Python Applications

Ages 16-17 | Future Pros

Build and test a small automation tool using sample data.

How to use this guide

Read each section, complete its four lab steps, then answer the questions in the online course. The online progress bar advances only after four correct answers.

Section 1: APIs

LEARNING GOAL: APIs. Your finished work should demonstrate: A script that handles sample API-shaped data.

EXPLAIN THE CONCEPT: An API lets one program request information from another. Start with a public sample response and handle missing fields safely. Think about where you might use this skill in a real or imaginary project. Describe the problem it helps you solve and how you would recognize a successful result.

ORIGINAL PROJECT LAB: 1. Create a sample JSON object with two fields. 2. Write Python that reads its values. 3. Display a helpful message if one field is missing. 4. Test with both complete and incomplete sample data. Finished result: A script that handles sample API-shaped data.

Understand with an example

The idea. An API contract describes a request, a response and failure behavior. A successful status does not by itself establish that the response contains usable application data. Validate the fields needed by the task before displaying a result.

Get ready. Use a fictional GET /items/7 response: status 200, body containing id 7 and title Atlas. No network request is made.

Worked example 1

Follow the reasoning. The caller first checks the status, then confirms that title is a nonempty string. It can display Atlas only after both checks pass. Keep response validation separate from the act of sending a request.

Worked example 2: change one thing

Compare the result. Keep status 200 but remove title. The caller should show a controlled unavailable-data message rather than assume that status 200 guarantees

the field.

Try one action at a time

1. Write the response contract.
2. Check the status.
3. Validate required fields.
4. Predict a controlled failure response.

A mistake to notice

Pause and check. Do not put credentials or private data into a practice request or screenshot.

Your own small challenge

Predict before trying. A fictional response has status 404 and an error message. Should the caller display it as an item title?

Check your thinking

No. Handle the unsuccessful response through the failure path and avoid treating the error body as the normal item schema.

Show what you know. Point, draw, demonstrate, type, speak or use your usual communication tool. A trusted adult can help without choosing your answer.

GO DEEPER: Identify an assumption, test a safe alternative, and explain the trade-off. Keep a short before-and-after record that someone else could follow.

SAFETY CHECK: Apply identity security, social-engineering defense, secure coding, cloud controls, incident handling, data protection, and professional ethics. Use pretend data, never real passwords or private information. Stop and ask a trusted adult if anything seems unsafe.

REVIEW QUESTIONS: What is the main idea? Why does the first step matter? How would you detect a mistake? What evidence proves the result? What would you improve? Answer in your own words before taking the quiz.

PORTFOLIO REFLECTION: Save a drawing, sanitized screenshot, or short note describing the goal, your approach, your evidence, and one lesson learned. Expected result: A script that handles sample API-shaped data.

Investigate with pictures and choices

An API request times out after sending a write.

What remains uncertain?

1. Whether the write committed
2. That nothing happened
3. That another write is always safe

Worked reasoning

Answer: Whether the write committed. A lost response does not establish that the remote operation failed.

Try a changed example

Design a lookup or idempotency contract before retrying the write.

Picture and meaning matches

API contract: Defines request and response

Test: Compares actual and expected

Version control: Records reviewed changes

Ways to respond

Choose, point, draw, demonstrate, type, speak or use your usual communication tool.

A helper can operate the controls while you choose.

Counting support if useful

Use zero to ten counters or dots. Touch each once. The last count names the quantity. Rearrange the same set and check whether anything was added or removed.

Choose this foundation practice only when it fits the learner goal.

My notes: _____

Hands-on lab

1. Create a sample JSON object with two fields.
2. Write Python that reads its values.
3. Display a helpful message if one field is missing.
4. Test with both complete and incomplete sample data.

Finished result: A script that handles sample API-shaped data.

