Created by https://github.com/samade747

https://x.com/samaddeveloper

Purpose to COVER UP OPENAI SDK ADVANCE LEVEL

Request: Read Once is Most Important

OpenAI Agents SDK Advanced Quiz: 450 Questions

Below is a set of 250 advanced-level multiple-choice questions designed to test deep knowledge of the OpenAI Agents SDK. Each question includes four options and the correct answer.

Agents (60 Questions)

1. **What happens when an Agent's** instructions **parameter is set to a function that takes** agent **and** context **as inputs?**
   - a) The function is ignored, and a default prompt is used
   - b) The function dynamically generates instructions based on the inputs
   - c) The function is executed only once during initialization
   - d) It raises a TypeError
   - **Correct answer: b**
2. **How does the** clone() **method of an Agent affect its properties?**
   - a) It creates a new Agent with identical properties unless overridden
   - b) It resets all properties to default values
   - c) It creates a new Agent with no properties set
   - d) It copies only the model property
   - **Correct answer: a**
3. **What is the effect of setting** model_settings.tool_choice **to** required**?**
   - a) The Agent must use at least one tool in every response
   - b) The Agent ignores all tools
   - c) The Agent uses tools only if explicitly instructed
   - d) It raises an error if no tools are provided
   - **Correct answer: a**
4. **When using a Pydantic model as** output_type**, what happens if the Agent's output does not conform to the model?**
   - a) The output is automatically corrected
   - b) A Pydantic ValidationError is raised

- c) The output is returned as plain text
- d) The Agent retries until the output conforms
- **Correct answer: b**

5. **How does the** context **parameter facilitate dependency injection in the Agents SDK?**
    - a) It stores the Agent's memory
    - b) It passes shared state to Agents, Tools, and Handoffs
    - c) It defines the execution environment
    - d) It manages API keys
    - **Correct answer: b**
6. **What is the purpose of the** hooks **property in the Agent class?**
    - a) To define custom tools
    - b) To handle lifecycle events like logging or data pre-fetching
    - c) To manage Handoffs
    - d) To configure the model's temperature
    - **Correct answer: b**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
7. **How can you ensure an Agent uses a specific model, such as** gpt-4o**, instead of the default?**
    - a) Set the model parameter in the Agent constructor
    - b) Define it in the instructions
    - c) Use the Runner.set_model() method
    - d) It cannot be changed
    - **Correct answer: a**
8. **What is the default value of** max_turns **in the** Runner.run() **method?**
    - a) 10
    - b) 50
    - c) 100
    - d) None (unlimited)
    - **Correct answer: cnone**
9. **What happens if an Agent's** instructions **parameter is set to** None**?**
    - a) The Agent uses a default system prompt
    - b) The Agent fails to initialize
    - c) The Agent operates without instructions
    - d) It raises a ValueError
    - **Correct answer: a**
10. **How does the** name **parameter in the Agent constructor affect tracing?**
    - a) It has no effect on tracing
    - b) It identifies the Agent in trace logs and debugging
    - c) It sets the display name for the UI
    - d) It defines the Agent's role
    - **Correct answer: b**
11. **What is the purpose of the** output_type **parameter in the Agent constructor?**
    - a) To specify the input format
    - b) To define the structured output format, e.g., Pydantic model

    - c) To configure the tool schema
    - d) To set the response language
    - **Correct answer: b**
12. **How does the Agent handle async functions as** instructions**?**
    - a) It awaits the function to resolve the instructions
    - b) It ignores async functions
    - c) It raises a TypeError
    - d) It converts them to sync functions
    - **Correct answer: a**
13. **What is the effect of setting** model_settings.temperature **to 0?**
    - a) The Agent generates more random responses
    - b) The Agent generates deterministic responses
    - c) The Agent ignores the temperature setting
    - d) It raises an error
    - **Correct answer: b**
14. **How can you modify an Agent's instructions after initialization?**
    - a) By calling the update_instructions() method
    - b) By using the clone() method with new instructions
    - c) By directly modifying the instructions attribute
    - d) It is not possible
    - **Correct answer: b**
15. **What is the role of the** AgentHooks **subclass in lifecycle events?**
    - a) It defines custom tools
    - b) It allows customization of logging or pre-fetching
    - c) It manages Handoffs
    - d) It configures the model
    - **Correct answer: b**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
16. **What happens if an Agent's** tools **list is empty and** tool_choice **is set to** required**?**
    - a) The Agent proceeds without tools
    - b) It raises an error
    - c) It uses a default tool
    - d) It ignores the tool_choice setting
    - **Correct answer: b**
17. **How does the Agent loop function in the SDK?**
    - a) It repeatedly calls the LLM until a condition is met
    - b) It executes tools in parallel
    - c) It manages multiple Agents simultaneously
    - d) It handles input validation
    - **Correct answer: a**
18. **What is the significance of the** Python-first **design in the Agents SDK?**
    - a) It requires Python for all operations
    - b) It leverages Python's language features for orchestration
    - c) It restricts the SDK to Python environments
    - d) It uses Python to generate prompts

  - **Correct answer: b**
19. **How can you access the Agent's context during execution?**
    - a) Through the context parameter in tools and handoffs
    - b) By calling Agent.get_context()
    - c) By accessing the context attribute directly
    - d) It is not accessible
    - **Correct answer: a**
20. **What is the purpose of the** max_tokens **parameter in** ModelSettings**?**
    - a) To limit the number of tools
    - b) To set the maximum length of the response
    - c) To define the input size
    - d) To configure the context window
    - **Correct answer: b**
21. **What happens if an Agent's** model **parameter is set to an invalid model name?**
    - a) It uses the default model
    - b) It raises an error during initialization
    - c) It ignores the model and proceeds
    - d) It retries with a fallback model
    - **Correct answer: b**
22. **How does the** clone() **method handle tool assignments?**
    - a) It resets all tools
    - b) It copies the original Agent's tools unless overridden
    - c) It requires reassigning tools
    - d) It ignores tools
    - **Correct answer: b**
23. **What is the effect of setting** model_settings.top_p **to 1.0?**
    - a) It disables nucleus sampling
    - b) It enables maximum randomness
    - c) It restricts output to top tokens
    - d) It raises an error
    - **Correct answer: a**

    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
24. **How can you implement custom logging for an Agent's lifecycle events?**
    - a) By subclassing AgentHooks and overriding methods
    - b) By setting the logs parameter in the Agent
    - c) By using the Runner.log() method
    - d) It is not possible
    - **Correct answer: a**
25. **What is the role of the** name **parameter in debugging?**
    - a) It sets the UI display name
    - b) It identifies the Agent in trace logs
    - c) It defines the Agent's role
    - d) It has no role in debugging
    - **Correct answer: b**
26. **How does the SDK handle multiple Agents in a single workflow?**

    - a) Through Handoffs and the Runner
    - b) By running Agents in parallel
    - c) By merging Agents into one
    - d) It does not support multiple Agents
    - **Correct answer: a**
27. **What is the purpose of the** input_guardrails **attribute?**
    - a) To validate user inputs before processing
    - b) To store input data
    - c) To configure tools
    - d) To manage Handoffs
    - **Correct answer: a**
28. **How can you specify a custom system prompt for an Agent?**
    - a) By setting the instructions parameter
    - b) By using the system_prompt attribute
    - c) By defining it in ModelSettings
    - d) It is not possible
    - **Correct answer: a**
29. **What happens if an async function is used as** instructions **without awaiting?**
    - a) It raises a RuntimeError
    - b) The SDK automatically awaits it
    - c) The function is ignored
    - d) It uses a default prompt
    - **Correct answer: b**
30. **How does the** output_guardrails **attribute function?**
    - a) It validates the Agent's output before returning
    - b) It stores output data
    - c) It configures the output format
    - d) It manages tool execution
    - **Correct answer: a**
31. **What is the default behavior of an Agent without tools?**
    - a) It raises an error
    - b) It operates as a text-based LLM
    - c) It uses a default tool
    - d) It delegates to another Agent
    - **Correct answer: b**
32. **How can you ensure an Agent retries on tool failure?**
    - a) By setting retry_on_failure in ModelSettings
    - b) By configuring the Runner's max_retries
    - c) By using a custom AgentHooks subclass
    - d) It is not supported
    - **Correct answer: b**
33. **What is the purpose of the** tool_choice **parameter in** ModelSettings**?**
    - a) To select the programming language
    - b) To control whether tools are used
    - c) To define the output format
    - d) To manage context
    - **Correct answer: b**
34. **How does the SDK handle Agent output when** output_type **is not set?**

- a) It returns a Pydantic model
- b) It returns plain text
- c) It raises an error
- d) It uses a default JSON schema
- **Correct answer: b**

35. **What is the effect of setting** model_settings.frequency_penalty **to a high value?**
    - a) It increases output diversity
    - b) It reduces repetition in responses
    - c) It limits tool usage
    - d) It disables the model
    - **Correct answer: b**
36. **How can you access an Agent's current state during execution?**
    - a) Through the Runner.get_state() method
    - b) By inspecting the context object
    - c) Via the Agent.state attribute
    - d) It is not accessible
    - **Correct answer: b**
37. **What is the purpose of the** max_turns **parameter in** Runner.run()**?**
    - a) To limit the number of tool calls
    - b) To set the maximum Agent iterations
    - c) To define the context size
    - d) To configure retry attempts
    - **Correct answer: b**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
38. **How does the SDK handle concurrent Agent execution?**
    - a) It runs Agents in parallel by default
    - b) It requires manual threading
    - c) It uses the Runner for sequential execution
    - d) It does not support concurrency
    - **Correct answer: c**
39. **What is the role of the** default_tool_choice **parameter in the Agent constructor?**
    - a) To set a default tool for all executions
    - b) To override ModelSettings.tool_choice
    - c) To define the tool schema
    - d) To manage Handoffs
    - **Correct answer: b**
40. **How can you implement custom validation for an Agent's input?**
    - a) By setting input_guardrails
    - b) By using a Pydantic model
    - c) By defining a custom tool
    - d) By modifying the context
    - **Correct answer: a**
41. **What happens if an Agent's** tools **list contains an invalid tool?**
    - a) It ignores the invalid tool
    - b) It raises a ValueError during initialization

- c) It uses a default tool
- d) It retries with a fallback tool
- **Correct answer: b**

42. **How does the** clone() **method handle** context**?**
    - a) It creates a new context
    - b) It copies the original context
    - c) It resets the context
    - d) It ignores the context
    - **Correct answer: b**
43. **What is the purpose of the** presence_penalty **in** ModelSettings**?**
    - a) To penalize tool usage
    - b) To encourage new topics in responses
    - c) To limit response length
    - d) To manage context size
    - **Correct answer: b**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
44. **How can you dynamically update an Agent's tools during execution?**
    - a) By modifying the tools attribute
    - b) By using the clone() method with new tools
    - c) By calling Agent.update_tools()
    - d) It is not possible
    - **Correct answer: b**
45. **What is the effect of setting** model_settings.stop **to a list of strings?**
    - a) It stops the Agent when those strings appear in the output
    - b) It filters the input for those strings
    - c) It limits tool names
    - d) It raises an error
    - **Correct answer: a**
46. **How does the SDK handle Agent errors during execution?**
    - a) It retries automatically
    - b) It raises an exception
    - c) It ignores the error and continues
    - d) It delegates to another Agent
    - **Correct answer: b**
47. **What is the role of the** AgentHooks.on_tool_call **method?**
    - a) To execute tools
    - b) To log or modify tool call behavior
    - c) To validate tool outputs
    - d) To manage Handoffs
    - **Correct answer: b**
48. **How can you ensure an Agent uses a specific tool first?**
    - a) By setting tool_choice to the tool's name
    - b) By ordering the tools list
    - c) By defining it in the instructions
    - d) It is not possible

- **Correct answer: a**

49. **What happens if an Agent's** output_type **is set to an invalid Pydantic model?**
    - a) It uses plain text output
    - b) It raises a TypeError
    - c) It ignores the output_type
    - d) It retries with a default model
    - **Correct answer: b**
50. **How does the SDK support multi-agent collaboration?**
    - a) Through Handoffs and shared context
    - b) By merging Agents
    - c) By running Agents in parallel
    - d) It does not support collaboration
    - **Correct answer: a**
51. **What is the purpose of the** AgentHooks.on_response **method?**
    - a) To modify the Agent's response before returning
    - b) To execute tools
    - c) To manage context
    - d) To handle Handoffs
    - **Correct answer: a**
52. **How can you limit an Agent's response length?**
    - a) By setting max_tokens in ModelSettings
    - b) By using a custom output_type
    - c) By defining it in the instructions
    - d) By modifying the context
    - **Correct answer: a**
53. **What is the effect of setting** tool_choice **to** none**?**
    - a) The Agent cannot use tools
    - b) The Agent uses tools randomly
    - c) The Agent uses all tools
    - d) It raises an error
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
54. **How does the SDK handle async tool execution?**
    - a) It awaits async tools automatically
    - b) It ignores async tools
    - c) It raises a TypeError
    - d) It converts them to sync tools
    - **Correct answer: a**
55. **What is the role of the** default_tool_choice **parameter?**
    - a) To set a fallback tool
    - b) To override ModelSettings.tool_choice
    - c) To define the tool schema
    - d) To manage context
    - **Correct answer: b**
56. **How can you debug an Agent's tool calls?**

    - a) By enabling tracing in the OpenAI Dashboard
    - b) By setting debug=True in the Agent
    - c) By using the Runner.debug() method
    - d) It is not possible
    - **Correct answer: a**
57. **What happens if an Agent's** instructions **function raises an error?**
    - a) It uses a default prompt
    - b) It raises the error during execution
    - c) It ignores the function
    - d) It retries the function
    - **Correct answer: b**
58. **How does the SDK ensure type safety for tool inputs?**
    - a) Through Pydantic models
    - b) By using JSON schemas
    - c) By manual validation
    - d) It does not ensure type safety
    - **Correct answer: a**
59. **What is the purpose of the** AgentHooks.on_error **method?**
    - a) To handle errors during tool execution
    - b) To log errors during Agent execution
    - c) To validate inputs
    - d) To manage Handoffs
    - **Correct answer: b**
60. **How can you customize an Agent's behavior for specific inputs?**
    - a) By using dynamic instructions
    - b) By modifying the model parameter
    - c) By setting tool_choice to auto
    - d) By defining a new output_type
    - **Correct answer: a**

Created by https://github.com/samade747

Tools (60 Questions)

61. **What types of tools are supported by the OpenAI Agents SDK?**
    - a) Only hosted tools
    - b) Hosted tools, function tools, and agents as tools
    - c) Only function tools
    - d) Only agents as tools
    - **Correct answer: b**
62. **How do you define a custom tool using a Python function?**
    - a) By using the @function_tool decorator
    - b) By subclassing the Tool class
    - c) By passing the function directly to the Agent
    - d) By registering it with the Runner

- **Correct answer: a**
- **Created by https://github.com/samade747**
- https://x.com/samaddeveloper
-

63. **What does the** @function_tool **decorator automatically generate?**
    - a) Tool name, description, and schema
    - b) Only the tool name
    - c) Only the schema
    - d) Nothing, it requires manual configuration
    - **Correct answer: a**
64. **Which libraries does the** @function_tool **decorator use to create a tool's schema?**
    - a) inspect, griffe, and pydantic
    - b) numpy and pandas
    - c) requests and urllib
    - d) json and yaml
    - **Correct answer: a**
65. **How can you use an Agent as a tool?**
    - a) By calling the as_tool method
    - b) By adding it to the tools list
    - c) By using the handoff() function
    - d) It is not possible
    - **Correct answer: a**
66. **What is the purpose of the** tool_name **parameter in the** as_tool **method?**
    - a) To set the Agent's display name
    - b) To override the default tool name
    - c) To define the tool's schema
    - d) To manage context
    - **Correct answer: b**
67. **What happens if a tool's schema does not match its function signature?**
    - a) The SDK automatically corrects it
    - b) It raises a ValueError
    - c) The tool is ignored
    - d) It uses a default schema
    - **Correct answer: b**
68. **How does the SDK handle async function tools?**
    - a) It awaits them automatically
    - b) It ignores async functions
    - c) It raises a TypeError
    - d) It converts them to sync functions
    - **Correct answer: a**
69. **What is the role of the** tool_description **parameter in** @function_tool**?**
    - a) It overrides the function's docstring
    - b) It sets the tool's name
    - c) It defines the input schema
    - d) It manages context
    - **Correct answer: a**

70. **How can you validate tool inputs?**
    - a) By using a Pydantic model in the function signature
    - b) By setting input_guardrails
    - c) By defining a custom schema
    - d) By using the Runner
    - **Correct answer: a**
71. **What is the default name of a tool created with** @function_tool**?**
    - a) The function's name
    - b) tool_<function_name>
    - c) custom_tool
    - d) It requires manual specification
    - **Correct answer: a**
72. **How does the SDK handle tool errors during execution?**
    - a) It retries the tool automatically
    - b) It raises an exception
    - c) It ignores the error
    - d) It delegates to another tool
    - **Correct answer: b**
73. **What is the purpose of hosted tools like** WebSearchTool**?**
    - a) To provide pre-built functionality
    - b) To manage Agent memory
    - c) To handle Handoffs
    - d) To validate inputs
    - **Correct answer: a**
74. **How can you customize the schema of a function tool?**
    - a) By defining a Pydantic model in the function signature
    - b) By setting schema_override
    - c) By using the tool_schema parameter
    - d) It is not possible
    - **Correct answer: a**
75. **What happens if a tool returns an invalid output type?**
    - a) The SDK converts it to a string
    - b) It raises a TypeError
    - c) It ignores the output
    - d) It retries the tool
    - **Correct answer: b**
76. **How does the** as_tool **method handle Agent output?**
    - a) It returns the Agent's output as a tool response
    - b) It ignores the Agent's output
    - c) It converts the output to a Pydantic model
    - d) It raises an error
    - **Correct answer: a**
77. **What is the role of the** tool_description **in the** as_tool **method?**
    - a) To override the Agent's instructions
    - b) To describe the tool's purpose to the LLM
    - c) To define the input schema
    - d) To manage context
    - **Correct answer: b**

78. **How can you ensure a tool is only called with specific inputs?**
    - a) By using a Pydantic model with validation
    - b) By setting input_guardrails
    - c) By defining a custom tool_choice
    - d) By modifying the context
    - **Correct answer: a**
79. **What happens if a tool's docstring is missing when using** @function_tool**?**
    - a) It uses a default description
    - b) It raises a ValueError
    - c) It ignores the description
    - d) It generates a description from the function name
    - **Correct answer: c**
80. **How does the SDK handle tool execution order?**
    - a) It follows the order in the tools list
    - b) It uses the LLM's decision
    - c) It executes tools in parallel
    - d) It requires manual specification
    - **Correct answer: b**
81. **What is the purpose of the** tool_name_override **parameter in** @function_tool**?**
    - a) To set the tool's description
    - b) To override the default tool name
    - c) To define the schema
    - d) To manage context
    - **Correct answer: b**
82. **How can you debug tool execution?**
    - a) By enabling tracing in the OpenAI Dashboard
    - b) By setting debug=True in the tool
    - c) By using the Runner.debug() method
    - d) It is not possible
    - **Correct answer: a**
83. **What happens if a tool's input does not match its Pydantic model?**
    - a) It raises a ValidationError
    - b) It uses default values
    - c) It ignores the input
    - d) It retries the tool
    - **Correct answer: a**
84. **How does the SDK handle tools with async dependencies?**
    - a) It awaits them automatically
    - b) It ignores async dependencies
    - c) It raises a TypeError
    - d) It converts them to sync dependencies
    - **Correct answer: a**
85. **What is the role of the** griffe **library in** @function_tool**?**
    - a) To generate tool schemas
    - b) To manage async execution
    - c) To validate inputs
    - d) To handle tracing
    - **Correct answer: a**

