LESSON 10

Python Challenge Build

40 minutes | Build Something in Python

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.

WHAT STUDENTS SHOULD BE ABLE TO DO

- Plan, build, document and test a Python program to solve a chosen problem.
- Conduct user testing with peers outside usual groups and refine the program accordingly.
- Identify ethical or legal issues and reflect on personal growth in computational problem-solving.

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.

Python 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.