



-Expert Verified, Online, **Free**.



CERTIFICATION TEST

-CertificationTest.net - Cheap & Quality Resources With Best Support

Question #1

Topic 1

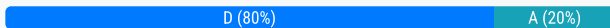
You are starting a new Claude application and have a small set of well-labeled examples that demonstrate the desired output format. You want to use these examples to guide Claude's behavior. How would you guide the application's behavior?

- A. Embed the examples in a database and retrieve them at runtime as reference material for the team.
- B. Use zero-shot prompting and rely on the model's general capability to produce the desired output format consistently across all incoming requests.
- C. Train a custom model on the labeled examples before deployment so that the application does not need to include the examples in any prompt during operation.
- D. Use multi-shot prompting by including the labeled examples in the prompt so Claude can match the desired output format on each request.

Suggested Answer: D

Explanation: Examples supplied in the prompt demonstrate the desired input-to-output pattern without retraining the model.

Community vote distribution



🗒️ **fae1a76** Most Recent 4 days ago

Selected Answer: D

A small set of well-labeled examples is exactly what few-shot (multi-shot) prompting is for.

upvoted 1 times

🗒️ **Jidnyasa** 4 days ago

Selected Answer: A

A. Embed the examples in a database and retrieve them at runtime as reference material for the team.

upvoted 1 times

🗒️ **414ae6a** 1 week ago

Selected Answer: D

Are these questions up-to-date and do they appear on the exam?

upvoted 1 times

🗒️ **fireweed** 1 month, 2 weeks ago

Selected Answer: D

Multi-shot prompting with varied examples will always be most likely to create consistent answers

upvoted 2 times

Your Claude application's content policy specifies categories of content it should not produce under any circumstance. The application currently has no mechanism to enforce this policy, and content matching these categories is appearing in the application's output. How would you enforce the content policy?

- A. Enhance the system prompt to contain explicit instructions for the categories to avoid, complete with examples of each category. Treat the strengthened prompt as the primary enforcement mechanism for the application's content policy across all responses.
- B. Remove the content policy entirely and let any output reach users during normal operation, accepting whatever content the application produces in response to incoming traffic.
- C. Add deterministic output filtering that checks responses against the content policy before they reach users.
- D. Move enforcement to users by asking them to report content policy violations after the violating content has already reached them in the application's responses.

Suggested Answer: *C*

Explanation: Enforce prohibited-output rules before delivery with deterministic checks; prompt instructions alone are not a guaranteed enforcement layer.

Currently there are no comments in this discussion, be the first to comment!

A new Claude model release includes performance improvements for several reasoning tasks but has changed the format of its responses to system prompts that use multi-section instructions. Your application uses multi-section system prompts heavily. Initial evaluation on the application's actual workload shows the new model performs 8 percent better on reasoning tasks but produces malformed output on roughly 3 percent of requests because of the format change. The team is debating whether to upgrade.

How would you decide?

- A. Upgrade and add a downstream validation step that catches the 3 percent malformed output before it reaches users, treating the validation step as the team's mitigation for the format change.
- B. Adapt the application's system prompt to the new model's format expectations and re-evaluate, then upgrade only if the adapted prompt eliminates the malformed output while preserving the reasoning improvements.
- C. Stay on the previous model permanently to avoid the malformed output rate and any future format changes that subsequent model releases might introduce in the application.
- D. Upgrade immediately, because the 8 percent reasoning improvement outweighs the 3 percent malformed output rate across the application's typical request distribution.

Suggested Answer: B

Explanation: Adapt the prompt and re-evaluate representative workloads before accepting a model upgrade that changes output behavior.

Community vote distribution

B (100%)

🗉  **sapkrishna** Most Recent 2 weeks ago

Selected Answer: B

You should upgrade to the new model, but only after implementing a remediation layer or updating your system prompt templates
upvoted 1 times

Your team's Claude application has been in production for a year, and the team has decided to formalize its testing strategy. Currently, the team writes ad-hoc tests for individual features but has no overall testing approach. What testing approach would you formalize?