86. **How can you specify a tool's input schema?**
    - a) By using a Pydantic model in the function signature
    - b) By setting schema_override
    - c) By defining a JSON schema
    - d) By using the tool_schema parameter
    - **Correct answer: a**
87. **What is the effect of using a hosted tool like** FileSearchTool**?**
    - a) It enables file-based operations
    - b) It manages Agent memory
    - c) It handles Handoffs
    - d) It validates outputs
    - **Correct answer: a**
88. **How does the SDK handle tool timeouts?**
    - a) It retries the tool
    - b) It raises a TimeoutError
    - c) It ignores the timeout
    - d) It delegates to another tool
    - **Correct answer: b**
89. **What is the purpose of the** tool_call_limit **parameter in** Runner.run()**?**
    - a) To limit the number of tool calls
    - b) To set the maximum response length
    - c) To define the context size
    - d) To manage retries
    - **Correct answer: a**
90. **How can you ensure a tool's output is structured?**
    - a) By returning a Pydantic model
    - b) By setting output_type in the tool
    - c) By defining a JSON schema
    - d) By using the Runner
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
91. **What happens if a tool is called with missing required parameters?**
    - a) It uses default values
    - b) It raises a ValidationError
    - c) It ignores the call
    - d) It retries the call
    - **Correct answer: b**
92. **How does the** as_tool **method handle Agent errors?**
    - a) It propagates the error to the calling Agent
    - b) It ignores the error
    - c) It retries the Agent
    - d) It uses a default response
    - **Correct answer: a**
93. **What is the role of the** inspect **library in** @function_tool**?**
    - a) To generate tool schemas

- b) To manage async execution
- c) To validate inputs
- d) To handle tracing
- **Correct answer: a**

94. **How can you prioritize a specific tool in an Agent's workflow?**
    - a) By setting tool_choice to the tool's name
    - b) By ordering the tools list
    - c) By defining it in the instructions
    - d) It is not possible
    - **Correct answer: a**
95. **What happens if a tool's function raises an exception?**
    - a) The SDK catches and logs it
    - b) It propagates to the Agent
    - c) It ignores the exception
    - d) It retries the tool
    - **Correct answer: b**
96. **How does the SDK support complex tool inputs?**
    - a) Through Pydantic models
    - b) By using JSON schemas
    - c) By manual validation
    - d) It does not support complex inputs
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
97. **What is the purpose of the** WebSearchTool**?**
    - a) To perform web searches
    - b) To manage Agent memory
    - c) To handle Handoffs
    - d) To validate inputs
    - **Correct answer: a**
98. **How can you debug a tool's execution path?**
    - a) By enabling tracing in the OpenAI Dashboard
    - b) By setting debug=True in the tool
    - c) By using the Runner.debug() method
    - d) It is not possible
    - **Correct answer: a**
99. **What happens if a tool's output exceeds the** max_tokens **limit?**
    - a) It is truncated
    - b) It raises an error
    - c) It is ignored
    - d) It retries the tool
    - **Correct answer: b**
100. **How does the SDK handle tools with multiple outputs?**
    - a) It supports multiple outputs via Pydantic models
    - b) It converts them to a single string
    - c) It raises a TypeError

- d) It ignores additional outputs
- **Correct answer: a**

101. **What is the role of the** pydantic **library in tools?**
    - a) To validate and generate schemas
    - b) To manage async execution
    - c) To handle tracing
    - d) To execute tools
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
102. **How can you ensure a tool is reusable across multiple Agents?**
    - a) By defining it with @function_tool
    - b) By registering it with the Runner
    - c) By using a Pydantic model
    - d) By setting reusable=True
    - **Correct answer: a**
103. **What happens if a tool's schema is invalid?**
    - a) It uses a default schema
    - b) It raises a ValueError
    - c) It ignores the schema
    - d) It retries the tool
    - **Correct answer: b**
104. **How does the SDK handle tool dependencies?**
    - a) Through the context object
    - b) By using a dependency manager
    - c) By manual configuration
    - d) It does not support dependencies
    - **Correct answer: a**
105. **What is the purpose of the** tool_call_limit **in** Runner.run()**?**
    - a) To limit the number of tool calls
    - b) To set the maximum response length
    - c) To define the context size
    - d) To manage retries
    - **Correct answer: a**
106. **How can you customize a tool's description?**
    - a) By setting the docstring or tool_description
    - b) By using a Pydantic model
    - c) By defining a JSON schema
    - d) By modifying the context
    - **Correct answer: a**
107. **What happens if a tool is called with an invalid context?**
    - a) It raises a ValueError
    - b) It uses a default context
    - c) It ignores the context
    - d) It retries the call
    - **Correct answer: a**

108. **How does the** as_tool **method handle Agent inputs?**
   - a) It validates them with a Pydantic model
   - b) It ignores the inputs
   - c) It converts them to strings
   - d) It raises an error
   - **Correct answer: a**

109. **What is the role of the** FileSearchTool**?**
   - a) To search files for relevant data
   - b) To manage Agent memory
   - c) To handle Handoffs
   - d) To validate inputs
   - **Correct answer: a**

110. **How can you ensure a tool's output is compatible with the Agent's** output_type**?**
   - a) By returning a Pydantic model
   - b) By setting output_type in the tool
   - c) By defining a JSON schema
   - d) By using the Runner
   - **Correct answer: a**

111. **What happens if a tool's execution time exceeds the Runner's timeout?**
   - a) It raises a TimeoutError
   - b) It retries the tool
   - c) It ignores the timeout
   - d) It delegates to another tool
   - **Correct answer: a**
   - **Created by https://github.com/samade747**
   - https://x.com/samaddeveloper
   -

112. **How does the SDK handle tools with dynamic schemas?**
   - a) Through Pydantic models
   - b) By using JSON schemas
   - c) By manual validation
   - d) It does not support dynamic schemas
   - **Correct answer: a**

113. **What is the purpose of the** tool_name_override **in the** as_tool **method?**
   - a) To set the tool's description
   - b) To override the default tool name
   - c) To define the schema
   - d) To manage context
   - **Correct answer: b**

114. **How can you test a tool's functionality before adding it to an Agent?**
   - a) By calling it directly with a test input
   - b) By using the Runner.test_tool() method
   - c) By enabling tracing
   - d) It is not possible
   - **Correct answer: a**

115. **What happens if a tool's output is** None**?**

    - a) It raises a ValueError
    - b) It is treated as a valid output
    - c) It retries the tool
    - d) It ignores the output
    - **Correct answer: b**
116. **How does the SDK handle tool versioning?**
    - a) It does not support versioning
    - b) Through the tool_version parameter
    - c) By using Pydantic models
    - d) By defining a version in the schema
    - **Correct answer: a**
117. **What is the role of the** tool_description_override **in** @function_tool**?**
    - a) To override the function's docstring
    - b) To set the tool's name
    - c) To define the input schema
    - d) To manage context
    - **Correct answer: a**
118. **How can you ensure a tool is only called by specific Agents?**
    - a) By using the context object
    - b) By setting restrict_to_agents
    - c) By defining it in the instructions
    - d) It is not possible
    - **Correct answer: a**
119. **What happens if a tool's schema includes an unsupported type?**
    - a) It raises a TypeError
    - b) It uses a default type
    - c) It ignores the type
    - d) It retries the schema
    - **Correct answer: a**
120. **How does the SDK handle tools with external API dependencies?**
    - a) Through the context object
    - b) By using a dependency manager
    - c) By manual configuration
    - d) It does not support external APIs
    - **Correct answer: a**

Created by https://github.com/samade747

Handoffs (50 Questions)

121. **What is the primary purpose of Handoffs in the OpenAI Agents SDK?**
   - a) To manage Agent memory
   - b) To enable task delegation between Agents
   - c) To validate inputs
   - d) To execute tools
   - **Correct answer: b**
122. **How are Handoffs represented to the LLM?**
   - a) As separate Agents
   - b) As tools that the Agent can call
   - c) As part of the instructions
   - d) As context objects
   - **Correct answer: b**
123. **What function is used to create a Handoff object?**
   - a) handoff()
   - b) delegate()
   - c) transfer()
   - d) agent_handoff()
   - **Correct answer: a**
124. **What is the default tool name for a Handoff?**
   - a) delegate_to_<agent_name>
   - b) transfer_to_<agent_name>
   - c) handoff_<agent_name>
   - d) agent_<agent_name>
   - **Correct answer: b**
125. **How can you customize a Handoff's tool name?**
   - a) By setting tool_name_override
   - b) By defining it in the instructions
   - c) By using a Pydantic model
   - d) It is not possible
   - **Correct answer: a**
126. **What is the purpose of the** on_handoff **callback?**
   - a) To execute custom code during a Handoff
   - b) To validate inputs
   - c) To manage context
   - d) To define the schema
   - **Correct answer: a**
127. **How does the** input_type **parameter in** handoff() **function?**
   - a) It specifies the Handoff's output format
   - b) It defines a Pydantic model for Handoff inputs
   - c) It sets the context type
   - d) It manages tool execution
   - **Correct answer: b**

   - **Created by https://github.com/samade747**
   - https://x.com/samaddeveloper

-

128. **What happens if a Handoff's input does not match its** input_type**?**
    - a) It raises a ValidationError
    - b) It uses a default input
    - c) It ignores the input
    - d) It retries the Handoff
    - **Correct answer: a**
129. **How can you include Handoff information in an Agent's prompt?**
    - a) By using RECOMMENDED_PROMPT_PREFIX
    - b) By setting handoff_prompt
    - c) By defining it in ModelSettings
    - d) It is not possible
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
130. **What is the role of the** input_filter **parameter in** handoff()**?**
    - a) To validate and transform Handoff inputs
    - b) To define the output format
    - c) To manage context
    - d) To execute tools
    - **Correct answer: a**
131. **How does the SDK handle Handoff errors?**
    - a) It raises an exception
    - b) It retries the Handoff
    - c) It ignores the error
    - d) It delegates to another Agent
    - **Correct answer: a**
132. **What is the effect of setting** tool_description_override **in** handoff()**?**
    - a) It overrides the default tool description
    - b) It sets the Handoff's name
    - c) It defines the input schema
    - d) It manages context
    - **Correct answer: a**
133. **How can you debug a Handoff's execution?**
    - a) By enabling tracing in the OpenAI Dashboard
    - b) By setting debug=True in the Handoff
    - c) By using the Runner.debug() method
    - d) It is not possible
    - **Correct answer: a**
134. **What happens if a Handoff's target Agent is invalid?**
    - a) It raises a ValueError
    - b) It uses a default Agent
    - c) It ignores the Handoff
    - d) It retries the Handoff
    - **Correct answer: a**

135. **How does the SDK handle multiple Handoffs in a single workflow?**
    - a) Through the Runner and shared context
    - b) By running Handoffs in parallel
    - c) By merging Handoffs
    - d) It does not support multiple Handoffs
    - **Correct answer: a**
136. **What is the purpose of the** handoffs **parameter in the Agent constructor?**
    - a) To define available Handoffs
    - b) To manage context
    - c) To validate inputs
    - d) To execute tools
    - **Correct answer: a**
137. **How can you ensure a Handoff is only triggered for specific inputs?**
    - a) By using input_filter
    - b) By setting tool_choice
    - c) By defining it in the instructions
    - d) By modifying the context
    - **Correct answer: a**
138. **What happens if a Handoff's** on_handoff **callback raises an error?**
    - a) It propagates the error
    - b) It ignores the callback
    - c) It retries the Handoff
    - d) It uses a default callback
    - **Correct answer: a**

139. **How does the** prompt_with_handoff_instructions **function assist Handoffs?**
    - a) It includes Handoff details in the Agent's prompt
    - b) It validates Handoff inputs
    - c) It manages context
    - d) It executes the Handoff
    - **Correct answer: a**
140. **What is the role of the** tool_name_override **in** handoff()**?**
    - a) To override the default tool name
    - b) To set the Handoff's description
    - c) To define the input schema
    - d) To manage context
    - **Correct answer: a**
141. **How can you validate a Handoff's input?**
    - a) By using a Pydantic model with input_type
    - b) By setting input_guardrails
    - c) By defining a custom schema

- d) By using the Runner
- **Correct answer: a**

142. **What happens if a Handoff's target Agent is not available?**
    - a) It raises a ValueError
    - b) It uses a default Agent
    - c) It ignores the Handoff
    - d) It retries the Handoff
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
143. **How does the SDK handle Handoff timeouts?**
    - a) It raises a TimeoutError
    - b) It retries the Handoff
    - c) It ignores the timeout
    - d) It delegates to another Agent
    - **Correct answer: a**
144. **What is the purpose of the** RECOMMENDED_PROMPT_PREFIX**?**
    - a) To include Handoff instructions in the prompt
    - b) To define the Agent's role
    - c) To validate inputs
    - d) To manage context
    - **Correct answer: a**
145. **How can you customize a Handoff's behavior?**
    - a) By using the on_handoff callback
    - b) By setting handoff_behavior
    - c) By defining it in the instructions
    - d) By modifying the context
    - **Correct answer: a**
146. **What happens if a Handoff's input is invalid?**
    - a) It raises a ValidationError
    - b) It uses a default input
    - c) It ignores the input
    - d) It retries the Handoff
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
147. **How does the SDK handle Handoff execution order?**
    - a) It follows the LLM's decision
    - b) It executes Handoffs in parallel
    - c) It requires manual specification
    - d) It follows the handoffs list order
    - **Correct answer: a**
148. **What is the role of the** input_type **in** handoff()**?**
    - a) To define a Pydantic model for inputs

- b) To set the output format
- c) To manage context
- d) To execute tools
- **Correct answer: a**

149. **How can you debug a Handoff's input validation?**
    - a) By enabling tracing in the OpenAI Dashboard
    - b) By setting debug=True in the Handoff
    - c) By using the Runner.debug() method
    - d) It is not possible
    - **Correct answer: a**
150. **What happens if a Handoff's** on_handoff **callback is not provided?**
    - a) It uses a default callback
    - b) It skips the callback
    - c) It raises a ValueError
    - d) It retries the Handoff
    - **Correct answer: b**
151. **How does the SDK handle Handoffs with complex inputs?**
    - a) Through Pydantic models
    - b) By using JSON schemas
    - c) By manual validation
    - d) It does not support complex inputs
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
152. **What is the effect of setting** tool_description_override **in** handoff()**?**
    - a) It overrides the default tool description
    - b) It sets the Handoff's name
    - c) It defines the input schema
    - d) It manages context
    - **Correct answer: a**
153. **How can you ensure a Handoff is only triggered by specific Agents?**
    - a) By using the context object
    - b) By setting restrict_to_agents
    - c) By defining it in the instructions
    - d) It is not possible
    - **Correct answer: a**
154. **What happens if a Handoff's target Agent raises an error?**
    - a) It propagates the error
    - b) It ignores the error
    - c) It retries the Handoff
    - d) It uses a default Agent
    - **Correct answer: a**
155. **How does the SDK support multi-agent Handoffs?**
    - a) Through the Runner and shared context
    - b) By running Handoffs in parallel
    - c) By merging Handoffs

- d) It does not support multi-agent Handoffs
- **Correct answer: a**

156. **What is the purpose of the** tool_name_override **in** handoff()**?**
    - a) To override the default tool name
    - b) To set the Handoff's description
    - c) To define the input schema
    - d) To manage context
    - **Correct answer: a**
157. **How can you validate a Handoff's output?**
    - a) By using output_guardrails
    - b) By setting output_type
    - c) By defining a custom schema
    - d) By using the Runner
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
158. **What happens if a Handoff's context is invalid?**
    - a) It raises a ValueError
    - b) It uses a default context
    - c) It ignores the context
    - d) It retries the Handoff
    - **Correct answer: a**
159. **How does the SDK handle Handoff retries?**
    - a) Through the Runner's max_retries
    - b) By using a custom callback
    - c) It does not support retries
    - d) By manual configuration
    - **Correct answer: a**
160. **What is the role of the** prompt_with_handoff_instructions **function?**
    - a) To include Handoff details in the prompt
    - b) To validate Handoff inputs
    - c) To manage context
    - d) To execute the Handoff
    - **Correct answer: a**
161. **How can you ensure a Handoff's input is transformed before execution?**
    - a) By using input_filter
    - b) By setting input_type
    - c) By defining a custom schema
    - d) By modifying the context
    - **Correct answer: a**
162. **What happens if a Handoff's** input_type **is not a Pydantic model?**
    - a) It raises a TypeError
    - b) It uses a default model
    - c) It ignores the input_type
    - d) It retries the Handoff

- **Correct answer: a**
  Created by https://github.com/samade747
- https://x.com/samaddeveloper
-
-

163. **How does the SDK handle Handoff tracing?**
    - a) Through the OpenAI Dashboard
    - b) By setting trace=True
    - c) By using the Runner.trace() method
    - d) It does not support tracing
    - **Correct answer: a**
164. **What is the purpose of the** on_handoff **callback in** handoff()**?**
    - a) To execute custom code during a Handoff
    - b) To validate inputs
    - c) To manage context
    - d) To define the schema
    - **Correct answer: a**
165. **How can you ensure a Handoff is only triggered for valid inputs?**
    - a) By using a Pydantic model with input_type
    - b) By setting input_guardrails
    - c) By defining a custom schema
    - d) By using the Runner
    - **Correct answer: a**
166. **What happens if a Handoff's** tool_name_override **is invalid?**
    - a) It raises a ValueError
    - b) It uses the default name
    - c) It ignores the override
    - d) It retries the Handoff
    - **Correct answer: a**
167. **How does the SDK handle Handoffs with dynamic inputs?**
    - a) Through Pydantic models
    - b) By using JSON schemas
    - c) By manual validation
    - d) It does not support dynamic inputs
    - **Correct answer: a**
168. **What is the effect of setting** tool_description_override **in** handoff()**?**
    - a) It overrides the default tool description
    - b) It sets the Handoff's name
    - c) It defines the input schema
    - d) It manages context
    - **Correct answer: a**
169. **How can you debug a Handoff's execution path?**
    - a) By enabling tracing in the OpenAI Dashboard
    - b) By setting debug=True in the Handoff
    - c) By using the Runner.debug() method
    - d) It is not possible
    - **Correct answer: a**
170. **What happens if a Handoff's target Agent is not configured?**

- a) It raises a ValueError
- b) It uses a default Agent
- c) It ignores the Handoff
- d) It retries the Handoff
- **Correct answer: a**

Created by https://github.com/samade747

Guardrails (40 Questions)

