Year 9 – Python programming with sequences of data

Unit introduction

This unit introduces learners to how data can be represented and processed in sequences, such as lists and strings. The lessons cover a spectrum of operations on sequences of data, that range from accessing an individual element to manipulating the entire sequence. Great care has been taken so that the selection of problems used in the programming tasks are realistic and engaging: learners will process solar system planets, book texts, capital cities, leaked passwords, word dictionaries, ECG data, and more.

A range of pedagogical tools are employed throughout the unit, with the most prominent being pair programming, live coding, and worked examples.

The Year 7 and 8 Programming units are prerequisites for this unit. It is assumed that learners are already able to write Python programs that display messages, receive keyboard input, use simple arithmetic expressions, and control the flow of program execution through selection and iteration structures.

Overview of lessons

Lesson	Brief overview	Learning objectives
1 Warm up	This introductory lesson serves a double purpose: it reconnects learners with Python, making sure they can read and create simple programs that use selection, and it also takes a step forward, providing a very gentle introduction to lists.	• Write programs that display messages, receive keyboard input, and use simple arithmetic expressions in assignment statements • Use selection (if-elif-else statements) to control the flow of program execution • Locate and correct common syntax errors

		• Create lists and access individual list items
2 Playlist	This lesson provides learners with an overview of the operations that are commonly performed on lists: adding, removing, or modifying items; locating or counting occurrences of particular items, etc. Learners are presented with a set of short challenges. They are asked to identify the list operations that would be relevant and apply them to perform the required tasks. Through these challenges, learners will indirectly gain a better understanding of the sort of problems where lists might be useful. They also get accustomed to using dot notation for list methods, although this is not the focus of the lesson.	• Perform common operations on lists or individual items
3 In a `while`, crocodile	This lesson revolves around iteration using `while` loops, offering learners a chance to retrieve and apply relevant knowledge. In the first activities, learners will practise using list operations in iterative contexts. Learners will be introduced to the similarities between lists and strings, which will be based on what they already know about operations relating to length, membership, and access to individual characters. The final activity requires them to apply these string operations in an iterative context.	• Use iteration (`while` statements) to control the flow of program execution • Perform common operations on lists or individual items • Perform common operations on strings or individual characters
4 The famous `for`	In this lesson, learners will use a `for`-loop to iterate over list items. They will initially study a range of examples — to familiarise themselves with its syntax, use, and mechanics — before moving on to apply what they've learnt to similar tasks. The activities involve iterating over lists of real-world textual and numerical data, requiring learners to recall and apply knowledge from the previous lessons.	• Use iteration (`for` statements) to iterate over list items • Perform common operations on lists or strings

	The lesson ends with a nod towards using for to iterate over the characters of a string, which may come in handy when learners attempt to solve problems independently.	
5 Make a thing	IIn this lesson, learners will be provided with a selection of meaningful mini-projects that will allow them to apply the knowledge and skills they have acquired so far. Each project contains a short introduction that provides context, a detailed description of what learners are expected to develop, and a set of clues that will support them in putting together a solution. Each learner is expected to select one of the mini-projects and complete it within this lesson, or in the first part of the next one. Before starting work on the projects, two short activities will provide learners with additional support around accumulating sums and using for to iterate over strings. This is generally important and will also prove useful in some of the projects.	• Use iteration (for loops) to iterate over lists and strings • Use variables to keep track of counts and sums • Combine key programming language features to develop solutions to meaningful problems
6 Wrap up	In this final lesson, learners will be given the opportunity to complete their mini-project or explore a second one. They will then take a quiz that will assess their grasp of the programming concepts they have encountered throughout the unit. An optional activity is also provided, for learners that finish early with their assessment quiz, or are simply keen on an additional challenge.	

Progression

This unit builds upon the Python unit learnt in Year 8.

Curriculum links

National curriculum links (Computing programmes of study: Key Stage 3)

Aims

- can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms, and data representation
- can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems

Subject content

- use two or more programming languages, at least one of which is textual, to solve a variety of computational problems
- understand how instructions are stored and executed within a computer system
- understand several key algorithms that reflect computational thinking; use logical reasoning to compare the utility of alternative algorithms for the same problem
- design, use, and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems.