OpenAI Agents SDK

Abdul Rehman (PIAIC & GIAIC)

MCQS

OpenAI Agents SDK - Introductory Section

1. What kind of AI apps does the OpenAI Agents SDK enable you to build?

- ✗a) Data analysis apps
- ✓**b) Agentic AI apps**
- ✗c) Static website generators
- ✗d) Mobile gaming apps

---

2. The Agents SDK is described as a production-ready upgrade of what previous experimentation for agents?

- ✗a) Hive
- ✗b) Colony
- ✓**c) Swarm**
- ✗d) Nexus

---

3. Which of the following is NOT listed as a primitive of the Agents SDK?

- ✗a) Agents
- ✗b) Handoffs
- ✓**c) Guardrails** *(Trick question! Guardrails **are** a primitive)*
- ✗d) Workflows

(Note: All are actual primitives, so this is a questionable phrasing — likely intended to test awareness.)

---

4. What are "Agents" defined as in the SDK?

- ✗a) Simple Python functions
- ✓**b) LLMs equipped with instructions and tools**
- ✗c) Data validation modules
- ✗d) User interface components

---

5. What do "Handoffs" allow agents to do?

- ✗a) Log their activities
- ✗b) Self-correct errors

- ✅**c) Delegate to other agents for specific tasks**
- ❌d) Interact with external APIs

---

6. "Guardrails" enable what functionality in relation to agent inputs?

- ❌a) Sending results to the LLM
- ❌b) Calling external tools
- ✅**c) Validating the inputs to agents**
- ❌d) Orchestrating multiple agents

---

7. In combination with what programming language are the SDK's primitives powerful enough to express complex relationships?

- ❌a) Java
- ❌b) JavaScript
- ✅**c) Python**
- ❌d) C++

---

8. What is a key benefit of the SDK having "very few abstractions"?

- ❌a) It makes the SDK more complex.
- ✅**b) It makes it quick to learn.**
- ❌c) It limits customization options.
- ❌d) It requires more boilerplate code.

---

9. Which design principle states that the SDK "Works great out of the box, but you can customize exactly what happens"?

- ❌a) Python-first
- ❌b) Few primitives
- ❌c) Agent loop
- ✅**d) Works great out of the box, but you can customize exactly what happens**

---

10. What does the "Agent loop" feature primarily handle?

- ❌a) Defining agent instructions
- ❌b) Generating automatic responses
- ✅**c) Calling tools, sending results to the LLM, and looping until the LLM is done**
- ❌d) Visualizing agent traces

---

11. The "Python-first" approach means using what for orchestration and chaining agents?

- ❌a) A new, proprietary domain-specific language
- ❌b) A visual programming interface

- ✅**c) Built-in language features**
- ❌d) External YAML configuration files

---

12. What specific capability do "Handoffs" provide for multiple agents?

- ❌a) Error handling
- ❌b) Parallel execution of single tasks
- ✅**c) Coordination and delegation**
- ❌d) Automatic fine-tuning

---

13. How do "Guardrails" typically behave if checks fail?

- ❌a) They log a warning and continue
- ❌b) They prompt the user for a new input
- ✅**c) They break early.**
- ❌d) They attempt to fix the input automatically

---

14. What do "Function tools" allow you to turn any Python function into?

- ❌a) A data validation model
- ❌b) A new agent
- ✅**c) A tool**
- ❌d) A guardrail

---

15. "Function tools" come with automatic schema generation and validation powered by what library?

- ❌a) NumPy
- ❌b) Pandas
- ✅**c) Pydantic**
- ❌d) SciPy

---

16. What is the benefit of the built-in "Tracing" feature?

- ❌a) It automates code deployment
- ❌b) It encrypts sensitive data
- ✅**c) It lets you visualize, debug and monitor workflows, and use OpenAI tools for evaluation, fine-tuning, and distillation.**
- ❌d) It generates user documentation automatically

---

17. What is the command to install the OpenAI Agents SDK?

- ❌a) npm install openai-agents
- ✅**b) pip install openai-agents**
- ❌c) conda install openai-agents
- ❌d) go get openai-agents

---

18. What is the name given to the Agent in the "Hello world" example?

- ✗a) "Haiku Generator"
- ✓**b) "Assistant"**
- ✗c) "Recursion Expert"
- ✗d) "Programming Helper"

---

19. What instruction is given to the Agent in the "Hello world" example?

- ✗a) "Write a poem about nature."
- ✗b) "Solve a math problem."
- ✓**c) "You are a helpful assistant"**
- ✗d) "Explain AI concepts."

---

20. What specific task is the Agent asked to perform in the "Hello world" example?

- ✗a) Generate a story about a robot
- ✗b) Explain quantum physics
- ✓**c) Write a haiku about recursion in programming.**
- ✗d) Translate a sentence into French

---

21. What is the expected final output of the "Hello world" example (the haiku)?

- ✗a) "AI, a new dawn, / Machines learn, systems grow, / Future now unfolds."
- ✓**b) "Code within the code, / Functions calling themselves, / Infinite loop's dance."**
- ✗c) "Program's gentle flow, / Logic weaves, commands obey, / Digital new form."
- ✗d) "Recursion deep, / Calls itself, repeats the task, / Stack grows, then unwinds."

---

22. What environment variable needs to be set to run the "Hello world" example successfully?

- ✗a) API_KEY
- ✗b) OPENAI_API_ENDPOINT
- ✓**c) OPENAI_API_KEY**
- ✗d) AGENT_SDK_KEY

---

23. The SDK is described as enabling you to build agentic AI apps in a package that is:

- ✗a) Complex and heavy
- ✓**b) Lightweight and easy-to-use**
- ✗c) Experimental and feature-limited
- ✗d) Dependent on many external abstractions

---

24. What previous project is the Agents SDK considered a "production-ready upgrade" of?

- ✗a) Core Agents
- ✗b) AgentLab
- ✓**c) Swarm**
- ✗d) AI Forge

---

25. The primitives of the Agents SDK (Agents, Handoffs, Guardrails) in combination with Python are powerful enough to express what?

- ✗a) Only simple linear processes
- ✗b) Basic data transformations
- ✓**c) Complex relationships between tools and agents**
- ✗d) Standalone data visualization

Quickstart

OpenAI Agents SDK - Quickstart Specifics MCQs

1. What is the recommended first step for setting up a new project to use the Agents SDK?

- ✗a) Install the SDK directly.
- ✗b) Set the OpenAI API key.
- ✓**c) Create a project directory and a Python virtual environment.**
- ✗d) Define your first agent immediately.

---

2. After creating the project directory and virtual environment, what is the next action advised before installing the SDK?

- ✗a) Create an OpenAI API key.
- ✓**b) Activate the virtual environment.**
- ✗c) Define the main function.
- ✗d) Run a test command.

---

3. The Agent constructor allows for optional configuration. Which specific optional config parameter is mentioned when creating your first agent?

- ✗a) api_version
- ✗b) debug_mode
- ✓**c) model_config**
- ✗d) trace_level

---

4. When adding "a few more agents" like History Tutor and Math Tutor, what specific parameter is added to provide context for handoff routing?

- ✗a) routing_instructions
- ✗b) context_description
- ✓**c) handoff_description**
- ✗d) specialty_area

---

5. What is the explicit instruction given to the Triage Agent regarding its role?

- ✗a) "You will answer all user questions directly."
- ✗b) "You will provide help with math or history problems."
- ✓**c) "You determine which agent to use based on the user's homework question."**
- ✗d) "You validate user inputs for safety."

---

6. How does the Triage Agent know which specialist agents (e.g., History Tutor, Math Tutor) it can delegate to?

- ✗a) It infers them from their instructions.
- ✓**b) They are provided in a list to the `handoffs` parameter of the Triage Agent.**
- ✗c) They are automatically discovered by the SDK.
- ✗d) It must manually import them during runtime.

---

7. In the "Run the agent orchestration" example, what specific method of the `Runner` class is used to initiate the agent workflow asynchronously?

- ✗a) Runner.run_sync()
- ✗b) Runner.execute()
- ✓**c) Runner.run()**
- ✗d) Runner.start()

---

8. The initial test query for the agent orchestration is "What is the capital of France?". Which agent is this query first sent to?

- ✗a) history_tutor_agent
- ✗b) math_tutor_agent
- ✓**c) triage_agent**
- ✗d) guardrail_agent

---

9. When defining a custom guardrail, what is the purpose of the `HomeworkOutput` class inheriting from `pydantic.BaseModel`?

- ✗a) To store the guardrail's internal state.
- ✗b) To define the agent's instructions.
- ✓**c) To provide a structured output type for the `guardrail_agent`'s check result.**

- ✗d) To activate the virtual environment.

---

10. The `guardrail_agent` is specifically instructed to check for what?

- ✗a) Malicious content in the input.
- ✗b) Grammatical errors in the user's question.
- ✓**c) If the user is asking about homework.**
- ✗d) The length of the user's input.

---

11. What is the role of `ctx.context` when running the `guardrail_agent` inside the `homework_guardrail` function?

- ✗a) It's unused in this context.
- ✓**b) It passes the run's context object down to the guardrail's agent sub-run.**
- ✗c) It specifies the model configuration for the guardrail.
- ✗d) It stores the guardrail's output.

---

12. The `homework_guardrail` function returns `GuardrailFunctionOutput`. What value determines if the "tripwire" is triggered?

- ✗a) final_output.is_homework
- ✗b) final_output.reasoning
- ✓**c) not final_output.is_homework**
- ✗d) result.final_output

---

13. How is the `homework_guardrail` attached to the `triage_agent` in the full workflow example?

- ✗a) As an output_guardrail.
- ✗b) Within the handoffs list.
- ✓**c) As an `InputGuardrail` in the `input_guardrails` list.**
- ✗d) As a function_tool.

---

14. What standard Python module is used to run the `async def main()` function in the full "Put it all together" example?

- ✗a) threading
- ✗b) concurrent
- ✓**c) asyncio**
- ✗d) multiprocessing

---

15. In the combined example, when the `triage_agent` is given "what is life" as an input, which part of the system is expected to trigger?

- ✗a) The History Tutor agent.
- ✗b) The Math Tutor agent.

- ✓**c) The `homework_guardrail`.**
- ✗d) The handoff mechanism.

---

16. To review what happened during an agent run, where are users instructed to navigate?

- ✗a) The project's local log files.
- ✗b) The `Runner` object's internal state.
- ✓**c) The Trace viewer in the OpenAI Dashboard.**
- ✗d) The Python interpreter's history.

---

17. The "Next steps" suggest learning how to build "more complex agentic flows." This implies the quickstart example primarily showcases what?

- ✗a) Only the most advanced features.
- ✗b) Only theoretical concepts.
- ✓**c) Foundational elements and a basic, yet functional, agent orchestration.**
- ✗d) Production-ready deployment strategies.

---

18. What is the explicit purpose of `handoff_descriptions` for the History Tutor and Math Tutor agents?

- ✗a) To provide alternative names.
- ✓**b) To help the Triage Agent determine routing based on context.**
- ✗c) To summarize their instructions.
- ✗d) To indicate if they are ready for production.

---

19. The SDK is described as a "lightweight, easy-to-use package with very few abstractions." How does the Quickstart example demonstrate this?

- ✗a) By showing complex inheritance hierarchies.
- ✓**b) By defining agents and their interactions with clear, concise Python code.**
- ✗c) By requiring extensive configuration files.
- ✗d) By relying on a graphical user interface.

---

20. The `is_homework` field in `HomeworkOutput` is a boolean. What is the type of the `reasoning` field?

- ✗a) list[str]
- ✗b) int
- ✓**c) str**
- ✗d) bool

Examples

OpenAI Agents SDK - Examples Section MCQs

1. The examples in the SDK repository are primarily intended to demonstrate what?

- ✗a) Optimal performance benchmarks.
- ✗b) The SDK's internal architecture.
- ✓**c) Different agent design patterns and capabilities.**
- ✗d) Compatibility with all Python versions.

---

2. If a developer wants to understand how to ensure an agent workflow follows a predictable sequence of steps, which example category would be most helpful?

- ✗a) basic
- ✓**b) agent_patterns**
- ✗c) handoffs
- ✗d) tool examples

---

3. The `agent_patterns` category mentions "Agents as tools." What does this imply about the design flexibility of the SDK?

- ✗a) Agents can only be used as primary responders.
- ✓**b) Agents themselves can be modular components utilized by other agents or systems.**
- ✗c) Tools must always invoke an agent.
- ✗d) The SDK only supports simple agent-tool interactions.

---

4. Which category would be essential for a developer trying to implement real-time interaction feedback or partial responses from the LLM?

- ✗a) handoffs
- ✗b) tool examples
- ✓**c) basic (due to "Streaming outputs")**
- ✗d) model providers

---

5. What distinguishes the "Dynamic system prompts" examples from typical agent instructions?

- ✗a) They are always very short.
- ✗b) They are defined as separate files.
- ✓**c) They allow the agent's guiding instructions to change based on runtime conditions or context.**
- ✗d) They are specific to non-OpenAI models.

---

6. For a user looking to integrate a feature like web search into their agent, which example category would guide them on how to do this using OAI-hosted functionalities?

- ✗a) agent_patterns
- ✓**b) tool examples**
- ✗c) basic
- ✗d) research_bot

---

7. The `model_providers` category is crucial for developers who are concerned with what aspect of their agent application?

- ✗a) Only using OpenAI's latest models.
- ✓**b) Utilizing LLMs from different vendors or custom sources.**
- ✗c) Optimizing model inference speed.
- ✗d) Ensuring model interpretability.

---

8. The `customer_service` example illustrates a system for an airline. What is the broader implication of such "built-out examples"?

- ✗a) They are purely theoretical demonstrations.
- ✗b) They only show basic agent setup.
- ✓**c) They showcase how to construct more complete, real-world, industry-specific applications.**
- ✗d) They primarily focus on debugging agent traces.

---

9. If a user wants to build an agent that can perform extensive information gathering similar to a specialized research assistant, which example would serve as a direct reference?

- ✗a) customer_service
- ✗b) handoffs
- ✓**c) research_bot**
- ✗d) agent_patterns

---

10. What specific type of AI interaction is highlighted by the `voice` examples category?

- ✗a) Visual recognition.
- ✗b) Text summarization.
- ✓**c) Conversational agents that interact via spoken language.**
- ✗d) Code generation.

---

11. The mention of TTS and STT models in the voice examples indicates the SDK's capability to bridge agents with what kind of interfaces?

- ✗a) Database interfaces.
- ✗b) Command-line interfaces.
- ✓**c) Audio-based interfaces.**
- ✗d) Graphical user interfaces.

---

12. What does the inclusion of `Parallel agent execution` in `agent_patterns` suggest about the SDK's runtime capabilities?

- ✗a) Agents must always run sequentially.
- ✗b) It supports basic multi-threading only.
- ✓**c) It can orchestrate multiple agents to work concurrently on tasks.**
- ✗d) It is limited to single-agent operations.

---

13. The organization of examples into categories like `handoffs`, `agent_patterns`, and `basic` is designed to help users primarily with what?

- ✗a) Identifying bugs in the SDK.
- ✓**b) Learning specific architectural approaches and foundational features.**
- ✗c) Benchmarking different LLM providers.
- ✗d) Understanding the SDK's installation process.

---

14. What could a user learn from the `Lifecycle events` examples within the `basic` category?

- ✗a) How to define an agent's name.
- ✗b) How to force an agent to use a tool.
- ✓**c) How to observe and react to different stages of an agent's operation.**
- ✗d) How to set up a virtual environment.

---

15. The phrase "Simple deep research clone" for `research_bot` implies it's designed to mimic what kind of functionality?

- ✗a) A search engine's indexing.
- ✗b) A human's ability to thoroughly investigate a topic.
- ✓**c) An automated system capable of in-depth information gathering and synthesis.**
- ✗d) A tool for quick factual lookups.

Documentation

Agents

OpenAI Agents SDK - Agents Section MCQs (Nuanced)

1. Beyond temperature and top_p, what broader category of parameters can be configured via `model_settings` for an LLM?

- ✗a) Only specific API keys.
- ✓**b) Model tuning parameters.**
- ✗c) Networking configurations.
- ✗d) Agent name and instructions.

---

2. The `get_weather` tool is decorated with `@function_tool`. What does this decorator implicitly handle for the Python function?

- ✗a) Automatic caching of results.
- ✓**b) Automatic schema generation for the LLM to understand and use the tool.**
- ✗c) Dynamic scaling of the function.
- ✗d) Error handling within the function's logic.

---

3. The Context mechanism is described as "dependency-injection." What does this term imply?

- ✗a) They must explicitly import all external modules.
- ✓**b) Dependencies are provided to them externally, rather than being created internally.**

- ✗c) They can only access global variables.
- ✗d) They can only inject other agents as dependencies.

---

4. In the `UserContext` example, what is `async def fetch_purchases()` designed to represent?

- ✗a) A static utility function.
- ✓**b) Asynchronous operations that can fetch data relevant to the user during an agent run.**
- ✗c) A method for storing user credentials.
- ✗d) A function for generating new users.

---

5. What does Pydantic's TypeAdapter flexibility suggest about output_type in tools/agents?

- ✗a) Only BaseModel subclasses are supported.
- ✗b) Only simple Python primitives are allowed.
- ✓**c) It allows for a broad range of Python types (dataclasses, lists, TypedDict, etc.).**
- ✗d) It implies a performance overhead.

---

6. What is the main benefit of combining `instruction: Extract calendar events from text` with `output_type=CalendarEvent`?

- ✗a) It prevents the agent from making any mistakes.
- ✗b) It allows the agent to generate plain text and structured data simultaneously.
- ✓**c) It guides the LLM to format identified events into structured, machine-readable format.**
- ✗d) It enables the agent to directly add events to a calendar.

---

7. What problem do modular handoffs between agents aim to solve?

- ✗a) Overfitting of a single large LLM.
- ✓**b) The challenge of one LLM needing to handle too many diverse tasks.**
- ✗c) Network latency between agents.
- ✗d) The cost of running multiple small LLMs.

---

8. Why is `context` passed to the `dynamic_instructions` function?

- ✗a) To allow the agent to modify its own name.
- ✓**b) To enable instructions to be personalized or conditional based on runtime data.**
- ✗c) To force the agent to use a specific tool.
- ✗d) To manage the agent's internal memory.

---

9. Why would a developer use "lifecycle events" (hooks)?

- ✗a) To dynamically change the agent's model.
- ✓**b) To perform actions like logging, monitoring, or data pre-fetching at specific stages.**
- ✗c) To bypass guardrail checks.

- ✗d) To automatically regenerate instructions.

---

10. What is the advantage of guardrails running "in parallel to the agent"?

- ✗a) It ensures the agent runs faster.
- ✗b) It allows for real-time model fine-tuning.
- ✓**c) It enables early breaking if input validation fails, saving resources.**
- ✗d) It simplifies the agent's instruction set.

---

11. What is the use of the `clone()` method on agents?

- ✗a) Running agents synchronously.
- ✓**b) Creating specialized versions of a base agent with minor changes.**
- ✗c) Forcing tool use behavior.
- ✗d) Defining dynamic instructions.

---

12. What does the `tool_choice` setting mean for the LLM's control?

- ✗a) The content of the tool's output.
- ✓**b) The selection of the most appropriate tool from the available list.**
- ✗c) Whether to run indefinitely.
- ✗d) Its own internal reasoning process.

---

13. Why does `tool_choice` reset to "auto" after each tool call?

- ✗a) Slow API responses.
- ✓**b) To avoid infinite loops where the LLM keeps calling tools unnecessarily.**
- ✗c) Incorrect tool output formats.
- ✗d) Overuse of `ModelSettings`.

---

14. What happens when `tool_use_behavior="stop_on_first_tool"` is set?

- ✗a) It continues to process with tool_choice="auto".
- ✓**b) It stops LLM processing after using the first tool and uses that result as final output.**
- ✗c) It re-runs the previous turn.
- ✗d) It triggers an input guardrail.

---

15. What does the trio of `instructions`, `model_settings`, and `tools` provide the LLM?

- ✗a) Only raw text input.
- ✗b) A fixed, unchangeable persona.
- ✓**c) A defined persona, configuration, and tool access for rich interaction.**
- ✗d) Solely conversational abilities.

Running Agents

OpenAI Agents SDK - Running Agents Section MCQs

✅Part 1: Running Methods & The Agent Loop

✅Part 1: Running Methods & The Agent Loop

1. **Which Runner method is asynchronous and returns a RunResult?**
   ❌a) Runner.run_sync()
   ❌b) Runner.run_streamed()
   ✅**c) Runner.run()**
   ❌d) Runner.execute()

2. **Runner.run_sync() internally uses:**
   ❌a) Runner.stream_events()
   ✅**b) Runner.run()**
   ❌c) Runner.execute_sync()
   ❌d) Runner.async_run()

3. **What does Runner.run_streamed() return?**
   ❌a) RunResult
   ✅**b) RunResultStreaming**
   ❌c) RunStream
   ❌d) StreamResult

4. **Initial input types to Runner methods?**
   ❌a) An integer or a boolean
   ❌b) A string or a dictionary
   ✅**c) A string (user message) or a list of input items**
   ❌d) A tuple or a set

5. **If the LLM produces final_output:**
   ❌a) The LLM does a handoff
   ❌b) The LLM produces tool calls
   ✅**c) The loop ends and the result is returned**
   ❌d) A MaxTurnsExceeded exception is raised

6. **If the LLM does a handoff:**
   ✗a) The loop ends immediately
   ✓**b) The current agent and input are updated, and the loop re-runs**
   ✗c) Tool calls are executed
   ✗d) The process waits for user intervention

7. **If the LLM produces tool calls:**
   ✗a) The loop ends with an error
   ✗b) The final_output is immediately returned
   ✓**c) Those tool calls are run, their results are appended, and the loop re-runs**
   ✗d) A handoff is automatically triggered

8. **If max_turns is exceeded:**
   ✗a) The agent simply stops without an error
   ✓**b) A MaxTurnsExceeded exception is raised**
   ✗c) The agent automatically restarts
   ✗d) The final_output is forced

9. **For output to be "final":**
   ✗a) It must have done at least one handoff
   ✓**b) There must be no tool calls**
   ✗c) It must exceed max_turns
   ✗d) It must be streamed

✓Part 2: Streaming & Run Config

10. **What does streaming provide?**
    ✗a) Debug logs only.
    ✓**b) Streaming events**
    ✗c) Performance metrics.
    ✗d) Model weights.

11. **Purpose of run_config?**
    ✗a) To configure the Agent class directly.
    ✗b) To define agent instructions.
    ✓**c) To configure some global settings for the entire agent run.**
    ✗d) To manage the virtual environment.

12. **Override model via run_config?**
    ✗a) default_model
    ✗b) global_llm
    ✓**c) model**
    ✗d) override_model

13. **Default model_provider?**
    ✗a) Google
    ✗b) Anthropic
    ✓**c) OpenAI**
    ✗d) Custom

14. **run_config.model_settings can be used to:**
    ✓**a) Override agent-specific settings like temperature or top_p globally.**
    ✗b) Define new tools for the agent.
    ✗c) Specify handoff behavior.
    ✗d) Disable tracing.

15. **run_config allows what guardrails?**
    ✗a) Only output guardrails.
    ✗b) Only input guardrails.
    ✓**c) Both input and output guardrails.**
    ✗d) Only global guardrails.

16. **handoff_input_filter allows you to:**
    ✗a) Disable specific handoffs.
    ✓**b) Edit the inputs that are sent to the new agent.**
    ✗c) Prioritize certain handoffs.
    ✗d) Redirect handoffs to external APIs.

17. **Disable tracing via:**
    ✗a) disable_trace
    ✗b) tracing_off
    ✓**c) tracing_disabled**
    ✗d) no_tracing

18. **trace_include_sensitive_data enables:**
    ✗a) Whether traces are enabled or disabled.
    ✓**b) Whether traces will include LLM and tool call inputs/outputs.**
    ✗c) The location where trace data is stored.
    ✗d) The format of the trace metadata.

19. **Recommended tracing parameter:**
    ✗a) trace_id
    ✗b) group_id
    ✓**c) workflow_name**
    ✗d) trace_metadata

✓Part 3: Conversations & Exceptions

20. **A Runner call = what in chat?**
    ✗a) A complete conversation.
    ✗b) A single independent query.
    ✓**c) A single logical turn**
    ✗d) A debugging session.

21. **Get next turn input via:**
    ✗a) to_output_list()
    ✗b) get_next_input()
    ✓**c) to_input_list()**
    ✗d) prepare_for_next_turn()

22. **What links traces across multiple runs?**
    ✗a) workflow_name
    ✗b) trace_id
    ✓**c) group_id (via thread_id)**
    ✗d) trace_metadata

23. **What is AgentsException?**
    ✗a) An exception specific to tool calls.
    ✓**b) The base class for all exceptions raised in the SDK.**
    ✗c) An exception related to model behavior.
    ✗d) An exception for user-made errors.

24. **MaxTurnsExceeded is raised when:**
    ✗a) When a guardrail is tripped.
    ✗b) When the model produces invalid outputs.
    ✓**c) When the run exceeds the max_turns passed to the run methods.**
    ✗d) When the user makes an error in the SDK.

25. **ModelBehaviorError is raised for:**
    ✗a) Exceeding max_turns.
    ✓**b) When the model produces invalid outputs (e.g., malformed JSON)**
    ✗c) User input validation failure.
    ✗d) Disabling tracing.

26. **UserError is raised when:**
    ✗a) Internal SDK bugs.
    ✗b) When the LLM generates an incorrect answer.

✓**c) When the person writing code using the SDK makes an error.**
✗d) Network connectivity issues.

27. **Guardrail exceptions:**
    ✗a) GuardrailError and TripwireError
    ✗b) InputGuardrailFailed and OutputGuardrailFailed
    ✓**c) InputGuardrailTripwireTriggered and OutputGuardrailTripwireTriggered**
    ✗d) ValidationError and CheckFailed

28. **Second turn input is created by:**
    ✗a) Only by the user's new question.
    ✗b) By the previous result.final_output.
    ✓**c) By combining result.to_input_list() with the new user message.**
    ✗d) By resetting the agent state.

29. **Initial Assistant instruction in example:**
    ✗a) "Write a haiku about recursion."
    ✓**b) "Reply very concisely."**
    ✗c) "What city is the Golden Gate Bridge in?"
    ✗d) "Explain your reasoning at each step."

30. **A Runner run represents what in a chat?**
    ✗a) A complete conversation history.
    ✗b) A single independent query.
    ✓**c) A single logical turn.**
    ✗d) A new conversation thread.

Results

OpenAI Agents SDK - Results Section MCQs

✓Part: Results & Run Items

1. **What type of object is returned when you call `Runner.run()` or `Runner.run_sync()`?**
   a) RunResultStreaming
   ✓**b) RunResult**
   c) RunResultBase
   d) RunnerResult

---

2. **If you call `Runner.run_streamed()`, what type of object do you receive?**
   ✓**a) RunResultStreaming**
   b) RunResult

c) RunResultBase
d) StreamingResult

---

3. **Both `RunResult` and `RunResultStreaming` inherit from which base class where most useful information is present?**
a) RunBase
✓**b) RunResultBase**
c) ResultBase
d) RunnerBase

---

4. **What does the `final_output` property contain?**
a) All intermediate steps of the agent run.
✓**b) The final output of the last agent that ran.**
c) The original input provided to the runner.
d) A list of all tool calls made.

---

5. **If the last agent that ran did not have an `output_type` defined, what will be the type of `final_output`?**
a) An object of last_agent.output_type
b) A list
✓**c) A str**
d) None

---

6. **Why is the `final_output` property of type `Any` in the SDK?**
a) To allow for easier debugging.
b) Because its type is statically known only when no handoffs occur.
✓**c) Due to handoffs, as any Agent might be the last agent, so the output type is not statically known.**
d) It is always a generic object, regardless of output type.

---

7. **Which method allows you to turn the result into an input list suitable for concatenating with original input for subsequent runs?**
a) result.get_input_list()
b) result.get_next_input()
✓**c) result.to_input_list()**
d) result.input_history()

---

8. **The `to_input_list()` method is convenient for what purpose?**
a) To reset the agent's state.
b) To explicitly define new handoffs.
✓**c) To take outputs of one agent run and pass them into another run or loop with new user inputs.**
d) To disable tracing for the next run.

---

9. **What does the `last_agent` property contain?**
   a) The initial agent that started the run.
   ✓**b) The last agent that ran.**
   c) A list of all agents involved in the run.
   d) The Triage Agent by default.

---

10. **When might storing and re-using the `last_agent` be useful?**
   a) When you want to force a specific tool use.
   ✓**b) If you have a frontline triage agent that hands off to a specialized agent, to re-use the specialized agent for follow-up.**
   c) To disable guardrails for the next run.
   d) To always reset the conversation.

---

11. **What does the `new_items` property contain?**
   a) Only the final response.
   ✓**b) The new items (RunItems) generated during the run.**
   c) A list of all model responses.
   d) The original input and final output combined.

---

12. **A `MessageOutputItem` indicates what from the LLM?**
   a) That a tool was invoked.
   b) That a handoff occurred.
   ✓**c) A message from the LLM.**
   d) A reasoning item.

---

13. **What does a `HandoffCallItem` indicate?**
   ✓**a) That the LLM called the handoff tool.**
   b) That a tool call output was received.
   c) That a guardrail was tripped.
   d) That the final output was generated.

---

14. **For a `HandoffOutputItem`, in addition to the raw item (tool response), what else can you access from it?**
   a) The LLM's full conversational history.
   b) The model_config of the handoff agent.
   ✓**c) The source and target agents.**
   d) The trace_id of the handoff.

---

15. **Which `RunItem` indicates that the LLM invoked a tool?**
   a) ToolResponseItem
   b) ToolResultItem
   ✓**c) ToolCallItem**
   d) ToolOutputItem

---

16. **What information can you access from a `ToolCallOutputItem`?**
 a) Only the raw tool call.
 b) Only the LLM's reasoning.
 ✓**c) The tool output.**
 d) The model_settings of the tool.

---

17. **What does a `ReasoningItem` indicate?**
 a) A message from a tool.
 b) A call to a handoff.
 ✓**c) A reasoning item from the LLM.**
 d) A final output from the agent.

---

18. **Which properties contain the results of the guardrails, if any?**
 a) guardrail_status
 ✓**b) input_guardrail_results and output_guardrail_results**
 c) check_results
 d) validation_outcomes

---

19. **The `raw_responses` property contains what?**
 a) Only user messages.
 ✓**b) The ModelResponses generated by the LLM.**
 c) The final summarized output.
 d) Intermediate tool call arguments.

---

20. **What does the `input` property of a `RunResultBase` object contain?**
 a) The processed input after guardrails.
 b) The input generated by the LLM.
 ✓**c) The original input you provided to the run method.**
 d) The inputs for the next turn.

Streaming

OpenAI Agents SDK - Streaming Section MCQs

Streaming & Stream Events – OpenAI Agents SDK

1. **What is the primary benefit of using streaming in agent runs?**
 a) To reduce the overall execution time of the agent.
 b) To enable offline processing of results.

✅**c) To show the end-user progress updates and partial responses as the run proceeds.**
d) To store all run data in a database.

---

2. **Which Runner method must you call to enable streaming for an agent run?**
   a) Runner.run()
   b) Runner.run_sync()
   ✅**c) Runner.run_streamed()**
   d) Runner.stream_events()

---

3. **When you call `Runner.run_streamed()`, what type of object is returned?**
   a) RunResult
   ✅**b) RunResultStreaming**
   c) StreamResult
   d) StreamingEvents

---

4. **How do you access the asynchronous stream of StreamEvent objects from a RunResultStreaming object?**
   a) By calling result.get_stream().
   b) By iterating directly over result.
   ✅**c) By calling result.stream_events().**
   d) By subscribing to a global event bus.

---

5. **What kind of events are RawResponsesStreamEvent?**
   a) High-level events indicating tool calls.
   ✅**b) Raw events passed directly from the LLM, in OpenAI Responses API format.**
   c) Events indicating agent handoffs.
   d) Debugging events for internal SDK processes.

---

6. **RawResponsesStreamEvent objects contain data with a type and data. Which specific data type is used for streaming LLM text token-by-token?**
   a) ResponseOutputEvent
   b) LLMTextEvent
   ✅**c) ResponseTextDeltaEvent**
   d) TokenDeltaEvent

---

7. **What is the primary use case for RawResponsesStreamEvent?**
   a) To identify when a tool has completed its execution.
   b) To determine which agent is currently active.
   ✅**c) To stream response messages to the user as soon as they are generated (e.g., token-by-token).**
   d) To capture the final output of the entire agent run.

---

8. **RunItemStreamEvents are described as "higher level events." What specific kind of updates do they provide?**

a) Token-by-token generation.
✓**b) Updates at the level of "message generated" or "tool ran".**
c) Changes in the LLM model used.
d) Debugging information about internal loop iterations.

---

9. **What information does an AgentUpdatedStreamEvent specifically provide?**
a) The current LLM model's temperature settings.
b) The total number of turns completed by the agent.
✓**c) Updates when the current agent changes (e.g., as a result of a handoff).**
d) The amount of time the agent has been running.

---

10. **In the example for RawResponsesStreamEvent, what is the `end=""` and `flush=True` in the print statement used for?**
a) To append a newline character after each token.
b) To prevent any output from being printed until the end.
✓**c) To print each token without a new line and immediately display it to the console.**
d) To store the tokens in a buffer for later processing.

---

11. **In the RunItemStreamEvents example, what is the instruction given to the Joker agent?**
a) "Tell me exactly 5 jokes."
b) "Reply very concisely."
✓**c) "First call the how_many_jokes tool, then tell that many jokes."**
d) "Generate a haiku about a clown."

---

12. **The how_many_jokes tool in the example returns a random integer. What Python module is imported for this functionality?**
a) math
✓**b) random**
c) numpy
d) statistics

---

13. **In the RunItemStreamEvents example, why is the `if event.type == "raw_response_event": continue` line used?**
a) To explicitly process raw events first.
b) To throw an error if raw events are received.
✓**c) To ignore token-level raw_response_event deltas and focus on higher-level events.**
d) To log raw response events to a file.

---

14. **When `event.type == "agent_updated_stream_event"` is true, what specific information is printed to the console?**
a) The total number of agents.
b) The previous agent's name.
✓**c) The name of the new agent.**
d) The reason for the agent update.

15. **What helper utility is used in the RunItemStreamEvents example to extract the text content from a message_output_item?**
a) event.item.get_text()
b) event.item.message_content
✓**c) ItemHelpers.text_message_output(event.item)**
d) event.item.to_string()

---

16. **If `event.item.type == "tool_call_output_item"`, what information is specifically printed?**
a) Only that a tool was called.
✓**b) The output received from the tool.**
c) The name of the tool called.
d) The arguments passed to the tool.

---

17. **What is the fundamental difference in granularity between RawResponsesStreamEvent and RunItemStreamEvents?**
a) One is for input, the other for output.
b) One is for synchronous runs, the other for asynchronous.
✓**c) RawResponsesStreamEvent provides token-level updates, while RunItemStreamEvents provide updates for fully generated logical items (messages, tool outputs).**
d) One is for debugging, the other for production.

---

18. **What would happen if `asyncio.run(main())` were not used in the examples?**
a) The Runner.run_streamed call would become synchronous.
✓**b) The asynchronous main function would not execute properly, as it needs an event loop.**
c) The print statements would not display output.
d) The agent would automatically switch to synchronous mode.

---

19. **The "Joker" agent's instructions ensure that a tool call (to `how_many_jokes`) occurs before generating jokes. This demonstrates streaming events in a scenario involving what?**
a) Only direct LLM responses.
✓**b) Both LLM responses and tool interactions.**
c) Only handoffs between agents.
d) Purely input validation.

---

20. **After `result.stream_events()` is called and iterated through, what does the RunResultStreaming object still contain?**
a) Only the raw token stream.
b) Only the last StreamEvent.
✓**c) The complete information about the run, including all new outputs produced.**
d) Only the final output of the agent.

REPL Utility

OpenAI Agents SDK - REPL Utility Section MCQs

✓REPL Utility – `run_demo_loop()`

1. **What is the name of the utility provided by the SDK for quick interactive testing?**
   a) Runner.run_interactive()
   ✓**b) run_demo_loop**
   c) Agent.repl()
   d) interactive_shell

---

2. **What is the primary function of `run_demo_loop`?**
   a) To deploy agents to production.
   b) To generate unit tests for agents.
   ✓**c) To prompt for user input in a loop, maintaining conversation history.**
   d) To evaluate agent performance metrics.

---

3. **By default, how does `run_demo_loop` handle model output?**
   a) It stores all output in a file.
   b) It prints the final output only after the loop exits.
   ✓**c) It streams model output as it is produced.**
   d) It sends output to an external logging service.

---

4. **Which of the following commands or actions will *not* allow you to exit the `run_demo_loop`?**
   a) Typing quit.
   b) Typing exit.
   c) Pressing Ctrl-D.
   ✓**d) Pressing Ctrl-C.**

Explanation: `quit`, `exit`, and `Ctrl+D` are explicitly handled by the REPL. Ctrl+C may raise a `KeyboardInterrupt` but is not the intended exit method per SDK docs.

---

5. **In the provided example code, what Agent instruction is set for the "Assistant"?**
   a) "Always respond in haiku form."
   b) "You are a helpful chatbot."
   ✓**c) "You are a helpful assistant."**
   d) "Provide help with math problems."

---

6. **The `main` function in the REPL utility example is defined as `async def main()`. What standard Python module is used to run this asynchronous function?**

a) threading
b) concurrent
✓**c) asyncio**
d) multiprocessing

---

7. **What is the main benefit of `run_demo_loop` "keeping the conversation history between turns"?**
   a) It makes the agent run faster.
   b) It disables external tool calls.
   ✓**c) It allows for continuous, context-aware interactions with the agent.**
   d) It only stores the last user input.

Tools

OpenAI Agents SDK - Tools Section MCQs

Part 1: Overview and Hosted Tools

1. **What is the fundamental role of "Tools" in the Agent SDK?**
   ✓**b) To let agents take actions like fetching data, running code, or calling external APIs.**
   ✗a) To define the agent's instructions.
   ✗c) To manage conversation history.
   ✗d) To configure global run settings.

2. **How many distinct classes of tools are there in the Agent SDK?**
   ✓**c) Three**
   ✗a) One
   ✗b) Two
   ✗d) Four

3. **Which class of tools runs on LLM servers alongside the AI models?**
   ✓**c) Hosted tools**
   ✗a) Function calling tools
   ✗b) Agents as tools
   ✗d) Custom tools

4. **Which of the following is NOT listed as a hosted tool offered by OpenAI?**
   ✓**d) DatabaseQueryTool**
   ✗a) WebSearchTool
   ✗b) FileSearchTool
   ✗c) ComputerTool

5. **The FileSearchTool allows retrieving information from what specific OpenAI service?**
   ✅**c) OpenAI Vector Stores**
   ❌a) OpenAI API Gateway
   ❌b) OpenAI Embeddings Service
   ❌d) OpenAI Image Recognition API

6. **What is the purpose of the CodeInterpreterTool?**
   ✅**c) To let the LLM execute code in a sandboxed environment.**
   ❌a) To translate code
   ❌b) To debug production code
   ❌d) To generate comments

7. **In the Hosted tools example, which parameter specifies the file source?**
   ✅**c) vector_store_ids**
   ❌a) file_paths
   ❌b) document_ids
   ❌d) storage_buckets

8. **The ComputerTool enables agents to perform what kind of tasks?**
   ✅**c) Automating computer use tasks.**
   ❌a) Automating image generation
   ❌b) Spreadsheet automation
   ❌d) Web scraping

Part 2: Function Tools - Definition and Automation

9. **Which decorator turns a Python function into a tool?**
   ✅**b) @function_tool**
   ❌a) @agent_tool
   ❌c) @tool_callable
   ❌d) @make_tool

10. **How is a function tool's name determined by default?**
    ✅**b) It uses the name of the Python function.**
    ❌a) Explicitly provided
    ❌c) Unique ID
    ❌d) From the docstring

11. **Where is the tool description extracted from?**
    ✅**b) From the docstring of the function.**
    ❌a) Function arguments

❌c) Config file
❌d) name_override

12. **Which Python module helps extract function signatures?**
✅**b) inspect**
❌a) sys
❌c) types
❌d) os

13. **Which library parses docstrings for descriptions?**
✅**b) griffe**
❌a) Sphinx
❌c) NumpyDocs
❌d) Pydoc

14. **What library is used for schema creation of inputs?**
✅**c) pydantic**
❌a) jsonschema
❌b) dataclasses
❌d) typing

15. **Which docstring formats are supported by griffe?**
✅**b) google, sphinx, numpy**
❌a) JSON, YAML, TOML
❌c) Markdown, reStructuredText
❌d) PEP 257, Epytext

16. **How to set a function tool's name explicitly?**
✅**b) name_override="my_tool"**
❌a) tool_name="my_tool"
❌c) Change variable name
❌d) Set `tool.name`

17. **How to prevent docstring parsing?**
✅**c) use_docstring_info = False**
❌a) parse_docstring
❌b) enable_docstring
❌d) auto_docstring

18. **In `fetch_weather`, what is `Location`?**
✅**b) typing_extensions.TypedDict**
❌a) Pydantic BaseModel

✗c) dict
✗d) dataclass

Part 3: Custom Function Tools & Agents as Tools

19. **Required parameter for manual FunctionTool creation?**
    ✓**c) on_invoke_tool**
    ✗a) auto_schema
    ✗b) docstring_format
    ✗d) tool_priority

20. **Expected return type of `on_invoke_tool`?**
    ✓**c) str**
    ✗a) Any
    ✗b) dict
    ✗d) None

21. **What method parses JSON into `FunctionArgs`?**
    ✓**c) FunctionArgs.model_validate_json()**
    ✗a) parse_raw()
    ✗b) from_json_string()
    ✗d) load_json()

22. **Why model "agents as tools"?**
    ✓**b) To let a central agent orchestrate other agents without a full handoff.**
    ✗a) Sequential execution
    ✗c) Debugging
    ✗d) Fewer LLM calls

23. **Method to convert Agent into a tool?**
    ✓**c) agent.as_tool()**
    ✗a) agent.to_tool()
    ✗b) agent.make_tool()
    ✗d) agent.create_tool()

24. **What parameters define agent-as-tool presentation?**
    ✓**c) tool_name and tool_description**
    ✗a) input_type, output_type
    ✗b) model_name, model_settings
    ✗d) max_turns, run_config

25. **What is the limitation of `agent.as_tool()`?**
✅**b) It doesn't support full config like `max_turns`.**
❌a) Cannot use FunctionTool
❌c) Only works with run_sync
❌d) Prevents context access

26. **What to do if `agent.as_tool()` config isn't enough?**
✅**c) Use `Runner.run` inside a custom function tool.**
❌a) Avoid agents-as-tools
❌b) Use Handoff
❌d) Override `as_tool`

Part 4: Output Extraction and Error Handling

27. **What is `custom_output_extractor` for?**
✅**c) To reformat or modify the sub-agent's output.**
❌a) Summarize
❌b) Translate
❌d) Filter sensitive input

28. **Why reverse `run_result.new_items`?**
✅**c) To find the latest JSON-like message efficiently.**
❌a) Chronological order
❌b) Oldest first
❌d) Memory efficiency

29. **What RunItem is `custom_output_extractor` looking for?**
✅**b) ToolCallOutputItem**
❌a) MessageOutputItem
❌c) HandoffOutputItem
❌d) ReasoningItem

30. **What happens if `failure_error_function` isn't defined?**
✅**b) Runs `default_tool_error_function` to inform LLM.**
❌a) Silently ignores
❌c) Re-raises
❌d) Retries

31. **What happens if you pass `None` to `failure_error_function`?**
✅**c) Tool call errors are re-raised for manual handling.**
❌a) Silently ignored
❌b) Returns empty
❌d) Notifies LLM

32. **What errors may be re-raised?**
✅**b) ModelBehaviorError or UserError**
❌a) NetworkError
❌c) MemoryError
❌d) AuthError

33. **Where must you handle errors in manual FunctionTool?**
✅**c) Inside `on_invoke_tool`**
❌a) Global handler
❌b) Outside
❌d) Using `failure_error_function`

34. **What does `ctx: RunContextWrapper[Any]` enable?**
✅**c) Access to context-specific data or dependencies**
❌a) Must be async
❌b) Only orchestrator use
❌d) Forces structured output

35. **Role of `TypedDict` & `BaseModel`?**
✅**c) Type hinting + schema generation/validation**
❌a) Execution order
❌b) DB integration
❌d) API endpoint config

Model Context Protocol (MCP)

OpenAI Agents SDK - Model Context Protocol (MCP) Section MCQs

1. **What is the primary function of the Model Context Protocol (MCP)?**
✅**c) To standardize how applications provide context and tools to LLMs.**
❌a) To manage the LLM's internal memory.
❌b) To define a new type of LLM.
❌d) To encrypt communication between agents.

