DigitalSkills.org

Teacher Learning Plan

Digital Skills Curriculum 2024/25

5th Class

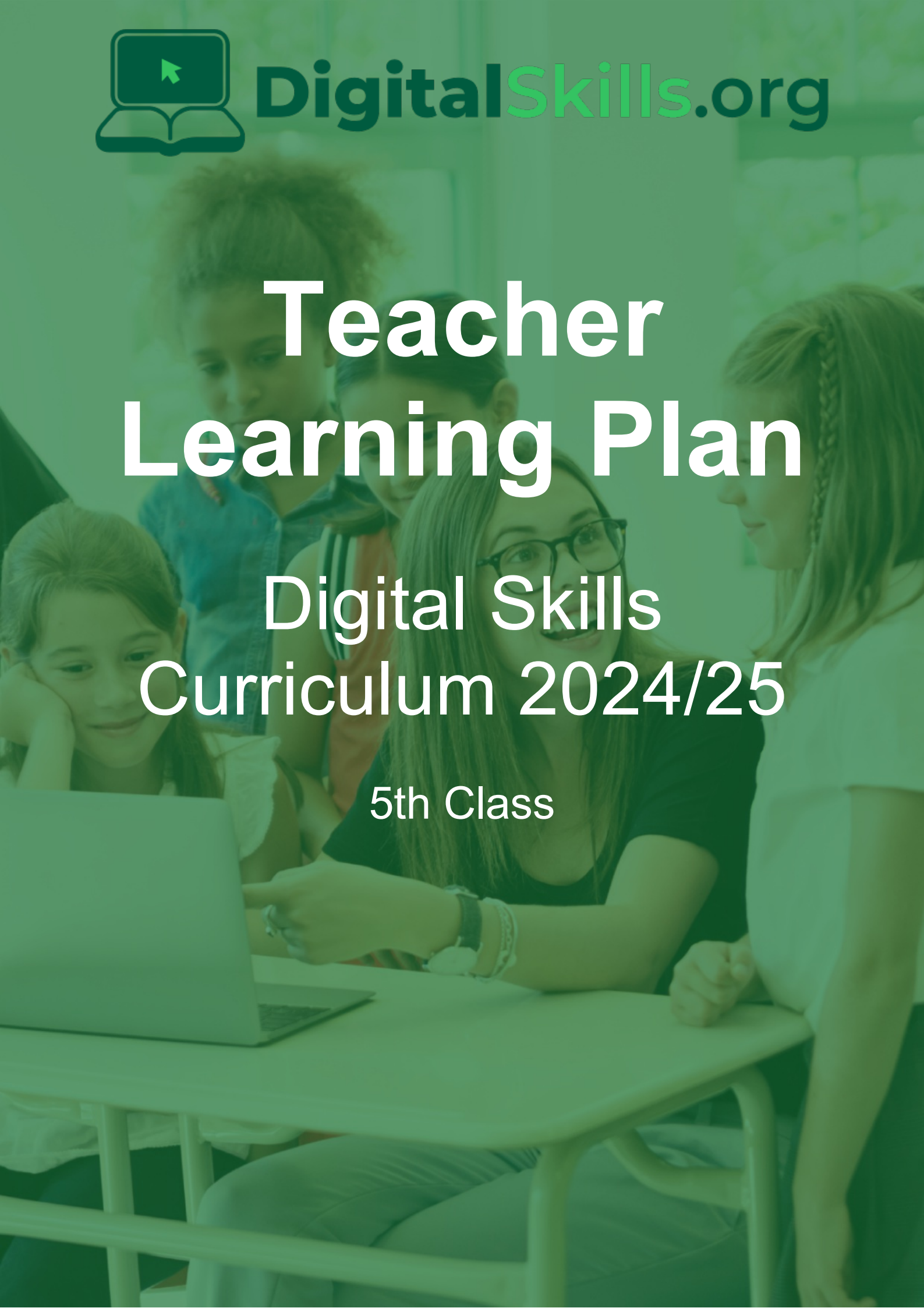


Table of Contents

How to Use This Learning Plan

This learning plan provides an overview of all the modules available for 5th Class, including their units, learning goals, and outcomes. Each module is designed to support both new and experienced teachers with easy-to-follow, step-by-step lessons.

Lesson Types

There are two types of lessons in the Digital Skills Curriculum:

- **Teacher-Led Lessons** – The teacher directs and leads students through the lesson, guiding them through the activities and discussions.
- **Teacher/Student-Led Lessons** – Teachers can choose to lead the lesson, or students can follow the step-by-step instructions to work through it independently.

Younger students require a fully guided approach, while older students often benefit from working at their own pace with teacher support as needed.

Flexible Curriculum Approach

Teachers have the flexibility to choose the modules that best fit their class needs. While there are enough lessons to cover a full school year, it is not necessary to complete all the modules. This allows teachers to tailor the learning experience to their students while ensuring they meet their educational goals.

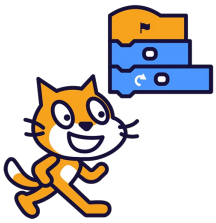
Student Access

Students log into the platform to access their lessons. They can follow the step-by-step instructions independently, or teachers can lead the lesson as needed.

Getting Started

1. **Review the Learning Plan:** Each module includes an overview of its goals, learning outcomes, lesson structure, and required resources. Start by familiarising yourself with the curriculum's scope.
2. **Plan Your Lessons:** Every lesson includes step-by-step guidance, accessible from your teacher dashboard. Adjust the pacing and delivery method based on your students' needs.
3. **Check Required Equipment:** Most lessons only require a laptop, Chromebook, or tablet. Some modules may include additional materials like microbits or LEDs. The required equipment is listed at the start of each module and each individual lesson.
4. **Support Student Learning:** Encourage students to work through the lessons. No prior coding experience is required—teachers can learn alongside their students.
5. **Use Assessments:** Each lesson includes a multiple-choice quiz to help assess student understanding and track progress.
6. **Need Help?:** We're always happy to answer your questions and give advice. You can contact our team at info@digitalskills.org or +44 020 4600 8710.

Module: Introduction to Coding



This module introduces students to the fundamentals of coding, starting with an overview of what coding is and its applications. Teachers should utilise visual aids and interactive discussions. The module progresses to hands-on experience with Scratch, a coding platform for creating games and animations. Teachers should familiarise themselves with Scratch and be prepared to assist students. The module culminates in students creating a Paddle Ball Game, reinforcing their understanding of moving sprites, changing backdrops, and using sensing blocks. Teachers should ensure students understand X and Y coordinates and Scratch coding blocks.

Duration	**Equipment**
2 weeks	Students can use any of these devices: • Chromebook/Laptop/PC • iPad/Tablet
Module Goals	**Module Outcomes**
1. Understand the concept of coding and its potential applications. 2. Gain proficiency in using Scratch for creating projects, including adding sprites and backdrops, and making sprites move. 3. Experiment with different code blocks in Scratch and learn from trial and error. 4. Create a Paddle Ball Game using Scratch, incorporating skills such as moving sprites, changing backdrops, and using sensing blocks. 5. Understand and apply the concepts of X and Y coordinates in the context of Scratch projects.	1. Understand the concept of coding and its applications. 2. Develop basic skills in Scratch, including creating projects, adding sprites and backdrops, and making sprites move. 3. Experiment with different code blocks in Scratch and learn from mistakes. 4. Create a Paddle Ball Game using Scratch, demonstrating the ability to move sprites, change backdrops, and use sensing blocks. 5. Understand and apply the concepts of X and Y coordinates in the context of Scratch coding.

