



第36回 「Introduction to Model Context Protocol」 Anthropic 公式 Claude Academy

Claude Code

つなぐ側を作る

公式の教材を、順番に

原石工房

隠れた才能に気づき、花開く

この資料について



原石工房

Anthropic とは関係のない、非公式の資料です

公式の学習サイト Claude Academy の日本語訳です

原文（英語）Introduction to Model Context Protocol /
Implementing a client

サーバーが動くようになったので、今度はつなぐ側です

全51回の、今日はここ



原石工房

Claude Code 101

1 2 3 4 5 6 7 8 9 10 11 12

第1回～第12回

Claude Code in Action

13 14 15 16 17 18 19 20 21

第13回～第21回

スキル

22 23 24 25 26 27

第22回～第27回

手分け (subagents)

28 29 30 31

第28回～第31回

MCP 入門

32 33 34 35 36 37 38 39 40

第32回～第40回

MCP 応用

41 42 43 44 45 46 47 48 49 50 51

第41回～第51回

今日はここ 第36回

今日のレッスン



原石工房

Introduction to MCP の第5回

原題 Implementing a client

訳 つなぐ側を作る

つなぐ側は、自分のプログラムがサーバーと話すためのものです

① つなぐ側の組み立て

原題 Understanding the Client Architecture

ふつうは、どちらか一方だけを作る

つなぐ側か、サーバーか。両方は作りません

この学びでは、働き合いを見もらうために両方作ります

つなぐ側は、2つの部品でできている

MCP クライアント

やりとりを使いやすいするために、自分で書くひととまとり

クライアント セッション

サーバーへの、実際のつながり（道具立てに入っている）

つながりは、終わったら片づけけないといけない

だから自分のひととまとりで包んで、後始末を自動でやらせます

包むのは、片づけを任せるため

ほんとうの仕事では、つなぐ側かサーバーのどちらか一方だけを作ります

ここでは、働き合いを見もらうために両方作ります

部品は2つ。MCP クライアントと、クライアント セッションです

① なぜ包むのか



原石工房

ふつうは、どちらか一方だけを作る

つなぐ側か、サーバーか。両方は作りません

この学びでは、働き合いを見もらうために両方作ります

つなぐ側は、2つの部品でできている

MCP クライアント

やりとりをしやすいように、自分で書くひとまとまり

クライアント セッション

サーバーへの、実際のつながり（道具立てに入っている）

つながりは、終わったら片づけないといけない

だから自分のひとまとまりで包んで、後始末を自動でやらせます

つながりは、終わったときにきちんと片づけなければなりません

だから、自分のひとまとまりで包みます

そうすれば、後始末を自動でやらせられます

包むのは、片づけを任せるため

② 作るのは、2つの働き



原石工房

原題 Implementing Core Client Functions

作るのは、2つの働きだけ

list_tools()

道具の一覧をもらう

やりとりに触り、もともと入っている働きを呼び、道具を返す

call_tool()

道具を動かす

道具の名前と入れる中身をサーバーへ渡し、結果を返す

命令を打つ画面のプログラムは、つなぐ側をこう使います

・ Claude へ渡す、使える道具の一覧をもらう

・ Claude が「これを使う」と言ったときに、その道具を動かす

入口は2つ。渡すと、動かす

list_tools() サーバーが持っている道具を、全部もらってくる

call_tool() 道具の名前と入れる中身を渡し、動かして結果を返す

命令を打つ画面は、この2つを使います

③ つなぐ側だけ、先に試す



原石工房

原題 Testing the Client

つなぐ側だけ、先に試せる

```
uv run mcp_client.py
```

自分の MCP サーバーにつながり、使える道具を書き出します
道具の決め書きが、説明や入れる中身の形まで出てくるはずです

つなぐ側のファイルの終わりに、試用の仕掛けが付いています

できたら、アプリ本体も走らせてみる

```
uv run main.py
```

「report.pdf の書類には、何が書いてありますか」

と尋ねてみると、ひと続きの流れが動きます

部品だけでも、全体でも、確かめられる

uv run mcp_client.py で、そのまま走らせられます

自分のサーバーにつながり、使える道具を書き出します

道具の決め書きが、説明や入れる中身の形まで出てきます

④ 全部つないで動かす



原石工房

原題 Putting It All Together

裏では、こうなっています

① 自分のアプリが、つなぐ側を使って、使える道具をもらう

② その道具を、こちらの問いと一緒に Claude へ送る

③ Claude が「この道具を使おう」と決める

④ 自分のアプリが、つなぐ側を使って、その道具を動かす

⑤ 結果が Claude へ返り、Claude がこちらへ答える

つなぐ側は、自分のアプリの筋道と、サーバーの働きとの橋
強い道具を、自分の流れの中へ取り込みやすくなります

uv run main.py で、アプリ本体を
走らせます

「report.pdf の書類には、何が書いて
ありますか」と尋ねてみます

裏では、5つの段が動いています

つなぐ側が出てくるのは、①と④



MCP クライアント

自分で書くひとまとまり

後始末を自動でやらせるために包む

クライアント セッション

道具立てに入っている

サーバーへの、実際のつながり

list_tools()

道具の一覧をもらう

Claude へ渡すために

call_tool()

道具を動かす

Claude が指した道具を、実際に

部品2つ、働き2つ。それだけ

今回のまとめ



原石工房

ふつうは、つなぐ側かサーバーのどちらか一方だけを作る

部品は2つ。自分で書くひとまとまりと、実際のつながり

包むのは、後始末を自動でやらせるため

作る働きは2つ。道具をもらう、道具を動かす

つなぐ側は、自分のアプリとサーバーの橋になる



Introduction to MCP の第6回

原題 Defining resources

訳 資料を定義する

道具を呼ばずに、中身をそのまま渡す道を見ます

全51回の、37回目です

その橋は、何と何をつないでいますか



原石工房

隠れた才能に気づき、花開く