Multiple-Choice Questions on Context Management (OpenAI Agents SDK)

**1. What are the two main types of "context" discussed in the SDK?**

A. User context and system context  
B. Local (code) context and context available to LLMs  
C. Tool context and instruction context  
D. Memory context and tracing context

**Answer:** B — The documentation distinguishes between *local context* available to your code and *LLM context* that the model sees.

**2. What class represents local context passed into tool functions and lifecycle hooks?**

A. LocalContextManager  
B. AgentContext  
C. RunContextWrapper  
D. ContextManager

**Answer:** C — Local context is represented by the RunContextWrapper class.

**3. How do you include your custom context object when running an agent?**

A. Pass it as a tool  
B. Use a global variable  
C. Provide it via the context parameter in Runner.run()  
D. Embed it in instructions

**Answer:** C — You pass your context object using the context parameter in Runner.run(...).

**4. True or False: Each agent run can use different types of context within the same run.**

A. True  
B. False

**Answer:** B — Every tool, lifecycle hook, etc., in a given agent run must use the same context type.

**5. Which of the following is not a typical use-case for the local context object?**

A. Storing user-specific data like username or UID  
B. Providing helper functions  
C. Keeping dependencies like loggers or data fetchers  
D. Directly influencing LLM generation results

**Answer:** D — Local context is not visible to the LLM and does not directly influence generation.

**6. Does the LLM see the contents of the local context object?**

A. Yes, always  
B. Only if context is added to instructions  
C. No, the context is strictly local to your code  
D. Only if explicitly serialized

**Answer:** C — The context object is not sent to the LLM; it's purely local.

**7. How can you make information in local context available to the LLM?**

A. Add it to the instructions (system prompt)  
B. Include it in the agent's input  
C. Expose it via function tools  
D. All of the above

**Answer:** D — You can use any of those strategies to surface context to the LLM.

**8. Adding dynamic content to the system prompt is referred to as:**

A. Instructions injection  
B. Context scripting  
C. LLM enrichment  
D. System augmentation

**Answer:** A — In the docs it's described as adding information to the Agent instructions, effectively augmenting the system prompt.

**9. When passing context via instructions, what capability does the instruction support?**

A. Only static strings  
B. Only functions returning strings  
C. Both static strings and functions that generate strings dynamically based on context  
D. Only JSON objects

**Answer:** C — Instructions can be static strings or dynamic functions receiving the context and returning a string.

**10. True or False: You can expose local context directly to the LLM without using tools or instructions.**

A. True  
B. False

**Answer:** B — You cannot expose context directly; it must be surfaced via messages or tools.

**11. What is a common pattern for defining the local context object?**

A. A JSON string  
B. A Pydantic or dataclass instance  
C. A Python dict  
D. A tuple

**Answer:** B — A common pattern is using a dataclass or Pydantic model.

**12. Given the following code, what does wrapper.context.name refer to?**

@dataclass

class UserInfo:

name: str

user\_info = UserInfo(name="John", uid=123)

A. The agent's name  
B. The tool name  
C. The user's name from context  
D. The LLM model name

**Answer:** C — wrapper.context.name refers to the user's name stored in the context.

**13. What happens if you pass a context object to one tool that differs in type from another within the same agent run?**

A. Silent override of context  
B. A type-checker error during runtime  
C. SDK will raise an error due to generic type enforcement  
D. It works as long as fields match

**Answer:** C — Using type generics like Agent[UserInfo] ensures mismatched context types cause errors.

**14. Which of the following is not a recommended way to surface local context to the LLM?**

A. System instructions  
B. Agent input  
C. Function tools  
D. Changing the local context object directly

**Answer:** D — Simply changing the local context object doesn’t make it visible to the LLM.

**15. What advantage does using function tools offer in relation to context?**

A. They ignore context  
B. They automatically surface the entire context to the LLM  
C. They allow on-demand access to context data when called by the LLM  
D. They make context persistence unnecessary

**Answer:** C — Function tools allow the LLM to fetch context on-demand.

**16. If you want the LLM to use the current date or user’s name, which method can you use?**

A. Local context only  
B. Hand it via the system prompt or input  
C. RunContextWrapper  
D. Wrapped context

**Answer:** B — Make it available through instructions or input so the LLM sees it.

**17. Where in the code example is the context object passed into the agent run?**

A. When creating the Agent object  
B. Inside the tool function  
C. In Runner.run(..., context=user\_info)  
D. As part of instructions

**Answer:** C — The context is passed in using the context parameter of Runner.run.

**18. What is displayed by calling print(result.final\_output) in the example?**

A. The local context object  
B. The LLM-generated response  
C. The tool’s return value using context data  
D. An error message

**Answer:** C — It prints the tool's response that uses context data (“The user John is 47 years old.”).

**19. True or False: You can mix local context types (like one tool expects UserInfo, another expects OtherInfo) in the same run.**

A. True  
B. False

**Answer:** B — All layers (tools, lifecycle hooks, etc.) must use the same context type.

**20. Why is it beneficial to use local context objects instead of global variables?**

A. It ensures type safety and keeps dependencies explicit  
B. Faster execution  
C. LLM uses local context for generation  
D. Enables automatic caching

**Answer:** A — Local context passed via RunContextWrapper provides type safety and explicit dependency management.