LESSON 27

Light Clapper

40 minutes | Introduction to Python | Outcome 3.11 | Outcome 3.12 | Outcome 3.13 | Outcome 3.14 | Outcome 2.6 | Outcome 2.7 | Outcome 1.22

WHAT THIS LESSON DOES

In this lesson, you'll create a sound-sensitive light clapper using your Micro:bit. Follow step-by-step instructions to programme it to detect claps and toggle the LED display, learning key coding concepts like loops, conditionals, and functions along the way.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
3.11	use and control digital inputs and outputs within an embedded system
3.12	measure and store data returned from an analogue input
3.13	develop a program that utilises digital and analogue inputs
3.14	design automated applications using embedded systems
2.6	construct algorithms using appropriate sequences, selections/conditionals, loops and operators to solve a range of problems, to fulfil a specific requirement
2.7	implement algorithms using a programming language to solve a range of problems
1.22	read, write, test, and modify computer programs

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand how to use a Micro:bit microphone to detect sound levels for interactive projects.
- Apply conditional logic and loops to monitor and respond to environmental inputs.
- Develop functions to manage state changes like toggling displays based on triggers.
- Implement pattern recognition for advanced input detection, such as double-clap sequences.
- Build skills in coding control structures and variables for real-world automation applications.

BEFORE THE LESSON

- Open the micro:bit Python editor yourself and confirm the simulator shows a microphone sound-level slider, since students test threshold values with it before any hardware.
- If physical micro:bits are used, charge or cable them up and check each USB connection registers, so the download step at the end does not stall.
- Run the final double-clap version once yourself so you know what a working timing window looks like when a student's does not toggle.

Light Clapper

HOW THE LESSON RUNS

- Introduction to the Light Clapper Project
- Import Modules and Basic Setup
- Read Microphone Sound Level
- Set Threshold and Detect Clap
- Add Toggle Function
- Advanced: Detect Double Clap
- Download your Code
- Wrap Up and Extensions

TEACHING MOVES

- 3. Read Microphone Sound Level. Have students drag the simulator slider while it scrolls so they see the 0 to 255 range with their own eyes. Insist they type the loop rather than paste, as the note asks.
- 4. Set Threshold and Detect Clap. Make the point that 100 is a guess, not a fact. Have them raise and lower the threshold and watch when the heart appears, so they understand it is tuned by testing.
- 5. Add Toggle Function. This is the hard step. Explain why global is needed: without it the function makes a new local light_on and the toggle never sticks. Point at the debounce sleep as the reason one clap does not flicker.
- 6. Advanced: Detect Double Clap. Draw the timing on the board: clap one starts a clock, a second clap within 1000ms toggles, otherwise reset. Let this be a stretch step, not everyone will reach it.
- 7. Download your Code. Walk one download live if hardware is new to the class. Real claps behave differently from the slider, so expect threshold retuning on the device.

WHAT TO WATCH FOR

- Students omit the global keyword and cannot see why the light never toggles, since the assignment silently creates a local variable. Have them print light_on inside and outside the function. Seeing it change inside but not outside makes the scope rule concrete.
- Students think a display that shows a heart in the simulator means the code is finished and correct. Ask them to predict what should happen at a level of exactly 100 given the > comparison, then test it. Looking right is not the same as behaving right.
- Students expect the same threshold to work on hardware as in the simulator. Explain the microphone reads a real noisy room. Have them re-test and adjust the number on the device.

DIFFERENTIATION

- Emerging: Give the working threshold code and have them focus only on changing the number and observing the effect, before touching functions. Pair them with someone at the simulator so a slider and a screen are shared while they read the loop aloud.
- Developing: Have them explain in their own words what global does before moving to the double-clap step. Ask them to add an else branch that clears the display and predict the result before running.
- Proficient: Push toward the double-clap timing logic and then a triple-clap extension, tracking clap_count and last_clap_time cleanly. Ask them to justify their debounce and threshold choices in writing, the kind of design reasoning a Coursework Project report needs.