□ 100 Q&A on Hooks

Basics of Hooks

1.

   Q: What are hooks in Agentic AI?
   A: Hooks are callback functions that let you inject custom logic at different stages of an agent’s execution.

2.

   Q: Why are hooks important?
   A: They allow monitoring, debugging, logging, and modifying behavior without altering core agent code.

3.

   Q: Name the two main types of hooks.
   A: Agent-level hooks and Run-level hooks.

4.

   Q: What is the difference between agent-level and run-level hooks?
   A: Agent-level hooks apply globally to an agent, while run-level hooks apply only to a single execution (run).

5.

   Q: Can hooks alter inputs and outputs?
   A: Yes, hooks can inspect and sometimes modify inputs, outputs, or intermediate states.

6.

   Q: Are hooks synchronous or asynchronous?
   A: They can be either, depending on the framework implementation.

7.

Q: Do hooks increase observability?
A: Yes, they enable detailed tracking of agent actions, inputs, and responses.

8.

Q: What is the primary purpose of hooks?
A: To customize, monitor, or extend agent behavior dynamically.

9.

Q: Do hooks require modifying the agent’s internal logic?
A: No, they can be added externally, making them non-intrusive.

10.

Q: Can multiple hooks be attached at once?
A: Yes, multiple hooks can be registered and executed in sequence.

Agent-Level Hooks

11.

Q: What are agent-level hooks?
A: Hooks that are defined once and apply across all runs of an agent.

12.

Q: Example use case of agent-level hooks?
A: Adding custom logging for every interaction the agent processes.

13.

Q: When would you use agent-level hooks?
A: When you need global monitoring or behavior enforcement across multiple executions.

14.

Q: Do agent-level hooks persist across sessions?
A: Yes, they remain as long as the agent object exists.

15.

Q: Can agent-level hooks enforce security rules?
A: Yes, by intercepting inputs and filtering unsafe requests.

16.

Q: What is the main drawback of agent-level hooks?
A: They affect all runs, which might not always be desirable.

17.

Q: Can agent-level hooks be removed?
A: Yes, they can usually be deregistered.

18.

Q: Which is better for debugging: agent-level or run-level hooks?
A: Run-level hooks, since debugging is usually run-specific.

19.

Q: Can agent-level hooks modify output format?
A: Yes, they can transform agent outputs globally.

20.

Q: Can you add multiple agent-level hooks of the same type?
A: Yes, they will run in sequence.

Run-Level Hooks

21.

Q: What are run-level hooks?
A: Hooks applied only during a single execution (run) of an agent.

22.

Q: Do run-level hooks persist after the run ends?
A: No, they exist only during that run.

23.

Q: Example use case of run-level hooks?
A: Tracking a specific conversation for debugging.

24.

Q: Why use run-level hooks over agent-level?
A: For temporary, case-specific customization.

25.

Q: Do run-level hooks interfere with other runs?
A: No, they are isolated to the current execution.

26.

Q: Can run-level hooks modify intermediate steps?
A: Yes, they can inspect and modify step outputs.

27.

Q: Can run-level hooks be used for A/B testing?
A: Yes, by applying different behaviors per run.

28.

Q: Can you attach run-level hooks dynamically?
A: Yes, before starting the run.

29.

Q: Are run-level hooks ideal for debugging rare issues?
A: Yes, since they won’t affect normal runs.

30.

Q: Do run-level hooks reduce risk of unintended global effects?
A: Yes, since they are temporary.

Common Hook Stages

31.

Q: Name some stages where hooks can be applied.
A: Before input, after output, before tool use, after tool use, error handling.

32.

Q: What is an “on_start” hook?
A: A hook that runs when the agent execution begins.

33.

Q: What is an “on_end” hook?
A: A hook that runs after the agent finishes execution.

34.

Q: What is an “on_error” hook?
A: A hook triggered if an error occurs.

35.

Q: What is an “on_step” hook?
A: A hook triggered at each reasoning/tool step.

36.

Q: Which hook is best for monitoring latency?
A: On_step or on_end hooks.

37.

Q: Which hook is best for input validation?
A: On_start hook.

38.

Q: Which hook is best for sanitizing final outputs?
A: On_end hook.

39.

Q: Which hook is best for retry logic?
A: On_error hook.

40.

Q: Which hook is best for tool usage monitoring?
A: On_tool_start or on_tool_end hooks.

Practical Applications

41.

Q: Can hooks help in logging user queries?
A: Yes, via on_start hooks.

42.

Q: Can hooks redact sensitive data before processing?
A: Yes, with pre-processing hooks.

43.

Q: Can hooks enforce word limits on outputs?
A: Yes, by modifying outputs in on_end hooks.

44.

Q: Can hooks control which tools the agent uses?
A: Yes, by validating tool calls.

45.

Q: Can hooks track token usage?
A: Yes, by monitoring inputs and outputs.

46.

Q: Can hooks be used to cache results?

A: Yes, to avoid recomputing for repeated queries.

47.

Q: Can hooks be used for debugging infinite loops?
A: Yes, by checking execution steps.

48.

Q: Can hooks send metrics to monitoring dashboards?
A: Yes, hooks can export logs and stats.

