LESSON 21

String Manipulation

40 minutes | Introduction to Python | Outcome 2.16 | Outcome 1.22 | Outcome 2.7 | Outcome 2.20 | Outcome 1.3 | Outcome 3.11

WHAT THIS LESSON DOES

In this lesson, you'll explore the essentials of handling text in Python. Learn to understand strings as character sequences, perform operations like concatenation, apply methods to transform text, use slicing for substrings, and practise with hands-on Micro:bit tasks.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
2.16	use data types that are common to procedural high-level languages
1.22	read, write, test, and modify computer programs
2.7	implement algorithms using a programming language to solve a range of problems
2.20	identify and fix/debug warnings and errors in computer code and modify as required
1.3	solve problems by deconstructing them into smaller units using a systematic approach in an iterative fashion
3.11	use and control digital inputs and outputs within an embedded system

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the concept of strings as sequences of characters for text handling in programming.
- Master basic string operations like concatenation and determining length.
- Apply various string methods to transform and manipulate text effectively.
- Utilise string slicing to extract specific portions of text data.
- Develop practical skills in text manipulation through hands-on exercises.

BEFORE THE LESSON

- Open the micro:bit Python editor and confirm it loads for the class, whether on physical boards or the simulator. Slicing and method output can be checked on the simulator alone if boards or cables are short.
- Run the step 3 and step 9 code snippets yourself first so you can see how display.scroll() handles each result and how long the scroll takes.

String Manipulation

HOW THE LESSON RUNS

- Introduction
- What are Strings?
- Basic String Operations
- String Methods
- Searching Strings
- Testing Strings
- Placeholders in Strings
- String Slicing
- Exercise: Manipulate a Message
- Wrap Up

TEACHING MOVES

- 2. What are Strings? Stress immutability early: you cannot change a character in place, every method returns a new string. This is the single idea that prevents confusion later when methods appear to do nothing.
- 3. Basic String Operations. Point out why str() wraps len(message): display.scroll needs text, and len returns a number. Ask what happens without it so they meet the type error deliberately, not by accident.
- 4. String Methods. Show that message.upper() does not change message unless you reassign it. Print the original after calling the method to prove the point rather than just stating it.
- 5. Searching Strings. Highlight that find() returns -1 when nothing is found, not an error. Have students predict the index before running, then check against the count of characters.
- 6. Testing Strings. These return True or False, so they belong inside if statements. Ask students where isdigit() would matter, for example checking a scrolled input is a number before converting it.
- 7. Placeholders in Strings. Contrast format() against the concatenation from earlier. Show the same greeting built both ways so students see format() avoids mixing quotes, plus signs and str() calls.
- 8. String Slicing. Draw the indexed row of characters on the board with position numbers under each letter. Emphasise end is excluded and negative indices count back from the last character.
- 9. Exercise: Manipulate a Message. Insist they attempt it before revealing the solution. Note the replace uses "FUN" not "Fun" because uppercase ran first: this ordering trap is the real learning point.

WHAT TO WATCH FOR

- Students believe a string method changes the original string, so they call msg.strip() on its own line and expect msg to be trimmed afterwards. Print the variable before and after. Show nothing changed, then show that assigning the result, stripped = msg.strip(), is what captures the new string.
- Confusion over slice bounds: students expect [1:4] to include position 4, giving four characters instead of three. Use the board diagram and count aloud: start included, end excluded. Have them work out [1:4] on "Hello" by hand before running it.
- In the final exercise, students chain replace("Fun", ...) after upper() and it silently fails because the text is now "FUN". Let it fail, then trace the message state after each step. This teaches that transformations are cumulative and case matters to replace().

DIFFERENTIATION

- Emerging: Give the step 3 snippet already typed and ask them only to change the words being joined, so they see output before writing from scratch. Pair with a student whose simulator or board is working if hardware is fighting them.
- Developing: Have them explain out loud what each line of the exercise leaves the message as before running it, so they reason about state rather than just typing.
- Proficient: Ask them to rewrite the final exercise output using format() placeholders instead of separate scroll calls. Challenge them to validate scrolled text with isdigit() before a conversion, evidence they could reuse in an Applied Learning Task.