- A. Adopt a test-driven development practice where unit tests are written before each feature is implemented and must pass before code is merged.
- B. Define unit tests for individual functions, integration tests for the Claude integration, and end-to-end tests for critical user flows, applied consistently across the codebase.
- C. Continue writing ad-hoc tests as features ship and introduce a peer review step to ensure each test adequately covers the feature being released.
- D. Define a single testing approach that uses end-to-end tests and apply it consistently across all new features as they are added to the codebase.

Suggested Answer: B

Explanation: A coherent strategy covers functions, integration boundaries, and critical complete user journeys.

Community vote distribution

B (100%)

🗨️ **fireweed** Most Recent 1 month, 2 weeks ago

Selected Answer: B

This speaks more to development best practice. It's usually safe to assume that there is no such thing as overkill when it comes to ensuring the product continues to function.

upvoted 1 times

Your Claude application makes high-volume API calls during business hours and very few calls overnight. The team is concerned about staying within rate limits during peak hours and wants to understand how the Claude API enforces those limits. How would you proceed?

- A. Examine the peak-hour request patterns in your application logs and smooth traffic by distributing requests more evenly across the business-hours window.
- B. Assess the average payload size of current API calls and consolidate requests where possible to reduce the total number of calls made during peak hours.
- C. Review the API documentation for streaming endpoints and evaluate whether migrating peak-hour calls to streaming reduces exposure to rate limit enforcement.
- D. Identify the rate limits, design the application to stay within them during peak hours, and use exponential backoff when limits are reached.

Suggested Answer: *D*

Explanation: Identify applicable limits, control peak traffic, and back off with bounded retries when throttled.

Currently there are no comments in this discussion, be the first to comment!

You are designing a Claude application that will require structured JSON output for downstream processing. The output schema is well-defined, and downstream systems will reject malformed JSON. Your application design would ...

- A. avoid structured output and use free-form text everywhere instead, on the grounds that free-form text is more flexible and handles edge cases better than structured schemas.
- B. structure the prompt to request output in a schema that is described in plain English, with downstream systems parsing whatever shape Claude produces.
- C. define a clear schema and structure the prompt to request output in that schema, with downstream systems handling any validation needed.
- D. define a clear schema, structure the prompt to request output in that schema, and validate Claude's output against the schema before passing it downstream.

Suggested Answer: *D*

Explanation: A requested schema must also be validated before downstream code consumes the result.

Currently there are no comments in this discussion, be the first to comment!

Your Claude application produces good responses for typical inputs but struggles with edge cases. You have several labeled examples of edge-case inputs and the desired response for each. You want to use these examples to improve the model's handling of edge cases. What is the best way to use these examples?

- A. Embed the examples in a database for the model to find during inference.
- B. Train a custom model on the edge-case examples and deploy that custom model in place of the team's current Claude integration.
- C. Add the labeled edge-case examples to the prompt as few-shot examples so the model can learn the pattern.
- D. Tell users to avoid submitting the edge-case inputs to the application by adding warnings in the application's user interface.

Suggested Answer: C

Explanation: Representative labeled edge cases in a few-shot prompt teach the desired behavior directly.

Currently there are no comments in this discussion, be the first to comment!

The team is debating whether to build a new capability as a custom tool or to use an existing built-in tool that nearly covers the use case but lacks one specific feature. How would you decide?

- A. Review the built-in tool's documentation and assess whether its existing capabilities are sufficient to cover the use case without any modification or extension.
- B. Identify whether the team has previously built similar custom tools and use those precedents to determine which approach is more consistent with the existing codebase.
- C. Examine the custom tool option by scoping the implementation effort and determining whether the additional flexibility justifies the development and maintenance cost.
- D. Evaluate whether the missing feature can be addressed by extending the built-in tool or warrants a custom tool, then choose accordingly.

Suggested Answer: *D*

Explanation: Compare the actual missing capability and extension cost before choosing an existing tool or a custom implementation.

Currently there are no comments in this discussion, be the first to comment!

Your Claude agent's hooks are currently triggered for every action, which slows down the agent significantly even when actions pose no risk. The team wants to scope hooks more carefully. How would you scope the hooks?

- A. Replace hooks with system prompt instructions on the grounds that prompt instructions can produce the same enforcement effect that hooks produce on the agent's actions.
- B. Scope hooks to only the high-risk actions, such as destructive operations or sensitive data access, and remove hooks from low-risk actions to balance safety with performance.
- C. Disable all hooks while the team re-scopes them, treating the period of no hook enforcement as a temporary state during the re-scoping work.
- D. Disable the agent during peak hours so the hook overhead does not slow the application down during the busiest periods of the day across the application's operation.