Week 1

Lesson: Introduction to Coding

□ Beginner	□ 10 mins	□ Teacher/Student led	□ Student Quiz

If possible play the video in step 1 on a large screen for all your students to watch together. For steps 2 and 3 you should discuss and demonstrate these with your students.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
• Understand the concept of coding or programming as giving step-by-step instructions to a computer. • Identify examples of household items that contain computers and can be given instructions. • Recognize the importance of precise and correct order of instructions in coding. • Practice giving specific instructions in a sequential order to achieve a desired outcome.	• Define coding as the process of giving step-by-step instructions to a computer. • Identify at least three household items that contain computers and can be given instructions. • Explain the importance of precise and correct order of instructions in coding. • Demonstrate the ability to give specific instructions in the correct order to move from one point to another using a provided image.

Lesson: Scratch Tutorial

□ Beginner	□ 40 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

This lesson introduces students to Scratch, a coding platform for creating games and animations. Teachers should familiarise themselves with the Scratch website and its functionalities. The lesson guides students through creating a project, removing the default sprite, adding a new sprite, making it move, adjusting values, creating a loop, adding a backdrop, and encourages further exploration. Teachers should be prepared to assist with any technical difficulties and encourage experimentation.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Understand and navigate the Scratch coding platform. 2. Manipulate sprites by adding, removing, and controlling their movements. 3. Apply basic coding concepts such as loops and event triggers. 4. Modify code blocks to alter sprite behaviour. 5. Explore and experiment with various Scratch functionalities to create unique projects.	1. Identify Scratch as a coding platform for creating games, animations and projects. 2. Navigate and utilise the Scratch website interface. 3. Remove default sprites and add new ones from the sprite library. 4. Implement basic coding blocks to manipulate sprite movement. 5. Modify values within code blocks to alter sprite behaviour. 6. Create a loop within the code to repeat specific actions. 7. Add a backdrop from the library to enhance the visual aspect of the project. 8. Explore and experiment with various code blocks to diversify sprite actions.

Week 2

Lesson: Paddle Ball Game

□ Beginner	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare to guide students through creating a Paddle Ball Game using Scratch. They'll learn to move sprites, change backdrops, and use sensing blocks. They'll create a new Scratch project, add a paddle and a football sprite, position the ball, make it bounce, control the paddle, make the ball bounce off the paddle, add a backdrop, add a game over line and program the game over. Ensure students understand X and Y coordinates, and how to use the Scratch coding blocks.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Develop skills in using Scratch to create a simple game. 2. Understand and apply the concept of sprites and backdrops in Scratch. 3. Learn to control sprite movements using mouse input. 4. Implement game logic using conditional statements in Scratch. 5. Understand and apply the concept of X and Y coordinates to position sprites.	1. Manipulate sprites and backdrops in Scratch. 2. Utilise X and Y coordinates to position sprites. 3. Implement code to control sprite movement and interaction. 4. Use sensing blocks to detect sprite collision and mouse position. 5. Create a game over condition using colour detection.

Module: Coding and Interactive Creators

This module guides students through creating interactive games and animations using Scratch, a visual programming language. Teachers should prepare to guide students through each project, encouraging experimentation with code. Some lessons require specific resources, such as uploading images, and are not suitable for tablets or iPads. The module covers a range of programming concepts, including controlling movements, creating clones, detecting collisions, and using variables.

Duration	Equipment
8 weeks	Students can use any of these devices: • Chromebook/Laptop/PC • iPad/Tablet
Module Goals	**Module Outcomes**
1. Master the use of Scratch, a visual programming language, to create interactive games and animations. 2. Develop skills in controlling character movements, creating clones, and detecting collisions within a digital environment. 3. Understand and apply the concept of variables to track game scores and lives in a game scenario. 4. Learn to incorporate randomness in game elements to enhance unpredictability and engagement. 5. Gain proficiency in using keyboard and mouse inputs to control game characters and elements.	1. Develop an interactive game using Scratch, incorporating character movements, clone creation, and collision detection. 2. Design and implement a penalty shootout game, focusing on scoring mechanics. 3. Program keyboard arrow keys to control a car in a racing game, including creating custom racing tracks. 4. Create a music video in Scratch, integrating music and animation elements. 5. Develop a colour battle game using multiple sprite costumes and randomness. 6. Design a space shooter game, utilising mouse controls, sprite cloning, and variable tracking for score and lives. 7. Program a functional digital clock, accurately representing hours, minutes, and seconds. 8. Participate in build battles, demonstrating coding proficiency through various challenges.

Week 1

Lesson: Banana Jump

□ Beginner	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare to guide students through creating a game using Scratch, a visual programming language. The game involves a cat avoiding falling bananas. Students will learn to control character movements, create clones, and detect collisions. They will start by creating a new Scratch project, add a backdrop, animate the cat, and add a bananas sprite. They will then program the bananas to move across the screen and the cat to jump. Finally, they will program the game to stop when the cat touches a banana. Encourage students to experiment with the code.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Understand and apply the basics of Scratch programming including character movements, creating clones, and detecting collisions. 2. Create a new project in Scratch and add elements such as backdrops and sprites. 3. Use loops and conditional statements to animate sprites and control their movements. 4. Implement collision detection to trigger game events. 5. Develop problem-solving and computational thinking skills through game development.	1. Utilise Scratch to create and control character movements in a game. 2. Implement backdrop and sprite additions from the library in Scratch. 3. Apply loops and costume changes to animate sprites in Scratch. 4. Develop a game mechanic of jumping using key press events and coordinate changes. 5. Integrate collision detection to trigger game events in Scratch.

Week 2

Lesson: Penalty Shootout

□ Beginner	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

This lesson involves creating a 'Penalty Shootout' game using Scratch. Students will learn to create a new Scratch project, download and add a goal backdrop, and add the Soccer Ball sprite. They will code the ball's movement, add the Casey sprite as the goalkeeper, and code the goalkeeper's dive. They will also learn to detect if the goalkeeper makes a save, if the ball hits the post, and if a goal is scored. This lesson will enhance their coding skills and understanding of game mechanics.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Learning Goals	Learning Outcomes
1. Develop skills in creating and managing a new Scratch project. 2. Learn to download and add backdrops to a Scratch project. 3. Understand how to add and manipulate sprites in Scratch, including movement and size changes. 4. Gain knowledge on how to use mouse clicks to interact with a Scratch project. 5. Acquire the ability to implement conditional statements in Scratch to detect interactions between sprites and backdrop.	1. Develop a new Scratch project and manipulate sprites and backdrops. 2. Implement code to control sprite size and position in Scratch. 3. Use mouse-click events to trigger actions in Scratch. 4. Apply repeat loops and conditional statements in Scratch to control sprite movement. 5. Design and implement a scoring system in a Scratch game.