171. **What is the primary purpose of Guardrails in the OpenAI Agents SDK?**
    - a) To manage Agent memory
    - b) To validate inputs and outputs
    - c) To execute tools
    - d) To handle Handoffs
    - **Correct answer: b**
172. **How are** input_guardrails **executed?**
    - a) In parallel with Agent execution
    - b) Before Agent execution
    - c) After Agent execution
    - d) They are not executed
    - **Correct answer: a**
173. **What happens if an input fails** input_guardrails **validation?**
    - a) It raises a ValidationError
    - b) It uses a default input
    - c) It ignores the input
    - d) It retries the input
    - **Correct answer: a**
174. **How can you define custom** input_guardrails**?**
    - a) By using a Pydantic model
    - b) By setting a validation function
    - c) By defining a JSON schema
    - d) By using the Runner
    - **Correct answer: b**
175. **What is the role of** output_guardrails**?**
    - a) To validate Agent outputs
    - b) To manage context
    - c) To execute tools
    - d) To handle Handoffs
    - **Correct answer: a**
176. **How does the SDK handle Guardrail errors?**

- a) It raises an exception
- b) It retries the validation
- c) It ignores the error
- d) It delegates to another Agent
- **Correct answer: a**

177. **What happens if an output fails** output_guardrails **validation?**
    - a) It raises a ValidationError
    - b) It uses a default output
    - c) It ignores the output
    - d) It retries the output
    - **Correct answer: a**
178. **How can you debug Guardrail validation?**
    - a) By enabling tracing in the OpenAI Dashboard
    - b) By setting debug=True in the Guardrail
    - c) By using the Runner.debug() method
    - d) It is not possible
    - **Correct answer: a**
179. **What is the purpose of parallel execution in Guardrails?**
    - a) To improve performance
    - b) To manage context
    - c) To execute tools
    - d) To handle Handoffs
    - **Correct answer: a**
180. **How can you ensure a Guardrail only validates specific inputs?**
    - a) By using a custom validation function
    - b) By setting input_type
    - c) By defining a JSON schema
    - d) By modifying the context
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
181. **What happens if a Guardrail's validation function raises an error?**
    - a) It propagates the error
    - b) It ignores the error
    - c) It retries the validation
    - d) It uses a default validation
    - **Correct answer: a**
182. **How does the SDK handle Guardrail timeouts?**
    - a) It raises a TimeoutError
    - b) It retries the validation
    - c) It ignores the timeout
    - d) It delegates to another Guardrail
    - **Correct answer: a**
183. **What is the role of** input_guardrails **in multi-agent workflows?**
    - a) To validate inputs across all Agents
    - b) To manage context

- c) To execute tools
- d) To handle Handoffs
- **Correct answer: a**

184. **How can you customize** output_guardrails**?**
   - a) By setting a validation function
   - b) By using a Pydantic model
   - c) By defining a JSON schema
   - d) By using the Runner
   - **Correct answer: a**
185. **What happens if a Guardrail's validation function is invalid?**
   - a) It raises a ValueError
   - b) It uses a default validation
   - c) It ignores the validation
   - d) It retries the validation
   - **Correct answer: a**
186. **How does the SDK support complex Guardrail validations?**
   - a) Through Pydantic models
   - b) By using JSON schemas
   - c) By manual validation
   - d) It does not support complex validations
   - **Correct answer: a**
187. **What is the purpose of** output_guardrails **in multi-agent workflows?**
   - a) To validate outputs across all Agents
   - b) To manage context
   - c) To execute tools
   - d) To handle Handoffs
   - **Correct answer: a**
188. **How can you debug a Guardrail's validation path?**
   - a) By enabling tracing in the OpenAI Dashboard
   - b) By setting debug=True in the Guardrail
   - c) By using the Runner.debug() method
   - d) It is not possible
   - **Correct answer: a**
189. **What happens if a Guardrail's validation function returns** False**?**
   - a) It raises a ValidationError
   - b) It uses a default validation
   - c) It ignores the validation
   - d) It retries the validation
   - **Correct answer: a**
   - **Created by https://github.com/samade747**
   - https://x.com/samaddeveloper
   -
190. **How does the SDK handle Guardrail retries?**
   - a) Through the Runner's max_retries
   - b) By using a custom callback
   - c) It does not support retries
   - d) By manual configuration

- **Correct answer: a**

191. **What is the role of** input_guardrails **in error handling?**
    - a) To catch and validate input errors
    - b) To manage context
    - c) To execute tools
    - d) To handle Handoffs
    - **Correct answer: a**
192. **How can you ensure a Guardrail only validates specific outputs?**
    - a) By using a custom validation function
    - b) By setting output_type
    - c) By defining a JSON schema
    - d) By modifying the context
    - **Correct answer: a**
193. **What happens if a Guardrail's validation function is async?**
    - a) The SDK awaits it automatically
    - b) It ignores the function
    - c) It raises a TypeError
    - d) It converts it to a sync function
    - **Correct answer: a**
194. **How does the SDK handle Guardrail tracing?**
    - a) Through the OpenAI Dashboard
    - b) By setting trace=True
    - c) By using the Runner.trace() method
    - d) It does not support tracing
    - **Correct answer: a**
195. **What is the purpose of** output_guardrails **in error handling?**
    - a) To catch and validate output errors
    - b) To manage context
    - c) To execute tools
    - d) To handle Handoffs
    - **Correct answer: a**
196. **How can you ensure a Guardrail's validation is reusable?**
    - a) By defining a reusable validation function
    - b) By setting reusable=True
    - c) By using a Pydantic model
    - d) By modifying the context
    - **Correct answer: a**
197. **What happens if a Guardrail's validation function is missing?**
    - a) It raises a ValueError
    - b) It uses a default validation
    - c) It ignores the validation
    - d) It retries the validation
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
198. **How does the SDK handle Guardrails with complex validations?**

- a) Through Pydantic models
- b) By using JSON schemas
- c) By manual validation
- d) It does not support complex validations
- **Correct answer: a**

199. **What is the role of** input_guardrails **in security?**
    - a) To prevent malicious inputs
    - b) To manage context
    - c) To execute tools
    - d) To handle Handoffs
    - **Correct answer: a**
200. **How can you debug a Guardrail's validation logic?**
    - a) By enabling tracing in the OpenAI Dashboard
    - b) By setting debug=True in the Guardrail
    - c) By using the Runner.debug() method
    - d) It is not possible
    - **Correct answer: a**
201. **What happens if a Guardrail's validation function times out?**
    - a) It raises a TimeoutError
    - b) It retries the validation
    - c) It ignores the timeout
    - d) It delegates to another Guardrail
    - **Correct answer: a**
202. **How does the SDK handle Guardrail execution order?**
    - a) It follows the order in the guardrails list
    - b) It executes Guardrails in parallel
    - c) It requires manual specification
    - d) It follows the LLM's decision
    - **Correct answer: b**
203. **What is the purpose of** output_guardrails **in multi-agent workflows?**
    - a) To validate outputs across all Agents
    - b) To manage context
    - c) To execute tools
    - d) To handle Handoffs
    - **Correct answer: a**
204. **How can you ensure a Guardrail's validation is async-compatible?**
    - a) By defining an async validation function
    - b) By setting async=True
    - c) By using a Pydantic model
    - d) By modifying the context
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
205. **What happens if a Guardrail's validation function returns** None**?**
    - a) It raises a ValueError
    - b) It is treated as a valid validation

- c) It retries the validation
- d) It ignores the validation
- **Correct answer: a**

206. **How does the SDK support custom Guardrail logic?**
    - a) Through custom validation functions
    - b) By using JSON schemas
    - c) By manual validation
    - d) It does not support custom logic
    - **Correct answer: a**
207. **What is the role of** input_guardrails **in performance?**
    - a) To improve input validation speed
    - b) To manage context
    - c) To execute tools
    - d) To handle Handoffs
    - **Correct answer: a**
208. **How can you ensure a Guardrail's validation is reusable across Agents?**
    - a) By defining a reusable validation function
    - b) By setting reusable=True
    - c) By using a Pydantic model
    - d) By modifying the context
    - **Correct answer: a**
209. **What happens if a Guardrail's validation function is invalid?**
    - a) It raises a ValueError
    - b) It uses a default validation
    - c) It ignores the validation
    - d) It retries the validation
    - **Correct answer: a**
210. **How does the SDK handle Guardrails with async dependencies?**
    - a) It awaits them automatically
    - b) It ignores async dependencies
    - c) It raises a TypeError
    - d) It converts them to sync dependencies
    - **Correct answer: a**

Created by https://github.com/samade747

Runner (40 Questions)

211. **What is the primary purpose of the Runner in the OpenAI Agents SDK?**
    - a) To manage Agent execution
    - b) To validate inputs
    - c) To execute tools
    - d) To handle Handoffs
    - **Correct answer: a**
212. **What does the** Runner.run_sync() **method do?**
    - a) Executes an Agent synchronously
    - b) Streams Agent output in real-time
    - c) Configures Agent tools
    - d) Validates input data
    - **Correct answer: a**
213. **How does the Runner support streaming execution?**
    - a) Through the run() method with streaming enabled
    - b) By setting stream=True in run_sync()
    - c) By using a custom callback
    - d) It does not support streaming
    - **Correct answer: a**
214. **What is the purpose of built-in tracing in the Runner?**
    - a) To visualize and debug workflows
    - b) To store user data
    - c) To execute Agents
    - d) To manage API keys
    - **Correct answer: a**

    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
215. **How can you access tracing information for a Runner execution?**
    - a) Through the OpenAI Dashboard
    - b) By checking the Agent's logs
    - c) By using the Runner.trace() method
    - d) It is not accessible
    - **Correct answer: a**
216. **What happens if the Runner's** max_turns **limit is reached?**
    - a) It raises a MaxTurnsExceededError
    - b) It retries the execution
    - c) It ignores the limit
    - d) It delegates to another Agent
    - **Correct answer: a**
217. **How can you configure the Runner's retry behavior?**
    - a) By setting max_retries
    - b) By using a custom callback

- c) By defining it in the instructions
- d) By modifying the context
- **Correct answer: a**

218. **What is the role of the** tool_call_limit **in** Runner.run()**?**
    - a) To limit the number of tool calls
    - b) To set the maximum response length
    - c) To define the context size
    - d) To manage retries
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -

219. **How does the Runner handle execution timeouts?**
    - a) It raises a TimeoutError
    - b) It retries the execution
    - c) It ignores the timeout
    - d) It delegates to another Agent
    - **Correct answer: a**

220. **What happens if the Runner's** run_sync() **method raises an error?**
    - a) It propagates the error
    - b) It retries the execution
    - c) It ignores the error
    - d) It uses a default response
    - **Correct answer: a**

221. **How can you debug a Runner's execution path?**
    - a) By enabling tracing in the OpenAI Dashboard
    - b) By setting debug=True in the Runner
    - c) By using the Runner.debug() method
    - d) It is not possible
    - **Correct answer: a**

222. **What is the purpose of the** max_retries **parameter in** Runner.run()**?**
    - a) To limit the number of retries
    - b) To set the maximum response length
    - c) To define the context size
    - d) To manage tool calls
    - **Correct answer: a**

223. **How does the Runner handle async Agent execution?**
    - a) Through the run() method
    - b) By setting async=True in run_sync()
    - c) By using a custom callback
    - d) It does not support async execution
    - **Correct answer: a**

224. **What happens if the Runner's** tool_call_limit **is reached?**
    - a) It raises a ToolCallLimitExceededError
    - b) It retries the tool call
    - c) It ignores the limit
    - d) It delegates to another tool

- **Correct answer: a**

225. **How can you customize the Runner’s behavior for specific Agents?**
    - a) By passing a custom context
    - b) By setting agent_behavior
    - c) By defining it in the instructions
    - d) By modifying the model_settings
    - **Correct answer: a**
226. **What is the role of the Runner in multi-agent workflows?**
    - a) To manage Agent execution and Handoffs
    - b) To validate inputs
    - c) To execute tools
    - d) To handle Guardrails
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
227. **How does the Runner support error handling?**
    - a) Through max_retries and exception propagation
    - b) By using a custom callback
    - c) By defining it in the instructions
    - d) By modifying the context
    - **Correct answer: a**
228. **What happens if the Runner’s execution is interrupted?**
    - a) It raises an InterruptedError
    - b) It retries the execution
    - c) It ignores the interruption
    - d) It delegates to another Agent
    - **Correct answer: a**
229. **How can you ensure the Runner uses a specific timeout?**
    - a) By setting timeout in run()
    - b) By using a custom callback
    - c) By defining it in the instructions
    - d) By modifying the context
    - **Correct answer: a**
230. **What is the purpose of the Runner’s tracing in multi-agent workflows?**
    - a) To visualize and debug interactions
    - b) To manage context
    - c) To execute tools
    - d) To handle Handoffs
    - **Correct answer: a**
231. **How does the Runner handle tool execution order?**
    - a) It follows the LLM’s decision
    - b) It executes tools in parallel
    - c) It requires manual specification
    - d) It follows the tools list order
    - **Correct answer: a**
232. **What happens if the Runner’s** max_retries **limit is reached?**

- a) It raises a MaxRetriesExceededError
- b) It ignores the limit
- c) It uses a default response
- d) It delegates to another Agent
- **Correct answer: a**

233. **How can you debug a Runner's tool calls?**
    - a) By enabling tracing in the OpenAI Dashboard
    - b) By setting debug=True in the Runner
    - c) By using the Runner.debug() method
    - d) It is not possible
    - **Correct answer: a**
234. **What is the role of the** context **in Runner execution?**
    - a) To pass shared state to Agents and tools
    - b) To manage API keys
    - c) To execute tools
    - d) To handle Handoffs
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - https://x.com/samaddeveloper
    -
235. **How does the Runner handle async tool calls?**
    - a) It awaits them automatically
    - b) It ignores async tools
    - c) It raises a TypeError
    - d) It converts them to sync tools
    - **Correct answer: a**
236. **What happens if the Runner's execution exceeds the** timeout**?**
    - a) It raises a TimeoutError
    - b) It retries the execution
    - c) It ignores the timeout
    - d) It delegates to another Agent
    - **Correct answer: a**
237. **How can you ensure the Runner retries on specific errors?**
    - a) By setting retry_on_errors
    - b) By using a custom callback
    - c) By defining it in the instructions
    - d) By modifying the context
    - **Correct answer: a**
238. **What is the purpose of the** max_turns **in** Runner.run()**?**
    - a) To limit the number of Agent iterations
    - b) To set the maximum response length
    - c) To define the context size
    - d) To manage tool calls
    - **Correct answer: a**
239. **How does the Runner handle Handoff execution?**
    - a) Through the context and Handoff objects
    - b) By running Handoffs in parallel

- c) By merging Handoffs
- d) It does not support Handoffs
- **Correct answer:

Advanced OpenAI Agents SDK Quiz

Created by https://github.com/samade747

This OpenAI Agents SDK quiz is designed for advanced learners and requires substantial preparation to succeed. Before attempting this assessment, ensure you have a solid foundation in Python programming, including object-oriented concepts, async/await patterns, decorators, and error handling. You'll need to thoroughly study Pydantic models for data validation, understanding field definitions, default values, and validation behavior. Dedicate significant time to the OpenAI Agents SDK documentation (https://openai.github.io/openai-agents-python/), focusing on core concepts like Agents, Tools, Handoffs, context management, and the agent execution loop. Practice writing and analyzing code that uses the `@function_tool` decorator, `Runner.run_sync()`, agent cloning, and multi-agent orchestration patterns. Review prompt engineering techniques from the OpenAI cookbook, particularly Chain-of-Thought prompting, system message design, and handling sensitive data. **Created by https://github.com/samade747**

Chat Models and Completions

1. In the OpenAI Python SDK, which API is now recommended as the primary interface for text generation?
   A. The original Completions API
   B. The new Responses API
   C. The ChatGPT web interface

D. The legacy Completion endpoint
Correct Answer: Bgithub.com

2. When using `client.chat.completions.create(...)` in openai-python, which parameter contains the conversation messages?
   A. `input_text`
   B. `prompt`
   C. `messages`
   D. `conversation`
   Correct Answer: Cgithub.com

3. In the new Responses API (`client.responses.create`), which of the following parameters must be provided for a basic text query?
   A. `prompt` and `model`
   B. `messages` and `model`
   C. `instructions` and `input`
   D. `context` and `query`
   Correct Answer: Cgithub.com

4. What role name does the openai-python library use for the system (developer) message in chat completions examples?
   A. `"system"`
   B. `"assistant"`
   C. `"user"`
   D. `"developer"`
   Correct Answer: Dopenai.github.io

In the example code below, which attribute contains the assistant's reply text?

python
CopyEdit

```
completion = client.chat.completions.create(

    model="gpt-4o",

    messages=[

        {"role": "developer", "content": "Talk like a pirate."},

        {"role": "user", "content": "How do I check if a Python
object is an instance of a class?"},

    ],

)
```

```
print(completion.choices[0].message.content)
```

5. A. `completion.result.text`
   B. `completion.choices[0].message.content`
   C. `completion.output`
   D. `completion.message.text`
   Correct Answer: Bgithub.com

6. In chat completions for agents, what is an example of a developer/system instruction affecting style?
   A. `"Always respond in haiku form."`
   B. `"Use emotions freely in responses."`
   C. `"You are a cat."`
   D. `"Ignore previous messages."`
   Correct Answer: Aopenai.github.io

7. Which HTTP library underlies the asynchronous `AsyncOpenAI` client in the openai-python SDK?
   A. `requests`
   B. `aiohttp`
   C. `httpx`
   D. `urllib3`
   Correct Answer: Cgithub.com

8. How do you initiate an asynchronous OpenAI client in Python?
   A. `client = AsyncOpenAI()`
   B. `client = OpenAI(async=True)`
   C. `client = OpenAI(use_async=True)`
   D. `client = AsyncClient()`
   Correct Answer: Agithub.com

9. How do you enable streaming of tokens from an API call in the openai-python SDK?
   A. Set `mode="stream"`
   B. Use `client.chat.stream()` instead of `create()`
   C. Pass `stream=True` to the `create()` method
   D. Streaming is not supported
   Correct Answer: Cgithub.com

10. In the Realtime API beta of openai-python, which library is used to manage WebSocket connections?
    A. `websockets`
    B. `socketio`
    C. `asyncio-websockets`

D. `requests`
Correct Answer: Agithub.com

11. **Created by https://github.com/samade747**

Function Calling and Tools

11. Which OpenAI model versions (as of late 2024) support the function-calling feature in chat completions?
    A. `gpt-3.5-turbo-0613` and `gpt-4-0613`
    B. `gpt-3.5-turbo` and `gpt-4o`
    C. Only `gpt-4-turbo-1106`
    D. All GPT-3.5 and GPT-4 models
    Correct Answer: Amedium.com

12. When defining a function for use with function calling, what JSON keys must be included for each function definition in the `functions` list?
    A. `{"func_name", "args", "desc"}`
    B. `{"name", "parameters", "returns"}`
    C. `{"name", "description", "parameters"}`
    D. `{"id", "signature", "body"}`
    Correct Answer: Cmedium.com

13. In the OpenAI ChatCompletion API, how do you tell the model to call a specific function with a given name?
    A. Set `function_call="specific_function_name"`
    B. Use `tool="specific_function"` parameter
    C. Include `{"name": "function_name"}` in the `function_call` argument
    D. Add a special token `CALL_FUNCTION[name]` in the message content
    Correct Answer: Cmedium.com

14. If the `function_call` parameter is set to `"auto"`, what happens?
    A. The model will not call any function.
    B. The model chooses whether or not to call a function if it deems it appropriate.
    C. It forces an error since `"auto"` is not allowed.
    D. All defined functions are called in order.
    Correct Answer: Bmedium.com

15. **Created by https://github.com/samade747**
16. https://x.com/samaddeveloper
17.

18. When the model returns a function call, its arguments are provided as a JSON string under which response field?
    A. `response.arguments`

B. `response.choices[0].message.function_call.args`
C. `response.choices[0].message.function_call.arguments`
D. `response.data.function_args`
Correct Answer: Cmedium.com

19. How can you convert the function-call arguments from the model (a JSON-formatted string) into a Python object?
A. By calling `parse_args(response)`
B. By evaluating with `eval(...)`
C. Using `json.loads(response.choices[0].message.function_call.arguments)`
D. No conversion needed; it's already a Python dict.
Correct Answer: Cmedium.com

20. In the Agents SDK, what decorator is used to turn a Python function into a tool?
A. `@tool`
B. `@function_tool`
C. `@agent_tool`
D. `@pydantic.tool`
Correct Answer: Bopenai.github.io

21. **Created by https://github.com/samade747**

22. https://x.com/samaddeveloper

23.

24. By default, what name is assigned to a function tool derived from a Python function?
A. The function's module path
B. A random UUID
C. The function's own name
D. `"function_tool_n"` where n is an index
Correct Answer: Copenai.github.io

25. How does the Agents SDK generate a tool's description by default when using `@function_tool`?
A. It leaves it blank.
B. It uses the function's docstring.
C. It calls the function with dummy inputs to infer behavior.
D. It requires you to explicitly pass a description.
Correct Answer: Bopenai.github.io

26. How does the Agents SDK generate JSON schemas for a function's parameters when creating a function tool?
A. It uses the Python `inspect` module and Pydantic to infer types from the function signature.
B. It requires the user to manually supply a JSON schema.

C. It ignores type information and treats all arguments as strings.
D. It uses the OpenAPI spec to generate schemas.
Correct Answer: Aopenai.github.io

27. What library does the Agents SDK use to parse function docstrings for generating parameter schemas?
A. `docutils`
B. `inspect` only (no docstring parsing)
C. `griffe`
D. `pydoc`
Correct Answer: Copenai.github.io

28. When a function tool raises an error during execution, what is the default behavior if you do **not** specify a `failure_error_function`?
A. The error is silently ignored.
B. The agent receives a default message indicating an error occurred.
C. The entire agent run is aborted with an exception.
D. The tool call is retried automatically.
Correct Answer: Bopenai.github.io

29. If you explicitly set `failure_error_function=None` for a function tool, what happens on tool error?
A. The default error message is used anyway.
B. The tool error is raised (e.g., `ModelBehaviorError` or `UserError`).
C. The agent ignores the error and continues.
D. The agent restarts the tool call automatically.
Correct Answer: Bopenai.github.io

30. In the Agents SDK, when manually creating a `FunctionTool` object, where should you handle possible exceptions from the tool invocation?
A. In a surrounding try/except around `Runner.run()`.
B. Inside the function decorated by `@failure_handler`.
C. Inside the `on_invoke_tool` method of your `FunctionTool`.
D. The Agents SDK will catch all errors automatically.
Correct Answer: Copenai.github.io

31. Which of the following is *not* a class of tools in the Agents SDK?
A. Hosted tools (provided by OpenAI)
B. Function-call tools (Python functions)
C. Agents-as-tools (using one agent in another)
D. Plugin tools (integrates third-party plugins)
Correct Answer: Dopenai.github.io
32. **Created by https://github.com/samade747**

Streaming and Asynchronous Usage

26. How can you iterate over events from a streaming response using the synchronous client?
    A. Using `for event in client.stream()`
    B. By polling `client.chat.completions.stream()`
    C. With a context manager: `with client.stream as s:`
    D. Using `for event in stream:` where `stream = client.chat.completions.create(..., stream=True)`
    Correct Answer: D[github.com](github.com)

27. How do you perform streaming with the asynchronous client?
    A. Call `.stream()` on an OpenAI ASGI app
    B. Use `async for event in stream:` after awaiting the create call with `stream=True`
    C. Use `client.chat.completions.stream()` coroutine
    D. Asynchronous streaming is not supported
    Correct Answer: B[github.com](github.com)

28. Which of the following response types indicates a streaming token in the openai-python SDK?
    A. `response.token`
    B. `response` (no specific indicator)
    C. `response.stream`
    D. `response.choices[0].delta` events
    Correct Answer: D[github.com](github.com)
    Created by https://github.com/samade747
29. https://x.com/samaddeveloper
30.

31. In the realtime (WebSocket) API example, which message role is used to send user text?
    A. `system`
    B. `assistant`
    C. `user`
    D. `developer`
    Correct Answer: C[github.com](github.com)

32. What happens when the realtime API sends an `"error"` event?
    A. The client will automatically throw an exception and close the connection.
    B. The connection remains open, and you must handle it by checking `event.type == 'error'`.
    C. All future events are discarded.
    D. The SDK reconnects automatically after a short delay.

Correct Answer: B

33. The openai-python SDK supports streaming via SSE. What does "SSE" stand for?
    A. Secure Socket Extension
    B. Server-Side Events
    C. Stateless Streaming Exchange
    D. Simple Stream Encoding
    Correct Answer: Bgithub.com

34. Which of these is a correct way to use the asynchronous client in an `async` function?
    A. `client = AsyncOpenAI(); response = client.chat.completions.create(...)`
    B. `client = AsyncOpenAI(); response = await client.chat.completions.create(...)`
    C. `client = OpenAI(async=True)`
    D. `client = OpenAI(); response = client.chat.completions.create_async(...)`
    Correct Answer: Bgithub.com

35. How do you enable support for multiple modalities (text and audio) in a realtime session?
    A. Set `modalities=['text', 'audio']` in `session.update`
    B. Use `client.realtime.enable_audio()`
    C. It's not supported yet
    D. By default both are enabled; no action needed
    Correct Answer: A

36. Which flag controls whether the async or sync interface is used for Realtime API?
    A. `client.config.realtime_mode = "async"`
    B. The use of `AsyncOpenAI` vs `OpenAI` class determines it
    C. `client.enable_realtime_streaming(True)`
    D. A special environment variable `OPENAI_REALTIME`
    Correct Answer: Bgithub.com
37. **Created by https://github.com/samade747**

Agents, Tools, and Handoffs

35. What are the core primitives of the OpenAI Agents SDK?
    A. Agents, Actors, and Tasks
    B. Agents, Tools, and Handoffs
    C. Models, Tools, and Workflows
    D. Agents, Sessions, and Plugins
    Correct Answer: Bopenai.github.io

36. In Agents SDK, what does an `Agent` represent?
    A. A wrapper around a reinforcement learning algorithm
    B. A large language model with a name, instructions, and a set of tools
    C. A stateful conversational user interface
    D. A finite state machine for dialog
    Correct Answer: Bopenai.github.io

37. Which parameter of `Agent(...)` is used to specify the model to use?
    A. `model_name`
    B. `llm`
    C. `model`
    D. `engine`
    Correct Answer: Copenai.github.io

38. How do you specify a Python tool function for use by an agent?
    A. Pass the function's name as a string in `tools`
    B. Decorate the function with `@function_tool` and include it in the `tools` list
    C. Register the function via `agent.add_tool(...)`
    D. Tools must be hosted services; Python functions cannot be tools
    Correct Answer: Bopenai.github.io

39. What is the purpose of the `tools` parameter when creating an agent?
    A. It specifies the paths to external executables.
    B. It lists the resources the agent can fetch from the web.
    C. It provides a set of actions (functions) the agent can perform to achieve its task.
    D. It identifies which neural network layers are trainable.
    Correct Answer: Copenai.github.io

40. In the Agents SDK, what does setting `tool_choice='required'` do for an agent?
    A. It forces the agent to use at least one of its tools in each turn.
    B. It prevents the agent from using any tools.
    C. It allows the agent to choose whether to use a tool.
    D. It requires that a specific tool must be used (the first in the list).
    Correct Answer: Aopenai.github.io

41. What effect does setting `tool_choice=None` have in the Agents SDK?
    A. Equivalent to `'auto'`; model decides whether to call a tool.
    B. Equivalent to `'none'`; disallows any tool use.
    C. It causes an error.
    D. It resets to the default behavior (usually `"auto"`).
    Correct Answer: Bopenai.github.io

42. Which setting can you use to make an agent stop after its first tool call instead of continuing?
    A. `agent.tool_choice='stop_on_first_tool'`

B. `agent.stop_on_tool=True`
C. `agent.tool_use_behavior="stop_on_first_tool"`
D. `runner.stop_after_tool=True`
Correct Answer: C[openai.github.io](openai.github.io)

43. What is the `output_type` parameter of an `Agent` used for?
A. Determining the format of debug logs.
B. Specifying the Python type that the agent should output (e.g. a Pydantic model).
C. Choosing between `str` or `list` as output.
D. It has been deprecated and does nothing.
Correct Answer: B[openai.github.io](openai.github.io)

44. By default, if you do not specify `output_type` for an agent, what is the output type?
A. A Python dict
B. Plain text (`str`)
C. A Pydantic BaseModel
D. JSON object
Correct Answer: B[openai.github.io](openai.github.io)

45. How can you convert an existing agent into a new agent with a different name or instructions?
A. Use the `copy()` method of the Agent.
B. Use Python's `deepcopy(agent)`.
C. Use the `agent.clone(name=..., instructions=...)` method.
D. Reinstantiate a new Agent with the old agent's properties manually.
Correct Answer: C[openai.github.io](openai.github.io)

46. What is a *handoff* in the context of the Agents SDK?
A. A function tool that uses `on_handoff` hook.
B. A sub-agent that the current agent can delegate tasks to.
C. A mechanism for transferring context between API calls.
D. A built-in function for ending a conversation.
Correct Answer: B[openai.github.io](openai.github.io)

47. How do you provide handoffs to an agent?
A. By setting `agent.handoffs = True`
B. By including other `Agent` objects in the `handoffs` list when creating the agent
C. By using a special `HandoffTool` in the `tools` list
D. By calling `agent.add_handoff(other_agent)` after creation
Correct Answer: B[openai.github.io](openai.github.io)

48. Can the instructions for an agent be generated dynamically at runtime?
A. No, instructions must be fixed strings.
B. Yes, by passing a function that takes context and agent and returns a string.
C. Only if the model is GPT-4.

D. Only if you set `dynamic_instructions=True`.
Correct Answer: B[openai.github.io]

49. If you want to observe an agent's lifecycle events (e.g. to log intermediate steps), which class should you use?
A. `RunnerTracer`
B. `AgentLifecycleLogger`
C. `AgentHooks`
D. `TraceManager`
Correct Answer: C[openai.github.io]

50. Which of the following are **hosted tools** provided by the OpenAI SDK (as shown in the documentation)?
A. `WebSearchTool`, `FileSearchTool`, `ComputerTool`, `CodeInterpreterTool`
B. `SQLTool`, `BrowserTool`, `FileIO`, `MathSolver`
C. `WeatherTool`, `DatabaseTool`, `ShellTool`
D. `AudioTool`, `VideoTool`, `ImageTool`
Correct Answer: A[openai.github.io]

51. What is the `LocalShellTool` used for in the Agents SDK?
A. Executing shell commands on the local machine
B. Running shell commands on the OpenAI servers
C. Providing an interactive REPL for debugging
D. It's an example placeholder, not functional
Correct Answer: A[openai.github.io]
Created by https://github.com/samade747
52. https://x.com/samaddeveloper
53.
54. What does the `CodeInterpreterTool` do?
A. It interprets code in a Docker container.
B. It executes Python code in a sandboxed environment.
C. It compiles C code into assembly.
D. It provides autocompletion for coding queries.
Correct Answer: B[openai.github.io]
55. **Created by https://github.com/samade747**

Agent Execution Loop and Orchestration

53. Which class do you use to execute agents in the Agents SDK?
A. `AgentRunner`
B. `Runner`
C. `AgentExecutor`
D. `Session`

Correct Answer: Bopenai.github.io
Created by https://github.com/samade747

54. https://x.com/samaddeveloper
55.
56. What does `Runner.run_sync(agent, input)` do under the hood?
    A. It compiles the agent to WebAssembly.
    B. It launches an interactive REPL.
    C. It calls `Runner.run(agent, input)` in the current thread.
    D. It creates a new thread for running the agent.
    Correct Answer: Copenai.github.io

57. How do you run an agent with streaming output?
    A. Use `runner = Runner(streaming=True)`
    B. Call `Runner.run_streamed(agent, input)`
    C. In `Runner.run()`, set `stream_output=True`
    D. Streaming isn't supported by the runner
    Correct Answer: Bopenai.github.io

58. What is the return type of `Runner.run_streamed()`?
    A. A Python generator of events
    B. `RunResultStreaming`
    C. `AsyncRunResult`
    D. `StreamingResponse`
    Correct Answer: Bopenai.github.io

59. In the agent loop, what happens if the model produces a *final output* with no tool calls?
    A. The loop stops and returns the output.
    B. The loop restarts with the same agent.
    C. The loop throws an exception.
    D. The loop ignores the output and continues waiting.
    Correct Answer: Aopenai.github.ioopenai.github.io

60. In the agent loop description, what triggers a handoff?
    A. The agent's `tools` list includes sub-agents.
    B. The model output indicates a change of agent (often via a special token).
    C. The runner reaches the maximum number of turns.
    D. The user explicitly calls `Runner.handoff(...)`.
    Correct Answer: Bopenai.github.io

61. If an agent run exceeds the specified `max_turns`, what happens?
    A. The runner quietly stops and returns partial output.
    B. The runner raises `MaxTurnsExceeded`.
    C. The runner ignores the limit and continues.
    D. The model is switched to a cheaper model automatically.

Correct Answer: Bopenai.github.io
Created by https://github.com/samade747

62. https://x.com/samaddeveloper
63.
64. Where can you configure a global model to override individual agents' models during a run?
    A. In `ModelSettings` passed to each agent
    B. Using the `run_config` parameter of `Runner.run`
    C. By setting an environment variable `GLOBAL_MODEL`
    D. This is not supported; each agent uses its own model.
    Correct Answer: Bopenai.github.io

65. Which approach to multi-agent orchestration relies on the LLM to decide the sequence of actions?
    A. Orchestrating via code
    B. Orchestrating via configuration file
    C. Orchestrating via LLM (agent planning)
    D. Orchestrating via hard-coded script
    Correct Answer: Copenai.github.io

66. Which pattern involves your code determining the sequence of agents to run?
    A. Orchestrating via LLM
    B. Orchestrating via code
    C. Orchestrating via prompts
    D. Orchestrating via GUI
    Correct Answer: Bopenai.github.io

67. What is one advantage of orchestrating agents via code instead of relying on a single agent's planning?
    A. It allows the agent to learn from data.
    B. It makes the flow more deterministic and predictable in cost and performance.
    C. It uses less memory at runtime.
    D. It automatically improves model accuracy.
    Correct Answer: Bopenai.github.io
    Created by https://github.com/samade747
68. https://x.com/samaddeveloper
69.
70. In the "Orchestrating via code" approach, how might you decide which next agent to run?
    A. Use the model's subjective choice of agent.
    B. Chain multiple agents by feeding one's structured output into another.
    C. Run all agents in parallel.
    D. Use a genetic algorithm to pick agents.
    Correct Answer: Bopenai.github.io

71. True or False: Structured outputs (e.g. JSON) from one agent can be parsed by your code to decide the next steps in a chain.

A. True
B. False
Correct Answer: Aopenai.github.io
72. **Created by https://github.com/samade747**

Guardrails and Context Management

66. What is “context” in the Agents SDK?
 A. The portion of memory used by the model
 B. Extra data (such as user info or dependencies) passed to tools and agents, *not* sent to the LLM
 C. The current conversation history stored in a file
 D. A synonym for “environment variables”
 Correct Answer: Bopenai.github.ioopenai.github.io
 Created by https://github.com/samade747
67. https://x.com/samaddeveloper
68.
69. Which class wraps the context object provided to `Runner.run()`?
 A. `RunContext`
 B. `ContextWrapper`
 C. `RunContextWrapper`
 D. `ContextManager`
 Correct Answer: Copenai.github.io

70. If you want your agent’s tools to access shared state or dependencies, how do you provide it?
 A. Pass them as global variables.
 B. Put them in an `Agent.config` property.
 C. Pass a context object to `Runner.run(..., context=...)`.
 D. Agents cannot have external dependencies.
 Correct Answer: Copenai.github.io

71. True or False: The context object passed to `Runner.run()` is sent to the LLM as part of the prompt.
 A. True
 B. False
 Correct Answer: Bopenai.github.io

72. What must you ensure about the context type when using multiple tools or agents in a run?
 A. Each tool can have a different context type.
 B. All tools and agents must use the *same* context type T (e.g., Agent[T]) for that run.
 C. Context types must be subclasses of `BaseModel`.
 D. Context types are irrelevant as long as `context` is not None.

Correct Answer: Bopenai.github.io
Created by https://github.com/samade747

73. https://x.com/samaddeveloper
74.
75. What are **input guardrails** in the Agents SDK?
    A. Validations that check the agent's output format before returning
    B. Checks that run on the user's initial input before invoking an agent
    C. Tools that filter web search results
    D. System-level AI monitors for toxicity
    Correct Answer: Bopenai.github.io

76. What happens when an input guardrail's tripwire is triggered?
    A. The agent execution continues normally.
    B. The agent run is aborted with `InputGuardrailTripwireTriggered`.
    C. The input is sanitized and passed to the agent.
    D. The guardrail schedules a follow-up agent to handle the issue.
    Correct Answer: Bopenai.github.io

77. What are **output guardrails**?
    A. Checks that run on the final agent output before returning it to the user.
    B. Extra output channels for debugging.
    C. The last few tokens emitted by the model.
    D. A form of rate limiting.
    Correct Answer: Aopenai.github.io

78. Under what condition are output guardrails evaluated?
    A. Only if the agent is the very first agent in a handoff chain.
    B. Only if the agent is the final agent in a handoff chain.
    C. On every intermediate tool call.
    D. Always, regardless of agent order.
    Correct Answer: Bopenai.github.io

79. If a guardrail (input or output) sets `tripwire_triggered=True`, what does the SDK do?
    A. Silently ignores the failure and continues.
    B. Raises an `InputGuardrailTripwireTriggered` or `OutputGuardrailTripwireTriggered` exception accordingly.
    C. Logs a warning but returns the output anyway.
    D. Prompts the model to try again.
    Correct Answer: Bopenai.github.ioopenai.github.io

80. On which agent(s) are input guardrails executed?
    A. Every agent in the chain.
    B. Only the last agent.
    C. Only the first (starting) agent.
    D. Never; input guardrails are global.

