LESSON 69

Studio: Reaction Time Tester in Python

40 minutes | Project studios (optional extension sessions) | Built in Python | Outcome 3.1 | Outcome 3.9

WHAT THIS LESSON DOES

Students use PRIMM to build a micro:bit reaction time tester in Python, working with variables, loops, lists and conditionals, then compare the text build with the block version they already know.

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.
3.9	Analyse code to determine its function and find errors.

WHAT STUDENTS SHOULD BE ABLE TO DO

- Build a timing game in Python with variables and conditionals (LO 3.1)
- Compare the Python build with the block version you made earlier (LO 3.9)

BEFORE THE BELL

- Students logged in to the Coding Ireland platform on classroom devices
- Open the micro:bit Python editor at python.microbit.org (new blank project)
- Physical micro:bits and USB cables ready if you plan to flash the finished game
- Students should already be able to make a simple block-based reaction or timing activity

WHAT YOU NEED

Item	Qty	Per	If you do not have it
tablet or laptop with Python	1	pair	run the lesson on the IWB with students taking turns

Studio: Reaction Time Tester in Python

HOW THE LESSON RUNS

4 min	introduction	Start: what we're building
3 min	content	The plan: five parts to build
2 min	content	Import Modules
3 min	content	High Scores List and Game Loop
2 min	content	Add a Random Delay
3 min	content	Get Reaction Time
3 min	content	Store Reaction Time in List
5 min	content	Create Function to Check if High Score
5 min	content	Display High Scores
2 min	content	Download your Code
2 min	content	Wrap Up and Extensions
3 min	content	Make sense: what worked, what broke
2 min	content	Reflect
1 min	summary	What you covered

TEACHING MOVES

- Model the first two code steps on the board, then let pairs type the rest line by line; avoid copy-paste so students notice structure
- Name the skills as they appear: decomposition when the game is broken into wait, react, store and display; algorithms when the loop and function steps are ordered
- Keep the blocks-versus-text comparison alive: ask briefly which part felt clearer in text once the timer and list are working
- If a pair finishes early, have them play several rounds and watch how the high-score list updates before offering an extension

WHAT TO WATCH FOR

- Mixing button A and button B (start versus react): point at the two while-not-pressed waits and ask which button ends each one
- Forgetting to append the reaction time or calling the high-score function with the wrong value: check that the measured time is stored before the list is sorted or shown
- Indentation errors after the function definition or inside the main loop: show the editor highlight and realign the block as a class

DIFFERENTIATION

- Support: Stay with the pair on the import and first loop; provide the shape of the wait-then-show sequence verbally without typing their code
- Extension: Average the stored times, add a second input such as shake, or tighten the random delay range
- Extension: Ask stronger pairs to explain why lower times rank higher before they change any code