2. **MCP is compared to which technology for its standardization role?**
✅**c) USB-C**
❌a) Wi-Fi
❌b) Bluetooth
❌d) Ethernet

3. **How many kinds of MCP servers are defined based on transport mechanism?**
✅**b) Three**
❌a) Two
❌c) Four
❌d) Five

4. **Which type runs "locally" as a subprocess of your app?**
✅**a) stdio servers**
❌b) HTTP over SSE servers
❌c) Streamable HTTP servers
❌d) Remote servers

5. **Which server class connects using Streamable HTTP transport?**
✅**c) MCPServerStreamableHttp**
❌a) MCPServerStdio
❌b) MCPServerSse
❌d) MCPServerHttp

6. **Which command starts the official MCP filesystem server?**
✅**c) @modelcontextprotocol/server-filesystem**
❌a) @modelcontextprotocol/server-http
❌b) @modelcontextprotocol/server-local
❌d) @modelcontextprotocol/server-console

7. **How are MCP servers integrated in an Agent instance?**
✅**b) Via the mcp_servers parameter.**
❌a) tools
❌c) model_settings
❌d) run_config

8. **What method is called to list available tools for the LLM?**
✅**b) list_tools()**
❌a) get_tools_list()
❌c) fetch_tools()
❌d) register_tools()

9. **What method is called when an MCP tool is invoked?**
✅**b) call_tool()**
❌a) execute_tool()
❌c) run_tool()
❌d) invoke_tool()

10. **Why implement caching for MCP servers?**
✅**b) To mitigate latency hits due to repeated list_tools() calls.**
❌a) Reduce memory use
❌c) Keep tools up-to-date
❌d) Simplify server code

11. **What parameter enables automatic tool list caching?**
✅**b) cache_tools_list**
❌a) enable_cache
❌c) auto_cache_tools
❌d) tool_cache_enabled

12. **When should you use automatic caching?**
✅**c) If you are certain the tool list will not change.**
❌a) For local servers
❌b) If tools change frequently
❌d) When debugging

13. **How to manually invalidate cached tools list?**
✅**c) invalidate_tools_cache()**
❌a) Restart the Agent
❌b) clear_cache()
❌d) set cache_tools_list=False

14. **What kind of server connects via a remote URL using SSE?**
✅**b) HTTP over SSE server**
❌a) stdio server
❌c) Streamable HTTP server
❌d) Localhost server

15. **What are command and args used for in MCPServerStdio?**
✅**c) To define the subprocess command/args that run the stdio server.**
❌a) Network address
❌b) Environment vars
❌d) Logging level

16. **Which MCP operations are captured in SDK tracing?**
✅**b) Calls to list_tools()**
❌a) Tool execution time only
❌c) Token generation
❌d) Input validation

17. **What other operations include MCP tracing info?**
    ✅**b) Function calls**
    ❌a) Agent handoffs
    ❌c) run_streamed events
    ❌d) Agent initialization

18. **Primary benefit of MCP for Agent developers?**
    ✅**c) Access a broad range of tools via standardized servers.**
    ❌a) Limits tools
    ❌b) Simplifies FunctionTool creation
    ❌d) Supports OpenAI tools only

19. **Where to find complete examples of MCP usage?**
    ✅**c) examples/mcp**
    ❌a) examples/tools
    ❌b) examples/handsoffs
    ❌d) examples/basic

20. **What challenge does MCP address?**
    ✅**b) Integrating LLMs with diverse external tools and data.**
    ❌a) Model accuracy
    ❌c) LLM latency
    ❌d) Prompt design

Handoffs

OpenAI Agents SDK - Handoffs Section MCQs

1. **What is the primary purpose of "Handoffs"?**
   ✅**c) To allow an agent to delegate tasks to another specialized agent.**
   ❌a) Run code in sandbox
   ❌b) Enable web search
   ❌d) Generate images

2. **How are handoffs represented to the LLM?**
   ✅**b) As tools**
   ❌a) Text instructions
   ❌c) API calls
   ❌d) State changes

3. **Default tool name for a Refund Agent handoff?**
   ✓**c) transfer_to_refund_agent**
   ✗a) refund_agent_tool
   ✗b) initiate_refund
   ✗d) handle_refunds

4. **Which Agent class parameter configures handoffs?**
   ✓**c) handoffs**
   ✗a) tools
   ✗b) delegations
   ✗d) sub_agents

5. **What else can `handoffs` accept besides an Agent?**
   ✓**b) A Handoff object**
   ✗a) FunctionTool
   ✗c) RunResult
   ✗d) RunContextWrapper

6. **Function to create a custom Handoff object?**
   ✓**c) handoff()**
   ✗a) create_handoff()
   ✗b) delegate_to()
   ✗d) define_handoff()

7. **Default tool name from `Handoff.default_tool_name()`?**
   ✓**c) transfer_to_<agent_name>**
   ✗a) handoff_to_agent_name
   ✗b) delegate_agent_name
   ✗d) handoff_agent_<name>

8. **What does tool_name_override do in handoff()?**
   ✓**c) Customize the tool name shown to LLM**
   ✗a) Change agent name
   ✗b) Specify return agent
   ✗d) Override tool description

9. **Purpose of `on_handoff` in handoff()?**
   ✓**b) Callback when handoff invoked, e.g., data fetch**
   ✗a) Prevent handoff
   ✗c) Redefine agent's instructions
   ✗d) Log output

10. `on_handoff` **can optionally receive?**
    ✓**c) LLM-generated input (via input_type)**
    ✗a) Full conversation history
    ✗b) Previous agent name
    ✗d) RunResult

11. **Purpose of** `input_type` **in handoff()?**
    ✓**b) Specify type of LLM-generated input to** `on_handoff`
    ✗a) Output type of target agent
    ✗c) Filter user messages
    ✗d) Async toggle

12. **Class used in EscalationData example for input?**
    ✓**b) BaseModel**
    ✗a) TypedDict
    ✗c) dataclass
    ✗d) NamedTuple

13. **How does the new agent see the conversation by default?**
    ✓**b) Sees the full previous history**
    ✗a) Starts fresh
    ✗c) Only last user msg
    ✗d) Only previous output

14. **Purpose of** `input_filter` **in handoff()?**
    ✓**b) Modify input or filter history for new agent**
    ✗a) Validate output
    ✗c) Define allowed handoffs
    ✗d) Set target LLM

15. **What argument does input_filter receive?**
    ✓**b) HandoffInputData**
    ✗a) RunResult
    ✗c) AgentContext
    ✗d) List[dict]

16. **Where are common input filters like** `remove_all_tools` **found?**
    ✓**b) agents.extensions.handoff_filters**
    ✗a) agents.core.filters
    ✗c) agents.utils.input_filters
    ✗d) agents.config.filters

17. **Recommended practice for handoff clarity?**
✅**c) Include handoff info in agent instructions using recommended prompts**
❌a) input_filter=None
❌b) Always define tool_name_override
❌d) Limit handoffs

18. **What is `RECOMMENDED_PROMPT_PREFIX`?**
✅**c) A suggested prefix for better LLM handoff understanding**
❌a) Dynamic prompt generator
❌b) Parameter to disable prompts
❌d) Default handoff name

19. **Function to auto-include handoff info in prompts?**
✅**c) prompt_with_handoff_instructions()**
❌a) add_handoff_instructions()
❌b) generate_handoff_prompt()
❌d) configure_handoff_prompt()

20. **Main benefit of specialized agents via handoffs?**
✅**c) Better specialization & modularity**
❌a) Reduce LLM calls
❌b) Make agents less modular
❌d) Simpler debugging

21. **In Triage agent example – which agent uses which handoff method?**
✅**b) billing_agent uses Agent directly, refund_agent uses handoff()**
❌a) refund_agent uses Agent directly
❌c) both use handoff()
❌d) both use Agent

22. **Purpose of `tool_description_override`?**
✅**c) Customize description seen by LLM**
❌a) Shorten description
❌b) Auto generate
❌d) Describe inputs

23. **What does EscalationData's `on_handoff` show?**
✅**c) Trigger structured actions/logs from LLM input**
❌a) Auto retraining
❌b) Passive observation
❌d) Disable further tool calls

24. **Why might an input_filter remove past tool calls?**
✅**c) Focus next agent only on new input/convo**
❌a) Block tool usage
❌b) Reduce tokens
❌d) Speed handoff

25. **What does modeling handoffs as tools enable?**
✅**c) LLM decides when/whom to delegate to**
❌a) Modify internal logic
❌b) Only run pre-defined functions
❌d) Bypass guardrails

Tracing

OpenAI Agents SDK - Tracing Section MCQs

� Part 1: Overview & Trace/Span Fundamentals

1. **Primary purpose of built-in tracing:**
✅**b) To collect a comprehensive record of events for debugging, visualization, and monitoring.**
❌a) Optimize performance
❌c) Version control
❌d) Encrypt data

2. **NOT collected by tracing:**
✅**d) Agent instruction changes**
❌a) LLM generations
❌b) Tool calls
❌c) Handoffs

3. **Is tracing enabled by default?**
✅**b) Enabled by default.**
❌a) Disabled
❌c) OS-dependent
❌d) Production-only

4. **Globally disable tracing?**
✅**b) Set `OPENAI_AGENTS_DISABLE_TRACING=1`.**
❌a) RunConfig
❌c) trace.disable_all()
❌d) Not possible

5. **Tracing under Zero Data Retention (ZDR)?**
   ✅**c) Tracing is unavailable.**
   ❌a) Auto-configured
   ❌b) Redacted traces
   ❌d) Custom processors required

6. **What is a Trace?**
   ✅**c) A single end-to-end operation or "workflow".**
   ❌a) LLM call
   ❌b) Agent action
   ❌d) Error log

7. **Required Trace property:**
   ✅**c) workflow_name**
   ❌a) started_at
   ❌b) parent_id
   ❌d) disabled

8. **Trace ID format:**
   ✅**b) trace_<32_alphanumeric>**
   ❌a) trace-<UUID>
   ❌c) trace_YYYYMMDD_HHMMSS
   ❌d) Any string

9. **group_id purpose:**
   ✅**c) Link multiple traces from same conversation/flow.**
   ❌a) Agent type
   ❌b) Severity
   ❌d) Duration limit

10. **What are spans?**
    ✅**b) Operations with start and end times.**
    ❌a) Workflow duration
    ❌c) Trace metadata
    ❌d) Final output

11. **Property pointing to parent span:**
    ✅**b) parent_id**
    ❌a) trace_id
    ❌c) group_id
    ❌d) span_data

12. **What does GenerationSpanData contain?**
✅**c) LLM generation details.**
❌a) Agent init
❌b) Tool I/O
❌d) Guardrails

▯ Part 2: Default Tracing & Creating Traces/Spans

13. **How are Runner.run methods traced?**
✅**b) Wrapped in a `trace()`**
❌a) custom_span
❌c) Not traced
❌d) agent_span

14. **Span type wrapping agent run:**
✅**c) agent_span()**
❌a) run_span
❌b) agent_exec_span
❌d) llm_call_span

15. **Default `workflow_name` if not set:**
✅**c) Agent trace**
❌a) Default workflow
❌b) SDK Trace
❌d) Unnamed trace

16. **Recommended way to create trace:**
✅**b) Use `trace()` as a context manager.**
❌a) Manual start/finish
❌c) RunConfig
❌d) asyncio.run()

17. **Update trace contextvars manually:**
✅**b) mark_as_current and reset_current**
❌a) set_current
❌c) start_current
❌d) activate

18. **Need to manually create spans?**
✅**b) No, most common ops are traced automatically.**
❌a) Yes

❌c) Only LLM
❌d) Only handoffs

19. **Track custom span info:**
✅**b) custom_span()**
❌a) log_span
❌c) new_span
❌d) record_span

20. **How are custom spans nested?**
✅**c) Using Python contextvars for current trace/span.**
❌a) Explicit parent_id
❌b) Global var
❌d) RunConfig flag

� Part 3: Sensitive Data & Custom Processors

21. **Span storing LLM I/O (may be sensitive):**
✅**c) generation_span()**
❌a) agent_span
❌b) tool_call_span
❌d) handoff_span

22. **Disable sensitive data capture:**
✅**c) RunConfig.trace_include_sensitive_data = False**
❌a) Env variable
❌b) metadata.sensitive
❌d) Use custom_span

23. **Audio spans store (by default):**
✅**c) Base64-encoded PCM data.**
❌a) File paths
❌b) Transcripts
❌d) Metadata

24. **Disable sensitive audio capture:**
✅**c) VoicePipelineConfig.trace_include_sensitive_audio_data = False**
❌a) RunConfig
❌b) trace_include_sensitive_data
❌d) Not possible

25. **Component that creates traces:**
✓**b) TraceProvider**
✗a) TraceManager
✗c) SpanFactory
✗d) TraceGenerator

26. **BatchTraceProcessor sends traces to:**
✓**c) BackendSpanExporter**
✗a) Log file
✗b) Remote DB
✗d) StreamEvent

27. **BackendSpanExporter purpose:**
✓**b) Export spans/traces to OpenAI backend.**
✗a) Process sensitive data
✗c) Human-readable output
✗d) Manage span lifecycle

28. **Add additional processor:**
✓**b) add_trace_processor()**
✗a) set_trace_processors
✗c) register_trace_processor
✗d) configure_trace_processor

29. **Consequence of `set_trace_processors()` use:**
✓**c) You must include OpenAI exporter explicitly.**
✗a) Auto includes both
✗b) Custom must forward
✗d) Local-only effect

30. **NOT a supported tracing backend:**
✓**d) Prometheus**
✗a) Weights & Biases
✗b) LangSmith
✗c) MLflow

31. **Why are multiple runs inside `trace("Joke workflow")`?**
✓**c) To group multiple `Runner.run()` calls into one trace.**
✗a) Trace final only
✗b) Disable tracing
✗d) Prevent concurrency

32. **Span for audio input (speech-to-text):**
✅**c) transcription_span()**
❌a) audio_input_span
❌b) speech_to_text_span
❌d) input_audio_span

33. **Span for audio output (text-to-speech):**
✅**b) speech_span()**
❌a) audio_output_span
❌c) text_to_speech_span
❌d) output_audio_span

34. **Parent span for related audio spans:**
✅**c) speech_group_span()**
❌a) audio_workflow_span
❌b) voice_interaction_span
❌d) composite_audio_span

35. **Purpose of Traces dashboard:**
✅**c) Debugging, visualizing, monitoring workflows.**
❌a) Raw data collection
❌b) Deployment management
❌d) Model optimization

Context Management

OpenAI Agents SDK - Context Management Section MCQs

� Part 1: Local Context

1. **Two main classes of context?**
✅**c) Context available locally to your code and Context available to LLMs**
❌a) Input/Output
❌b) Static/Dynamic
❌d) User/System Context

2. **Class representing local context?**
✅**c) RunContextWrapper**
❌a) AgentContext
❌b) LLMContext
❌d) LocalContext

3. **How to pass custom context to an agent run?**
✅**c) As a context keyword argument to Runner.run() methods**
❌a) Agent constructor
❌b) Env variable
❌d) Global config

4. **Accessing your context object from a wrapper?**
✅**b) wrapper.context**
❌a) get_context()
❌c) wrapper.data
❌d) wrapper.T

5. **Most important thing to know about context objects?**
✅**b) Every agent, tool, and hook must use the same type**
❌a) Must be immutable
❌c) Auto-sent to LLM
❌d) Only primitives allowed

6. **Common use case for local context?**
✅**c) Carrying contextual data like UID, logger, etc.**
❌a) Instructions
❌b) LLM history
❌d) Grounding responses

7. **Is local context visible to the LLM?**
✅**c) No, purely for local use**
❌a) Yes, always
❌b) If configured
❌d) If Pydantic

8. **UserInfo structure in example?**
✅**c) dataclasses.dataclass**
❌a) NamedTuple
❌b) BaseModel
❌d) TypedDict

9. **First argument of function tool needing context?**
✅**c) RunContextWrapper[UserInfo] (or RunContextWrapper[T])**
❌a) RunContext
❌b) Context
❌d) Any

▯ Part 2: Agent / LLM Context

10. **What does the LLM see for generation?**
    ✅**c) The conversation history**
    ❌a) Local context
    ❌b) RunConfig
    ❌d) Agent name/tools

11. **To make data visible to LLM?**
    ✅**b) Include in conversation history**
    ❌a) Direct function call
    ❌c) API call
    ❌d) Encrypted metadata

12. **System prompt = developer message = ?**
    ✅**c) Adding it to Agent instructions**
    ❌a) Input to Runner.run
    ❌b) Function tools
    ❌d) Retrieval/web search

13. **Instructions can be:**
    ✅**c) Static or dynamic functions returning strings**
    ❌a) Static only
    ❌b) Dynamic only
    ❌d) Pydantic only

14. **Common data for instructions:**
    ✅**b) Always useful info like name/date**
    ❌a) On-demand info
    ❌c) Computation results
    ❌d) Confidential data

15. **Runner.run input vs Agent instructions:**
    ✅**b) Input appears lower in prompt hierarchy**
    ❌a) Preferred for static info
    ❌c) Invisible to LLM
    ❌d) For sensitive data only

16. **Function tools are for what context type?**
    ✅**c) On-demand context**
    ❌a) Static

❌b) Pre-loaded
❌d) Internal logging

17. **When LLM *might* need data? Use:**
✅**c) Function tools**
❌a) Instructions
❌b) context=...
❌d) input_filter

18. **Retrieval / web search is for:**
✅**c) Fetching relevant data to ground responses**
❌a) Local context
❌b) Agent state
❌d) History encryption

19. **Grounding supported by:**
✅**c) Retrieval or web search**
❌a) run_config.metadata
❌b) RunContextWrapper
❌d) on_handoff

20. **Tell LLM current datetime always:**
✅**b) Include it dynamically in Agent instructions**
❌a) Use a function tool
❌c) context=...
❌d) Add to final_output

21. **Which is true about RunContextWrapper?**
✅**b) Can pass dependencies like loggers**
❌a) Prevents tool calls
❌c) Auto-added to LLM
❌d) Always visible in Traces

22. **Main takeaway from local vs LLM context?**
✅**c) Clear separation between internal and LLM data**
❌a) All data must be visible to LLM
❌b) LLM accesses Python objects
❌d) Local = debug, LLM = prod

23. **Agent[UserInfo] + context=ProductInfo → ?**
✅**c) Error or unexpected behavior due to type mismatch**
❌a) Adapts automatically

❌b) Ignores context
❌d) ProductInfo overrides

24. **Dynamic Agent instructions allow for:**
✅**c) Personalized system prompts using local context**
❌a) Real-time API calls
❌b) Change LLM model
❌d) Disable tracing

25. **Best way to enhance LLM reasoning w/ external info:**
✅**c) Inject into conversation history or expose via tools**
❌a) Modify model weights
❌b) Use hosted tools only
❌d) Rely on pre-trained knowledge

Guardrails

OpenAI Agents SDK - Guardrails Section MCQs

Part 1: General Concepts and Types

1. **How do guardrails run in relation to agents?**
✅**b) In parallel to your agents**
❌a) Sequentially
❌c) Only after
❌d) As sub-process

2. **Key benefit of guardrails (slow/expensive model example)?**
✅**b) Immediately halt agent execution and save time/money if misuse is detected**
❌a) Enhance reasoning
❌c) Detailed tracing
❌d) Rephrase input

3. **How many guardrail types are described?**
✅**b) Two**

❌a) One
❌c) Three
❌d) Four

4. **Which runs on initial user input?**
✅**c) Input guardrails**
❌a) Output
❌b) Pre-processing
❌d) User

5. **Which runs on final agent output?**
✅**c) Output guardrails**
❌a) Input
❌b) Post-processing
❌d) Final

� Part 2: Input and Output Guardrail Mechanics

6. **What does an Input Guardrail receive first?**
✅**a) The same input passed to the agent**
❌b) Agent instructions
❌c) LLM output
❌d) History

7. **Input guardrail result is wrapped in?**
✅**b) InputGuardrailResult**
❌a) GuardrailResult
❌c) GuardrailOutput
❌d) InputCheckResult

8. **If `.tripwire_triggered` is true → ?**
✅**c) Raises InputGuardrailTripwireTriggered exception**
❌a) Logs and continues
❌b) Rephrases input
❌d) Reroutes agent

9. **When do input guardrails run?**
✅**c) Only if the agent is the first in the flow**
❌a) If tools enabled
❌b) If streaming
❌d) If input has sensitive data

10. **Why are guardrails attached to Agent object?**
✅**c) Because they relate to the specific agent (better readability)**
❌a) Enforce global policy
❌b) Simplify run()
❌d) Modify state

11. **Key difference in input vs output guardrail execution?**
✅**c) Input: on user input; Output: on final agent output**
❌a) Tool timing
❌b) Sync/async
❌d) Fast vs slow model

12. **When do output guardrails run?**
✅**c) Only if the agent is the last agent in the execution flow**
❌a) JSON output
❌b) Handoff
❌d) LLM message

� Part 3: Tripwires and Implementation

13. **What does a tripwire signal?**
✅**c) Input/output failed guardrail check**
❌a) Success
❌b) Ready to rerun
❌d) New type detected

14. **Effect of triggered tripwire?**
✅**c) Halts execution, raises exception**
❌a) Self-corrects
❌b) Logs warning
❌d) Re-prompts user

15. **Required return type for guardrail functions?**
✅**d) GuardrailFunctionOutput**
❌a) bool
❌b) str
❌c) dict

16. **How is math homework checked in example?**
✅**c) Runs another Agent (guardrail agent)**
❌a) Keyword search
❌b) External API
❌d) Forbidden phrase DB

17. **How is `.tripwire_triggered` determined?**
    ✅**c) From final_output's is_math_homework / is_math boolean**
    ❌a) Hardcoded
    ❌b) Always False
    ❌d) Context state

18. **Python construct to handle guardrail trip?**
    ✅**c) try...except for TripwireTriggered exception**
    ❌a) if/else
    ❌b) while loop
    ❌d) assert

19. **Input types accepted by `math_guardrail`?**
    ✅**c) str or list[TResponseInputItem]**
    ❌a) str only
    ❌b) list only
    ❌d) Any

20. **Decorator for input guardrail?**
    ✅**c) @input_guardrail**
    ❌a) @guardrail
    ❌b) @agent_guardrail
    ❌d) @check_input

21. **Decorator for output guardrail?**
    ✅**d) @output_guardrail**
    ❌a) @guardrail
    ❌b) @agent_guardrail
    ❌c) @check_output

22. **Why is OutputGuardrailTripwireTriggered raised?**
    ✅**c) To prevent finalizing/sending non-compliant output**
    ❌a) Retry
    ❌b) Alt destination
    ❌d) Help LLM understand output

23. **Why use a separate agent for guardrails?**
    ✅**b) To leverage LLM for complex logic**
    ❌a) Must use agents
    ❌c) Fewer deps
    ❌d) Sync execution

24. **What's in `output_info` of GuardrailFunctionOutput?**
    ✅**c) final_output from internal guardrail agent**
    ❌a) Boolean
    ❌b) Error msg
    ❌d) Raw input

25. **Why run guardrails "in parallel"?**
    ✅**b) Efficiency – avoid costly operations if early issues**
    ❌a) Code complexity
    ❌c) Model choice
    ❌d) UI responsiveness

Orchestrating Multiple Agents

OpenAI Agents SDK - Orchestrating Multiple Agents Section MCQs

□ Part 1: Overview and Orchestrating via LLM

---

1. What does "Orchestration" refer to in agent apps?

- ✅**b) The flow of agents, including their order and decision-making for what happens next.**
- ❌a) Deployment process
- ❌c) LLM memory management
- ❌d) Training agent models

---

2. Two main ways to orchestrate agents:

- ✅**c) Allowing the LLM to make decisions and Orchestrating via code.**
- ❌a) Synchronous and Asynchronous
- ❌b) Local and Remote
- ❌d) Sequential and Parallel

---

3. What defines an LLM-based agent?

- ✅**c) Instructions, Tools, and Handoffs.**
- ❌a) Memory, Sensors, Actuators
- ❌b) Models, Data, Logging
- ❌d) Prompts, Outputs, Traces

---

4. How does LLM tackle open-ended tasks?

- ✅**b) Autonomously plans, reasons, and uses tools/handoffs.**

- ❌a) Follows script
- ❌c) Always asks for human help
- ❌d) Randomly chooses

---

5. Which is NOT a tool for LLM agents?

- ✅**d) Model Fine-tuning.**
- ❌a) Web search
- ❌b) File search
- ❌c) Computer use

---

6. When is LLM orchestration ideal?

- ✅**b) Open-ended tasks where LLM reasoning is needed.**
- ❌a) Deterministic tasks
- ❌c) Low-resource tasks
- ❌d) Rule-based logic

---

7. Important tactic in LLM orchestration:

- ✅**c) Use good prompts to define tools and parameters.**
- ❌a) Fewer agents
- ❌b) Avoid tools
- ❌d) Disable monitoring

---

8. Meaning of "introspect and improve":

- ✅**b) Run in a loop, self-critique, and learn from errors.**
- ❌a) Manually update code
- ❌c) Restrict access
- ❌d) Give only positive feedback

---

9. Why use specialized agents?

- ✅**b) So each agent excels in one thing.**
- ❌a) Makes debugging harder
- ❌c) Reduces agent count
- ❌d) Forces handoffs

---

10. Why "Invest in evals"?

- ✅**b) Helps train and improve agent performance.**
- ❌a) Reduces LLM cost
- ❌c) Eases deployment
- ❌d) Real-time error fixing

□□ Part 2: Orchestrating via Code

11. Benefit of code orchestration:

- ✅**b) More deterministic and predictable (speed, cost).**
- ❌a) Flexibility
- ❌c) Fewer prompts needed
- ❌d) Easier LLM integration

12. Structured output use case:

- ✅**c) Classify task, then choose agent.**
- ❌a) Write full blog
- ❌b) Parallel tasks
- ❌d) Self critique

13. What is "chaining agents"?

- ✅**c) Output of one → Input of next agent.**
- ❌a) Run simultaneously
- ❌b) One agent, many jobs
- ❌d) Recursive calls

14. Chaining example:

- ✅**c) Writing a blog post (research → write → critique).**
- ❌a) Customer service
- ❌b) Math problems
- ❌d) Translation

15. Pattern using a while loop:

- ✅**c) Iterative feedback until criteria met.**
- ❌a) Parallel tasks
- ❌b) Task classification
- ❌d) Structured data creation

16. Python method for parallel agents:

- ✅**c) asyncio.gather**
- ❌a) threading.Thread
- ❌b) multiprocessing.Process

- ❌d) ThreadPoolExecutor

---

17. When to use parallel agents:

- ✅**b) When tasks are independent and need speed.**
- ❌a) Tasks depend on each other
- ❌c) Let LLM decide
- ❌d) Debugging

---

18. Why mix LLM and code orchestration?

- ✅**c) Combines LLM reasoning with code precision.**
- ❌a) You must pick one
- ❌b) Makes design harder
- ❌d) Only works for simple tasks

---

19. Where are orchestration examples?

- ✅**c) In `examples/agent_patterns` folder.**
- ❌a) SDK README
- ❌b) Blog post
- ❌d) API docs

---

20. Best for precise, cost-controlled tasks:

- ✅**b) Orchestrating via code.**
- ❌a) LLM-based
- ❌c) LLM-heavy hybrid
- ❌d) Human-in-loop

Models

Models

OpenAI Agents SDK - Models Section MCQs

� Part 1: OpenAI Models and Non-OpenAI Integration

1. What is the recommended model class to use with OpenAI in the SDK?
   a) OpenAIChatCompletionsModel
   ✅**b) OpenAIResponsesModel**
   c) OpenAITextCompletionModel
   d) OpenAILegacyModel

2. Which API does OpenAIChatCompletionsModel use?
   a) Responses API
   ✅**b) Chat Completions API**
   c) Embeddings API
   d) Moderations API

3. What dependency is needed to use most non-OpenAI models?
   a) openai-agents[all]
   b) openai-agents[models]
   ✅**c) openai-agents[litellm]**
   d) openai-agents[external]

4. What prefix is used when integrating with LiteLLM models?
   a) ollama/
   b) external/
   ✅**c) litellm/**
   d) custom/

5. How can you integrate other LLM providers (besides LiteLLM)?
   a) Modify Runner constructor
   b) Edit SDK source code
   ✅**c) Use `set_default_openai_client`**
   d) Tool-level ModelFactory

6. `set_default_openai_client()` is useful when the endpoint is:
   a) Proprietary API
   ✅**b) OpenAI-compatible API endpoint**
   c) XML-based
   d) GraphQL-based

7. Where is ModelProvider applied to cover all agents in one run?
   a) Tool function level
   ✅**b) Runner.run level**
   c) Global app lifecycle
   d) Inside Agent constructor

8. How do you assign different LLM providers to different agents?
   a) set_default_openai_client
   b) model_provider in Runner
   ✅**c) Set `Agent.model` per agent**
   d) Use only LiteLLM

9. What to do if you don't have an OpenAI API key?
   a) Use dummy key
   ✅**b) Disable tracing via `set_tracing_disabled()`**
   c) Nothing needed
   d) Tracing adjusts automatically

10. Which API is often used with non-OpenAI models in examples?
   a) Embeddings API
   ✅**b) Chat Completions API/model**
   c) Completions API
   d) Fine-tuning API

� Part 2: Mixing & Matching Models

11. Why use different models per agent?
   a) Increase latency
   b) Add security complexity
   ✅**c) Use smaller model for triage, larger for complex tasks**
   d) Limit agents

12. Which is NOT a valid way to select a model?
   a) Provide model name like "gpt-3.5-turbo"
   b) Use ModelProvider with name
   c) Use OpenAIChatCompletionsModel(...)
   ✅**d) Write model in Agent instructions**

13. Why use a single model shape per workflow?
   a) Ensure pricing consistency
   b) Simplify logging
   ✅**c) Model shapes support different tools/features**
   d) Optimize disk

14. What to ensure if mixing model shapes?
   a) Don't mix
   b) Only use Responses

c) Use only Chat
✓**d) Features must exist in both shapes**

15. Purpose of `triage_agent` in model mix example?
    a) Analyze data
    b) Multilingual generation
    ✓**c) Route to agent based on language**
    d) Evaluate other agents

16. Which param is used for settings like temperature?
    a) model_config
    b) llm_settings
    ✓**c) model_settings**
    d) config_params

17. Effect of `temperature=0.1`?
    a) More creative
    ✓**b) More deterministic, less random**
    c) Increase length
    d) Reduce tokens

🔹 Part 3: Issues with Other LLM Providers

18. Cause of "Tracing client error 401"?
    a) Model not compatible
    ✓**b) No OpenAI API key for tracing**
    c) Bad internet
    d) Temp too high

19. How to fix tracing error 401?
    a) Set empty key
    b) Use OpenAIResponsesModel
    ✓**c) Call `set_tracing_export_api_key(...)` with OpenAI key**
    d) Update agent instructions

20. Fix for 404 error due to Responses API unsupported?
    a) Disable tracing
    b) Change litellm version
    ✓**c) Use `set_default_openai_api("chat_completions")` or switch to OpenAIChatCompletionsModel**
    d) Set custom ModelProvider

21. Error message when json_schema not supported?
   a) Host not found
   b) Invalid model name
   ✅**c) BadRequestError: `'response_format.type'...`**
   d) Unexpected keyword

22. SDK suggestion when provider lacks JSON schema support?
   a) Manually parse text
   b) Set temp to 0
   ✅**c) Use provider that supports JSON schema**
   d) Reduce input size

23. What to consider when mixing providers?
   a) All features same
   b) Performance same
   ✅**c) Feature differences like multimodal & structured output**
   d) API keys interchangeable

24. Precaution with text-only models?
   a) Long inputs
   b) JSON inputs
   ✅**c) Filter out multimodal inputs**
   d) One-word prompts

25. OpenAI API key for tracing must be from:
   a) Current provider
   ✅**b) platform.openai.com**
   c) Any cloud provider
   d) Local only

26. SDK model interactions are mainly:
   a) Synchronous
   ✅**b) Asynchronous**
   c) Blocking
   d) Multi-threaded

27. How does SDK get API key & URL for OpenAI?
   a) Passed manually
   b) Hardcoded
   ✅**c) From `OPENAI_API_KEY` & `OPENAI_BASE_URL` env vars**
   d) Generates each time

28. Example parameter in ModelSettings:
   a) api_key
   b) base_url
   c) model_name
   ✓**d) temperature**

29. Result if provider doesn't support `json_schema`?
   a) Output is converted
   b) Agent uses OpenAI
   ✓**c) App breaks with malformed JSON**
   d) Model gives empty string

30. Where are example configs for non-OpenAI models?
   a) examples/models/
   b) examples/providers/
   ✓**c) examples/model_providers/**
   d) examples/custom_models/

Using any model via LiteLLM

OpenAI Agents SDK - Using any Model via LiteLLM Section MCQs

Using Any Model via LiteLLM – MCQs

1. What is the current status of the LiteLLM integration in the Agents SDK?
   a) Stable and production-ready
   ✓**b) In beta**
   c) Deprecated
   d) Under experimental development

2. If you encounter issues with the LiteLLM integration, especially with smaller model providers, what is the recommended action?
   a) Revert to using OpenAI models only
   b) Wait for a new SDK version automatically
   ✓**c) Report the issues via GitHub issues**
   d) Try a different LiteLLM version

3. What is LiteLLM described as?
   a) A new type of LLM from OpenAI
   ✓**b) A library that allows you to use 100+ models via a single interface**

c) A debugging tool for agents
d) An SDK for creating custom tools

4. To enable the LiteLLM integration, which specific command should you run?
a) pip install litellm
b) pip install openai-agents
✅**c) pip install "openai-agents[litellm]"**
d) pip install openai-agents-models

5. Once LiteLLM is installed, which class should you use within an Agent to specify a LiteLLM model?
a) OpenAIResponsesModel
b) LitellmClient
✅**c) LitellmModel**
d) ExternalModel

6. In the provided example, when running the script, what information will you be prompted to enter?
a) Only the model name
b) Only the API key
✅**c) A model name and an API key**
d) The agent's instructions

7. Which of the following is an example of a model name format for LiteLLM given in the documentation?
a) gpt-4.1
✅**b) openai/gpt-4.1**
c) litellm_gpt-4.1
d) gpt-4.1-openai

8. Where can you find a full list of models supported by LiteLLM?
a) In the Agents SDK documentation
b) On the OpenAI API reference page
✅**c) In the LiteLLM providers docs**
d) Within the agents.extensions.models module

9. In the example `main()` function, what are the `model` and `api_key` arguments passed to when creating the LitellmModel instance?
a) name and instructions
b) tools and handoffs
✅**c) model and api_key**
d) temperature and top_p

10. What specific instruction is given to the Assistant agent in the example, which dictates its response style?
a) "You are a helpful assistant."

b) "You only speak English."
✓**c) "You only respond in haikus."**
d) "You must use the get_weather tool."

11. The `get_weather` function in the example is decorated with `@function_tool`. This means it will be exposed to the LLM as what?
a) An internal utility
b) A local context provider
✓**c) A callable tool**
d) A guardrail

12. The example code includes `set_tracing_disabled()`. What is the likely reason for including this in a LiteLLM example?
a) Tracing is not compatible with LiteLLM models
b) It's a standard practice for all agent examples
✓**c) Users might not have an OpenAI API key for tracing, which is needed by default**
d) To improve the performance of LiteLLM models

13. How does the example script obtain the model and api_key if they are not provided via command-line arguments?
a) It uses predefined environment variables
b) It fetches them from a configuration file
✓**c) It prompts the user for input using `input()`**
d) It generates them dynamically

14. What library is used in the example to handle asynchronous execution of the main function?
a) threading
b) multiprocessing
✓**c) asyncio**
d) concurrent.futures

15. The LiteLLM integration allows you to use `anthropic/claude-3-5-sonnet-20240620`. This demonstrates the ability to use models from providers other than just OpenAI, such as:
a) Google
b) Cohere
✓**c) Anthropic**
d) Hugging Face

Configuring the SDK

OpenAI Agents SDK - Configuring the SDK Section MCQs

� Part 1: API Keys and Clients

1. **By default, where does the Agents SDK look for the OpenAI API key for LLM requests and tracing?**
   a) In a config.ini file
   b) Directly from ~/.openai/credentials
   ✅**c) In the OPENAI_API_KEY environment variable**
   d) As a command-line argument

2. **If you cannot set the OPENAI_API_KEY environment variable before your app starts, which function can you use to configure it?**
   a) set_openai_key()
   ✅**b) set_default_openai_key()**
   c) configure_api_key()
   d) set_env_variable()

3. **By default, what type of OpenAI client instance does the SDK create?**
   a) OpenAI (synchronous)
   ✅**b) AsyncOpenAI**
   c) SyncOpenAI
   d) OpenAIClient

4. **To use a custom AsyncOpenAI instance (e.g., with a different base_url), which function should you use?**
   a) set_custom_client()
   b) configure_openai_instance()
   ✅**c) set_default_openai_client()**
   d) override_openai_client()

5. **What is the default OpenAI API that the Agents SDK uses?**
   a) The Chat Completions API
   ✅**b) The Responses API**
   c) The Embeddings API
   d) The Legacy Completions API

6. **To override the default OpenAI API to use the Chat Completions API, which function is available?**
   a) use_chat_completions_api()
   b) set_openai_api_type("chat_completions")
   ✅**c) set_default_openai_api("chat_completions")**
   d) configure_api_version()

Part 2: Tracing and Logging

7. **What is the default state of tracing in the Agents SDK?**
   ✓**a) Enabled**
   b) Disabled
   c) It depends on the environment variable
   d) It requires manual configuration for each run

8. **By default, where does tracing get its API key from?**
   a) A separate OPENAI_TRACING_KEY environment variable
   b) A dedicated tracing configuration file
   ✓**c) The same OpenAI API keys used for LLM requests (environment variable or set_default_openai_key())**
   d) It generates a temporary key for each trace

9. **Which function allows you to set a specific API key to be used only for tracing, separate from LLM requests?**
   a) set_tracing_key()
   b) configure_trace_api()
   ✓**c) set_tracing_export_api_key()**
   d) override_trace_key()

10. **How can you disable tracing entirely in the SDK?**
    a) By setting OPENAI_TRACING_ENABLED=False environment variable
    ✓**b) By calling set_tracing_disabled(True)**
    c) By removing the openai.agents.tracing logger handler
    d) Tracing cannot be disabled

11. **By default, without any handlers set, which logging levels from the SDK's Python loggers are sent to stdout?**
    a) All logs (DEBUG, INFO, WARNING, ERROR, CRITICAL)
    b) Only ERROR logs
    ✓**c) Only WARNING and ERROR logs**
    d) INFO and above

12. **To enable verbose logging from the SDK to stdout, which convenience function can be used?**
    a) enable_debug_logging()
    b) set_verbose_logging(True)
    ✓**c) enable_verbose_stdout_logging()**
    d) configure_logging_level("VERBOSE")

13. **If you want to customize logging beyond the convenience function, which Python module is recommended to use?**
    a) sys

✅**b) logging**
c) os
d) configparser

14. **What are the names of the two main Python loggers in the SDK that you can customize?**
a) "openai.sdk" and "openai.logs"
✅**b) "openai.agents" and "openai.agents.tracing"**
c) "agents.main" and "agents.debug"
d) "openai.llm" and "openai.tools"

15. **To prevent logging of LLM inputs and outputs, which environment variable should be set?**
a) OPENAI_AGENTS_DISABLE_LLM_LOGS=1
b) OPENAI_AGENTS_NO_LLM_DATA=1
✅**c) OPENAI_AGENTS_DONT_LOG_MODEL_DATA=1**
d) OPENAI_AGENTS_HIDE_PROMPTS=1

16. **To disable logging of tool inputs and outputs, which environment variable should be set?**
a) OPENAI_AGENTS_DISABLE_TOOL_LOGS=1
b) OPENAI_AGENTS_NO_TOOL_DATA=1
c) OPENAI_AGENTS_HIDE_TOOL_CALLS=1
✅**d) OPENAI_AGENTS_DONT_LOG_TOOL_DATA=1**

17. **If you set a logger's level to logging.INFO, what types of logs will typically be displayed (assuming a handler is present)?**
a) Only INFO logs
b) DEBUG and INFO logs
✅**c) INFO, WARNING, ERROR, and CRITICAL logs**
d) Only WARNING and ERROR logs

18. **What is the default output stream for logs when no handlers are explicitly added, as per the logging.StreamHandler example?**
✅**a) stdout**
b) stderr
c) A file named agents.log
d) A network socket

19. **Why might you want to disable logging of sensitive data (like LLM or tool inputs/outputs)?**
a) To improve application performance significantly
✅**b) To ensure user data privacy and compliance**
c) To reduce the number of API calls made to OpenAI
d) To make debugging easier

20. **The ability to set base_url when configuring a custom AsyncOpenAI client is particularly useful for what scenario?**
    ✅**a) When you want to use a proxy server or an OpenAI-compatible local LLM**
    b) When you want to specifically connect to OpenAI's European data center
    c) When you are using a different API version for OpenAI
    d) When you want to enable real-time streaming of responses

Agent Visualization

OpenAI Agents SDK - Agent Visualization Section MCQs

▯ Part: Agent Visualization in the SDK

1. **What is the primary purpose of Agent Visualization in the SDK?**
   a) To automatically optimize agent performance.
   b) To generate executable code from agent definitions.
   ✅**c) To create a structured graphical representation of agents and their relationships.**
   d) To simulate agent behavior without running them.