Suggested Answer: *B*

Explanation: Apply hooks at the operations that need enforcement; avoid unnecessary overhead on low-risk actions.

Currently there are no comments in this discussion, be the first to comment!

You are running Claude Code as part of an automated continuous integration pipeline. The pipeline needs Claude Code to execute a set of well-defined tasks without prompting for confirmation, and the output needs to be captured for downstream processing. How would you configure the pipeline?

- A. Disable Claude Code's confirmation prompts globally across all environments so the pipeline runs without interruption from any prompt.
- B. Run Claude Code in headless mode with the required permissions configured in settings.json and capture its output for downstream processing.
- C. Replace Claude Code with a different tool that does not require any configuration to operate without confirmation prompts in the pipeline.
- D. Run Claude Code in interactive mode and have a developer manually approve every confirmation prompt while the pipeline executes its tasks.

Suggested Answer: B

Explanation: Headless execution with explicit scoped permissions supports unattended CI and captured output without globally disabling safeguards.

Currently there are no comments in this discussion, be the first to comment!

The Anthropic API deprecated a request parameter that your Claude application uses in approximately 40 places across the codebase. The deprecation notice gives a six-month window before the parameter is removed and recommends a replacement parameter with slightly different semantics. You would respond to the deprecation by ...

- A. swapping all 40 call sites in a single change right away to prevent drawn-out migration work that will delay ongoing functioning.
- B. migrating all 40 call sites in a single change near the removal date to ensure the deprecated parameter continues to work as long as possible.
- C. adding regression tests to cover the parameter's behavior, then migrating call sites in batches that you validate against regression tests.
- D. keeping the deprecated parameter in place while writing a wrapper function around it to insulate the rest of the codebase from the eventual change.

Suggested Answer: C

AI-derived guidance: Regression coverage and validated batches reduce migration risk when replacement semantics differ.

 **ami007** Most Recent 2 weeks ago

Selected Answer: C

A six-month window is enough to migrate safely. The replacement has slightly different semantics, so regression tests come first, then batched migration validated against them. A (single change now) and B (single change at the deadline) are both risky big-bang migrations. D keeps the deprecated parameter, so it only delays the problem.

upvoted 1 times

Your Claude application's API keys are stored in a secrets manager. The team is debating whether the same key should be used in development, staging, and production environments. How would you handle the keys across environments?

- A. Use a single development key everywhere on the grounds that production keys are too risky to deploy across the application's three environments during normal operation.
- B. Use distinct keys for each environment so a compromise in one environment does not affect the others during normal operation across the application's lifecycle.
- C. Rotate the same key across environments at random intervals on the grounds that random rotation provides isolation between environments without requiring distinct keys.
- D. Use the same key across all environments for simplicity and treat the propagation of any compromise as a known operational tradeoff for the team's key management approach.

Suggested Answer: B

AI-derived guidance: Environment-specific credentials limit the blast radius of a compromised key.

 **ami007** Most Recent 2 weeks ago

Selected Answer: B

Use separate keys per environment so a leak in dev doesn't expose production, and so you can revoke or rotate each independently. The other options all weaken isolation.

upvoted 1 times

Your team's Claude agent has accumulated several customizations that bypass the SDK's defaults, including custom history management, retry logic, and error handling. A new team member has proposed reverting all the customizations to maintain the codebase more easily. The tech lead disagrees and says each customization was added for a reason. How would you advise the team?

- A. Revert all customizations to the SDK's defaults to standardize the codebase.
- B. Decide on each customization individually based on its original reason and the SDK's current capabilities.
- C. Keep all customizations, trusting that the tech lead's original reasoning is still valid.
- D. Migrate the agent off the SDK and rebuild it with a custom loop.

Suggested Answer: B

AI-derived guidance: Retain or remove each customization according to its current purpose and what the SDK now provides.

 **ami007** Most Recent 2 weeks ago

Selected Answer: B

Judge each customization on its original reason and whether the SDK now covers it.
upvoted 1 times

A Claude application is occasionally refusing to answer questions that should be in scope, including questions the application has answered correctly in the past. You want to investigate. What is the first step of your investigation?

