LESSON 46

Strand 3 challenge build

40 minutes | Strand 3: Coding at the next level | Outcome 3.1 | Outcome 3.6 | Outcome 3.9 | Outcome 3.8 | Outcome 3.5 | Outcome 3.10

WHAT THIS LESSON DOES

Plan, build, document and test a small Python program solving a problem you choose. Have it tested by a classmate from outside your group and make one improvement based on their feedback. Note any ethical or legal issues.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
3.1	Creatively design and write code for short programming tasks, demonstrating operators for assignment, arithmetic, comparison and Boolean combinations.
3.6	Document programs so others can understand them.
3.9	Analyse code to determine its function and find errors.
3.8	Test their work with external users.
3.5	Investigate any ethical or legal issues their project raises.
3.10	Reflect on how their learning and problem-solving developed.

WHAT STUDENTS SHOULD BE ABLE TO DO

- Plan, build, document and test a small Python program that solves a problem you choose (LOs 3.1, 3.6, 3.9)
- Test your program with users beyond your own group and use what you observe to improve it (LO 3.8)
- Investigate any ethical or legal issues your program raises, and reflect on how your problem-solving has changed (LOs 3.5, 3.10)

BEFORE THE BELL

- Have paper available for students who prefer to sketch initial plans by hand.
- Agree and post the class save or export method before the period starts.
- Decide cross-group user-test pairs in advance so you can post them on the board before the build freeze.
- Optional: pre-seed a starter file with the three scaffold comment headings for support students.
- Student devices with the class Python editor ready; saving method agreed
- Cross-group pairs already posted from end of build step

Strand 3 challenge build

HOW THE LESSON RUNS

4 min	introduction	Start
3 min	content	Pick a small problem
3 min	content	What your finished build must do
4 min	content	Hit Must before the freeze
9 min	content	How to get there
3 min	content	After the freeze
5 min	content	Test with someone who did not build it
3 min	content	What done looks like
3 min	content	Make sense
2 min	content	Reflect
1 min	summary	What you covered

TEACHING MOVES

- Circulate during idea selection to ensure projects remain small in scope.
- Keep Must as the only checklist until the freeze; reveal After freeze only after features stop so the core path is not competing with polish.
- State clearly that Must alone is enough to go to the swap.
- Call the mid-build checkpoint so stuck students move to the three-step scaffold before polish time is lost.
- Post cross-group pairs before the build step ends so the swap does not burn minutes.
- Call the user-test timeboxes so the improve-and-retest half is not skipped.
- Keep the final reflection short with only two or three student contributions.

WHAT TO WATCH FOR

- Selecting overly complex problems: redirect to a single input-process-output task.
- Explaining the program to the tester: remind builders to observe silently.
- Skipping comments until the end: prompt students to add them as they code.

DIFFERENTIATION

- Support: offer sentence starters such as: "Input I need: … / Decision the program makes: … / Output it should show: … / One awkward input to try: …"
- Support: point to the optional three-step scaffold (one input, if/else or list, print a clear result) and allow planning on paper so everyone has a runnable path by the freeze.
- Support: pre-seed a starter file with the three comment headings already pasted.
- Extension: challenge students to incorporate lists or additional decisions.