リスト1　効果音を提供するMCPサーバ（main_mcp_tools.py）

```
 1 from fastmcp import FastMCP
 2 from typing import Annotated
 3 from pydantic import Field
 4
 5 mcp = FastMCP(name="Sound player", port=5000, host="127.0.0.1")
 6
 7 def make_sound_play_tool():
 8     files = [
 9         "start_quiz.mp3",
10         "correct.mp3",
11         "try_again.mp3",
12         "incorrect.mp3",
13     ]
14
15     desc = f"play_sound: available audio files: {' '.join(files)}"
16
17     @mcp.tool()
18     def play_sound(
19             sound_file: Annotated[str, Field(description=desc)]):
20         """Play sound."""
21         print(f"playing {sound_file}")
22
23
24 if __name__ == "__main__":
25     make_sound_play_tool()
26
27     mcp.run(transport="http")
```

リスト2　AIクイズ・エージェント本体（main_ai_app.py）

```
001 #!/usr/bin/env python
002
003 import argparse
004 import json
005 import xml.etree.ElementTree as ET
006
007 import requests
008
009 URL_OLLAMA = "http://localhost:11434/api/chat"
010 URL_MCP_SERVER = "http://localhost:5000/mcp"
011
012
013 def get_system_prompt(tool_name, tool_desc, tool_input):
014     SYSTEM_PROMPT = f"""You are an AI quiz assistant.
015 Please present a quiz to the user and evaluate their input as either correct or incorrect.
016 The correct answer to the quiz must be one of the following: dog, cat, horse, or giraffe.
017
018 ## Message Format
019
020 Your response must follow the XML format shown below.
021 Only two tags are allowed within the <response> tag: <message> and <call>.
022 The <message> tag is mandatory, while the <call> tag is optional.
023
024 Follow this format strictly and do not add any extra lines.
025 Do not wrap the reply message in a Markdown code block.
026 If the user's answer is incorrect, provide additional hints and continue the quiz until the correct answer is given.
027
028 ## Example Response
029
030 Replace %TOOL_NAME%, %PARAM_NAME%, and %PARAM_VALUE% according to the MCP Tools section.
031
032 <response>
033 <message>Correct! Cool answer!</message>
034 <call><name>TOOL_NAME</name><arguments><%PARAM_NAME%>%PARAM_VALUE%</%PARAM_NAME%></arguments></call>
035 </response>
036
037 Sample <call> Tag
038
039 <call><name>add</name><arguments><lhs>1</lhs><rhs>2</rhs></arguments></call>
040
041 ## MCP Tools
042
043 You can use the following MCP Tools.
044
045 ### TOOL_NAME: {tool_name}
046
047 Description:
048 {tool_desc}
049
050 Input Schema:
051 {tool_input}
052
053     args = parser.parse_args()
054     return args
055
056
057 def get_response(model, messages):
```

```
    data = {
        "model": model,
        "messages": messages,
        "stream": False,
    }
    response = requests.post(URL_OLLAMA, json=data)
    if not response.ok:
        print(f"request failed due to {response.status_code}")
        raise Exception("invalid response")

    return response.json()["message"]


def init_mcp_server(mcp_session, url):
    """
    Init MCP server and get its tools.

    Continue MCP session with id = 2.

    Parameters
    ----------
    mcp_session: requests.Session
    url: MCP server URL

    Returns
    -------
    Tools List Response
      See https://modelcontextprotocol.io/docs/learn/architecture
      Tools Discovery > Tools List Response
    """
    mcp_session.headers.update({
        "Content-Type": "application/json",
        "Accept": "application/json, text/event-stream",
    })

    # initialization
    mcp_init_msg = {
        "jsonrpc": "2.0",
        "id": 1,
        "method": "initialize",
        "params": {
            "protocolVersion": "2025-06-18",
            "capabilities": {
                "elicitation": {}
            },
            "clientInfo": {
                "name": "example-client",
                "version": "1.0.0"
            }
        }
    }
    res0 = mcp_session.post(url, json=mcp_init_msg, stream=True)
    if not res0.ok:
        raise Exception("fail to initialize MCP server")

    mcp_session.headers.update(
        {
            "mcp-session-id": res0.headers["mcp-session-id"]
        }
    )

    # notification
    mcp_notification_msg = {
        "jsonrpc": "2.0",
        "method": "notifications/initialized",
    }
    res1 = mcp_session.post(url, json=mcp_notification_msg, stream=True)
    if not res1.ok:
        raise Exception("notifications/initialized failed")

    # get MCP tools
    mcp_list_tool_msg = {
        "jsonrpc": "2.0",
        "id": 2,
        "method": "tools/list",
        "params": {
            "cursor": "optional-cursor-value",
        },
    }
    res2 = mcp_session.post(
        URL_MCP_SERVER,
        json=mcp_list_tool_msg,
    )
    if not res2.ok:
        raise Exception("tools/list failed")

    return res2

```

```
146
147 def call_mcp_tool(mcp_session, url, mcp_id, tool_name, tool_args):
148     """
149     Call MCP tool.
150
151     Parameters
152     ----------
153     mcp_session: requests.Session
154     url: MCP server URL
155     mcp_id: mcp message id
156     tool_name: tool name
157     tool_args: xml.etree.ElementTree.Element
158     """
159     params = {
160         "name": tool_name,
161         "arguments": {c.tag: c.text for c in tool_args},
162     }
163
164     body = {
165         "jsonrpc": "2.0",
166         "id": mcp_id,
167         "method": "tools/call",
168         "params": params,
169     }
170
171     # print(f"call MCP tool: {body=}")
172
173     res = mcp_session.post(url,
174                            json=body)
175     if not res.ok:
176         raise Exception("tools/call failed")
177
178
179 def main():
180     args = parse_args()
181
182     # init MCP server
183     mcp_session = requests.Session()
184     res2 = init_mcp_server(mcp_session, URL_MCP_SERVER)
185
186     print("---- res2 -----")
187     print(res2.text)
188     print("---- res2 -----")
189     print("")
190
191     tools_json = json.loads(res2.text.split("¥r¥n")[1][6:])
192
193     # get first tools
194     assert len(tools_json["result"]["tools"]) == 1
195     tool = tools_json["result"]["tools"][0]
196
197     tool_name = tool["name"]
198     tool_desc = tool["description"]
199     tool_input = tool["inputSchema"]
200
201     # send system prompt to get first quiz
202     messages = [
203         {
204             "role": "system",
205             "content": get_system_prompt(tool_name, tool_desc, tool_input)
206         }
207     ]
208
209     res_msg = get_response(args.model, messages)
210     # Uncomment the line below to display the first quiz
211     print(res_msg)
212     messages.append(res_msg)
213     assistant_msg = res_msg["content"]
214
215     mcp_id = 2
216     while True:
217         root = ET.fromstring(assistant_msg)
218         msg = root.find("message").text
219
220         # call MCP tool
221         call = root.find("call")
222         if call is not None:
223             tool_name = call.find("name").text
224             tool_args = call.find("arguments")
225
226             call_mcp_tool(mcp_session, URL_MCP_SERVER,
227                           mcp_id, tool_name, tool_args)
228             mcp_id += 1
229
230         # show chat message
231         print(msg)
232         user_input = input("input answer: ")
233         user_input.strip()
```

```
234
235         # append user message
236         messages.append(
237             {
238                 "role": "user",
239                 "content": user_input,
240             }
241         )
242
243         res_msg = get_response(args.model, messages)
244         # Uncomment the line below to display the raw response
245         # print(res_msg)
246         assistant_msg = res_msg["content"]
247
248         # append agent message
249         messages.append(res_msg)
250
251
252
253 if __name__ == "__main__":
254     main()
```

リスト3　ollamaの起動

```bash
docker pull ollama/ollama:latest

# ollama serve 実行済みのコンテナを作成
docker run -d --rm -v ollama:/root/.ollama -p 11434:11434 --name ollama ollama/ollama

# 利用するモデルのダウンロード
docker exec ollama ollama pull gemma3:12b
```

リスト4　MCPサーバとAIアプリケーションの実行

```bash
# コンソール 1: MCP Server の実行
python src/main_mcp_tools.py

# コンソール 2: AI アプリケーションの実行.
python src/main_ai_app.py --model gemma3:12b
```