2. **Which external library is used by the Agents SDK for generating these graphical representations?**
   a) Matplotlib
   b) NetworkX
   ✅**c) Graphviz**
   d) Mermaid.js

3. **To install the necessary dependencies for agent visualization, which optional dependency group should you install?**
   a) "openai-agents[graph]"
   ✅**b) "openai-agents[viz]"**
   c) "openai-agents[draw]"
   d) "openai-agents[render]"

4. **Which function is used to generate an agent visualization graph?**
   a) generate_graph()
   b) visualize_agents()
   c) create_diagram()
   ✅**d) draw_graph()**

5. **In the generated graph, how are Agents represented visually?**
   a) Green ellipses.
   ✅**b) Yellow boxes (rectangles).**
   c) Blue circles.
   d) Red diamonds.

6. **How are Tools represented in the generated agent visualization graph?**
   a) Yellow boxes.
   ✓**b) Green ellipses.**
   c) Blue circles.
   d) Red diamonds.

7. **What do solid arrows in the generated graph typically represent?**
   a) Tool invocations.
   b) Data flow between agents.
   ✓**c) Agent-to-agent handoffs.**
   d) Conditional execution paths.

8. **What do dotted arrows in the generated graph indicate?**
   a) Agent-to-agent handoffs.
   ✓**b) Tool invocations.**
   c) Bidirectional communication.
   d) Error pathways.

9. **What is the purpose of the `__start__` node in the visualization?**
   a) It represents the end of the execution flow.
   b) It indicates a sub-agent.
   ✓**c) It indicates the entry point of the agent system.**
   d) It marks a debugging breakpoint.

10. **In the example usage, which agent is passed to draw_graph to visualize the entire structure?**
    a) spanish_agent
    b) english_agent
    ✓**c) triage_agent**
    d) get_weather

11. **By default, how does draw_graph display the generated graph?**
    ✓**a) Inline.**
    b) In a separate browser tab.
    c) As a text-based ASCII diagram.
    d) It automatically saves it as a PDF.

12. **To display the generated graph in a separate window, what method should be chained to the draw_graph function call?**
    a) .display()
    b) .show()
    ✓**c) .view()**
    d) .open()

13. **To save the generated graph as a file, which parameter should be used in the draw_graph function?**
    a) output_file
    b) save_as
    ✅**c) filename**
    d) export_path

14. **If you use `draw_graph(triage_agent, filename="agent_graph")`, what will be the name and default format of the generated file?**
    a) agent_graph.txt
    b) agent_graph.pdf
    c) agent_graph.svg
    ✅**d) agent_graph.png**

15. **The `triage_agent` in the example is configured with `handoffs=[spanish_agent, english_agent]`. How will these relationships be visually represented in the graph?**
    a) With green ellipses connecting the agents.
    b) With dotted arrows from triage_agent to spanish_agent and english_agent.
    ✅**c) With solid arrows from triage_agent to spanish_agent and english_agent.**
    d) These relationships will not be shown in the graph.

Release Process

OpenAI Agents SDK - Release Process Section MCQs

▯ Part: Agents SDK Versioning

1. **What versioning format does the OpenAI Agents SDK project follow?**
   a) Major.Minor.Patch (X.Y.Z)
   b) Year.Month.Day (YYYY.MM.DD)
   ✅**c) A modified semantic versioning using the form 0.Y.Z**
   d) Incremental build numbers (0.0.0.X)

---

2. **What does the leading `0` in the `0.Y.Z` versioning scheme signify for the SDK?**
   a) It's a stable, production-ready release.
   b) It indicates a pre-alpha stage.
   ✅**c) The SDK is still evolving rapidly.**
   d) It is a long-term support release.

---

3. **When are Minor (Y) versions incremented (e.g., from 0.0.x to 0.1.x)?**
   a) For any new feature.
   b) For bug fixes only.

✓**c) For breaking changes to any public interfaces that are not marked as beta.**
d) For changes to private interfaces.

---

4. **If a user wants to avoid breaking changes in their project, what is the recommended approach for pinning the SDK version?**
   a) Pinning to 0.Y.Z for specific Y values.
   b) Pinning to 0.Y.X with X as a wildcard.
   ✓**c) Pinning to 0.0.x versions.**
   d) Not pinning at all, and always using the latest.

---

5. **What kind of changes trigger an increment in Patch (Z) versions?**
   a) Only major architectural overhauls.
   b) Changes that require users to rewrite significant parts of their code.
   ✓**c) Non-breaking changes.**
   d) Changes that introduce new models.

---

6. **Which of the following types of changes would result in a Patch (Z) version increment?**
   a) Renaming a public class.
   b) Changing the required arguments for a public function.
   ✓**c) A bug fix.**
   d) Removing a publicly exposed method.

---

7. **If the SDK goes from version 0.0.5 to 0.1.0, what type of change has likely occurred?**
   a) A non-breaking new feature.
   b) A bug fix.
   ✓**c) A breaking change to a public interface.**
   d) An update to a beta feature.

---

8. **If a new feature is added to the SDK, but it does not break any existing public interfaces, what version component will be incremented?**
   a) The leading 0.
   b) The Minor (Y) version.
   ✓**c) The Patch (Z) version.**
   d) A new component will be added.

---

9. **Changes made to "private interfaces" typically result in an increment of which version component?**
   a) Minor (Y) version, as they are still changes.
   b) The leading 0, to indicate internal shifts.
   ✓**c) Patch (Z) version, as they are non-breaking for public users.**
   d) No version increment, as they are internal.

---

10. **If a beta feature receives an update, but it's not a bug fix, which version component would typically be incremented?**
    a) Minor (Y) version.
    b) The leading 0.
    ✅**c) Patch (Z) version.**
    d) A separate beta version number.

Voice agents

OpenAI Agents SDK - Quickstart (Voice) Section

□ Part 1: Prerequisites & Core Concepts

1. **What is the specific command to install the optional voice dependencies for the Agents SDK?**
   a) pip install 'openai-agents[audio]'
   b) pip install 'openai-agents[speech]'
   ✅**c) pip install 'openai-agents[voice]'**
   d) pip install 'openai-agents[sound]'

---

2. **What is the main concept introduced for voice-enabled applications in the SDK?**
   a) AudioProcessor
   b) SpeechWorkflow
   ✅**c) VoicePipeline**
   d) AgentListener

---

3. **How many steps are involved in the VoicePipeline process?**
   a) 2
   ✅**b) 3**
   c) 4
   d) 5

---

4. **Which of the following is NOT one of the three steps in the VoicePipeline?**
   a) Run a speech-to-text model.
   ✅**b) Run a sentiment analysis model.**
   c) Run your code (usually an agentic workflow).
   d) Run a text-to-speech model.

---

5. **The VoicePipeline takes □ Audio Input and produces what as its final output?**
   a) Text response
   b) A transcribed file
   ✅**c) □ Audio Output**
   d) A video stream

🔲 Part 2: Agent Setup for Voice Pipeline

6. **In the example, what is the purpose of the `get_weather` function?**
   a) To transcribe audio input.
   b) To convert text to speech.
   ✅**c) To act as a function_tool for agents to get weather information.**
   d) To play audio output.

---

7. **What is the `spanish_agent`'s primary instruction regarding language?**
   a) It translates English to Spanish.
   b) It can speak both English and Spanish.
   ✅**c) It must speak in Spanish.**
   d) It only understands Spanish commands.

---

8. **Both the Spanish agent and the Assistant agent in the example use which specific LLM model?**
   a) gpt-4o
   ✅**b) gpt-4o-mini**
   c) gpt-3.5-turbo
   d) claude-3-5-sonnet-20240620

---

9. **What is the Assistant agent instructed to do if the user speaks in Spanish?**
   a) Attempt to respond in Spanish itself.
   b) Ask the user to switch to English.
   ✅**c) Handoff to the spanish_agent.**
   d) Call a special translate tool.

---

10. **The Assistant agent is configured with `handoffs=[spanish_agent]`. What does this enable?**
   a) The Assistant agent can directly call spanish_agent's tools.
   b) The Assistant agent can receive handoffs from the spanish_agent.
   ✅**c) The Assistant agent can delegate tasks to the spanish_agent.**
   d) It creates a bidirectional communication link between them.

---

11. **What function from `agents.extensions.handoff_prompt` is used to enhance the agents' instructions?**
   a) add_handoff_instructions
   ✅**b) prompt_with_handoff_instructions**
   c) handoff_aware_prompt
   d) enrich_instructions_for_handoff

🔲 Part 3: Running the Voice Pipeline

12. **In the VoicePipeline setup, which specific workflow class is used with the agent instance?**
   a) MultiAgentVoiceWorkflow

b) VoiceAgentWorkflow
✓**c) SingleAgentVoiceWorkflow**
d) DefaultVoiceWorkflow

---

13. **For simplicity in the quickstart example, what kind of audio input is generated using `np.zeros`?**
a) A short spoken phrase.
✓**b) 3 seconds of silence.**
c) A random noise signal.
d) A pre-recorded song.

---

14. **What is the `samplerate` configured for the `sounddevice.OutputStream` player in the example?**
a) 8000 Hz
b) 16000 Hz
✓**c) 24000 Hz**
d) 44100 Hz

---

15. **When streaming the result from `pipeline.run()`, which event type signifies that actual audio data is being received for playback?**
a) voice_stream_event_text
b) voice_stream_event_start
✓**c) voice_stream_event_audio**
d) voice_stream_event_end

---

16. **What library is used in the example to handle asynchronous operations like running the pipeline and streaming results?**
a) threading
b) concurrent.futures
✓**c) asyncio**
d) multiprocessing

---

17. **If you wanted to test this voice quickstart example with actual microphone input, where would you find a demo for that?**
a) In the examples/voice/live directory.
b) In the examples/voice/microphone directory.
✓**c) In the examples/voice/static directory.**
d) In the examples/voice/interactive directory.

---

18. **What is the purpose of `set_tracing_disabled()` being called in the "Put it all together" section?**
a) To improve audio quality.
b) To reduce memory usage.
✓**c) To prevent potential errors if an OpenAI API key for tracing is not set.**
d) To switch to a different LLM model.

19. **The buffer for AudioInput is created with `dtype=np.int16`. What does this specify?**
   a) The number of audio channels.
   b) The duration of the audio.
   ✅**c) The data type of the audio samples (16-bit integers).**
   d) The sampling rate of the audio.

---

20. **In a real-world scenario, what would the buffer for AudioInput typically contain instead of silence?**
   a) Pre-recorded music.
   ✅**b) Data from a microphone.**
   c) Text from a user.
   d) Video frames.

Pipelines and Workflows

OpenAI Agents SDK - Pipelines and Workflows Section MCQs

▯ Part 1: VoicePipeline Overview and Configuration

1. **What is the primary function of the `VoicePipeline` class?**
   a) To manage agent memory.
   ✅**b) To facilitate turning agentic workflows into a voice application.**
   c) To visualize agent interactions.
   d) To handle LLM model configuration.

---

2. **Which of the following is *not* a responsibility of the VoicePipeline as described?**
   a) Transcribing input audio.
   b) Detecting when audio ends.
   ✅**c) Providing pre-built agent instructions.**
   d) Turning workflow output back into audio.

---

3. **When configuring a VoicePipeline, which of the following is a mandatory component to pass?**
   a) model provider
   b) tracing config
   ✅**c) workflow**
   d) prompt language

---

4. **The `config` parameter of VoicePipeline allows configuration of various aspects. Which of these is an example given for config settings related to tracing?**
   a) Agent name.
   b) Number of handoffs.
   ✅**c) Whether audio files are uploaded.**
   d) The tool definitions.

---

5. **Which specific setting within the VoicePipeline's config allows for mapping model names to actual model instances?**
   a) tts_model_settings
   b) stt_model_settings
   ✅**c) model_provider**
   d) workflow_config

� Part 2: Running the Pipeline and Handling Results

6. **Which method is used to initiate a voice pipeline run?**
   a) start()
   b) execute()
   ✅**c) run()**
   d) process_audio()

---

7. **Which AudioInput type is suitable when you have a complete audio transcript and do not need to detect when the speaker is done speaking?**
   ✅**a) AudioInput**
   b) StreamedAudioInput
   c) RecordedAudioInput
   d) BatchAudioInput

---

8. **For what kind of applications is `AudioInput` particularly useful?**
   a) Real-time conversational AI with interruptions.
   ✅**b) Push-to-talk apps where the user's speaking completion is clear.**
   c) Systems requiring continuous microphone listening.
   d) Applications focused on long-form dictation.

---

9. **Which AudioInput type should be used if the pipeline needs to automatically detect when a user is done speaking?**
   a) AudioInput
   ✅**b) StreamedAudioInput**
   c) ContinuousAudioInput
   d) ActivityDetectedAudioInput

---

10. **What is the process called by which the voice pipeline automatically runs the agent workflow at the right time when using `StreamedAudioInput`?**
    a) Voice activity detection (VAD).
    b) Turn-taking.
    c) Endpointing.
    ✓**d) Activity detection.**

---

11. **The result of a voice pipeline run (`pipeline.run(input)`) is an object of what type?**
    a) VoiceResult
    b) AudioOutput
    c) AgentResponse
    ✓**d) StreamedAudioResult**

---

12. **Which type of VoiceStreamEvent contains a chunk of audio that can be played back?**
    a) VoiceStreamEventText
    ✓**b) VoiceStreamEventAudio**
    c) VoiceStreamEventData
    d) VoiceStreamEventStream

---

13. **Which type of VoiceStreamEvent would inform you that a new turn in the conversation has started or ended?**
    a) VoiceStreamEventStatus
    b) VoiceStreamEventControl
    ✓**c) VoiceStreamEventLifecycle**
    d) VoiceStreamEventInfo

---

14. **What should you do when handling a VoiceStreamEventError?**
    a) Ignore it and continue playing audio.
    b) Restart the entire pipeline.
    ✓**c) Implement error handling logic (`elif event.type == "voice_stream_event_error"`).**
    d) Log it only to a file.

⊖Part 3: Best Practices – Interruptions

15. **Does the Agents SDK currently offer built-in interruption support for `StreamedAudioInput`?**
    a) Yes, it's fully supported.
    b) Only for specific models.
    ✓**c) No, it does not.**
    d) It's in alpha stage.

---

16. **If you want to handle interruptions within your application, which type of events should you listen for?**
    a) VoiceStreamEventAudio
    b) VoiceStreamEventError

✓**c) VoiceStreamEventLifecycle**
d) Any VoiceStreamEvent type

---

17. **Which specific `VoiceStreamEventLifecycle` event indicates that a new turn was transcribed and processing by the workflow is beginning?**
a) turn_completed
✓**b) turn_started**
c) audio_transcribed
d) workflow_beginning

---

18. **The `turn_ended` VoiceStreamEventLifecycle event triggers after what occurs?**
a) The user starts speaking again.
b) The workflow completes its processing.
✓**c) All the audio was dispatched for a respective turn.**
d) The speech-to-text model finishes.

---

19. **A suggested method to handle interruptions by muting/unmuting the microphone involves listening to which two `VoiceStreamEventLifecycle` events?**
a) turn_started and audio_sent
✓**b) turn_started and turn_ended**
c) transcription_complete and response_received
d) mic_on and mic_off

---

20. **What is the fundamental purpose of an "agentic workflow" that the VoicePipeline turns into a voice app?**
a) It's a predefined sequence of steps that always executes the same way.
✓**b) It's an AI-driven process where autonomous AI agents make decisions, take actions, and coordinate tasks.**
c) It's a simple script to perform speech-to-text and text-to-speech.
d) It's a static set of rules for handling conversational turns.

Voice Pipeline Tracing

OpenAI Agents SDK - Voice Pipeline Tracing MCQs

---

1. **How are voice pipelines traced in the Agents SDK?**
✓**b) They are automatically traced, similar to agents.**
✗a) They must be manually traced by the developer.
✗c) Tracing is disabled by default for voice pipelines.
✗d) Only errors in voice pipelines are traced.

---

2. **What class is specifically used to configure tracing for a VoicePipeline?**
   ✓**c) VoicePipelineConfig**
   ✗a) TracingConfig
   ✗b) VoiceTracingSettings
   ✗d) PipelineConfig

---

3. **By default, what is the status of tracing for voice pipelines?**
   ✓**c) Enabled.**
   ✗a) Disabled.
   ✗b) It depends on the global tracing setting.
   ✗d) It is only enabled in debug mode.

---

4. **Which field in VoicePipelineConfig controls whether tracing is enabled or disabled for the pipeline?**
   ✓**c) tracing_disabled**
   ✗a) enable_tracing
   ✗b) disable_tracing
   ✗d) trace_status

---

5. **The `trace_include_sensitive_data` field in `VoicePipelineConfig` specifically controls sensitive data in traces for which part of the system?**
   ✓**b) Data specifically from the voice pipeline (e.g., audio transcripts).**
   ✗a) Only data from within your agent workflow.
   ✗c) Data from external tools only.
   ✗d) All sensitive data across the entire SDK.

---

6. **Which field controls whether actual audio data is included in the voice pipeline traces?**
   ✓**d) trace_include_sensitive_audio_data**
   ✗a) trace_audio_only
   ✗b) include_audio_in_traces
   ✗c) trace_sensitive_audio

---

7. **What is the purpose of the `workflow_name` field in `VoicePipelineConfig`?**
   ✓**c) It sets the name of the trace workflow.**
   ✗a) It sets the name of the agent within the workflow.
   ✗b) It defines the input prompt for the speech-to-text model.
   ✗d) It specifies the audio file to be used.

---

8. **The `group_id` field in `VoicePipelineConfig` serves what function?**
   ✓**b) To link multiple traces (e.g., from the same conversation).**
   ✗a) To specify the group of agents involved in the pipeline.
   ✗c) To categorize pipelines based on their functionality.
   ✗d) To set permissions for trace access.

9. **If you want to add arbitrary key-value pairs to a voice pipeline trace, which field in `VoicePipelineConfig` should you use?**
✅**c) trace_metadata**
❌a) additional_data
❌b) custom_props
❌d) extra_info

---

10. **By default, will a voice pipeline trace include audio transcripts, assuming default VoicePipelineConfig settings?**
✅**c) Yes, as trace_include_sensitive_data defaults to True.**
❌a) No, sensitive data is disabled by default.
❌b) Yes, but only for errors.
❌d) It depends on an environment variable.

API Reference

Agents

OpenAI Agents SDK - Agents Module Functions MCQs

� OpenAI Agents SDK – Tracing & Key Management MCQs

---

1. **What is the primary purpose of `set_default_openai_key()`?**
✅**c) To set the OpenAI API key for LLM requests (and optionally tracing).**
❌a) To configure a specific LLM model.
❌b) To enable verbose logging.
❌d) To disable tracing globally.

---

2. **If the `OPENAI_API_KEY` environment variable is already set, what happens if `set_default_openai_key()` is called with a different key?**
✅**b) The key provided to `set_default_openai_key()` will be used instead.**
❌a) The environment variable takes precedence.
❌c) An error will be raised.
❌d) Both keys will be used interchangeably.

---

3. **By default, when `set_default_openai_key()` is called, is the provided key also used for tracing?**
✅**b) Yes, `use_for_tracing` defaults to True.**
❌a) No, tracing requires a separate configuration.
❌c) Only if the `OPENAI_API_KEY` environment variable is not set.
❌d) Only if verbose logging is enabled.

---

4. **Which function allows you to provide a custom `AsyncOpenAI` client instance to the SDK?**
✅**c) set_default_openai_client()**
❌a) configure_openai_client()
❌b) set_custom_openai_client()
❌d) override_openai_client()

---

5. **What is a common use case for providing a custom `AsyncOpenAI` client via `set_default_openai_client()`?**
✅**c) To connect to a different base_url for an OpenAI-compatible endpoint.**
❌a) To disable all tracing.
❌b) To change the default logging level.
❌d) To automatically retry failed LLM requests.

---

6. **By default, which OpenAI API does the Agents SDK use for LLM requests?**
✅**b) The Responses API.**
❌a) The Chat Completions API.
❌c) The Embeddings API.
❌d) The Moderation API.

---

7. **To switch the default OpenAI API for LLM requests to the Chat Completions API, which function would you use?**
✅**c) set_default_openai_api("chat_completions")**
❌a) use_chat_completions_api()
❌b) set_openai_api_type("chat_completions")
❌d) configure_api_version("chat_completions")

---

8. **If you want to use a specific API key only for sending traces to the backend, which function should you call?**
✅**c) set_tracing_export_api_key()**
❌a) set_default_openai_key(key, use_for_tracing=False)
❌b) set_tracing_key_only()
❌d) configure_trace_destination()

---

9. **What is the purpose of `set_tracing_disabled(disabled: bool)`?**
✅**d) To globally enable or disable tracing for the entire SDK.**
❌a) To disable specific types of traces.

✗b) To only disable tracing for a single agent.
✗c) To disable tracing for the OpenAI API calls.

---

10. **What argument should be passed to `set_tracing_disabled()` to turn off all tracing?**
✓**c) True**
✗a) "off"
✗b) 0
✗d) False

---

11. **For advanced customization, if you want to replace the SDK's default trace processing mechanism entirely, which function is used?**
✓**c) set_trace_processors()**
✗a) add_trace_processor()
✗b) register_trace_handler()
✗d) override_default_tracer()

---

12. **The `set_trace_processors()` function accepts a list of objects that implement which specific interface?**
✓**c) TracingProcessor**
✗a) TraceHandler
✗b) LogProcessor
✗d) TraceExporter

---

13. **What is the effect of calling `enable_verbose_stdout_logging()`?**
✓**c) It enables more detailed SDK logs (e.g., INFO, DEBUG) to be printed to stdout.**
✗a) It changes the default OpenAI API to Chat Completions.
✗b) It disables all tracing.
✗d) It sets a custom OpenAI API key.

---

14. **When would `enable_verbose_stdout_logging()` be most useful?**

✓**c) During development and debugging.**
✗a) In production environments to monitor system health.
✗b) When deploying the application to a server.
✗d) For generating final user reports.

---

15. **If `set_default_openai_key()` is called with `use_for_tracing=False`, and no other tracing API key is set, what will happen to tracing?**

✓**d) Tracing will attempt to use the `OPENAI_API_KEY` environment variable if available; otherwise, it will likely not export traces.**
✗a) Tracing will use a dummy key.

❌b) Tracing will automatically try to find a key from `OPENAI_API_KEY` environment variable.
❌c) Tracing will be entirely disabled.

Agents Module

OpenAI Agents SDK - Agents Module MCQs

✅Part 1: Core Agent Attributes and Behaviors

1. **What is the `instructions` attribute of an Agent primarily used for?**
   ✅**c) To act as the "system prompt" describing the agent's behavior and goal.**
   ❌a) To define the agent's name.
   ❌b) To specify the tools the agent can use.
   ❌d) To list the agents it can handoff to.

---

2. **If an Agent's `instructions` is set to a function, what must that function return?**
   ✅**c) A string.**
   ❌a) A Prompt object.
   ❌b) An Agent instance.
   ❌d) A list of tool names.

---

3. **The `handoff_description` attribute is used when an agent is included in which specific part of another agent's configuration?**
   ✅**b) Its `handoffs` list.**
   ❌a) Its tools list.
   ❌c) Its `mcp_servers` list.
   ❌d) Its `input_guardrails`.

---

4. **If an Agent's `model` attribute is not explicitly set, what model will it default to using?**
   ✅**b) The default model configured in `openai_provider.DEFAULT_MODEL` (currently "gpt-4o").**
   ❌a) gpt-3.5-turbo
   ❌c) A local, CPU-only model.
   ❌d) The last used model in the application.

---

5. **What is the purpose of the `model_settings` attribute?**
   ✅**b) To configure model-specific tuning parameters like temperature.**
   ❌a) To configure tracing options.
   ❌c) To set the API key for the model.
   ❌d) To define the model's supported output types.

---

6. **Which attribute defines a list of other Agent instances that the current agent can delegate tasks to?**
✓**c) `handoffs`**
✗a) `delegates`
✗b) `sub_agents`
✗d) `collaborators`

---

7. **What does `input_guardrails` specifically check, and under what condition do they run?**
✓**c) They check inputs before generating a response, running only if the agent is the first in the chain.**
✗a) They check the agent's final output, always running.
✗b) They check tool call results, running only if a tool is called.
✗d) They check agent instructions for compliance.

---

8. **The `output_type` attribute of an Agent defaults to `str` if not provided. What is a common way to customize it for structured output?**
✓**c) Passing a regular Python type like a dataclass or Pydantic model.**
✗a) Passing a list of strings.
✗b) Setting it to Any.
✗d) Using a Callable that returns the output.

---

9. **What is the default `tool_use_behavior` for an Agent?**
✓**c) `"run_llm_again"`**
✗a) "stop_on_first_tool"
✗b) A list of specific tool names.
✗d) A ToolsToFinalOutputFunction.

---

10. **If `tool_use_behavior` is set to `"stop_on_first_tool"`, what happens after the first tool call?**
✓**c) The output of the first tool call is used as the final output, and the LLM does not process it.**
✗a) The LLM processes the tool output and generates a new response.
✗b) The agent raises an error indicating immediate termination.
✗d) The agent proceeds to call all other available tools.

✓Part 2: Advanced Agent Configuration & Methods

11. **What is the primary purpose of MCP servers in the context of an Agent?**
✓**c) To dynamically provide tools to the agent.**
✗a) To manage agent authentication.
✗b) To provide a distributed computing environment.
✗d) To store conversation history.

---

12. **What crucial action must the user perform when using `mcp_servers` with an Agent?**
✓**c) Call `server.connect()` before passing them and `server.cleanup()` when done.**
✗a) Ensure they are all running on the same port.

✗b) Disable tracing for those servers.
✗d) Provide a `handoff_description` for each server.

---

13. **What does the `reset_tool_choice: bool = True` attribute aim to prevent?**
✓**c) The agent from entering an infinite loop of tool usage.**
✗a) The agent from using too many tools in a single turn.
✗b) The LLM from generating overly long responses.
✗d) Errors related to tool schema mismatches.

---

14. **How does the `clone()` method facilitate agent configuration?**
✓**c) It creates a copy of the agent with specified arguments changed.**
✗a) It creates an exact duplicate of the agent that runs in parallel.
✗b) It allows deep copying all mutable attributes to prevent side effects.
✗d) It saves the agent's current state to disk.

---

15. **When transforming an Agent into a Tool using `as_tool()`, what is a key difference in how the new agent receives input compared to a handoff?**
✓**c) It receives generated input (e.g., a specific query or data).**
✗a) It receives the full conversation history.
✗b) It receives a summary of the conversation.
✗d) It receives direct access to the calling agent's internal state.

---

16. **What is a key difference in control flow when an Agent is called via `as_tool()` versus being used as a handoff?**
✓**b) With `as_tool()`, the new agent returns control to the original calling agent.**
✗a) With `as_tool()`, the new agent takes over the conversation.
✗c) Handoffs are synchronous, while `as_tool()` calls are asynchronous.
✗d) `as_tool()` calls are always faster than handoffs.

---

17. **If an Agent has its `prompt` attribute set to a `DynamicPromptFunction`, what method would you call to get the resolved prompt during a run?**
✓**b) `get_prompt()`**
✗a) `get_resolved_prompt()`
✗c) `resolve_dynamic_prompt()`
✗d) `get_configured_prompt()`

---

18. **The `get_mcp_tools()` method allows an agent to access tools from what source?**
✓**b) From configured Model Context Protocol servers.**
✗a) Only from its directly configured tools list.
✗c) From a global tool registry.
✗d) From external API endpoints directly.

---

19. **Which attribute allows a class to receive callbacks on various lifecycle events of an agent?**
    ✓**c) `hooks`**
    ✗a) `event_listeners`
    ✗b) `callback_handlers`
    ✗d) `lifecycle_managers`

---

20. **If `output_type` is set to `AgentOutputSchema(MyClass, strict_json_schema=False)`, what does `strict_json_schema=False` enable?**
    ✓**c) Non-strict schema validation, allowing for more flexible JSON outputs.**
    ✗a) The agent to output any data type, ignoring MyClass.
    ✗b) A faster schema conversion process.
    ✗d) Automatic schema generation based on the LLM's response.

OpenAI Agents SDK - Runner MCQs

✓Part 1: Running Workflows (`run`, `run_sync`, `run_streamed`)

1. **What is the primary function of the Runner class?**
   ✓**b) To execute agent workflows.**
   ✗a) To configure OpenAI API keys.
   ✗c) To visualize agent graphs.
   ✗d) To manage trace processors.

---

2. **Which describes the loop mechanism of an agent workflow run?**
   ✓**a) Agent invoked → Tool calls → Handoff → Final output.**
   ✗b) Agent invoked → Final output → Handoff → Tool calls.
   ✗c) Final output → Agent invoked → Tool calls → Handoff.
   ✗d) Handoff → Tool calls → Agent invoked → Final output.

---

3. **When does the agent workflow loop terminate?**
   ✓**c) When the agent generates a final output (of its agent.output_type).**
   ✗a) When max_turns is reached.
   ✗b) When all available tools have been called.
   ✗d) When a handoff occurs.

---

4. **Which exception is raised if the max_turns limit is exceeded?**
   ✓**d) MaxTurnsExceeded**
   ✗a) TurnLimitExceeded
   ✗b) MaxIterationsReached
   ✗c) LoopLimitError

5. **What defines a "turn" in the context of max_turns?**
   ✓c) One AI invocation (including any tool calls that might occur).
   ✗a) Every time a tool is called.
   ✗b) Every time an agent receives new input.
   ✗d) One successful agent response.

6. **Which `run` method is asynchronous?**
   ✓a) `run`
   ✗b) `run_sync`
   ✗c) `run_streamed`
   ✗d) `run_async_only`

7. **When should `run_sync()` be avoided?**
   ✓c) If there's already an event loop (e.g., in an async function, Jupyter notebook, or FastAPI).
   ✗a) When using many tools.
   ✗b) When tracing is enabled.
   ✗d) When the agent has handoffs.

8. **Return type of `run_streamed()` method?**
   ✓c) `RunResultStreaming`
   ✗a) RunResult
   ✗b) StreamedResult
   ✗d) EventStream

9. **Purpose of `previous_response_id` argument in run methods?**
   ✓d) To skip passing in input from the previous turn when using OpenAI models via the Responses API.
   ✗a) To link traces to previous runs.
   ✗b) To enable continuous conversation history management.
   ✗c) To specify the previous agent in a handoff chain.

10. **What is unique about input guardrails in a workflow run?**
    ✓c) Only the first agent's input guardrails are run.
    ✗a) All agents in the chain run their input guardrails.
    ✗b) Input guardrails are only run after a tool call.
    ✗d) Input guardrails are ignored if max_turns is set.

✓Part 2: RunConfig - Global Settings

11. **Primary purpose of the `RunConfig` dataclass?**
    ✓c) To configure global settings for the entire agent run.
    ✗a) To configure individual agents.

✗b) To define agent instructions.
✗d) To manage environment variables.

---

12. **Effect of setting a model in `RunConfig` on agents?**
✓**b) It will override the model set on every agent within that run.**
✗a) It is ignored if an agent has its own model specified.
✗c) It only applies to agents that do not have a model attribute set.
✗d) It is only used for tracing purposes.

---

13. **Role of `model_provider` in `RunConfig`?**
✓**c) To resolve string model names to actual model implementations.**
✗a) To manage multiple agent instances concurrently.
✗b) To provide global tracing configurations.
✗d) To handle handoffs between different agents.

---

14. **Interaction between `RunConfig.model_settings` and agent-specific settings?**
✓**c) Any non-null values in RunConfig model_settings will override agent-specific settings.**
✗a) RunConfig model_settings are ignored if agent-specific settings exist.
✗b) RunConfig model_settings only apply to agents without specific settings.
✗d) They are merged, with agent-specific settings taking precedence for conflicts.

---

15. **Function of `handoff_input_filter` in `RunConfig`?**
✓**c) To apply a global filter to inputs sent to all handoffs.**
✗a) To block certain inputs from reaching the initial agent.
✗b) To filter the output of agents before a handoff.
✗d) To determine which agent to handoff to.

---

16. **Precedence when both `Handoff.input_filter` and `RunConfig.handoff_input_filter` are present?**
✓**c) `Handoff.input_filter`.**
✗a) `RunConfig.handoff_input_filter`.
✗b) They are both applied in sequence.
✗d) An error is raised due to conflict.

---

17. **Where do input_guardrails from RunConfig apply?**
✓**b) To the initial input of the entire run.**
✗a) To every agent's input in the workflow.
✗c) To inputs specifically for tools.
✗d) To inputs after a handoff.

---

18. **Where are output_guardrails from RunConfig applied?**
✓**c) The final output of the entire run.**

✗a) The output of every agent in the workflow.
✗b) The output of tool calls only.
✗d) The output of handoffs only.

✓Part 3: Tracing Configuration within RunConfig

19. **Default status of `tracing_disabled` in `RunConfig`?**
    ✓**b) False (tracing is enabled).**
    ✗a) True (tracing is disabled).
    ✗c) It depends on an environment variable.
    ✗d) It's ignored if trace_id is provided.

---

20. **Effect of `trace_include_sensitive_data=False`?**
    ✓**b) Spans will still be created, but sensitive data (like LLM inputs/outputs, tool data) will not be included.**
    ✗a) No spans will be created for sensitive events.
    ✗c) Tracing will be completely disabled.
    ✗d) Only error traces will be recorded.

---

21. **Default `workflow_name` if not specified in `RunConfig`?**
    ✓**b) 'Agent workflow'**
    ✗a) The name of the starting_agent.
    ✗c) None (no workflow name).
    ✗d) A random UUID.

---

22. **Purpose of providing `trace_id` in `RunConfig`?**
    ✓**b) To use a custom, pre-defined trace ID.**
    ✗a) To automatically generate a new trace ID.
    ✗c) To disable automatic trace ID generation.
    ✗d) To link runs to a specific user.

---

23. **Which attribute links multiple traces in a conversation/process?**
    ✓**c) `group_id`**
    ✗a) `trace_id`
    ✗b) `workflow_name`
    ✗d) `trace_metadata`

---

24. **What can be included in `trace_metadata`?**
    ✓**c) An optional dictionary of additional, arbitrary metadata.**
    ✗a) Only specific predefined metadata keys.
    ✗b) Sensitive data that is otherwise filtered.
    ✗d) Only model parameters and versions.

---

25. **Meaning of `GuardrailTripwireTriggered` exception?**
✓**c) A predefined guardrail condition has been violated.**
✗a) The agent has successfully completed its task.
✗b) The maximum number of turns has been reached.
✗d) An external API call has failed.

OpenAI Agents SDK - REPL Module MCQs

✓run_demo_loop Function – MCQs with Answers

1. **What is the primary purpose of the `run_demo_loop` function?**
✓**c) To provide a simple REPL loop for manual testing and debugging of an agent.**
✗a) To deploy agents to a production environment.
✗b) To generate automated test cases for agents.
✗d) To visualize agent workflow diagrams.

---

2. **What kind of interaction model does `run_demo_loop` facilitate?**
✓**c) A conversational, turn-based loop.**
✗a) One-shot query-response.
✗b) Batch processing of inputs.
✗d) Purely programmatic execution without user input.

---

3. **How is the conversation state handled across turns within `run_demo_loop`?**
✓**b) It is preserved across turns.**
✗a) It is reset after each turn.
✗c) It is saved to a file after each turn.
✗d) It is handled by the underlying LLM only, not the loop.

---

4. **Which of the following commands can be used to stop the `run_demo_loop`?**
✓**c) exit or quit**
✗a) stop or end
✗b) close or terminate
✗d) break or halt

---

5. **What type of argument must be passed as the `agent` parameter to `run_demo_loop`?**
✓**d) An instance of an Agent.**
✗a) A string representing the agent's name.
✗b) An AgentConfig object.
✗c) A function that returns an agent.

---

6. **By default, how does `run_demo_loop` display the agent's output?**
✓**c) It streams the agent output as it's generated.**

❌a) It waits for the full response and then prints it.
❌b) It saves the output to a log file.
❌d) It displays a summary only.

---

7. **What is the default value of the `stream` parameter in `run_demo_loop`?**
✅**b) True**
❌a) False
❌c) None
❌d) It’s a required parameter with no default.

---

8. **If `stream` is set to False in `run_demo_loop`, what is the expected behavior for displaying the agent's response?**
✅**c) The full response will be displayed after it is completely generated.**
❌a) The response will not be displayed.
❌b) The response will be displayed word by word.
❌d) Only the final line of the response will be shown.

---

9. **From which file path is the `run_demo_loop` function sourced?**
✅**c) src/agents/repl.py**
❌a) src/agents/main.py
❌b) src/agents/utils.py
❌d) src/agents/demo.py

---

10. **What is a primary benefit of `run_demo_loop` for developers working with agents?**
✅**c) It enables quick manual testing and interactive debugging from the command line.**
❌a) It automates unit testing.
❌b) It provides performance metrics.
❌d) It's a tool for deploying production agents.

OpenAI Agents SDK - Tools MCQs

✅Part 1: General Tool Concepts and `FunctionTool`

1. **What is the Tool module-attribute?**
✅**c) It’s a Union type defining all possible tool types.**
❌a) It’s an abstract base class for all tools.
❌b) It’s a list of all available tool instances.
❌d) It’s a factory for creating new tool instances.

---

2. **What does `FunctionToolResult` encapsulate?**
   ✓**b) The tool that was run, its output, and the RunItem produced.**
   ✗a) The JSON schema of the tool.
   ✗c) Only the FunctionTool name and description.
   ✗d) The success or failure status of the tool call.

---

3. **Recommended way to create a FunctionTool?**
   ✓**c) Use the `function_tool` helper.**
   ✗a) Directly instantiate `FunctionTool()`.
   ✗b) Use `Tool.create_function()`.
   ✗d) Load it from a configuration file.

---

4. **Which attribute shows the name to the LLM?**
   ✓**a) name**
   ✗b) description
   ✗c) params_json_schema
   ✗d) on_invoke_tool

---

5. **Purpose of the `description` attribute?**
   ✓**c) To describe the tool's purpose to the LLM.**
   ✗a) To provide internal documentation.
   ✗b) To specify the function's arguments.
   ✗d) To define the tool's return type.

---

6. **How does `on_invoke_tool` receive arguments?**
   ✓**c) A JSON string.**
   ✗a) A Python dictionary.
   ✗b) A list of strings.
   ✗d) A Pydantic model instance.

---

7. **What must `on_invoke_tool` return?**
   ✓**c) A string representation (or something `str()` can be called on).**
   ✗a) An integer.
   ✗b) A boolean.
   ✗d) A complex Python object.

---

8. **How can errors be communicated from `on_invoke_tool`?**
   ✓**b) By raising an Exception or returning a string error message.**
   ✗a) By returning None.
   ✗c) By setting a global error flag.
   ✗d) By calling `Runner.fail()`.

---

9. **Why is `strict_json_schema=True` recommended?**
   ✓**b) It increases the likelihood of correct JSON input from the LLM.**
   ✗a) It improves performance.
   ✗c) It makes the tool easier to debug.
   ✗d) It enables asynchronous execution.

---

10. **Purpose of `is_enabled` as a `Callable`?**
   ✓**c) To dynamically enable/disable the tool based on the run context and agent.**
   ✗a) To determine if the tool has been used previously.
   ✗b) To enable/disable based on user input.
   ✗d) To indicate if the tool is currently running.

