



Raspberry Pi

Year 7 – Programming II

Unit introduction

Programming II follows on from the foundations built in ‘Programming I’. It is vital that learners complete ‘Programming I’ before beginning this unit.

This unit begins right where ‘Programming I’ left off. Learners will build on their understanding of the control structures’ sequence, selection, and iteration (the big three), and develop their problem-solving skills. Learners will learn how to create their own subroutines, develop their understanding of decomposition, learn how to create and use lists, and build upon their problem-solving skills by working through a larger project at the end of the unit.

Overview of lessons

Lesson	Brief overview	Learning objectives
Lesson 7: You've got the moves!	This lesson is designed to formalise the use of subroutines, a technique that has been introduced gently over the previous unit. Learners will create a dance battle game by decomposing dance moves and creating subroutines for each move.	<ul style="list-style-type: none">Define a subroutine as a group of instructions that will run when called by the main program or other subroutinesDefine decomposition as breaking a problem down into

		<p>smaller, more manageable subproblems</p> <ul style="list-style-type: none"> ● Identify how subroutines can be used for decomposition
Lesson 8: Fly cat fly!	Learners are introduced to the concept of condition-controlled loops by using the PRIMM approach with a Scratch game called 'Fly cat, fly!'. They will predict, run, investigate, and modify code in order to build confidence with using condition-controlled loops.	<ul style="list-style-type: none"> ● Identify where condition-controlled iteration can be used in a program ● Implement condition-controlled iteration in a program
Lesson 9: Loop the loop!	Learners should have a grasp of each type of iteration available to them in Scratch. This lesson focuses on when each type of iteration should be used. It will give learners the evaluative skills to implement iteration in their own programs as they start to develop them.	<ul style="list-style-type: none"> ● Evaluate which type of iteration is required in a program
Lesson 10: Treasure those lists!	Learners are introduced to lists during this lesson. There is initially an educator-led demonstration on a simple shopping list application created in Scratch. Learners then dig deeper into lists by navigating through a treasure hunt game. The object of the game is to collect and swap the right items in order to reach the next level. Learners should use their investigation skills to discover the essential tools that Scratch can offer surrounding lists.	<ul style="list-style-type: none"> ● Define a list as a collection of related elements that are referred to by a single name ● Describe the need for lists ● Identify when lists can be used in a program ● Use a list
Lesson 11: Translate this! (Part 1)	Learners are given a scenario to create a translation quiz for a Modern Foreign Languages teacher. The learners will decompose the problem and start to build a Scratch program to meet the requirements. This is a pair programming project that takes place over two lessons; pairs will develop their programs to differing levels. A rubric is to be used for	<ul style="list-style-type: none"> ● Decompose a larger problem into smaller subproblems ● Apply appropriate constructs to solve a problem
Lesson 12: Translate this! (Part 2)		

	peer- or self-assessment to check progress. Extension activities allow learners to explore more challenging aspects of the solution. In Lesson 12, learners will be given a multiple choice quiz as a formal final assessment.	
--	--	--

Progression

This unit progresses students' knowledge and understanding of...

Please see the learning graph for this unit for more information about progression.

Note: this also covers objectives for Year 7 – Programming I

Curriculum links

National curriculum links

- To use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; to make appropriate use of data structures (for example, lists, tables, or arrays); to design and develop modular programs that use procedures or functions
- To understand several key algorithms that reflect computational thinking; use logical reasoning to compare the utility of alternative algorithms for the same problem
- To understand simple Boolean logic (for example, AND, OR, and NOT)
- To create, reuse, revise, and repurpose digital artefacts for a given audience, with attention to trustworthiness, design, and usability

Assessment

Self- and peer-assessment

This unit contains a challenging pair programming project that learners will complete. This can be self- or peer-assessed using the rubric.

- Please see the documents in Lesson 11: Translate this! Part 1

Summative assessment

Please see the assessment question and answer documents for this unit.

Subject knowledge

This unit focuses on the development of the following key techniques:

- Decomposition
- Subroutines
- Condition-controlled iteration
- Lists
- Problem solving

Scratch is used throughout the unit so it is important that you are comfortable with the language. In order to get a feel for the level of skill required, why not try the activities in the unit yourself before using them with your learners?

Enhance your subject knowledge to teach this unit through the following training opportunities:

Online training courses

- [Secondary Programming Pedagogy: Inspiring Computing Teaching](#) (ncce.io/secondarypedagogy)
- [Programming 101](#) (ncce.io/prog101)
- [Programming 102](#) (ncce.io/progr102)

Face-to-face courses

[Key stage three computing for non specialists](#) (ncce.io/f2fks3)

Resources are updated regularly — the latest version is available at: [ncce.io/tcc](#).

This resource is licensed under the Open Government Licence, version 3. For more information on this licence, see [ncce.io/ogl](#).