Week 3

Lesson: Racing Car

□ Beginner	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

This lesson involves creating a racing car game using Scratch. Teachers should familiarise themselves with Scratch and its features. The lesson starts with creating a new Scratch project and adding a car sprite. The car is then resized and a racing track is drawn. The car is placed on the track and a speed variable is created. The car's location is detected and it is programmed to move forwards, backwards, left, and right. Finally, students test drive their car. Teachers should ensure students understand each step before moving on.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Develop skills in creating and managing a new Scratch project. 2. Understand and apply the process of adding and modifying sprites in Scratch. 3. Gain knowledge in drawing and editing backdrops in Scratch. 4. Learn to create and manipulate variables to control sprite properties. 5. Acquire skills in programming sprite movements and interactions using Scratch blocks.	1. Develop a new Scratch project and remove the default sprite. 2. Upload a provided car sprite into the Scratch project. 3. Resize the car sprite to 10% of its original size using Scratch code. 4. Draw a racing track using the backdrop editor in Scratch. 5. Position the car sprite on the track and code its starting position and direction. 6. Create a 'speed' variable to control the car's speed. 7. Program the car to detect its location and adjust its speed accordingly. 8. Code the car to move forwards and backwards using the up and down arrow keys. 9. Code the car to turn left and right using the left and right arrow keys. 10. Test drive the car using the programmed controls and observe its speed changes on different surfaces.

Week 4

Lesson: Music Video

□ Intermediate	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

In this lesson, students will create a music video using Scratch. They'll start by setting up a new project, then choose and download a music file. They'll learn to add the music to their project and play it. They'll also choose backdrops and sprites, programming them to change every few seconds. Finally, they'll get creative, using code to add effects and make their video unique. Ensure students understand the coding concepts and encourage creativity.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Develop skills in creating a new project on Scratch. 2. Understand how to select and incorporate music into a Scratch project. 3. Learn to code music playback in response to a user action. 4. Gain knowledge on adding and manipulating backdrops for visual appeal. 5. Learn to add and program sprites for interactive elements. 6. Enhance creativity by exploring different coding techniques to create unique visual effects.	1. Create and manage a new Scratch project. 2. Select and download suitable music files for the project. 3. Implement code to play the chosen music in the Scratch project. 4. Choose and integrate appropriate backdrops into the project. 5. Add and program sprites to perform various actions in the project. 6. Apply creative coding techniques to enhance the visual appeal of the project.

Week 5

Lesson: Red v Green v Blue

□ Intermediate	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare to guide students in creating a new Scratch project, focusing on creating and manipulating sprites. Students will create a red dot sprite, then duplicate and recolour it to create green and blue versions. They will then learn to create a 'count' variable and use it to clone the dot sprite 100 times. The clones will be coded to appear randomly on the screen, move around, and 'infect' each other, changing colours according to a set rule. Encourage students to think of ways to improve the project.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Develop skills in creating and managing a new Scratch project. 2. Acquire knowledge in creating and modifying sprites and costumes in Scratch. 3. Understand and apply the concept of variables in Scratch programming. 4. Learn to code sprite clones and manage their behaviour in Scratch. 5. Enhance problem-solving skills by improving and customising the project.	1. Develop a new Scratch project and remove the default sprite. 2. Create a red dot sprite using the sprite editor. 3. Generate green and blue costumes for the sprite. 4. Create a 'count' variable for counting up to 100. 5. Code the sprite to clone itself 100 times using the 'count' variable. 6. Programme each clone to randomly select a costume and position, and then appear on the screen. 7. Code the dots to move in a random direction and bounce off the screen edges. 8. Programme the dots to change colour when they touch another dot, according to specific rules. 9. Improve the project based on personal ideas and creativity.

Week 6

Lesson: Space Shooter

□ Advanced	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

This lesson involves creating a 'Space Shooter' game using Scratch. Students will learn to create a new project, add a backdrop, create and set up variables for score and lives, add and program sprites (Rocketship, Ball, Balloon1), detect mouse clicks, create clones of sprites, and implement game logic for scoring points and losing lives. The lesson concludes with testing the game and discussing potential improvements. Teachers should familiarise themselves with Scratch and the game mechanics to effectively guide the students.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Develop skills in creating and managing a new Scratch project. 2. Understand and apply the concept of variables in game development. 3. Gain proficiency in adding and programming sprites in Scratch. 4. Learn to implement game mechanics such as scoring and lives. 5. Enhance problem-solving skills by improving and modifying the game.	1. Construct a new Scratch project and remove the default sprite. 2. Integrate the Stars backdrop into the project. 3. Create and utilise 'score' and 'lives' variables to track game progress. 4. Initiate the variables at the start of the game. 5. Add and modify the Rocketship sprite to the project. 6. Program the Rocketship sprite to follow mouse pointer and respond to up arrow key. 7. Add and adjust the Ball sprite to the project. 8. Add and adjust the Balloon1 sprite to the project. 9. Implement code to detect a mouse click and broadcast a message. 10. Program the Ball sprite to fire upon receiving the broadcasted message. 11. Make the Balloon1 sprite appear randomly on the screen and float in a random direction. 12. Score a point when a balloon is hit by a ball. 13. Decrease a life when a balloon touches the Rocketship sprite. 14. End the game when no lives are left. 15. Play and evaluate the game, considering potential improvements.

Week 7

Lesson: Make a Clock

□ Advanced	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

This lesson involves creating a functional clock using Scratch. Teachers should familiarise themselves with the Scratch platform and the specific code blocks used in this lesson. The lesson involves creating a new Scratch project, uploading a clock face, drawing and programming the seconds, minutes, and hours hands, and finally, adding a challenge to turn the clock into an alarm clock. Emphasise the importance of starting the hands from the centre and the mathematical calculations involved in programming the hands.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Understand and apply the basics of Scratch programming. 2. Create and manipulate sprites in Scratch. 3. Understand the concept of time and how it is represented on a clock. 4. Apply mathematical concepts to program the movement of clock hands. 5. Extend programming skills by adding an alarm feature to the clock.	1. Create a new Scratch project, demonstrating understanding of the Scratch interface and project creation process. 2. Upload a clock face image to the Scratch project, applying knowledge of how to import external resources. 3. Draw and position the seconds hand on the clock face, showing proficiency in the Scratch drawing tools and sprite positioning. 4. Program the seconds hand to move according to the current time, demonstrating understanding of Scratch's time blocks and basic mathematical operations. 5. Draw and position the minutes hand, further applying the Scratch drawing tools and sprite positioning. 6. Program the minutes hand to move according to the current time, extending the use of Scratch's time blocks and mathematical operations. 7. Draw and position the hour hand, continuing to apply the Scratch drawing tools and sprite positioning. 8. Program the hour hand to move according to the current time, demonstrating a deeper understanding of Scratch's time blocks, mathematical operations, and the concept of modulus. 9. Refine the hour hand movement to account for the passing minutes, showing an ability to enhance the accuracy of the clock's functionality. 10. Extend the project by turning the clock into an alarm clock, demonstrating creativity and problem-solving skills in Scratch programming.