Correct Answer: C[openai.github.io](openai.github.io)
Created by https://github.com/samade747

81. https://x.com/samaddeveloper
82.
83. Which environment variable disables logging of model inputs/outputs in the Agents SDK?
    A. `OPENAI_LOG_MODEL_INPUTS=0`
    B. `DISABLE_MODEL_LOGGING`
    C. `OPENAI_AGENTS_DONT_LOG_MODEL_DATA=1`
    D. `AGENTS_NO_LOGGING=true`
    Correct Answer: C[openai.github.io](openai.github.io)

84. Which environment variable disables logging of tool inputs/outputs in the Agents SDK?
    A. `OPENAI_AGENTS_NO_TOOL_LOG=1`
    B. `OPENAI_AGENTS_DONT_LOG_TOOL_DATA=1`
    C. `DISABLE_TOOL_LOGS=1`
    D. `AGENTS_LOGGING=None`
    Correct Answer: B[openai.github.io](openai.github.io)
85. **Created by https://github.com/samade747**

File Uploads and API Integration

79. How do you upload a training data file for fine-tuning using the OpenAI Python SDK?
    A. `client.upload_data(filepath, purpose='fine-tune')`
    B. `client.fine_tune.upload(file=...)`
    C. `client.files.create(file=open(...), purpose='fine-tune')`
    D. `client.fine_tune.create_file(...)`
    Correct Answer: C[github.com](github.com)

80. Which Python types are accepted for the `file` parameter in `client.files.create`?
    A. Only file paths (`str`)
    B. Bytes, a PathLike (e.g., `pathlib.Path`), or a `(filename, bytes, content_type)` tuple
    C. A file descriptor integer
    D. Only raw JSON strings
    Correct Answer: B[github.com](github.com)

81. If you pass a `pathlib.Path` to `client.files.create` in the asynchronous client, what happens?
    A. It will be rejected; only file objects are allowed.
    B. The SDK automatically reads the file contents asynchronously.

C. The path is converted to a string and sent.
D. An exception is thrown.
Correct Answer: Bgithub.com

82. What `purpose` string do you use when uploading a file for fine-tuning?
    A. `"classifications"`
    B. `"answers"`
    C. `"fine-tune"`
    D. `"data"`
    Correct Answer: Cgithub.com

83. How would you use the SDK to list your uploaded files?
    A. `client.files.list()` and iterate over the returned `File` objects
    B. `client.list_files()`
    C. `client.files.get()`
    D. There is no list method; use HTTP directly
    Correct Answer: Agithub.com

84. By default, how many times will the OpenAI Python library retry a transient error (such as rate limit or 500 error)?
    A. 0 times (no retries by default)
    B. 1 time
    C. 2 times
    D. 5 times
    Correct Answer: Cgithub.com
    Created by https://github.com/samade747
85. https://x.com/samaddeveloper
86.
87. How can you disable the built-in retry behavior in the OpenAI Python SDK?
    A. Set `retries=False` in the `OpenAI()` constructor.
    B. Use `client.with_options(max_retries=0)` or `OpenAI(max_retries=0)`.
    C. Delete the `.cache` file created by the SDK.
    D. It cannot be changed.
    Correct Answer: Bgithub.com

88. What is the default timeout for requests in openai-python?
    A. 30 seconds
    B. 60 seconds
    C. 10 minutes
    D. No timeout (wait indefinitely)
    Correct Answer: Cgithub.com

89. Which exception class does the library raise if a request times out?
    A. `openai.APIConnectionError`
    B. `openai.APITimeoutError`

C. `openai.APIStatusError`
D. `TimeoutError` (the built-in)
Correct Answer: Bgithub.com

90. Which of the following errors would cause an automatic retry by default?
A. HTTP 400 Bad Request
B. Network connectivity issues (e.g. DNS failure)
C. HTTP 200 OK with error in body
D. KeyboardInterrupt during request
Correct Answer: Bgithub.com

91. In the OpenAI Python SDK, which exception is raised for HTTP status code 401?
A. `openai.AuthenticationError`
B. `openai.PermissionDeniedError`
C. `openai.NotFoundError`
D. `openai.APIConnectionError`
Correct Answer: Agithub.com

92. Which exception is raised for HTTP status code 404?
A. `openai.BadRequestError`
B. `openai.AuthenticationError`
C. `openai.NotFoundError`
D. `openai.PermissionDeniedError`
Correct Answer: Cgithub.com
Created by https://github.com/samade747
93. https://x.com/samaddeveloper
94.
95. For HTTP status 429 (Too Many Requests), which exception class is used?
A. `openai.RateLimitError`
B. `openai.RetryError`
C. `openai.TrafficError`
D. `openai.ThrottleError`
Correct Answer: Agithub.com

96. True or False: All exceptions in the openai-python SDK inherit from `openai.APIError`.
A. True
B. False
Correct Answer: Agithub.com

97. Which of the following HTTP status codes is automatically retried by default in the openai-python client?
A. 200
B. 409
C. 418

D. 301
Correct Answer: B[github.com]

98. How do you configure the default OpenAI API key programmatically if you can't set the environment variable?
A. `openai.api_key = "sk-..."`
B. `set_default_openai_key("sk-...")` from the Agents SDK
C. `client.api_key = "sk-..."` after creating a client
D. This is not possible; you must use an environment variable
Correct Answer: B[openai.github.io]

99. How can you switch the Agents SDK to use the Chat Completions API instead of the default (Responses API)?
A. `set_default_openai_api("chat_completions")`
B. `agent.use_chat_api = True`
C. `Runner.chat_mode(True)`
D. You cannot change it; the SDK always uses Chat API
Correct Answer: A[openai.github.io]
Created by https://github.com/samade747

100. https://x.com/samaddeveloper

101.

102. What function from the Agents SDK would you use to supply a custom `AsyncOpenAI` client instance?
A. `set_openai_client(client)`
B. `set_default_openai_client(custom_client)`
C. `Agents.set_client(custom_client)`
D. `openai.client = custom_client`
Correct Answer: B[openai.github.io]

103. How can you disable the built-in tracing in the Agents SDK entirely?
A. Set `tracing=False` when creating the agent.
B. Call `set_tracing_disabled(True)`.
C. Remove the OpenAI API key.
D. There is no way to disable tracing.
Correct Answer: B[openai.github.io]

104. Which function enables verbose logging to stdout in the Agents SDK?
A. `enable_verbose_stdout_logging()`
B. `logging.enable_verbose()`
C. `set_debug_mode(True)`
D. `openai.agents.enable_logging()`
Correct Answer: A[openai.github.io]

105. The Agents SDK uses Python's `logging` module. What environment variable can you set to see info-level logs from the openai-python library?

A. `OPENAI_DEBUG=1`
B. `LOGLEVEL=info`
C. `OPENAI_LOG=info`
D. `AGENTS_LOG=debug`
Correct Answer: Cgithub.com

106. Which class do you use to configure HTTP proxies or transports for openai-python?
A. `RequestsSession`
B. `DefaultHttpxClient`
C. `HttpClientTransport`
D. `ProxyConnector`
Correct Answer: Bgithub.com

107. What happens to the HTTP client's connections when the OpenAI client object is garbage collected?
A. They remain open forever (memory leak).
B. They are automatically closed.
C. They are handed off to the next client instance.
D. The SDK does not manage connections at all.
Correct Answer: Bgithub.com

108. How can you ensure that the HTTP connections used by the OpenAI client are closed promptly after use?
A. Manually delete the client object.
B. Use the client as a context manager: `with OpenAI() as client:`.
C. Call `client.close()` explicitly.
D. Both B and C are correct.
Correct Answer: Dgithub.com

109. Which class should you use to interact with Azure OpenAI endpoints in the openai-python library?
A. `AzureClient`
B. `AzureOpenAI`
C. `OpenAI` (with a special flag)
D. `AzureGPT`
Correct Answer: Bgithub.com

110. **Created by https://github.com/samade747**

Prompt Engineering and System Message Control

104. What does "Chain-of-Thought" (CoT) prompting encourage the model to do?
A. Break down its reasoning into explicit step-by-step plans or thoughts
B. Use attention weights more efficiently

C. Ignore user instructions if they conflict
D. Return shorter, concise answers
Correct Answer: Acookbook.openai.com

105. According to the OpenAI Cookbook, how can you increase a model's pass rate on complex tasks?
A. Use a larger model.
B. Induce the model to "think out loud" with detailed planning instructions.
C. Use few-shot examples only.
D. Disable temperature randomness.
Correct Answer: Bcookbook.openai.com

106. In the example "You are a helpful assistant. Provide the steps…" prompt, why is it suggested that the model can "think step by step" before and after each action?
A. To generate longer stories.
B. To improve reasoning by simulating the model's internal thought process.
C. To avoid repeating answers.
D. To test the model's ability to loop.
Correct Answer: Bcookbook.openai.com

107. In chat completions, what role is typically used for high-level instructions (sometimes called "system messages")?
A. `"user"`
B. `"assistant"`
C. `"developer"` (or `"system"`)
D. `"operator"`
Correct Answer: Copenai.github.io

108. True or False: The system/developer message can significantly change the style and format of the model's responses.
A. True
B. False
Correct Answer: Aopenai.github.io

109. What is a reason to carefully design the system message in a prompt?
A. It has no effect on the output.
B. It can help enforce constraints (like style or safety) on the model's output.
C. It only matters for speech recognition tasks.
D. It is only relevant for GPT-2, not GPT-3/4.
Correct Answer: Bopenai.github.io

110. Which of the following best practices relates to handling sensitive user data in prompts?
A. Include all user PII in system messages for context.
B. Avoid including sensitive personal information in prompts, or mask it if necessary.
C. Always pass raw user chats as system messages for debugging.
D. Send user passwords to the model to let it fetch user data.

Correct Answer: Bopenai.github.io

111. Which technique can mitigate privacy concerns when logging agent activity?
A. Logging everything in plaintext for audit.
B. Setting `OPENAI_AGENTS_DONT_LOG_MODEL_DATA=1` to avoid logging inputs/outputs.
C. Storing logs in an unprotected S3 bucket.
D. Disabling all logging and tracing.
Correct Answer: Bopenai.github.io

112. What is the purpose of using phrases like “think step by step” or “You must plan extensively” in prompts?
A. To debug the model.
B. To explicitly instruct the model to use chain-of-thought reasoning.
C. To reduce token usage.
D. To confuse the model.
Correct Answer: Bcookbook.openai.com

113. In the provided developer prompt example (SWE-bench), which of these is *not* part of the instructions for how the model should behave?
A. Solve the problem autonomously before responding.
B. Check edge cases and test thoroughly.
C. Only use internet resources, ignoring the testbed folder.
D. Think step by step and reflect on each action.
Correct Answer: Ccookbook.openai.com

114. What is the effect of including “NEVER end your turn without having solved the problem” in the system prompt?
A. It forces the model into an infinite loop.
B. It encourages the model to keep working until the fix is correct, promoting thoroughness.
C. It resets the model’s context.
D. It causes the model to only output “Done” and stop.
Correct Answer: Bcookbook.openai.com

115. In prompt engineering, why might you include the steps “Find the exact error message text,” “Identify which file caused the error,” etc., as part of the instructions?
A. To have the model ignore these steps.
B. To force the model to explicitly follow a debugging process.
C. To test if the model can copy instructions correctly.
D. To ensure the user writes these steps manually.
Correct Answer: Bcookbook.openai.com

116. How does asking the agent to “Test your code rigorously after each change” influence its behavior?
A. It tells the model to skip writing tests.
B. It encourages it to use testing tools (if available) and verify fixes iteratively.

C. It causes the model to hallucinate test data.
D. It prevents the model from using tools.
Correct Answer: Bcookbook.openai.com

117. What is meant by “structured outputs” in the context of the Agents SDK?
A. Binary output only.
B. Outputs that follow a defined schema (e.g., JSON matching a Pydantic model).
C. Plain text outputs only.
D. Encrypted responses.
Correct Answer: Bopenai.github.ioopenai.github.io

118. If an agent is configured with an `output_type`, what format does it use for its response?
A. Markdown text
B. JSON (or Pydantic-generated) format per that output type
C. Plain `str` regardless
D. It alternates between JSON and text randomly
Correct Answer: Bopenai.github.io

119. In chain-of-thought prompting, why do we often instruct the model to “plan extensively” or “write out your reasoning”?
A. To consume more tokens.
B. Because LLMs naturally hide their reasoning.
C. Because GPT-4.1 does not generate an internal chain of thought unless prompted to do so.
D. To prevent any use of tools.
Correct Answer: Ccookbook.openai.com

120. How might you handle a prompt that includes confidential data you cannot reveal, while still getting the agent to perform a task?
A. Remove or mask sensitive fields and see if the agent can still reason.
B. Send the confidential data anyway; trust the model.
C. Use a very high temperature to confuse the model.
D. Only use numeric data in prompts.
Correct Answer: Aopenai.github.io

121. What does setting the environment variable `OPENAI_AGENTS_DONT_LOG_TOOL_DATA=1` do?
A. Disables all logging in the Agents SDK.
B. Prevents logging the inputs and outputs of tools specifically.
C. Prevents tool usage by agents.
D. Converts all logs to JSON format.
Correct Answer: Bopenai.github.io

122. True or False: A carefully designed system (developer) message is an example of prompt engineering.
A. True

B. False
Correct Answer: Aopenai.github.iocookbook.openai.com

123. What technique involves the model executing a step-by-step plan before taking an action?
A. Zero-shot learning
B. Few-shot prompting
C. Chain-of-Thought prompting
D. ReAct prompting
Correct Answer: Ccookbook.openai.com

124. If you want the model to provide reasoning before answering, which phrase might you include in the user or system message?
A. “Answer quickly:”
B. “Think step by step:”
C. “Do not use your brain:”
D. “Focus on the result only:”
Correct Answer: Bcookbook.openai.com

125. Why is it recommended to “invest in good prompts” when using an agent?
A. Prompts have no impact on the agent’s behavior.
B. Good prompts clarify available tools, constraints, and problem context, improving performance.
C. So that the model uses fewer tokens.
D. Because agents ignore instructions otherwise.
Correct Answer: Bopenai.github.io

126. What is a possible downside of having a single agent responsible for all tasks without specialized sub-agents?
A. Increased security
B. The agent might be too general and not excel at specialized tasks.
C. It always produces more accurate results.
D. It is easier to control.
Correct Answer: Bopenai.github.io

127. Which of the following is an example of prompt engineering to control model behavior?
A. Using a Python debugger during inference.
B. Including system messages that specify the tone, style, or step-by-step instructions.
C. Changing the model’s source code.
D. Running the model on a faster GPU.
Correct Answer: Bcookbook.openai.com

128. In the context of Agents, what is a “tool choice” policy?
A. It’s how you name your function tools.
B. It’s a setting that influences when or whether the model uses tools (`auto`,

`required`, `none`, or a specific tool).
C. It's the order in which tools are called.
D. It determines which model to use for which tool.
Correct Answer: Bopenai.github.io

129. Why might you use an "output guardrail" on an agent?
A. To validate the agent's final output (e.g. check for profanity or format) before delivering it.
B. To speed up tool calls.
C. To ensure the agent uses at least one tool.
D. To log intermediate LLM tokens.
Correct Answer: Aopenai.github.io

130. What is the purpose of running guardrails in **parallel** to the agent?
A. To improve agent response times.
B. To perform checks (on input or output) concurrently, so that the main agent can be aborted if needed without waiting.
C. To visualize agent progress.
D. To allow multiple models to run at once.
Correct Answer: Bopenai.github.io

131. If you want an agent to stop and return output as soon as it calls a tool, which setting do you use?
A. `agent.force_tool_use=True`
B. `agent.stop_on_tool_call=True`
C. `Agent.tool_use_behavior="stop_on_first_tool"`
D. This is not configurable.
Correct Answer: Copenai.github.io

132. In prompt engineering, what is a "tripwire"?
A. A hidden test for the model.
B. A flag in guardrail results indicating a failure condition (like prohibited content).
C. The mechanism that starts the agent loop.
D. A name for a prompt suffix.
Correct Answer: Bopenai.github.io

133. What environment variable would you set to turn off tracing logs if they are too verbose in the Agents SDK?
A. `OPENAI_TRACE=off`
B. `AGENTS_TRACE=false`
C. `OPENAI_AGENTS_DONT_LOG_MODEL_DATA=0`
D. `set_tracing_disabled(True)` (call function instead)
Correct Answer: Dopenai.github.io

134. What is *not* a recommended practice when prompt engineering with sensitive user data?

A. Sanitizing or anonymizing PII before including in prompts.
B. Using guardrails to screen for privacy leaks.
C. Logging full user data in plaintext for debugging.
D. Asking the model to avoid recalling private info.
Correct Answer: Copenai.github.io

135. Which of these is a sign of good prompt engineering for agents?
A. A prompt that yields consistent, interpretable agent behavior using available tools and knowledge.
B. A prompt that the model often misunderstands.
C. Long, irrelevant context that confuses the model.
D. Relying on random chance that the model does the right thing.
Correct Answer: Aopenai.github.io

136. The Agents SDK suggests reviewing which technique from the OpenAI Cookbook for improving reasoning?
A. Few-shot learning
B. Chain-of-Thought prompting
C. Knowledge distillation
D. Dropout regularization
Correct Answer: Bcookbook.openai.com

137. What role do system messages play in open-domain conversational applications?
A. They are ignored by modern models.
B. They are equivalent to user inputs.
C. They provide high-level instructions and guidelines to the model before conversation.
D. They only define the termination condition.
Correct Answer: Copenai.github.io

138. In an agent prompt, why might you list a numbered plan or checklist for the model to follow?
A. To confuse the model.
B. To punish the model if it fails.
C. To give explicit workflow instructions and improve reliability.
D. It has no effect unless you also give code.
Correct Answer: Ccookbook.openai.com

139. When chaining agents via code, how can you use structured outputs to determine the next step?
A. By having the model always return a numeric flag.
B. By parsing the structured (JSON) output and using logic to choose the next agent or action.
C. By training a neural network on outputs.
D. By random selection among agents.
Correct Answer: Bopenai.github.io

140. Why is it often better to have specialized agents rather than one super-general agent?
A. Specialized agents are cheaper.
B. Specialized agents can be fine-tuned for narrow tasks, improving accuracy and efficiency.
C. A single agent cannot call tools.
D. It doesn't matter; performance is the same.
Correct Answer: Bopenai.github.io

141. What is a recommended way to debug or monitor an agentic flow?
A. Set `debug=True` in Runner.
B. Use the built-in tracing and visualization tools provided by the SDK.
C. Print raw LLM tokens to console.
D. Disable all safety checks.
Correct Answer: Bopenai.github.io

142. How would you add an API key to the environment for use by the Python SDK?
A. `export OPENAI_API_KEY="sk-..."`
B. `os.setenv("API_KEY")` in code
C. Include it in a `.netrc` file
D. Use only the key from the UI (no environment variable)
Correct Answer: Agithub.com

143. What is the purpose of `Runner.run(..., stream=True)`?
A. To enable streaming in `Runner` (alias for `run_streamed`).
B. It's an invalid parameter (stream not supported by Runner).
C. To get real-time audio.
D. To stream debug logs.
Correct Answer: Aopenai.github.io

144. If you want the agent to treat the first user message as both context and initial query, how should you provide the input to `Runner.run()`?
A. As a list of `ChatItem`s including system and user messages
B. As a raw string (the user message)
C. By writing to a file and passing the file path
D. You must always provide a `ToolCall` list
Correct Answer: Bopenai.github.io

145. In multi-agent orchestration, what does "allowing the LLM to make decisions" typically involve?
A. Giving the LLM control of tool usage and handoffs via its reasoning in the prompt.
B. Having the user manually choose the next agent.
C. Using a rule-based system separate from the LLM.
D. Using only open-loop prompts with no model guidance.
Correct Answer: Aopenai.github.io

146. **Created by https://github.com/samade747**

147. In Agents SDK, what method would you call to continue an agent loop after a tool call and its result?
A. `Runner.continue()`
B. `Runner.run()` again with updated input
C. No explicit call needed; the Runner loop handles it automatically.
D. `agent.next_turn()`
Correct Answer: Copenai.github.io

148. What indicates that an agent has produced a "final output" according to the runner?
A. The agent explicitly calls `agent.finish()`.
B. The model outputs a special token `<END>`.
C. The output type matches and there are no pending tool calls.
D. The user says "thank you".
Correct Answer: Copenai.github.io

149. How do you instruct the runner to use a different model than each agent's own model?
A. Set `agent.model` before running.
B. Use the `run_config` parameter's `model` field in `Runner.run()`.
C. Change the environment variable `MODEL_OVERRIDE`.
D. It's not supported to override per-run.
Correct Answer: Bopenai.github.io

150. In the Agents SDK, what is the significance of `RunResult.final_output`?
A. It contains the tool outputs from the last turn.
B. It's the final answer produced by the agent.
C. It's a boolean indicating success.
D. It logs the conversation history.
Correct Answer: Bopenai.github.io

151. What does `Runner.run()` return?
A. `None`
B. `RunResult` (with final output, history, etc.)
C. A generator of `RunResult` objects
D. Just the final string output
Correct Answer: Bopenai.github.io

152. Why might you want to override the default `tool_choice` back to `"auto"` after a tool call?
A. To prevent the model from calling tools repeatedly in a loop.
B. To force it to always use the same tool.
C. To save the conversation history.
D. The SDK does this automatically; you never override it manually.

