Agent Architecture

agent.py → uv run agent.py

↓

--main--

asyncio.run(main())

↓

main()

config → [contains different mcp-servers]
1) models.json → list of models.
2) profile.yaml → agent configs.

1) open("config/profiles.yaml")

"""gets all server lists"""

mcp_servers:-
1) mcp_server_1.py id: Math
2) mcp_server_2.py id: documents
3) mcp_server_3.py id: websearch.

core
1) context.py
2) loop.py
3) session.py
4) strategy.py

core.sessions → mcp
→ multimcp

MultiMCP(mcp_servers)

↓

await initialize()

↓

updating all tools into its local class variable.

MultiMCP.session_tools[server_key] have all tools.

from core.context import AgentContext.

In agent.py

```
from core.loop import AgentLoop

try:
user_input = input("User Query>")
```

→ get the context = AgentContext(...)

agent = AgentLoop(context)

→ "agent.run()"

It will run the loop

The loop contains below flow:-

loop

perception

① run_perception

→ gets the user input

→ extract the perception.

→ return perception.

```
{ intent: <—>,
  entities = [],
  tool_hint = None,
  tags = []
  selected_servers = }
```

1) selected_servers

2) selected_tools

planning :-

generate_plan ← Decision

plan = generate_plan(..)

① From memory item → memory_texts

② Load_prompt(planning mode / exploration_mode)

③ prompt with template

④ Added previous interation to avoid repeating same state.

[Bug fix]

LLM_Query → using

raw = model.generate_text (prompt)

↓

Taking raw output and regex Strip the python command to run it in sandbox

↓

Action

run_python_sandbox()

↓

solve_fn()

↓

result

↓

final_answer: {result}