



Purple Mash Computing Scheme

Year 3 Coding

Contents

Overview

- Key Learning
- Knowledge Organiser and Learning Mat
- Concept Map and Quiz
- Key Terms Videos

Lesson 1: Using Flowcharts

- Aim
- Success Criteria
- Resources
- Preparation

Lesson 2: Timers

- Aim
- Success Criteria
- Resources
- Preparation

Lesson 3: Repeats

- Aim
- Success Criteria
- Resources
- Preparation

Lesson 4: Nesting

- Aim
- Success Criteria
- Resources
- Preparation

Lesson 5: Design a Program

- Aim
- Success Criteria
- Resources
- Preparation
- Teaching Notes

Need more support? Contact us:

Tel: +44(0)208 203 1781 | Email: support@2simple.com | X: [@2simplesoftware](https://twitter.com/2simplesoftware)

Lesson 6: Coding a Program

Aim

Success Criteria

Resources

Preparation

Assessment

Overview

This unit builds on learning from previous coding units. Children will explore timers, the repeat command and the importance of nesting code. They will build up to using design documentation to code a program.

Key Learning

- To understand what a flowchart is and how flowcharts are used in computer programming.
- To understand that there are different types of timers.
- To understand how to use the repeat command.
- To understand the importance of nesting when coding, testing and debugging.
- To design a program.
- To use design documentation to code a program.

Knowledge Organiser and Learning Mat

These resources can be used to support learning across the unit.

- [Knowledge Organiser](#)
- [Learning Mat](#)

Concept Map and Quiz

Use these resources to gauge children's understanding before or after the unit has been taught.

- [Concept Map](#)
- [Quiz](#)

Key Terms Videos

Teacher videos explaining key coding terminology.

- [Key Terms Explained](#)

Lesson 1: Using Flowcharts

Aim

- To understand what a flowchart is and how flowcharts are used in computer programming.

Success Criteria

- I can read and explain a flowchart
- I can use a flowchart to create a computer program.

Preparation

- Set **Coding Using Flowcharts** as a 2Do.

Please find all lesson resources below.

Resources

- [Slideshow - Lesson 1](#)
- [Coding Using Flowcharts](#)

Lesson 2: Timers

Aim

- To understand that there are different types of timers.

Success Criteria

- I can create a program that uses a timer-after command.
- I can create a program that uses a timer-every command.
- I understand when to use the different types of timers.

Preparation

- Set **Tick Tock Clock** as a 2Do.
- Set **Rockets** as a 2Do.
- (Optional Extension) Set **Lightning Scene** as a 2Do so children can check their predictions.

Please find all lesson resources below.

Resources

- [Slideshow - Lesson 2](#)
- [Tick tock clock challenge](#)
- [Tick tock clock challenge Solution](#)
- [Rockets](#)
- [Rockets Solution](#)
- [Lightning Scene](#)

Lesson 3: Repeats

Aim

- To understand how to use the repeat command.

Success Criteria

- I understand how the turtle object moves.
- I can use the repeat command with an object.
- I can create a computer program that includes the use of the repeat command

Preparation

- Set **Freecode Chimp** as a 2Do for the class.
- Gather a class set of whiteboards.

Please find lesson resources below and additional teaching notes.

Resources

- [Slideshow - Lesson 3](#)
- [Free Code Chimp](#)

Lesson 4: Nesting

Aim

- To understand the importance of nesting when coding, testing and debugging.

Success Criteria

- I can run, test and debug programs.
- I can pay attention to how code blocks are nested when debugging programs.

Preparation

- Set **Jumping Monkey** as a 2Do.
- Set **Superheroes** as a 2Do.
- Set **Newton and the Apple** as a 2Do.

Resources

- [Slideshow - Lesson 4](#)
- [Jumping Monkey](#)
- [Jumping Monkey Solution](#)
- [Superheroes](#)
- [Superheroes Solution](#)
- [Newton and the Apple](#)
- [Newton Solution](#)

Lesson 5: Design a Program

Aim

- To design a program.

Success Criteria

- I can create design documentation to plan the look of a program.
- I can create design documentation to plan the algorithm for a program using my 2Code knowledge.

Preparation

- Provide printed copies of the **Storyboard Planner** (with 4 boxes or 6 boxes) for the class.
- Provide paper or direct children to **2Chart** to create a flowchart.
- Set **Freecode Chimp** as a 2Do; children will continue with this 2Do in the next lesson so set a sufficiently long closing date for the 2Do.

Please find all lesson resources and teacher notes below.

Resources

- [Slideshow - Lesson 5](#)
- [Lightning Scene](#)
- [Moon Phases](#)
- [Solar System](#)
- [Viking Discovery](#)
- [Storyboard Planner - 4 boxes](#)
- [Storyboard Planner - 6 boxes](#)
- [Free Code Chimp](#)

Teaching Notes

The following resources will be examined as a class they do not need to be set as 2Dos, they are linked within the slideshow. Use all, or just a selection, of these examples when directed to in the lesson.

- Lightning Scene
- Moon Phases
- Solar System
- Viking Discovery

Lesson 6: Coding a Program

Aim

- To use design documentation to code a program.

Success Criteria

- I can use my design documents to guide my coding.
- I can test and make sensible attempts to debug the code.

Preparation

- Children will need their program designs from the previous lesson. If these are not finished, they should complete them before coding.
- Children will need to open **Freecode Chimp** from their 2Dos set in the last lesson.
- (Optional Extension) Class set of Post-it notes.

Please find all lesson resources below.

Resources

- [Slideshow - Lesson 6](#)

Assessment

The following information is an exemplar of what a child at an expected level would be able to demonstrate when completing this unit with additional exemplars to demonstrate how this would vary for a child with emerging or exceeding achievements.

Emerging

- Children can design and code a program that follows a simple sequence. They can make good attempts to 'read' code and predict what will happen in a program which can help them to correct errors. Children's designs for their programs, show that they are thinking of the structure of a simple program in logical, achievable steps.

Expected

- Children can design and code simple sequential programs, using timers effectively to create delay effects and understanding the difference between "timer-after" and "timer-every" commands. Their designs reflect logical thinking with clear steps and event-driven actions. Most can explain their command choices and use the repeat function to draw shapes like squares. They show a growing understanding of code structure, which aids in debugging. Additionally, they integrate multimedia elements such as sound, animation, and images. They apply actions to multimedia elements to animate them. Children can also reflect on their progress and how prior learning has supported their development.

Exceeding

- Children's designs show that they are thinking of the required task and how to accomplish this in code. They can identify an error within a program that prevents it following the desired algorithm and then fix it. Children make intuitive attempts to debug their own programs as they increase in complexity. Children are able to use the repeat command to produce outcomes beyond the set task. They have a good understanding of timers within timers in a program evidenced in their program designs. Children exhibit greater ease at fixing their own bugs as their coding becomes more complex.