Correct Answer: A[openai.github.io]

153. What does the `RunContextWrapper` allow you to do in a tool function?
A. Modify the system prompt on the fly.
B. Access the shared context object (`wrapper.context`) and other run metadata.
C. Send additional messages to the model.
D. Pause and resume the agent.
Correct Answer: B[openai.github.io]

154. If an agent's instructions are a function, which arguments does it receive?
A. Only the user input.
B. The Agent instance and the RunContextWrapper.
C. The model's hidden state.
D. Nothing; it's a static string.
Correct Answer: B[openai.github.io]

155. What does setting `model_settings.tool_choice='none'` do?
A. Requires the model to use all tools.
B. Disallows any tool usage (agent must answer without tools).
C. Resets to default tool choice.
D. Causes an exception.
Correct Answer: B[openai.github.io]

156. In the function tools documentation, what is `custom_output_extractor` used for?
A. To modify the LLM's decoder stack.
B. To post-process an agent-as-tool's output.
C. To change how a function tool's output is extracted from an agent tool.
D. It's a placeholder for future use.
Correct Answer: C[openai.github.io]

157. Which of the following is *not* an example of a hosted tool listed in the documentation?
A. `WebSearchTool()`
B. `FileSearchTool()`
C. `SQLTool()`
D. `ImageGenerationTool()`
Correct Answer: C[openai.github.io]

158. True or False: Agents in the Agents SDK can have any Python object as their context.
A. True
B. False
Correct Answer: A[openai.github.io]

159. How do you declare the context type for an agent in Python?

A. By setting `agent.context_type` property.

B. Using generics: e.g., `agent = Agent[UserContext](...)`.

C. It's automatically inferred at runtime.

D. By subclassing `ContextType`.

Correct Answer: Bopenai.github.io

160. What is the role of the `UserContext` example in the context management section?

A. It shows a Pydantic model for outputs.

B. It defines a custom context type with user information.

C. It registers a new tool.

D. It's used to demonstrate error handling.

Correct Answer: Bopenai.github.io

161. In the example `fetch_user_age(wrapper: RunContextWrapper[UserInfo])`, what is `wrapper.context.name`?

A. The variable name "name" from the code.

B. A random user name generated.

C. The `name` attribute from the `UserInfo` context passed to the run.

D. The agent's name.

Correct Answer: Copenai.github.io

162. How does an input guardrail produce its result?

A. It directly stops the runner without returning a result object.

B. By returning an `InputGuardrailResult` object wrapping `GuardrailFunctionOutput`.

C. By writing to a special log file.

D. Guardrails do not return values.

Correct Answer: Bopenai.github.io

163. What should a guardrail function return?

A. A string error message or `None`

B. A boolean indicating pass/fail

C. A `GuardrailFunctionOutput` instance

D. A modified user input

Correct Answer: Copenai.github.io

164. If no tools are used in an agent run, how does the runner determine the output?

A. It will always raise an exception.

B. It uses the LLM output text as final output (converted to `str`).

C. It looks for a special `final_output` key.

D. It returns an empty string by default.

Correct Answer: Bopenai.github.ioopenai.github.io

165. Which of the following is **not** a valid value for `ModelSettings.tool_choice` in the Agents SDK?
A. `'auto'`
B. `'required'`
C. `'none'`
D. `'all'`
Correct Answer: Dopenai.github.io

166. **Created by https://github.com/samade747**

167. Which built-in exception indicates a guardrail tripwire was triggered?
A. `GuardrailTripwireError`
B. `TripwireActivated`
C. `InputGuardrailTripwireTriggered` or `OutputGuardrailTripwireTriggered`
D. `GuardrailFailureException`
Correct Answer: Copenai.github.ioopenai.github.io

168. Where do you attach guardrail functions in the Agents SDK?
A. To the `Runner` by passing a `guardrails` list.
B. As part of each `Tool` definition.
C. In the agent's `guardrails` parameter (list of functions).
D. In a special `agents.guardrails` global config.
Correct Answer: Copenai.github.ioopenai.github.io

169. How do you create a tool that retrieves data from an OpenAI vector store?
A. `FileSearchTool(vector_store_ids=[...])`
B. `agent.add_tool("FileSearch", vector_store_ids=[...])`
C. Using the `@vector_tool` decorator
D. `FileSearchTool(max_results=10)` (default store)
Correct Answer: Aopenai.github.io

170. What is the difference between "local context" and "LLM context"?
A. Local context runs on your code (not seen by LLM); LLM context is what the model sees.
B. They are the same thing with different names.
C. Local context is for debugging; LLM context is for inference.
D. Local context is a built-in Python variable.
Correct Answer: Aopenai.github.io

171. What must you include when running multiple agents that share the same context?
A. Nothing special; context is automatically global.
B. The `context` parameter in `Runner.run()` so all agents receive the same `RunContextWrapper`.
C. A special "context manager" agent.

D. You cannot share context between agents.
Correct Answer: B[openai.github.io]

172. **Created by [https://github.com/samade747]**

173. True or False: You can use any Python object (e.g., dataclass or Pydantic model) as the context in `Runner.run`.
A. True
B. False
Correct Answer: A[openai.github.io]

174. Which class wraps context and provides access to both context and the agent?
A. `RunContext`
B. `AgentContext`
C. `RunContextWrapper`
D. `ContextManager`
Correct Answer: C[openai.github.io]

175. How do you define a Pydantic model for the agent to output structured results?
A. Pass `output_type=MyModel` where `MyModel` is a subclass of `BaseModel`.
B. Set `structured_output=True`.
C. Use `agent_type='pydantic'`.
D. Agents cannot output Pydantic models.
Correct Answer: A[openai.github.io]

176. What happens internally when you set `output_type=CalendarEvent` (a Pydantic model)?
A. The model uses Chain-of-Thought to fill the model.
B. The agent uses JSON-mode to produce output matching the schema of `CalendarEvent`.
C. The agent ignores tools to focus on structure.
D. The response is converted to an image.
Correct Answer: B[openai.github.io]

177. How does specifying an `output_type` affect the OpenAI API call for the agent?
A. It sets a special LLM mode.
B. It tells the API to use the structured (JSON) response format.
C. It only affects post-processing in Python.
D. It disables token streaming.
Correct Answer: B[openai.github.io]

178. Which of the following can **not** be used as an `output_type` for an agent?
A. A Pydantic `BaseModel` class
B. A `dataclass` with typing annotations
C. A `TypedDict` type
D. A raw `socket` object

Correct Answer: Dopenai.github.io

179. When defining a function tool with Pydantic schemas, which method is used to get the JSON schema of the Pydantic model?
A. `model.to_schema()`
B. `model.get_json_schema()`
C. `model.schema()`
D. `model.json()`
Correct Answer: Cmedium.com

180. In the function tool example, what does `StepByStepAIResponse.schema()` return?
A. A string representation of the model name
B. A dict matching the JSON schema of the Pydantic model
C. The final answer as text
D. A list of validation errors
Correct Answer: Bmedium.com

181. True or False: The Agents SDK allows using TypedDicts or dataclasses as output types as long as they can be adapted by Pydantic.
A. True
B. False
Correct Answer: Aopenai.github.io

182. In guardrail implementation, what is a “tripwire” conceptually?
A. A hook that launches a new agent
B. A signal (boolean flag) in the result indicating a guardrail failure
C. A log entry for debugging
D. A data structure for storing user queries
Correct Answer: Bopenai.github.io

183. What exception will be raised if an output guardrail’s `tripwire_triggered` is true?
A. `OutputGuardrailTripwireTriggered`
B. `AgentExit`
C. `ValueError`
D. No exception; it just returns a special value
Correct Answer: Aopenai.github.io

184. Which agent hook would you override to log each tool invocation?
A. `on_tool_start` in a subclass of `AgentHooks`
B. `on_tool_end` in the runner
C. `log_tool_call` in the client
D. There is no hook for tool calls
Correct Answer: A (by subclassing `AgentHooks`)openai.github.io

185. If an agent is using GPT-4.1 and you want it to explicitly plan and reflect between tools, how should you prompt it?
A. Let it call tools silently without commentary.
B. Include instructions like "think step by step and reflect after each action."
C. Turn off temperature.
D. Do not provide instructions; GPT-4.1 reasons by default.
Correct Answer: Bcookbook.openai.com

186. What is the relationship between structured outputs and the OpenAI "JSON API"?
A. They are unrelated.
B. Structured outputs means the model uses the newer JSON API for responses.
C. Structured outputs disable the JSON API.
D. JSON API is deprecated for structured outputs.
Correct Answer: Bopenai.github.io

187. In multi-agent settings, what is the role of an "agent-as-tool"?
A. It allows one agent to call another agent's logic as if it were a function tool.
B. It's the default way agents communicate.
C. It compiles multiple agents into one.
D. There is no such concept; all agents act separately.
Correct Answer: Aopenai.github.io

188. What does the `HostedMCPTool` do?
A. It hosts a web page for visualizing agents.
B. It provides access to tools exposed by a remote Model Context Protocol server.
C. It syncs tools across multiple agents.
D. It converts hosted tools into local ones.
Correct Answer: Bopenai.github.io

189. Which statement about agent handoffs is true?
A. A handoff is triggered automatically whenever an agent uses a tool.
B. Handoffs allow an agent to delegate to specialized sub-agents based on instructions.
C. Handoffs are only for managing context, not tasks.
D. Once an agent hands off, it can never resume.
Correct Answer: Bopenai.github.io

190. **Created by https://github.com/samade747**

191. How do you define an agent that uses another agent as a tool?
A. Include the other agent in the `tools` list of the first agent.
B. Set `agent.use_agent_tool = True`.
C. Only handoffs exist; "agent-as-tool" is unsupported.
D. Wrap the other agent in a `FunctionTool`.
Correct Answer: Aopenai.github.io

192. Which method on a `Agent` object could you call to duplicate it with slight modifications?

A. `agent.copy()`
B. `agent.clone(...)`
C. `agent.replicate()`
D. `Agent.copy(agent)`
Correct Answer: Bopenai.github.io

193. When designing prompts for agents, why is it suggested to allow the agent to "critique itself" or run in a loop?
A. Because LLMs always make mistakes on first try.
B. To implement Chain-of-Thought with reflection and iterative improvement.
C. To generate more content volume.
D. It's not suggested; agents should be single-pass.
Correct Answer: Bopenai.github.io

194. What does the term "structured outputs" typically imply in the context of an LLM response?
A. The response is the same as a normal string.
B. The response is formatted (e.g. JSON) so it can be parsed into objects.
C. The response is in binary format.
D. The response is shorter.
Correct Answer: Bopenai.github.io

195. In prompt engineering, what is the advantage of explicitly listing steps (1, 2, 3, ...) in the instructions?
A. The model ignores numbered lists.
B. It helps guide the model through a specific workflow or reasoning path.
C. It forces the model to terminate the response early.
D. None; it just wastes tokens.
Correct Answer: Bcookbook.openai.com

196. What is a common reason to use a Pydantic model as an agent's output type?
A. To enforce a schema and automatically validate the response format.
B. To increase the randomness of the model output.
C. To make the agent output shorter.
D. Pydantic models cannot be used for agent output.
Correct Answer: Aopenai.github.io

197. What is implied by using the Agents SDK "Python-first" approach as mentioned in the docs?
A. Agents are written only in plain Python scripts.
B. You can orchestrate agents using Python constructs (like loops and conditionals) rather than a new DSL.
C. You must use Python 2.7.
D. The SDK is exclusively for writing Python-aware LLMs.
Correct Answer: Bopenai.github.io

198. Why are guardrails run *in parallel* to the agent?
A. To make the agent run in multiple threads.
B. So that a fast check can abort the run before wasting time/money on expensive models if the input/output is invalid.
C. To confuse the LLM.
D. They actually run sequentially after the agent finishes.
Correct Answer: Bopenai.github.io

199. When creating an agent with `Agent(name="Assistant", instructions="...", model="...")`, what is the default behavior if no tools are provided?
A. The agent will still try to answer using only the LLM.
B. The agent will throw an error.
C. The agent will not respond.
D. The agent will automatically add the search and code tools.
Correct Answer: Aopenai.github.io

200. What is the effect of the `max_num_results` parameter in `FileSearchTool`?
A. Limits the number of search results returned.
B. Sets the maximum size of file to search.
C. It's not a valid parameter (typo).
D. Determines how many files can be uploaded.
Correct Answer: Aopenai.github.io

201. In a multi-agent app, what would you use to evaluate and provide feedback to an agent until it meets criteria?
A. Chain-of-Thought
B. An evaluator agent that runs in a loop with the original agent
C. A higher temperature
D. A single-shot agent only
Correct Answer: Bopenai.github.io

202. What OpenAI resource is recommended for improving prompt design and agent reasoning?
A. Their mobile app
B. The OpenAI Cookbook and Evals framework
C. Unofficial Reddit threads only
D. The Agents SDK has built-in automatic optimization
Correct Answer: Bopenai.github.iocookbook.openai.com

203. Which of the following is an example of a valid tool-handling strategy in prompt engineering?
A. Listing tools in the prompt and asking the agent to choose the appropriate one by name.
B. Hiding tool usage from the agent entirely.
C. Letting the agent infer tool usage without any instructions.
D. Forcing the agent to call all tools in sequence.

Correct Answer: Aopenai.github.io

204. What should you do if you see an error event with `event.error.code` while using the realtime API?
A. Immediately restart the entire application.
B. Handle it in your event loop (e.g., log it, inform the user) since the connection stays open.
C. Ignore it; the API will auto-correct.
D. Close and reopen the connection automatically in your code.
Correct Answer: B

205. **Created by https://github.com/samade747**

206. Thanks for Reading it
https://x.com/samaddeveloper

Open ai sdk ended with 450 approx Questions

Python OOP Advanced Quiz: 200 Questions

Below is a set of 200 advanced-level multiple-choice questions designed to test mastery of Object-Oriented Programming in Python. Each question includes four options and the correct answer.

Classes and Objects (40 Questions)

1. **What is the purpose of the** __init__ **method in a Python class?**
   - a) To define class attributes
   - b) To initialize instance attributes
   - c) To create a static method
   - d) To define a class method
   - **Correct answer: b**
2. **What happens if a class defines** __slots__**?**
   - a) It prevents dynamic attribute creation
   - b) It enables multiple inheritance
   - c) It defines a static method
   - d) It creates a metaclass
   - **Correct answer: a**
3. **How does Python handle instance attribute lookup?**
   - a) It checks the class first, then the instance
   - b) It checks the instance first, then the class
   - c) It checks the metaclass first

    - d) It raises an AttributeError
    - **Correct answer: b**
4. **What is the effect of using** __slots__ = ('x', 'y') **in a class?**
    - a) It restricts instances to only x and y attributes
    - b) It defines class-level attributes
    - c) It enables dynamic method creation
    - d) It prevents inheritance
    - **Correct answer: a**
5. **What is the purpose of the** __new__ **method?**
    - a) To initialize instance attributes
    - b) To create a new instance before __init__
    - c) To define a static method
    - d) To manage class inheritance
    - **Correct answer: b**
6. **What happens if** __init__ **returns a value other than** None**?**
    - a) It is ignored
    - b) It raises a TypeError
    - c) It replaces the instance
    - d) It modifies the class
    - **Correct answer: b**
7. **How can you define a class attribute?**
    - a) Inside the __init__ method
    - b) Directly in the class body
    - c) Using the @property decorator
    - d) In the __new__ method
    - **Correct answer: b**
8. **What is the role of the** self **parameter in instance methods?**
    - a) It refers to the class
    - b) It refers to the instance
    - c) It refers to the metaclass
    - d) It is optional
    - **Correct answer: b**
9. **What is the output of** type(obj).__name__ **for an instance** obj **of class** MyClass**?**
    - a) obj
    - b) MyClass
    - c) type
    - d) instance
    - **Correct answer: b**
10. **How does Python handle method overriding in a class?**
    - a) It prevents overriding by default
    - b) It uses the method defined in the instance’s class
    - c) It raises a NameError
    - d) It calls the parent class method
    - **Correct answer: b**
11. **What is the purpose of the** __del__ **method?**
    - a) To define a destructor
    - b) To initialize the instance
    - c) To create a new instance

- d) To manage class attributes
- **Correct answer: a**

12. **Created by https://github.com/samade747**

13. Thanks for Reading it
https://x.com/samaddeveloper
14.
15. **What happens if a class attribute shadows an instance attribute?**
    - a) The instance attribute takes precedence
    - b) The class attribute takes precedence
    - c) It raises an AttributeError
    - d) It merges both attributes
    - **Correct answer: a**
16. **How can you access a class attribute from an instance method?**
    - a) Using self.__class__.attribute
    - b) Using self.attribute
    - c) Using type(self).attribute
    - d) Both a and c
    - **Correct answer: d**
17. **What is the effect of defining __slots__ in a base class?**
    - a) It applies only to the base class
    - b) It applies to all subclasses
    - c) It prevents subclassing
    - d) It raises a TypeError
    - **Correct answer: a**
18. **How does Python handle __new__ in a subclass?**
    - a) It ignores the subclass's __new__
    - b) It calls the subclass's __new__ first
    - c) It calls the parent's __new__ first
    - d) It raises a TypeError
    - **Correct answer: b**
19. **What is the purpose of the __getattribute__ method?**
    - a) To customize attribute access
    - b) To define a static method
    - c) To initialize attributes
    - d) To manage inheritance
    - **Correct answer: a**
20. **What happens if __getattribute__ raises an AttributeError?**
    - a) It falls back to __getattr__
    - b) It raises the AttributeError
    - c) It uses a default attribute
    - d) It retries the access
    - **Correct answer: a**
21. **How can you create a singleton class in Python?**
    - a) By overriding __new__ to return the same instance
    - b) By using __slots__