- A. Identify which questions have been refused and compare them against previously answered questions to determine what changed between the successful and refused interactions.
- B. Test whether removing or adjusting those instructions resolves the behavior.
- C. Assess whether the refusals cluster around specific input patterns, question types, or time periods by analyzing the distribution of refused requests in the logs.
- D. Examine traces of the refused requests to identify what triggers the refusal, whether input patterns, system prompt content, or other context.

Suggested Answer: *D*

AI-derived guidance: Inspect complete refused-request traces to locate the trigger before changing prompts or application behavior.

ami007 Most Recent 2 weeks ago

Selected Answer: D

Investigate first by examining traces of the refused requests to see what actually triggers the refusal.

upvoted 2 times

Reda12q 2 weeks ago

Selected Answer: A

First step: collect the refused cases and compare them with the successful cases.

upvoted 2 times

Your Claude application processes 50-page legal contracts and produces summaries with citation references back to the source. The team is debating whether to send each contract whole or split into smaller pieces. The contracts fit within Claude's context window. Initial testing shows that whole- document processing produces summaries with stronger cross-section reasoning but occasionally drifts on citation accuracy in later sections. Chunked processing produces stronger citation accuracy per chunk but loses cross-section reasoning. The team has not decided which property matters more. How would you guide the team's decision?

- A. Assess the cost and latency implications of both approaches against the application's performance requirements before recommending which processing strategy to adopt.
- B. Examine the use cases where cross-section reasoning failures occur and assess whether whole-document processing meets the application's accuracy requirements across a representative sample of contracts.
- C. Review the citation accuracy results from chunked processing and determine whether the loss of cross-section reasoning produces summaries that still meet the application's quality bar.
- D. Identify which property matters more for the application's actual use case and let that decision drive the chunking approach, then validate the choice against representative contracts.

Suggested Answer: D

AI-derived guidance: Choose the processing strategy from the actual priority between citation fidelity and cross-section reasoning, then validate it.

ami007 Most Recent 2 weeks ago

Selected Answer: D

The team hasn't decided which property matters more, so decide that first from the real use case, then validate on representative contracts.
upvoted 1 times

A teammate is reviewing the team's threat model for a Claude application and has asked you to identify the categories of AI-specific threats that the model should cover. The teammate has already listed traditional web application threats and wants to know what additional categories apply to a Claude application.

Which AI-specific threat categories would you add?

- A. Supply chain attacks on the Claude SDK because the SDK itself is the only point of vulnerability that a Claude application introduces beyond traditional web application threats.
- B. Network-level denial of service and physical infrastructure attacks, because these categories cover the threats most likely to affect any Claude application in production.
- C. Cross-site scripting and SQL injection, because these traditional web application threats apply with equal weight to any application that uses Claude in any way.
- D. Prompt injection, data leakage from prompts or context, jailbreak attempts, and unsafe model output that bypasses application controls.

Suggested Answer: D

AI-derived guidance: AI-specific risks include untrusted instructions, disclosure of context, jailbreaks, and unsafe generated output.

 **ami007** Most Recent 2 weeks ago

Selected Answer: D

Prompt injection, data leakage, jailbreaks, and unsafe output are the AI-specific categories.

upvoted 1 times

A team has deployed a multi-agent system in which a primary agent decomposes user requests and delegates subtasks to three specialized subagents: one for data retrieval, one for analysis, and one for report generation. In production, the team observes that subagents are making redundant tool calls, occasionally exceeding token budgets, and sometimes producing outputs that contradict each other – all of which the primary agent passes along without catching. What is the most appropriate way to address these failures?

- A. Strengthen the primary agent's management layer to enforce per-subagent tool budgets, validate outputs against a defined schema before passing them forward, and establish explicit handoff contracts between stages.
- B. Give each subagent read access to the other subagents' outputs so they can identify and resolve contradictions without routing back through the primary agent.
- C. Add retry logic to each subagent so that when a tool call fails, the subagent retries automatically before escalating – and configure each subagent to log its tool calls and outputs to a shared trace so the team can audit redundancy and contradictions after the fact.
- D. Collapse the three subagents into a single large-context model call that handles retrieval, analysis, and generation in one pass.