✓Part 2: Specialized Hosted Tools

11. **Which tool allows LLM to search a vector store?**
   ✓**b) FileSearchTool**
   ✗a) WebSearchTool
   ✗c) CodeInterpreterTool
   ✗d) ImageGenerationTool

---

12. **FileSearchTool is supported with which API?**
   ✓**b) OpenAI models using the Responses API.**
   ✗a) All OpenAI models via standard API.
   ✗c) Locally hosted open-source models.
   ✗d) Any model with tool-calling capabilities.

---

13. **Which WebSearchTool attribute customizes geography?**
   ✓**b) user_location**
   ✗a) search_context_size
   ✗c) max_num_results
   ✗d) filters

---

14. **What describes the environment in `ComputerTool`?**
   ✓**b) computer**
   ✗a) shell_executor
   ✗c) system_interface
   ✗d) tool_config

---

15. **Role of `on_approval_request` in HostedMCPTool?**
   ✓**b) To provide a function for programmatic approval or rejection.**
   ✗a) To log all tool calls.
   ✗c) To automatically approve all MCP calls.
   ✗d) To notify user that tool call has finished.

16. **Return type of `MCPToolApprovalFunction`?**
    ✅**c) MCPToolApprovalFunctionResult**
    ❌a) bool
    ❌b) str
    ❌d) None

---

17. **Tool that executes code in a sandbox?**
    ✅**d) CodeInterpreterTool**
    ❌a) LocalShellTool
    ❌b) ComputerTool
    ❌c) HostedMCPTool

---

18. **Typical function for `executor` in LocalShellTool?**
    ✅**b) A function that executes a command on a shell.**
    ❌a) A function that generates a command string.
    ❌c) A function that parses shell output.
    ❌d) A function that validates shell commands.

✅Part 3: `function_tool` Helper and Error Handling

19. **What does `function_tool` parse automatically?**
    ✅**b) The JSON schema for the tool's parameters.**
    ❌a) The ToolContext type.
    ❌c) The agent's output_type.
    ❌d) The RunConfig settings.

---

20. **How is the tool's description populated by default?**
    ✅**c) From the function's docstring.**
    ❌a) From a YAML file.
    ❌b) From a default generic string.
    ❌d) From the function's name.

---

21. **If `failure_error_function=None` and call fails?**
    ✅**c) An Exception will be raised, causing the run to fail.**
    ❌a) A generic error message is sent to the LLM.
    ❌b) The tool will retry automatically.
    ❌d) The error will be silently ignored.

---

22. **Purpose of `use_docstring_info` in `function_tool`?**
    ✅**c) To control whether the docstring is used for tool/arg descriptions.**
    ❌a) To enable syntax highlighting.

❌b) To enforce docstring style.
❌d) To validate length.

---

23. **Requirement for `RunContextWrapper` in wrapped function?**
✅**c) It must match the context type (TContext) of the agent.**
❌a) It must be Any.
❌b) It must be None.
❌d) It can be any dataclass.

---

24. **Which is NOT a valid Tool type?**
✅**d) DatabaseQueryTool**
❌a) ImageGenerationTool
❌b) WebSearchTool
❌c) ComputerTool

---

25. **When is `reason` used in MCPToolApprovalFunctionResult?**
✅**b) When the tool call is rejected.**
❌a) When the tool call is approved.
❌c) After the call has completed.
❌d) As a general comment.

OpenAI Agents SDK - Results MCQs

✅Part 1: `RunResultBase` (Common Attributes and Methods)

1. **What does it mean that `RunResultBase` is an ABC?**
✅**c) It cannot be directly instantiated and defines an interface for its subclasses.**
❌a) It can be directly instantiated.
❌b) It provides concrete implementations.
❌d) It is a utility class.

---

2. **Which attribute contains all new messages, tool calls, and outputs?**
✅**b) new_items**
❌a) input
❌c) raw_responses
❌d) final_output

---

3. **What does `raw_responses` provide?**
✅**b) The raw, unfiltered LLM responses.**

✗a) Processed and filtered LLM outputs.
✗c) Only the final response.
✗d) A summary of LLM usage.

---

4. **The `final_output` attribute represents:**
✓**c) The output of the last agent in the workflow.**
✗a) The initial input.
✗b) Output of every agent.
✗d) A list of all intermediates.

---

5. **What is stored in `input_guardrail_results`?**
✓**c) Guardrail results for the initial input messages.**
✗a) Guardrails for tool inputs.
✗b) For handoff inputs.
✗d) For the final output.

---

6. **Meaning of `last_agent` being an abstractmethod?**
✓**c) Subclasses must implement this property.**
✗a) Provides a default agent.
✗b) It's optional.
✗d) Refers to the starting agent.

---

7. **Purpose of `last_response_id` property?**
✓**c) Get the ID of the last model response.**
✗a) ID of first model response.
✗b) Unique ID for the run.
✗d) User who started the run.

---

8. **Type checking in `final_output_as(cls)` by default:**
✓**b) Static typechecking only, no runtime check by default.**
✗a) Strict runtime checking.
✗c) Best-effort conversion.
✗d) Deep object validation.

---

9. **How to enforce runtime TypeError on mismatch in `final_output_as`?**
✓**b) raise_if_incorrect_type**
✗a) strict_mode
✗c) enforce_type
✗d) validate_output

---

10. **What does `to_input_list()` do?**
✓**c) Creates a new input list by merging the original input with all new items.**

❌a) Converts final output to input.
❌b) Filters based on guardrails.
❌d) Saves input to a file.

✅Part 2: `RunResult` and `RunResultStreaming`

11. **Class returned by `Runner.run()` (non-streamed)?**
    ✅**a) RunResult**
    ❌b) RunResultBase
    ❌c) RunResultStreaming
    ❌d) StreamEvent

---

12. **How does `RunResult` differ from `RunResultBase`?**
    ✅**c) Provides a concrete implementation for `last_agent`.**
    ❌a) Adds streaming.
    ❌b) More guardrail data.
    ❌d) Different I/O attributes.

---

13. **Main purpose of `RunResultStreaming`?**
    ✅**c) To stream semantic events during a run.**
    ❌a) Store final results.
    ❌b) Offline analysis.
    ❌d) Manage concurrent runs.

---

14. **Initial value of `final_output` in `RunResultStreaming`?**
    ✅**b) None**
    ❌a) ""
    ❌c) First generated output
    ❌d) Placeholder object

---

15. **When is `is_complete` True in streaming?**
    ✅**c) When the agent finishes and final output is produced.**
    ❌a) After first turn.
    ❌b) When stream_events is called.
    ❌d) When cancel is called.

---

16. **How to stop `RunResultStreaming` run?**
    ✅**c) Call the `cancel()` method.**
    ❌a) stop_stream()
    ❌b) SystemExit
    ❌d) Let max_turns exceed

---

17. **What is yielded by `stream_events()`?**
✓**c) `StreamEvent` objects with type field and data.**
✗a) Raw LLM chunks
✗b) Python dicts
✗d) Only error messages

---

18. **Exception if `max_turns` is exceeded in stream:**
✓**c) MaxTurnsExceeded**
✗a) TurnLimitExceededError
✗b) MaxIterationsReached
✗d) StreamingLimitError

---

19. **Exception raised if a guardrail is tripped?**
✓**c) GuardrailTripwireTriggered**
✗a) GuardrailViolation
✗b) SecurityAlert
✗d) PolicyViolationError

---

20. **When does `last_agent` hold true final agent (streaming)?**
✓**c) After `is_complete` is True.**
✗a) Immediately after `run_streamed()`.
✗b) After first StreamEvent.
✗d) After `cancel()` is called.

OpenAI Agents SDK - Streaming Events MCQs

✓Part: StreamEvent & Its Variants

1. **What is `StreamEvent` defined as?**
✓**b) A TypeAlias that can be one of RawResponsesStreamEvent, RunItemStreamEvent, or AgentUpdatedStreamEvent.**
✗a) Concrete class
✗c) Abstract base class
✗d) Generator function

---

2. **Which type of `StreamEvent` carries raw LLM deltas?**
✓**b) RawResponsesStreamEvent**
✗a) RunItemStreamEvent
✗c) AgentUpdatedStreamEvent
✗d) SemanticStreamEvent

3. **What is the `type` of `RawResponsesStreamEvent`?**
✅**c) 'raw_response_event'**
❌a) raw_event
❌b) llm_response
❌d) model_data

---

4. **What does a `RunItemStreamEvent` wrap?**
✅**c) Semantic events that wrap a RunItem.**
❌a) Raw LLM tokens
❌b) Debugging logs
❌d) System updates

---

5. **Which attribute classifies a `RunItemStreamEvent` action?**
✅**b) name**
❌a) type
❌c) data
❌d) status

---

6. **Which is NOT a valid `RunItemStreamEvent.name`?**
✅**d) "run_completed"**
❌a) message_output_created
❌b) tool_called
❌c) handoff_occured ✅(Valid spelling is with double "r", i.e., "handoff_occurred")

---

7. **What does `item` in `RunItemStreamEvent` hold?**
✅**c) The actual RunItem object that was created.**
❌a) Raw LLM output
❌b) Previous event
❌d) String description

---

8. **Purpose of `AgentUpdatedStreamEvent`?**
✅**c) Indicates a new agent is running (after a handoff).**
❌a) Agent state changed
❌b) New tool added
❌d) Agent completion

---

9. **What's in `new_agent` of `AgentUpdatedStreamEvent`?**
✅**c) The new active `Agent` instance.**
❌a) Previous agent name
❌b) Agent identifier
❌d) Config dict

10. **Which attribute to check first in a stream event?**
    ✅**b) type**
    ❌a) name
    ❌c) item
    ❌d) data

---

11. **If `name = "tool_output"` in `RunItemStreamEvent`, what's in `item`?**
    ✅**c) Output from a tool call.**
    ❌a) Message object
    ❌b) Handoff request
    ❌d) Tool definition

---

12. **Primary use of `RawResponsesStreamEvent`?**
    ✅**b) Access raw LLM tokens/chunks.**
    ❌a) Debug logic
    ❌c) Track handoffs
    ❌d) Guardrail monitoring

---

13. **What event when an agent uses a tool (e.g. web search)?**
    ✅**c) RunItemStreamEvent (name="tool_called")**
    ❌a) AgentUpdatedStreamEvent
    ❌b) RawResponsesStreamEvent
    ❌d) ToolExecutionEvent ❌(Not a real type)

---

14. **What does `AgentUpdatedStreamEvent` imply?**
    ✅**c) Likely a handoff occurred.**
    ❌a) Run completed
    ❌b) Agent error
    ❌d) New input to same agent

---

15. **Difference: `data` (RawResponsesStreamEvent) vs. `item` (RunItemStreamEvent)?**
    ✅**c) data = raw LLM stream, item = structured semantic action.**
    ❌a) data = parsed, item = raw
    ❌b) data = string, item = dict
    ❌d) Interchangeable ❌(Not true)

OpenAI Agents SDK - Handoffs MCQs

✅Part 1: Handoff Concepts and Data Structures

1. **What is the primary purpose of a "handoff" in the Agents SDK?**
   ✅**b) To delegate a task or conversation to another agent.**
   ❌a) To save the agent's state to a database.
   ❌c) To restart the current agent.
   ❌d) To synchronize multiple agent instances.

2. **Expected input/output for a HandoffInputFilter function:**
   ✅**c) Takes HandoffInputData, returns HandoffInputData.**
   ❌a) Takes str, returns str.
   ❌b) Takes list[RunItem], returns list[RunItem].
   ❌d) Takes Agent, returns Agent.

3. **Attribute containing pre-run conversation history:**
   ✅**b) input_history**
   ❌a) new_items
   ❌c) pre_handoff_items
   ❌d) current_turn_items

4. **What does new_items include in HandoffInputData?**
   ✅**c) Items generated during the current agent turn, including handoff-related items.**
   ❌a) Only the final output of the previous agent.
   ❌b) The entire conversation history.
   ❌d) Only raw LLM responses.

5. **What RunItem typically triggers a handoff?**
   ✅**c) The documentation implies a specific RunItem related to the handoff being invoked.**
   ❌a) message_output_created
   ❌b) tool_output
   ❌d) reasoning_item_created

✅Part 2: Handoff Dataclass Attributes and Behavior

6. **Purpose of tool_name and tool_description in Handoff:**
   ✅**c) To allow the LLM to recognize and decide when to "call" this handoff as a tool.**
   ❌a) For internal logging
   ❌b) To identify the agent being handed off to
   ❌d) To generate unique IDs

7. **What must on_invoke_handoff return?**
   ✅**c) An instance of Agent[TContext]**

❌a) A boolean
❌b) A string
❌d) RunContextWrapper

8. **Precedence between input_filter in RunConfig and Handoff:**
   ✅**c) Handoff.input_filter**
   ❌a) RunConfig
   ❌b) Both sequentially
   ❌d) System error

9. **Effect of modifying input history via input_filter during streaming:**
   ✅**c) Nothing will be streamed as a result of this function; items already streamed.**
   ❌a) All events re-streamed
   ❌b) Next turn only
   ❌d) InputFilteredEvent sent

10. **Why strict_json_schema=True is recommended for Handoff:**
    ✅**c) Increases likelihood of correct JSON from LLM**
    ❌a) Makes handoffs faster
    ❌b) Allows more flexibility
    ❌d) Disables input validation

11. **What does input_json_schema define?**
    ✅**c) The JSON schema for inputs to the next agent during handoff.**
    ❌a) Internal state
    ❌b) Final output
    ❌d) RunContextWrapper

✅Part 3: handoff Helper Function

12. **Primary role of handoff helper:**
    ✅**c) To simplify the creation of Handoff objects**
    ❌a) Executes handoff immediately
    ❌b) Registers with a registry
    ❌d) Manages approval

13. **What can the agent parameter also be?**
    ✅**c) A function that returns an agent**
    ❌a) Agent name (str)
    ❌b) Config dictionary
    ❌d) List of agents

14. **Effect of on_handoff on schema and description:**
✅**c) Signature and docstring can auto-generate them**
❌a) Must set manually
❌b) Overrides with defaults
❌d) Prevents generation

15. **Purpose of input_type parameter:**
✅**c) Defines the expected input type for on_handoff**
❌a) Return type
❌b) HandoffInputData
❌d) Agent type

16. **True statement about on_handoff and input_type:**
✅**c) input_type is only relevant if on_handoff takes input**
❌a) on_handoff always required
❌b) input_type always required
❌d) on_handoff only works with input_type=None

17. **What does tool_name_override change?**
✅**c) The name of the "tool" used by LLM for handoff**
❌a) Agent name
❌b) Default name from helper
❌d) Function name

18. **Best use-case for handoffs:**
✅**c) Multi-agent system solving complex problems**
❌a) Single simple task
❌b) Repetitive calculations
❌d) Single API interaction

19. **What does handoff_input_data.all_items contain?**
✅**c) input_history + pre_handoff_items + new_items (maybe filtered)**
❌a) input_history only
❌b) new_items only
❌d) Only raw LLM responses

20. **What does it mean that handoff can be used as a decorator?**
✅**c) You can use @handoff(...) on a function to create the handoff**
❌a) Must call with ()
❌b) Cannot take parameters
❌d) Returns a boolean

OpenAI Agents SDK - Lifecycle MCQs

✅Part 1: RunHooks (Global Run Lifecycle)

1. **Primary purpose of RunHooks:**
   ✅**b) To receive callbacks on various lifecycle events for an entire agent run.**
   ❌a) Manage config
   ❌c) Define state
   ❌d) Handle tool errors

2. **When is on_agent_start called?**
   ✅**c) Before any agent is invoked, and each time the current active agent changes.**
   ❌a) Only once at start
   ❌b) After agent task
   ❌d) When a tool is called

3. **on_agent_end signals what?**
   ✅**b) Agent produced a final output**
   ❌a) Agent started
   ❌c) About to be garbage collected
   ❌d) Agent error

4. **on_handoff agent parameters:**
   ✅**c) from_agent, to_agent**
   ❌a) current_agent, next_agent
   ❌b) initiator_agent, receiver_agent
   ❌d) primary_agent, secondary_agent

5. **When is on_tool_start called?**
   ✅**b) Before any tool is invoked by any agent in the run.**
   ❌a) After tool result
   ❌c) FunctionTool only
   ❌d) Agent requests info

6. **What does on_tool_end provide that on_tool_start doesn't?**
   ✅**c) The result (output) of the tool's execution.**
   ❌a) Agent
   ❌b) Description
   ❌d) Context

7. **To register global lifecycle callbacks, subclass:**
   ✅**b) RunHooks**
   ❌a) AgentHooks
   ❌c) GlobalHooks
   ❌d) WorkflowHooks

8. **Implication of async RunHooks methods:**
   ✅**c) They should be defined using `async def` and can use `await`.**
   ❌a) Must return None
   ❌b) Execute synchronously
   ❌d) Run in separate process

✅Part 2: AgentHooks (Specific Agent Lifecycle)

9. **Main difference between AgentHooks vs RunHooks:**
   ✅**c) AgentHooks = specific agent callbacks, RunHooks = global run callbacks.**
   ❌a) Debug vs production
   ❌b) Sync vs async
   ❌d) Single-turn vs multi-turn

10. **How to register AgentHooks:**
    ✅**c) Assign to `agent.hooks` attribute.**
    ❌a) Via `Runner.run()`
    ❌b) Global config
    ❌d) Override register

11. **When is AgentHooks.on_start called?**
    ✅**c) Before the agent becomes active each time.**
    ❌a) Start of run
    ❌b) After turn
    ❌d) On tool call

12. **What does `source` refer to in AgentHooks.on_handoff?**
    ✅**c) The agent handing off to this agent.**
    ❌a) Target agent
    ❌b) Handoff output
    ❌d) Triggering tool

13. **Correct on_handoff sequence when Agent A → Agent B:**
    ✅**a) RunHooks.on_handoff(from_agent=A, to_agent=B) AND AgentHooks(B).on_handoff(agent=B, source=A).**
    ❌b), c), d) are incorrect variations.

14. **If a tool is invoked by an agent with AgentHooks:**
    ✅**c) Both RunHooks.on_tool_start and AgentHooks.on_tool_start are called.**
    ❌a), b), d) miss one of them.

15. **When is AgentHooks.on_end called?**
    ✅**b) When this agent produces a final output.**
    ❌a) Reasoning step
    ❌c) Entire run ends
    ❌d) Agent deactivation

✅Part 3: General Lifecycle Concepts

16. **Key benefit of lifecycle hooks:**
    ✅**c) They enable custom logging, monitoring, and behavior injection.**
    ❌a) Auto-optimize performance
    ❌b) Replace code
    ❌d) Built-in UI tools

17. **What does TContext represent?**
    ✅**c) The context type for the agent run.**
    ❌a) Agent type
    ❌b) Tool type
    ❌d) StreamEvent type

18. **To log every tool call in a multi-agent system:**
    ✅**b) RunHooks.on_tool_start**
    ❌a) Per-agent is harder
    ❌c), d) are unrelated

19. **Custom setup when a specific agent becomes active:**
    ✅**b) AgentHooks.on_start (on that agent)**
    ❌a) Global start
    ❌c) Handoff tracking
    ❌d) Tool calls

20. **What happens if you use lifecycle methods without async?**
    ✅**c) Runtime errors or blocking behavior may occur.**
    ❌a) Auto-wrapped
    ❌b) Run in thread
    ❌d) Silently ignored

OpenAI Agents SDK - Items MCQs

✅❌Part 1: Type Aliases and RunItem Overview

1. **What does TResponse serve as a type alias for?**
   a) RunResult ✕
   b) AgentResponse ✕
   ✅**c) Response from the OpenAI SDK**
   d) StreamEvent ✕

2. **Which TypeAlias represents any semantic item generated by an agent?**
   a) ToolCallItemTypes ✕
   b) TResponseOutputItem ✕
   ✅**c) RunItem**
   d) ModelResponse ✕

3. **ToolCallItemTypes specifically represents:**
   a) Any type of agent message ✕
   b) Any type of handoff event ✕
   ✅**c) Any type of raw tool call item (function, computer, file search, MCP, image, shell)**
   d) Any type of agent output ✕

4. **Which of the following is NOT a type of RunItem?**
   a) ReasoningItem ✕
   b) HandoffOutputItem ✕
   c) MCPApprovalRequestItem ✕
   ✅**d) InputMessageItem**

5. **The purpose of TResponseInputItem is to:**
   a) Represent items returned by the model ✕
   ✅**b) Represent input parameters for the model**
   c) Represent items from ItemHelpers ✕
   d) Represent streaming output ✕

✅❌Part 2: RunItemBase and Subclasses

6. **In RunItemBase[T], what does T represent?**
   a) The agent that generated the item ✕
   b) The tool type ✕

✅**c) The type of the raw_item**
d) RunContextWrapper ❌

7. **Which attribute is common to all subclasses of RunItemBase?**
a) output ❌
✅**b) agent**
c) source_agent ❌
d) name ❌

8. **raw_item attribute in RunItemBase is always:**
a) StreamEvent or ModelResponse ❌
b) ToolCallItemTypes or HandoffCallItem ❌
✅**c) ResponseOutputItem or ResponseInputItemParam**
d) Agent or Tool ❌

9. **RunItem subclass for messages directly from LLM:**
a) ReasoningItem ❌
b) ToolCallOutputItem ❌
✅**c) MessageOutputItem**
d) HandoffCallItem ❌

10. **Unique attributes of HandoffOutputItem:**
a) output and reasoning ❌
b) tool_name and tool_description ❌
✅**c) source_agent and target_agent**
d) input_history and new_items ❌

11. **raw_item type within a HandoffCallItem:**
a) McpCall ❌
b) TResponseInputItem ❌
✅**c) ResponseFunctionToolCall**
d) ResponseComputerToolCall ❌

12. **ToolCallItem raw_item is ToolCallItemTypes, meaning:**
a) Only FunctionTool calls ❌
✅**b) Can be function, shell, computer, etc.**
c) Always tool output ❌
d) No raw data ❌

13. **RunItem subclass with processed tool output separate from raw_item:**
a) HandoffOutputItem ❌
b) MessageOutputItem ❌

c) ReasoningItem ✗
✓**d) ToolCallOutputItem**

14. **What does ReasoningItem represent?**
a) A completed task ✗
b) An error message ✗
✓**c) Agent's internal thought step**
d) Request for more input ✗

15. **MCPApprovalRequestItem is for:**
a) General human input ✗
b) Approval of regular tools ✗
✓**c) MCP-specific tool approval**
d) Reply to previous request ✗

✓✗Part 3: ModelResponse and ItemHelpers

16. **ModelResponse.output contains:**
a) Final string output ✗
b) One token ✗
✓**c) List of TResponseOutputItems**
d) Reasoning steps ✗

17. **Purpose of response_id:**
a) Identifies agent ✗
b) Tracks turn number ✗
✓**c) ID for reusing in future model calls**
d) Model name ✗

18. **to_input_items() does what?**
a) Converts to string ✗
b) Filters tool calls ✗
✓**c) Converts output into valid input items**
d) Saves response ✗

19. **Method to extract last content or refusal:**
✓**a) extract_last_content**
b) extract_last_text ✗
c) text_message_output ✗
d) get_message_text ✗

20. **Method to get only text (ignore refusal):**
a) extract_last_content ❌
✅**b) extract_last_text**
c) text_message_outputs ❌
d) tool_call_output_item ❌

21. **Return type of text_message_output():**
a) list[str] ❌
b) MessageOutputItem ❌
✅**c) str**
d) None ❌

22. **tool_call_output_item() creates:**
a) Executes a tool ❌
b) Parses tool call ❌
✅**c) Creates raw tool call output with string result**
d) Gets tool name ❌

23. **Normalize inputs into TResponseInputItems list:**
a) extract_last_text ❌
b) text_message_outputs ❌
✅**c) input_to_new_input_list**
d) to_input_items ❌

24. **Primary purpose of Items module:**
a) Database handling ❌
b) UI rendering ❌
✅**c) Standardize agent data and events**
d) Define agent logic ❌

25. **To inspect messages/tool calls in RunResult:**
a) raw_responses ❌
b) final_output ❌
✅**c) new_items**
d) input_guardrail_results ❌

OpenAI Agents SDK - Run Context MCQs

✅❌RunContextWrapper Quiz Answers

1. **What is the primary purpose of RunContextWrapper?**
a) To manage LLM model configurations ❌
b) To store agent conversation history for the LLM ❌

✅**c) To wrap a custom context object and track run usage, accessible by user-implemented code**
d) To define the agent's response format ❌

2. **Which type parameter does RunContextWrapper use to define the type of the custom context object?**
   a) TAgent ❌
   b) TResult ❌
   ✅**c) TContext**
   d) TInput ❌

3. **Is the context object within RunContextWrapper passed to the LLM as part of the prompt?**
   a) Yes, always ❌
   b) Yes, but only if it's a string ❌
   ✅**c) No, contexts are explicitly not passed to the LLM**
   d) Only in streaming mode ❌

4. **Which attribute of RunContextWrapper is used to track resource consumption like token counts?**
   a) context ❌
   b) response_id ❌
   ✅**c) usage**
   d) output ❌

5. **When using RunResultStreaming, what is true about the usage attribute of RunContextWrapper?**
   a) It is always perfectly up-to-date ❌
   b) It is reset to zero after each stream event ❌
   ✅**c) It will be stale until the last chunk of the stream is processed**
   d) It only tracks input tokens, not output tokens ❌

6. **How is the usage attribute typically initialized if not explicitly provided during the run setup?**
   a) It starts as None ❌
   ✅**b) Using a default_factory to create a new Usage object**
   c) It is a static attribute shared across all runs ❌
   d) It is lazily loaded only when accessed ❌

7. **To what parts of your agent application can the context object within RunContextWrapper provide data or dependencies?**
   a) Only to the Runner itself ❌
   b) Only to the LLM ❌
   ✅**c) Tool functions, callbacks, hooks, and other user-implemented code**
   d) Only for logging purposes ❌

8. **If you want to pass a database connection object to all your agent's tools, where would you typically store this connection?**

a) Directly in the agent's configuration ✕
b) As a global variable ✕
✓**c) Within the custom context object passed to Runner.run(), accessible via RunContextWrapper.context**
d) In environment variables ✕

9. **What happens if you don't provide a custom context object to Runner.run()?**
a) The run will fail ✕
✓**b) The context attribute of RunContextWrapper will likely be None (or its default if TContext allows it)**
c) A default context object will be automatically generated with random data ✕
d) usage tracking will be disabled ✕

10. **The RunContextWrapper helps to adhere to which software design principle by providing a dedicated object for dependencies?**
a) High Coupling ✕
b) Data Duplication ✕
✓**c) Dependency Injection**
d) Global State Management ✕

OpenAI Agents SDK - Usage MCQs

✓✕Usage Dataclass Quiz Answers

1. **What is the primary purpose of the Usage dataclass?**
a) To manage agent internal state ✕
b) To store conversation history ✕
✓**c) To track resource consumption (like LLM requests and tokens) during an agent run**
d) To configure agent behavior ✕

2. **The requests attribute in Usage counts:**
a) The number of user inputs ✕
b) The number of handoffs between agents ✕
✓**c) The total number of API requests made to the LLM**
d) The number of tools executed ✕

3. **Which attribute in Usage specifically tracks the total tokens sent to the LLM?**
a) output_tokens ✕
✓**b) input_tokens**
c) total_tokens ✕
d) requests ✕

4. **The input_tokens_details attribute provides:**
   a) A summary of output tokens ✗
   ✓**b) More granular information about input tokens, such as cached tokens**
   c) Details about the LLM model used ✗
   d) The timestamp of the last input ✗

5. **What kind of information might be found within output_tokens_details?**
   a) The latency of the LLM response ✗
   b) The number of API errors ✗
   ✓**c) Details about output tokens, potentially including reasoning_tokens**
   d) The number of tools called by the agent ✗

6. **The total_tokens attribute is calculated as the sum of:**
   a) requests and input_tokens ✗
   ✓**b) input_tokens and output_tokens**
   c) requests and output_tokens ✗
   d) cached_tokens and reasoning_tokens ✗

7. **By default, what is the initial value for requests, input_tokens, output_tokens, and total_tokens when a Usage object is created?**
   a) None ✗
   b) 1 ✗
   ✓**c) 0**
   d) field(default_factory=...) ✗

8. **Where would you typically find an instance of the Usage dataclass during an agent run?**
   a) Directly on the Agent object ✗
   b) As a global variable in the SDK ✗
   ✓**c) As an attribute of RunContextWrapper**
   d) Within the StreamEvent object ✗

9. **Why might the usage metrics be "stale" when dealing with streamed responses?**
   a) Because streaming is only for output, not input ✗
   b) Because the agent might be paused ✗
   ✓**c) Because the full usage can only be calculated once all chunks of the stream have been processed**
   d) Because the LLM does not provide usage details during streaming ✗

10. **Which attribute would you check to understand how many tokens were used for an agent's internal thought processes if the model distinguishes them?**
    a) input_tokens ✗
    b) total_tokens ✗
    c) requests ✗
    ✓**d) output_tokens_details.reasoning_tokens (assuming OutputTokensDetails contains this field)**

OpenAI Agents SDK - Exceptions MCQs

✅❌Agents SDK – Exception Handling Quiz Answers

1. **What is the base class for all exceptions within the OpenAI Agents SDK?**
   a) BaseException ❌
   b) PythonException ❌
   ✅**c) AgentsException**
   d) SDKError ❌

2. **What is the primary role of RunErrorDetails?**
   a) To define new types of exceptions ❌
   b) To handle exceptions automatically ❌
   ✅**c) To collect diagnostic data when an exception occurs during an agent run**
   d) To log successful agent runs ❌

3. **When is a MaxTurnsExceeded exception raised?**
   a) When the agent takes too long to respond ❌
   ✅**b) When the agent run exceeds a predefined maximum number of turns (iterations)**
   c) When the LLM generates a very long output ❌
   d) When a tool call takes too many turns ❌

4. **Which exception indicates that the underlying LLM has behaved in an unexpected or invalid way, such as calling a non-existent tool?**
   a) UserError ❌
   b) AgentsException ❌
   ✅**c) ModelBehaviorError**
   d) InternalError ❌

5. **If the LLM provides malformed JSON when attempting to call a tool, which exception is most likely to be raised?**
   a) UserError ❌
   ✅**b) ModelBehaviorError**
   c) InputGuardrailTripwireTriggered ❌
   d) MaxTurnsExceeded ❌

6. **A UserError is raised when:**
   a) The agent cannot understand the user's input ❌
   b) The LLM generates an incorrect response ❌
   ✅**c) The developer using the SDK has made an error in their code or configuration**
   d) An external API returns an error ❌

7. **Which exception is specifically associated with a safety mechanism that monitors content entering the agent's processing flow?**
   a) OutputGuardrailTripwireTriggered ❌
   ✅**b) InputGuardrailTripwireTriggered**
   c) ModelBehaviorError ❌
   d) UserError ❌
8. **Both InputGuardrailTripwireTriggered and OutputGuardrailTripwireTriggered exceptions have a common attribute providing details about the triggered guardrail. What is this attribute called?**
   a) error_message ❌
   b) trigger_details ❌
   ✅**c) guardrail_result**
   d) policy_violation ❌

9. **When an agent's generated response violates a predefined safety or policy guideline, which exception would be raised?**
   a) InputGuardrailTripwireTriggered ❌
   b) ModelBehaviorError ❌
   ✅**c) OutputGuardrailTripwireTriggered**
   d) AgentResponseError ❌

10. **What is a key benefit of having a specific exception hierarchy like AgentsException and its subclasses?**
   a) It makes the SDK faster ❌
   b) It allows for automatic self-correction of agents ❌
   ✅**c) It enables developers to catch and handle distinct categories of errors more effectively**
   d) It reduces the number of tokens consumed ❌

11. **If you wanted to catch any exception that originates specifically from the OpenAI Agents SDK, but not general Python exceptions, which exception type would you catch?**
   a) Exception ❌
   ✅**b) AgentsException**
   c) ModelBehaviorError ❌
   d) UserError ❌

12. **An agent is configured with max_turns=5. On its 6th attempt to get a final answer, it is still thinking. What exception will be raised?**
   a) ModelBehaviorError ❌
   b) UserError ❌
   ✅**c) MaxTurnsExceeded**
   d) InputGuardrailTripwireTriggered ❌

13. **What type of issue would most likely lead to a UserError?**
   a) The LLM providing a refusal message ❌
   ✅**b) A typo in the name of a tool provided in the agent's configuration**
   c) An external API being down ❌
   d) The agent generating a very long message ❌

14. **The guardrail_result attribute is important for debugging guardrail-related exceptions because it provides:**
    a) The recommended next step for the agent ✗
    b) The time the guardrail was activated ✗
    ✓**c) Detailed data about the specific guardrail triggered and the reason for the violation**
    d) A list of all available guardrails ✗

15. **If InputGuardrailTripwireTriggered occurs, what does it primarily prevent?**
    a) The agent from making tool calls ✗
    ✓**b) Potentially problematic content from reaching the agent's processing logic**
    c) The agent from generating a final response ✗
    d) The LLM from receiving any input ✗

OpenAI Agents SDK - Guardrails MCQs

Part 1: Guardrail Concepts and Output

1. **Primary purpose of guardrails?**
   a) To enhance LLM's reasoning capabilities ✗
   ✓**b) To implement safety, validation, and control checks on agent inputs and outputs**
   c) To optimize agent response time ✗
   d) To manage external tool integrations ✗

2. **Significance of `tripwire_triggered: bool`?**
   a) Indicates the guardrail performed a basic check ✗
   ✓**b) Means a serious violation occurred; agent execution halted**
   c) Signifies successful operation ✗
   d) Optional field ✗

3. **`output_info` in `GuardrailFunctionOutput` is:**
   a) Main output of the agent ✗
   ✓**b) Optional info about guardrail's findings**
   c) Mandatory success/failure string ✗
   d) The input that triggered the guardrail ✗

4. **Which dataclass represents the result of InputGuardrail run?**
   a) GuardrailFunctionOutput ✗
   b) OutputGuardrailResult ✗
   ✓**c) InputGuardrailResult**
   d) GuardrailStatus ✗

Part 2: InputGuardrail and OutputGuardrail

5. **InputGuardrails check:**
   a) Agent's final output ✗
   ✓**b) Input messages or data**
   c) Agent internal state ✗
   d) Tool call results ✗

6. **If `tripwire_triggered` is True in InputGuardrail, what is raised?**
   a) OutputGuardrailTripwireTriggered ✗
   b) ModelBehaviorError ✗
   ✓**c) InputGuardrailTripwireTriggered**
   d) MaxTurnsExceeded ✗

7. **Use case for InputGuardrail?**
   a) Validate final response ✗
   b) Ensure tool returns JSON ✗
   ✓**c) Detect off-topic user input**
   d) Check LLM token usage ✗

8. **OutputGuardrails run on:**
   a) Input received ✗
   b) Intermediate reasoning ✗
   ✓**c) Final agent output**
   d) Tool call parameters ✗

9. **Extra attributes in `OutputGuardrailResult` vs `InputGuardrailResult`?**
   a) guardrail_function and name ✗
   b) tripwire_triggered and output_info ✗
   ✓**c) agent_output and agent**
   d) input_history and new_items ✗

10. **Params for `guardrail_function` in OutputGuardrail:**
    a) (context, agent, input_message) ✗
    ✓**b) (context, agent, agent_output)**
    c) (output) ✗
    d) (tool, result) ✗

11. **Purpose of `name` attribute in Guardrails:**
    a) LLM name ✗
    ✓**b) For tracing and logging**
    c) Agent owner name ✗
    d) Guardrail type ✗

Part 3: Decorators and Usage

12. **Benefit of `@input_guardrail()` / `@output_guardrail()`?**
    a) Auto-fix violations ✗
    ✓**b) Simplify creation of Guardrail instances from functions**
    c) Bypass checks ✗
    d) Only for sync functions ✗

13. **Can `@input_guardrail` function be async?**
    ✓**a) Yes, both sync and async are supported**
    b) No ✗
    c) Only if it doesn't return output ✗
    d) Only with MaybeAwaitable ✗

14. **Turn function `check_content_safety` into OutputGuardrail?**
    a) OutputGuardrail(func=...) ✗
    b) output_guardrail_func(...) ✗
    ✓**c) `@output_guardrail` above function definition**
    d) check_content_safety.as_guardrail() ✗

15. **Use `@input_guardrail(name="...")` — how must decorator be used?**
    a) Without parentheses ✗
    ✓**b) With parentheses**
    c) Not possible ✗
    d) Must be below function ✗

16. **What happens if OutputGuardrail returns `tripwire_triggered=True`?**
    a) Run continues ✗
    b) Auto-rephrase ✗
    ✓**c) OutputGuardrailTripwireTriggered is raised**
    d) Human review ✗

17. **InputGuardrail receives `str | list[TResponseInputItem]`. Meaning?**
    a) Only simple text ✗
    ✓**b) Can process string or list of OpenAI-like input items**
    c) List of RunItem ✗
    d) Only RunContextWrapper ✗

18. **`agent_output` in OutputGuardrailResult is:**
    a) Input to agent ✗
    ✓**b) Final output checked by the guardrail**

c) Raw LLM response ❌
d) Agent summary ❌

19. **Why InputGuardrail runs “in parallel”?**
a) To increase tokens ❌
b) Instant LLM feedback ❌
✅**c) Early intervention before resource waste**
d) Real-time stream ❌

20. **Primary difference between Input vs Output Guardrails?**
a) Prevent errors vs ensure performance ❌
✅**b) Input prevents bad content entering; output prevents bad content leaving**
c) Input = tool, Output = message ❌
d) Input = dev-defined, Output = model-defined ❌

OpenAI Agents SDK - Model Settings MCQs

Part 1: Core Parameters & Behavior

1. **Primary function of `ModelSettings`?**
a) Define agent name ❌
b) Manage API keys ❌
✅**c) Hold optional config for calling an LLM**
d) Store conversation history ❌

2. **Controls LLM randomness/creativity?**
a) top_p ❌
✅**b) temperature**
c) frequency_penalty ❌
d) max_tokens ❌

3. **How to make output more deterministic?**
a) Higher temperature (e.g., 1.5) ❌
✅**b) Lower temperature (e.g., 0.2)**

c) None ✕
d) No effect ✕

4. **What does `top_p` do?**
a) Penalize frequent tokens ✕
b) Limit total output tokens ✕
✅**c) Keep most probable tokens above a cumulative threshold**
d) Force tool call ✕

5. **Reduce LLM repetition – adjust:**
a) presence_penalty ✕
✅**b) frequency_penalty**
c) repetition_penalty ✕*(not in SDK)*
d) token_penalty ✕*(non-existent)*

6. **What does `tool_choice="required"` mean?**
a) Can't call tools ✕
b) Model decides tool/message ✕
✅**c) Model must call a tool**
d) Call all tools ✕

7. **Effect of `parallel_tool_calls=True`?**
a) Synchronous execution ✕
✅**b) Generate multiple tool calls in one turn**
c) Choose tools by parallelism ✕
d) Prioritize tool execution ✕

8. **`truncation='auto'` means?**
a) Always truncate ✕
b) Raise error if input long ✕
✅**c) Auto-drop old messages if input too long**
d) Disable truncation ✕

9. **Purpose of `max_tokens`?**
a) Limit input tokens ✕
b) Limit agent turns ✕
✅**c) Max output tokens generated by LLM**
d) Limit request count ✕

10. **Which setting is used for explicit internal thoughts?**
a) metadata ✕
b) extra_args ✕

c) tool_choice ✗
✓**d) reasoning**

Part 2: Customization & Advanced Behavior

11. **Add custom fields to HTTP request body?**
    a) extra_query ✗
    ✓**b) extra_body**
    c) extra_headers ✗
    d) metadata ✗

12. **What is `extra_args` for?**
    a) Only numeric values ✗
    b) Only strings ✗
    ✓**c) Arbitrary keyword args to provider API**
    d) List of tools ✗

13. **Default behavior of `store` attribute?**
    a) False ✗
    ✓**b) True**
    c) Depends on provider ✗
    d) Raises error ✗

14. **If `include_usage=True`, what is included?**
    a) Agent name ✗
    b) Conversation ID ✗
    ✓**c) Token usage info (e.g., input/output tokens)**
    d) Error messages ✗

15. **Function of `resolve()` method?**
    a) Reset to default ✗
    b) Random settings ✗
    ✓**c) Merge override values into new settings instance**
    d) Validate settings ✗

16. **Result of resolving override with `temperature=0.2`, `max_tokens=100` on global settings (`temperature=0.7`)?**
    a) temp=0.7, max_tokens=None ✗
    b) temp=0.7, max_tokens=100 ✗
    c) temp=0.2, max_tokens=None ✗
    ✓**d) temp=0.2, max_tokens=100**

17. **Why check API docs when using ModelSettings?**
    a) SDK has bugs ✗
    b) Ensure all None ✗
    ✓**c) Not all models/providers support all parameters**
    d) Validate `resolve()` ✗

18. **Encourage novelty – which setting?**
    a) frequency_penalty ✗
    ✓**b) presence_penalty (set high)**
    c) temperature (low) ✗
    d) max_tokens (low) ✗

19. **Send custom HTTP headers – which attribute?**
    a) metadata ✗
    b) extra_body ✗
    ✓**c) extra_headers**
    d) extra_args ✗

20. **What config pattern does `resolve()` enable?**
    a) Flat ✗
    b) Static ✗
    ✓**c) Hierarchical/layered**
    d) Random ✗

OpenAI Agents SDK - Agent Output MCQs

✓✗AgentOutputSchemaBase & AgentOutputSchema – Answer Key with Wrong Answers

1. **What is the primary goal of the AgentOutputSchemaBase and AgentOutputSchema classes?**
   ✗a) To store the agent's internal reasoning.
   ✗b) To configure the LLM's temperature.
   ✓**c) To define, validate, and parse the structured output produced by the LLM.**
   ✗d) To manage external tool definitions.