    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**
22. **What is the role of the** __call__ **method in a class?**
    - a) To make instances callable
    - b) To initialize the instance
    - c) To define a static method
    - d) To manage class attributes
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - Thanks for Reading it https://x.com/samaddeveloper
    -
23. **What is the output of** isinstance(obj, MyClass) **for an instance** obj **of** MyClass**?**
    - a) True
    - b) False
    - c) None
    - d) Raises a TypeError
    - **Correct answer: a**
24. **How does Python handle dynamic attribute creation?**
    - a) It prevents it by default
    - b) It allows it unless __slots__ is defined
    - c) It requires a decorator
    - d) It raises an AttributeError
    - **Correct answer: b**
25. **What is the purpose of the** __setattr__ **method?**
    - a) To customize attribute setting
    - b) To initialize attributes
    - c) To define a static method
    - d) To manage inheritance
    - **Correct answer: a**
26. **What happens if** __setattr__ **causes an infinite recursion?**
    - a) It raises a RecursionError
    - b) It ignores the recursion
    - c) It uses a default attribute
    - d) It retries the operation
    - **Correct answer: a**
27. **How can you prevent attribute deletion in a class?**
    - a) By overriding __delattr__
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**
28. **What is the role of the** __dir__ **method?**
    - a) To list available attributes and methods

  - b) To initialize the instance
  - c) To define a static method
  - d) To manage class attributes
  - **Correct answer: a**
29. **What happens if a class defines both** __getattribute__ **and** __getattr__**?**
    - a) __getattr__ is ignored
    - b) __getattribute__ is called first
    - c) __getattr__ is called first
    - d) It raises a TypeError
    - **Correct answer: b**
30. **How can you create a class with immutable attributes?**
    - a) By overriding __setattr__ to raise an error
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**
31. **What is the purpose of the** __repr__ **method?**
    - a) To define a human-readable string representation
    - b) To define a machine-readable string representation
    - c) To initialize the instance
    - d) To manage class attributes
    - **Correct answer: b**
32. **What is the difference between** __str__ **and** __repr__**?**
    - a) __str__ is for debugging, __repr__ is for users
    - b) __repr__ is for debugging, __str__ is for users
    - c) They are identical
    - d) __str__ is for class attributes
    - **Correct answer: b**
33. **How does Python handle attribute lookup for undefined attributes?**
    - a) It calls __getattr__ if defined
    - b) It raises an AttributeError
    - c) It uses a default attribute
    - d) It retries the lookup
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - Thanks for Reading it
      https://x.com/samaddeveloper
    -
34. **What is the effect of defining** __slots__ **in a subclass?**
    - a) It inherits the base class's __slots__
    - b) It defines new slots for the subclass
    - c) It prevents slot usage
    - d) It raises a TypeError
    - **Correct answer: b**
35. **How can you create a class with a custom attribute access pattern?**

- a) By overriding __getattribute__ and __setattr__
- b) By using __slots__
- c) By defining a static method
- d) By using a metaclass
- **Correct answer: a**

36. **What happens if __new__ returns an instance of a different class?**
    - a) It raises a TypeError
    - b) It uses the returned instance
    - c) It ignores the return value
    - d) It retries the creation
    - **Correct answer: b**
37. **How does Python handle instance deletion?**
    - a) By calling __del__ if defined
    - b) By raising a TypeError
    - c) By ignoring the deletion
    - d) By resetting the instance
    - **Correct answer: a**
38. **What is the purpose of the __bool__ method?**
    - a) To define the truth value of an instance
    - b) To initialize the instance
    - c) To define a static method
    - d) To manage class attributes
    - **Correct answer: a**
39. **How can you create a class with a fixed set of attributes?**
    - a) By using __slots__
    - b) By overriding __setattr__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**
40. **What happens if __getattribute__ is called recursively?**
    - a) It raises a RecursionError
    - b) It ignores the recursion
    - c) It uses a default attribute
    - d) It retries the operation
    - **Correct answer: a**
41. **How does Python handle attribute deletion with __delattr__?**
    - a) It customizes the deletion behavior
    - b) It prevents deletion
    - c) It raises a TypeError
    - d) It ignores the deletion
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - Thanks for Reading it
      https://x.com/samaddeveloper
    -

42. **What is the role of the** __hash__ **method?**
    - a) To define the hash value of an instance
    - b) To initialize the instance
    - c) To define a static method
    - d) To manage class attributes
    - **Correct answer: a**
43. **How can you ensure a class is immutable after initialization?**
    - a) By overriding __setattr__ to prevent changes
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**

Inheritance and Polymorphism (40 Questions)

41. **What is the purpose of the** super() **function in Python?**
    - a) To call a parent class's method
    - b) To initialize a subclass
    - c) To define a static method
    - d) To manage class attributes
    - **Correct answer: a**
42. **How does Python resolve method calls in multiple inheritance?**
    - a) Using the Method Resolution Order (MRO)
    - b) Using depth-first search
    - c) Using breadth-first search
    - d) It raises a TypeError
    - **Correct answer: a**
43. **What is the output of** MyClass.__mro__ **for a class** MyClass**?**
    - a) A tuple of base classes in resolution order
    - b) A list of instance attributes
    - c) A dictionary of class attributes
    - d) A set of methods
    - **Correct answer: a**
44. **What happens if a subclass overrides a parent class method?**
    - a) The subclass method is called
    - b) The parent class method is called
    - c) It raises a NameError
    - d) It merges both methods
    - **Correct answer: a**
45. **How can you call a parent class's** __init__ **method from a subclass?**
    - a) Using super().__init__()
    - b) Using self.__init__()
    - c) Using Parent.__init__()
    - d) Both a and c
    - **Correct answer: d**
46. **What is the effect of multiple inheritance in Python?**
    - a) It allows a class to inherit from multiple base classes

- b) It prevents method overriding
- c) It raises a TypeError
- d) It restricts attribute access
- **Correct answer: a**

47. **How does Python handle diamond inheritance?**
    - a) Using the C3 linearization algorithm
    - b) Using depth-first search
    - c) Using breadth-first search
    - d) It raises a TypeError
    - **Correct answer: a**
48. **What is the purpose of the** isinstance() **function in inheritance?**
    - a) To check if an object is an instance of a class or its subclasses
    - b) To check if a class is a subclass
    - c) To initialize an instance
    - d) To manage class attributes
    - **Correct answer: a**
49. **What is the role of the** issubclass() **function?**
    - a) To check if a class is a subclass of another
    - b) To check if an object is an instance
    - c) To initialize a subclass
    - d) To manage class attributes
    - **Correct answer: a**
50. **What happens if a subclass does not define** __init__**?**
    - a) It uses the parent class's __init__
    - b) It raises a TypeError
    - c) It uses a default __init__
    - d) It prevents instantiation
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - Thanks for Reading it https://x.com/samaddeveloper
    -
51. **How can you override a parent class's method in a subclass?**
    - a) By defining a method with the same name
    - b) By using the @override decorator
    - c) By calling super().override()
    - d) It is not possible
    - **Correct answer: a**
52. **What is the effect of calling** super() **in a method with multiple inheritance?**
    - a) It calls the next method in the MRO
    - b) It calls all parent methods
    - c) It raises a TypeError
    - d) It ignores the call
    - **Correct answer: a**

53. **How does Python handle method resolution in a complex inheritance hierarchy?**
    - a) Using the C3 linearization algorithm
    - b) Using depth-first search
    - c) Using breadth-first search
    - d) It raises a TypeError
    - **Correct answer: a**
54. **What happens if two parent classes define the same method in multiple inheritance?**
    - a) The method from the first parent in the MRO is used
    - b) Both methods are called
    - c) It raises a NameError
    - d) It merges the methods
    - **Correct answer: a**
55. **How can you access a parent class's method without using** super()**?**
    - a) By calling ParentClass.method(self)
    - b) By using self.method()
    - c) By defining a static method
    - d) It is not possible
    - **Correct answer: a**
56. **What is the purpose of the** __mro__ **attribute?**
    - a) To define the method resolution order
    - b) To initialize the class
    - c) To manage class attributes
    - d) To create a static method
    - **Correct answer: a**
57. **What happens if a class inherits from two classes with conflicting attributes?**
    - a) The attribute from the first class in the MRO is used
    - b) It raises an AttributeError
    - c) It merges the attributes
    - d) It ignores the attributes
    - **Correct answer: a**
58. **How can you implement polymorphism in Python?**
    - a) By overriding methods in subclasses
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**
59. **What is the output of** super(MyClass, self).__init__() **in a subclass?**
    - a) It calls the parent class's __init__
    - b) It raises a TypeError
    - c) It initializes the subclass
    - d) It ignores the call
    - **Correct answer: a**
60. **How does Python handle method overriding in a deep inheritance hierarchy?**
    - a) It uses the method from the closest subclass
    - b) It calls all overridden methods
    - c) It raises a NameError

    - d) It merges the methods
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - Thanks for Reading it https://x.com/samaddeveloper
    -

61. **What is the effect of using** super() **without arguments in Python 3?**
    - a) It automatically resolves the parent class
    - b) It raises a SyntaxError
    - c) It calls the subclass's method
    - d) It ignores the call
    - **Correct answer: a**
62. **How can you check the MRO of a class?**
    - a) Using MyClass.__mro__
    - b) Using MyClass.mro()
    - c) Using type(MyClass).mro()
    - d) Both a and b
    - **Correct answer: d**
63. **What happens if a subclass calls a parent method that does not exist?**
    - a) It raises an AttributeError
    - b) It uses a default method
    - c) It ignores the call
    - d) It retries the call
    - **Correct answer: a**
64. **How can you implement cooperative multiple inheritance?**
    - a) By using super() to call parent methods
    - b) By defining a static method
    - c) By using __slots__
    - d) By using a metaclass
    - **Correct answer: a**
65. **What is the role of the** mro() **method in a class?**
    - a) To return the method resolution order
    - b) To initialize the class
    - c) To manage class attributes
    - d) To create a static method
    - **Correct answer: a**
66. **What happens if a class inherits from an invalid base class?**
    - a) It raises a TypeError
    - b) It uses a default base class
    - c) It ignores the base class
    - d) It retries the inheritance
    - **Correct answer: a**
67. **How does Python handle method resolution in diamond inheritance?**
    - a) Using the C3 linearization algorithm
    - b) Using depth-first search

- c) Using breadth-first search
- d) It raises a TypeError
- **Correct answer: a**
- **Created by https://github.com/samade747**
- Thanks for Reading it https://x.com/samaddeveloper
-

68. **What is the purpose of the** @classmethod **decorator in inheritance?**
    - a) To define a method that takes the class as the first argument
    - b) To define a static method
    - c) To initialize the instance
    - d) To manage class attributes
    - **Correct answer: a**
69. **How can you call a grandparent class's method in a subclass?**
    - a) Using GrandparentClass.method(self)
    - b) Using super(ParentClass, self).method()
    - c) Using super().__init__()
    - d) Both a and b
    - **Correct answer: d**
70. **What happens if a subclass defines a method that conflicts with a parent's property?**
    - a) The subclass method takes precedence
    - b) The parent property takes precedence
    - c) It raises a TypeError
    - d) It merges both
    - **Correct answer: a**
71. **How does Python handle super calls in a complex MRO?**
    - a) It follows the MRO to resolve the next class
    - b) It calls all parent classes
    - c) It raises a TypeError
    - d) It ignores the call
    - **Correct answer: a**
72. **What is the effect of using** super() **in a static method?**
    - a) It raises a TypeError
    - b) It calls the parent's static method
    - c) It initializes the class
    - d) It ignores the call
    - **Correct answer: a**
73. **How can you implement method overloading in Python?**
    - a) Python does not support method overloading
    - b) By using the @overload decorator
    - c) By defining multiple methods with the same name
    - d) By using a metaclass
    - **Correct answer: a**
74. **What happens if a subclass overrides a parent's class method?**

- a) The subclass method is called
- b) The parent method is called
- c) It raises a NameError
- d) It merges both methods
- **Correct answer: a**

75. **How does Python handle attribute resolution in multiple inheritance?**
    - a) Using the MRO
    - b) Using depth-first search
    - c) Using breadth-first search
    - d) It raises a TypeError
    - **Correct answer: a**
76. **What is the purpose of the** @staticmethod **decorator in inheritance?**
    - a) To define a method that does not take self or cls
    - b) To initialize the class
    - c) To manage class attributes
    - d) To create a class method
    - **Correct answer: a**
77. **How can you ensure a subclass calls all parent** __init__ **methods in multiple inheritance?**
    - a) By using super().__init__() in cooperative inheritance
    - b) By calling each parent's __init__ explicitly
    - c) By defining a static method
    - d) Both a and b
    - **Correct answer: a**
78. **What happens if a subclass inherits from two classes with conflicting MROs?**
    - a) It raises a TypeError
    - b) It uses the first class's MRO
    - c) It ignores the conflict
    - d) It merges the MROs
    - **Correct answer: a**
79. **How does Python handle polymorphism with dynamic typing?**
    - a) By allowing methods to be called based on object type
    - b) By requiring explicit type declarations
    - c) By preventing method overriding
    - d) By raising a TypeError
    - **Correct answer: a**
80. **What is the effect of using** super() **in a class with no parent?**
    - a) It raises a TypeError
    - b) It calls object's method
    - c) It ignores the call
    - d) It retries the call
    - **Correct answer: bCreated by https://github.com/samade747**
    - Thanks for Reading it https://x.com/samaddeveloper
    -

Encapsulation and Properties (30 Questions)

81. **What is the purpose of the** @property **decorator?**
    - a) To define a getter method for an attribute
    - b) To initialize an instance
    - c) To define a static method
    - d) To manage class attributes
    - **Correct answer: a**
82. **How can you create a read-only property?**
    - a) By defining a @property without a setter
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**
83. **What is the effect of using a leading underscore in an attribute name (e.g.,** _attr**)?**
    - a) It marks the attribute as private by convention
    - b) It prevents attribute access
    - c) It makes the attribute read-only
    - d) It raises an AttributeError
    - **Correct answer: a**
84. **How does Python handle double-underscore attributes (e.g.,** __attr**)?**
    - a) It performs name mangling
    - b) It makes them read-only
    - c) It prevents attribute access
    - d) It raises a TypeError
    - **Correct answer: a**
85. **What is the output of accessing** obj.__attr **in a class with** __attr **defined?**
    - a) It raises an AttributeError
    - b) It accesses the mangled attribute
    - c) It accesses the attribute directly
    - d) It retries the access
    - **Correct answer: a**
86. **How can you access a mangled attribute (e.g.,** __attr**) from outside the class?**
    - a) Using _ClassName__attr
    - b) Using self.__attr
    - c) Using ClassName.__attr
    - d) It is not possible
    - **Correct answer: a**
87. **What is the purpose of the** @attr.setter **decorator?**
    - a) To define a setter method for a property
    - b) To initialize the instance
    - c) To define a static method
    - d) To manage class attributes
    - **Correct answer: a**

- **Created by https://github.com/samade747**

- Thanks for Reading it https://x.com/samaddeveloper
-

88. **What happens if a property's setter is not defined?**
    - a) The property is read-only
    - b) It raises a TypeError
    - c) It uses a default setter
    - d) It ignores the property
    - **Correct answer: a**
89. **How can you create a property with both getter and setter?**
    - a) Using @property and @attr.setter
    - b) Using __slots__
    - c) Using a static method
    - d) Using a metaclass
    - **Correct answer: a**
90. **What is the role of the** @attr.deleter **decorator?**
    - a) To define a deleter method for a property
    - b) To initialize the instance
    - c) To define a static method
    - d) To manage class attributes
    - **Correct answer: a**
91. **What happens if a property's getter raises an error?**
    - a) It propagates the error
    - b) It uses a default value
    - c) It ignores the error
    - d) It retries the access
    - **Correct answer: a**
92. **How can you implement encapsulation without name mangling?**
    - a) By using a single underscore (e.g., _attr)
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**
93. **What is the effect of defining a property with a getter only?**
    - a) The property is read-only
    - b) It raises a TypeError
    - c) It uses a default setter
    - d) It ignores the property
    - **Correct answer: a**
94. **How does Python handle property overriding in a subclass?**
    - a) The subclass property takes precedence
    - b) The parent property takes precedence
    - c) It raises a TypeError
    - d) It merges both properties
    - **Correct answer: a**

- **Created by https://github.com/samade747**

- Thanks for Reading it https://x.com/samaddeveloper
-

95. **What is the purpose of name mangling with** __attr**?**
    - a) To prevent accidental attribute overriding in subclasses
    - b) To make attributes read-only
    - c) To define a static method
    - d) To manage class attributes
    - **Correct answer: a**
96. **How can you create a computed property?**
    - a) By using @property with a getter method
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**
97. **What happens if a property's setter raises an error?**
    - a) It propagates the error
    - b) It uses a default value
    - c) It ignores the error
    - d) It retries the operation
    - **Correct answer: a**
98. **How can you ensure a property is immutable?**
    - a) By defining only a getter
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**
99. **What is the role of the** @property **decorator in encapsulation?**
    - a) To control attribute access
    - b) To initialize the instance
    - c) To define a static method
    - d) To manage class attributes
    - **Correct answer: a**
100. **How does Python handle property deletion with a deleter?**
    - a) It calls the deleter method
    - b) It raises a TypeError
    - c) It ignores the deletion
    - d) It retries the deletion
    - **Correct answer: a**
101. **What happens if a property's deleter is not defined?**
    - a) Attribute deletion raises an AttributeError
    - b) It uses a default deleter
    - c) It ignores the deletion
    - d) It retries the deletion
    - **Correct answer: a**
102. **How can you create a property with validation?**
    - a) By defining a setter with validation logic

- b) By using __slots__
- c) By defining a static method
- d) By using a metaclass
- **Correct answer: a**
- **Created by https://github.com/samade747**

- Thanks for Reading it
  https://x.com/samaddeveloper
-

103. **What is the effect of using** _attr **vs** __attr **for encapsulation?**
   - a) _attr is a convention, __attr uses name mangling
   - b) Both use name mangling
   - c) Both are conventions
   - d) _attr prevents access
   - **Correct answer: a**
104. **How does Python handle property inheritance?**
   - a) Subclasses can override properties
   - b) Properties are not inherited
   - c) It raises a TypeError
   - d) It merges properties
   - **Correct answer: a**
105. **What happens if a property's getter is overridden in a subclass?**
   - a) The subclass getter is used
   - b) The parent getter is used
   - c) It raises a TypeError
   - d) It merges both getters
   - **Correct answer: a**
106. **How can you create a property with a custom getter and setter?**
   - a) Using @property and @attr.setter
   - b) Using __slots__
   - c) Using a static method
   - d) Using a metaclass
   - **Correct answer: a**
107. **What is the purpose of the** __setattr__ **method in encapsulation?**
   - a) To control attribute setting
   - b) To initialize the instance
   - c) To define a static method
   - d) To manage class attributes
   - **Correct answer: a**
108. **How does Python handle property access with** __getattribute__**?**
   - a) __getattribute__ is called before the property
   - b) The property is called first
   - c) It raises a TypeError
   - d) It ignores the property
   - **Correct answer: a**
109. **What happens if a property's setter is called with an invalid value?**

- a) It raises an error defined in the setter
- b) It uses a default value
- c) It ignores the value
- d) It retries the operation
- **Correct answer: a**
- **Created by https://github.com/samade747**
- Thanks for Reading it https://x.com/samaddeveloper
-

110. **How can you ensure a property is only set during initialization?**
    - a) By overriding __setattr__ to restrict changes
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**

Abstract Base Classes and Interfaces (30 Questions)

