LESSON 5

Introduction to Data Structures

40 minutes | Foundations of Computer Science | Outcome 2.16 | Outcome 2.18 | Outcome 2.6 | Outcome 2.7 | Outcome 1.22 | Outcome 1.6 | Outcome 1.7

WHAT THIS LESSON DOES

In this lesson, you'll explore the basics of data structures using MakeCode for micro:bit. Learn what data structures are, understand arrays and lists, and practise creating, modifying, adding, and removing elements in arrays through hands-on steps.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
2.16	use data types that are common to procedural high-level languages
2.18	collect, store and sort both continuous and discrete data
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
1.6	explain the operation of a variety of algorithms
1.7	develop algorithms to implement chosen solutions

WHAT STUDENTS SHOULD BE ABLE TO DO

- Grasp the fundamental concept of data structures and their significance in programming.
- Differentiate between arrays and lists, understanding their unique characteristics.
- Create and initialise arrays using MakeCode for micro:bit.
- Manipulate array elements by adding, removing, and modifying data.
- Locate specific elements within an array using index positions.

BEFORE THE LESSON

- Open makecode.microbit.org yourself and confirm it loads through the school filter on student machines. The whole lesson runs in this editor.
- Have the simulator ready to demonstrate. Students can complete every step in the on-screen micro:bit simulator, so physical devices and cables are optional, not required.
- Confirm students can save or download their MakeCode projects, since each step builds on the last across the period.

Introduction to Data Structures

HOW THE LESSON RUNS

- Introduction
- What are Data Structures?
- Understanding Arrays and Lists
- Creating a Simple Array
- Adding to an Array
- Removing from an Array
- Finding an Element
- Modifying an Array
- Wrap Up

TEACHING MOVES

- 2. What are Data Structures? Lean on the kitchen analogy: cupboard for plates, drawer for cutlery. Ask students to name their own everyday examples of organised storage before you introduce arrays and lists formally.
- 3. Understanding Arrays and Lists. Stress that indexing starts at zero, not one. Draw the scores array on the board with index labels underneath so element 20 sits at position 1, not 2.
- 4. Creating a Simple Array. Have students predict what showNumber(scores.length) will display before they run it. Confirm they see 3, the count, not the last value.
- 5. Adding to an Array. Point out push adds to the end. Ask what the display shows after one press and after three presses, then let them test it in the simulator.
- 6. Removing from an Array. Emphasise pop removes the last item only. Have students press A then B and watch length go up then down, connecting the block to the number on screen.
- 7. Finding an Element. Make the -1 result explicit: search for a value not in the array first so students see what not found looks like before searching for 20.
- 8. Modifying an Array. Reinforce that index 1 is the second element. Have students say aloud which score changes before they shake, so the zero-based counting sticks.

WHAT TO WATCH FOR

- Students count array positions from 1, so they expect index 1 to be the first element. Return to the labelled board diagram. Ask them to point to element at index 0, then index 1, and confirm against the modify step where index 1 changes the second score.
- Confusing scores.length with the value of an element, expecting length to show the last score. Show that length is a count. Add or remove an item and have them watch the count change while the stored values stay the same.
- Assuming push and pop can add or remove at any position, not just the end. Demonstrate that both operations work only on the end of the array. Contrast with the index assignment in the modify step, which targets a specific position.

DIFFERENTIATION

- Emerging: If the editor is fighting them, seat them beside a working machine and give the exact starter array to type so they reach a running program. Have them read each block aloud and say what it does before pressing run.
- Developing: Ask them to predict the display value at each button press before testing, then check their prediction against the simulator.
- Proficient: Challenge them to store several scores and display the highest, or to search for a value entered rather than the fixed 20. Ask them to explain in writing why an array is efficient for access by index but less flexible for insertion, framing arrays versus lists at Higher Level depth.