Task 2: AgentsSDK 40 Question And Answers

Compiled by Ashna Ghazanfar

Slot: Tuesday 7 to 10

Roll number: 423485

1. What is the main advantage of custom tool behaviour functions?

It helps you control how an agent use tool, we can create custom logics hence giving us full control over the agent

2. Which method is used to execute and agent Asynchronously?

By using await Runner.run() we can run the agent Asynchronously it lets multiple tasks run without blocking hence ideal for fast executions

3. What is the purpose of RunContextWrapper?

It wraps extra data and passes it into agent's thinking, tools and output handling

4. What does Runner.run_sync() do?

It runs agent in a blocking way (Synchronous way) until the result is available

5. What does extra='forbid' do in pydantic config?

It rejects any extra field that isn't present in our model ensuring Strict validation

6. Advantage of strict schemas?

It prevents mismatches and typos making our data safe from any unexpected inputs

7. What happens when combining StopAtTools with tool names?

Agent stops immediately after calling all the tools in the list giving us control over the system Useful with multiple agents

8. Purpose of Context in Open AI SDK?

It's a container for the runtime data like user Info settings... we pass it across agents tools etc.

9. Key factor for choosing tool_use_behavior?

It controls how freely an agent can call tools like auto, never and manual We can pick from these on the basis of how Independent we want the agent to be

10. What is handoff_description?

When using multiple agents this helps displaying the description of the agent after handoff to other agent

11. What does is_final_output means?

It tells us if the final answer from the tool is given or executed if true the agent will stop immediately if false the agent will continue calling more tools

12. Hosted v/s Function tools?

Hosted:- provided by open AI, external tools Function:- these are custom tools built using python

13. How to convert agent to tool?

By wrapping an agent as a function tool so it can be called like other tools

14. What method returns all tools available to an agent?

By using the agent.get_tool() method

15. What is the first parameter of every function tool?

The first parameter is always context (Just like we have self while making classes)

16. Purpose of get_system_prompt() method?

It returns the overall system prompt for the agent it is used to define roles and tasks to the agent

17. Input Vs output guardrails?

InputGuardrail: checks users input before agent sees it OutputGuardrails: checks agent output before it's returned to user

18. What is the instructions parameter in agent?

It is the main prompt that defines the agents personality and job it will do

19. What does the reset_tool_choice parameter control?

It resets the tool selection between agents after checking if the agent remembers the last tool If it remembers (True) it selects every step again It does not (False) reuse the last tool

20. What happens if the tool raises an exception?

It will raise an error and the tool will stop executions

21. What does the clone() method do?

Creates exact copy of the agent with all settings, helpful for testing

22. What is ToolToFinalOutputFunction?

It is a handler that defines how tool results are converted to a final agent output Used for summarizing multiple tool results

23. What is the return type of Runner.run()?

It returns a Run result in the form of object Contains final message and tool results

24. What happens if a custom tool behaviour function returns is_final_output=False?

The agent will keep running and calling tools Only after using it as True the agent will stop at the last tool

25. When to use StopAtTools Vs stop_on_first_tool?

stop_on_first_tool = stop after any tool StopAtTool only stops if a specific tool is used

26. Default value of tool_use_behavior?

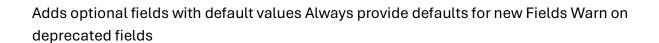
It is AUTO means the agent can decide freely when to use tools

27. What's the difference between run_llm_again and stop_on_first_tool in terms of performance?
run_llm_agent re runs the LLM again and again it is slow stop_on_first_tool is fast but basic
28. Which pydantic v2 decorator is used for field validation?
By using @field_validator()
29. What is the purpose of model_settings in an agent?
Sets LLM related settings like temperature, tokens, penalty
30. How do you enable non-strict mode for flexible schemas?
By using extra="allow" in your pydantic config It basically does not raise error if we get extra fields not defined in our schema
31. What does the handoff_description parameter do?

It tells the next tool what this agent was doing and why it's passing control

backward compatibility?

32. How do you implement schema evolution while maintaining



33. Can dynamic instructions be Async functions?

Yes they can be Async

34. What happens when the tool_use_behavior is set to stop_on_first_tool?

Agent stops right after the first tool call and skips all the tools ahead

35. What's the difference between mutable and immutable context patterns?

Mutable can be changed anytime Immutable cannot be changed once fixed

36. What causes the error 'additional Properties should not be set for object types'?

This happens when we send extra fields in the input that the schema doesn't allow Happens in strict mode (extra= 'forbid')

37. When should you use non-strict schema's strict schemas?

Can be used while handling flexible data For backend compatibility Accepting data from external sources

38. In a custom tool behaviour function, what parameters does it receive?

It receives context (first param), tool_call(tool_input and tool_name) Provides us flexibility

39. What does Runner.run_streamed() return?

Returns an Async generator that yields events in real time step by step

40. How do you create dynamic instructions that change based on context?

By customizing agent's instructions Using a function instead of string in instructions And then accessing value from the context object and returning them Hence creating dynamic instructions for the agent

Thanks for reading!