Week 8

Lesson: Build Battles

□ Advanced	□ 60 mins	□ Teacher led

Prepare to facilitate a series of build battles using Scratch. Start with an introduction, then guide students through three timed challenges: a 10-minute space-themed project, a 5-minute sports-themed project, and a 1-minute open-themed project. Ensure students understand the time limits and how to submit their projects. Be ready to manage the sharing and judging of projects.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Develop proficiency in using Scratch for quick project creation. 2. Apply creative thinking to design and execute projects under time constraints. 3. Adapt to different themes and incorporate them into coding projects. 4. Improve presentation skills through project sharing and discussion. 5. Enhance competitive spirit and teamwork through build battles.	1. Create a Scratch project with a space theme within a 10-minute timeframe. 2. Present the created project to peers within a 2-minute timeframe. 3. Develop a Scratch project with a sports theme within a 5-minute timeframe. 4. Present the sports-themed project to peers within a 2-minute timeframe. 5. Construct a Scratch project with any theme within a 1-minute timeframe and present it to peers within a 2-minute timeframe.

Module: Game Design Studio

This module guides students through the creation of various games using MakeCode Arcade. Each lesson is hands-on and interactive, allowing students to learn by doing. Teachers should ensure students understand each step before moving on, and encourage experimentation with the code to add new features to the games. The module concludes with a 'Brainstorming Blast' lesson where students brainstorm and create their own projects, fostering creativity and teamwork.

Duration	**Equipment**
8 weeks	Students can use any of these devices: • Chromebook/Laptop/PC • iPad/Tablet
Module Goals	**Module Outcomes**
1. Master the use of MakeCode Arcade for creating interactive games, including sprite creation, movement controls, and game mechanics. 2. Develop proficiency in designing and implementing game elements such as characters, maps, obstacles, and goals. 3. Understand and apply coding concepts to control game dynamics, including collision detection, scoring systems, and win conditions. 4. Enhance problem-solving skills through the creation and debugging of complex game projects. 5. Strengthen creativity and teamwork abilities through collaborative game design and development projects.	1. Create and manipulate sprites in MakeCode Arcade, including movement and interaction. 2. Design and implement game mechanics such as lives, collision effects, scoring, and game termination. 3. Develop complex games with features like mazes, timers, and AI behaviours. 4. Apply creativity and coding skills to design original arcade projects. 5. Present and critique game projects, demonstrating teamwork and constructive feedback.

Week 1

Lesson: First Arcade Project

□ Beginner	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

This lesson guides students through creating their first arcade project using MakeCode Arcade. They will learn about the code editor, how to create a new project, add a sprite, choose a sprite from the gallery, move the sprite, draw a tile map, draw walls, make the camera follow the sprite, add projectiles, set their direction and speed, detect overlap, lose a life, and finally, send the code to a handheld device. The lesson is hands-on and interactive, allowing students to learn by doing.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Understand and utilise MakeCode Arcade for creating games. 2. Manipulate the Code Editor to build and modify game elements. 3. Create and customise sprites for use in a game. 4. Develop a tile map and implement walls for game navigation. 5. Implement game mechanics such as projectiles, sprite movement, and life count.	1. Understand the functions and features of MakeCode Arcade. 2. Use the MakeCode Arcade code editor to create a new project and add a sprite. 3. Manipulate the sprite's movements using the direction buttons in the simulator. 4. Create and edit a tile map, including drawing walls and setting the camera to follow the sprite. 5. Design and implement projectiles, including setting their direction and speed, and programming responses to overlaps with the player's sprite.

Week 2

Lesson: Space Dodge

□ Beginner	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare to guide students through creating a 'Space Dodge' game using MakeCode Arcade. Students will learn to create a new project, design a spaceship sprite, control the spaceship, set the number of lives, create asteroids, set their position and velocity, auto destroy them when they move off the screen, and detect when an asteroid hits the spaceship. Ensure students understand the importance of correct variable selection and the effect of different values in the code.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Develop skills in using MakeCode Arcade to create a game. 2. Understand and apply the concept of sprites in game development. 3. Implement controls for game characters using code. 4. Apply the concept of randomisation in game elements for varied gameplay. 5. Understand and implement game mechanics such as collision detection and life count.	1. Design and create a spaceship sprite using MakeCode Arcade. 2. Control the spaceship's movement with arrow keys and set boundaries to prevent it from going off the screen. 3. Set the number of lives for the spaceship. 4. Create and design asteroid sprites that appear at random positions on the screen. 5. Set the velocity of the asteroids to make them move across the screen. 6. Implement a function to auto-destroy asteroids when they move off the screen. 7. Program the game to detect when an asteroid hits the spaceship, causing it to lose a life and trigger a camera shake effect.

Week 3

Lesson: Bat Battle

□ Beginner	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

This lesson guides students through creating a game using MakeCode Arcade. They will learn to create and control a player sprite, generate enemy sprites, and program interactions between them. The lesson includes coding for scoring points and ending the game. Teachers should ensure students understand each step before moving on, and encourage experimentation with the code to add new features to the game.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Develop skills in using MakeCode Arcade to create an interactive game. 2. Understand how to create, control, and position player and enemy sprites. 3. Learn to program game interactions such as shooting projectiles and detecting overlaps. 4. Gain knowledge on how to keep score and end the game in MakeCode Arcade. 5. Enhance problem-solving and debugging skills by experimenting with the code and adding new features.	1. Create and control a player sprite in MakeCode Arcade. 2. Generate enemy sprites at random positions. 3. Program interactions between player and enemy sprites. 4. Implement a scoring system for hitting targets. 5. End the game when an enemy sprite hits the player sprite.

Week 4

Lesson: Prison Break

□ Intermediate	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare to guide students through the creation of a 'Prison Break' game using MakeCode Arcade. They will design a character, create a maze, set a goal and add a timer for challenge. Ensure familiarity with the MakeCode Arcade interface, sprite creation, and basic coding concepts. Encourage creativity in maze and character design, and emphasise the importance of testing at each stage. Celebrate their accomplishment in creating a complex, interactive game.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Develop skills in creating and controlling a character sprite in MakeCode Arcade. 2. Understand and apply the concept of designing a maze in a game environment. 3. Learn to set and implement game goals, such as reaching a specific location. 4. Gain knowledge on how to add and use a timer to increase game difficulty. 5. Enhance problem-solving skills through the creation and navigation of a maze game.	1. Create a new project using MakeCode Arcade. 2. Design and implement a sprite character for the game. 3. Enable character movement using joystick or keyboard arrow keys. 4. Design a maze using the tile map editor and set wall boundaries. 5. Implement camera follow feature to track sprite movement. 6. Set a goal tile within the maze for the character to reach. 7. Implement a winning condition when the character reaches the goal tile. 8. Add a countdown timer to increase game challenge.

Week 5

Lesson: Arcade Build Battles

□ Intermediate	□ 60 mins	□ Teacher led

Prepare to facilitate a series of build battles where students create coding projects within set time limits. Ensure students understand the time constraints and how to share their projects. The battles will vary in length and complexity, from a 15-minute arcade project, to a 5-minute themed project, and finally a 1-minute character design task.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Develop and apply coding skills to create an Arcade project within a specified time limit. 2. Design and create a unique character in Arcade within a one-minute timeframe. 3. Enhance project management skills by adhering to strict time constraints during project development. 4. Improve communication skills by sharing and presenting created projects to peers. 5. Cultivate a competitive spirit and teamwork through participation in build battles.	1. Create an Arcade project within a 15-minute time frame. 2. Share the created project within a 2-minute time frame. 3. Develop an Arcade project with any theme within a 5-minute time frame. 4. Design a character in Arcade within a 1-minute time frame. 5. Share the designed character within a 2-minute time frame.