リスト5　AIアプリケーションの起動後に表示される内容

```text
---- res2 -----
event: message
data: {"jsonrpc":"2.0","id":2,"result":{"tools":[{"name":"play_sound","description":"Play
sound.","inputSchema":{"properties":{"sound_file":{"description":"play_sound: available audio files: start_quiz.mp3
correct.mp3 try_again.mp3
incorrect.mp3","type":"string"}},"required":["sound_file"],"type":"object"},"_meta":{"_fastmcp":{"tags":[]}}}]}}


---- res2 -----

Let's start a quiz! I'm thinking of an animal. It's known for its loyalty and is often called 'man's best friend'. What
animal am I?
input answer:
```

リスト6　コンソールで「cat」と入力

```text
input answer: cat [Enter]
Incorrect! While cats are wonderful pets, my animal is a bit different. This animal often barks and wags its tail. Think
about animals that are known for fetching and guarding.  It's a common pet and comes in many breeds. Is it a dog, cat,
horse, or giraffe?
input answer:
```

リスト7　コンソールで「dog」と入力

```text
input answer: Oh, is it a dog? [Enter]
Correct! You got it! My animal is a dog. Well done!
input answer:
```

リスト8　play_soundという音再生ツールが利用可能

```json
{
  "jsonrpc": "2.0",
  "id": 2,
  "result": {
    "tools": [
      {
        "name": "play_sound",            # ツール名
        "description": "Play sound.",
        "inputSchema": {                 # ツールへの入力
          "properties": {
```

```
            "sound_file": {
              # start_quiz.mp3 などのファイルが再生可能であることが記載されている
              "description": "play_sound: available audio files: start_quiz.mp3 correct.mp3 try_again.mp3 incorrect.mp3",
              "type": "string"
            }
          },
          "required": [
            "sound_file"
          ],
          "type": "object"
        },
        "_meta": {
          "_fastmcp": {
            "tags": []
          }
        }
      }
    ]
  }
}
```

リスト9　AIクイズ・アシスタントとして振る舞うように指示

```text
You are an AI quiz assistant.
Please present a quiz to the user and evaluate their input as either correct or incorrect.
The correct answer to the quiz must be one of the following: dog, cat, horse, or giraffe.

あなたは AI クイズアシスタントです。
ユーザにクイズを出し、回答が正解か不正解か答えて下さい。
クイズの答えは犬・猫・馬・キリンのいずれかにして下さい。
```

リスト10　システム・プロンプトのplay_sound説明箇所

```text
### TOOL_NAME: play_sound

Description:
Play sound.

Input Schema:
{'properties': {'sound_file': {'description': 'play_sound: available audio files: start_quiz.mp3 correct.mp3 try_again.mp3
incorrect.mp3', 'type': 'string'}}, 'required': ['sound_file'], 'type': 'object'}
'}]
```

リスト11　role, contentを持つJSONが返される

```json
{
  'role': 'assistant',
  'content':
     "<response>
       <message>...</message>
       <call>
         <name>play_sound</name>
         <arguments>
           <sound_file>start_quiz.mp3</sound_file>
         </arguments>
       </call>
     </response>"}
```

リスト12　MCPツール呼び出しのXMLをJSON形式に変換

```text
<call>
  <name>play_sound</name>
  <arguments>
    <sound_file>correct.mp3</sound_file>    // ここを JSON の "sound_file": "try_again.mp3" に変換
  </arguments>
</call>

↓
{
  "name": play_sound,
  "arguments": {"sound_file": "correct.mp3"}
}
```

リスト13　MCPサーバに送信するJSONファイルの内容

```
{
   'jsonrpc': '2.0',
   'id': 2,
   'method': 'tools/call',
   'params': {
      'name': 'play_sound',
      'arguments': {
         'sound_file': 'correct.mp3'
```

```
      }
    }
}
```

リスト14　正解数を尋ねた結果「one quiz」と答えた

```text
input answer: How many quizzes have I answered correctly so far?
You've correctly answered one quiz so far!
```