111. **What is the purpose of the** abc **module in Python?**
    - a) To define abstract base classes
    - b) To initialize instances
    - c) To define static methods
    - d) To manage class attributes
    - **Correct answer: a**
112. **How can you define an abstract method in a class?**
    - a) Using the @abstractmethod decorator
    - b) Using __slots__
    - c) Using a static method
    - d) Using a metaclass
    - **Correct answer: a**
113. **What happens if a subclass of an ABC does not implement an abstract method?**
    - a) It raises a TypeError on instantiation
    - b) It uses a default implementation
    - c) It ignores the abstract method
    - d) It retries the implementation
    - **Correct answer: a**
114. **How does the** ABC **class from the** abc **module function?**
    - a) It prevents instantiation of abstract classes
    - b) It initializes instances
    - c) It defines static methods
    - d) It manages class attributes
    - **Correct answer: a**
    - **Created by https://github.com/samade747**

- Thanks for Reading it
  https://x.com/samaddeveloper
-

115. **What is the role of the** @abstractmethod **decorator?**
   - a) To mark a method as required in subclasses
   - b) To initialize the instance
   - c) To define a static method
   - d) To manage class attributes
   - **Correct answer: a**
116. **How can you create an abstract property?**
   - a) Using @abstractmethod and @property
   - b) Using __slots__
   - c) Using a static method
   - d) Using a metaclass
   - **Correct answer: a**
117. **What happens if a subclass overrides an abstract method with a property?**
   - a) It is allowed
   - b) It raises a TypeError
   - c) It uses the parent's implementation
   - d) It ignores the override
   - **Correct answer: a**
118. **How does Python enforce interface-like behavior?**
   - a) Using abstract base classes
   - b) Using __slots__
   - c) Using static methods
   - d) Using metaclasses
   - **Correct answer: a**
119. **What is the purpose of the** register() **method in an ABC?**
   - a) To register a class as a virtual subclass
   - b) To initialize the instance
   - c) To define a static method
   - d) To manage class attributes
   - **Correct answer: a**

   - **Created by https://github.com/samade747**

   - Thanks for Reading it
     https://x.com/samaddeveloper
   -
120. **What happens if you instantiate an ABC with abstract methods?**
   - a) It raises a TypeError
   - b) It uses default implementations
   - c) It ignores the abstract methods
   - d) It retries the instantiation
   - **Correct answer: a**
121. **How can you check if a class is a subclass of an ABC?**

- a) Using issubclass()
- b) Using isinstance()
- c) Using type()
- d) Using super()
- **Correct answer: a**

122. **What is the effect of using** @abstractclassmethod**?**
   - a) It marks a class method as abstract
   - b) It initializes the class
   - c) It defines a static method
   - d) It manages class attributes
   - **Correct answer: a**
123. **How does Python handle virtual subclasses with** register()**?**
   - a) They are treated as subclasses for isinstance() checks
   - b) They inherit all methods
   - c) They raise a TypeError
   - d) They ignore the registration
   - **Correct answer: a**
   - **Created by https://github.com/samade747**
   - Thanks for Reading it https://x.com/samaddeveloper
   -
124. **What happens if a subclass implements an abstract property incorrectly?**
   - a) It raises a TypeError
   - b) It uses the parent's implementation
   - c) It ignores the property
   - d) It retries the implementation
   - **Correct answer: a**
125. **How can you define an abstract static method?**
   - a) Using @abstractmethod and @staticmethod
   - b) Using __slots__
   - c) Using a metaclass
   - d) Using a property
   - **Correct answer: a**
126. **What is the role of the** abc.ABCMeta **metaclass?**
   - a) To enforce abstract class behavior
   - b) To initialize instances
   - c) To define static methods
   - d) To manage class attributes
   - **Correct answer: a**
127. **How does Python handle ABCs in multiple inheritance?**
   - a) Using the MRO
   - b) Using depth-first search
   - c) Using breadth-first search
   - d) It raises a TypeError
   - **Correct answer: a**

128. **What happens if a virtual subclass does not implement abstract methods?**
   - a) It is allowed
   - b) It raises a TypeError
   - c) It uses default implementations
   - d) It ignores the abstract methods
   - **Correct answer: a**
129. **How can you enforce that a class implements multiple interfaces?**
   - a) By inheriting from multiple ABCs
   - b) By using __slots__
   - c) By defining static methods
   - d) By using a metaclass
   - **Correct answer: a**
130. **What is the purpose of the** @abstractproperty **decorator?**
   - a) To mark a property as abstract
   - b) To initialize the instance
   - c) To define a static method
   - d) To manage class attributes
   - **Correct answer: a**
131. **How does Python handle ABCs with no abstract methods?**
   - a) They can be instantiated
   - b) They raise a TypeError
   - c) They use default implementations
   - d) They ignore instantiation
   - **Correct answer: a**
   - **Created by https://github.com/samade747**
   - Thanks for Reading it
     https://x.com/samaddeveloper
   -
132. **What happens if a subclass overrides an abstract method with a static method?**
   - a) It is allowed
   - b) It raises a TypeError
   - c) It uses the parent’s implementation
   - d) It ignores the override
   - **Correct answer: a**
133. **How can you register a class as a virtual subclass of an ABC?**
   - a) Using ABC.register()
   - b) Using __slots__
   - c) Using a static method
   - d) Using a metaclass
   - **Correct answer: a**
134. **What is the effect of using** @abstractclassmethod **in a class?**
   - a) It requires subclasses to implement the class method
   - b) It initializes the class
   - c) It defines a static method

- d) It manages class attributes
- **Correct answer: a**

135. **How does Python handle ABCs in** isinstance() **checks?**
    - a) It includes registered virtual subclasses
    - b) It excludes virtual subclasses
    - c) It raises a TypeError
    - d) It ignores ABCs
    - **Correct answer: a**
136. **What happens if an ABC defines an abstract method and a concrete method?**
    - a) Subclasses must implement the abstract method
    - b) Subclasses inherit the concrete method
    - c) Both a and b
    - d) It raises a TypeError
    - **Correct answer: c**
137. **How can you create an abstract property with a setter?**
    - a) Using @abstractmethod and @property.setter
    - b) Using __slots__
    - c) Using a static method
    - d) Using a metaclass
    - **Correct answer: a**
138. **What is the role of the** abc.ABC **class?**
    - a) To simplify ABC creation with ABCMeta
    - b) To initialize instances
    - c) To define static methods
    - d) To manage class attributes
    - **Correct answer: a**
139. **What happens if a subclass implements an abstract method incorrectly?**
    - a) It raises a TypeError
    - b) It uses the parent's implementation
    - c) It ignores the method
    - d) It retries the implementation
    - **Correct answer: a**
140. **How does Python handle ABCs with multiple abstract methods?**
    - a) Subclasses must implement all abstract methods
    - b) Subclasses can ignore some methods
    - c) It uses default implementations
    - d) It raises a TypeError
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - Thanks for Reading it https://x.com/samaddeveloper
    -

Metaclasses and Descriptors (30 Questions)

141. **What is a metaclass in Python?**
   - a) A class that defines how classes are created
   - b) A class that initializes instances
   - c) A class that defines static methods
   - d) A class that manages attributes
   - **Correct answer: a**
142. **How can you define a custom metaclass?**
   - a) By subclassing type
   - b) By using __slots__
   - c) By defining a static method
   - d) By using a property
   - **Correct answer: a**
143. **What is the purpose of the** __metaclass__ **attribute in Python 2?**
   - a) To specify the metaclass
   - b) To initialize the class
   - c) To define a static method
   - d) To manage class attributes
   - **Correct answer: a**
144. **How do you specify a metaclass in Python 3?**
   - a) Using the metaclass keyword in the class definition
   - b) Using __slots__
   - c) Using a static method
   - d) Using a property
   - **Correct answer: a**
145. **What is the role of the** __init_subclass__ **method?**
   - a) To customize subclass initialization
   - b) To initialize instances
   - c) To define a static method
   - d) To manage class attributes
   - **Correct answer: a**
   - **Created by https://github.com/samade747**

   - Thanks for Reading it
     https://x.com/samaddeveloper
   -
146. **What happens if a metaclass's** __new__ **returns an instance of a different class?**
   - a) It uses the returned instance
   - b) It raises a TypeError
   - c) It ignores the return value
   - d) It retries the creation
   - **Correct answer: a**
147. **How does a metaclass affect class creation?**
   - a) It customizes the class creation process
   - b) It initializes instances
   - c) It defines static methods

- d) It manages class attributes
- **Correct answer: a**

148. **What is a descriptor in Python?**
  - a) A class with __get__, __set__, or __delete__
  - b) A class that initializes instances
  - c) A class that defines static methods
  - d) A class that manages attributes
  - **Correct answer: a**
149. **How can you create a custom descriptor?**
  - a) By defining __get__, __set__, or __delete__
  - b) By using __slots__
  - c) By defining a static method
  - d) By using a metaclass
  - **Correct answer: a**

  - **Created by https://github.com/samade747**

  - Thanks for Reading it https://x.com/samaddeveloper
  -
150. **What is the purpose of the __get__ method in a descriptor?**
  - a) To customize attribute access
  - b) To initialize the instance
  - c) To define a static method
  - d) To manage class attributes
  - **Correct answer: a**
151. **What happens if a descriptor's __set__ method raises an error?**
  - a) It propagates the error
  - b) It uses a default value
  - c) It ignores the error
  - d) It retries the operation
  - **Correct answer: a**
152. **How does Python handle descriptor precedence over instance attributes?**
  - a) Descriptors take precedence
  - b) Instance attributes take precedence
  - c) It raises a TypeError
  - d) It merges both
  - **Correct answer: a**
153. **What is the role of the __delete__ method in a descriptor?**
  - a) To customize attribute deletion
  - b) To initialize the instance
  - c) To define a static method
  - d) To manage class attributes
  - **Correct answer: a**

  - **Created by https://github.com/samade747**

- Thanks for Reading it https://x.com/samaddeveloper
-

154. **How can you create a read-only descriptor?**
    - a) By defining __get__ without __set__
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**
155. **What happens if a metaclass conflicts with a base class's metaclass?**
    - a) It raises a TypeError
    - b) It uses the base class's metaclass
    - c) It ignores the conflict
    - d) It merges the metaclasses
    - **Correct answer: a**
156. **How does a descriptor's** __get__ **method receive the instance?**
    - a) As the second argument
    - b) As the first argument
    - c) As a keyword argument
    - d) It does not receive the instance
    - **Correct answer: a**
157. **What is the purpose of the** __prepare__ **method in a metaclass?**
    - a) To customize the namespace during class creation
    - b) To initialize the instance
    - c) To define a static method
    - d) To manage class attributes
    - **Correct answer: a**
158. **How can you create a descriptor with validation?**
    - a) By defining __set__ with validation logic
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**

    - **Created by https://github.com/samade747**

    - Thanks for Reading it https://x.com/samaddeveloper
    -
159. **What happens if a descriptor's** __get__ **method is not defined?**
    - a) It raises an AttributeError
    - b) It uses a default getter
    - c) It ignores the access
    - d) It retries the access
    - **Correct answer: a**
160. **How does Python handle descriptor lookup in a class?**
    - a) It checks the class's __dict__ first

- b) It checks the instance’s __dict__ first
- c) It raises a TypeError
- d) It ignores the descriptor
- **Correct answer: a**

161. **What is the effect of using a metaclass to enforce attribute validation?**
    - a) It customizes class creation with validation
    - b) It initializes instances
    - c) It defines static methods
    - d) It manages class attributes
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - Thanks for Reading it https://x.com/samaddeveloper
    -
162. **How can you create a descriptor that logs attribute access?**
    - a) By defining __get__ with logging logic
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**
163. **What happens if a metaclass’s __init__ method raises an error?**
    - a) It propagates the error
    - b) It uses a default initialization
    - c) It ignores the error
    - d) It retries the initialization
    - **Correct answer: a**
164. **How does Python handle descriptors in multiple inheritance?**
    - a) Using the MRO
    - b) Using depth-first search
    - c) Using breadth-first search
    - d) It raises a TypeError
    - **Correct answer: a**
165. **What is the role of the __call__ method in a metaclass?**
    - a) To customize class instantiation
    - b) To initialize the instance
    - c) To define a static method
    - d) To manage class attributes
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - Thanks for Reading it https://x.com/samaddeveloper
    -
166. **How can you create a descriptor with a custom deleter?**

- a) By defining __delete__
- b) By using __slots__
- c) By defining a static method
- d) By using a metaclass
- **Correct answer: a**

167. **What happens if a descriptor's __set__ method is not defined?**
    - a) The attribute is read-only
    - b) It raises a TypeError
    - c) It uses a default setter
    - d) It ignores the setter
    - **Correct answer: a**
168. **How does a metaclass's __new__ method affect class creation?**
    - a) It customizes the class creation process
    - b) It initializes instances
    - c) It defines static methods
    - d) It manages class attributes
    - **Correct answer: a**
169. **What is the purpose of the __init_subclass__ method in a base class?**
    - a) To customize subclass initialization
    - b) To initialize instances
    - c) To define a static method
    - d) To manage class attributes
    - **Correct answer: a**
170. **How does Python handle descriptor conflicts in a class?**
    - a) The first descriptor in the MRO takes precedence
    - b) It raises a TypeError
    - c) It merges the descriptors
    - d) It ignores the descriptors
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - Thanks for Reading it https://x.com/samaddeveloper
    -

Advanced OOP Patterns (30 Questions)

171. **What is the purpose of the Singleton pattern in Python?**
    - a) To ensure a class has only one instance
    - b) To initialize multiple instances
    - c) To define a static method
    - d) To manage class attributes
    - **Correct answer: a**
172. **How can you implement the Singleton pattern using a metaclass?**
    - a) By overriding __call__ to return the same instance
    - b) By using __slots__

- c) By defining a static method
- d) By using a property
- **Correct answer: a**

173. **What is the Factory pattern in Python?**
    - a) A pattern for creating objects without specifying the exact class
    - b) A pattern for initializing instances
    - c) A pattern for defining static methods
    - d) A pattern for managing class attributes
    - **Correct answer: a**
174. **How can you implement the Factory pattern?**
    - a) Using a class method to create instances
    - b) Using __slots__
    - c) Using a metaclass
    - d) Using a property
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - Thanks for Reading it https://x.com/samaddeveloper
    -
175. **What is the purpose of the Decorator pattern in Python?**
    - a) To dynamically add functionality to objects
    - b) To initialize instances
    - c) To define static methods
    - d) To manage class attributes
    - **Correct answer: a**
176. **How can you implement the Decorator pattern?**
    - a) By wrapping a class with another class
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**
177. **What is the Observer pattern in Python?**
    - a) A pattern for notifying objects of state changes
    - b) A pattern for initializing instances
    - c) A pattern for defining static methods
    - d) A pattern for managing class attributes
    - **Correct answer: a**
178. **How can you implement the Observer pattern?**
    - a) Using a list of observers and a notify method
    - b) Using __slots__
    - c) Using a metaclass
    - d) Using a property
    - **Correct answer: a**
179. **What is the purpose of the Strategy pattern?**
    - a) To define interchangeable algorithms

- b) To initialize instances
- c) To define static methods
- d) To manage class attributes
- **Correct answer: a**

180. **How can you implement the Strategy pattern in Python?**
   - a) By defining a class with interchangeable methods
   - b) By using __slots__
   - c) By defining a static method
   - d) By using a metaclass
   - **Correct answer: a**

181. **What is the role of the Context Manager pattern in Python?**
   - a) To manage resource acquisition and release
   - b) To initialize instances
   - c) To define static methods
   - d) To manage class attributes
   - **Correct answer: a**

182. **How can you create a custom context manager?**
   - a) By defining __enter__ and __exit__
   - b) By using __slots__
   - c) By defining a static method
   - d) By using a metaclass
   - **Correct answer: a**
   - **Created by https://github.com/samade747**
   - Thanks for Reading it
     https://x.com/samaddeveloper
   -

183. **What happens if a context manager's __exit__ method raises an error?**
   - a) It propagates the error
   - b) It ignores the error
   - c) It retries the operation
   - d) It uses a default exit
   - **Correct answer: a**

184. **How can you implement the Adapter pattern in Python?**
   - a) By creating a class that adapts one interface to another
   - b) By using __slots__
   - c) By defining a static method
   - d) By using a metaclass
   - **Correct answer: a**

185. **What is the purpose of the Facade pattern?**
   - a) To provide a simplified interface to a complex subsystem
   - b) To initialize instances
   - c) To define static methods
   - d) To manage class attributes
   - **Correct answer: a**

186. **How can you implement the Facade pattern?**

- a) By creating a class with simplified methods
- b) By using __slots__
- c) By defining a static method
- d) By using a metaclass
- **Correct answer: a**

187. **What is the role of the Proxy pattern in Python?**
    - a) To control access to an object
    - b) To initialize instances
    - c) To define static methods
    - d) To manage class attributes
    - **Correct answer: a**
    - **Created by https://github.com/samade747**
    - Thanks for Reading it https://x.com/samaddeveloper
    -
188. **How can you implement the Proxy pattern?**
    - a) By creating a class that delegates to another object
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**
189. **What is the purpose of the Command pattern?**
    - a) To encapsulate a request as an object
    - b) To initialize instances
    - c) To define static methods
    - d) To manage class attributes
    - **Correct answer: a**
190. **How can you implement the Command pattern?**
    - a) By defining a class with an execute method
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**
191. **What is the role of the Template Method pattern?**
    - a) To define a skeleton for an algorithm
    - b) To initialize instances
    - c) To define static methods
    - d) To manage class attributes
    - **Correct answer: a**
192. **How can you implement the Template Method pattern?**
    - a) By defining a base class with a template method
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**

193. **What happens if a context manager's __enter__ method raises an error?**
   - a) It propagates the error
   - b) It ignores the error
   - c) It retries the operation
   - d) It uses a default entry
   - **Correct answer: a**
194. **How can you implement the Builder pattern in Python?**
   - a) By creating a class that builds objects step-by-step
   - b) By using __slots__
   - c) By defining a static method
   - d) By using a metaclass
   - **Correct answer: a**
195. **What is the purpose of the Chain of Responsibility pattern?**
   - a) To pass requests along a chain of handlers
   - b) To initialize instances
   - c) To define static methods
   - d) To manage class attributes
   - **Correct answer: a**
196. **How can you implement the Chain of Responsibility pattern?**
   - a) By defining a class with a successor reference
   - b) By using __slots__
   - c) By defining a static method
   - d) By using a metaclass
   - **Correct answer: a**
197. **What is the role of the Mediator pattern?**
   - a) To centralize communication between objects
   - b) To initialize instances
   - c) To define static methods
   - d) To manage class attributes
   - **Correct answer: a**
   - **Created by https://github.com/samade747**
   - Thanks for Reading it https://x.com/samaddeveloper
   -
198. **How can you implement the Mediator pattern?**
   - a) By creating a class that manages object interactions
   - b) By using __slots__
   - c) By defining a static method
   - d) By using a metaclass
   - **Correct answer: a**
199. **What is the purpose of the Memento pattern?**
   - a) To capture and restore an object's state
   - b) To initialize instances
   - c) To define static methods
   - d) To manage class attributes

- **Correct answer: a**

200. **How can you implement the Memento pattern in Python?**
    - a) By creating a class to store and restore state
    - b) By using __slots__
    - c) By defining a static method
    - d) By using a metaclass
    - **Correct answer: a**