Suggested Answer: A

AI-derived guidance: Explicit budgets, output validation, and handoff contracts let the coordinator detect redundancy and conflicting results.

ami007 Most Recent 2 weeks ago

Selected Answer: A

The failures are coordination problems (budgets, validation, contradictions), so the fix is a stronger orchestration layer with per-subagent limits, schema validation, and explicit handoff contracts.

upvoted 1 times

A teammate has asked you to explain when a Skill would be the right choice over an MCP server. The teammate is unsure how the two differ in practice when both can be reused across teams. How would you explain the distinction?

- A. A Skill is preferable for cross-team reuse because it loads more efficiently than an MCP server during normal operation in the team's typical multi-team workloads.
- B. A Skill bundles prompts, scripts, and data into a package the model loads as a unit while a MCP server exposes resources, tools, and prompts through a standard client interface.
- C. A Skill and an MCP server are equivalent extension mechanisms that the team can use interchangeably for any reusable capability that needs to be accessible across teams.
- D. A Skill is the older mechanism and an MCP server is the newer one, so the team should prefer an MCP server for any reusable capability that the team builds going forward.

Suggested Answer: B

AI-derived guidance: Skills package reusable procedural guidance and resources; MCP defines a standard interface for external tools, resources, and prompts.

 **ami007** Most Recent 2 weeks ago

Selected Answer: B

This is the accurate distinction: a Skill packages instructions, scripts, and resources that the model loads, while an MCP server exposes tools, resources, and prompts through a standard client protocol. The other options make false claims (equivalence, "older vs newer", efficiency).
upvoted 1 times

You are designing a Claude application that will process customer support tickets in two stages: a triage stage that classifies tickets and a response stage that drafts replies. The team is debating whether to use a single Claude call that handles both stages or separate Claude calls for each stage. How would you structure the application?

- A. Use separate Claude calls for triage and response, because each stage has distinct inputs, outputs, and success criteria that benefit from focused prompts.
- B. Use a single Claude call for triage and then use a non-Claude rule-based system for response generation, on the grounds that rule-based systems are more reliable for drafting replies.
- C. Use multiple Claude calls in parallel that each draft a complete ticket reply, then have a fourth Claude call select the best one to send to the customer.
- D. Use a single Claude call for both stages, on the grounds that a single call is cheaper than multiple calls in any production Claude application setup.

Suggested Answer: A

AI-derived guidance: Separate stages allow focused prompts and independent classification and response quality checks.

 **ami007** Most Recent 2 weeks ago

Selected Answer: A

Triage and response have different inputs, outputs, and success criteria, so separate focused calls are easier to prompt, test, and debug.
upvoted 1 times

Your Claude application uses structured output that is consumed by downstream code. The team wants to handle malformed or unexpected output gracefully so it does not crash downstream systems. The best choice for handling this issue would be to ...

- A. use the output the model produces and leave schema enforcement to a later phase of production after the application is more complete.
- B. apply defensive parsing to handle unexpected fields, missing values, and type mismatches that the downstream code might otherwise crash on.
- C. crash on any unexpected output and require manual recovery on affected requests the application handles during normal operation.
- D. silently drop any output that does not match expectations so downstream systems do not see the malformed output the application receives from the model.

Suggested Answer: B

AI-derived guidance: Defensive parsing and validation prevent unexpected model output from crashing downstream processing.

ami007 Most Recent 2 weeks ago

Selected Answer: B

Defensive parsing handles missing fields, extra fields, and type mismatches without crashing.

upvoted 1 times

A teammate has asked you to explain why the team's Claude application is billed for output tokens at a different rate than input tokens. They had assumed the rate was the same for both.

How would you explain the difference?

- A. Output tokens are typically billed at a higher rate than input tokens, and cost models for the application should reflect both rates separately.
- B. Output tokens are typically billed at the same rate as input tokens, and the apparent rate difference is a billing error to report to Anthropic.
- C. Output tokens are typically billed at a lower rate than input tokens, because output tokens are cheaper to produce than input tokens are to process.
- D. Output tokens are not billed at all, because cost is determined entirely by the input tokens sent to the model on each request.

Suggested Answer: A

AI-derived guidance: Input and generated output have separate token prices and should be budgeted independently.

Currently there are no comments in this discussion, be the first to comment!