LESSON 7

Pseudo-code and Flow Charts: Plan It, Then Build It

40 minutes | Think Like a Programmer

WHAT THIS LESSON DOES

Write a quiz algorithm as pseudo-code and as a flow chart, build and match-check it in Scratch, then extend the plan, swap with a classmate, and flag where the wording needs to be more precise.

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand how to represent algorithms in pseudo-code.
- Create flow charts to visually depict algorithmic decisions.
- Implement planned algorithms by building programs in Scratch.
- Evaluate the precision of algorithms by flagging unclear steps in a peer plan.

BEFORE THE BELL

- Prepare paper or notebooks for students to write pseudo-code and draw flow charts.
- Have space on the board ready for demonstrating the sample pseudo-code and drawing the matching shapes plan live.
- Pre-assign pairs for the swap step so logistics do not eat minutes.
- Be ready to show the class Scratch save routine live near the end of step 5 (for example File, Save to your computer as QuizPlan_Name in the class folder).

Pseudo-code and Flow Charts: Plan It, Then Build It

HOW THE LESSON RUNS

4 min	introduction	Start
2 min	content	Two Ways to Write a Plan
3 min	content	The Quiz You Will Build, as Pseudo-code
5 min	content	Write the Pseudo-code and Draw the Flow Chart
2 min	content	Get Something Running Early
6 min	content	Build the Decision Path and Run Both Paths
3 min	content	Check the Plan Against the Blocks
8 min	content	Extend Your Plan, Swap It, and Flag One Unclear Step
1 min	content	Save Your Project
3 min	content	Make Sense
2 min	content	Reflect
1 min	summary	What You Covered

TEACHING MOVES

- Keep the sample pseudo-code on the board and draw the shapes plan live in step 2 while students write their own; teach shapes only as each matching line appears.
- Get plans started within the first ten minutes; push anyone with two solid pseudo-code lines into Scratch early so something runs before the full build.
- Circulate during planning to check step precision; insist on testable lines such as "if answer equals 32".
- Open the build step with a short whole-class diamond rebuild so the idea lands even if some builds are unfinished; name it as testing and debugging.
- In the peer step, require extend + swap + precision note from everyone; put a 4-minute timer on the extend rewrite, hard-stop after the flag, and treat peer build as early-finisher only.
- Encourage students to flag vague steps in peer plans rather than correcting them silently.
- Show the class save routine live before the final two minutes.

WHAT TO WATCH FOR

- Students write vague steps like 'greet them'. Solution: Insist on specific actions such as 'ask for a name then say hello'.
- Flow charts miss decision diamonds for questions. Solution: Remind them to use diamond shapes for yes/no branches.
- Built program does not match the plan. Solution: Have students cross-check each step after coding; model two lines as a class.
- Score climbs on re-runs because it was never reset. Solution: Require SET score to 0 at the start of the script so plan and program still match.
- Extension plan is only a fragment. Solution: Require a full rewritten short plan a classmate can build from scratch, using the before then after lives example as a model of complete length.

DIFFERENTIATION

- Support: Offer the sample pseudo-code as sentence starters; core finish line is SET score plus the counties decision path only.
- Support: Pair students needing help with those who are confident planners.
- Extension: Early finishers complete the peer build to the decision diamond, or add a costume change on correct still matching the plan.