2. **AgentOutputSchemaBase is an ABC. What does this imply?**
   ✗a) It can be directly instantiated.
   ✓**b) It must be subclassed, and its abstract methods must be implemented.**
   ✗c) It is a concrete implementation.
   ✗d) It is only used for plain text outputs.

3. **The `is_plain_text()` method in AgentOutputSchemaBase determines:**
   ✗a) If the input to the agent is plain text.
   ✗b) If the agent can only generate plain text.
   ✓**c) If the expected output type is simple plain text (vs. a JSON object).**
   ✗d) If the LLM supports plain text mode.

4. **Which method in AgentOutputSchemaBase is responsible for returning the JSON schema of the expected output?**
   ✗a) `get_schema()`
   ✓**b) `json_schema()`**
   ✗c) `get_output_format()`
   ✗d) `schema_definition()`

5. **If `is_plain_text()` returns `False`, which other method of AgentOutputSchemaBase becomes relevant?**
   ✗a) `name()`
   ✗b) `validate_json()`
   ✓**c) `json_schema()`**
   ✗d) `is_strict_json_schema()`

6. **When `validate_json(json_str: str)` is called, what should it do if the json_str is invalid or doesn't conform to the schema?**
   ✗a) Return None.
   ✗b) Return an empty dictionary.
   ✓**c) Raise a `ModelBehaviorError`.**
   ✗d) Log a warning and continue.

7. **What is the main parameter passed to the AgentOutputSchema constructor?**
   ✗a) `json_schema_dict`
   ✓**b) `output_type: type[Any]`**
   ✗c) `agent_name`
   ✗d) `validation_function`

8. **Why does the documentation "strongly recommend" setting `strict_json_schema=True` in AgentOutputSchema?**
   ✗a) Because it makes the agent run faster.
   ✗b) Because it reduces LLM token usage.
   ✓**c) Because it increases the likelihood of the LLM producing correct and valid JSON output.**
   ✗d) Because it allows for more flexible schema definitions.

9. **If `strict_json_schema=True`, what impact does it generally have on the JSON schema communicated to the LLM?**
   ✗a) It makes the schema more complex.
   ✗b) It allows for any JSON structure.

✅**c) It constrains the JSON schema features used, simplifying it for the LLM.**
❌d) It forces the LLM to output plain text.

10. **What does the AgentOutputSchema use internally to generate the `json_schema()` based on the `output_type`?**
   ❌a) It fetches a predefined schema from a server.
   ❌b) It requires manual schema definition for each type.
   ✅**c) It dynamically generates the schema from the provided Python `output_type` (e.g., a Pydantic model).**
   ❌d) It reads the schema from a `.json` file.

11. **If an AgentOutputSchema is initialized with `output_type=str`, what would its `is_plain_text()` method return?**
   ❌a) False
   ✅**b) True**
   ❌c) None
   ❌d) It would raise an error.

12. **The `name()` method in AgentOutputSchema returns:**
   ❌a) A fixed string "AgentOutput".
   ❌b) The name of the agent.
   ✅**c) The name of the output_type (e.g., "str", "dict", "MyPydanticModel").**
   ❌d) A randomly generated UUID.

13. **You define an AgentOutputSchema with a Pydantic model as its `output_type`. When the LLM returns a JSON string, which method will convert that string into an instance of your Pydantic model and validate it?**
   ❌a) `json_schema()`
   ❌b) `is_strict_json_schema()`
   ❌c) `name()`
   ✅**d) `validate_json()`**

14. **What is the consequence if `validate_json()` raises a `ModelBehaviorError`?**
   ❌a) The agent will automatically re-attempt the LLM call.
   ❌b) The LLM will receive a warning message.
   ✅**c) The agent run will likely halt or enter an error state, as the LLM failed to produce valid output.**
   ❌d) The output will be silently dropped.

15. **What is the benefit of defining a strict output schema for an agent?**
   ❌a) It makes the agent's responses always shorter.
   ❌b) It allows the LLM more freedom in its output format.
   ✅**c) It makes the agent's outputs more predictable and easier for downstream code to consume reliably.**
   ❌d) It is primarily for performance optimization.

16. **Which scenario would most strongly indicate the need to define an AgentOutputSchema that is not `is_plain_text`?**
❌a) The agent needs to respond with a simple "Hello!".
❌b) The agent is writing a long essay.
✅**c) The agent needs to provide structured data, like a user's address with separate fields for street, city, and zip code.**
❌d) The agent is generating a code snippet.

17. **If you create a custom class that needs to handle agent output schema and validation, what must it inherit from?**
❌a) `AgentOutputSchema`
✅**b) `AgentOutputSchemaBase`**
❌c) `ResponseOutputItem`
❌d) `ModelSettings`

18. **The AgentOutputSchema typically uses the provided `output_type` to:**
❌a) Determine the agent's personality.
❌b) Generate the necessary API keys.
✅**c) Infer the expected JSON structure and validation rules.**
❌d) Choose the best LLM provider.

19. **Why is it important for AgentOutputSchema to potentially raise a `ModelBehaviorError` during validation?**
❌a) To signal a problem with the user's input.
❌b) To indicate a network issue.
✅**c) To clearly identify when the LLM itself has failed to adhere to the expected output format.**
❌d) To suggest a different LLM model.

20. **What is the relationship between AgentOutputSchemaBase and AgentOutputSchema?**
❌a) AgentOutputSchemaBase extends AgentOutputSchema.
❌b) They are unrelated but serve similar purposes.
✅**c) AgentOutputSchema is a concrete implementation of the abstract AgentOutputSchemaBase.**
❌d) AgentOutputSchema is a utility class for AgentOutputSchemaBase.

OpenAI Agents SDK - Function Schema MCQs

✅❌FuncSchema Quiz – Answer Key with Wrong Answers

1. **What is the main purpose of the `FuncSchema` dataclass?**
❌a) To define how agents interact with users.
❌b) To manage external API keys for tools.

✅**c) To capture the schema of a Python function for an LLM to use as a tool.**
❌d) To store the execution history of tool calls.

2. **Which attribute of `FuncSchema` is crucial for defining the types and validation rules for a function's parameters?**
   ❌a) `name`
   ❌b) `description`
   ✅**c) `params_pydantic_model`**
   ❌d) `params_json_schema`

3. **The `params_json_schema` attribute in `FuncSchema` is derived directly from what?**
   ❌a) The function's docstring.
   ❌b) A manual JSON file.
   ✅**c) The `params_pydantic_model`.**
   ❌d) The LLM's capabilities.

4. **Setting `strict_json_schema: bool = True` in `FuncSchema` is strongly recommended because it:**
   ❌a) Reduces the number of tool calls the LLM makes.
   ❌b) Makes the tool execution faster.
   ✅**c) Increases the likelihood of the LLM providing correct JSON input for the tool.**
   ❌d) Allows the LLM to call any function.

5. **What does the `to_call_args()` method of `FuncSchema` do?**
   ❌a) Converts a function call into a JSON string.
   ✅**b) Converts validated data from the Pydantic model into (*args, **kwargs) for function execution.**
   ❌c) Generates the Pydantic model from arguments.
   ❌d) Validates the raw JSON output from the LLM.

6. **`FuncDocumentation` is a dataclass used to hold metadata primarily extracted from where?**
   ❌a) Function signatures.
   ✅**b) Function docstrings.**
   ❌c) External configuration files.
   ❌d) LLM descriptions.

7. **Which function is used to extract detailed metadata (name, description, parameter descriptions) from a Python function's docstring?**
   ❌a) `function_schema`
   ✅**b) `generate_func_documentation`**
   ❌c) `get_function_metadata`
   ❌d) `extract_docstring`

8. **What is the main utility function for creating a `FuncSchema` object from a Python function?**
   ✗a) `FuncSchema.__init__`
   ✗b) `generate_func_documentation`
   ✓**c) `function_schema`**
   ✗d) `create_tool_schema`

9. **If you want the LLM to see a different name for your tool than your Python function's `__name__`, which parameter of `function_schema` would you use?**
   ✗a) `use_docstring_info`
   ✗b) `strict_json_schema`
   ✓**c) `name_override`**
   ✗d) `docstring_style`

10. **The `takes_context: bool` attribute in `FuncSchema` indicates whether the function expects:**
   ✗a) A history of previous tool calls.
   ✓**b) A `RunContextWrapper` argument as its first parameter.**
   ✗c) A list of all available agents.
   ✗d) A database connection.

11. **If `use_docstring_info=False` in `function_schema`, what is the consequence?**
   ✗a) The function will not be able to be called.
   ✗b) The `params_pydantic_model` will not be generated.
   ✓**c) The `description` and `param_descriptions` for the `FuncSchema` will not be derived from the docstring.**
   ✗d) The `strict_json_schema` will automatically become False.

12. **What does `FuncSchema.signature` hold?**
   ✗a) A hash of the function's code.
   ✗b) A unique ID for the function.
   ✓**c) The `inspect.Signature` object of the original Python function.**
   ✗d) The LLM's "signature" for generating calls.

13. **The `params_json_schema` is the actual JSON representation sent to the LLM. What is its primary role from the LLM's perspective?**
   ✗a) To determine the function's execution priority.
   ✓**b) To inform the LLM about the required arguments and their types for the tool call.**
   ✗c) To tell the LLM if the function is asynchronous.
   ✗d) To help the LLM decide if it should generate a message or a tool call.

14. **When would you typically set `description_override` in `function_schema`?**
   ✗a) When the function has no docstring.
   ✗b) When you want the LLM to ignore the function.
   ✓**c) When the docstring's description isn't ideal for how the LLM should interpret the tool.**
   ✗d) When the function's name is too long.

15. **What type of model must `params_pydantic_model` inherit from?**
    ✗a) `dict`
    ✗b) `Any`
    ✓**c) `pydantic.BaseModel`**
    ✗d) `FunctionArguments`

16. **Which of the following is NOT a direct attribute of `FuncSchema`?**
    ✓**d) `docstring_style`**
    ✗a) `name`
    ✗b) `signature`
    ✗c) `takes_context`

17. **If `generate_func_documentation()` cannot auto-detect the docstring style, what can you do?**
    ✗a) It will raise an error.
    ✓**b) You can explicitly provide the style parameter.**
    ✗c) The description will remain None.
    ✗d) You must rewrite the docstring.

18. **What is the benefit of `FuncSchema` being able to handle functions that `takes_context`?**
    ✗a) It allows the LLM to directly modify the context.
    ✗b) It simplifies the tool execution by removing the need for context.
    ✓**c) It allows tool functions to access shared run-specific data and dependencies.**
    ✗d) It is purely for logging purposes.

19. **In the overall agent workflow, where does the `FuncSchema` primarily get used?**
    ✗a) It's returned as the final output of the agent.
    ✗b) It's part of the LLM's internal state.
    ✓**c) It's sent to the LLM to define available tools for function calling.**
    ✗d) It's used by InputGuardrails for validation.

20. **What is the role of `FuncDocumentation` in the creation of a `FuncSchema`?**
    ✗a) It replaces the `FuncSchema` entirely.
    ✓**b) It acts as an intermediate step to extract natural language descriptions from docstrings for the `FuncSchema`.**
    ✗c) It validates the `FuncSchema`.
    ✗d) It converts `FuncSchema` back into a Python function.

OpenAI Agents SDK - Model Interface MCQs

✅❌Agents SDK – Model Interface Quiz Answer Key

1. **What is the main benefit of the "Model interface" in the Agents SDK?**
   ❌a) It hardcodes specific LLM providers.
   ❌b) It always uses the latest OpenAI model.
   ✅**c) It decouples the agent's logic from specific LLM implementations, allowing for easy swapping of models/providers.**
   ❌d) It automatically optimizes LLM calls for cost.

2. **Which Enum is used to configure the level of logging and data capture for LLM interactions?**
   ❌a) `LogLevel`
   ✅**b) `ModelTracing`**
   ❌c) `TraceMode`
   ❌d) `ModelConfig`

3. **If you want to enable tracing for LLM calls but specifically exclude the actual input/output data, which `ModelTracing` option would you choose?**
   ❌a) `DISABLED`
   ❌b) `ENABLED`
   ✅**c) `ENABLED_WITHOUT_DATA`**
   ❌d) `VERBOSE`

4. **The `Model` class is an ABC. What does this indicate about it?**
   ❌a) It is a concrete implementation of an LLM.
   ✅**b) It defines the abstract contract for any LLM wrapper and cannot be directly instantiated.**
   ❌c) It represents a specific LLM provider.
   ❌d) It is a utility class for managing model settings.

5. **Which `Model` method is designed for making a single, complete LLM request and waiting for the full response?**
   ❌a) `stream_response`
   ✅**b) `get_response`**
   ❌c) `send_query`
   ❌d) `process_input`

6. **The `get_response` method of `Model` returns what type of object?**
   ❌a) `AsyncIterator[TResponseStreamEvent]`
   ❌b) `str`
   ✅**c) `ModelResponse`**
   ❌d) `dict`

7. **The `stream_response` method of `Model` returns what type of object?**
   ✗a) `ModelResponse`
   ✗b) `str`
   ✗c) `list[TResponseStreamEvent]`
   ✓**d) `AsyncIterator[TResponseStreamEvent]`**

8. **Both `get_response` and `stream_response` methods of `Model` accept which common parameter to configure LLM behavior (e.g., temperature, max tokens)?**
   ✗a) `system_instructions`
   ✗b) `tools`
   ✓**c) `model_settings`**
   ✗d) `tracing`

9. **What is the purpose of the `output_schema` parameter in the `Model` methods?**
   ✗a) To define the input format for the LLM.
   ✗b) To specify the data type of `system_instructions`.
   ✓**c) To guide the LLM on the expected structured output format using `AgentOutputSchemaBase`.**
   ✗d) To limit the total number of turns.

10. **The `ModelProvider` base interface is responsible for:**
   ✗a) Executing LLM calls directly.
   ✓**b) Looking up and providing Model instances by name.**
   ✗c) Handling all agent-level business logic.
   ✗d) Managing conversation history.

11. **If you wanted to switch from using OpenAI's `gpt-4o` to Google's `gemini-pro`, which part of the model interface would primarily be swapped or configured?**
   ✗a) The `ModelSettings`.
   ✗b) The `system_instructions`.
   ✓**c) The `ModelProvider` implementation.**
   ✗d) The `tracing` setting.

12. **The `tools` parameter passed to `Model` methods (`get_response`, `stream_response`) contains:**
   ✗a) References to external APIs.
   ✗b) Configuration for internal agent mechanisms.
   ✓**c) A list of Tool objects that the LLM is aware of and can call.**
   ✗d) User input messages.

13. **The `Model` methods are async. What does this imply about how they should typically be called in Python?**
   ✗a) They must be called in a separate thread.
   ✓**b) They should be awaited.**
   ✗c) They return immediately.
   ✗d) They are synchronous by default.

14. **What type of input does the `input` parameter of `Model` methods expect?**
   ✗a) Only `str`
   ✓**b) `str | list[TResponseInputItem]`**
   ✗c) `ModelResponse`
   ✗d) `AsyncIterator`

15. **`ModelTracing.ENABLED` includes:**
   ✗a) Only request headers.
   ✗b) Only response bodies.
   ✓**c) All data, including inputs and outputs.**
   ✗d) No data, only timestamps.

16. **Which of the following is NOT a parameter commonly passed to Model's `get_response` or `stream_response` methods?**
   ✗a) `system_instructions`
   ✗b) `model_settings`
   ✓**c) `run_context`**
   ✗d) `tools`

17. **If a custom LLM integration needs to be built for the Agents SDK, what base class must the LLM wrapper inherit from?**
   ✗a) `ModelProvider`
   ✓**b) `Model`**
   ✗c) `Agent`
   ✗d) `BaseModel`

18. **What is the role of `TResponseStreamEvent` in the `stream_response` method's return type?**
   ✗a) It represents the final, complete response.
   ✗b) It's an error type.
   ✓**c) It represents individual chunks or events received during a streaming response from the LLM.**
   ✗d) It's the type of the tool call.

19. **The `previous_response_id` parameter is described as "Generally not used by the model, except for the OpenAI Responses API." What does this suggest about its primary purpose?**
   ✗a) It's for debugging purposes only.
   ✗b) It's a universal identifier for all LLM calls.
   ✓**c) It's an optional, provider-specific identifier used to maintain context or state in certain APIs.**
   ✗d) It's used for caching.

20. **What is the core responsibility of the `get_model(model_name: str | None)` method within `ModelProvider`?**

❌a) To train a new LLM.
❌b) To initialize the ModelSettings for a given model.
✅**c) To retrieve a specific LLM Model instance by its name.**
❌d) To validate the model's output.

OpenAI Agents SDK - `OpenAIChatCompletionsModel` MCQs

1. **The `OpenAIChatCompletionsModel` class is a concrete implementation for interacting with which specific API?**
   ❌a) OpenAI Assistants API
   ❌b) Google Gemini API
   ✅**c) OpenAI Chat Completions API**
   ❌d) Local LLM API

2. **What is the base class that `OpenAIChatCompletionsModel` inherits from?**
   ❌a) `ABC`
   ❌b) `ModelProvider`
   ✅**c) `Model`**
   ❌d) `ChatModelBase`

3. **The `stream_response` method in `OpenAIChatCompletionsModel` is designed to:**
   ❌a) Return a single, complete response from the LLM.
   ✅**b) Yield partial messages and usage information as they are generated.**
   ❌c) Only process input messages without generating output.
   ❌d) Store the entire conversation history in a stream.

4. **When `stream_response` yields usage information, how accurate is it for ongoing streams?**
   ❌a) It is always perfectly accurate and final.
   ❌b) It is unavailable until the stream completes.
   ✅**c) It is updated incrementally but might not be final until the last chunk is processed.**
   ❌d) It only tracks input tokens during streaming.

5. **Which of the following parameters is *not* listed as an input to the `stream_response` method of `OpenAIChatCompletionsModel`?**
   ❌a) `model_settings`
   ❌b) `tools`
   ✅**c) `api_key`**
   ❌d) `output_schema`
   □ *API keys are handled at the client/provider level, not passed directly.*

6. **The `stream_response` method returns an `AsyncIterator` yielding `TResponseStreamEvent`. What do these events represent?**
   ❌a) Complete LLM responses.
   ❌b) Errors encountered during the stream.
   ✅**c) Individual chunks or partial data received during the streaming process.**
   ❌d) Summaries of the conversation.

7. **By inheriting from `Model`, `OpenAIChatCompletionsModel` guarantees that it provides a consistent interface for:**
   ❌a) Only streaming interactions.
   ❌b) Only non-streaming interactions.
   ✅**c) Both streaming and non-streaming (implied `get_response` implementation) interactions with LLMs.**
   ❌d) Only local model inference.

8. **The `tools` parameter passed to `stream_response` would be translated by `OpenAIChatCompletionsModel` into what format for the OpenAI API?**
   ❌a) A plain string of tool names.
   ❌b) A list of Python function objects.
   ✅**c) OpenAI's specific function-calling (or tool-calling) JSON format.**
   ❌d) A dictionary of tool descriptions.

9. **What is the role of `model_settings` when calling `OpenAIChatCompletionsModel.stream_response`?**
   ❌a) To define the system_instructions.
   ✅**b) To configure LLM-specific parameters like temperature, max_tokens, and tool choice for the OpenAI call.**
   ❌c) To specify the tracing level.
   ❌d) To manage handoff points.

10. **If an agent is built using the generic `Model` interface, could it seamlessly use `OpenAIChatCompletionsModel`?**
   ✅**a) Yes, because `OpenAIChatCompletionsModel` implements the `Model` interface.**
   ❌b) No, it would require significant code changes.
   ❌c) Only if streaming is disabled.
   ❌d) Only if no tools are used.

OpenAI Agents SDK - `OpenAIResponsesModel & Converter` MCQs

✅❌OpenAIResponsesModel Quiz – Answer Key

1. **OpenAIResponsesModel is a concrete implementation of which base class?**
   ❌a) `OpenAIChatCompletionsModel`
   ✅**b) `Model`**
   ❌c) `ModelProvider`
   ❌d) `BaseModel`

2. **What is the key distinction between OpenAIChatCompletionsModel and OpenAIResponsesModel?**
   ✗a) One supports streaming, the other does not.
   ✗b) One supports tools, the other does not.
   ✓**c) They interact with different OpenAI APIs (Chat Completions vs. Responses API).**
   ✗d) One is for input, the other for output.

3. **The `stream_response` method of `OpenAIResponsesModel` returns an `AsyncIterator` yielding what type of events?**
   ✗a) `TResponseStreamEvent`
   ✓**b) `ResponseStreamEvent`**
   ✗c) `OpenAIEvent`
   ✗d) `StreamChunk`

4. **When `OpenAIResponsesModel.stream_response` yields data, what information does it include besides partial messages?**
   ✗a) Agent internal state
   ✗b) Tool call history
   ✓**c) Usage information**
   ✗d) Current time
5. **What is the primary role of the `Converter` class within the `openai_responses.py` module?**
   ✗a) To manage API authentication.
   ✗b) To cache LLM responses.
   ✓**c) To translate data formats between the Agents SDK and the OpenAI Responses API.**
   ✗d) To perform sentiment analysis on LLM outputs.

6. **Which of the following is most likely a task performed by the `Converter`?**
   ✗a) Executing Python functions called by the LLM.
   ✗b) Determining the temperature for an LLM call.
   ✓**c) Transforming a `Tool` object into the OpenAI Responses API's tool definition JSON.**
   ✗d) Handling network retries for API calls.

7. **If the Agents SDK uses a generic `ModelSettings` object, but the OpenAI Responses API expects a specific JSON structure for model parameters, which component is responsible for this translation?**
   ✗a) `OpenAIResponsesModel` directly
   ✗b) The `Model` base class
   ✓**c) The `Converter` class**
   ✗d) `AgentOutputSchema`

8. **What is implied about the `get_response` method for `OpenAIResponsesModel` given its inheritance from `Model`?**
   ✗a) It is not implemented.
   ✗b) It only supports streaming.
   ✓**c) It must also be implemented to handle non-streaming requests.**
   ✗d) It is handled automatically by the base class.

9. **The `ResponseStreamEvent` type, as used by `OpenAIResponsesModel.stream_response`, suggests:**
   ✗a) It's a generic event type for any streaming API.
   ✗b) It's an error-only event type.
   ✓**c) It's a specialized, potentially richer event type specific to the OpenAI Responses API.**
   ✗d) It only contains token counts.

10. **The consistency of parameters (`system_instructions`, `model_settings`, `tools`) across `Model` implementations (e.g., OpenAIChatCompletionsModel and OpenAIResponsesModel) enables:**
   ✗a) Automatic model selection by the SDK.
   ✗b) Lower API costs.
   ✓**c) Easy interchangeability of LLM backends without major code changes.**
   ✗d) Faster LLM response times.

11. **Why would a developer choose `OpenAIResponsesModel` over `OpenAIChatCompletionsModel`?**
   ✗a) OpenAIChatCompletionsModel does not support tools.
   ✗b) OpenAIResponsesModel is generally faster.
   ✓**c) To leverage specific features, richer streaming events, or advanced prompt paradigms offered by the OpenAI Responses API.**
   ✗d) OpenAIResponsesModel does not require an OpenAI API key.

12. **The `Converter` helps to bridge the gap between:**
   ✗a) User input and agent output.
   ✗b) Synchronous and asynchronous calls.
   ✓**c) The SDK's generic interfaces and a specific LLM provider's API formats.**
   ✗d) Different agent personalities.

13. **If `OpenAIResponsesModel` receives an `AgentOutputSchemaBase` object, which component is most likely to translate this into the precise format the OpenAI Responses API expects for structured output?**
   ✗a) The `ModelSettings`
   ✗b) The `tools` list
   ✓**c) The `Converter`**
   ✗d) The `previous_response_id`

14. **What is the role of `previous_response_id` in the `OpenAIResponsesModel.stream_response` method?**
   ✗a) To specify the maximum length of the response.
   ✗b) To identify the user making the request.
   ✓**c) To provide contextual continuity for the OpenAI Responses API.**
   ✗d) To track the total usage.

15. **The `OpenAIResponsesModel` is part of a broader design pattern where `Model` is an abstraction. What does this allow the SDK to be?**
   ✗a) Tightly coupled to OpenAI.
   ✗b) Limited to a single LLM.
   ✓**c) Flexible and extensible to support various LLM providers.**
   ✗d) Strictly for chat-based applications.

OpenAI Agents SDK - MCP Servers MCQs

✓✗MCPServer Quiz – Answer Key

1. **What is the primary purpose of the `MCPServer` base class?**
   ✗a) To manage LLM context windows.
   ✓**b) To define a common interface for interacting with external services hosting tools.**
   ✗c) To configure OpenAI API keys.
   ✗d) To store agent conversation history.

2. **Which of the following is an abstract method that all concrete `MCPServer` implementations must provide?**
   ✗a) `get_version()`
   ✗b) `send_heartbeat()`
   ✗c) `process_message()`
   ✓**d) `call_tool()`**

3. **The `connect()` and `cleanup()` methods in `MCPServer` are both:**
   ✗a) Synchronous
   ✓**b) Asynchronous**
   ✗c) Optional
   ✗d) Used only for local servers

4. **What does the `list_tools()` method of `MCPServer` return?**
   ✗a) A dictionary of tool names and descriptions.
   ✗b) A list of `FuncSchema` objects.
   ✓**c) A list of `Tool` objects.**
   ✗d) The current status of the server.

5. **`MCPServerStdioParams` is used to configure which type of MCP server?**
   ✗a) Remote HTTP server
   ✗b) Server-Sent Events server
   ✓**c) Standard input/output (subprocess) server**
   ✗d) WebSocket server

6. **In `MCPServerStdioParams`, what does the `command` attribute specify?**
   ✗a) The name of the tool to call.
   ✗b) The URL of the server.
   ✓**c) The executable to run to start the server (e.g., python).**
   ✗d) A command to send to the LLM.

7. **The `cache_tools_list` parameter in MCPServerStdio (and others) is primarily for:**
   ✗a) Ensuring tool definitions are always up-to-date.
   ✗b) Reducing memory usage on the client side.
   ✓**c) Improving latency by avoiding repeated round-trips to list tools if they are static.**
   ✗d) Forcing the server to reload its tools.

8. **Which method would you call to force a refresh of the cached tool list on an MCPServer?**
   ✗a) `refresh_tools()`
   ✓**b) `invalidate_tools_cache()`**
   ✗c) `reset_cache()`
   ✗d) `clear_tools()`

9. **`MCPServerSseParams` requires which essential attribute to define the server connection?**
   ✗a) `command`
   ✗b) `encoding`
   ✓**c) `url`**
   ✗d) `cwd`

10. **Which MCP server implementation is suitable for connecting to a remote service that streams updates over HTTP?**
    ✗a) `MCPServerStdio`
    ✓**b) `MCPServerSse`**
    ✓**(Also accepted: c) `MCPServerStreamableHttp`**
    ✗d) `MCPServerLocal`

11. **What is the role of the `sse_read_timeout` attribute in `MCPServerSseParams`?**
    ✗a) Timeout for the initial HTTP request.
    ✗b) Timeout for connecting to the server process.
    ✓**c) The timeout for the Server-Sent Events (SSE) connection itself.**
    ✗d) Timeout for a single tool call.

12. **`MCPServerStreamableHttp` likely uses a transport protocol that is a specialization or evolution of:**
    ✗a) Stdio
    ✓**b) HTTP with Server-Sent Events (SSE)**
    ✗c) WebSockets
    ✗d) UDP

13. **If `cache_tools_list` is True and the tools on the server actually change, what must be done for the client to see the new tools?**
    ✗a) Restart the agent entirely.
    ✗b) Call `connect()` again.
    ✓**c) Call `invalidate_tools_cache()`.**
    ✗d) It updates automatically.

14. **The `call_tool()` method returns a `CallToolResult`. What does this object typically contain?**
    ✗a) The name of the tool called.
    ✗b) The arguments passed to the tool.
    ✓**c) The result or outcome of the tool invocation.**
    ✗d) The tool's description.

15. **What type of parameter is `terminate_on_close` specifically associated with?**
    ✓**a) `MCPServerStdioParams`**
    ✗b) `MCPServerSseParams`
    ✗c) `MCPServerStreamableHttpParams`
    ✗d) All MCPServer parameters

16. **Which attribute in `MCPServerStdioParams` is used to specify the initial working directory for the subprocess?**
    ✗a) `env`
    ✗b) `command`
    ✓**c) `cwd`**
    ✗d) `args`

17. **The `name` property of `MCPServer` is:**
    ✗a) An optional field that defaults to None.
    ✓**b) An abstract property that must return a readable string.**
    ✗c) Used only for debugging purposes.
    ✗d) The unique identifier for the server.

18. **What would be a good use case for `MCPServerStdio`?**
    ✗a) Connecting to a public web API.
    ✓**b) Integrating a local Python script running as a separate process that exposes tools.**
    ✗c) Receiving real-time stock updates from a remote server.
    ✗d) Communicating with an LLM provider.

19. **The `client_session_timeout_seconds` parameter refers to:**
    ✗a) The timeout for the `connect()` method.
    ✗b) The maximum duration a tool call can take.
    ✓**c) The read timeout for the underlying MCP ClientSession.**
    ✗d) The time before the server automatically cleans up.

20. **The Model Context Protocol (MCP) aims to facilitate communication between agents and:**
    ✗a) Different LLM providers.
    ✗b) Other agents in a multi-agent system.
    ✓**c) External, self-contained services or "tools."**
    ✗d) Human users for feedback.

OpenAI Agents SDK - MCPUtil MCQs

1. **What is the primary role of the `MCPUtil` class?**
   ✗a) To manage MCP server authentication.
   ✓**b) To provide utilities for interoperability between MCP and Agents SDK tools.**
   ✗c) To implement new MCP transport protocols.
   ✗d) To store MCP server configuration.

2. **The `get_all_function_tools` method is used to retrieve tools from:**
   ✗a) A single MCP server.
   ✓**b) A list of multiple MCP servers.**
   ✗c) Local Python functions only.
   ✗d) Directly from the LLM.

3. **What is the return type of `get_all_function_tools`?**
   ✗a) list[FuncSchema]
   ✗b) list[MCPServer]
   ✓**c) list[Tool]**
   ✗d) list[str]

4. **The `convert_schemas_to_strict` parameter in `MCPUtil` methods is important for:**
   ✗a) Reducing network latency.
   ✗b) Caching tool definitions.
   ✓**c) Ensuring LLMs generate correct arguments by using strict JSON schemas.**
   ✗d) Changing the tool's name.

5. **Which method would you use if you only wanted to retrieve tools from one specific MCP server?**
   ✗a) `get_all_function_tools`
   ✓**b) `get_function_tools`**
   ✗c) `to_function_tool`
   ✗d) `invoke_mcp_tool`

6. **The `to_function_tool` method converts a generic `Tool` object into what specific type?**
   ✗a) FuncSchema
   ✗b) CallToolResult
   ✓**c) FunctionTool**
   ✗d) MCPServer

7. **Why is the `server: MCPServer` parameter needed in `to_function_tool`?**
   ✗a) To get the server's name for logging.
   ✗b) To check if the server is currently connected.
   ✓**c) To establish the link back to the specific MCP server that will execute the tool.**
   ✗d) To invalidate the server's tool cache.

8. **What is the purpose of the `invoke_mcp_tool` method?**
   ✗a) To discover available tools on an MCP server.
   ✗b) To connect to an MCP server.
   ✓**c) To execute a specific tool on an MCP server with provided arguments.**
   ✗d) To convert an MCP tool's result to a different format.

9. **The `input_json` parameter in `invoke_mcp_tool` is:**
   ✗a) The raw user input message.
   ✗b) The tool's description.
   ✓**c) The arguments for the tool call, provided as a JSON string.**
   ✗d) A list of previous tool call results.

10. **What is the return type of `invoke_mcp_tool`?**
   ✗a) dict[str, Any]
   ✓**b) CallToolResult**
   ✗c) Any
   ✗d) str

11. **All methods within `MCPUtil` are defined as:**
   ✗a) Instance methods.
   ✓**b) Class methods.**
   ✗c) Static methods.
   ✗d) Abstract methods.

12. **Sequence of MCPUtil operations to use an MCP tool:**
   ✗a) `invoke_mcp_tool` → `get_function_tools`
   ✗b) `to_function_tool` → `invoke_mcp_tool` → `get_function_tools`
   ✓**c) `get_function_tools` (or `get_all_function_tools`) → `to_function_tool` → `invoke_mcp_tool`**
   ✗d) `get_all_function_tools` → `invoke_mcp_tool`

13. **The `context: RunContextWrapper[Any]` parameter in `invoke_mcp_tool` is for:**
   ✗a) Storing the tool's output.
   ✗b) Managing network connections.
   ✓**c) Providing the tool with access to the agent's current runtime context.**
   ✗d) Specifying the LLM model to use.

14. **If `convert_schemas_to_strict` is False, potential drawback:**
   ✗a) Slower tool execution.
   ✗b) Inability to discover tools.
   ✓**c) LLMs might struggle to generate valid JSON arguments due to a less constrained schema.**
   ✗d) Tools will not be able to return a result.

15. **MCPUtil bridges:**
   ✗a) Python and JavaScript code.
   ✗b) Synchronous and asynchronous operations.
   ✓**c) The Agents SDK's internal tool representation and external MCP-compliant services.**
   ✗d) Text-based and image-based inputs.

16. **Which is NOT a direct parameter to `invoke_mcp_tool`?**
   ✗a) server
   ✗b) tool
   ✓**c) output_schema**
   ✗d) input_json

17. **Which method benefits most from tool list caching?**
   ✗a) `invoke_mcp_tool`
   ✗b) `to_function_tool`
   ✓**c) `get_function_tools` / `get_all_function_tools`**
   ✗d) None of them directly...

18. **MCPUtil methods are async where appropriate. This means:**
   ✗a) Block execution until result is available.
   ✓**b) Perform non-blocking I/O operations, typically when interacting with MCP servers.**
   ✗c) Run in a separate thread.
   ✗d) Execute multiple tools in parallel.

19. **Why does MCPUtil exist as a separate class?**
   ✗a) To avoid circular dependencies.
   ✗b) To keep `MCPServer` focused on its core...
   ✗c) To provide a central point...
   ✓**d) All of the above.**

20. **If `invoke_mcp_tool` returns a `str`, what is the most likely format?**
   ✗a) Plain text summary
   ✓**b) JSON string**
   ✗c) XML string
   ✗d) Python literal

Tracing

OpenAI Agents SDK - Tracing Module MCQs

1. **What is the main purpose of the Tracing module in the Agents SDK?**
   ✅**c) To provide observability and debugging for agent workflows.**
   ❌a) To manage API authentication.
   ❌b) To handle asynchronous operations.
   ❌d) To store conversation history persistently.

---

2. **A Trace in the tracing module represents:**
   ✅**b) A complete logical workflow or request.**
   ❌a) A single operation within a workflow.
   ❌c) An error log entry.
   ❌d) A configuration setting for the agent.

---

3. **A Span in the tracing module represents:**
   ✅**c) A single operation or unit of work within a trace.**
   ❌a) The entire conversation flow.
   ❌b) A collection of related traces.
   ❌d) An external API call only.

---

4. **Which abstract base class defines the interface for creating and managing traces and spans?**
   ✅**c) TraceProvider**
   ❌a) TracingProcessor
   ❌b) SpanData
   ❌d) Trace

---

5. **Which method on TraceProvider would you use to add a component that processes trace and span events?**
   ✅**c) register_processor()**
   ❌a) create_span()
   ❌b) get_current_trace()
   ❌d) set_disabled()

---

6. **What is the primary responsibility of a TracingProcessor?**
   ✅**c) To consume and process trace and span events (e.g., for logging or export).**
   ❌a) To generate unique trace IDs.

❌b) To start and finish spans.
❌d) To disable tracing dynamically.

---

7. **Which TracingProcessor method should not block or raise exceptions to avoid interfering with application logic?**
✅**d) on_span_end()**
❌a) shutdown()
❌b) force_flush()
❌c) on_trace_start()

---

8. **GenerationSpanData is a specific type of SpanData that would typically capture details about:**
✅**c) LLM model input/output messages, model configuration, and usage.**
❌a) Tool invocation inputs and outputs.
❌b) Agent handoffs.
❌d) Custom, arbitrary data.

---

9. **If you want to create a span that records arbitrary, structured data for a custom operation, which SpanData type would you use?**
✅**c) CustomSpanData**
❌a) AgentSpanData
❌b) FunctionSpanData
❌d) ResponseSpanData

---

10. **What are the two primary methods used to define the boundaries (start and end) of a Span or Trace?**
✅**c) start() and finish()**
❌a) create() and destroy()
❌b) init() and close()
❌d) open() and shut()

---

11. **What is the benefit of using a `with` statement (context manager) with span creation functions like `tracing.generation_span()`?**
✅**c) It automatically calls start() when entering and finish() when exiting the block.**
❌a) It makes the span immutable.
❌b) It automatically exports the span to a database.
❌d) It disables the span if an error occurs.

---

12. **If you create a new Trace or Span but do not call its start() method or use it as a context manager, what happens?**
✅**b) It will not be recorded by the tracing system.**
❌a) It will automatically start when the first child span is created.
❌c) It will only record errors.
❌d) It will throw an error immediately.

---

13. **The `parent` parameter in span creation functions (e.g., `agent_span()`) is used to:**
✓**c) Explicitly link the new span to an existing parent trace or span, forming a hierarchy.**
✗a) Specify the LLM model to be used.
✗b) Define the ownership of the span in a multi-user system.
✗d) Determine if the span should be disabled.

---

14. **To temporarily stop all tracing without changing individual span creation calls, which global function would you use?**
✓**d) set_tracing_disabled(True)**
✗a) force_flush()
✗b) set_trace_provider(None)
✗c) remove_all_processors()

---

15. **What format does `time_iso()` method on `TraceProvider` return the current time in?**
✓**c) ISO 8601 format**
✗a) Unix timestamp
✗b) Milliseconds since epoch
✗d) A localized date and time string

---

16. **Which SpanData type would be most appropriate for recording a control transfer between two different agents in a multi-agent system?**
✓**b) HandoffSpanData**
✗a) AgentSpanData
✗c) CustomSpanData
✗d) ResponseSpanData

---

17. **The `trace_id` property on a Trace object ensures:**
✓**c) The trace can be uniquely identified and correlated across different operations.**
✗a) The trace is always processed synchronously.
✗b) The trace can be easily exported to an external API key.
✗d) The trace is immutable once created.

---

18. **If a TracingProcessor queues spans before sending them to an external system, which method would you call to ensure all pending spans are immediately dispatched?**
✓**c) force_flush()**
✗a) shutdown()
✗b) on_span_end()
✗d) register_processor()

---

19. **What is the primary difference between `generation_span` and `response_span` creation utilities?**
✓**c) generation_span captures full model generation details (input, output, model, usage), while response_span might focus on a simpler model response identifier.**
✗a) generation_span is for tool calls, response_span is for LLM calls.

❌b) generation_span captures only errors, response_span captures success.
❌d) There is no functional difference; they are aliases.

---

20. **What is the recommended way to generate a new unique span identifier if you need to provide one explicitly?**
✅**c) Use util.gen_span_id().**
❌a) Hardcode a random string.
❌b) Use uuid.uuid4().hex.
❌d) Rely on the TraceProvider to generate it implicitly.

OpenAI Agents SDK - Trace and Span Creation MCQs

1. **When creating a new trace using `tracing.trace()`, which parameter is required?**
✅**c) workflow_name**
❌a) trace_id
❌b) group_id
❌d) metadata

---

2. **A trace created with `tracing.trace(...)` will automatically start recording upon creation.**
✅**b) False**
❌a) True

---

3. **What is the recommended way to ensure a trace or span automatically starts and finishes, even if exceptions occur?**
✅**c) Use it as a context manager (with trace(...):)**
❌a) Always call start() and finish() in a try...finally block.
❌b) Set auto_start=True during creation.
❌d) Call tracing.get_current_trace().start() manually.

---

4. **Which function would you use to create a span specifically designed to track an LLM's reasoning process, including its input and output messages?**
✅**c) generation_span()**
❌a) function_span()
❌b) agent_span()
❌d) response_span()

---

5. **The `disabled=True` parameter in span/trace creation functions will:**
✅**b) Return a Trace or Span object, but it will not be recorded.**
❌a) Cause an error if tracing is globally enabled.
❌c) Only record errors within that trace/span.
❌d) Prevent the start() method from being called.

---

6. **Which function is used to create a span for a tool or function invocation?**
✓**b) function_span()**
✗a) agent_span()
✗c) custom_span()
✗d) mcp_tools_span()

---

7. **To explicitly link a newly created span to an existing trace or span as its parent, which parameter should be used?**
✓**c) parent**
✗a) workflow_name
✗b) group_id
✗d) trace_id

---

8. **What is the primary difference in data captured by `generation_span` compared to `response_span`?**
✓**c) generation_span captures detailed input messages, model config, and usage; response_span focuses more on the final output/response object.**
✗a) response_span includes tool calls; generation_span does not.
✗b) response_span is for agent communication; generation_span is for user output.
✗d) generation_span records only errors; response_span records successes.

---

9. **Which function allows you to create a versatile span for any custom operation and attach arbitrary data?**
✓**c) custom_span()**
✗a) agent_span()
✗b) function_span()
✗d) guardrail_span()

---

10. **If you need to retrieve the currently active span in the execution context, which function would you call?**
✓**b) get_current_span()**
✗a) get_current_trace()
✗c) get_active_context()
✗d) get_global_span()

---

11. **The `metadata` parameter in `tracing.trace()` allows you to attach:**
✓**b) Any arbitrary dictionary of user-defined information.**
✗a) Only pre-defined key-value pairs.
✗c) Only security credentials.
✗d) A list of child span IDs.

