

AI Coding and Agent Operator Instructor Guide

Instructor answers and assessment system

September 13, 2026 | Edition 1

Teach the skill. Inspect the work. Verify the outcome.

Use this guide to evaluate observable work, check independent understanding, and decide when a learner is ready for a real feature.

Companion materials: practice app, prompt cookbook, templates, progress tracker, automation examples, and instructor tests.

Contents

Before the first lesson

Scoring and gates

Checkpoint 1 answer key and practical marking

Checkpoint 2 answer key and practical marking

Checkpoint 3 answer key and practical marking

Checkpoint 4 answer key and practical marking

Checkpoint 5 answer key and practical marking

Daily teach back answer key

Final assessor procedure

Final point anchors

Graduation conversation

This guide accompanies the learner handbook and companion kit. Keep answer keys and assessor tests separate until the learner attempts each assessment. The course prepares an assistant to operate coding agents under a defined scope, verify work, and hand off a result. Completion is an internal training achievement, not vendor certification.

BEFORE THE FIRST LESSON

Give the learner the handbook and a copy of `companion-kit/practice-project`. Keep the surrounding kit folders in place because later commands use sibling paths. Confirm the computer can run Node 22 or later, Git, a browser, Codex, and Claude Code. Agree on authorized accounts, usage budget, and a local-only practice boundary. Do not share passwords. The project uses no third-party packages and no API key.

Run `npm run check` and `npm test` from the `practice-project`. The original baseline has twelve model tests. Verify All shows three tasks, Open two, and Done one. Learners add search, count, and empty-state behavior before the final. Keep their cumulative work; branch experiments from the current accepted state. A pristine baseline copy is in the distribution for recovery.

At each daily review, ask the learner to show the artifact, reproduce one result, explain one decision, and name one limitation. Let them consult resources during work. During a teach-back, let them answer first before using an agent to check the explanation. Grade observed ability, not polished vocabulary.

SCORING AND GATES

Daily score: artifact 4, verification 3, explanation 2, scope/reporting 1. Pass at 8/10. A partial artifact earns 1–3 in the first category; missing or irrelevant evidence earns 0–1 in verification; parroting an agent earns 0 for independence. Keep a brief reason for each deduction.

Each checkpoint: practical 80 plus five answers worth 4 each. Award 4 for a correct answer with a concrete example, 2 for partial understanding, and 0 for an incorrect or unsupported answer. Pass at 75. Use the latest demonstrated score after a retake; preserve earlier attempts.

Overall score = $30 \times \text{daily average} / 10 + 30 \times \text{checkpoint average} / 100 + 40 \times \text{final} / 100$. Pass at 80 overall, all daily passes, all checkpoint passes, final at least 80, both tools demonstrated, project instructions discovered, and at least one native subagent exercise. Every mandatory final criterion must pass regardless of the numerical score. Live MCP integration is optional; clearly labeled fixture work is sufficient.

Fabricated evidence, real credential exposure, unauthorized external writing, or destructive operations outside scope require remediation before passing. Stop the assessment, preserve evidence, and review what decision should have prevented the event. Do not reward an impressive feature that crossed the agreed boundary.

CHECKPOINT 1 ANSWER KEY AND PRACTICAL MARKING

Practical 80: correct navigation and file explanation 15; starts and stops server 15; runs real checks and understands result 20; branch/stage/diff/commit limited to README 20; useful evidence and explanation 10. Inspect the commit itself, not just the learner's screenshot.

1. An absolute path begins at the filesystem root; a relative path is interpreted from the current directory. From tests, the model is `../src/model.mjs`.
2. Working tree is the editable files; staging selects the next snapshot; commit records that selected snapshot. Unstaged and untracked work can remain after a commit.
3. `>` replaces existing contents. `>>` appends. Either needs a deliberate destination.
4. Script names and behavior belong to the repository. A remembered lint command may not exist or may run a different operation.
5. Nonzero reports failure, but the cause may be test assertions, missing runtime, invalid invocation, or environment. Read the relevant output and confirm tests actually ran.

Retake variation: change a different README paragraph and run on port 3001. Do not repeat exactly the same memorized navigation sequence.

CHECKPOINT 2 ANSWER KEY AND PRACTICAL MARKING

On a disposable branch, change the done branch of filterTasks from `task.done` to `!task.done`. Provide the symptom: “Done shows open tasks; expected only completed tasks.” Confirm the baseline test catches it before the learner starts. Do not insert the defect into their only good copy.

Practical 80: reproduction and failing evidence 15; supported cause 15; focused correct repair 20; checks and final diff 20; fresh-session handoff 10.