Check your understanding

1. Which idea best describes APIs?

A. A test checks expected behavior for a known input.

B. An API lets one program request information from another.

C. Git records changes so you can review how a project evolved.

D. It has no purpose in this project.

2. What is the first step in this lab?

A. Write Python that reads its values.

B. Display a helpful message if one field is missing.

C. Create a sample JSON object with two fields.

D. Test with both complete and incomplete sample data.

3. What should you do after step two?

A. Write Python that reads its values.

B. Test with both complete and incomplete sample data.

C. Create a sample JSON object with two fields.

D. Display a helpful message if one field is missing.

4. Which result shows that your lab is finished?

A. A script that handles sample API-shaped data.

B. A copied answer without checking it

C. A skipped lab

D. A blank page with no work

Answer review

1. B. An API lets one program request information from another.

2. C. Start with: Create a sample JSON object with two fields.

3. D. Step three is: Display a helpful message if one field is missing.

4. A. The goal is: A script that handles sample API-shaped data.

Section 2: Testing

LEARNING GOAL: Testing. Your finished work should demonstrate: Two documented checks of your script.

EXPLAIN THE CONCEPT: A test checks expected behavior for a known input. Include a normal case and an edge case. Think about where you might use this skill in a real or imaginary project. Describe the problem it helps you solve and how you would recognize a successful result.

ORIGINAL PROJECT LAB: 1. Write down one expected result for complete sample data. 2. Write another for a missing field. 3. Run the script with both inputs. 4. Fix the code until both results match the plan. Finished result: Two documented checks of your script.

Understand with an example

The idea. A useful test compares observed behavior with an independently specified

expectation. Cover boundaries and invalid inputs, not only the values that are easy to make pass. A test can be wrong when its expectation repeats the same mistaken assumption as the code.

Get ready. A function accepts whole-number quantities 1 through 5 inclusive.

Worked example 1

Follow the reasoning. Choose 0,1,5,6. Expected acceptance is no,yes,yes,no. These cases expose both ends of the allowed range. Add a nonnumeric case if the caller may supply one.

Worked example 2: change one thing

Compare the result. An implementation accepts quantity<5 rather than quantity<=5. The test for 5 fails even though 1 succeeds; this is evidence of a boundary defect.

Try one action at a time

1. State the allowed input range.
2. Choose boundary and invalid cases.
3. Write expected results first.
4. Compare actual evidence when code is run.

A mistake to notice

Pause and check. A green test suite proves only the checked expectations under the tested conditions.

Your own small challenge

Predict before trying. Would tests for 2 and 3 expose the upper-bound defect?

Check your thinking

No. Both satisfy either expression. Include 5 to distinguish the intended rule from the faulty one.

Show what you know. Point, draw, demonstrate, type, speak or use your usual communication tool. A trusted adult can help without choosing your answer.

GO DEEPER: Identify an assumption, test a safe alternative, and explain the trade-off. Keep a short before-and-after record that someone else could follow.

SAFETY CHECK: Apply identity security, social-engineering defense, secure coding, cloud controls, incident handling, data protection, and professional ethics. Use pretend data, never real passwords or private information. Stop and ask a trusted

adult if anything seems unsafe.

REVIEW QUESTIONS: What is the main idea? Why does the first step matter? How would you detect a mistake? What evidence proves the result? What would you improve? Answer in your own words before taking the quiz.

PORTFOLIO REFLECTION: Save a drawing, sanitized screenshot, or short note describing the goal, your approach, your evidence, and one lesson learned. Expected result: Two documented checks of your script.

Investigate with pictures and choices

A function accepts empty input under its contract.

What should a test include?

1. Only the largest normal example
2. A screenshot with no assertion
3. The expected empty-input behavior

Worked reasoning

Answer: The expected empty-input behavior. Test normal and boundary cases against the declared behavior.

Try a changed example

Write expected output before executing the test.

Picture and meaning matches

API contract: Defines request and response

Test: Compares actual and expected

Version control: Records reviewed changes

Ways to respond

Choose, point, draw, demonstrate, type, speak or use your usual communication tool. A helper can operate the controls while you choose.

Counting support if useful

Use zero to ten counters or dots. Touch each once. The last count names the quantity. Rearrange the same set and check whether anything was added or removed. Choose this foundation practice only when it fits the learner goal.

My notes: _____

Hands-on lab

1. Write down one expected result for complete sample data.
2. Write another for a missing field.
3. Run the script with both inputs.
4. Fix the code until both results match the plan.