Week 6

Lesson: Dino Jump

□ Advanced	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

In this lesson, students will create an interactive game called 'Dino Jump' using MakeCode Arcade. They will learn how to draw a map, create a dino character, make it jump, add obstacles, keep score, and determine when the game is won. The lesson involves creating a new arcade project, drawing the map, creating the dino sprite, adding gravity and movement to the dino, making it jump, adding cactuses as obstacles, detecting collision with a tree, keeping score, and setting a win condition. The lesson concludes with a play and review session.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Develop skills in creating an interactive game using MakeCode Arcade. 2. Understand how to draw a map, create a character, and add movement and gravity effects. 3. Learn to add obstacles and implement collision detection for game over scenarios. 4. Gain knowledge on how to keep score and determine winning conditions in a game. 5. Reflect on the game design process and the elements involved in creating an engaging game.	1. Create a new arcade project on the MakeCode Arcade website. 2. Draw a map for the Dino Jump game, including ground tiles, walls, and a finish tile. 3. Create a dino sprite using the sprite editor and code. 4. Implement gravity and movement for the dino sprite, making it fall to the ground and move forwards through the map. 5. Program the A button to make the dino jump when it is touching the ground. 6. Add trees to the map as obstacles for the dino to jump over. 7. Implement game over functionality when the dino sprite hits a tree. 8. Keep score based on how long the player can go without hitting a tree. 9. Detect when the player reaches the end of the game and display a winning screen. 10. Play and review the Dino Jump game, reflecting on the elements of game design learned during the lesson.

Week 7

Lesson: Monster Battle Arena

□ Advanced	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare to guide students through creating a 'Monster Battle Arena' game using MakeCode Arcade. They will learn to create player-controlled and AI-controlled sprites, implement combat mechanics, health systems, and AI behaviours. Students will also learn to create a new project, design sprites, make the monster move, implement a health system, create a combat system, and determine the winner. Encourage creativity and experimentation with the game's features.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Develop a player-controlled sprite and an AI-controlled monster in a game using MakeCode Arcade. 2. Create and manage a new project in MakeCode Arcade. 3. Implement a health system for player and monster sprites. 4. Develop a combat system where player and monster sprites can inflict damage on each other. 5. Implement a win/lose condition based on the health of player and monster sprites.	1. Create and control a player sprite using MakeCode Arcade, ensuring it moves smoothly within the screen boundaries. 2. Program an AI-controlled monster sprite with randomized movement, simulating intelligent behavior. 3. Implement a health system that tracks and displays the health values of both the player and the monster during the game. 4. Develop a combat system that reduces player health upon collision with the monster and allows the player to shoot projectiles at the monster. 5. Determine the game's winner by programming conditions that end the game when either the player's or monster's health reaches zero.

Week 8

Lesson: Game Lab

□ Advanced	□ 60 mins	□ Teacher led

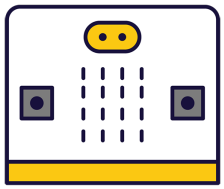
In this lesson, 'Brainstorming Blast', students will brainstorm ideas for their own MakeCode Arcade projects. Start by introducing the lesson and demonstrating a simple MakeCode Arcade project. Divide students into small groups for brainstorming, reminding them of the importance of teamwork. Set a timer for the brainstorming session and encourage students to keep their ideas simple and achievable. After brainstorming, each group will present their project idea and receive feedback from the class. Students will then create their projects in MakeCode Arcade, with the teacher providing assistance as needed. Finally, conduct a 'Show and Tell' session where each group presents their project to the class.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Develop and articulate original ideas for a simple MakeCode Arcade project. 2. Collaborate effectively in small groups to brainstorm and refine project ideas. 3. Present project ideas clearly and constructively, incorporating feedback from peers and teachers. 4. Apply basic MakeCode Arcade blocks to create a simple game or interactive project. 5. Reflect on the process of project creation, identifying learning points and areas for improvement.	1. Brainstorm and develop a simple, achievable idea for a MakeCode Arcade project. 2. Collaborate effectively within a group to discuss and refine project ideas. 3. Present a project idea to the class, explaining the concept, sprites, and tile maps planned for use. 4. Constructively receive and incorporate feedback to improve the project plan. 5. Create a MakeCode Arcade project based on the brainstormed idea, demonstrating basic proficiency in using MakeCode Arcade blocks.

Module: Microbit Masterclass



This module will guide teachers through an engaging exploration of microbits, pocket-sized programmable computers. Teachers will introduce students to coding, creating projects, and programming microbits to display messages, play melodies, and respond to movement. Subsequent lessons will involve designing games, creating an alarm system, designing a weather station, and even turning a microbit into a pet. Teachers should ensure students understand the utility of variables, functions, and conditionals, and encourage experimentation with different blocks from the toolbox.

Duration	Equipment
8 weeks	Students can use any of these devices: • Chromebook/Laptop/PC Required Equipment: • Microbit

Module Goals	Module Outcomes
1. Master the fundamentals of microbits, including project creation, code addition and deletion, and programming for various responses. 2. Develop skills in designing and coding interactive games using microbits, focusing on reaction time measurement. 3. Apply knowledge of microbits to create a Magic 8 Ball, enhancing understanding of random response generation. 4. Design and implement a microbits-based alarm system, demonstrating proficiency in using sensors and setting threshold values. 5. Create a microbits weather station, showcasing ability to display sensor readings based on button presses. 6. Utilise radio signals to detect proximity between two microbits, demonstrating understanding of radio group setup and message transmission. 7. Design and code a guessing game on microbits, reinforcing skills in variable creation, button programming, and game logic implementation. 8. Transform a microbit into an interactive pet, demonstrating creativity and understanding of user interaction.	1. Program a microbit to display messages, react to button presses, show icons, play melodies, and respond to movement. 2. Design and create a reaction timer game using a microbit, improving reaction times to random visual prompts. 3. Develop a Magic 8 Ball project using a microbit, providing random responses to questions upon shaking. 4. Create a microbit-based alarm system, utilising sensors to detect movement and sound and activate the system. 5. Design a microbit weather station, displaying different sensor readings based on button presses. 6. Use radio signals between two microbits to detect proximity, creating a 'Microbit Finder'. 7. Develop a 'Higher or Lower' game on a microbit, programming button inputs and implementing game logic. 8. Transform a microbit into an interactive pet with different emotions based on user interaction.