49.

Q: Can hooks enforce compliance filters?
A: Yes, by blocking prohibited outputs.

50.

Q: Can hooks notify external services during execution?
A: Yes, via webhooks or API calls.

Advanced Hook Usage

51.

Q: Can hooks implement rate limiting?
A: Yes, at input validation stage.

52.

Q: Can hooks modify system prompts dynamically?
A: Yes, via pre-execution hooks.

53.

Q: Can hooks store execution traces?
A: Yes, for later debugging.

54.

Q: Can hooks assist in role-based access control?
A: Yes, by validating user identity at start.

55.

Q: Can hooks prevent unsafe tool commands?
A: Yes, by filtering tool calls.

56.

Q: Can hooks be conditional?
A: Yes, they can check conditions before acting.

57.

Q: Can hooks trigger retries automatically?
A: Yes, via error-handling hooks.

58.

Q: Can hooks limit execution steps?
A: Yes, by monitoring run progress.

59.

Q: Can hooks simulate human feedback?
A: Yes, by intercepting outputs and adjusting responses.

60.

Q: Can hooks be used for testing agent behavior?
A: Yes, by mocking inputs/outputs.

Coding & Implementation

61.

Q: In Python, how are hooks usually implemented?

A: As functions passed into the agent or runner.

62.

Q: Can hooks be async functions in Python?
A: Yes, for non-blocking tasks.

63.

Q: What argument do most hooks accept?
A: Execution context or event data.

64.

Q: Can hooks raise exceptions?
A: Yes, to stop execution.

65.

Q: Can hooks log intermediate reasoning steps?
A: Yes, by accessing chain-of-thought metadata.

66.

Q: Can hooks modify temperature or top_p dynamically?
A: Yes, by adjusting model parameters before generation.

67.

Q: Are hook calls ordered?
A: Yes, usually executed in registration order.

68.

Q: Can hooks return modified data?
A: Yes, depending on their stage.

69.

Q: Can hooks capture tool errors?
A: Yes, via tool-specific error hooks.

70.

Q: Can hooks interact with external APIs?
A: Yes, they can call external services.

Debugging & Monitoring with Hooks

71.

Q: How do hooks help debugging?
A: By logging inputs, outputs, and errors.

72.

Q: Can hooks capture timestamps?
A: Yes, useful for latency monitoring.

73.

Q: Can hooks visualize execution flow?
A: Yes, by exporting traces.

74.

Q: Can hooks detect stuck executions?
A: Yes, by measuring execution time.

75.

Q: Can hooks track tool choice frequency?
A: Yes, for optimization.

76.

Q: Can hooks monitor user satisfaction indirectly?
A: Yes, via patterns in inputs/outputs.

77.

Q: Can hooks create audit logs?
A: Yes, for compliance tracking.

78.

Q: Can hooks detect hallucinations?
A: Indirectly, by validating outputs against constraints.

79.

Q: Can hooks replay a run?
A: Yes, by storing and reusing inputs/outputs.

80.

Q: Can hooks measure token efficiency?
A: Yes, by comparing input vs output token counts.

Real-World Examples

81.

Q: Example: A company wants to log all customer queries. Which hook do they use?
A: On_start agent-level hook.

82.

Q: Example: A researcher wants to debug a single conversation. Which hook do they use?
A: Run-level on_step hook.

83.

Q: Example: An app wants to block harmful outputs. Which hook is best?
A: On_end hook with content filter.

84.

Q: Example: A developer wants retries for failed API calls. Which hook do they use?
A: On_error hook.

85.

Q: Example: A chatbot wants to log time taken for each step. Which hook do they use?
A: On_step hook.

86.

Q: Example: A tool call must never execute unsafe shell commands. Which hook do they use?
A: On_tool_start validation hook.

87.

Q: Example: A medical assistant must log all advice. Which hook do they use?
A: Agent-level on_end hook.

88.

Q: Example: A QA system needs to test different outputs. Which hook do they use?
A: Run-level output modification hook.

89.

Q: Example: An AI tutor must explain errors. Which hook do they use?
A: On_error hook with custom message.

90.

Q: Example: A compliance system must redact credit card numbers. Which hook do they use?
A: On_start input filter hook.

Summary & Edge Cases

91.

Q: Can hooks slow down execution?
A: Yes, if they perform heavy operations.

92.

Q: Should hooks be idempotent?

A: Yes, to avoid duplicate effects.

93.

Q: Can hooks crash the agent?
A: Yes, if not written carefully.

94.

Q: Can hooks be nested?
A: Yes, hooks can call other hooks.

95.

Q: Can hooks log sensitive data?
A: Yes, so careful handling is required.

96.

Q: Can hooks be disabled dynamically?
A: Yes, by removing them from the execution context.

97.

Q: Are hooks framework-dependent?
A: Yes, but the concept is common across frameworks.

98.

Q: Should hooks modify chain-of-thought directly?
A: No, usually avoided to maintain reasoning integrity.

99.

Q: Are hooks part of observability tools?
A: Yes, they’re essential for observability.

100.

Q: In summary, what’s the value of hooks?
A: Hooks provide extensibility, monitoring, debugging, and control without altering core agent logic.