---

12. **A `handoff_span` is specifically designed to track:**
✓**c) The transfer of control or responsibility between different agents.**
✗a) Data transfer between a model and a database.
✗b) The parsing of user input.
✗d) The loading of agent configurations.

13. **Which of the following span creation functions would be most appropriate for recording whether a "PII Detector" rule was activated?**
✅**d) guardrail_span()**
❌a) function_span()
❌b) custom_span()
❌c) agent_span()

---

14. **The `result` parameter in `mcp_tools_span()` typically contains:**
✅**c) A list of tool names or descriptions discovered from the MCP server.**
❌a) The input query to the MCP server.
❌b) The duration of the MCP call.
❌d) The error message from the MCP server.

---

15. **In `transcription_span()` and `speech_span()`, what does the `input` parameter typically represent?**
✅**c) Base64 encoded audio bytes (for transcription) or text (for speech synthesis).**
❌a) The LLM prompt.
❌b) The name of the audio file.
❌d) A URL to an audio stream.

---

16. **If you omit the `span_id` parameter when creating a span, what generally happens?**
✅**c) The system will automatically generate a unique ID for the span.**
❌a) An error is raised, requiring a span_id.
❌b) The span is automatically disabled.
❌d) The span will inherit the trace_id as its span_id.

---

17. **The `group_id` parameter in `tracing.trace()` is useful for:**
✅**b) Linking multiple, distinct traces that belong to the same overarching conversation or process.**
❌a) Categorizing traces by their workflow_name.
❌c) Setting access control for traces.
❌d) Defining the maximum number of spans a trace can contain.

---

18. **What is the common parameter found in almost all the span creation functions that allows for immediate non-recording?**
✅**c) disabled**
❌a) sync
❌b) enabled
❌d) silent

---

19. **A `SpeechGroupSpanData` would likely encapsulate:**
✅**b) A logical grouping of related speech operations, possibly representing a spoken turn.**
❌a) A single word generated by TTS.

❌c) The configuration settings for a speech model.
❌d) An error during speech processing.

---

20. **When calling `function_span`, if the `input` and `output` parameters are `None`, but the name is provided, will the span still be recorded (assuming tracing is enabled and it's started)?**
✅**c) Yes, the span will be recorded with its name and timestamps, but without input/output data.**
❌a) No, input and output are required fields for function_span.
❌b) Only if a parent span is explicitly provided.
❌d) It depends on the TracingProcessor implementation.

OpenAI Agents SDK - Trace MCQs

1. **What is the primary role of a Trace object in the tracing module?**
✅**c) To represent a complete logical workflow or end-to-end operation.**
❌a) To manage individual operation details.
❌b) To process and export trace data.
❌d) To generate unique identifiers for spans.

---

2. **Which of the following is an abstract property of the Trace class?**
✅**c) name**
❌a) duration
❌b) start_time
❌d) children

---

3. **The `trace_id` property of a Trace is used for:**
✅**c) Uniquely identifying and correlating all related spans within a workflow.**
❌a) Sorting traces by their creation time.
❌b) Defining the type of workflow.
❌d) Enabling or disabling the trace.

---

4. **When `trace.start(mark_as_current=True)` is called:**
✅**c) The trace is set as the active trace for subsequent implicit parent-child relationships.**
❌a) The trace is immediately exported.
❌b) All previously started traces are finished.
❌d) Tracing is globally disabled.

---

5. **What happens when `trace.finish(reset_current=True)` is called on a trace that was previously marked as current?**
✅**c) The trace is removed from the current tracing context.**
❌a) The trace is restarted.

✗b) It forces a flush of all trace processors.
✗d) An error is raised because a trace cannot be reset.

---

6. **The `export()` method on the Trace abstract base class is intended for:**
✓**c) Converting the trace's data into a dictionary for serialization and external consumption.**
✗a) Printing the trace details to the console.
✗b) Storing the trace in a temporary cache.
✗d) Loading trace data from a file.

---

7. **Which class is a "no-operation" (dummy) implementation of Trace?**
✓**d) NoOpTrace**
✗a) TraceProvider
✗b) TracingProcessor
✗c) TraceImpl

---

8. **When would a `NoOpTrace` typically be returned by the TraceProvider?**
✓**c) When tracing is globally disabled or explicitly disabled for that specific trace.**
✗a) When an error occurs during trace creation.
✗b) When a trace successfully completes its execution.
✗d) Only in development environments.

---

9. **What is the primary benefit of using `NoOpTrace` when tracing is disabled?**
✓**c) It provides a consistent interface, preventing the need for `if tracing_enabled:` checks in application code.**
✗a) It consumes less memory than TraceImpl.
✗b) It allows for different export formats.
✗d) It speeds up the `start()` and `finish()` methods.

---

10. **Which class is the concrete implementation of Trace that actively records data and interacts with TracingProcessors?**
✓**c) TraceImpl**
✗a) Trace
✗b) NoOpTrace
✗d) BaseTrace

---

11. **What action does `TraceImpl.start()` method invoke?**
✓**b) It records the start time and notifies TracingProcessors via `on_trace_start()`.**
✗a) It immediately calls `export()`.
✗c) It creates new child spans automatically.
✗d) It throws an exception if a trace_id is not provided.

---

12. **TraceImpl notifies registered TracingProcessors at which points in its lifecycle?**
✓**c) At the beginning (`on_trace_start`) and end (`on_trace_end`) of the trace.**
✗a) Only when an error occurs.
✗b) Only upon `export()`.
✗d) Only when a new span is added to it.

---

13. **If `trace.export()` on a `NoOpTrace` is called, what will it return?**
✓**b) None**
✗a) An empty dictionary.
✗c) An error.
✗d) A dictionary containing only the `trace_id`.

---

14. **The `name` property of a Trace helps in:**
✓**b) Providing a human-readable description for the type of workflow being traced.**
✗a) Distinguishing between `TraceImpl` and `NoOpTrace`.
✗c) Determining the parent-child relationship of spans.
✗d) Setting the priority level for trace processing.

---

15. **What type of object does the `export()` method on `TraceImpl` return?**
✓**c) A dict[str, Any] (dictionary).**
✗a) A list of strings.
✗b) A serialized JSON string.
✗d) A custom `TraceData` object.

---

16. **A Trace is described as the "root level object" because:**
✓**c) It is the top-most container for a logical workflow, encompassing multiple spans.**
✗a) It has no parent.
✗b) It always runs first.
✗d) It is stored in the root directory.

---

17. **If you create a Trace instance but do not call its `start()` method, what is the consequence for tracing?**
✓**b) It will not be recorded by the tracing system, as its lifecycle events won't be triggered.**
✗a) It will automatically start when the first span is added.
✗c) It will be recorded as a pending trace.
✗d) Only its `trace_id` will be recorded.

---

18. **The `reset_current` parameter in `finish()` is primarily concerned with:**
✓**c) Managing the global context of the currently active trace.**
✗a) Resetting the trace's start time.
✗b) Clearing all child spans from the trace.
✗d) Re-exporting the trace data.

---

19. **Why are Trace properties like `trace_id` and `name` abstract methods?**
✓**c) To enforce that concrete Trace implementations must provide these essential pieces of information.**
✗a) To allow for different data types.
✗b) To indicate they can be changed dynamically.
✗d) To make them optional for specific trace types.

---

20. **If an application uses `TraceImpl` objects, which other component is essential for processing and potentially storing the trace data?**
✓**c) TracingProcessor**
✗a) SpanData
✗b) `gen_trace_id` utility
✗d) ResponseSpanData

OpenAI Agents SDK - Span MCQs

1. **What does a Span fundamentally represent in the tracing module?**
✓**b) A single unit of work or an individual operation within a trace.**
✗a) A complete workflow.
✗c) A global configuration setting.
✗d) An error log.

---

2. **The `Span` class is defined as `Generic[TSpanData]`. What is the purpose of `TSpanData`?**
✓**c) It allows different span types to carry specific, structured data relevant to their operation.**
✗a) It represents the unique ID of the span.
✗b) It indicates whether the span is currently active.
✗d) It defines the parent of the span.

---

3. **Which of the following methods is abstract in the Span class?**
✓**c) export()**
✗a) get_id()
✗b) start()
✗d) get_duration()

---

4. **When `span.start(mark_as_current=True)` is called:**
✓**c) The span is set as the active span in the tracing context, making it the default parent for new child spans.**
✗a) The span is immediately sent to all processors.
✗b) The span's parent is automatically reset.
✗d) Tracing is temporarily disabled for this span.

---

5. **What is the primary effect of calling `span.finish()`?**
✓**b) It marks the end of the span's execution and potentially notifies processors.**
✗a) It restarts the span's timer.

❌c) It creates a new child span.
❌d) It exports the entire trace.

---

6. **Which class is a "no-operation" (dummy) implementation of Span?**
✅**b) NoOpSpan**
❌a) SpanProcessor
❌c) SpanData
❌d) SpanInterface

---

7. **NoOpSpan objects are primarily used when:**
✅**b) Tracing is disabled, to provide a consistent API without actual recording.**
❌a) A span has no parent.
❌c) A span represents an error condition.
❌d) Only asynchronous operations are being traced.

---

8. **What is the main advantage of NoOpSpan for developers?**
✅**c) It simplifies application code by removing the need for `if tracing_enabled:` checks around span operations.**
❌a) It provides detailed debugging information even when disabled.
❌b) It allows for dynamic enabling/disabling of individual spans at runtime.
❌d) It automatically flushes trace data to a persistent store.

---

9. **Which class is the concrete implementation of Span that actively records data and interacts with TracingProcessors?**
✅**c) SpanImpl**
❌a) Span
❌b) NoOpSpan
❌d) BaseSpan

---

10. **What does `SpanImpl` do when its `start()` method is called?**
✅**c) It records the start time and notifies registered TracingProcessors.**
❌a) It calculates the span's total duration.
❌b) It exports the span data immediately.
❌d) It recursively starts all its child spans.

---

11. **If `span.finish(reset_current=True)` is called on a `SpanImpl` that was the current span, what is the likely outcome regarding the tracing context?**
✅**c) The current span context will revert to its parent (or None if it had no parent).**
❌a) The span will remain the current span.
❌b) The entire trace will be finished.
❌d) A new span will be automatically created.

---

12. **SpanImpl maintains a reference to its parent for what purpose?**
    ✓**c) To establish the hierarchical relationship within a trace.**
    ✗a) To determine if it should be recorded.
    ✗b) To set its name property.
    ✗d) To control its disabled status.

---

13. **What type of information does a SpanImpl object typically hold through its TSpanData instance?**
    ✓**c) Specific details relevant to the operation it represents (e.g., model input/output for a generation span).**
    ✗a) Global tracing configuration.
    ✗b) A list of all traces in the system.
    ✗d) The API key for the tracing backend.

---

14. **TracingProcessors receive notifications from `SpanImpl` at which points?**
    ✓**c) When the span starts (`on_span_start`) and when it finishes (`on_span_end`).**
    ✗a) Only when an error occurs within the span.
    ✗b) Only when the `export()` method is called.
    ✗d) Periodically while the span is active.

---

15. **What is the return type of the `finish()` method on the Span abstract class?**
    ✓**c) None**
    ✗a) bool
    ✗b) dict
    ✗d) Span

---

16. **If you create a Span instance using a helper function (like `tracing.agent_span`) but do not use a `with` statement or call `start()/finish()` manually, what will happen?**
    ✓**b) The span will not be recorded by the tracing system because its lifecycle events are not triggered.**
    ✗a) It will automatically start and finish when the Python interpreter exits.
    ✗c) It will only record errors.
    ✗d) It will implicitly become a child of the root trace.

---

17. **The `mark_as_current` parameter in `start()` is a bool. If set to True for a Span, it means:**
    ✓**c) This span becomes the default parent for any subsequent spans created without an explicit parent.**
    ✗a) This span is the root of a new trace.
    ✗b) This span will run in a separate thread.
    ✗d) This span should be exported immediately.

---

18. **Span objects are designed to be:**
    ✓**c) Hierarchical and nested within a Trace.**
    ✗a) Independent and unrelated.
    ✗b) Only for error logging.
    ✗d) Immutable after creation.

---

19. **Which of the following is NOT a method or property directly defined on the Span abstract base class?**
✓**c) export()** *(This belongs to `SpanData`, not directly to `Span`)*
✗a) start()
✗b) finish()
✗d) mark_as_current *(not a method, it's a parameter in `start()`)*

---

20. **When dealing with tracing, the distinction between `SpanImpl` and `NoOpSpan` is handled by the:**
✓**b) TraceProvider**
✗a) TracingProcessor
✗c) SpanData types
✗d) The application code itself with `if` statements.

OpenAI Agents SDK - Processor and Exporter MCQs

1. **What is the primary role of a TracingProcessor?**
✓**c) To receive and process lifecycle events (start/end) from traces and spans.**
✗a) To create new traces and spans.
✗b) To store trace data persistently.
✗d) To generate unique IDs for tracing.

---

2. **Which method on TracingProcessor is called when a trace begins?**
✓**b) on_trace_start()**
✗a) on_span_start()
✗c) start_trace()
✗d) process_trace_begin()

---

3. **The `on_span_end()` method in TracingProcessor has a crucial recommendation: it "Should not block or raise exceptions." Why is this important?**
✓**c) To prevent interference with the application's critical path and avoid performance degradation or crashes.**
✗a) To ensure all data is immediately exported.
✗b) To reduce the memory footprint of the processor.
✗d) To allow for easier debugging.

---

4. **When should the `shutdown()` method of a TracingProcessor typically be called?**
✓**c) When the application or tracing system is stopping, for cleanup.**
✗a) Every time a span finishes.
✗b) When a new trace is created.
✗d) After every `force_flush()` operation.

---

5. **What is the purpose of the `force_flush()` method on a TracingProcessor?**
✓**c) To compel the processor to immediately send any buffered or queued traces/spans.**
✗a) To disable all ongoing tracing.

✗b) To clear all previously exported data.
✗d) To restart the trace collection process.

---

6. **What is the main responsibility of a TracingExporter?**
✓**c) To send processed trace and span data to an external destination (e.g., console, backend).**
✗a) To define the structure of trace and span data.
✗b) To manage the lifecycle of traces and spans.
✗d) To filter out sensitive information from traces.

---

7. **The `export()` method of TracingExporter takes which type of argument?**
✓**d) A list[Trace | Span[Any]] (a list of traces and/or spans).**
✗a) A single Trace object.
✗b) A single Span object.
✗c) A dict representation of a trace or span.

---

8. **Which interface is more general, dealing with events, while the other is more specialized, dealing with sending data out?**
✓**b) TracingProcessor is general; TracingExporter is specialized.**
✗a) TracingExporter is general; TracingProcessor is specialized.
✗c) Both are equally general.
✗d) Neither is considered general; they serve distinct, unrelated purposes.

---

9. **A TracingProcessor might use a TracingExporter internally.**
✓**a) True**
✗b) False

---

10. **If a TracingProcessor buffers spans for batch sending, which method would clear that buffer and send the data immediately?**
✓**c) force_flush()**
✗a) shutdown()
✗b) on_span_end()
✗d) on_trace_end()

---

11. **What is passed as an argument to `on_span_start()`?**
✓**c) The Span[Any] object that just started.**
✗a) The span_id string.
✗b) The SpanData object.
✗d) The Trace object that the span belongs to.

---

12. **The `on_trace_end()` method is typically where a TracingProcessor would get the complete data for a trace. Why?**
✓**b) Because at this point, all child spans should have finished, and the trace's full context is available.**

❌a) Because traces are stateless.
❌c) Because the trace ID is only generated at the end.
❌d) To prevent resource leaks.

---

13. **Which scenario would most benefit from implementing a custom TracingProcessor?**
✅**b) You want to send trace data to a specific third-party observability platform.**
❌a) You need to create a new type of span.
❌c) You want to define the hierarchy of spans.
❌d) You want to start and stop traces automatically.

---

14. **The TracingProcessor methods (`on_trace_start`, `on_trace_end`, etc.) are abstract because:**
✅**b) They define a contract that concrete processor implementations must fulfill.**
❌a) They are optional methods.
❌c) They are placeholders for future functionality.
❌d) They can be overridden by users at runtime.

---

15. **If a TracingExporter fails to send data to its backend (e.g., network error), what is generally the desired behavior?**
✅**c) To log the error and possibly retry later without blocking the main application thread.**
❌a) To crash the application.
❌b) To retry indefinitely, blocking the application.
❌d) To immediately disable all tracing.

---

16. **What does `Span[Any]` signify in the method signatures of TracingProcessor (e.g., `on_span_start(span: Span[Any])`)?**
✅**c) The span's generic `TSpanData` type parameter is not constrained, meaning it can be any concrete SpanData type.**
❌a) The span can be of any type, but its data is unknown.
❌b) The span's SpanData is optional.
❌d) The span is only for Any type of errors.

---

17. **If a TracingProcessor implements a queuing mechanism for spans, where would it typically add a span to the queue?**
✅**b) In on_span_end().**
❌a) In on_trace_start().
❌c) In shutdown().
❌d) In force_flush().

---

18. **TracingProcessor and TracingExporter are found in the `processor_interface.py` file, indicating they are:**
✅**c) Abstract interfaces defining contracts for custom implementations.**
❌a) Concrete implementations ready for direct use.
❌b) Utility functions for internal use only.
❌d) Configuration classes.

19. **Calling `trace.finish()` or `span.finish()` triggers which methods on registered TracingProcessors?**
✓**c) on_trace_end() or on_span_end() respectively.**
✗a) force_flush()
✗b) shutdown()
✗d) export()

---

20. **Is it possible for a single Trace to be processed by multiple TracingProcessors?**
✓**a) Yes, multiple processors can be registered with the TraceProvider.**
✗b) No, only one processor can handle a trace at a time.
✗c) Only if they implement different TracingExporters.
✗d) Only if the Trace is marked as `disabled=False`.

OpenAI Agents SDK - Concrete Processors & Exporters MCQs

1. **Which concrete TracingExporter is best suited for quickly viewing trace and span data during local development or debugging?**
✓**b) ConsoleSpanExporter**
✗a) BackendSpanExporter
✗c) BatchTraceProcessor
✗d) default_exporter()

---

2. **The BackendSpanExporter is designed to send tracing data via which protocol?**
✓**c) HTTP**
✗a) FTP
✗b) WebSocket
✗d) SMTP

---

3. **Which environment variable does BackendSpanExporter primarily check for authentication by default?**
✓**c) OPENAI_API_KEY**
✗a) TRACING_API_KEY
✗b) OPENAI_AUTH_TOKEN
✗d) BACKEND_TRACE_KEY

---

4. **What is the main benefit of BatchTraceProcessor's use of a background thread for exporting spans?**
✓**c) It minimizes performance impact on the main application thread by offloading network I/O.**
✗a) It allows for real-time, synchronous data transfer.
✗b) It simplifies the configuration of exporters.
✗d) It guarantees immediate delivery of all spans.

---

5. **If the queue size in a BatchTraceProcessor reaches max_queue_size, what might happen to new spans?**
✓**d) They might be dropped to prevent memory exhaustion.**
✗a) They are immediately exported individually.
✗b) They cause the processor to raise an exception.
✗c) They trigger a shutdown() call.

---

6. **The max_retries parameter in BackendSpanExporter is used to:**
✓**c) Specify the maximum number of attempts to resend a failed HTTP request.**
✗a) Limit the number of spans in a batch.
✗b) Define how many times the export() method can be called.
✗d) Set the maximum number of times a trace can be processed.

---

7. **What is the default endpoint for BackendSpanExporter?**
✓**c) https://api.openai.com/v1/traces/ingest**
✗a) http://localhost:8080/traces
✗b) https://api.openai.com/v1/traces/export
✗d) http://tracing.example.com/api

---

8. **The schedule_delay parameter in BatchTraceProcessor controls:**
✓**b) The interval (in seconds) at which the background thread checks the queue for spans to export.**
✗a) How long the main application thread pauses before generating new spans.
✗c) The delay before retrying a failed export.
✗d) The time before the processor shuts down automatically.

---

9. **Which method on BackendSpanExporter should be called when the application is exiting to ensure all resources are properly released?**
✓**a) shutdown()**
✗b) force_flush()
✗c) close()
✗d) disconnect()

---

10. **What does the export_trigger_ratio in BatchTraceProcessor help to achieve?**
✓**b) It defines a threshold (based on queue fullness) to trigger an immediate export, even if the scheduled delay hasn't passed.**
✗a) It sets the maximum time a span can stay in the queue.
✗c) It determines the proportion of spans that should be dropped.
✗d) It controls the growth of the exponential backoff delay.

---

11. **What is the primary advantage of BatchTraceProcessor over simply calling exporter.export() for every on_span_end event?**
✓**c) Reduced network overhead and improved application performance by batching requests.**
✗a) Simpler configuration.
✗b) Guaranteed real-time delivery.
✗d) Automatic error recovery without retries.

12. **The default_exporter() function returns an instance of which class?**
    ✅**b) BackendSpanExporter**
    ❌a) ConsoleSpanExporter
    ❌c) BatchTraceProcessor
    ❌d) TracingExporter (abstract class)

---

13. **The default_processor() function returns an instance of which class?**
    ✅**c) BatchTraceProcessor**
    ❌a) ConsoleSpanExporter
    ❌b) BackendSpanExporter
    ❌d) TracingProcessor (abstract class)

---

14. **If you want to configure the BackendSpanExporter with a specific OpenAI project ID, which parameter would you use in its constructor?**
    ✅**c) project**
    ❌a) organization
    ❌b) api_key
    ❌d) endpoint

---

15. **BackendSpanExporter implements retry logic for network requests. This feature primarily contributes to:**
    ✅**c) Improved reliability and fault tolerance.**
    ❌a) Faster data transfer.
    ❌b) Lower CPU utilization.
    ❌d) Reduced memory consumption.

---

16. **What kind of objects does ConsoleSpanExporter typically receive in its export() method?**
    ✅**c) Trace and Span objects (which it then converts to dictionaries for printing).**
    ❌a) Already formatted JSON strings.
    ❌b) Raw binary data.
    ❌d) File paths.

---

17. **If a BatchTraceProcessor receives many spans rapidly, and export_trigger_ratio is set to 0.9, what happens when the queue reaches 90% of max_queue_size?**
    ✅**c) An export attempt is triggered immediately.**
    ❌a) Spans are immediately dropped.
    ❌b) The processor halts.
    ❌d) The schedule_delay is increased.

---

18. **What is the type of object that BatchTraceProcessor takes as its primary argument in its constructor?**
    ✅**b) TracingExporter**
    ❌a) TracingProcessor

❌c) Span
❌d) Trace

---

19. **Calling force_flush() on a BatchTraceProcessor causes:**
    ✅c) Any queued spans/traces to be immediately sent through its exporter.
    ❌a) The background thread to stop.
    ❌b) All previously exported data to be deleted.
    ❌d) The schedule_delay to be reset.

---

20. **When setting up tracing for a production application that sends data to an OpenAI backend, which combination of components is generally recommended?**
    ✅c) default_processor() (which uses BatchTraceProcessor and default_exporter()).
    ❌a) ConsoleSpanExporter directly in application code.
    ❌b) BackendSpanExporter with manual export() calls.
    ❌d) Custom TracingProcessor and TracingExporter from scratch.

OpenAI Agents SDK - Scope MCQs

1. **What is the primary function of the Scope module in the tracing system?**
   ✅b) To manage the currently active span and trace in the execution context.
   ❌a) To manage API keys for backend exporters.
   ❌c) To define abstract interfaces for processors.
   ❌d) To export trace data to external systems.

---

2. **Why is the concept of a "current" span/trace, managed by Scope, important for tracing?**
   ✅c) It enables the automatic establishment of parent-child relationships for newly created spans.
   ❌a) It allows for faster processing of trace data.
   ❌b) It determines the output format of exported traces.
   ❌d) It defines the maximum depth of a trace.

---

3. **If you create a new span without explicitly providing a parent argument, how does the tracing system typically determine its parent?**
   ✅c) It uses the currently active span or trace managed by Scope as the parent.
   ❌a) It assigns a random parent from existing traces.
   ❌b) It creates a new root trace for the span.
   ❌d) The span will not be associated with any parent.

---

4. **Which Python mechanism is commonly used by Scope (or similar context management systems) to store the active trace/span for the current execution flow?**
   ✅c) Thread-local storage or contextvars
   ❌a) Global variables

❌b) Class attributes
❌d) Database sessions

---

5. **The get_current_trace() and get_current_span() utility functions directly rely on which module to retrieve the active tracing context?**
✅**c) Scope (implicitly, via the TraceProvider which uses it)**
❌a) TracingProcessor
❌b) TracingExporter
❌d) SpanData

---

6. **Without the functionality provided by Scope, how would you typically have to link spans to their parents?**
✅**c) You would need to explicitly pass the parent argument to every span creation function.**
❌a) All spans would automatically be top-level traces.
❌b) Spans would be linked by their name property.
❌d) Spans would only link to the workflow_name.

---

7. **The management of the "current" trace and span helps ensure that traces are:**
✅**c) Well-formed and their spans are correctly nested hierarchically.**
❌a) Always disabled for production.
❌b) Limited to a fixed number of spans.
❌d) Sorted alphabetically by their name.

---

8. **When a TraceImpl or SpanImpl calls start(mark_as_current=True), it interacts with Scope to:**
✅**c) Set itself as the active trace/span in the current context.**
❌a) Begin sending data to an exporter.
❌b) Calculate its duration.
❌d) Register a new processor.

---

9. **When a TraceImpl or SpanImpl calls finish(reset_current=True), it interacts with Scope to:**
✅**b) Remove itself as the active trace/span and potentially revert to its parent's context.**
❌a) Export its data.
❌c) Mark itself as an error.
❌d) Increase the max_queue_size.

---

10. **Is Scope typically something an end-user developer directly interacts with by calling its methods?**
✅**b) No, it's usually managed internally by the tracing library (e.g., via TraceImpl, SpanImpl, and context managers).**
❌a) Yes, it's frequently used for manual context switching.
❌c) Only during debugging.
❌d) Only when defining custom span types.

---

11. **Which of the following best describes Scope's role in making tracing "ergonomic"?**
✓**c) It simplifies instrumentation by automatically propagating the parent context, reducing boilerplate.**
❌a) It provides default values for all span parameters.
❌b) It optimizes the performance of network calls.
❌d) It filters out unnecessary trace data.

---

12. **If Scope fails to manage the current context correctly, what might be a visible symptom in your traced data?**
✓**b) Spans appearing as root spans when they should be children, or incorrect nesting.**
❌a) Traces being too short.
❌c) Exported data being corrupted.
❌d) Processors failing to shut down.

---

13. **The concept of Scope ensures that the tracing context is:**
✓**c) Specific to the current execution flow or thread.**
❌a) Globally immutable.
❌b) Shared across all processes.
❌d) Only active when a ConsoleSpanExporter is used.

---

14. **Scope contributes to ensuring that a Trace is a well-formed:**
✓**c) Tree (or directed acyclic graph - DAG)**
❌a) List
❌b) Flat sequence
❌d) Queue

---

15. **If a NoOpTrace or NoOpSpan is created, does Scope still manage its "current" status if mark_as_current=True is used?**
✓**b) No, NoOpTrace and NoOpSpan do not interact with the actual context management provided by Scope for recording.**
❌a) Yes, but it will still perform no-op operations.
❌c) Only if explicitly configured.
❌d) It depends on the Python version.

---

16. **Which parameter directly influences Scope's state when calling trace.start() or span.start()?**
✓**c) mark_as_current**
❌a) workflow_name
❌b) metadata
❌d) disabled

---

17. **The primary benefit of Scope is related to:**
✓**c) Context propagation.**
❌a) Data serialization.
❌b) Asynchronous operations.
❌d) Error handling.

18. **If a Trace is started with mark_as_current=True, and then a Span is started within that trace without an explicit parent, what object does Scope ensure the span is associated with?**
    ✅**c) The currently active Trace (if no Span is current).**
    ❌a) A new, independent trace.
    ❌b) The global root of all traces.
    ❌d) A randomly selected parent.

---

19. **The Scope mechanism helps in correctly attributing spans to their parent operations, which is vital for:**
    ✅**c) Visualizing the flow of execution and understanding dependencies.**
    ❌a) Reducing API call latency.
    ❌b) Compressing trace data.
    ❌d) Encrypting sensitive information.

---

20. **The abstract base class Trace and Span define methods (start, finish) that, when implemented by TraceImpl and SpanImpl, interact with Scope to:**
    ✅**c) Update the current active tracing context.**
    ❌a) Validate input parameters.
    ❌b) Choose the correct exporter.
    ❌d) Manage concurrent access to resources.

OpenAI Agents SDK - Trace Provider Setup MCQs

1. **What is the main purpose of set_trace_provider()?**
   ✅**c) To globally configure the central TraceProvider instance for the tracing utilities.**
   ❌a) To create a new trace.
   ❌b) To get the currently active span.
   ❌d) To export trace data immediately.

---

2. **The TraceProvider is described as the "central, overarching component" responsible for:**
   ✅**c) Creating Trace and Span objects, and managing the tracing context.**
   ❌a) Only printing traces to the console.
   ❌b) Only sending data to a backend.
   ❌d) Implementing on_trace_start and on_span_end.

---

3. **If set_trace_provider() is called, which subsequent tracing operation will NOT use the newly set provider?**
   ✅**d) All of the above will use the newly set provider.**
   ❌a) tracing.trace()
   ❌b) tracing.agent_span()
   ❌c) tracing.get_current_trace()

---

4. **Why is it beneficial to be able to set_trace_provider()?**
   ✅**c) It allows for flexible configuration and dependency injection of tracing behavior.**

❌a) It makes traces run faster.
❌b) It reduces the memory footprint of traces.
❌d) It automatically handles all network retries.

---

5. **What type of object does set_trace_provider() expect as its provider argument?**
✅**c) TraceProvider**
❌a) TracingProcessor
❌b) TracingExporter
❌d) SpanData

---

6. **What does get_trace_provider() return?**
✅**c) The currently configured global TraceProvider instance.**
❌a) A new TraceProvider instance every time.
❌b) The default NoOpTraceProvider.
❌d) A list of all active traces.

---

7. **A common use case for get_trace_provider() is:**
✅**c) To inspect or potentially dynamically modify the behavior of the active provider (e.g., add/remove processors).**
❌a) To directly create a new Trace without using tracing.trace().
❌b) To clear all existing trace data.
❌d) To force a flush of all pending exports.

---

8. **If set_trace_provider() has not been explicitly called, what kind of TraceProvider might get_trace_provider() return by default in a typical tracing setup?**
✅**b) A default TraceProvider (e.g., TraceProviderImpl) or a NoOpTraceProvider if tracing is off.**
❌a) A ConsoleSpanExporter.
❌c) An error because no provider is set.
❌d) None.

---

9. **The functions in setup.py emphasize the importance of having a single, global TraceProvider.**
✅**a) True**
❌b) False

---

10. **What is the primary benefit of set_trace_provider() in terms of testing your application?**
✅**c) It allows you to inject mock or disabled TraceProvider instances for isolated testing.**
❌a) It automatically generates test cases.
❌b) It highlights performance bottlenecks.
❌d) It provides a built-in testing framework.

---

11. **Where in an application's lifecycle would set_trace_provider() typically be called?**
✅**b) During application initialization or startup.**

✗a) Inside every function that creates a span.
✗c) Immediately before the application shuts down.
✗d) Only when an error occurs.

---

12. **If set_trace_provider(NoOpTraceProvider()) is called, what will be the effect on tracing.trace() and tracing.agent_span() calls?**
    ✓**c) They will return NoOpTrace and NoOpSpan objects, effectively disabling tracing.**
    ✗a) They will continue to record data but won't export it.
    ✗b) They will raise an exception.
    ✗d) They will buffer all traces and spans indefinitely.

---

13. **get_trace_provider() is used to access the provider, not to create it.**
    ✓**a) True**
    ✗b) False

---

14. **Why is managing the TraceProvider globally via set_trace_provider and get_trace_provider preferable to passing it around as an argument to every function?**
    ✓**b) It simplifies application code by providing a globally accessible tracing context without explicit argument passing.**
    ✗a) It makes the code more complex.
    ✗c) It only works for single-threaded applications.
    ✗d) It reduces the number of TracingProcessors.

---

15. **If you have a custom TraceProvider implementation that includes unique debugging methods, how would you access those methods after setting your custom provider?**
    ✓**c) You can retrieve it using get_trace_provider() and then call your custom methods.**
    ✗a) You cannot, only standard methods are accessible.
    ✗b) You must cast the result of get_trace_provider() to your custom type.
    ✗d) Custom methods are automatically called.

---

16. **What kind of parameters does set_trace_provider() take?**
    ✓**c) An instance of a class that implements TraceProvider.**
    ✗a) A string representing the provider name.
    ✗b) A dictionary of configuration settings.
    ✗d) A boolean to enable/disable tracing.

---

17. **If a TraceProvider is not set, attempts to create traces or spans using the tracing utilities might:**
    ✓**b) Result in NoOpTrace or NoOpSpan objects, or potentially an error depending on the default provider's behavior.**
    ✗a) Automatically set up a default, fully functional provider.
    ✗c) Crash the application immediately.
    ✗d) Export data to a temporary file.

---

18. **The setup.py module essentially provides the API for:**
✓**c) Bootstrapping the tracing system within an application.**
✗a) Defining new SpanData structures.
✗b) Implementing TracingProcessor logic.
✗d) Converting trace data to JSON.

---

19. **Calling set_trace_provider() multiple times with different providers during an application's runtime would:**
✓**b) Change the active TraceProvider to the most recently set one, potentially altering tracing behavior mid-execution.**
✗a) Be an error, as it can only be set once.
✗c) Cause all previous providers to flush their data.
✗d) Have no effect after the first call.

---

20. **Which of the following is NOT a direct responsibility of the set_trace_provider() or get_trace_provider() functions themselves?**
✓**c) Performing the actual collection or export of trace data.**
✗a) Allowing global configuration of tracing.
✗b) Providing access to the current TraceProvider.
✗d) Enabling dependency injection for the tracing core.

OpenAI Agents SDK - SpanData MCQs

1. What is the primary role of SpanData in the tracing module?

✗a) To manage the parent-child relationships of spans.

✗b) To define the start and end times of a span.

✓**c) To represent the specific payload or contextual attributes of a particular type of span.**

✗d) To export traces to a backend.

---

2. Which of the following is an abstract property of the SpanData class?

✗a) duration

✗b) parent_id

✓**c) type**

❌d) timestamp

---

3. The export() method on SpanData is responsible for:

❌a) Sending the data to a TracingExporter.

✅**b) Serializing the span's specific data into a dictionary format.**

❌c) Printing the span data to the console.

❌d) Calculating the cost of the span.

---

4. Which SpanData implementation would typically include input messages, output messages, model, and usage?

❌a) FunctionSpanData

❌b) ResponseSpanData

✅**c) GenerationSpanData**

❌d) AgentSpanData

---

5. If you want to track the arguments passed to and results returned from a custom tool your agent calls, which SpanData type would be most appropriate?

❌a) AgentSpanData

✅**b) FunctionSpanData**

❌c) CustomSpanData

❌d) MCPListToolsSpanData

---

6. Which SpanData type is designed for flexible, user-defined operations where you can attach arbitrary key-value pairs?

❌a) AgentSpanData

❌b) FunctionSpanData

✅**c) CustomSpanData**

❌d) SpeechGroupSpanData

---

7. A HandoffSpanData specifically captures information about:

✗a) Data transfer between a model and a database.

✓**b) The transfer of control or responsibility between different agents.**

✗c) The initial user prompt.

✗d) The final response generated by the agent.

---

8. What key piece of information does GuardrailSpanData typically record?

✗a) The LLM's confidence score.

✗b) The duration of the guardrail check.

✓**c) The name of the guardrail and its triggered status (boolean).**

✗d) The policy document used by the guardrail.

---

9. TranscriptionSpanData and SpeechSpanData are specifically for operations related to:

✗a) Image processing.

✗b) Data parsing.

✓**c) Speech-to-text and text-to-speech.**

✗d) Database queries.

---

10. Which SpanData type would be used to represent the overall activity of an agent, potentially encompassing multiple internal steps?

✓**a) AgentSpanData**

✗b) CustomSpanData

✗c) ResponseSpanData

✗d) SpeechGroupSpanData

---

11. The ResponseSpanData object primarily focuses on:

✗a) The agent's internal reasoning steps.

✗b) Tool calls made by the agent.

✓**c) The final output or message generated by the agent for the user/system.**

❌d) The configuration of the agent.

---

12. The type property of a SpanData object allows consumers of trace data to:

❌a) Determine the original source file of the span.

✅**b) Understand the specific kind or category of operation represented by the span.**

❌c) Authenticate the span's origin.

❌d) Calculate the exact duration of the span.

---

13. MCPListToolsSpanData would be used when an agent interacts with a system to:

❌a) Execute a specific tool.

❌b) Monitor MCP server health.

✅**c) Retrieve a list of available tools from the MCP server.**

❌d) Configure the MCP server.

---

14. What distinguishes SpanData from Span itself?

❌a) SpanData manages lifecycle; Span carries payload.

✅**b) Span manages lifecycle and hierarchy; SpanData carries the specific, contextual payload.**

❌c) SpanData is abstract; Span is concrete.

❌d) SpanData is only for errors; Span is for successes.

---

15. If a Span is created using tracing.generation_span(), what type of SpanData object will it typically hold internally?

❌a) AgentSpanData

❌b) FunctionSpanData

✅**c) GenerationSpanData**

❌d) CustomSpanData

---

16. What is the benefit of having a generic Span[TSpanData] for the Span class, instead of just Span[Any]?

❌a) It makes the code run faster.

✗b) It allows for different export formats.

✓**c) It provides type-safety, ensuring that a Span instance is associated with the correct, structured data for its type.**

✗d) It disables tracing if the data type is incorrect.

---

17. The SpeechGroupSpanData suggests a purpose of:

✗a) Recording individual spoken words.

✗b) Storing audio file paths.

✓**c) Grouping related speech operations, possibly representing a complete spoken turn or interaction segment.**

✗d) Translating speech into different languages.

---

18. All concrete SpanData implementations inherit from SpanData and must implement its abstract methods/properties.

✓**a) True**

✗b) False

---

19. Which method is consistently defined across all SpanData implementations for serialization purposes?

✗a) to_json()

✓**b) export()**

✗c) serialize()

✗d) get_data()

---

20. When would you typically instantiate a SpanData object directly in your application code?

✗a) Every time you call span.start().

✓**b) You generally don't; you use tracing helper functions (e.g., tracing.agent_span) which create the appropriate SpanData internally.**

✗c) Only when using ConsoleSpanExporter.

✗d) When retrieving data from a backend.

OpenAI Agents SDK - Utility Functions MCQs

1. What is the primary purpose of the `time_iso()` function?

❌a) To calculate the duration of a span.

❌b) To set the system's time zone.

✅**c) To return the current time in ISO 8601 format.**

❌d) To measure CPU usage.

---

2. Why is using ISO 8601 format important for timestamps in tracing?

❌a) It makes the timestamps shorter.

❌b) It encrypts the time information.

✅**c) It ensures consistent ordering, accurate duration calculation, and interoperability with other systems.**

❌d) It only records local time.

---

3. The `gen_trace_id()` function generates an ID that is primarily used for:

❌a) Identifying individual operations within a function.

❌b) Linking a span to its direct parent.

✅**c) Correlating all spans that belong to a single logical workflow.**

❌d) Categorizing different types of spans.

---

4. What characteristic is crucial for a `trace_id` to ensure proper correlation across distributed services?

❌a) It must be a short integer.

❌b) It must be human-readable.

✅**c) It must be globally unique (or have a very high probability of uniqueness).**

❌d) It must be sorted alphabetically.

---

5. The `gen_span_id()` function generates an ID that is primarily used for:

❌a) Linking multiple traces together.

✅**b) Uniquely identifying an individual operation (unit of work) within a trace.**

✗c) Specifying the type of data within a span.

✗d) Setting the name of a span.

---

6. Which function is used to link multiple distinct traces that belong to a broader, ongoing conversation or process?

✗a) gen_trace_id()

✗b) gen_span_id()

✓**c) gen_group_id()**

✗d) time_iso()

---

7. A `group_id` differs from a `trace_id` in that a `group_id` links:

✗a) Spans to other spans.

✗b) Spans to their trace.

✓**c) Traces to other traces within a larger logical conversation.**

✗d) Workflows to external systems.

---

8. If `time_iso()` returns `2025-06-24T06:39:01.123456Z`, what does the `Z` at the end signify?

✗a) Zone-specific time.

✗b) Zero milliseconds.

✓**c) Zulu time (UTC).**

✗d) Daylight Saving Time.

---

9. Which of these functions typically produces a UUID or similar high-entropy random string?

✗a) time_iso()

✗b) ConsoleSpanExporter

✓**c) gen_trace_id() and gen_span_id()**

✗d) set_trace_provider()

---

10. What is a primary use case for `gen_group_id()` in an AI assistant application?

✗a) To uniquely identify each LLM call.

✗b) To identify individual tool invocations.

✓**c) To link all traces from a single, multi-turn conversational session.**

✗d) To generate names for new agents.

---

11. The utility functions in `util.py` are essential because they provide the necessary:

✗a) Configuration settings

✗b) Network communication

✓**c) Unique identifiers and temporal context**

✗d) Data compression algorithms

---

12. The output of `time_iso()` is a:

✗a) datetime object

✗b) float representing milliseconds since epoch

✓**c) str**

✗d) int representing Unix timestamp

---

13. If `gen_span_id()` is called twice within the same trace, what is guaranteed about the two IDs generated?

✗a) They will be identical.

✗b) They will be sequential numbers.

✓**c) They will be unique.**

✗d) They will share the same prefix.

---

14. `gen_trace_id()`, `gen_span_id()`, and `gen_group_id()` are designed to create IDs that are:

✗a) Short and easy to remember.

