LESSON 38

Conditionals and loops in Python

40 minutes | Strand 3: Coding at the next level | Built in Python | Outcome 3.1 | Outcome 3.4

WHAT THIS LESSON DOES

Students use PRIMM to explore conditionals and loops in Python on the micro:bit, building from simple if statements through nested structures and combined loop-and-condition patterns, then solve two short challenges.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
3.1	creatively design and write code for short programming tasks to demonstrate the use of operators for assignment, arithmetic, comparison, and Boolean combinations
3.4	describe program flow control, e.g. parallel or sequential flow of control (language dependent)

WHAT STUDENTS SHOULD BE ABLE TO DO

- Use conditionals and loops in Python with comparison operators and Boolean combinations (and, or, not) in the conditions, and explain how indentation controls structure (LO 3.1)
- Describe sequential program flow control in Python, and contrast it with the micro:bit's handlers running at the same time as an example of parallel flow of control (LO 3.4)

BEFORE THE BELL

- Students logged in to the platform with a Python micro:bit editor ready (USB or simulator as your room uses).
- micro:bits charged or simulators open, and one shared board view for modelling the first steps.
- Decide how the class will save Python work, and be ready to tell students that at the end so the files are available next lesson.

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

Conditionals and loops in Python

HOW THE LESSON RUNS

4 min	introduction	Start: what we're building
2 min	content	Two ways to change what runs
2 min	content	Understanding Conditional Statements
2 min	content	Simple If Statement
2 min	content	If-Else Statement
2 min	content	If-Elif-Else Statement
3 min	content	Nested If Statements
2 min	content	Introduction to Loops
2 min	content	For Loops
3 min	content	While Loops
2 min	content	Nested Loops
2 min	content	Using Loops with Conditionals
2 min	content	Exercise: Decision Maker
2 min	content	Exercise: Countdown Timer
2 min	content	Wrap Up
3 min	content	Make sense: what worked, what broke
2 min	content	Reflect
1 min	summary	What you covered

TEACHING MOVES

- Model indentation on the board with a clear indent mark so students see that the block under if or for is what runs.
- When you reach the Decision Maker exercise, take a minute at the board to say how this program runs: Python starts at the top of the file, runs each line in order, and inside while True it comes back around and checks button A, then button B, one after the other, over and over. Contrast that with the way the same job is written in the block editor, where you attach a handler to button A and a separate handler to button B and both sit waiting at once, so neither has to wait its turn. Ask what would happen in the Python version if a button were pressed and released during the sleep, and use their answer to make the point that sequential code can only notice what happens when it gets around to looking.
- For the exercises, keep the challenge brief: a loop plus a compound or multi-branch condition is enough.
- Write one prediction on the board at the start, for example what a simple if does when its condition is false, and come back to it in Make sense.

WHAT TO WATCH FOR

- Missing or uneven indentation after if, elif, else, for or while: the editor shows an IndentationError; realign the block to one consistent indent.
- Forgetting to update the loop variable in a while loop, causing an infinite loop: stop the program, then add the increment or change inside the loop.
- Comparing without converting types (for example scrolling a number without str): fix with str() on the value passed to display.scroll.

DIFFERENTIATION

- Support: For students who stall at the start of the first exercise, sit with them and get only the while True line and the sleep working, with the display scrolling anything at all, before either branch is added.
- Extension: Ask confident students to add a compound condition with and/or (for example both buttons) or a nested check inside the countdown.
- Extension: Invite a pair to explain, using their own button program, how their Python code checks one button and then the other in order, and how that differs from two handlers waiting at the same time.