LESSON 22

Lists and Tuples

80 minutes (2 periods) | Introduction to Python | Outcome 2.16 | Outcome 2.18 | Outcome 2.6 | Outcome 2.7 | Outcome 2.8 | Outcome 1.22 | Outcome 1.7

WHAT THIS LESSON DOES

Explore the essentials of Python data structures in this lesson. You'll learn about creating and manipulating lists, performing operations like sorting and iteration, and understand the differences between mutable lists and immutable tuples through hands-on coding tasks.

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
2.8	apply basic search and sorting algorithms and describe the limitations and advantages of each algorithm
1.22	read, write, test, and modify computer programs
1.7	develop algorithms to implement chosen solutions

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the fundamental concepts and characteristics of lists and tuples as data structures.
- Differentiate between mutable and immutable collections to choose the appropriate structure for specific programming needs.
- Apply basic operations like appending, removing, sorting, and iterating to manage data in lists.
- Recognise the efficiency and use cases of tuples for fixed, unchangeable data.
- Develop skills in organising and manipulating data collections effectively in programs.

BEFORE THE LESSON

- Confirm the micro:bit editor runs and code flashes to a device, or that the simulator is available for those without hardware. Have cables ready and devices cabled up.
- Have the running example code from steps 3 and 5 saved somewhere students can copy from, so a typo in the starter does not stall the class.
- Test one full flash yourself first: display.scroll of a list prints with brackets and quotes, which surprises students expecting bare numbers.

Lists and Tuples

HOW THE LESSON RUNS

- Introduction
- What are Lists?
- Basic List Operations
- Iterating over Lists
- Sorting Lists
- What are Tuples?
- Iterating over Tuples
- Lists vs Tuples
- Exercise: List Sorting and Iteration
- Exercise:
- Wrap Up

TEACHING MOVES

- 2. What are Lists? Draw the list on the board with index numbers written under each item, and add the negative indices underneath. Make fruits[0] and fruits[-1] concrete before anyone types.
- 3. Basic List Operations. Stop after each display.scroll so students see the list change between append, remove and pop. Point out pop returns the removed item while remove does not.
- 4. Iterating over Lists. Walk one loop pass by hand: name the loop variable, show it takes each item in turn. Have them predict the output before flashing.
- 5. Sorting Lists. Contrast sort, which changes the list in place and returns nothing, with sorted, which returns a new list. Show that x = numbers.sort() gives None, a classic trap.
- 6. What are Tuples? Ask what happens if they try days.append or days[0] = something. Let them hit the error, then name it: tuples are immutable.
- 8. Lists vs Tuples. Build a two-column board list: mutable versus immutable, changing data versus fixed data. Anchor each side with an example, scores versus days of the week.
- 9. Exercise: List Sorting and Iteration. Have students initialise the sum variable to 0 before the loop. The commonest fail is adding to a variable that was never created, or resetting it inside the loop.
- 10. Exercise:. Get them to trace the even check with one number aloud before coding. Watch that the even sum starts at 0 outside the loop and the filter uses % 2 == 0, not = .

WHAT TO WATCH FOR

- Students think sort returns the sorted list and write sorted_list = numbers.sort(), getting None. Show live that numbers.sort() prints None while numbers itself is now sorted. sort changes in place; sorted returns a new list.
- The running total is reset inside the loop or never initialised, so the sum is wrong or the program errors. Point to where total = 0 must sit: above the loop, once. Trace two passes by hand to show the value accumulating.
- Students expect display.scroll of a list to show clean numbers, then are confused by brackets and quotes. Explain str of a whole list gives its printed form with punctuation. To show items alone, iterate and scroll each one, as in the loop steps.

DIFFERENTIATION

- Emerging: Give them the working code from step 3 to type and run, then change one value and predict the output before flashing. Pair with a student whose device is flashing, so environment trouble does not become the whole lesson.
- Developing: Ask them to explain aloud why sort changes numbers but sorted does not, before moving on. Have them add a print or scroll after each operation to see the list at every stage.
- Proficient: Ask why a tuple can be a dictionary key but a list cannot, linking immutability to fixed data. Higher Level treatment of the same idea. Challenge them to compute the average in exercise 9 using len, handling the integer division carefully.