✗b) Always sequential.

✓**c) Highly unlikely to collide (be duplicate).**

✗d) Directly convertible to integers for arithmetic.

---

15. When a Span finishes, its duration is calculated using its start and end timestamps. These timestamps are typically formatted using which utility function's output?

✗a) gen_trace_id()

✓**b) time_iso()**

✗c) gen_span_id()

✗d) gen_group_id()

---

16. What would be a potential problem if `gen_trace_id()` did not produce globally unique IDs?

✗a) Spans would not be able to link to their parent.

✗b) TracingProcessors would fail to export data.

✓**c) It would be difficult to correctly correlate and view all operations belonging to a single workflow, especially in distributed systems.**

✗d) The BackendSpanExporter would use the wrong endpoint.

---

17. The `util.py` module contains:

✗a) Concrete TraceProvider implementations

✗b) Abstract interfaces for tracing

✓**c) Helper functions for ID generation and time formatting**

✗d) Classes for managing tracing context

---

18. If you were building a custom trace visualization tool, what would be the most important piece of information to properly reconstruct the trace hierarchy?

✗a) Only trace_id

✗b) Only span_id

✓**c) trace_id, span_id, and a parent_span_id (or direct parent reference)**

✗d) group_id and time_iso()

---

19. Which function would you *not* expect to directly use `time_iso()` in its internal logic?

❌a) TraceImpl.start()

❌b) SpanImpl.finish()

✅**c) gen_span_id() (ID generation is separate from timestamping)**

❌d) BatchTraceProcessor (for scheduling or flushing)

---

20. The functions in `util.py` are generally meant to be:

❌a) Modified by end-users

❌b) Used only for internal testing

✅**c) Called by the tracing library's core components (e.g., TraceImpl, SpanImpl) to ensure consistency**

❌d) Passed as arguments to TracingExporters

Voice

OpenAI Agents SDK - VoicePipeline MCQs

1. What are the three main steps orchestrated by the VoicePipeline?

- ❌a) Text-to-Text, Voice Recognition, Audio Saving
- ❌b) Audio Playback, Image Generation, Text Analysis
- ✅**c) Transcribe audio to text, Run workflow (text-to-text), Convert text to streaming audio**
- ❌d) Record Audio, Process Video, Send Email

---

2. The VoicePipeline is described as "opinionated." What does this imply?

- ❌a) It allows for complete customization of every internal component
- ❌b) It provides recommendations but no concrete implementation
- ✅**c) It offers a predefined, common structure suitable for typical voice agent use cases**
- ❌d) It only works with specific voice assistants

---

3. Which component is explicitly required when initializing a VoicePipeline?

- ✗a) stt_model
- ✗b) tts_model
- ✓**c) workflow**
- ✗d) config

---

4. The `workflow` parameter in VoicePipeline's constructor must be an instance of a class inheriting from:

- ✗a) AudioInput
- ✗b) STTModel
- ✓**c) VoiceWorkflowBase**
- ✗d) StreamedAudioResult

---

5. What happens if `stt_model` is not provided when creating a VoicePipeline?

- ✗a) The pipeline will only work with text input
- ✓**b) An error will be raised**
- ✗c) A default OpenAI Speech-to-Text model will be used
- ✗d) The pipeline will automatically try to detect the model

---

6. The `run()` method of VoicePipeline is an async method. Why is this significant?

- ✗a) It means the method runs on a separate CPU core
- ✓**b) It indicates support for non-blocking operations, essential for real-time voice interactions**
- ✗c) It signifies that it only processes static audio files
- ✗d) It requires a specific file system for operation

---

7. Which type of audio input allows for continuous feeding of audio data to the VoicePipeline?

- ✗a) AudioInput
- ✓**b) StreamedAudioInput**
- ✗c) AudioBuffer
- ✗d) StaticAudio

---

8. What is the return type of the VoicePipeline.run() method?

- ✗a) str (the transcribed text)
- ✗b) AudioInput (the processed audio)
- ✓**c) StreamedAudioResult (for playing audio as it's generated)**
- ✗d) dict (containing pipeline statistics)

---

9. The ability to stream audio output via `StreamedAudioResult` primarily helps to:

- ✗a) Reduce network bandwidth usage
- ✓**b) Improve the perceived responsiveness of the agent by playing audio as it's synthesized**
- ✗c) Ensure perfect audio fidelity
- ✗d) Enable offline processing

---

10. What is the role of `VoiceWorkflowBase` within the VoicePipeline?

- ✗a) To perform speech recognition
- ✗b) To convert text to audio
- ✓**c) To act as the core logic or "brain" of the agent, processing text and generating text responses**
- ✗d) To manage audio input/output devices

---

11. The `VoicePipelineConfig` parameter allows for configuration of:

- ✗a) The specific workflow to use
- ✗b) The api_key for models
- ✓**c) Technical parameters like audio sample rates or streaming chunk sizes**
- ✗d) The name of the voice agent

---

12. If a VoicePipeline is created without specifying `tts_model`, what will be used?

- ✗a) No text-to-speech conversion will occur
- ✗b) An error will be thrown during run()
- ✓**c) A default OpenAI Text-to-Speech model**
- ✗d) A console-based text output

---

13. `AudioInput` is used for:

- ✗a) Live microphone input
- ✓**b) A single, static buffer of audio data**
- ✗c) A continuous stream of audio
- ✗d) Textual input converted to audio

---

14. The VoicePipeline directly encapsulates the logic for:

- ✗a) Database interactions
- ✗b) Sending emails
- ✓**c) The complete audio-to-text-to-audio conversational loop**
- ✗d) User authentication

---

15. What is the benefit of an "opinionated" pipeline design for developers?

- ❌a) It forces developers to implement all components from scratch
- ✅**b) It provides a quick and robust starting point for common use cases, reducing initial setup complexity**
- ❌c) It makes it harder to integrate with external services
- ❌d) It removes the possibility of using different STT/TTS models

---

16. Can `stt_model` and `tts_model` be provided as simple string names (e.g., "whisper") instead of explicit model objects?

- ✅**a) Yes**
- ❌b) No, they must be full model objects

---

17. The VoicePipeline is found in `src/agents/voice/pipeline.py`, indicating its focus on:

- ❌a) General purpose AI agents
- ❌b) Multi-modal agents combining text and images
- ✅**c) Agents primarily interacting via spoken language**
- ❌d) Backend data processing

---

18. If a VoiceWorkflowBase produces multiple text responses sequentially, how does VoicePipeline handle them for output?

- ❌a) It concatenates them into a single response
- ❌b) It only uses the first response
- ❌c) It discards all but the last response
- ✅**d) It converts each text response into streaming audio, sending them in sequence**

---

19. Which of these is NOT a direct function of the VoicePipeline itself, but rather part of the workflow it runs?

- ❌a) Transcribing audio
- ❌b) Converting text to speech
- ✅**c) Deciding what to say based on the transcribed text (e.g., calling tools, reasoning)**
- ❌d) Managing audio input streams

---

20. The VoicePipeline's design aims to facilitate:

- ❌a) Batch processing of large text documents
- ❌b) Offline training of AI models
- ✅**c) Building real-time, interactive conversational voice agents**
- ❌d) Scientific data analysis

- # OpenAI Agents SDK - Voice Workflow MCQs

1. What is the main purpose of `VoiceWorkflowBase`?

- ✗a) To manage audio input/output devices
- ✗b) To convert audio to text
- ✓**c) To define the interface for the core logic ("brain") of a voice agent**
- ✗d) To handle network communication for models

---

2. The `run` method of `VoiceWorkflowBase` receives `transcription: str` as input. What does this transcription represent?

- ✗a) The agent's generated text response
- ✗b) A configuration setting for the workflow
- ✓**c) The text converted from the user's spoken audio**
- ✗d) A unique ID for the workflow

---

3. The `run` method of `VoiceWorkflowBase` returns an `AsyncIterator[str]`. What is the significance of `AsyncIterator` and `str`?

- ✗a) It means the workflow returns a single, complete string asynchronously
- ✗b) It means the workflow processes a list of strings in a batch
- ✓**c) It allows the workflow to stream multiple textual responses over time, enabling incremental text-to-speech**
- ✗d) It indicates that the workflow only handles error messages

---

4. What is the recommended approach for implementing the `run` method of a custom `VoiceWorkflowBase`?

- ✗a) Directly call a Text-to-Speech model
- ✗b) Perform complex database operations
- ✓**c) Call `Runner.run_streamed()` with an `Agent` and yield text events from its result**
- ✗d) Save the transcription to a file

---

5. What is the primary function of `VoiceWorkflowHelper.stream_text_from()`?

- ✗a) To transcribe audio from a `RunResultStreaming`
- ✓**b) To extract and yield only the text events from a `RunResultStreaming` object**
- ✗c) To convert text events into audio
- ✗d) To manage the history of an agent

---

6. `SingleAgentVoiceWorkflow` is a concrete implementation of which abstract base class?

- ✗a) VoicePipeline
- ✓**b) VoiceWorkflowBase**
- ✗c) TracingProcessor
- ✗d) Agent

---

7. For which scenario is `SingleAgentVoiceWorkflow` most suitable?

- ✗a) Workflows requiring multiple `Runner` calls
- ✓**b) Simple voice agents that directly wrap a single `Agent` instance**
- ✗c) Workflows with complex custom message history logic
- ✗d) Workflows that only perform speech-to-text

---

8. What key conversational feature does `SingleAgentVoiceWorkflow` provide out-of-the-box?

- ✗a) Advanced error detection
- ✗b) Real-time language translation
- ✓**c) Automatic addition of transcription and agent result to the agent's input history**
- ✗d) Dynamic model selection

---

9. The `SingleAgentWorkflowCallbacks.on_run()` method is called:

- ✗a) After the agent has finished its response
- ✗b) When the VoicePipeline starts
- ✓**c) When the `SingleAgentVoiceWorkflow` begins processing a new transcription**
- ✗d) Periodically during the workflow's execution

---

10. If you need to implement a voice agent that involves dynamic branching logic based on the user's input, which class should you directly subclass?

- ✗a) SingleAgentVoiceWorkflow
- ✗b) VoiceWorkflowHelper
- ✓**c) VoiceWorkflowBase**
- ✗d) Agent

---

11. What is the type of the main component passed to the constructor of `SingleAgentVoiceWorkflow`?

- ✗a) VoicePipelineConfig
- ✓**b) Agent[Any]**
- ✗c) AsyncIterator[str]
- ✗d) SingleAgentWorkflowCallbacks

---

12. The `AsyncIterator` return type of `VoiceWorkflowBase.run()` directly supports which feature of `VoicePipeline`?

- ✗a) Background processing of audio
- ✗b) Synchronous execution
- ✓**c) Streaming audio output (Text-to-Speech)**
- ✗d) Batch processing of transcriptions

---

13. `VoiceWorkflowHelper.stream_text_from()` is a classmethod because:

- ✗a) It needs access to instance-specific data
- ✓**b) It operates on a `RunResultStreaming` object and doesn't require an instance of `VoiceWorkflowHelper` itself**
- ✗c) It needs to be overridden by subclasses
- ✗d) It handles global configuration

---

14. What does the term "workflow" signify in the context of `VoiceWorkflowBase`?

- ✗a) A predefined sequence of audio files
- ✗b) The process of training a new AI model
- ✓**c) Any code logic that receives a transcription and yields text to be spoken**
- ✗d) A data storage mechanism

---

15. `SingleAgentVoiceWorkflow` is described as a "simple" workflow. This implies it might lack advanced features like:

- ✗a) The ability to run an agent
- ✗b) Text-to-speech conversion
- ✓**c) Custom message history management or multiple calls to `Runner` within one turn**
- ✗d) Basic conversational turns

---

16. The `VoiceWorkflowBase` provides the interface for the text-to-text core of the voice agent.

- ✓**a) True**
- ✗b) False

---

17. If a `VoiceWorkflowBase` implementation yields multiple strings, how does the `VoicePipeline` handle them?

- ✗a) It concatenates them into one long string
- ✗b) It ignores all but the first string
- ✓**c) It converts each yielded string into a segment of streaming audio in sequence**
- ✗d) It buffers them until the entire conversation is complete

---

18. `SingleAgentWorkflowCallbacks` provides `on_run()` method to allow developers to insert custom logic at what point?

- ✗a) After the agent produces its final response
- ✗b) When the entire pipeline shuts down
- ✓**c) At the very beginning of the `SingleAgentVoiceWorkflow`'s execution for a new transcription**
- ✗d) When the text-to-speech model starts

---

19. Why might a developer choose to subclass `VoiceWorkflowBase` instead of using `SingleAgentVoiceWorkflow`?

- ✗a) To avoid using any agents
- ✗b) To only process static audio input
- ✓**c) To implement custom multi-agent orchestration, complex state management, or conditional logic**
- ✗d) To print all trace data to the console

---

20. The `VoiceWorkflowBase`'s design, with its `AsyncIterator[str]` return type, aligns with the concept of:

- ✗a) Batch processing
- ✓**b) Stream processing and responsiveness**
- ✗c) Synchronous blocking operations
- ✗d) One-shot function calls

:

OpenAI Agents SDK - Audio Input MCQs

1. Which class is designed for providing static, complete audio data to the VoicePipeline?

- ✗a) StreamedAudioInput
- ✓**b) AudioInput**
- ✗c) AudioBuffer
- ✗d) VoicePipelineConfig

---

2. What type of data is primarily stored in the `buffer` attribute of `AudioInput`?

- ✗a) A Python list of integers
- ✗b) A base64 encoded string
- ✓**c) A NumPy array (NDArray) of int16 or float32**
- ✗d) A raw bytes object without format

---

3. The `frame_rate` attribute of `AudioInput` defaults to:

- ✗a) 8000 Hz
- ✗b) 16000 Hz
- ✓**c) 24000 Hz (implied by `DEFAULT_SAMPLE_RATE`)**
- ✗d) 44100 Hz

---

4. Which method of `AudioInput` would you use if you needed to send the audio data as part of a JSON payload that expects embedded binary data?

- ✗a) to_audio_file()
- ✓**b) to_base64()**
- ✗c) get_buffer()
- ✗d) save_to_disk()

---

5. What is the primary use case for `StreamedAudioInput`?

- ✗a) Processing a pre-recorded audio file from a database
- ✓**b) Handling real-time, continuous audio input from sources like a live microphone**
- ✗c) Converting text to speech
- ✗d) Analyzing audio for specific keywords in a batch

---

6. Which method is used to add new audio data to an ongoing `StreamedAudioInput`?

- ✗a) set_buffer()
- ✗b) add_audio()
- ✓**c) append_audio_data()**
- ✗d) push_audio()

---

7. The `add_audio()` method in `StreamedAudioInput` is async. This indicates its suitability for:

- ✗a) Blocking the main thread until audio is fully processed
- ✓**b) Non-blocking operations, crucial for maintaining responsiveness in real-time applications**
- ✗c) Synchronous file I/O
- ✗d) Executing only once per pipeline run

---

8. If you have a `.wav` file stored on disk that you want to process through `VoicePipeline`, which input class would you typically use?

- ✗a) StreamedAudioInput
- ✓**b) AudioInput**
- ✗c) Both equally
- ✗d) Neither, you'd need a custom input handler

---

9. The `to_audio_file()` method returns a tuple containing filename, bytes (as BytesIO), and content_type. This is useful for:

- ✗a) Performing in-memory audio analysis
- ✗b) Directly playing the audio through speakers
- ✓**c) Preparing the audio for API calls that expect file-like uploads (e.g., as multipart/form-data)**
- ✗d) Compressing the audio data

---

10. What does `sample_width: int = 2` typically signify for audio data in `AudioInput`?

- ✗a) The number of audio channels
- ✓**b) That each audio sample occupies 2 bytes (e.g., 16-bit audio)**
- ✗c) The audio's sampling rate
- ✗d) The duration of the audio in seconds

---

11. The `channels` attribute of `AudioInput` represents:

- ✗a) The frequency range of the audio
- ✓**b) The number of audio channels (e.g., mono or stereo)**
- ✗c) The depth of the audio buffer
- ✗d) The volume level

---

12. A key difference between `AudioInput` and `StreamedAudioInput` is that `AudioInput` deals with audio that is:

- ✗a) Always very short
- ✗b) Processed only in the background
- ✓**c) Entirely available upfront**
- ✗d) Encrypted

---

13. What would be a scenario where `StreamedAudioInput` is indispensable?

- ✗a) Analyzing a pre-recorded podcast
- ✗b) Converting a large audio file to text offline
- ✓**c) Building a voice assistant that responds while the user is still speaking**
- ✗d) Generating an audio summary from a textual transcript

---

14. `AudioInput` is defined as a dataclass. This implies its primary role is to:

- ✗a) Provide complex business logic
- ✓**b) Serve as a simple container for data with auto-generated methods like `__init__`, `__repr__`, etc.**
- ✗c) Manage external file system operations
- ✗d) Implement abstract methods

---

15. Both `AudioInput` and `StreamedAudioInput` expect audio data in their buffer or add_audio methods to be:

- ✗a) Raw Python lists
- ✗b) Base64 encoded strings
- ✓**c) NumPy arrays (NDArray)**
- ✗d) Bytes objects

---

16. Which of the following is NOT a characteristic of `StreamedAudioInput`?

- ✓**c) It is typically used for pre-recorded, static audio files**
- ✗a) It supports pushing audio chunks incrementally
- ✗b) Its add_audio method is asynchronous
- ✗d) It is designed for real-time audio processing

---

17. If `to_base64()` were called on an `AudioInput` object containing 16-bit mono audio at 24kHz for 1 second, the resulting string would be:

- ✗a) A very short string, as it's compressed
- ✗b) Always `"audio_data_base64"`
- ✓**c) A longer string, as base64 encoding expands binary data by about 33%**
- ✗d) An integer value representing the audio length

---

18. The `DEFAULT_SAMPLE_RATE` is a common audio characteristic that defines:

- ✗a) The volume level
- ✓**b) How many samples per second are taken from an analog audio signal**
- ✗c) The number of channels
- ✗d) The bit depth of each sample

---

19. Which class would you use to provide input to the `VoicePipeline` if your audio comes from a continuous byte stream without a known end?

- ✗a) AudioInput (after buffering the entire stream)
- ✓**b) StreamedAudioInput**
- ✗c) VoicePipelineConfig
- ✗d) STTModel

---

20. The functions `to_audio_file()` and `to_base64()` of `AudioInput` serve as:

- ✗a) Primary methods for audio playback
- ✗b) Methods for altering the audio content
- ✓**c) Utility methods for different ways of representing or transmitting the static audio data**
- ✗d) Internal-only methods for pipeline processing

OpenAI Agents SDK - StreamedAudioResult MCQs

. What is `StreamedAudioResult` designed to be?

- ✗a) The input to the VoicePipeline.
- ✗b) A configuration object for TTS models.

- ✅**c) The output of a VoicePipeline, streaming events and audio.**
- ❌d) A class for recording audio from a microphone.

---

2. The primary benefit of `StreamedAudioResult`'s streaming nature is:

- ❌a) Reduced overall processing time.
- ❌b) Simplified debugging.
- ✅**c) Improved perceived latency and responsiveness in conversational AI.**
- ❌d) Guaranteed perfect audio fidelity.

---

3. Which type of model is explicitly required in the `StreamedAudioResult` constructor?

- ❌a) STTModel
- ✅**b) TTSModel**
- ❌c) Agent
- ❌d) VoiceWorkflowBase

---

4. Why does `StreamedAudioResult` need a `TTSModel` in its constructor?

- ❌a) To manage the conversation history.
- ❌b) To transcribe the input audio.
- ✅**c) Because `StreamedAudioResult` is responsible for converting the workflow's text output into audio.**
- ❌d) To generate unique IDs for audio segments.

---

5. What is the return type of the `StreamedAudioResult.stream()` method?

- ❌a) str (a single audio file path)
- ❌b) bytes (a complete audio buffer)
- ✅**c) AsyncIterator[VoiceStreamEvent]**
- ❌d) dict (containing audio metadata)

---

6. `VoiceStreamEvent` objects yielded by `stream()` can represent:

- ❌a) Only raw audio bytes.
- ❌b) Only error messages.
- ✅**c) Chunks of audio data, textual parts of the response, and end-of-speech signals.**
- ❌d) Configuration changes during streaming.

---

7. The `stream()` method being async implies it's designed for:

- ❌a) Synchronous, blocking execution.
- ❌b) Batch processing of large audio files.
- ✅**c) Non-blocking, real-time delivery of events.**
- ❌d) Offloading computation to a different process.

---

8. Which parameter in the `StreamedAudioResult` constructor dictates specific voice characteristics like speaking rate or pitch?

- ✗a) tts_model
- ✓**b) tts_settings**
- ✗c) voice_pipeline_config
- ✗d) workflow

---

9. What would typically cause a `VoiceStreamEvent` of type `AudioEvent` to be yielded?

- ✗a) A new user transcription is received.
- ✗b) The VoicePipeline is initialized.
- ✓**c) A chunk of text from the workflow has been synthesized into audio by the TTS model.**
- ✗d) The agent calls a tool.

---

10. If the `VoiceWorkflowBase` yields text incrementally, how does `StreamedAudioResult` leverage this?

- ✗a) It waits for all text to be available before starting TTS.
- ✗b) It discards all but the last piece of text.
- ✓**c) It converts each yielded text chunk into audio in real-time, sending it as `AudioEvents`.**
- ✗d) It sends the text directly to the user without conversion.

---

11. The `voice_pipeline_config` parameter in `StreamedAudioResult`'s constructor is relevant for:

- ✗a) Defining the agent's behavior logic.
- ✗b) Setting the API key for the TTS model.
- ✓**c) Providing overall pipeline settings like audio format or chunking for the output stream.**
- ✗d) Managing the input audio buffer.

---

12. A `StreamedAudioResult` is designed to be consumed:

- ✗a) Only once, then it's exhausted.
- ✓**b) As an asynchronous stream, allowing an external client to process events as they arrive.**
- ✗c) By saving all its content to a file first.
- ✗d) Through synchronous blocking calls.

---

13. What kind of event might signal to the consuming application that the agent has finished its spoken response for the current turn?

- ✗a) An AudioEvent with an empty buffer.
- ✗b) A TextEvent with an empty string.
- ✓**c) An `EndOfSpeechEvent` (or similar signal within `VoiceStreamEvent`).**
- ✗d) A TimeoutEvent.

14. The `StreamedAudioResult` effectively acts as the orchestrator for which step of the VoicePipeline?

- ✗a) Transcription (STT).
- ✗b) Workflow execution.
- ✓**c) Text-to-Speech (TTS) and output streaming.**
- ✗d) Input audio buffering.

15. What is `StreamedAudioResult`'s relationship to `VoicePipeline`?

- ✗a) `StreamedAudioResult` is a component inside the VoicePipeline.
- ✗b) `StreamedAudioResult` configures the VoicePipeline.
- ✓**c) `StreamedAudioResult` is the return value of the `VoicePipeline.run()` method.**
- ✗d) They are unrelated classes.

16. What would happen if `StreamedAudioResult` did not have access to the `tts_model` and `tts_settings`?

- ✗a) It would still stream events, but without any audio data.
- ✗b) It would fallback to a default TTS model.
- ✓**c) It would not be able to perform text-to-speech conversion and thus couldn't stream audio.**
- ✗d) It would only stream `TextEvents`.

17. The design of `StreamedAudioResult` directly supports which common requirement for modern voice interfaces?

- ✗a) Offline processing of large audio datasets.
- ✗b) Batch processing of hundreds of requests simultaneously.
- ✓**c) Low-latency, interactive conversational experiences.**
- ✗d) Manual audio editing capabilities.

18. `StreamedAudioResult` is located in `src/agents/voice/result.py`, reinforcing its role as:

- ✗a) A utility for audio file manipulation.
- ✗b) A backend database connector.
- ✓**c) A component focused on the outcome/output of voice processing.**
- ✗d) An input formatter.

19. The use of an `AsyncIterator` for `stream()` allows the consuming application to:

- ✗a) Block until all audio is generated.
- ✗b) Request specific events by index.
- ✓**c) Process events as they are produced, without waiting for the entire stream to finish.**
- ✗d) Store the entire stream in memory before processing.

20. Which of these is not a direct responsibility of `StreamedAudioResult`?

- ✗a) Converting text to audio.
- ✗b) Streaming generated audio data.
- ✓**c) Deciding the agent's textual response based on business logic.**
- ✗d) Emitting events related to the voice stream.

OpenAI Agents SDK - VoicePipelineConfig MCQs

. What is the primary purpose of the `VoicePipelineConfig` dataclass?

- ✗a) To store the actual audio data for the pipeline.
- ✗b) To define the abstract interface for a voice workflow.
- ✓**c) To provide configurable options for customizing a VoicePipeline.**
- ✗d) To manage the streaming output of the pipeline.

---

2. By default, `VoicePipelineConfig` uses which model provider?

- ✗a) CustomVoiceModelProvider
- ✗b) GoogleCloudVoiceModelProvider
- ✓**c) OpenAIVoiceModelProvider**
- ✗d) AzureVoiceModelProvider

---

3. If `tracing_disabled` is set to True in `VoicePipelineConfig`, what is the effect?

- ✗a) Only audio data tracing is disabled.
- ✗b) Only sensitive data tracing is disabled.
- ✓**c) Tracing for the VoicePipeline's operations is completely disabled.**
- ✗d) Tracing for the internal VoiceWorkflow is also disabled, regardless of its own settings.

---

4. The `trace_include_sensitive_data` attribute in `VoicePipelineConfig` specifically refers to sensitive data from:

- ✗a) The user's input audio.
- ✗b) The VoiceWorkflow's internal operations.
- ✓**c) The VoicePipeline's own processing (e.g., transcriptions generated by the pipeline).**
- ✗d) All data within the entire application.

---

5. What is the default value for `trace_include_sensitive_audio_data`?

- ✓**a) False**
- ✗b) True
- ✗c) None
- ✗d) Depends on the model_provider.

6. Which attribute allows you to link multiple traces from the same conversation or process together?

- ✗a) workflow_name
- ✗b) trace_metadata
- ✗c) stt_settings
- ✓**d) group_id**

---

7. If `group_id` is not explicitly provided in `VoicePipelineConfig`, what happens?

- ✗a) It defaults to an empty string.
- ✓**b) A random group_id is generated using `gen_group_id()`.**
- ✗c) An error is raised.
- ✗d) It reuses the workflow_name as the group ID.

---

8. `VoicePipelineConfig` includes separate settings objects for:

- ✗a) Input and output channels.
- ✗b) Audio format and sample rate.
- ✓**c) Speech-to-Text (STT) and Text-to-Speech (TTS) models.**
- ✗d) Tracing and non-tracing operations.

---

9. The `workflow_name` attribute in `VoicePipelineConfig` defaults to:

- ✗a) Default Voice Workflow
- ✗b) Main Agent Workflow
- ✓**c) Voice Agent**
- ✗d) The name of the VoiceWorkflowBase instance.

---

10. What kind of data can be added to a trace using the `trace_metadata` attribute?

- ✗a) Only predefined integer values.
- ✗b) Only string identifiers.
- ✓**c) An optional dictionary of additional custom key-value pairs.**
- ✗d) Raw audio buffers.

---

11. The `stt_settings` attribute within `VoicePipelineConfig` allows customization of:

- ✗a) The voice of the TTS model.
- ✗b) The number of channels for audio output.
- ✓**c) Parameters specific to the Speech-to-Text model (e.g., language).**
- ✗d) The maximum queue size for audio input.

---

12. The `VoicePipelineConfig` is a dataclass which means:

- ✗a) It must contain abstract methods.
- ✓**b) It's primarily a container for data with auto-generated methods like `__init__`, `__repr__`, etc.**
- ✗c) It is designed for complex procedural logic.
- ✗d) It can only be inherited from, not instantiated directly.

---

13. Which of these attributes provides a global kill switch for tracing all pipeline operations?

- ✗a) trace_include_sensitive_data
- ✗b) trace_metadata
- ✓**c) tracing_disabled**
- ✗d) group_id

---

14. The `VoicePipelineConfig` is passed to the VoicePipeline's constructor to:

- ✗a) Trigger an immediate run of the pipeline.
- ✗b) Return the final audio result.
- ✓**c) Configure the pipeline's behavior and its internal components.**
- ✗d) Only set up the `tts_model`.

---

15. If `trace_include_sensitive_audio_data` is False, but `trace_include_sensitive_data` is True, what will be traced from the pipeline?

- ✗a) No sensitive data at all.
- ✓**b) Non-audio sensitive data (e.g., text) will be traced, but audio data will not.**
- ✗c) Only audio data will be traced.
- ✗d) Both audio and non-audio sensitive data will be traced.

---

16. What is the type of the default value provided for `model_provider`?

- ✗a) TTSModel
- ✗b) STTModel
- ✓**c) VoiceModelProvider**
- ✗d) VoicePipelineConfig

---

17. The `tts_settings` attribute focuses on configuring:

- ✗a) The input audio sample rate.
- ✗b) The transcription language.
- ✓**c) The characteristics of the synthesized speech (e.g., voice, pitch, rate).**
- ✗d) The buffer size for streaming input.

---

18. `VoicePipelineConfig` allows for control over the VoicePipeline's interaction with which observability feature?

- ✗a) Logging
- ✗b) Metrics
- ✓**c) Tracing**
- ✗d) Alerts

---

19. Setting `group_id` helps in:

- ✗a) Decreasing the latency of audio processing.
- ✗b) Increasing the accuracy of transcription.
- ✓**c) Providing contextual linkage across multiple conversation turns/traces in a monitoring system.**
- ✗d) Reducing the size of exported trace data.

---

20. When creating a `VoicePipeline`, if you don't provide a `VoicePipelineConfig` object, what happens?

- ✗a) An error is raised.
- ✗b) The pipeline runs with no configuration, leading to unpredictable behavior.
- ✓**c) A default `VoicePipelineConfig` instance with all its default values is used.**
- ✗d) The `VoicePipeline` prompts the user for configuration details.

OpenAI Agents SDK - VoiceStreamEvent MCQs

. What is VoiceStreamEvent defined as?

- ✗a) A concrete class for all voice stream events.
- ✓**b) A TypeAlias (Union) of different specific voice stream event types.**
- ✗c) An abstract base class for streaming.
- ✗d) A configuration object for audio.

---

2. From which method are VoiceStreamEvents primarily streamed?

- ✗a) VoicePipeline.run()
- ✓**b) StreamedAudioResult.stream()**
- ✗c) VoiceWorkflowBase.run()
- ✗d) AudioInput.add_audio()

---

3. Which of the following is NOT a defined type of VoiceStreamEvent in the provided snippet?

- ✓**a) VoiceStreamEventAudio**
- ✓**b) VoiceStreamEventLifecycle**
- ✓**c) VoiceStreamEventError**
- ✗**d) VoiceStreamEventText ← Correct answer: d**, not defined in the Union in the source.

---

4. What is the main content of a VoiceStreamEventAudio object?

- ✗a) A string representing transcribed text.
- ✗b) An integer indicating audio volume.
- ✓**c) A NumPy array (NDArray) containing raw audio data.**
- ✗d) A boolean indicating if audio is playing.

---

5. The `type` attribute of VoiceStreamEventAudio will always have which literal string value?

- ✓**a) "audio"**
- ✗b) "stream_audio"
- ✗c) "voice_stream_event_audio"
- ✗d) "audio_data"

---

6. Which VoiceStreamEvent type is used to signal the beginning or end of a conversational segment?

- ✗a) VoiceStreamEventAudio
- ✓**b) VoiceStreamEventLifecycle**
- ✗c) VoiceStreamEventError
- ✗d) VoiceStreamEventMetadata

---

7. The `event` attribute of VoiceStreamEventLifecycle can take which values?

- ✗a) "start", "stop", "pause"
- ✗b) "audio_started", "audio_stopped", "audio_error"
- ✓**c) "turn_started", "turn_ended", "session_ended"**
- ✗d) "processing", "responding", "idle"

---

8. When would a VoiceStreamEventError typically be yielded?

- ✗a) Whenever the VoicePipeline starts.
- ✓**b) When an unhandled exception occurs within the pipeline's streaming process.**
- ✗c) When the audio input is empty.
- ✗d) After the VoicePipeline successfully completes.

---

9. What information is contained within a VoiceStreamEventError object?

- ✗a) A simple error code.
- ✗b) A string message only.
- ✓**c) The actual Python Exception object that occurred.**
- ✗d) The timestamp of the error.

---

10. The `type` attribute across all VoiceStreamEvent dataclasses serves what purpose?

- ✗a) It determines the size of the event.
- ✗b) It specifies the encoding of the data.
- ✓**c) It provides a clear, programmatic way to identify the specific kind of event.**
- ✗d) It indicates the priority of the event.

---

11. Why is it beneficial to stream VoiceStreamEvents instead of providing a single, complete output?

- ✗a) It consumes less memory on the server.
- ✗b) It guarantees lower network latency.
- ✓**c) It enables real-time responsiveness and allows client applications to react incrementally.**
- ✗d) It simplifies the internal logic of the VoicePipeline.

---

12. A client receiving `VoiceStreamEventLifecycle` with `event="turn_started"` might:

- ✗a) Immediately stop listening for user input.
- ✗b) Start playing a pre-recorded message.
- ✓**c) Update its UI to indicate the agent is processing/responding.**
- ✗d) Close the connection.

---

13. The `data` attribute in VoiceStreamEventAudio can be None. When might this occur?

- ✗a) Only if an error happened during audio synthesis.
- ✓**b) Potentially to signal silence, a pause, or the end of an audio segment without actual samples.**
- ✗c) Only if the tts_model is disabled.
- ✗d) If the frame_rate is zero.

---

14. `VoiceStreamEventLifecycle` with `event="session_ended"` would typically be used to signal:

- ✗a) That a single turn of conversation has finished.
- ✓**b) That the entire voice interaction session has concluded.**
- ✗c) That an error occurred during the session.
- ✗d) That audio data is about to be sent.

---

15. What type of Python construct defines `VoiceStreamEvent`?

- ✗a) class
- ✗b) enum
- ✓**c) TypeAlias (a union of dataclasses)**
- ✗d) protocol

---

16. Which VoiceStreamEvent is crucial for playing back the agent's response to the user?

- ❌a) VoiceStreamEventLifecycle
- ✅**b) VoiceStreamEventAudio**
- ❌c) VoiceStreamEventError
- ❌d) All of them are equally crucial.

---

17. If a VoiceStreamEventError is received, what should a robust client application ideally do?

- ❌a) Ignore it and continue processing.
- ❌b) Immediately crash.
- ✅**c) Log the error, potentially display a user-friendly message, and decide whether to terminate or attempt recovery.**
- ❌d) Send the error back to the VoicePipeline.

---

18. The `Literal` type hints used for `type` and `event` attributes mean:

- ❌a) The values can be any string.
- ✅**b) The values are restricted to a specific set of predefined string constants.**
- ❌c) The values are numerical.
- ❌d) The values are optional.

---

19. The events defined in `src/agents/voice/events.py` directly support the responsiveness provided by:

- ❌a) AudioInput
- ❌b) VoiceWorkflowBase
- ✅**c) StreamedAudioResult**
- ❌d) VoicePipelineConfig

---

20. A VoiceStreamEvent does not directly contain:

- ❌a) Audio data.
- ❌b) Lifecycle signals.
- ❌c) Error objects.
- ✅**d) The transcribed user input.**

OpenAI Agents SDK - STTWebsocketConnectionError MCQs

1. From which base class does STTWebsocketConnectionError directly inherit?

- ❌a) Exception

- ✓**b) AgentsException**
- ✗c) WebsocketError
- ✗d) VoiceException

---

2. What specific type of connection failure does STTWebsocketConnectionError indicate?

- ✗a) An HTTP API connection failure for text processing.
- ✗b) A database connection failure.
- ✓**c) A WebSocket connection failure for Speech-to-Text services.**
- ✗d) A connection timeout during file download.

---

3. In the context of a VoicePipeline, when would this exception typically be raised?

- ✗a) When the Text-to-Speech model fails to synthesize audio.
- ✗b) When the VoiceWorkflowBase encounters an internal logic error.
- ✓**c) When the real-time connection to the Speech-to-Text service cannot be established or is lost.**
- ✗d) When the VoicePipelineConfig is invalid.

---

4. Which of the following is a common reason for an STTWebsocketConnectionError?

- ✗a) Incorrect grammar in the user's speech.
- ✗b) The STT model providing a low confidence score for transcription.
- ✓**c) A firewall blocking the WebSocket connection.**
- ✗d) Running out of disk space on the client machine.

---

5. This exception primarily focuses on an issue with the:

- ✗a) Quality of transcription.
- ✗b) Authentication of the user.
- ✓**c) Underlying communication channel.**
- ✗d) Performance of the agent's response.

---

6. If you catch an STTWebsocketConnectionError, what is the most immediate area you should investigate?

- ✗a) The Text-to-Speech model's configuration.
- ✗b) The logic within your VoiceWorkflowBase.
- ✓**c) Network connectivity and the STT service's status/URL.**
- ✗d) The amount of RAM available on your system.

---

7. Which type of STT interaction is most likely to rely on WebSockets and thus potentially raise this error?

- ✗a) Batch processing of large, pre-recorded audio files.
- ✓**b) Real-time, continuous speech transcription.**
- ✗c) Offline language model training.
- ✗d) Single, short audio clip analysis.

8. By inheriting from AgentsException, STTWebsocketConnectionError allows for:

- ✗a) Automatic retry mechanisms.
- ✓**b) More granular error handling specific to the SDK's functionalities.**
- ✗c) Direct access to the VoicePipeline's internal state.
- ✗d) Automatic logging to a central server.

---

9. Which of these scenarios would less likely directly cause an STTWebsocketConnectionError (though it might be a subsequent issue)?

- ✗a) The STT server going offline.
- ✗b) An invalid WebSocket endpoint URL.
- ✓**c) The user speaking very quietly, leading to poor audio input.**
- ✗d) A proxy server interfering with the WebSocket handshake.

---

10. The presence of "Websocket" in the exception name is significant because:

- ✗a) It means the error only occurs on web browsers.
- ✗b) It indicates the error is related to web security.
- ✓**c) It points to a persistent, bidirectional communication protocol used for streaming.**
- ✗d) It refers to the use of webhooks for notifications.

---

11. What would be an appropriate troubleshooting step if you encounter this exception?

- ✗a) Change the tts_model setting.
- ✗b) Adjust the workflow_name in VoicePipelineConfig.
- ✓**c) Ping the STT service's domain or check its status page.**
- ✗d) Modify the VoiceWorkflowHelper logic.

---

12. This exception is part of the `src/agents/voice/exceptions.py` module, indicating its specific relevance to:

- ✗a) General Python errors.
- ✗b) Database errors.
- ✓**c) Voice-related functionalities within the Agents SDK.**
- ✗d) File system operations.

---

13. Could an incorrect API key or authentication token lead to an STTWebsocketConnectionError?

- ✓**a) Yes, as the service might refuse the WebSocket handshake due to authentication failure.**
- ✗b) No, authentication issues would raise a different error.
- ✗c) Only if the API key is too long.
- ✗d) Only if the API key is too short.

---

14. If you write a try...except block to handle errors from VoicePipeline, which exception type would catch all SDK-specific errors including this one?

- ✗a) Exception
- ✗b) RuntimeError
- ✓**c) AgentsException**
- ✗d) STTException

---

15. STTWebsocketConnectionError suggests that the problem is preventing the agent from:

- ✗a) Responding to the user in audio.
- ✗b) Calling external tools.
- ✓**c) Receiving and processing spoken input from the user.**
- ✗d) Storing conversation history.

---

16. What kind of communication is typical over a WebSocket connection used for STT?

- ✗a) Only client sending audio to server.
- ✗b) Only server sending transcription to client.
- ✓**c) Bidirectional streaming of audio from client and text from server.**
- ✗d) One-time request-response cycles.

---

17. If a developer uses a Speech-to-Text service that relies on HTTP POST requests rather than WebSockets, would this specific exception likely be raised for connection failures?

- ✗a) Yes, because it's a general STT error.
- ✓**b) No, because the error is specific to WebSocket connections.**
- ✗c) Only if the HTTP POST request times out.
- ✗d) It depends on the VoicePipelineConfig.

---

18. The term "connection" in the exception name refers to the link between:

- ✗a) The VoicePipeline and the VoiceWorkflowBase.
- ✗b) The STTModel and the TTSModel.
- ✓**c) The SDK client (running your agent) and the remote STT service.**
- ✗d) The agent and the user's local audio device.

---

19. Which Python module would you look into if you wanted to implement specific error handling for this exception?

- ✗a) src/agents/voice/pipeline.py
- ✗b) src/agents/voice/workflow.py
- ✓**c) src/agents/voice/exceptions.py**
- ✗d) src/agents/voice/input.py

---

20. What is a key characteristic that distinguishes STTWebsocketConnectionError from other general Exception types?

- ❌a) It is always a fatal error.
- ❌b) It provides a detailed stack trace of the entire application.
- ✅**c) It provides specific context about a failure in the STT WebSocket communication, aiding targeted debugging.**
- ❌d) It automatically retries the connection.

OpenAI Agents SDK - Voice Model MCQs

What is the primary purpose of the TTSVoice type alias?

- ❌a) To define the temperature setting for TTS models.
- ❌b) To specify the audio buffer size for streaming.
- ✅**c) To list predefined, literal string values for available TTS voices.**
- ❌d) To indicate the language of the TTS output.

---

2. Which attribute in TTSModelSettings allows for post-processing of audio data before it's streamed?

- ❌a) voice
- ❌b) buffer_size
- ❌c) instructions
- ✅**d) transform_data**

---

3. The text_splitter attribute in TTSModelSettings is crucial for:

- ❌a) Detecting conversational turns in audio.
- ✅**b) Chunking text into smaller pieces for streaming Text-to-Speech.**
- ❌c) Splitting audio data into separate channels.
- ❌d) Transforming the data type of the audio.