Finished result: Two documented checks of your script.

Check your understanding

1. Which idea best describes Testing?

A. It has no purpose in this project.

B. An API lets one program request information from another.

C. A test checks expected behavior for a known input.

D. Git records changes so you can review how a project evolved.

2. What is the first step in this lab?

A. Fix the code until both results match the plan.

B. Write another for a missing field.

C. Run the script with both inputs.

D. Write down one expected result for complete sample data.

3. What should you do after step two?

A. Run the script with both inputs.

B. Write another for a missing field.

C. Fix the code until both results match the plan.

D. Write down one expected result for complete sample data.

4. Which result shows that your lab is finished?

A. A blank page with no work

B. Two documented checks of your script.

C. A copied answer without checking it

D. A skipped lab

Answer review

1. C. A test checks expected behavior for a known input.
2. D. Start with: Write down one expected result for complete sample data.
3. A. Step three is: Run the script with both inputs.
4. B. The goal is: Two documented checks of your script.

Section 3: Git

LEARNING GOAL: Git. Your finished work should demonstrate: A local project with a clean first commit.

EXPLAIN THE CONCEPT: Git records changes so you can review how a project evolved. Commit small, understandable updates without secrets. Think about where you might use this skill in a real or imaginary project. Describe the problem it helps you solve and how you would recognize a successful result.
ORIGINAL PROJECT LAB: 1. Initialize a local Git repository for the sample script. 2. Add a README describing its purpose. 3. Check that no passwords or keys are in the files. 4. Commit the tested script with a clear message. Finished result: A local project with a clean first commit.

Understand with an example

The idea. Version control records snapshots of tracked work. A branch names a line of development; a commit documents a particular change. Review the diff before recording a snapshot so unrelated or sensitive files do not enter the history.
Get ready. Imagine a repository with README and scorer.py. A task changes only the score threshold.

Worked example 1

Follow the reasoning. Review the diff in scorer.py, verify the expected boundary behavior and commit that focused change with a factual message. The README stays unchanged because this task did not need it.

Worked example 2: change one thing

Compare the result. A local notes file contains a pretend credential. Exclude it from the commit and replace sensitive examples with placeholders. Ignoring a file later would not remove an already recorded secret from history.

Try one action at a time

1. Inspect changed files.
2. Review the intended diff.
3. Verify behavior and exclude secrets.
4. Record a focused commit.

A mistake to notice

Pause and check. A commit records content; it does not automatically verify that content.

Your own small challenge

Predict before trying. Two branches change the same threshold differently. What must a conflict resolution decide?

Check your thinking

Choose the intended requirement, reconcile the code and rerun relevant boundary checks. Merely removing conflict markers does not establish correctness.
Show what you know. Point, draw, demonstrate, type, speak or use your usual communication tool. A trusted adult can help without choosing your answer.
GO DEEPER: Identify an assumption, test a safe alternative, and explain the trade-off. Keep a short before-and-after record that someone else could follow.
SAFETY CHECK: Apply identity security, social-engineering defense, secure coding, cloud controls, incident handling, data protection, and professional ethics. Use pretend data, never real passwords or private information. Stop and ask a trusted adult if anything seems unsafe.
REVIEW QUESTIONS: What is the main idea? Why does the first step matter? How would you detect a mistake? What evidence proves the result? What would you improve? Answer in your own words before taking the quiz.
PORTFOLIO REFLECTION: Save a drawing, sanitized screenshot, or short note describing the goal, your approach, your evidence, and one lesson learned. Expected result: A local project with a clean first commit.

Investigate with pictures and choices

A branch contains a secret accidentally.
What should the response include?
1. Post the secret in a bug report
2. Follow a trusted rotation and removal process
3. Only rename the file

Worked reasoning

Answer: Follow a trusted rotation and removal process. Deleting a visible string alone does not invalidate an exposed credential.

Try a changed example

Use a synthetic placeholder to rehearse an incident handoff.

