LESSON 23

Step Counter Pedometer

40 minutes | Introduction to Python | Outcome 2.6 | Outcome 2.7 | Outcome 2.9 | Outcome 2.16 | Outcome 3.11 | Outcome 3.12 | Outcome 3.13

WHAT THIS LESSON DOES

In this lesson, you'll build a simple step counter pedometer using your Micro:bit's accelerometer to detect shakes as steps. Follow step-by-step instructions to code a variable for tracking steps, implement loops, and add a reset feature.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
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
2.9	assemble existing algorithms or create new ones that use functions (including recursive), procedures, and modules
2.16	use data types that are common to procedural high-level languages
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

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand how to use an accelerometer to detect movement as steps.
- Develop skills in using variables to track and accumulate data.
- Master the implementation of loops and conditionals for continuous monitoring.
- Learn to create and utilise functions for reusable code like resetting values.
- Apply coding concepts to build a functional step counter on a Micro:bit.

BEFORE THE LESSON

- Open python.microbit.org yourself first and confirm the simulator loads through the school filter, and that its shake gesture control works.
- If using physical micro:bits, have them cabled up with working USB leads and confirm code can flash to one device.
- Decide whether students save to a shared location or their own drive, so nobody loses a working .hex.

Step Counter Pedometer

HOW THE LESSON RUNS

Introduction to the Step Counter
Import Modules
Initialise Step Counter
Set Up the Main Loop
Increment and Display Steps
Add Reset with Button
Create a Reset Function
Download your Code
Wrap Up and Extensions

TEACHING MOVES

- 1. Introduction to the Step Counter. Frame the finished goal out loud: a device that counts shakes as steps and resets. Have everyone start a fresh project so nobody carries stray code from earlier.
- 2. Import Modules. Stress that from microbit import * runs silently and doing nothing visible is correct here. That reassurance prevents students thinking their code is broken.
- 3. Initialise Step Counter. Insist they type, not paste. Point out steps = 0 must sit before the loop so the count exists before anything adds to it.
- 4. Set Up the Main Loop. Explain was_gesture('shake') resets itself once read, so each shake registers once. Show the simulator's shake control before they run it.
- 5. Increment and Display Steps. Unpack steps += 1 as steps = steps + 1 on the board. This accumulator idea is worth more than the pedometer itself.
- 6. Add Reset with Button. Point out why str(steps) is needed to join a number to text. Note elif means shake and reset cannot both fire in one pass.
- 7. Create a Reset Function. Explain that global steps lets the function change the outer variable rather than making its own. Remove it live to show the count silently failing to reset.
- 9. Wrap Up and Extensions. Recap the four ideas by name: loop, conditional, accumulator variable, function. Offer button B as a stretch and make everyone save a .hex.

WHAT TO WATCH FOR

- Students think steps += 1 compares rather than accumulates, or reset steps to 0 inside the loop so it never counts up. Trace two shakes by hand on the board, showing steps go 0, 1, 2. Confirm steps = 0 sits once before the loop, not inside it.
- Students expect the accumulated count to survive without global inside the function, so reset appears to do nothing. Have them predict the output before and after adding global steps, then run both versions and compare.
- Students assume a page or simulator that shows a number is therefore working correctly, without testing edge cases. Ask them to reset, then shake, and confirm the count restarts from 1, not from the old total.

DIFFERENTIATION

- Emerging: Give them the import and steps = 0 lines already typed so they start from the loop. Pair them with a working simulator and have them describe what each shake should do before they run it.
- Developing: Ask them to explain aloud why elif is used instead of a second if before moving on. Have them add a comment beside global steps saying what it does in their own words.
- Proficient: Challenge them to add button B to display the count without resetting, then a threshold that scrolls a message every 10 steps. Ask them to write the reset function as evidence they could reuse in an Applied Learning Task, noting where else a reusable function would help.