LESSON 18

Variables and Data Types

80 minutes (2 periods) | Introduction to Python | Outcome 2.16 | Outcome 2.6 | Outcome 2.7 | Outcome 1.22 | Outcome 2.20 | Outcome 3.13 | Outcome 2.18

WHAT THIS LESSON DOES

In this lesson, you'll explore the basics of variables and data types in Python using your Micro:bit. Learn how to create variables, assign different data types, and apply them in simple programs to store and manipulate information effectively.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
2.16	use data types that are common to procedural high-level languages
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
2.20	identify and fix/debug warnings and errors in computer code and modify as required
3.13	develop a program that utilises digital and analogue inputs
2.18	collect, store and sort both continuous and discrete data

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the concept of variables as containers for storing data in programming.
- Identify and differentiate between various data types such as integers, floats, strings, and booleans.
- Apply correct syntax rules for naming and creating variables in Python.
- Demonstrate the ability to check and convert data types using built-in functions.
- Utilise variables and data types in simple programs to perform operations and display results.

BEFORE THE LESSON

- Open python.microbit.org yourself and confirm it loads through the school filter; the whole lesson depends on it.
- Have micro:bits and USB cables ready, one per student or pair, and confirm they flash from the editor before class.
- Test the calculator and temperature exercises yourself so you know how division returns a float and how the display scrolls slowly.

Variables and Data Types

HOW THE LESSON RUNS

Introduction to Variables
What are Variables?
Variable Syntax
Integer Data Type
Float Data Type
String Data Type
Boolean Data Type
Checking Data Types
Type Conversion
Exercise: Create and Display Variables
Exercise: Simple Calculator
Exercise: Temperature Converter
Wrap Up

TEACHING MOVES

- 2. What are Variables? Stress that the equals sign means assign, not equals. Say it aloud as 'age gets 16', not 'age equals 16'. This framing prevents the comparison confusion that appears later with booleans and conditions.
- 3. Variable Syntax. Have students predict which of a few names are legal before you reveal the rules: 2score, my age, school_year, if. Point out case sensitivity concretely: myAge and myage are two different boxes.
- 5. Float Data Type. Note that any division with / gives a float even when the answer is whole. This explains why the calculator exercise shows a decimal for division. Foreshadow it now.
- 7. Boolean Data Type. Point out True and False are capitalised and are not strings. Writing "True" in quotes makes a string that is always truthy, a common trap. Connect booleans forward to if statements.
- 8. Checking Data Types. Frame type() as a debugging tool, not just theory. Show a case where an int stored as a string breaks arithmetic, so students see why checking matters.
- 9. Type Conversion. Demonstrate a deliberate failure: int("hello") throwing an error. Then int("42") succeeding. Let them see the error message so it is not scary later.
- 10. Exercise: Create and Display Variables. Circulate and check each display scrolls before students move on. Ask them to change values to their own name, age and height so they own the code rather than copying it.
- 11. Exercise: Simple Calculator. Ask students to predict the division result before running, then compare. Point out that swapping a and b to floats changes the other results too.
- 12. Exercise: Temperature Converter. Confirm the class reads 20C as 68F. Have them test 0 and 100 as checks against known answers, then add the boolean freezing check as the stretch.

WHAT TO WATCH FOR

- Students read = as 'is equal to' and expect it to test a value rather than store one. Say assignments aloud as 'gets': 'sum gets a plus b'. Reserve the word equals for later comparison work with ==.
- Writing booleans in quotes, e.g. likes_python = "True", making a string instead of a boolean. Run type() on both "True" and True side by side so students see str versus bool, and note the string is always truthy in a condition.
- Assuming a whole-number division result will be an integer. Have them run 6 / 3 and read the scrolled result. Show that / always yields a float in Python, whatever the numbers.

DIFFERENTIATION

- Emerging: If the editor or micro:bit is not cooperating, pair them with a working setup and let them type into a partner's editor so they still practise the syntax. Give the variable names first and let them fill only the values, removing one thing to get wrong.