Picture and meaning matches

API contract: Defines request and response

Test: Compares actual and expected

Version control: Records reviewed changes

Ways to respond

Choose, point, draw, demonstrate, type, speak or use your usual communication tool.

A helper can operate the controls while you choose.

Counting support if useful

Use zero to ten counters or dots. Touch each once. The last count names the quantity. Rearrange the same set and check whether anything was added or removed.

Choose this foundation practice only when it fits the learner goal.

My notes: _____

Hands-on lab

1. Initialize a local Git repository for the sample script.
2. Add a README describing its purpose.
3. Check that no passwords or keys are in the files.
4. Commit the tested script with a clear message.

Finished result: A local project with a clean first commit.

Check your understanding

1. Which idea best describes Git?

A. It has no purpose in this project.

B. An API lets one program request information from another.

C. Git records changes so you can review how a project evolved.

D. A test checks expected behavior for a known input.

2. What is the first step in this lab?

A. Commit the tested script with a clear message.

B. Add a README describing its purpose.

C. Check that no passwords or keys are in the files.

D. Initialize a local Git repository for the sample script.

3. What should you do after step two?

A. Check that no passwords or keys are in the files.

B. Add a README describing its purpose.

C. Commit the tested script with a clear message.

D. Initialize a local Git repository for the sample script.

4. Which result shows that your lab is finished?

A. A blank page with no work

B. A local project with a clean first commit.

C. A copied answer without checking it

D. A skipped lab

Answer review

1. C. Git records changes so you can review how a project evolved.
2. D. Start with: Initialize a local Git repository for the sample script.
3. A. Step three is: Check that no passwords or keys are in the files.
4. B. The goal is: A local project with a clean first commit.

Section 4: Guided Project & Worked Examples

COURSE INVESTIGATION: A retry must respect completed work

YOUR PRACTICE INPUT: A synthetic file-processing job reads records A, B, and C. It saves A, then fails before confirming B.

WHY IT WORKS: Restarting from the beginning without an identity or checkpoint rule may duplicate A. A processed-record key should describe the business item, not the attempt.

PLAN BEFORE YOU ACT: Draw or write the starting state, the question you need to answer, and the evidence you expect to see. A paper trace is acceptable; label it as a prediction unless you actually run a test.

WORK THE CASE: Design a local dry-run ledger with record ID, content digest concept, status, and error. Trace first attempt, retry, and conflict.

COMPARE WITH THE EXPECTED RESULT: A is not silently processed twice; the B outcome is reconciled; conflicting A content is surfaced rather than overwritten.

EXPLAIN YOUR REASONING: Show the connection between the supplied input, each decision, and your result. If a number, label, or observation differs, return to the first differing step rather than changing several things together.

KEEP YOUR EVIDENCE: Save the initial plan, a drawing or sanitized output, and one explanation of what the result does and does not prove. Use pointing, speech, drawing, typing, or your usual communication method.

LEARNING GOAL: Guided Project & Worked Examples. Your finished work should demonstrate: A completed guided project with documented evidence: Repository/source, README, Test evidence, Security/configuration note.

EXPLAIN THE CONCEPT: Review the course concepts through two applied examples. Refactor a small script into functions with clear inputs, outputs, and error

handling. Compare a hard-coded secret with environment-based configuration and explain why source control must not contain credentials. Guided project: Build an automation utility with arguments/input validation, functions, logging, error handling, tests, and documentation. Work with synthetic or fictional information and ask an adult for help when appropriate. Think about where you might use this skill in a real or imaginary project. Describe the problem it helps you solve and how you would recognize a successful result.
ORIGINAL PROJECT LAB: 1. Read both worked examples and explain the reasoning. 2. Build an automation utility with arguments/input validation, functions, logging, error handling, tests, and documentation. 3. Record the steps you completed and the result you observed. 4. Compare your result to the goal and correct one issue.
Finished result: A completed guided project with documented evidence: Repository/source, README, Test evidence, Security/configuration note.

Understand with an example

The idea. A reliable API client project joins contract validation, explicit failure behavior and repeatable tests. Separate a simulated fixture from a real service response when describing evidence.
Get ready. Plan an item viewer using fictional fixtures: valid 200/title, 200/missing title and 404/error.