Week 1

Lesson: Exploring Microbits

□ Beginner	□ 60 mins	□ Teacher/Student led	□ Student Quiz

Prepare to introduce students to the world of microbits, a pocket-sized programmable computer. The lesson will involve creating a new project on the MakeCode for microbit website, familiarising with the project editor, and writing code to display numbers, names, and icons. Students will also learn to delete code, connect their microbits to their computers, and program their microbits to play music. The lesson concludes with an exploration phase where students can experiment with different blocks from the toolbox.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit

Learning Goals	Learning Outcomes
1. Understand the basic functionality and features of a microbit. 2. Create a new project using the MakeCode for microbit website. 3. Use the Project Editor to write and simulate code. 4. Program the microbit to display numbers and text on its LED grid. 5. Program the microbit to respond to button presses with specific actions.	1. Identify the functions and capabilities of a microbit. 2. Create a new project on the MakeCode for microbit website. 3. Understand the layout and functions of the Project Editor. 4. Write and execute code to display numbers and names on the microbit. 5. Program the microbit to respond to button presses with specific displays. 6. Connect and download code to an actual microbit device. 7. Compose and program a melody to play on the microbit. 8. Explore and experiment with different coding blocks and functions.

Week 2

Lesson: Reaction Timer

□ Beginner	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare to guide students in creating a 'Reaction Timer' project using Micro:bit. They'll start by setting up a new project, then create a welcome message and a countdown. Next, they'll add a random delay to make the game unpredictable. They'll create variables to store time stamps, and finally, record the player's reaction time. Familiarise yourself with the code snippets provided.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit

Learning Goals	Learning Outcomes
1. Develop skills in creating and managing a new project on the Micro:bit platform. 2. Acquire knowledge on how to create and display messages using code. 3. Understand and apply the concept of countdowns and delays in programming. 4. Learn to create and utilise variables for storing time stamps. 5. Gain proficiency in recording and displaying user interactions in real-time.	1. Develop a new project using the Micro:bit website. 2. Construct a welcome message to display upon powering on the Microbit. 3. Create a countdown sequence with visual cues using code. 4. Implement a random delay function in the game for unpredictability. 5. Create and utilise variables to store time stamps. 6. Record and display player reaction time upon button press.

Week 3

Lesson: Microbit Magic 8 Ball

□ Beginner	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare to guide students through creating a new Microbit project on the makecode.microbit.org website. They will learn to display the number 8, detect a shake gesture, and create a variable called 'Random Number'. They will set this variable to a random number between 0 and 4 when the Microbit is shaken. Each number will correspond to a different response from the magic 8 ball. Students will then check the value of the variable and add custom messages for each condition. Encourage creativity in crafting these messages. They can then expand the range of the 'Random Number' variable to add more messages. Finally, they will download the code to their Microbit and play.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit

Learning Goals	Learning Outcomes
1. Understand and utilise the Microbit project creation process. 2. Implement code to display a number on the Microbit. 3. Apply gesture detection functionality in Microbit programming. 4. Create and manipulate variables to generate random numbers. 5. Develop conditional statements to check variable values and display corresponding messages.	1. Develop a new Microbit project using makecode.microbit.org. 2. Display the number 8 on the Microbit when it is not being shaken. 3. Program the Microbit to detect a shake gesture. 4. Create and utilise a variable called 'Random Number' to generate a random number between 0 and 4 when the Microbit is shaken. 5. Check the value of the 'Random Number' variable and associate it with a specific response. 6. Add custom messages for each possible value of the 'Random Number' variable. 7. Expand the range of the 'Random Number' variable to add more messages. 8. Download and implement the code on a physical Microbit device.

Week 4

Lesson: Creating a Microbits Alarm System

□ Intermediate	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare for the 'Creating a Microbits Alarm System' lesson by familiarising yourself with the Microbits MakeCode editor. Understand the process of creating new projects and setting up variables. Grasp the concept of arming and disarming the alarm system using button inputs. Understand how to define a function for the alarm and set up triggers based on sound and light thresholds. Finally, be ready to guide students through testing their alarm systems in a simulator or on a physical device.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit

Learning Goals	Learning Outcomes
1. Understand and apply the concept of variables in Microbits programming. 2. Develop skills to use input functions for button presses on the Microbit device. 3. Learn to create and utilise functions for specific tasks in coding. 4. Gain knowledge on using sensor inputs (sound and light) to trigger events. 5. Apply testing and debugging skills to ensure the functionality of the Microbits alarm system.	1. Develop a new project using the Microbits MakeCode editor. 2. Establish sound and light threshold variables for the alarm system. 3. Implement a function to arm the alarm system using the A button on the Microbit. 4. Design a function to disarm the alarm system using the B button on the Microbit. 5. Define a function to sound and flash the alarm when triggered. 6. Set up alarm triggers that monitor sensor values and activate the alarm when thresholds are crossed. 7. Test the alarm system in a simulator and on a physical Microbits device.

Week 5

Lesson: Designing a Microbits Weather Station

□ Intermediate	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare for this lesson by familiarising yourself with the MakeCode for Microbit platform and the coding language used. Understand the purpose of variables and how they can be initialised and manipulated. Be prepared to guide students through the process of creating a new project, configuring buttons and sensors, creating a 'forever' loop, and testing their program. Encourage reflection on the learning process and potential applications of the skills learned.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit

Learning Goals	Learning Outcomes
1. Understand how to create a new project on MakeCode for Microbit. 2. Learn to declare and initialise variables in a Microbit project. 3. Gain skills in configuring Microbit buttons and sound detection. 4. Develop the ability to create a 'forever' loop and display sensor data. 5. Reflect on the learning process and consider potential applications of Microbit sensors.	1. Develop a new project using MakeCode for Microbit. 2. Declare and initialise variables 'mode' and 'reading' for sensor data display and storage. 3. Configure Button A to set 'mode' to 1 when pressed. 4. Configure Button B to set 'mode' to 2 when pressed. 5. Configure Buttons A and B together to set 'mode' to 3 when pressed simultaneously. 6. Configure the Microbit to switch to mode 4 when a loud sound is detected. 7. Create a 'forever' loop to check the value of 'mode' and display the relevant sensor data. 8. Test the program using a physical Microbit or the simulator on the MakeCode website. 9. Reflect on the learning process, understanding how the different sensors on the Microbit work and potential other projects.

Week 6

Lesson: Microbit Finder

□ Advanced	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare two Microbits and ensure one is portable. The lesson involves creating a code to be downloaded onto both Microbits, using A and B buttons to set 'lost' and 'finder' Microbits. The project requires creating two variables, setting a radio group for communication, programming buttons to set modes, sending and receiving messages, and displaying proximity. The code is then downloaded onto both Microbits for a practical demonstration.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit

Learning Goals	Learning Outcomes
1. Understand and utilise Microbit's radio communication feature. 2. Create and manipulate variables in a Microbit project. 3. Program Microbit's buttons to perform specific actions. 4. Interpret signal strength to determine proximity between two Microbits. 5. Apply coding skills to create a practical Microbit application.	1. Programme two Microbits to act as a 'finder' and a 'lost' device using A and B buttons. 2. Create and utilise 'mode' and 'signal' variables to control Microbit actions. 3. Set up a radio group for Microbits to communicate with each other. 4. Code the 'lost' Microbit to continuously send a signal for the 'finder' Microbit to detect. 5. Interpret the signal strength of the received message to determine the proximity of the 'lost' Microbit.

