_API_KEY environment variable is required")

translator = deepl.Translator(api_key)

# Text to translate
text = "The rain in Spain falls mainly in the plain"

try:
    # Translate to Spanish with casual formality
```




F Read file
F Read text file
F Read media file
F Read multiple files
F Write file
F Edit file
F Create directory

How does an MCP use tools?

Toolformer: Language Models Can Teach Themselves to Use Tools

Timo Schick **Jane Dwivedi-Yu** **Roberto Dessì**[†] **Roberta Raileanu**
Maria Lomeli **Luke Zettlemoyer** **Nicola Cancedda** **Thomas Scialom**

Meta AI Research [†]Universitat Pompeu Fabra

Abstract

Language models (LMs) exhibit remarkable abilities to solve new tasks from just a few examples or textual instructions, especially at scale. They also, paradoxically, struggle with basic functionality, such as arithmetic or factual lookup, where much simpler and smaller models excel. In this paper, we show that

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

PH

You now have access to a special calculation tool, which you can access at any time when you need to do arithmetic precisely. To use this calculation tool, simply output this format:

<Calculator>{expression}</Calculator>

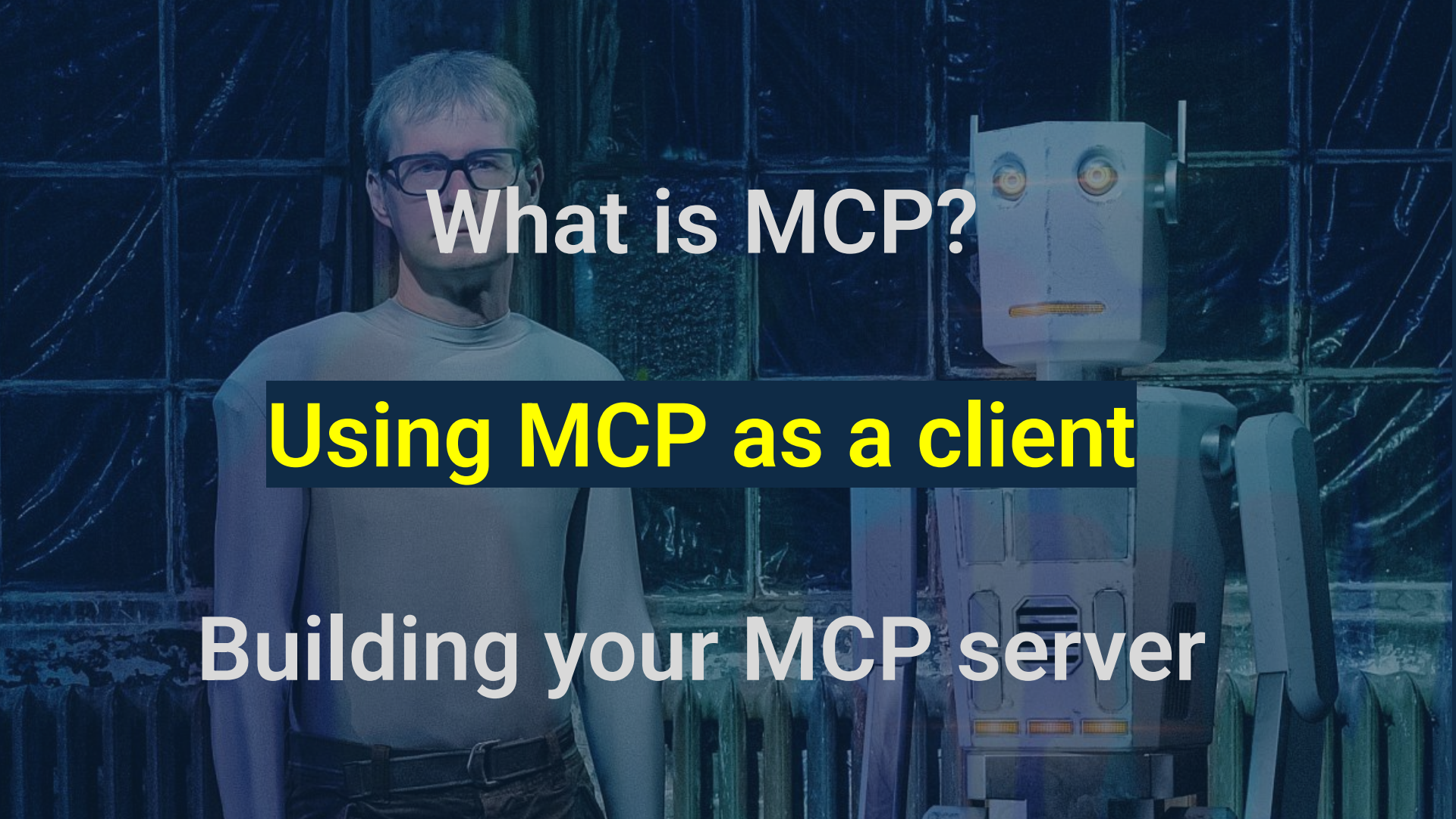
where {expression} is any arithmetic expression.

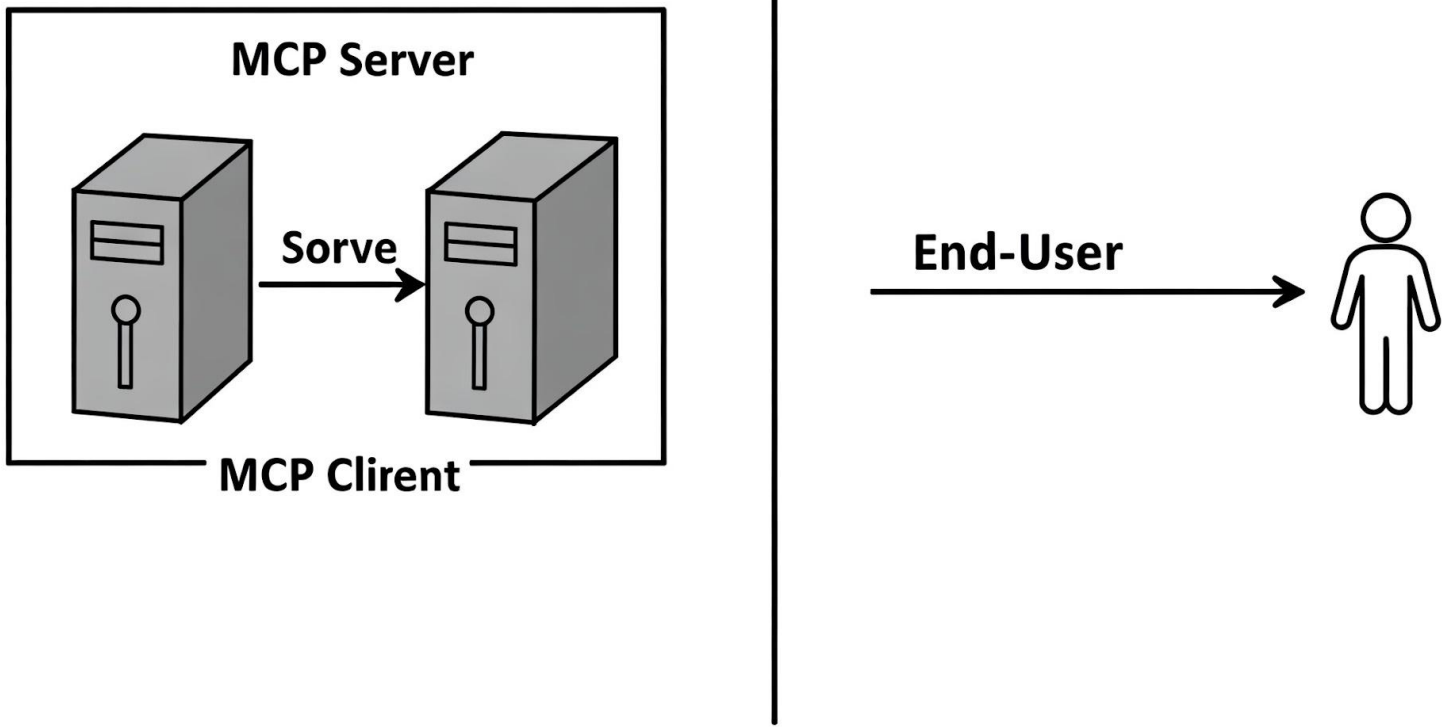
So, for example, to calculate 8 times 7, you'd output <Calculator>8 * 7</Calculator>

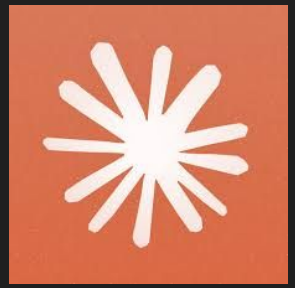
Tool use system prompt

When you call the Claude API with the `tools` parameter, we construct a special system prompt from the tool definitions, tool configuration, and any user-specified system prompt. The constructed prompt is designed to instruct the model to use the specified tool(s) and provide the necessary context for the tool to operate properly:

```
In this environment you have access to a set of tools you can use to ans t
{{ FORMATTING INSTRUCTIONS }}
String and scalar parameters should be specified as is, while lists and objec
Here are the functions available in JSONSchema format:
{{ TOOL DEFINITIONS IN JSON SCHEMA }}
{{ USER SYSTEM PROMPT }}
{{ TOOL CONFIGURATION }}
```


What is MCP?
Using MCP as a client
Building your MCP server


MCP Server
Sorve
MCP Clirent
End-User



```
{
  "mcpServers": {
    "jokes in JavaScript": {
      "command": "node",
      "args": [
        "/Users/sofo/Code/mcp/jokes-js/jokes.js"
      ]
    }
  }
}
```

Mac: ~/Library/Application Support/Claude/claude_desktop_config.json
Windows: %APPDATA%\Claude\claude_desktop_config.json
Linux: ~/.config/Claude/claude-desktop-config.json

```
{
  "mcpServers": {
    "deepl": {
      "command": "npx",
      "args": ["-y", "/Users/sofo/Code/deepl-mcp-server"],
      "env": {
        "DEEPL_API_KEY": "[redacted]"
      }
    }
  }
}
```

Mac: ~/Library/Application Support/Claude/claude_desktop_config.json
Windows: %APPDATA%\Claude\claude_desktop_config.json
Linux: ~/.config/Claude/claude-desktop-config.json

```
"jokes in Python": {
  "command": "/Users/sofo/.local/bin/uv",
  "args": [
    "--directory",
    "/Users/sofo/Code/deepl-mcp-server/workshops/jokes-python",
    "run",
    "fastmcp",
    "run",
    "jokes.py"
  ]
},
```

PH Can you translate "The rain in Spain falls mainly in the plain" for me, but make it very formal?

I'll translate that phrase and make it more formal for you.

D translate-text

Request

```
{
  `text`: `The rain in Spain falls mainly in the plain`,
  `formality`: `more`,
  `targetLang`: `es`
}
```

Response

```
La lluvia en España cae principalmente en la llanura
```

The formal Spanish translation is: **"La lluvia en España cae principalmente en la llanura"**

This maintains the formal tone you requested while preserving the meaning and even the alliterative quality of the original English phrase.

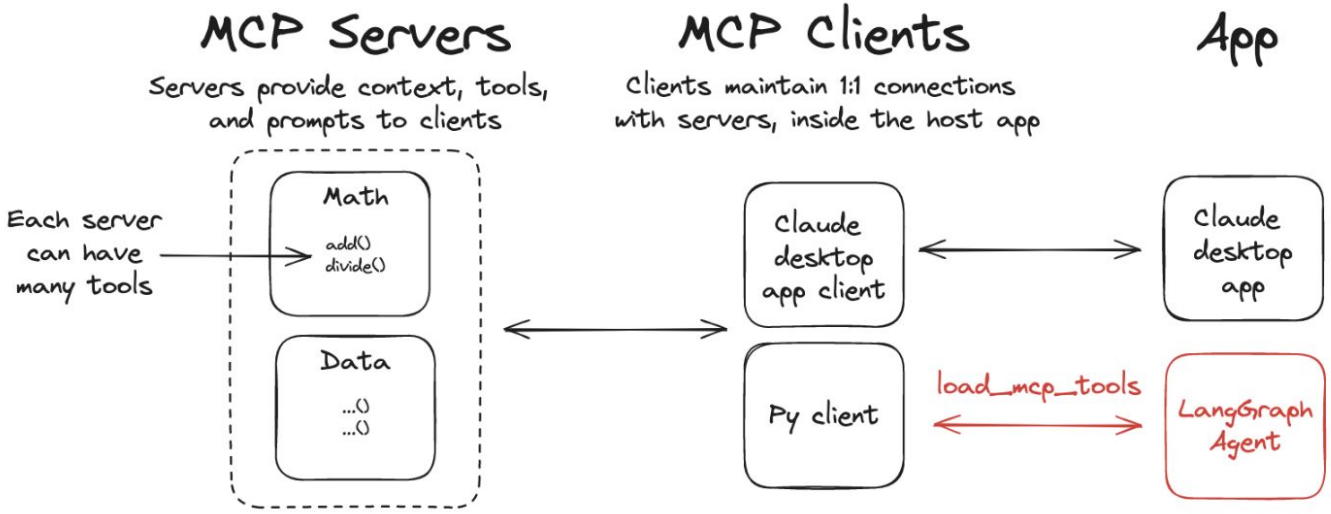
Home

Get started
Quickstart
LangGraph basics
Deployment

Prebuilt agents
Overview
Running agents
Streaming
Models
Tools
MCP Integration
Context
Memory
Human-in-the-loop
Multi-agent
Evals
Deployment
UI

MCP Integration

Model Context Protocol (MCP) is an open protocol that standardizes how applications provide tools and context to language models. LangGraph agents can use tools defined on MCP servers through the `langchain-mcp-adapters` library.



Install the `langchain-mcp-adapters` library to use MCP tools in LangGraph:

```
pip install langchain-mcp-adapters
```

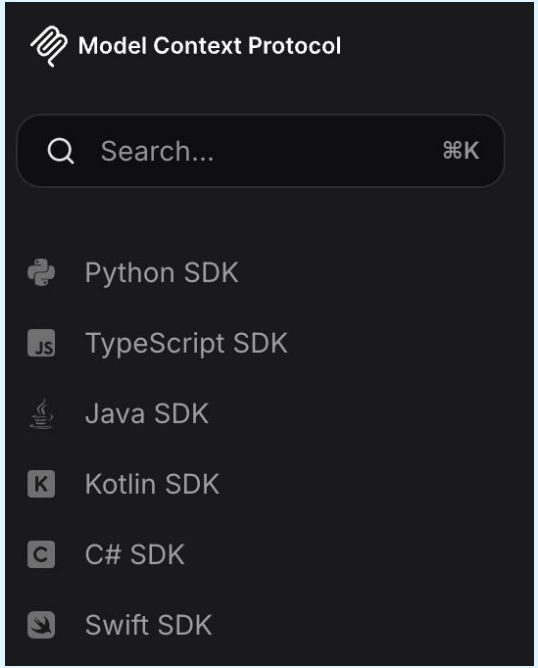

What is MCP?
Using MCP as a client
Building your MCP server

What you'll need: **this code**

bit.ly/mcp-code

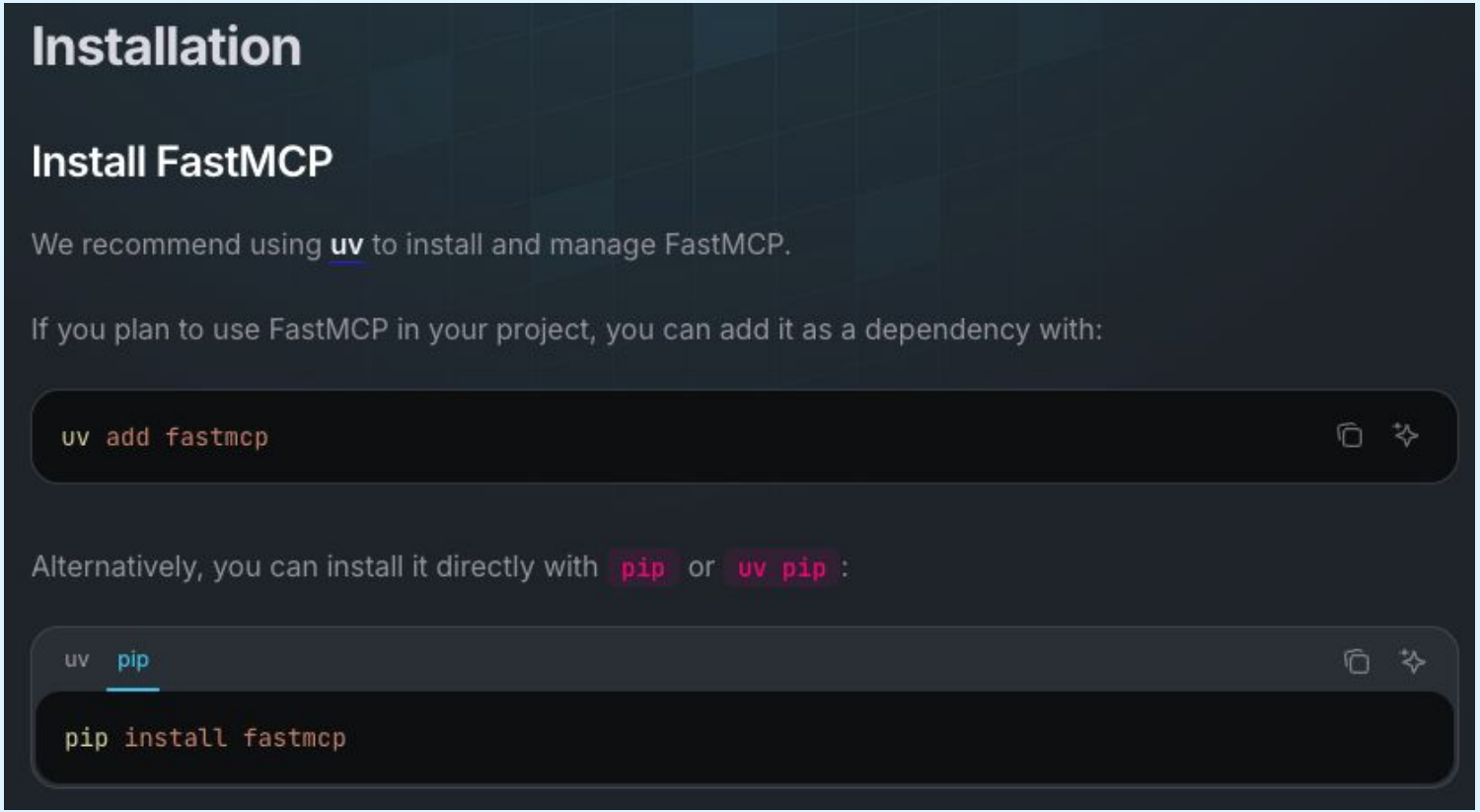
What you'll need: **a coding language**

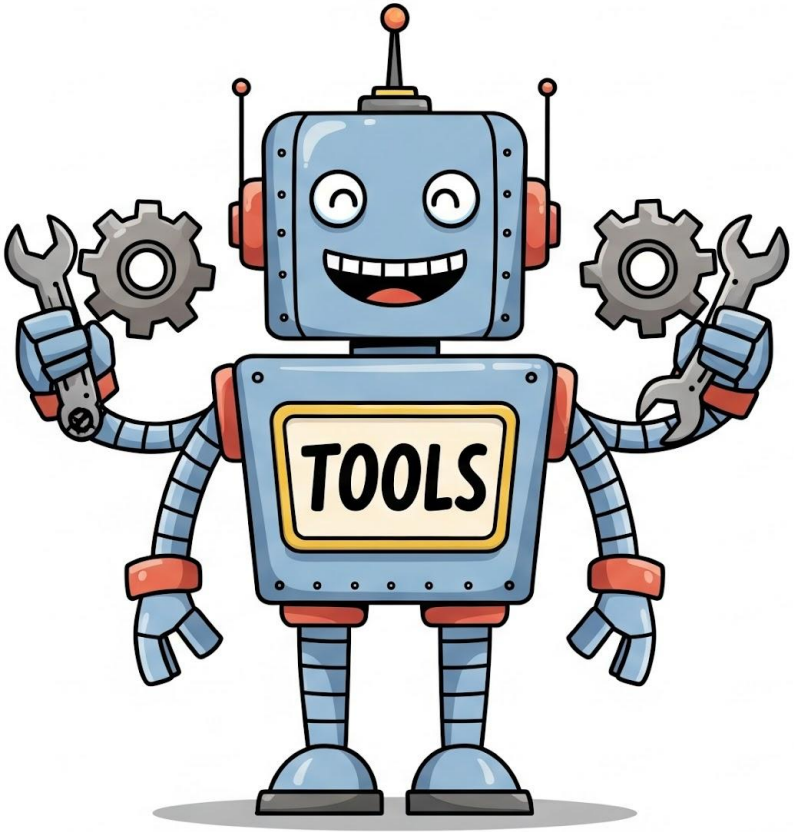
We'll show examples in JavaScript and Python

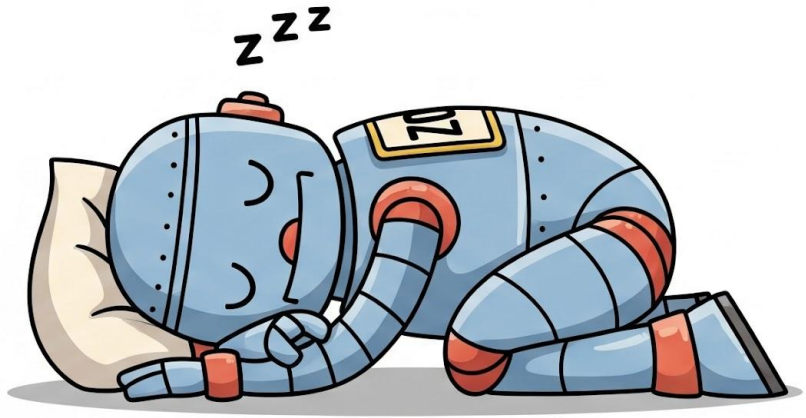


https://modelcontextprotocol.io/quickstart/server has links to client libraries

What you'll need:




TOOLS


z z z
RESOURCES


z z z
PROMPTS

Building your MCP server

1. instantiate an MCP server
2. write a function for each tool
3. define MCP configurations for tools and add to server
4. connect the server to a transport and go!

APIs you can use

We'll use **publicapi.dev/official-joke-api**

But you could use:

- yours
- DeepL's 😇
- **apipheny.io/free-api**

1. Instantiate an MCP server

```
from fastmcp import FastMCP

mcp = FastMCP("jokes")
```

2. Write a function for each tool

name	`get_joke_by_id`
description	`"Get a joke with a specific id (valid range: 1-451)"`
typed parameters	`id: Annotated[int, Field(ge=1, le=451)]`
{examples}	

2. Write a function for each tool

```
@mcp.tool
def get_consistent_joke() -> str:
    '''Tell the same joke, every time.'''
    return "What's brown and sticky?\nA stick!
```

3. Define MCP configurations for tools and add to server

4. Connect the server to a transport and go!

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

stdio

streamable http

```python
# A teeny-tiny version of the Jokes MCP server

from fastmcp import FastMCP
mcp = FastMCP("one joke")

@mcp.tool
def get_consistent_joke() -> str:
 return "What's brown and sticky? A stick!"

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

A tool that calls the Jokes API

```python
@mcp.tool
def get_joke() -> str:
  "Get a random joke"
  response = requests.get(API_BASE_URL + '/random_joke')
  json = response.json()
  return extract_joke(json)
```

A tool with input validation

```python
@mcp.tool
def get_joke_by_id(id: Annotated[int, Field(ge=1, le=451)]) -> str:
  "Get a joke with a specific id (valid range: 1-451)"
  response = requests.get(f"{API_BASE_URL}/jokes/{id}")
  json = response.json()
  return extract_joke(json)
```

MCP Inspector

In-depth guide to using the MCP Inspector for testing and debugging Model Context Protocol servers

The **MCP Inspector** is an interactive developer tool for testing and debugging MCP servers. While the **Debugging Guide** covers the Inspector as part of the overall debugging toolkit, this document provides a detailed exploration of the Inspector's features and capabilities.

```
fastmcp dev jokes.py
```

Using Javascript

server.tool() takes 5 params:

- **name**: string
- **description**: string
- *{**parameters**: schema}*
- **run**: (params: object) => { [type : string, content: string], …}
- *{**examples**}*

Javascript

```
const { McpServer } = require("@modelcontextprotocol/sdk/server/mcp.js");
const { StdioServerTransport } = require("@modelcontextprotocol/sdk/server/stdio.js");
const z = require("zod");
const axios = require("axios");
```

A tool with input validation

```
// Slightly more advanced: a tool which wants to be passed an integer
in a fixed range

server.tool(
  'get-joke-by-id',
  'Get a joke with a specific id. The id must be between 1 and 451.',
  {
    id: z.number().int().min(1).max(451)
  },
  jokeByID
);
```

A tool with input validation

```
async function jokeByID({ id }) {
  try {
    const res = await axios.get(`${API_BASE_URL}/jokes/${id}`);
    return mcpTextContentify(extractJoke(res.data));
  } catch (error) {
    handleJokeFetchError(error);
  }
}
```

```
return {
  content: [
    {
      type: "text",
      text: "I'm the text!"
    }
  ]
};
```

```
// Helper function which wraps a string or strings in the object
structure MCP expects (simplified)

function mcpTextContentify(param) {
 let strings = typeof(param) == 'string' ? [param] : param;

 const contentObjects = strings.map(
   str => ({
       type: "text",
       text: str
     })
 );
```

```
server.tool(
  'get-consistent-joke',
  'Tell the same joke, every single time',
  async () => mcpTextContentify(consistentJoke)
);
```

DeepL