Worked example 1

Follow the reasoning. Expected outputs are the item title, a data-unavailable message and a not-found message respectively. A test should assert these distinct outcomes rather than accept any nonempty screen text.

Worked example 2: change one thing

Compare the result. Change the valid fixture title from Atlas to Orbit. Only the displayed title expectation changes; malformed and not-found behavior should remain fixed.

Try one action at a time

1. Define request and response assumptions.
2. Prepare three distinct fixtures.
3. Predict visible outcomes.
4. Compare changed content and stable failures.

A mistake to notice

Pause and check. Do not report fixture checks as proof that a live external API was reached.

Your own small challenge

Predict before trying. If the service times out, is it equivalent to an item not existing?

Check your thinking

No. A timeout leaves the item’s existence unresolved. Use a controlled temporary-failure path rather than inventing a not-found result.
Show what you know. Point, draw, demonstrate, type, speak or use your usual communication tool. A trusted adult can help without choosing your answer.
GO DEEPER: Identify an assumption, test a safe alternative, and explain the trade-off. Keep a short before-and-after record that someone else could follow.
SAFETY CHECK: Apply identity security, social-engineering defense, secure coding, cloud controls, incident handling, data protection, and professional ethics. Use pretend data, never real passwords or private information. Stop and ask a trusted adult if anything seems unsafe.
REVIEW QUESTIONS: What is the main idea? Why does the first step matter? How would you detect a mistake? What evidence proves the result? What would you improve? Answer in your own words before taking the quiz.
PORTFOLIO REFLECTION: Save a drawing, sanitized screenshot, or short note describing the goal, your approach, your evidence, and one lesson learned. Expected result: A completed guided project with documented evidence: Repository/source, README, Test evidence, Security/configuration note.

Investigate with pictures and choices

An automation acts on every matching file.
What helps a safe first run?
1. A bounded dry run showing intended changes
2. An unlimited run with no output
3. Removing every check

Worked reasoning

Answer: A bounded dry run showing intended changes. Preview the scope and verify the intended operation before effects.

Try a changed example

Test a no-match and wrong-path case using a temporary synthetic folder.

Picture and meaning matches

API contract: Defines request and response

Test: Compares actual and expected

Version control: Records reviewed changes

Ways to respond

Choose, point, draw, demonstrate, type, speak or use your usual communication tool.

A helper can operate the controls while you choose.

Counting support if useful

Use zero to ten counters or dots. Touch each once. The last count names the quantity. Rearrange the same set and check whether anything was added or removed.

Choose this foundation practice only when it fits the learner goal.

My notes: _____

Hands-on lab

1. Read both worked examples and explain the reasoning.
2. Build an automation utility with arguments/input validation, functions, logging, error handling, tests, and documentation.
3. Record the steps you completed and the result you observed.
4. Compare your result to the goal and correct one issue.

Finished result: A completed guided project with documented evidence: Repository/source, README, Test evidence, Security/configuration note.

Check your understanding

1. Which idea best describes Guided Project & Worked Examples?

A. An API lets one program request information from another.

B. Review the course concepts through two applied examples.

C. A test checks expected behavior for a known input.

D. It has no purpose in this project.

2. What is the first step in this lab?

A. Build an automation utility with arguments/input validation, functions, logging,

error handling, tests, and documentation.

B. Record the steps you completed and the result you observed.

C. Read both worked examples and explain the reasoning.

D. Compare your result to the goal and correct one issue.

3. What should you do after step two?

A. Build an automation utility with arguments/input validation, functions, logging,

error handling, tests, and documentation.

B. Compare your result to the goal and correct one issue.

C. Read both worked examples and explain the reasoning.

D. Record the steps you completed and the result you observed.

4. Which result shows that your lab is finished?

A. A completed guided project with documented evidence: Repository/source, README,

Test evidence, Security/configuration note.

