

The logo consists of a black rectangular box with a white, torn-paper-like corner on the top right. Inside the box, the word "purple" is written in a bold, lowercase, purple sans-serif font, and the word "mash" is written below it in a bold, lowercase, white sans-serif font.

**purple
mash**

Purple Mash Computing Scheme

Logo

Contents

Overview

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

Lesson 1: Introduction to Logo

- Aim
- Success Criteria
- Resources
- Preparation

Lesson 2: Drawing Letters

- Aim
- Success Criteria
- Resources
- Preparation

Lesson 3: Change Line Colour and Thickness and use the Repeat Command

- Aims
- Success Criteria
- Resources
- Preparation

Lesson 4: Using Procedures

- Aim
- Success Criteria
- Resources
- Preparation

Assessment

Overview

This unit introduces children to Logo programming, focusing on developing their understanding of basic commands and programming concepts. Children will learn to create shapes and letters using instructions that contain advanced features such as procedures.

Key Learning

- To know key commands and input simple instructions.
- To use a variety of commands to create shapes using multi-line mode.
- To use the Repeat command.
- To change the line thickness and colour.
- To use procedures to write instructions.

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](#)

Lesson 1: Introduction to Logo

Aim

- To know key commands and input simple instructions.

Success Criteria

- I can use commands to create onscreen shapes.
- I can make predictions about outputs.

Preparation

- Gather a class set of whiteboards.
- Set **Shape 1** as a 2Do.
- Set **Shape 2** as a 2Do.
- Set **Shape 3** as a 2Do.
- Set **Shape 4** as a 2Do.
- (Optional Extension) Set **Logo** as a 2Do or direct children to open it from the tools section of Purple Mash to complete the extension task.

Please find all lesson resources below.

Resources

- [Slideshow - Lesson 1](#)
- [Logo](#)
- [Shape 1](#)
- [Shape 2](#)
- [Shape 3](#)
- [Shape 4](#)

Lesson 2: Drawing Letters

Aim

- To use a variety of commands to create shapes using multi-line mode.

Success Criteria

- I can use the PU and PD commands.
- I can use the multi-line mode to write instructions.
- I can write instructions to draw letters.

Preparation

- Gather a class set of whiteboards.
- Set **Letter S** as a 2Do.
- Set **Letter F** as a 2Do.
- Set **Letter E** as a 2Do.
- Set **Letter C** as a 2Do.
- (Optional Extension) Set **Logo** as a 2Do or direct children to open it from the tools section of Purple Mash to complete the extension task.

Please find all lesson resources below.

Resources

- [Slideshow - Lesson 2](#)
- [Logo](#)
- [Letter S](#)
- [Letter F](#)
- [Letter E](#)
- [Letter C](#)

Lesson 3: Change Line Colour and Thickness and use the Repeat Command

Aims

- To use the Repeat command.
- To change the line thickness and colour.

Success Criteria

- I can use the 'Repeat' function.
- I can change the colour of the line.
- I can change the thickness of the line.

Preparation

- Print out the **Drawing Shapes PDF** for each child.
- (Optional Extension) Set **Logo** as a 2Do or direct children to open it from the tools section of Purple Mash to complete the extension task.

Please find all lesson resources below.

Resources

- [Slideshow - Lesson 3](#)
- [Logo](#)
- [Drawing Shapes](#)

Lesson 4: Using Procedures

Aim

- To use procedures to write instructions.

Success Criteria

- I can write procedures.
- I can use procedures to write instructions.

Preparation

- Set **Procedure 1** as a 2Do.
- Set **Procedure 2** as a 2Do.
- Set **Procedure 3** as a 2Do.
- (Optional Extension) Set **Logo** as a 2Do or direct children to open it from the tools section of Purple Mash to complete the extension task.

Please find all lesson resources below.

Resources

- [Slideshow - Lesson 4](#)
- [Logo](#)
- [Procedure 1](#)
- [Procedure 2](#)
- [Procedure 3](#)

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 read short programs and often predict outcomes using logical thinking, though not always accurately. They usually plan code in small steps, one or two commands at a time. When code doesn't work as expected, they sometimes find and fix errors independently, but longer code may require support. They understand that the repeat command makes actions happen more than once, though they may not plan repeats in advance, relying instead on trial and error. Children can create procedures but may not fully appreciate their benefit. They don't always recognise that procedures improve code quality and help avoid repeating the same instructions.

Expected

- Children can read multi-step programs and accurately predict the outcome. They plan sequences using two or more commands before running the code. When code doesn't work as expected, they may find and fix errors independently, though complex code might still require support. They understand the repeat command and can plan simple repeat structures rather than relying on trial-and-error. Children explore repeating procedures to create more complex patterns, showing some mathematical understanding. They recognise that procedures make code more efficient and use them appropriately. They can adapt instructions within repeat commands to create and refine common shapes, using procedures to improve efficiency.

Exceeding

- Children enjoy challenging themselves by planning sequences with several commands before running the code to check the outcome. They confidently use PU (pen up) and PD (pen down) commands to create specific drawing effects. When their code doesn't behave as expected, they apply logical reasoning and debugging techniques to find and fix errors independently. They create and call procedures efficiently, understanding their value in making code clearer and avoiding repetition. Children experiment with repeating procedures to build more complex patterns, demonstrating their understanding of the mathematical ideas involved. They can read and understand increasingly complex programs, including those with procedures within repeats, and accurately predict the outcome.