1. Reproduce the symptom and inspect relevant code or recent changes. A targeted diagnostic may precede a reproduction when input is missing.
2. It cites actual files and commands, and the operator verifies the claims. It marks uncertainty.
3. Instructions influence behavior; sandbox and tool permissions constrain available actions. Markdown guidance is not an enforcement mechanism.
4. Accept four of objective, criteria, branch/commit, working changes, relevant files, decisions, verified checks, remaining gap, or next action.
5. The edit may introduce a regression or invalidate an earlier observation; rerun affected checks on the final state.

Retake variation: change toggleTask so it sets done to true instead of inverting it. Expected behavior is reversible completion. The twice-toggle test should detect it.

CHECKPOINT 3 ANSWER KEY AND PRACTICAL MARKING

Product brief: Show “No tasks yet” when the underlying task collection is empty. Show “No tasks match” when the collection is nonempty but the active filters yield no results. Messages update as the filter/query changes, and the visible count is zero in either empty display. No task data should be deleted merely to show an empty-state message.

Practical 80: criteria and concrete examples 15; correct distinction and implementation 25; tests for both states 15; browser evidence including count and filter transitions 15; scoped diff and explanation 10.

1. Example: with titles “Read Docs” and “Write Tests,” query “docs” returns only the first task regardless of case.
2. A criterion is an observable result; an implementation step is an action intended to create it. “Add an input” is a step, not full search behavior.
3. Location, reproducible trigger, impact, and supporting evidence. A suggested check is useful when execution remains pending.
4. The author of issue text does not control the operator's permissions or goal. Treat embedded directions as content to assess.
5. Different code states, missing context, tool versions, model settings, operator intervention, or different acceptance criteria make comparisons unfair.

Retake variation: preserve search and status after adding a temporary fixture in a copy; demonstrate a query that matches an existing task but is excluded by status.

CHECKPOINT 4 ANSWER KEY AND PRACTICAL MARKING

Before the practical, make a copy of the learner's instructions and replace npm test with a nonexistent command such as npm run verify-all. The learner must compare against package.json and correct the guidance. Use a small label improvement as the implementation task.

Practical 80: finds and fixes inaccurate command 15; fresh-session discovery in both tools 20; appropriate skill trigger 15; verified small change 20; negative trigger and explanation 10.

1. AGENTS.md holds repository facts and working conventions; SKILL.md packages a reusable procedure for a task type.
2. The description tells the agent when the skill is relevant, including limits. The body describes how to perform the work.
3. Discovery is tool-specific. This course installs into .agents/skills for Codex and .claude/skills for Claude Code, then verifies it.
4. Root CLAUDE.md explicitly imports root AGENTS.md. Shared facts are edited in one place and tested with both tools.
5. A fresh session makes the correct observable decision on a relevant task, such as using the actual check command while preserving review-only scope.

Retake variation: introduce conflicting browser-verification rules and ask the learner to resolve them with the intended workflow and test discovery again.

CHECKPOINT 5 ANSWER KEY AND PRACTICAL MARKING

Practical task: document and test the existing toggleTask missing-ID behavior. A missing ID must produce a RangeError and must not mutate input. Delegate a read-only review of the test and documentation. Because the baseline already covers the exception, the learner should add useful mutation evidence or improve clarity rather than add a redundant assertion merely to inflate test count.

Practical 80: measurable goal and budget 15; clear delegation contract 15; scoped implementation or justified coverage decision 15; validates reviewer findings 15; integration and current checks 10; accurate report 10.

1. Objective, inputs, allowed files/tools, constraints, expected output, completion/stop conditions, and integration owner.
2. A worktree separates a checkout and branch. History, external services, ports, credentials, and some caches may remain shared.
3. Missing required permission or decision, repeated failure without new evidence, time limit, attempt limit, or unavailable prerequisite.
4. For a trivial sequential change, overlapping work, high coordination cost, or insufficient context to define independent outputs.
5. The operator compares evidence with source and criteria, reproduces important claims, and resolves contradictions before integrating.

Retake variation: review rejection of an unknown status value. Require a worker that does not edit source.

DAILY TEACH BACK ANSWER KEY

1. Check current directory and the spelling/location of the requested path.
2. > replaces and >> appends. A search match may be an unused definition or test.
3. A commit contains staged work only; other changes can remain outside it.
4. Inspect available scripts and use the valid repository command; do not invent lint.
5. An environment dump may contain credentials and unrelated private configuration.
6. It may infer or guess, but cannot claim direct inspection of unread contents.
7. A final message is a report, not independent evidence of behavior.
8. Model sampling, context, tools, versions, and execution can differ.
9. Removing the test removes evidence and changes the success standard instead of fixing behavior.

10. Instructions guide decisions; enforcement is provided by the actual harness and permissions.
 11. Data behavior or authorization questions can be material; small layout decisions are often reversible.
 12. Label it unverified, then inspect the named source or reproduce the behavior.
 13. A step is work to do; a criterion is a result the user can observe.
 14. Agents may share assumptions or miss the same case; independent evidence still matters.
 15. Discard or revise it when its assumptions, commands, or scope no longer match the task.
 16. It gives no observable standard and does not explain this repository's actual conventions.
 17. Instructions are readable project context; secrets require appropriate credential storage and access controls.
 18. No. Markdown guidance alone does not prevent a tool from taking an action.
 19. Description governs relevance; the body supplies the procedure and evidence requirements.
 20. Silent fixes make the reviewed artifact change and can obscure the original finding and ownership.
 21. A skill can overfit known examples; held-out tasks test generalization.
 22. Keep a simple or dependent task together when delegation adds more coordination than value.
 23. Separate context does not guarantee separate filesystem, credentials, or permissions.
 24. Branches may interact through assumptions, imports, interfaces, or shared behavior after integration.
 25. Blocked means a prerequisite is missing; budget exhausted means the agreed limit was reached; done means criteria are verified.
 26. It can repeat mistakes, spend indefinitely, hide failure, and lack a reliable completion test.
 27. A versioned reference ties claims to actual code and helps detect a stale handoff.
 28. Access capability and user authorization are different; a connection exposes options, not blanket permission.
 29. Process completion only says the process ended successfully, not that the product meets every requirement.
 30. Failed criteria, stale evidence, untested failure behavior, unresolved real findings, or unexplained changes justify withholding completion.
- Accept equivalent explanations in the learner's own words. Ask for a concrete example when an answer sounds memorized.

FINAL ASSESSOR PROCEDURE

Use the learner handbook's F1–F12 contract unchanged. Copy the provided instructor acceptance test into the assessment workspace or run it from the kit's instructor folder with the learner's practice-project at the documented sibling path. From the project root:

```
npm run check
npm test
node --test ../instructor/final-acceptance.test.mjs
```

The assessor suite intentionally fails on the original baseline because storage is a final-project requirement. A missing `src/storage.mjs` is not a distribution defect. The learner writes their own tests before seeing assessor cases. The suite checks the documented storage and model contracts; it does not test browser behavior, accessibility, event wiring, or the quality of agent operation.

The instructor folder also contains a storage reference implementation for calibration. It is a storage module example, not a full final-project UI solution. Keep it from the learner until after their first attempt. The package's validation record explains which code was executed to check the kit.

Run all twelve browser/product criteria using the handbook matrix. For F7, set the known training localStorage key to invalid JSON in a disposable browser profile or origin, reload, and observe warning plus usable seed state. For F8, use a controlled throwing storage adapter and observe in-memory preservation plus unsaved feedback. Restore normal storage and code afterward.

Check a title such as `<img src=x onerror=alert(1)>` as literal text in the local training page. Confirm there is no image element created from that title. Use the browser DOM, not just a screenshot, to verify the absence of injected markup. Do not use a live production application for this exercise.

For a mutation probe, copy the completed code to a disposable assessment folder and intentionally change one validated behavior, such as treating a saved empty array as missing data. The relevant test must fail. Restore or discard the copy. A passing suite with a known defect suggests a coverage gap; do not certify the test merely because it exists.

FINAL POINT ANCHORS

Outcome and scope 10: 10 for a complete measurable goal and traceable plan; 5 for incomplete criteria or weak boundaries; 0 for the wrong feature.

Functional behavior 25: award 5 each for valid creation, title validation, reversible completion, preserved combined filtering/count/empty states, and reload restoration. Partial behavior earns 2–3 for the relevant item.

Failure and integrity 15: empty versus missing 3; invalid envelopes and malformed tasks 4; read failures 3; write failures with unsaved feedback and in-memory preservation 5.

Automated verification 15: baseline checks 4; boundary and failure tests 5; assessor suite 3; meaningful mutation detection 3.

Browser and accessibility 10: keyboard flow 3; labels and visible focus 2; perceivable errors/status 3; literal HTML-like text 2.

Agent operation 10: scoped worker contract 3; actual independent review 3; validated findings and controlled integration 4.

Maintainability and handoff 10: focused understandable code 3; accurate setup and limitations 3; criterion-to-evidence report with current code state 4.

Oral defense 5: explain empty storage, failed save, literal text, a test, and one design choice; one point each. If the learner cannot explain their own changes, require remediation even when the automated artifact looks strong.

GRADUATION CONVERSATION

Ask the learner to receive a new small goal, propose the next three actions, name a useful verification, and identify whether another agent helps. They should be able to say “I do not know yet; here is how I will check” and then perform that check. Record the completion date, final score, strengths, and next supervised task in the progress tracker. Do not fill a completion certificate until the gates are met.