B. A copied answer without checking it

C. A skipped lab

D. A blank page with no work

Answer review

1. B. Review the course concepts through two applied examples.

2. C. Start with: Read both worked examples and explain the reasoning.

3. D. Step three is: Record the steps you completed and the result you observed.

4. A. The goal is: A completed guided project with documented evidence:

Repository/source, README, Test evidence, Security/configuration note.

Section 5: Independent Challenge & Mastery

INDEPENDENT TRANSFER CASE: A retry must respect completed work

KEEP THE ORIGINAL BASELINE: A synthetic file-processing job reads records A, B, and C. It saves A, then fails before confirming B.

CHANGE ONE CONDITION: The input contains A again with different content. A duplicate key with conflicting content needs an explicit rejection or version rule.

YOUR TASK: Design a local dry-run ledger with record ID, content digest concept, status, and error. Trace first attempt, retry, and conflict.

PREDICT, THEN CHECK: State which part of your original reasoning should remain true and which part must be recalculated or retested. Keep the original and changed results side by side.

REVIEW CRITERIA: A is not silently processed twice; the B outcome is reconciled; conflicting A content is surfaced rather than overwritten.

MENTOR DISCUSSION: Explain why the changed condition matters. Identify a tempting

wrong conclusion, show the observation that would challenge it, and choose one next test. A statement of uncertainty is better than inventing a result.
PORTFOLIO NOTE: Keep your baseline, changed input, prediction, observed or paper-traced result, explanation, and one improvement. These practice prompts do not replace the course's existing quiz or final assessment.
LEARNING GOAL: Independent Challenge & Mastery. Your finished work should demonstrate: An independent demonstration with evidence: Repository/source, README, Test evidence, Security/configuration note.
EXPLAIN THE CONCEPT: Apply what you learned without copying the guided solution. Independent challenge: Integrate a safe mock/public API response, handle timeout/error/invalid data cases, and document the contract. Safety and responsibility: Use approved APIs and synthetic data; keep secrets server-side or in environment configuration; never automate unauthorized access or scraping. Reflect on your work: What did you build or discover? What did you test or change after feedback? What safe or responsible choice did you make? Think about where you might use this skill in a real or imaginary project. Describe the problem it helps you solve and how you would recognize a successful result.
ORIGINAL PROJECT LAB: 1. Explain the challenge and make a short plan. 2. Integrate a safe mock/public API response, handle timeout/error/invalid data cases, and document the contract. 3. Test the result and record evidence of your work. 4. Review the safety scenario and describe one improvement. Finished result: An independent demonstration with evidence: Repository/source, README, Test evidence, Security/configuration note.

Understand with an example

The idea. A handoff should explain input rules, error cases, tests and evidence boundaries. Another learner needs enough information to reproduce the behavior without access to your private accounts.
Get ready. Design a fictional catalog client with one valid response and two failures.

Worked example 1

Follow the reasoning. One plan requires id and name. A valid record displays its name; a missing name produces controlled feedback; an unsuccessful status does not enter the normal renderer.

Worked example 2: change one thing

Compare the result. Add a duplicate record id to a list fixture. Define whether the contract rejects it or the client flags it; do not silently assume every list entry has a unique identity.

Try one action at a time

1. Write the contract.
2. Predict success and failures.
3. Document a changed fixture.
4. Present sanitized evidence and limitations.

A mistake to notice

Pause and check. A detailed plan does not justify claiming a completed live integration.

Your own small challenge

Predict before trying. Provide a README test table with input, expected result and observed result columns.

Check your thinking

Keep observed results blank or marked not run until execution occurs. Expected results alone are design evidence.

Show what you know. Point, draw, demonstrate, type, speak or use your usual communication tool. A trusted adult can help without choosing your answer.

GO DEEPER: Identify an assumption, test a safe alternative, and explain the trade-off. Keep a short before-and-after record that someone else could follow.

SAFETY CHECK: Apply identity security, social-engineering defense, secure coding, cloud controls, incident handling, data protection, and professional ethics. Use pretend data, never real passwords or private information. Stop and ask a trusted adult if anything seems unsafe.

REVIEW QUESTIONS: What is the main idea? Why does the first step matter? How would you detect a mistake? What evidence proves the result? What would you improve? Answer in your own words before taking the quiz.