Week 7

Lesson: Microbit Pet

□ Advanced	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

In this lesson, students will transform their Microbits into interactive pets. They will use emoji icons and sounds to make the Microbits seem lifelike, programming them to respond to different actions such as shaking, touching, and flipping. Students will create functions for different states of the pet, like happy, sad, hungry, bored, and asleep. They will also learn to use the Microbit's sensors to detect these actions. The lesson involves coding in the Microbit's online editor, testing the code in a simulator, and finally downloading it onto their Microbits.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit

Learning Goals	Learning Outcomes
1. Develop skills in creating and using functions in Microbit programming. 2. Understand and apply the use of different sensors on the Microbit device. 3. Gain knowledge in programming interactive responses using sound and visual cues. 4. Learn to use random time intervals in programming for unpredictable outcomes. 5. Enhance problem-solving skills by debugging and testing code in a simulator and on a physical device.	1. Program Microbit to display different emoji icons and sounds to simulate pet behaviours. 2. Create a new Microbit project using the provided website. 3. Develop functions such as 'happy', 'feedme', and 'play' to control pet behaviours. 4. Implement gesture controls to interact with the Microbit pet, such as shaking, flipping, and touching the logo. 5. Test the programmed Microbit pet in the simulator and download the code onto a physical Microbit.

Week 8

Lesson: Microbit Lab

□ Advanced	□ 60 mins	□ Teacher led

Prepare to introduce the concept of Microbit projects, demonstrating a simple LED pattern to inspire creativity. Organise students into small groups for brainstorming, emphasising teamwork and achievable project ideas. Facilitate a feedback session after idea presentations, guiding project simplification if necessary. Assist during project creation, encouraging peer support and discovery sharing. Finally, conduct a 'Show and Tell' session, celebrating student effort and creativity, reinforcing learning objectives and the importance of teamwork.

Students can use any of these devices (and can share if necessary):

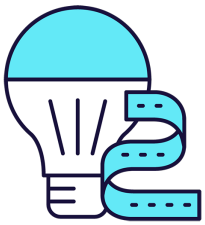
- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit

Learning Goals	Learning Outcomes
1. Develop creative and achievable project ideas using basic Microbit blocks. 2. Collaborate effectively in small groups to brainstorm, plan and execute a Microbit project. 3. Present project ideas clearly and receive feedback constructively. 4. Apply problem-solving skills to create a Microbit project based on the brainstormed idea. 5. Reflect on the project creation process, discussing changes made, challenges faced, and skills learned.	1. Brainstorm and develop a simple Microbit project idea in a group setting. 2. Present the project idea to the class, explaining the planned LED patterns and inputs. 3. Receive, incorporate, and respond to feedback on the project idea. 4. Create a Microbit project based on the brainstormed idea, using basic Microbit blocks. 5. Present the final Microbit project to the class, explaining the coding process and any changes made during the project creation.

Module: Exploring Electronics and Light



This module explores the exciting world of electronics and light, using Microbit and LED strips. Teachers will guide students through creating colourful displays, sound-activated lights, visual thermometers, and even a precision game. The module encourages creativity, problem-solving, and teamwork, with students brainstorming and implementing their own Microbit projects. Teachers should ensure students understand each step and concept before progressing, and provide assistance during the project creation stages.

Duration	Equipment
8 weeks	Students can use any of these devices: • Chromebook/Laptop/PC Required Equipment: • LED Strip with crocodile clips • Microbit
Module Goals	**Module Outcomes**
1. Understand and apply the principles of programming LED strips using Microbit projects. 2. Develop skills to create interactive LED displays that respond to sound and temperature changes. 3. Design and implement a game using LED strip and Microbit programming. 4. Enhance creativity and problem-solving skills through the design of LED flags and stacking effects. 5. Apply teamwork and project management skills in brainstorming and executing a group Microbit project.	1. Programme a strip of LEDs to display colourful patterns using Microbit. 2. Design and implement an LED Strip Clapper that responds to sound, specifically a clap, to turn on and off. 3. Convert an LED strip into a visual thermometer that lights up and changes colour according to the current temperature. 4. Create a voice-activated 'Shooting Stars' display using an LED strip and Microbit. 5. Design and code tricolour flags using LED strips. 6. Create a stacking effect on an LED strip, controlled by Microbit, with the ability to increase and decrease the size of the stack. 7. Develop an LED Strip Precision Game that involves timing and accuracy. 8. Brainstorm, design, and implement a simple Microbit project in a team, demonstrating creativity and teamwork.

Week 1

Lesson: Microbit LED Strip

□ Beginner	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare to guide students through programming a 30 LED strip using Microbits. Ensure understanding of creating a new Microbit project and adding the neopixel extension. Facilitate the setup of the LED strip and programming it to turn red. Assist with downloading the project onto the Microbit. Encourage creativity when programming the strip to show a rainbow of colours and rotating the rainbow. Finally, encourage exploration of other code blocks in the Neopixel toolbox.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- LED Strip with crocodile clips

Learning Goals	Learning Outcomes
1. Understand and apply the process of programming a strip of LEDs using Microbits. 2. Develop skills in creating a new Microbit project and adding the necessary extensions. 3. Gain proficiency in setting up and programming the LED strip to display various colours. 4. Learn to download and implement the project on Microbits, observing the effects on the LED strip. 5. Explore and experiment with different code blocks in the Neopixel toolbox for creative lighting effects.	1. Program a strip of 30 LEDs to light up in different ways using Microbits. 2. Create a new Microbit project and add the neopixel extension. 3. Set up the LED strip and interact with it using a variable. 4. Program the A button on the Microbit to turn all the LEDs red. 5. Program the LED strip to show a rainbow of colours when the Microbit turns on.

Week 2

Lesson: LED Strip Clapper

□ Beginner	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

In this lesson, students will create an LED Strip Clapper using a Microbit project. They will add the neopixel extension, set up the LED strip, and create an 'on' variable. The lesson will guide them to detect a clap, turning the LED strip on and off accordingly. They will download their code onto their microbit, connect it to the LED strip, and explore further improvements. Familiarity with Microbit and basic coding is beneficial.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- LED Strip with crocodile clips

Learning Goals	Learning Outcomes
1. Develop skills in creating and managing a new Microbit project. 2. Understand and apply the neopixel extension for programming an LED strip. 3. Learn to set up and interact with the LED strip using variables. 4. Gain knowledge on creating and manipulating variables to control the state of the LED strip. 5. Develop the ability to detect sound inputs and use them to trigger changes in the LED strip's state.	1. Develop a new Microbit project using makecode.microbit.org. 2. Integrate the neopixel extension into the project for LED strip programming. 3. Establish a variable for the LED strip and set its value to 30. 4. Create an 'on' variable to control the LED strip's state. 5. Implement a sound detection feature to trigger the LED strip's state change.

Week 3

Lesson: Microbit LED Strip Thermometer

□ Beginner	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare for this lesson by familiarising yourself with the Microbit project platform and the neopixel extension. Understand how to set up the LED strip and how to program the A button to display temperature. Be ready to guide students in lighting up the LED lights according to temperature readings and downloading their projects onto their Microbits. Ensure you know how to correctly connect the LED strip to the Microbit.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- LED Strip with crocodile clips

Learning Goals	Learning Outcomes
1. Develop skills to create and manage a new Microbit project. 2. Understand and apply the neopixel extension to program the LED strip. 3. Gain knowledge on setting up the LED strip and displaying temperature on the Microbit screen. 4. Learn to light up the LED lights on the strip according to the temperature readings. 5. Acquire practical skills in downloading the project, connecting the LED strip to the Microbit, and testing the functionality.	1. Create a new Microbit project using makecode.microbit.org. 2. Add the neopixel extension to the project for LED strip programming. 3. Set up the LED strip in the project with a value of 30, representing the 30 LEDs on the strip. 4. Program the A button to display the temperature on the Microbit screen. 5. Display the temperature by lighting up the LED lights on the strip, with the number of lights corresponding to the temperature reading. 6. Download the project and transfer it to the Microbit. 7. Connect the LED strip to the Microbit using the specified pin connections and power it using a USB cable.