---

4. The TTSModel.run() method returns an AsyncIterator[bytes]. What does bytes represent in this context?

- ❌a) The raw text input.
- ❌b) A base64 encoded string.
- ✅**c) Chunks of audio data in PCM format.**
- ❌d) An error message.

---

5. What is the main role of the TTSModel abstract base class?

- ❌a) To configure the input audio settings.
- ❌b) To provide a factory for STT models.

- ✓**c) To define the interface for any Text-to-Speech model.**
- ✗d) To manage the conversation history.

---

6. StreamedTranscriptionSession is designed for:

- ✗a) Transcribing static audio files.
- ✓**b) Streaming text transcriptions from continuous audio input.**
- ✗c) Converting text to speech in real-time.
- ✗d) Managing database connections for transcription data.

---

7. Which attribute in STTModelSettings can be used to guide the STT model's transcription, especially for domain-specific vocabulary?

- ✗a) language
- ✗b) temperature
- ✓**c) prompt**
- ✗d) turn_detection

---

8. The STTModel.transcribe() method is used for:

- ✗a) Real-time, continuous audio input.
- ✓**b) Static AudioInput (complete audio buffers).**
- ✗c) Generating audio from text.
- ✗d) Detecting silence in an audio stream.

---

9. What is the purpose of VoiceModelProvider?

- ✗a) To directly perform STT and TTS conversions.
- ✗b) To define the settings for STT and TTS models.
- ✓**c) To act as a factory for creating STT and TTS model instances by name.**
- ✗d) To store the configuration of the entire VoicePipeline.

---

10. If TTSModelSettings.speed is set to 0.5, how would the TTS model read the text?

- ✗a) Twice as fast as normal.
- ✗b) At a random speed.
- ✓**c) At half the normal speed.**
- ✗d) It would ignore the setting.

---

11. STTModel.create_session() returns an instance of:

- ✗a) STTModelSettings
- ✗b) AudioInput
- ✓**c) StreamedTranscriptionSession**

- ✗d) VoiceModelProvider

---

12. The instructions attribute in TTSModelSettings can help control the TTS model's:

- ✗a) Input audio language.
- ✗b) Transcription accuracy.
- ✓**c) Tone or how it handles partial sentences.**
- ✗d) Network connectivity.

---

13. Which of these tracing parameters are passed to both STTModel.transcribe() and STTModel.create_session()?

- ✗a) group_id
- ✗b) workflow_name
- ✓**c) trace_include_sensitive_data and trace_include_sensitive_audio_data**
- ✗d) buffer_size

---

14. What does the AsyncIterator[str] return type of StreamedTranscriptionSession.transcribe_turns() signify?

- ✗a) It returns a single, complete transcription string.
- ✓**b) It yields textual transcriptions incrementally, turn by turn.**
- ✗c) It returns a list of all detected turns at once.
- ✗d) It produces a stream of audio bytes.

---

15. TTSModelSettings.dtype = int16 indicates:

- ✗a) The voice model uses 16 different voices.
- ✗b) The speed of the audio is 1.6x.
- ✓**c) The audio data will be returned in 16-bit integer format.**
- ✗d) The buffer size is limited to 16 bytes.

---

16. What is the role of STTModelSettings.turn_detection?

- ✗a) To identify the speaker's accent.
- ✗b) To determine the language of the audio.
- ✓**c) To configure how conversational turns are recognized in streamed audio.**
- ✗d) To set the overall volume of the transcribed text.

---

17. If you have a custom STT model you want to integrate, which abstract class would you need to implement?

- ✗a) VoiceModelProvider
- ✗b) TTSModel
- ✓**c) STTModel**
- ✗d) StreamedTranscriptionSession

---

18. The VoiceModelProvider interface is designed for:

- ❌a) Directly converting audio to text.
- ❌b) Applying settings to models.
- ✅**c) Abstracting the access and instantiation of specific STT/TTS model implementations.**
- ❌d) Handling errors during model execution.

---

19. Why is StreamedTranscriptionSession.close() an async method?

- ❌a) It needs to perform immediate, blocking operations.
- ✅**b) To allow for asynchronous resource cleanup (e.g., closing network connections).**
- ❌c) It only operates on local files.
- ❌d) It is meant to be called at the very start of a session.

---

20. The model_name property is present in both TTSModel and STTModel to:

- ❌a) Specify the file path of the model.
- ❌b) Indicate the version of the model.
- ✅**c) Provide a unique identifier for the specific model being used.**
- ❌d) Store a description of the model's capabilities.

OpenAI Agents SDK - Text Utility MCQs

1. What is the primary function of get_sentence_based_splitter?

- ❌a) To concatenate multiple text strings into one.
- ❌b) To perform sentiment analysis on text.
- ✅**c) To return a function that splits text into chunks based on sentence boundaries.**
- ❌d) To convert text into audio.

---

2. The get_sentence_based_splitter function itself returns a:

- ❌a) str
- ❌b) list[str]
- ✅**c) Callable (a function)**
- ❌d) tuple[str, str]

---

3. Why is sentence-based splitting particularly useful for Text-to-Speech (TTS)?

- ❌a) It reduces the overall size of the text.
- ❌b) It simplifies grammatical analysis.
- ✅**c) It helps ensure natural intonation and pacing by sending complete thoughts/sentences to the TTS model.**

- ✗d) It speeds up the initial transcription process.

---

4. What is the default value for the min_sentence_length parameter?

- ✗a) 5
- ✗b) 10
- ✓**c) 20**
- ✗d) 50

---

5. If a sentence detected by the splitter is shorter than min_sentence_length, what might happen?

- ✗a) It will be discarded.
- ✗b) It will cause an error.
- ✓**c) It might be grouped with adjacent sentences to form a larger chunk.**
- ✗d) It will be sent as a separate, very small chunk regardless.

---

6. The function returned by get_sentence_based_splitter takes which type of input?

- ✗a) A list of sentences.
- ✗b) An audio buffer.
- ✓**c) A single str (the text to be split).**
- ✗d) A TTSModelSettings object.

---

7. What is the return type of the function returned by get_sentence_based_splitter?

- ✓**a) list[str]**
- ✗b) AsyncIterator[str]
- ✗c) str
- ✗d) tuple[str, str]

Note: This is actually context-dependent. Earlier versions returned `tuple[str, str]` *per call, while in newer simplified implementations, a full list may be returned. Let me know your SDK version to adjust.*

---

8. In the tuple[str, str] returned by the splitter function, what does the first str typically represent?

- ✗a) The remaining, unprocessed text.
- ✗b) A random segment of the input.
- ✓**c) A complete chunk of text (one or more full sentences) ready for processing.**
- ✗d) An error message.

---

9. What does the second str in the returned tuple[str, str] typically represent?

- ✗a) The first sentence found.
- ✓**b) The remaining part of the text that couldn't form a complete sentence-based chunk yet.**
- ✗c) A copy of the entire input text.

- ✗d) A concatenated version of all sentences.

---

10. The get_sentence_based_splitter is part of the src/agents/voice/utils.py module, indicating it's a:

- ✗a) Core pipeline component.
- ✗b) Model definition.
- ✓**c) Helper function for common tasks within voice agent development.**
- ✗d) Exception handler.

---

11. Could get_sentence_based_splitter be used outside of a VoicePipeline context?

- ✓**a) Yes, it's a general utility for text processing.**
- ✗b) No, it's tightly coupled to the pipeline's internal state.
- ✗c) Only if you manually import all voice models.
- ✗d) Only for audio file processing.

---

12. If min_sentence_length is set to 0, what would be a likely behavior of the returned splitter?

- ✗a) It would always return empty strings.
- ✗b) It would treat every character as a sentence.
- ✓**c) It would likely split on every sentence boundary, even for very short sentences.**
- ✗d) It would not split the text at all.

---

13. What problem does the text_splitter (where get_sentence_based_splitter is typically used) help solve for streaming TTS?

- ✗a) Ensuring the correct language is used.
- ✗b) Reducing the CPU load of the TTS model.
- ✓**c) Preventing unnatural pauses or intonation issues in streamed audio caused by arbitrary text chunking.**
- ✗d) Authenticating with the TTS service.

---

14. Is get_sentence_based_splitter an asynchronous function?

- ✗a) Yes, because it deals with streaming.
- ✓**b) No, the function it returns operates synchronously on strings.**
- ✗c) Only if min_sentence_length is very high.
- ✗d) Only when used within an AsyncIterator.

---

15. What type of splitting logic does this utility specifically employ?

- ✗a) Character-based splitting.
- ✗b) Word-based splitting.
- ✓**c) Sentence-based splitting.**

- ✗d) Paragraph-based splitting.

---

16. If the input text to the splitter function is "Hello.", and min_sentence_length=20, what might the output tuple be?

- ✗a) ("Hello.", "")
- ✓**b) ("", "Hello.")**
- ✗c) ("Hello", ".")
- ✗d) ("Hello", "!")

---

17. The Callable type hint signifies that the returned object is:

- ✗a) A class instance.
- ✗b) A variable.
- ✓**c) Something that can be called like a function.**
- ✗d) A data structure.

---

18. This utility is typically used as the value for which attribute of TTSModelSettings?

- ✗a) voice
- ✗b) instructions
- ✓**c) text_splitter**
- ✗d) buffer_size

---

19. get_sentence_based_splitter is an example of a:

- ✗a) Class method.
- ✗b) Static method.
- ✓**c) Factory function.**
- ✗d) Constructor.

---

20. What is the primary goal of the get_sentence_based_splitter in improving the user experience of a voice agent?

- ✗a) To make the agent respond faster.
- ✗b) To make the agent's voice sound more human-like.
- ✓**c) To ensure that the agent's spoken responses flow naturally and are easily understandable.**
- ✗d) To minimize the data transferred over the network.

OpenAI Agents SDK - OpenAIVoiceModelProvider MCQs

1. What is the primary function of OpenAIVoiceModelProvider?

- ✗a) To manage audio input/output devices.

- ❌b) To define a new type of OpenAI model.
- ✅**c) To provide access to OpenAI's Speech-to-Text and Text-to-Speech models.**
- ❌d) To handle network communication for voice agents.

---

2. OpenAIVoiceModelProvider inherits from which abstract base class?

- ❌a) TTSModel
- ❌b) STTModel
- ✅**c) VoiceModelProvider**
- ❌d) AsyncOpenAI

---

3. Which parameter in the constructor allows you to specify a custom OpenAI API endpoint?

- ❌a) api_key
- ✅**b) base_url**
- ❌c) openai_client
- ❌d) organization

---

4. If the api_key parameter is None during OpenAIVoiceModelProvider initialization, where will it attempt to get the API key from?

- ❌a) It will raise an immediate error.
- ✅**b) It will typically look for the `OPENAI_API_KEY` environment variable or other default locations.**
- ❌c) It will use a publicly available key.
- ❌d) It will prompt the user for the key.

---

5. The openai_client parameter in the constructor is provided for what purpose?

- ❌a) To automatically create a new API key.
- ✅**b) To allow injection of an already-configured AsyncOpenAI client instance.**
- ❌c) To specify which OpenAI model to use globally.
- ❌d) To disable all tracing.

---

6. If openai_client is provided to the constructor, what happens to api_key and base_url passed to the same constructor?

- ✅**a) They are ignored, as the provided openai_client takes precedence for configuration.**
- ❌b) They are used to reconfigure the provided openai_client.
- ❌c) They cause an error if both are present.
- ❌d) They are stored for future use but not applied immediately.

---

7. Which method of OpenAIVoiceModelProvider is responsible for providing an STT model?

- ❌a) get_voice_model()

- ✗b) create_transcriber()
- ✓**c) get_stt_model()**
- ✗d) get_text_model()

---

8. The get_tts_model() method returns an instance that conforms to which abstract class?

- ✗a) STTModel
- ✓**b) TTSModel**
- ✗c) VoiceModelProvider
- ✗d) StreamedTranscriptionSession

---

9. What is the typical OpenAI STT model name you would pass to get_stt_model()?

- ✗a) "tts-1"
- ✗b) "gpt-4"
- ✓**c) "whisper-1"**
- ✗d) "davinci"

---

10. The organization and project parameters are primarily used for:

- ✗a) Defining the agent's behavior.
- ✗b) Configuring audio input/output devices.
- ✓**c) Billing and resource management within OpenAI.**
- ✗d) Specifying the language of the voice model.

---

11. What kind of models does OpenAIVoiceModelProvider explicitly provide?

- ✗a) Only Large Language Models (LLMs).
- ✗b) Only Image Generation Models.
- ✓**c) Speech-to-Text and Text-to-Speech models.**
- ✗d) Only Custom Trained Models.

---

12. If model_name is None when calling get_tts_model(), what will the provider likely do?

- ✗a) Raise an error, as a model name is always required.
- ✗b) Return a random TTS model.
- ✓**c) Return a default TTS model (e.g., "tts-1").**
- ✗d) Revert to a local, non-OpenAI model.

---

13. The OpenAIVoiceModelProvider bridges the gap between the generic VoicePipeline framework and:

- ✗a) Local file storage.
- ✗b) User interface components.
- ✓**c) OpenAI's specific API services for voice.**
- ✗d) Other cloud providers.

14. Is OpenAIVoiceModelProvider a concrete class or an abstract class?

- ❌a) Abstract, as it has abstract methods.
- ✅**b) Concrete, as it provides implementations for its base class's abstract methods.**
- ❌c) It's neither, it's a dataclass.
- ❌d) It depends on the Python version.

---

15. What is a key advantage of having a VoiceModelProvider abstraction like OpenAIVoiceModelProvider?

- ❌a) It eliminates the need for API keys.
- ❌b) It forces all models to be from OpenAI.
- ✅**c) It allows the VoicePipeline to be model-agnostic, easily switching between different STT/TTS providers.**
- ❌d) It directly handles audio streaming to the user.

---

16. What type of client is typically instantiated internally by OpenAIVoiceModelProvider if openai_client is not provided?

- ❌a) OpenAIClient (synchronous)
- ✅**b) AsyncOpenAI**
- ❌c) requests.Session
- ❌d) httpx.Client

---

17. The methods get_stt_model and get_tts_model both take model_name: str | None. What does this indicate?

- ❌a) Only specific model names are allowed, no defaults.
- ❌b) The model name is always required.
- ✅**c) A specific model name can be requested, or a default will be used if None is provided.**
- ❌d) The model name is ignored.

---

18. In the context of a VoicePipelineConfig, an instance of OpenAIVoiceModelProvider would be assigned to which attribute?

- ❌a) stt_settings
- ❌b) tts_settings
- ❌c) workflow_name
- ✅**d) model_provider**

---

19. OpenAIVoiceModelProvider is concerned with:

- ❌a) The actual content of the user's speech.
- ❌b) The conversational flow and agent's logic.
- ✅**c) Providing the underlying AI services for voice processing.**
- ❌d) Storing historical data of voice interactions.

20. What would be a reason to provide a custom AsyncOpenAI client via the constructor, rather than letting the provider create one?

- ❌a) To bypass API key validation.
- ❌b) To use a synchronous OpenAI client.
- ✅**c) To apply shared configurations, custom headers, or manage connection pools externally for the OpenAI client.**
- ❌d) To force the use of local models.

OpenAI Agents SDK - OpenAI STT MCQs

1. From which base class does OpenAISTTTranscriptionSession inherit?

- ❌a) STTModel
- ✅**b) StreamedTranscriptionSession**
- ❌c) AudioInput
- ❌d) AsyncOpenAI

---

2. What is the primary function of OpenAISTTTranscriptionSession?

- ❌a) To synthesize speech from text
- ✅**b) To manage a continuous, real-time speech transcription session with OpenAI's STT service**
- ❌c) To transcribe a single, complete audio file
- ❌d) To configure OpenAI API keys

---

3. OpenAISTTModel inherits from which abstract base class?

- ❌a) TTSModel
- ✅**b) STTModel**
- ❌c) VoiceModelProvider
- ❌d) OpenAISTTTranscriptionSession

---

4. Which parameter is required in the OpenAISTTModel constructor to specify the particular OpenAI STT model to use?

- ❌a) api_key
- ✅**b) model**
- ❌c) base_url
- ❌d) settings

---

5. What type of client does OpenAISTTModel require in its constructor to make API calls to OpenAI?

- ❌a) SyncOpenAI
- ❌b) requests.Session

- ✅**c) AsyncOpenAI**
- ❌d) HTTPClient

---

6. The transcribe method of OpenAISTTModel is designed for:

- ❌a) Streaming audio input in real-time
- ❌b) Yielding transcription turns continuously
- ✅**c) Transcribing static, complete AudioInput objects**
- ❌d) Converting text to speech

---

7. What is the return type of the OpenAISTTModel.transcribe() method?

- ❌a) AsyncIterator[str]
- ❌b) bytes
- ✅**c) str**
- ❌d) AudioInput

---

8. Which method of OpenAISTTModel is used to initiate a real-time, streamed transcription process?

- ❌a) transcribe()
- ❌b) run_session()
- ✅**c) create_session()**
- ❌d) start_transcription()

---

9. When OpenAISTTModel.create_session() is called, what does it return?

- ❌a) The full transcription string
- ❌b) An AudioInput object
- ✅**c) A StreamedTranscriptionSession (specifically an OpenAISTTTranscriptionSession)**
- ❌d) An AsyncOpenAI client

---

10. The input parameter for OpenAISTTModel.transcribe() is of type AudioInput. What does this imply about the audio data?

- ❌a) It's expected to be pushed incrementally
- ❌b) It must come from a live microphone
- ✅**c) It represents a complete, static audio segment**
- ❌d) It's always in a compressed format

---

11. Which of the following is a common OpenAI STT model name used with OpenAISTTModel?

- ❌a) "tts-1"
- ❌b) "gpt-3.5-turbo"
- ✅**c) "whisper-1"**

- ❌d) "dall-e-3"

---

12. The `trace_include_sensitive_data` and `trace_include_sensitive_audio_data` parameters in both `transcribe` and `create_session` methods are related to:

- ❌a) Data compression
- ❌b) Audio processing quality
- ✅**c) Observability and tracing configuration**
- ❌d) API rate limits

---

13. The OpenAISTTTranscriptionSession class internally manages the communication to OpenAI's STT service, which often involves:

- ❌a) One-time HTTP POST requests
- ❌b) UDP broadcast messages
- ✅**c) WebSocket or streaming HTTP connections**
- ❌d) FTP transfers

---

14. What is the role of STTModelSettings when passed to OpenAISTTModel methods?

- ❌a) To define the model_name
- ❌b) To provide the api_key
- ✅**c) To configure specific transcription parameters like prompt or language**
- ❌d) To manage the AsyncOpenAI client instance

---

15. If you wanted to continuously send audio from a live microphone to OpenAI for transcription, you would first call `OpenAISTTModel.create_session()` and then interact with the returned:

- ❌a) OpenAISTTModel object
- ❌b) AudioInput object
- ✅**c) OpenAISTTTranscriptionSession object**
- ❌d) STTModelSettings object

---

16. What does the `OpenAISTTTranscriptionSession.transcribe_turns()` method yield?

- ❌a) Audio bytes
- ❌b) Error messages
- ✅**c) Text transcriptions, typically one "turn" at a time**
- ❌d) Configuration settings

---

17. If an OpenAISTTModel is initialized with model="whisper-1", but you pass a different model name to its transcribe() or create_session() methods, what would likely happen?

- ❌a) The method would ignore the new model name

- ✓**b) The constructor model is used by default, but a new model name passed to the method may override it if allowed**
- ✗c) An error would be raised
- ✗d) It would dynamically switch to the new model for that call without warning

---

18. The OpenAISTTModel is considered a "concrete" class because:

- ✗a) It does not inherit from any other class
- ✗b) It is defined in a Python file
- ✓**c) It provides actual implementations for the abstract methods of its base class STTModel**
- ✗d) It only deals with static data

---

19. OpenAISTTTranscriptionSession.close() is essential for:

- ✗a) Starting a new transcription
- ✗b) Retrieving the final transcript
- ✓**c) Properly cleaning up resources associated with the streamed transcription connection**
- ✗d) Changing the STT model settings

---

20. Which class is responsible for encapsulating the audio data itself, that is passed to OpenAISTTModel.transcribe()?

- ✗a) STTModelSettings
- ✗b) StreamedAudioInput
- ✓**c) AudioInput**
- ✗d) TTSModel

OpenAI Agents SDK - OpenAI TTS MCQs

1. From which base class does OpenAITTSModel inherit?

- ✗a) STTModel
- ✓**b) TTSModel**
- ✗c) VoiceModelProvider
- ✗d) AsyncOpenAI

---

2. What is the primary function of OpenAITTSModel?

- ✗a) To transcribe spoken audio into text
- ✗b) To manage real-time STT transcription sessions
- ✓**c) To convert text into speech using OpenAI models**
- ✗d) To provide configuration settings for voice models

3. Which parameter in the OpenAITTSModel constructor specifies the particular OpenAI TTS model to use?

- ✗a) api_key
- ✓**b) model**
- ✗c) base_url
- ✗d) settings

4. What type of client does OpenAITTSModel require in its constructor to interact with the OpenAI API?

- ✗a) SyncOpenAI
- ✗b) requests.Session
- ✓**c) AsyncOpenAI**
- ✗d) HTTPClient

5. The run method of OpenAITTSModel takes `text` and `settings` as parameters. What is the type of settings?

- ✗a) STTModelSettings
- ✓**b) TTSModelSettings**
- ✗c) VoicePipelineConfig
- ✗d) dict

6. What is the return type of the OpenAITTSModel.run() method?

- ✗a) str
- ✗b) bytes
- ✓**c) AsyncIterator[bytes]**
- ✗d) float

7. The AsyncIterator[bytes] return type of run() is crucial for:

- ✗a) Reducing file size
- ✗b) Ensuring perfect fidelity
- ✓**c) Enabling real-time streaming and low-latency playback**
- ✗d) Offline batch processing

8. Which attribute from TTSModelSettings would OpenAITTSModel use to determine the voice's gender and tone?

- ✗a) speed
- ✗b) buffer_size
- ✓**c) voice**
- ✗d) instructions

9. If TTSModelSettings.speed is set, how does OpenAITTSModel leverage this?

- ✗a) Changes pitch
- ✓**b) Adjusts speaking rate**
- ✗c) Increases volume
- ✗d) Adds pauses

---

10. The instructions parameter in TTSModelSettings passed to OpenAITTSModel.run() is used for:

- ✗a) Grammar correction
- ✗b) Language specification
- ✓**c) Guiding the model's tone or partial sentence handling**
- ✗d) API retry logic

---

11. What kind of audio format are the byte chunks typically in when yielded by OpenAITTSModel.run()?

- ✗a) MP3
- ✗b) WAV
- ✓**c) PCM**
- ✗d) AAC

---

12. OpenAITTSModel is considered a "concrete" class because:

- ✗a) Uses a specific model name
- ✗b) Is defined in a .py file
- ✓**c) Implements the abstract methods of TTSModel**
- ✗d) Relies on AsyncOpenAI client

---

13. Which class typically provides the AsyncOpenAI client instance to OpenAITTSModel's constructor?

- ✗a) TTSModelSettings
- ✗b) VoicePipelineConfig
- ✓**c) OpenAIVoiceModelProvider**
- ✗d) StreamedAudioResult

---

14. The text_splitter setting passed via TTSModelSettings to OpenAITTSModel.run() helps in:

- ✗a) Volume control
- ✗b) Detecting silence
- ✓**c) Breaking input into chunks for natural streaming**
- ✗d) Translation

---

15. What happens if the model parameter passed to OpenAITTSModel's constructor is invalid?

- ✗a) Defaults to "tts-1"
- ✗b) Uses STT instead
- ✓**c) Likely results in API call error**
- ✗d) Picks closest model

---

16. OpenAITTSModel directly interacts with:

- ✗a) Microphone
- ✗b) VoiceWorkflowBase
- ✓**c) OpenAI's TTS API endpoint**
- ✗d) Local file storage

---

17. The transform_data attribute in TTSModelSettings allows for:

- ✗a) Change model at runtime
- ✗b) Convert PCM to MP3
- ✓**c) Apply function to raw audio before streaming**
- ✗d) Split into tracks

---

18. Primary benefit of AsyncIterator in run() for conversational AI:

- ✗a) Offline use
- ✗b) Lower memory on OpenAI side
- ✓**c) Early playback while generating response**
- ✗d) Detailed logging

---

19. Which TTSVoice would you typically pass to TTSModelSettings for OpenAITTSModel?

- ✗a) "male" / "female"
- ✗b) "standard" / "neural"
- ✓**c) "alloy", "echo", "nova", etc.**
- ✗d) "english", "spanish"

---

20. OpenAITTSModel represents output generation in the voice pipeline, converting:

- ✗a) Speech to text
- ✓**b) Text to speech**
- ✗c) Text to text
- ✗d) Speech to speech

Extensions

OpenAI Agents SDK - Handoff Filters MCQs

1. What is the primary function of `remove_all_tools`?

- ✗a) To add new tools to a HandoffInputData object.
- ✗b) To execute all tools defined in the pipeline.
- ✓**c) To filter out all tool-related items from HandoffInputData.**
- ✗d) To log all tool usage to a file.

---

2. `remove_all_tools` operates on which specific data type?

- ✗a) str
- ✗b) dict
- ✓**c) HandoffInputData**
- ✗d) ToolCall

---

3. Which of the following is explicitly filtered out by `remove_all_tools`?

- ✗a) User messages
- ✗b) Agent responses
- ✓**c) Web search outputs**
- ✗d) Conversation summaries

---

4. The function signature `-> HandoffInputData` indicates that `remove_all_tools` returns:

- ✗a) A boolean value indicating success or failure.
- ✗b) The original, unmodified HandoffInputData.
- ✓**c) A modified HandoffInputData object with tool items removed.**
- ✗d) A different data type altogether.

---

5. Why might you want to remove tool information during a handoff to a human agent?

- ✗a) To increase the speed of the handoff.
- ✗b) To reduce the memory footprint of the agent.
- ✓**c) To present a cleaner, more relevant, and less technical view of the conversation.**
- ✗d) To allow the human agent to re-run the tools.

---

6. Which of these tool-related items does `remove_all_tools` target?

- ✗a) Only file search results.
- ✗b) Only function call inputs.
- ✗c) Only web search queries.
- ✓**d) File search, web search, and function calls (including output).**

---

7. A key reason for using handoff filters like `remove_all_tools` is to enhance:

- ✗a) Agent processing speed.
- ✗b) Model accuracy.
- ✓**c) Security and privacy during data transfer.**
- ✗d) The number of tools available to the agent.

---

8. If a HandoffInputData contains a record of an agent making an `add_to_cart` function call and its successful output, what would `remove_all_tools` do?

- ✗a) Leave the `add_to_cart` record untouched.
- ✓**b) Remove both the `add_to_cart` function call and its output.**
- ✗c) Only remove the `add_to_cart` function call, keeping the output.
- ✗d) Only remove the `add_to_cart` output, keeping the function call.

---

9. What is the main benefit of simplification that `remove_all_tools` offers for handoff data?

- ✗a) It makes the data smaller in size.
- ✗b) It encrypts sensitive data.
- ✓**c) It makes the data more digestible and focused for the receiving entity.**
- ✗d) It adds metadata to the handoff.

---

10. In what scenario would `remove_all_tools` be most beneficial?

- ✗a) When debugging an agent's tool usage internally.
- ✓**b) When transferring a conversation context from an AI agent to a human customer support agent.**
- ✗c) When training a new tool usage model.
- ✗d) When conducting performance benchmarks of tool execution.

---

11. This function is categorized under "Handoff filters" because it:

- ✗a) Filters out handoff requests.
- ✗b) Filters which agent receives the handoff.
- ✓**c) Modifies the data before it is handed off.**
- ✗d) Filters data after it has been received.

---

12. Could `remove_all_tools` prevent a human agent from seeing a user's sensitive query that was part of a web search tool call?

- ✓**a) Yes, if the web search tool call and its input are part of the filtered items.**
- ✗b) No, it only filters output, not input.
- ✗c) Only if the query was explicitly marked as sensitive.
- ✗d) It's unrelated to user queries.

---

13. If `HandoffInputData` represents the complete conversation history including tool calls, `remove_all_tools` aims to present a view of the conversation that is primarily focused on:

- ✗a) Technical diagnostics.
- ✗b) Detailed tool execution steps.
- ✓**c) The natural language dialogue and core intent.**
- ✗d) External API response structures.

---

14. `remove_all_tools` helps prevent misinterpretation by the receiving entity because:

- ✗a) It translates tool outputs into natural language.
- ✓**b) Raw tool outputs can be confusing if the recipient doesn't have the context or capability to interpret them.**
- ✗c) It ensures all tool outputs are fully explained.
- ✗d) It adds warnings to uninterpretable data.

---

15. Does `remove_all_tools` modify the agent's internal state or only the data being prepared for handoff?

- ✗a) It modifies the agent's internal state.
- ✓**b) It primarily modifies the data being prepared for handoff.**
- ✗c) It modifies both.
- ✗d) It makes a read-only copy.

---

16. If an agent used a tool named `check_stock`, what would `remove_all_tools` filter out?

- ✗a) Only the fact that `check_stock` was attempted.
- ✗b) Only the numerical output of `check_stock`.
- ✓**c) Both the `check_stock` function call and its output.**
- ✗d) Nothing, as `check_stock` is not explicitly mentioned in the documentation snippet.

---

17. The `src/agents/extensions/handoff_filters.py` path suggests that this function is part of:

- ✗a) The core agent runtime.
- ✗b) The main VoicePipeline.
- ✓**c) Optional extensions or utilities for agent handoff scenarios.**
- ✗d) Low-level model definitions.

---

18. `remove_all_tools` contributes to providing a more concise representation of data during handoff by:

- ❌a) Summarizing tool outputs into a single sentence.
- ❌b) Compressing the HandoffInputData object.
- ✅**c) Removing detailed, potentially irrelevant tool execution logs.**
- ❌d) Encrypting the entire handoff data.

---

19. Why would `HandoffInputData` likely include tool items in the first place before filtering?

- ❌a) They are necessary for basic conversation flow.
- ✅**b) The HandoffInputData usually captures the full, rich internal state and history of the agent's processing.**
- ❌c) Tools are always public information.
- ❌d) It's a design error.

---

20. Which principle does `remove_all_tools` align with in data transfer scenarios?

- ❌a) Principle of Maximum Information.
- ✅**b) Principle of Least Privilege/Need-to-know.**
- ❌c) Principle of Data Redundancy.
- ❌d) Principle of Universal Data Access.

OpenAI Agents SDK - Handoff Prompt MCQs

1. What is the main purpose of RECOMMENDED_PROMPT_PREFIX?

- ❌a) To define the agent's personality.
- ✅**b) To provide system-level instructions about the multi-agent system and handoffs.**
- ❌c) To store user conversation history.
- ❌d) To list available tools for the agent.

---

2. RECOMMENDED_PROMPT_PREFIX is a:

- ❌a) Function.
- ❌b) Class.
- ✅**c) String constant (module attribute).**
- ❌d) Data structure.

---

3. The instruction *"Transfers between agents are handled seamlessly in the background; do not mention or draw attention to these transfers in your conversation with the user"* primarily serves to improve:

- ❌a) Agent processing speed.
- ❌b) Model accuracy.

- ✗c) Developer debugging experience.
- ✓**d) User experience and conversational flow.**

---

4. What type of function call does RECOMMENDED_PROMPT_PREFIX instruct agents to use for handoffs?

- ✗a) call_external_api()
- ✗b) initiate_transfer()
- ✗c) handoff_conversation()
- ✓**d) transfer_to_<agent_name>**

---

5. What does the prompt_with_handoff_instructions function do?

- ✗a) It executes a handoff.
- ✓**b) It prepends RECOMMENDED_PROMPT_PREFIX to a given agent prompt.**
- ✗c) It removes handoff instructions from a prompt.
- ✗d) It validates the correctness of a handoff prompt.

---

6. If `original_prompt = "You are a sales agent."`, what would `prompt_with_handoff_instructions(original_prompt)` return?

- ✗a) original_prompt unchanged.
- ✗b) Only RECOMMENDED_PROMPT_PREFIX.
- ✓**c) RECOMMENDED_PROMPT_PREFIX followed by original_prompt.**
- ✗d) original_prompt followed by RECOMMENDED_PROMPT_PREFIX.

---

7. The phrase "# System context" at the beginning of RECOMMENDED_PROMPT_PREFIX is a hint for the LLM that these instructions are:

- ✗a) From the user.
- ✗b) Optional suggestions.
- ✓**c) Core directives about its operational environment.**
- ✗d) Debugging information.

---

8. Why is it important for the agent to know that it's part of a "multi-agent system"?

- ✗a) To allow it to communicate directly with other agents.
- ✗b) To enable it to train other agents.
- ✓**c) To provide context for its ability to hand off tasks to other specialized agents.**
- ✗d) To help it choose the right language for the user.

---

9. If an agent is not given the RECOMMENDED_PROMPT_PREFIX, and is expected to perform handoffs, what is a likely outcome?

- ✗a) It will automatically figure out how to hand off.

- ✗b) It will refuse to communicate with the user.
- ✓**c) It might not know how to call the transfer_to_<agent_name> function or might explicitly mention transfers to the user.**
- ✗d) It will only respond in short sentences.

---

10. The `prompt_with_handoff_instructions` function returns a `str`. This means the output is a:

- ✗a) List of instructions.
- ✗b) Boolean value.
- ✓**c) Single, combined string.**
- ✗d) Dictionary of prompt components.

---

11. This handoff prompt mechanism helps in guiding the agent's:

- ✗a) Memory management.
- ✗b) Audio processing.
- ✓**c) Tool selection and conversational etiquette during transfers.**
- ✗d) External API integration.

---

12. The RECOMMENDED_PROMPT_PREFIX specifies that handoffs are "seamlessly in the background" from whose perspective?

- ✗a) The developer's.
- ✗b) The agent's.
- ✓**c) The user's.**
- ✗d) The system administrator's.

---

13. What is the type of `prompt` parameter in `prompt_with_handoff_instructions`?

- ✗a) list[str]
- ✓**b) str**
- ✗c) HandoffInputData
- ✗d) AgentConfig

---

14. The RECOMMENDED_PROMPT_PREFIX defines two primary abstractions of the Agents SDK. What are they?

- ✗a) Users and Tools.
- ✗b) Input and Output.
- ✓**c) Agents and Handoffs.**
- ✗d) Workflows and Models.

---

15. If `prompt_with_handoff_instructions` were not used, and you manually tried to add the prefix, what might be a potential issue?

- ✗a) The prompt would become too short.
- ✗b) The RECOMMENDED_PROMPT_PREFIX might change unexpectedly.
- ✓**c) You might accidentally modify the original RECOMMENDED_PROMPT_PREFIX or miss future updates.**
- ✗d) The agent would ignore the instructions.

---

16. The statement "An agent encompasses instructions and tools" from the prefix means:

- ✗a) Agents can only understand instructions.
- ✗b) Agents are purely rule-based.
- ✓**c) An agent has a set of instructions guiding its behavior and access to external functionalities.**
- ✗d) Agents are limited to internal operations only.

---

17. The `src/agents/extensions/handoff_prompt.py` path indicates that these are:

- ✗a) Mandatory core components for any agent.
- ✗b) Tools for external system integration.
- ✓**c) Optional extensions to help manage agent handoffs.**
- ✗d) Low-level hardware drivers.

---

18. What is the benefit of making `transfer_to_<agent_name>` a "general" naming convention?

- ✗a) It prevents name collisions.
- ✗b) It allows for dynamic agent creation.
- ✓**c) It provides a standardized and predictable way for the LLM to learn and invoke handoff tools.**
- ✗d) It makes the system more secure.

---

19. RECOMMENDED_PROMPT_PREFIX could be considered a form of:

- ✗a) Output formatting.
- ✗b) User input validation.
- ✓**c) System message or meta-prompting.**
- ✗d) Error handling.

---

20. When an agent calls a `transfer_to_<agent_name>` function, who handles the actual "transfer" in the background according to the prompt?

- ✗a) The user.
- ✗b) The receiving agent.
- ✓**c) The Agents SDK itself (implied by "Transfers between agents are handled seamlessly").**
- ✗d) An external API.

OpenAI Agents SDK - LitellmModel MCQs

1. What is the primary purpose of the LitellmModel class?

- ❌a) To integrate only OpenAI models into the Agents SDK.
- ❌b) To manage local LLM deployments.
- ✅**c) To enable the Agents SDK to use any LLM supported by the LiteLLM library.**
- ❌d) To provide a new type of conversational agent.

---

2. LitellmModel inherits from which base class?

- ❌a) Agent
- ❌b) VoiceModelProvider
- ✅**c) Model**
- ❌d) AsyncOpenAI

---

3. What problem does LiteLLM aim to solve for developers working with multiple LLM providers?

- ❌a) Reducing the size of LLM models.
- ❌b) Improving the training speed of LLMs.
- ✅**c) Providing a unified interface to various LLM APIs, simplifying integration.**
- ❌d) Automating the creation of new LLM models.

---

4. Which of the following LLM providers can LiteLLM (and thus LitellmModel) potentially access?

- ❌a) Only OpenAI and Anthropic.
- ❌b) Only Google Gemini.
- ❌c) Only Mistral.
- ✅**d) OpenAI, Anthropic, Gemini, Mistral, and many others.**

---

5. What is a significant benefit of using LitellmModel in terms of model selection?

- ❌a) It forces the use of a single, fixed LLM.
- ✅**b) It allows easy switching between different LLM providers without changing core agent code.**
- ❌c) It automatically selects the highest-cost model.
- ❌d) It only supports open-source models.

---

6. If you want to optimize costs by using the cheapest available LLM, how might LitellmModel (via LiteLLM) help?

- ❌a) By compressing prompts.
- ❌b) By pre-caching all possible responses.

- ✓**c) By supporting routing requests to the most cost-effective provider.**
- ✗d) By reducing the number of API calls made.

---

7. LitellmModel acts as a(n) ______ within the Agents SDK to connect to external LLMs.

- ✗a) Executor
- ✗b) Logger
- ✓**c) Adapter/Bridge**
- ✗d) Validator

---

8. What does "Model Agnosticism" mean in the context of LitellmModel?

- ✗a) The model is unaware of the prompt.
- ✗b) The model doesn't require an API key.
- ✓**c) The agent code does not need to be tightly coupled to a specific LLM provider's API.**
- ✗d) The model cannot be fine-tuned.

---

9. If an agent in the SDK is configured to use a LitellmModel, and the primary LLM provider fails, what capability might LiteLLM offer?

- ✗a) It will immediately raise an error.
- ✗b) It will halt the agent's execution.
- ✓**c) It can be configured for automatic fallbacks to alternative models/providers.**
- ✗d) It will attempt to fix the failing provider.

---

10. By integrating LiteLLM, LitellmModel contributes to:

- ✗a) Increased complexity in LLM integration.
- ✗b) Limited access to new LLMs.
- ✓**c) Simplified and unified API interaction for various LLMs.**
- ✗d) Slower response times from LLMs.

---

11. Where is the source code for LitellmModel located?

- ✓**a) src/agents/core/models/litellm_model.py**
- ✗b) src/agents/main/litellm_model.py
- ✗c) src/agents/extensions/models/litellm_model.py
- ✗d) src/litellm/model.py

---

12. The LitellmModel class allows developers to achieve which of the following without modifying core agent logic?

- ✗a) Changing the agent's personality.
- ✓**b) Swapping the underlying LLM provider (e.g., from OpenAI to Anthropic).**
- ✗c) Developing new tools.

- ✗d) Deploying the agent to a new server.

---

13. Which of these is not a direct benefit of using LitellmModel (and LiteLLM)?

- ✓**a) Simplified integration with multiple LLM APIs.**
- ✓**b) Potential for cost optimization.**
- ✗c) Training new, custom LLM models from scratch.
- ✓**d) Improved reliability through fallbacks.**

✓Trick question—**option C is NOT a benefit**, while the others **are**.

---

14. When an Agent uses a LitellmModel, the agent communicates with:

- ✗a) Directly with the OpenAI API.
- ✗b) Directly with the Anthropic API.
- ✓**c) The LitellmModel instance, which then uses LiteLLM to route the request.**
- ✗d) A local .json file containing LLM responses.

---

15. The LitellmModel likely implements abstract methods from its Model base class such as:

- ✗a) get_api_key()
- ✗b) configure_database()
- ✓**c) run_llm() or generate_response() (inferred)**
- ✗d) install_dependencies()

---

16. If a new LLM provider emerges, what is the most likely way it would be supported by the Agents SDK via LitellmModel?

- ✗a) The LitellmModel class would need to be completely rewritten.
- ✗b) A new specific Agent subclass for that provider would be created.
- ✓**c) LiteLLM would add support for it, and LitellmModel would automatically gain access.**
- ✗d) The Agents SDK would need a new VoiceModelProvider.

---

17. LitellmModel is categorized under `src/agents/extensions` because it:

- ✗a) Is a core, mandatory part of every agent.
- ✗b) Is an experimental feature that is not yet stable.
- ✓**c) Provides optional, extended functionality for integrating diverse LLMs.**
- ✗d) Is only for internal developer use.

---

18. What kind of API calls does LiteLLM typically unify?

- ✗a) File system calls.
- ✗b) Database queries.

- ✅**c) Large Language Model (LLM) API calls.**
- ❌d) Operating system calls.

---

19. A developer looking to leverage a wide array of commercial and open-source LLMs in their agent with minimal code changes would find which class most beneficial?

- ❌a) OpenAITTSModel
- ❌b) OpenAISTTModel
- ✅**c) LitellmModel**
- ❌d) VoiceModelProvider

---

20. The integration of LitellmModel into the Agents SDK demonstrates a design principle of:

- ❌a) Tightly coupling to specific services.
- ✅**b) Modularity and extensibility.**
- ❌c) Limiting external dependencies.
- ❌d) Prioritizing performance over flexibility.