PORTFOLIO REFLECTION: Save a drawing, sanitized screenshot, or short note describing the goal, your approach, your evidence, and one lesson learned. Expected result: An

independent demonstration with evidence: Repository/source, README, Test evidence, Security/configuration note.

Investigate with pictures and choices

A reviewer needs to reproduce a bug fix.

What should you supply?

1. Only the word fixed
2. A renamed branch
3. Inputs, expected result and test output

Worked reasoning

Answer: Inputs, expected result and test output. A reviewable change connects its requirement with reproducible evidence.

Try a changed example

Run the same case from a documented clean baseline.

Picture and meaning matches

API contract: Defines request and response

Test: Compares actual and expected

Version control: Records reviewed changes

Ways to respond

Choose, point, draw, demonstrate, type, speak or use your usual communication tool. A helper can operate the controls while you choose.

Counting support if useful

Use zero to ten counters or dots. Touch each once. The last count names the quantity. Rearrange the same set and check whether anything was added or removed. Choose this foundation practice only when it fits the learner goal.

My notes: _____

Hands-on lab

1. Explain the challenge and make a short plan.
2. Integrate a safe mock/public API response, handle timeout/error/invalid data cases, and document the contract.
3. Test the result and record evidence of your work.

4. Review the safety scenario and describe one improvement.

Finished result: An independent demonstration with evidence: Repository/source, README, Test evidence, Security/configuration note.

Check your understanding

1. Which idea best describes Independent Challenge & Mastery?

A. It has no purpose in this project.

B. An API lets one program request information from another.

C. Apply what you learned without copying the guided solution.

D. A test checks expected behavior for a known input.

2. What is the first step in this lab?

A. Review the safety scenario and describe one improvement.

B. Integrate a safe mock/public API response, handle timeout/error/invalid data cases, and document the contract.

C. Test the result and record evidence of your work.

D. Explain the challenge and make a short plan.

3. What should you do after step two?

A. Test the result and record evidence of your work.

B. Integrate a safe mock/public API response, handle timeout/error/invalid data cases, and document the contract.

C. Review the safety scenario and describe one improvement.

D. Explain the challenge and make a short plan.

4. Which result shows that your lab is finished?

A. A blank page with no work

B. An independent demonstration with evidence: Repository/source, README, Test evidence, Security/configuration note.

C. A copied answer without checking it

D. A skipped lab

Answer review

1. C. Apply what you learned without copying the guided solution.

2. D. Start with: Explain the challenge and make a short plan.

3. A. Step three is: Test the result and record evidence of your work.

4. B. The goal is: An independent demonstration with evidence: Repository/source, README, Test evidence, Security/configuration note.

Mastery & safe-use plan

Ages 16-17

Digital safety: Apply identity security, social-engineering defense, secure coding,

cloud controls, incident handling, data protection, and professional ethics.
AI literacy: Use AI as an accountable engineering assistant: validate outputs, protect data, document use, test security, and keep humans responsible.
Challenge: Work through a realistic ticket, incident, architecture, data, or application scenario with professional evidence.
Portfolio: Produce a career-ready artifact such as a repository, incident report, architecture diagram, dashboard, automation, or technical brief.
Mastery check: Present the work as if to a mentor or interviewer: problem, approach, evidence, tradeoffs, security, and next improvement.

Project worksheet

My project: Build an automation tool
Who will use or review it? _____
My plan: _____
Checkpoint 1: A script that handles sample API-shaped data.
Checkpoint 2: Two documented checks of your script.
Checkpoint 3: A local project with a clean first commit.
Checkpoint 4: A completed guided project with documented evidence: Repository/source, README, Test evidence, Security/configuration note.
Checkpoint 5: An independent demonstration with evidence: Repository/source, README, Test evidence, Security/configuration note.
One thing I tested: _____
One thing I changed after testing: _____
What I learned: _____
Trusted adult or mentor feedback: _____

Family review

Show your finished work to a trusted adult. Explain one thing you learned and one thing you would improve.

Beat IT Kids | For learning at home. Do not put real passwords or private details in practice projects.