Week 4

Lesson: Shooting Stars

□ Intermediate	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare to guide students in creating a Microbit project, adding the neopixel extension, and setting up the LED strip. Facilitate the creation of a 'star' that lights up with a loud sound, and ensure students can test this on their LED strip. Assist students in making the 'star' shoot along the strip and adding random colours. Finally, ensure students can download and test their code, encouraging them to create multiple shooting stars.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- LED Strip with crocodile clips

Learning Goals	Learning Outcomes
1. Develop skills in creating and managing a new Microbit project. 2. Understand and apply the neopixel extension to program the LED strip. 3. Gain proficiency in setting up and programming the LED strip using code blocks. 4. Learn to utilise the microphone in the microbit to detect sound and trigger LED actions. 5. Acquire knowledge on how to test and debug the project on the LED strip. 6. Master the concept of pixel shifting to create the illusion of moving light. 7. Experiment with random colour generation for the LED strip. 8. Learn to download and implement the code onto the microbit for real-world testing.	1. Create and manage a new Microbit project. 2. Integrate the neopixel extension into the project. 3. Set up and programme the LED strip using the provided code. 4. Develop a function to light up the first LED on the strip white when a loud sound is detected. 5. Test the function on the LED strip and ensure it works as expected. 6. Implement a function to make the 'star' shoot along the strip. 7. Enhance the function to display stars in random colours. 8. Download and test the final code on the microbit, ensuring different colour 'stars' shoot along the strip when a loud noise is made.

Week 5

Lesson: LED Flags

□ Intermediate	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare to guide students in creating LED flags using a Microbit project. They will need to understand how to add the neopixel extension and set up the LED strip. Facilitate as they create bicolor and tricolor flags, using the example of Malta and Ireland respectively. Encourage creativity and problem-solving skills for the challenge of representing the American flag.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- LED Strip with crocodile clips

Learning Goals	Learning Outcomes
1. Understand and apply the concept of bicolor and tricolor flags using LED strips. 2. Create and manage a new Microbit project effectively. 3. Utilise the neopixel extension to program the LED strip. 4. Develop skills to set up and interact with the LED strip using code. 5. Apply coding skills to create complex patterns, such as the American flag, on the LED strip.	1. Construct bicolor and tricolor flags using LED strips. 2. Utilise the neopixel extension to program the LED strip. 3. Set up and interact with the LED strip using a variable. 4. Apply the concept of ranges to light up specific sections of the LED strip. 5. Code the LED strip to represent complex flag designs, such as the American flag.

Week 6

Lesson: LED Stacking

□ Advanced	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare for this lesson by familiarising yourself with the Microbit project platform and the neopixel extension. Understand how to set up an LED strip and create variables to store the strip and the amount of LEDs. Be ready to guide students in creating a function to show the LED stack, and programming buttons to increase and decrease the stack. Ensure students know how to download their code and connect their LED strip to their microbit.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- LED Strip with crocodile clips

Learning Goals	Learning Outcomes
1. Develop skills in creating and managing a new Microbit project. 2. Understand and apply the neopixel extension for LED programming. 3. Learn to set up and interact with the LED strip using variables. 4. Develop competency in creating and using functions to control LED display. 5. Gain experience in programming button controls to manipulate LED stack size.	1. Create a new Microbit project using makecode.microbit.org. 2. Add the neopixel extension to the project for LED strip programming. 3. Set up the LED strip with a variable storing the strip, set to a value of 30. 4. Create an 'amount' variable to store the number of LEDs in the stack. 5. Develop a 'showStack' function to display the stack of lit LEDs. 6. Create a range of LEDs on the strip to light up, using the 'amount' variable, and call the 'showStack' function from the 'on start' block. 7. Program button A to increase the LED stack by adding 1 to the 'amount' variable and calling the 'showStack' function. 8. Program button B to decrease the LED stack by subtracting 1 from the 'amount' variable and calling the 'showStack' function. 9. Download the code onto a microbit, connect the LED strip using crocodile clips, and test the LED stack's increase and decrease functions with buttons A and B.

Week 7

Lesson: LED Strip Precision Game

□ Advanced	□ 60 mins	□ Teacher/Student led	□ Student Quiz	□ Student Challenge

Prepare to guide students through creating an interactive LED strip game using a Microbit project. Familiarise yourself with the neopixel extension and the process of setting up the LED strip. Understand the purpose of the four variables: 'target', 'position', 'delay', and 'increment'. Be ready to explain how to set up the level, create a refresh function, and make the blue light move. Prepare to guide students through the steps of going back to the start, hitting the target, and handling a missed target. Finally, ensure you can assist students in downloading their code and playing the game.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- LED Strip with crocodile clips

Learning Goals	Learning Outcomes
1. Understand and apply the concept of LED strip programming using Microbit. 2. Develop skills in creating and manipulating variables in a coding project. 3. Learn to create and use functions for specific tasks within a coding project. 4. Gain proficiency in using conditional statements to control game outcomes. 5. Develop the ability to download and test code on a physical device.	1. Program an LED strip to light up specific LEDs in response to user input. 2. Create and manipulate variables to control game mechanics in a Microbit project. 3. Implement the neopixel extension to interact with an LED strip. 4. Design a function to refresh LED lights based on variable values. 5. Download and test the code on a physical Microbit device.

Week 8

Lesson: Microbit Lab

□ Advanced	□ 60 mins	□ Teacher led

Prepare to introduce the concept of Microbit projects, demonstrating a simple LED pattern to inspire creativity. Organise students into small groups for brainstorming, emphasising teamwork and achievable project ideas. Facilitate a feedback session after idea presentations, guiding project simplification if necessary. Assist during project creation, encouraging peer support and discovery sharing. Finally, conduct a 'Show and Tell' session, celebrating student effort and creativity, reinforcing learning objectives and the importance of teamwork.

Students can use any of these devices (and can share if necessary):

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit

Learning Goals	Learning Outcomes
1. Develop creative and achievable project ideas using basic Microbit blocks. 2. Collaborate effectively in small groups to brainstorm, plan and execute a Microbit project. 3. Present project ideas clearly and receive feedback constructively. 4. Apply problem-solving skills to create a Microbit project based on the brainstormed idea. 5. Reflect on the project creation process, discussing changes made, challenges faced, and skills learned.	1. Brainstorm and develop a simple Microbit project idea in a group setting. 2. Present the project idea to the class, explaining the planned LED patterns and inputs. 3. Receive, incorporate, and respond to feedback on the project idea. 4. Create a Microbit project based on the brainstormed idea, using basic Microbit blocks. 5. Present the final Microbit project to the class, explaining the coding process and any changes made during the project creation.