CODING IRELAND

Teacher Learning Plan

Digital Skills Curriculum 2024/25

3rd Year

Table of Contents

How to Use This Learning Plan

This learning plan provides an overview of all the modules available for 3rd Year, 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.

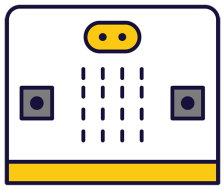
Student Access

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@codingireland.ie or 01 584 9955.

Module: Introduction to Microbit Programming



This module introduces students to the fascinating world of microbit programming. Teachers should guide students through creating new projects, exploring the project editor, and writing and deleting code. The first lesson focuses on programming microbits to display messages, react to button presses, show icons, and play melodies. The second lesson involves creating a reaction timer game, teaching students how to create variables, add random delays, and record reaction times. Teachers should encourage experimentation and exploration throughout.

Duration	**Equipment**
2 weeks	Students can use any of these devices: • Chromebook/Laptop/PC Required Equipment: • Microbit
Module Goals	**Module Outcomes**
1. Understand the fundamentals of microbits and their programming. 2. Develop proficiency in creating, adding, and deleting code in the project editor. 3. Gain skills in programming microbits to display messages, react to button presses, show icons, play melodies, and respond to movement. 4. Design and create a reaction timer game using a Microbit. 5. Learn to create variables, store time stamps, and record a player's reaction time in a game.	1. Understand and utilise the project editor on the MakeCode for Microbit website. 2. Create, add, and delete code to program a Microbit to display messages, react to button presses, and play melodies. 3. Connect a Microbit to a computer and upload the programmed code. 4. Design and create a reaction timer game using a Microbit, incorporating random visual prompts. 5. Use variables to store time stamps and record a player's reaction time in the game.

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):

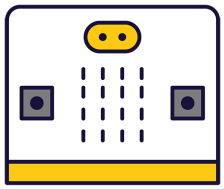
- Chromebook/Laptop/PC

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.

Module: Advanced Microbit Applications



This module delves into advanced applications of Microbits, starting with sensor graphs and culminating in a creative lab project. Encourage students to understand the utility of variables, functions, and if-else conditions. Facilitate critical and creative thinking, especially in the Seismic and Meteorological Station project. The 'Microbit Finder' lesson will require students to grasp radio group communication. The 'Chase the Dot' game development lesson is a fun way to reinforce coding concepts. Finally, the 'Microbit Lab' encourages teamwork and creativity.

Duration	**Equipment**
9 weeks	Students can use any of these devices: • Chromebook/Laptop/PC Required Equipment: • Microbit
Module Goals	**Module Outcomes**
1. Master the use and application of Microbit sensors to interpret and respond to different inputs. 2. Develop proficiency in creating functional Microbit applications such as an alarm system and a seismic and meteorological station. 3. Gain skills in programming Microbit games that incorporate variables, functions, and gesture controls. 4. Understand the principles of radio communication between Microbits and apply this knowledge to create a proximity detector. 5. Enhance creativity, critical thinking, and teamwork skills through the design and implementation of a unique Microbit project.	1. Interpret and utilise Microbit sensors to create interactive graphs. 2. Design and implement a Microbit alarm system using sensor data and threshold values. 3. Develop a multi-device IoT network using Microbits to monitor and display environmental data. 4. Create a time-based game on Microbit using variables and functions. 5. Use radio signals between two Microbits to detect proximity. 6. Design and code an interactive game on Microbit using gestures for control. 7. Brainstorm, design, and implement a simple Microbit project in a team.

Week 1

Lesson: Microbit Sensor Graphs

Beginner	60 mins	Teacher/Student led	Student Quiz	Student Challenge

For the 'Microbit Sensor Graphs' lesson, teachers should familiarise themselves with the makecode.microbit.org website and how to create a new project. They should understand how the light level sensor works and how to display and graph the light level. Teachers should also explore how to graph other sensors such as temperature, compass heading, acceleration, magnetic forces, and sound level. Lastly, they should encourage students to get creative with their graphs and consider how these sensors could be used in other projects.

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.microbit.org. 2. Learn to display the light level on the Microbit. 3. Develop skills to graph the light level and observe changes. 4. Gain knowledge on graphing other sensors such as temperature, compass heading, acceleration, magnetic forces and sound level. 5. Apply creativity to graph other possible values and incorporate sensor usage in different projects.	1. Create a new project on makecode.microbit.org. 2. Display the light level on the Microbit. 3. Graph the light level changes on the Microbit. 4. Graph the readings from other sensors on the Microbit, including temperature, compass heading, acceleration, magnetic forces, and sound level. 5. Design and implement a creative application using the Microbit's sensors.

Week 2

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 3

Lesson: Microbit Seismic and Meteorological Station

Intermediate	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to guide students through programming four Microbits to create a Seismic and Meteorological Station. Each Microbit will have a unique role: monitoring temperature, light levels, and seismic activity, displaying temperature data, indicating day or night based on light levels, and alerting seismic activity. Students will use MakeCode for Microbit and learn to set up radio communication, event handlers, and display functions. Encourage students to test their projects and consider improvements such as displaying movement intensity, measuring sound levels, recording historical data, or sending alerts based on specific conditions.

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 role of each Microbit in the Seismic and Meteorological Station and how they communicate with each other. 2. Develop skills in coding and programming Microbits to monitor and display temperature, light levels, and seismic activity. 3. Test and troubleshoot the coded Microbits to ensure they function as intended in the Seismic and Meteorological Station. 4. Apply critical thinking to improve the functionality of the Seismic and Meteorological Station. 5. Develop an understanding of how meteorological and seismic data can be collected, displayed, and used in real-world applications.	1. Program four separate Microbits to perform unique roles in a Seismic and Meteorological Station. 2. Code the 'Seismic and Meteorological Station' Microbit to monitor and wirelessly broadcast temperature, light levels, and 'seismic' activity data. 3. Code the 'Temperature Display' Microbit to receive and display the temperature data from the 'Seismic and Meteorological Station' Microbit. 4. Code the 'Day/Night Indicator' Microbit to receive light level data and display an indication of whether it's day or night. 5. Code the 'Seismic Alert' Microbit to receive 'seismic' activity data and display an alert if seismic activity is detected.

Week 4

Lesson: Exactly 11

Intermediate	60 mins	Teacher/Student led	Student Quiz	Student Challenge

This lesson involves creating a Microbit project where students guess when 11 seconds have passed. They will learn to create and use variables 'starttime', 'taken', and 'difference'. They will also learn to use the 'running time (ms)' block, calculate absolute values, and use conditional statements to display results. The lesson involves coding and understanding the concept of milliseconds.

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 utilising variables in Microbit projects. 2. Understand and apply the concept of running time in programming. 3. Gain proficiency in using mathematical operations to calculate time differences. 4. Learn to use conditional statements to display different outcomes based on user input. 5. Enhance problem-solving skills through the creation of a time-guessing game.	1. Develop a new Microbit project. 2. Create and utilise 'starttime' variable to record the start time. 3. Establish a 'taken' variable to store the time elapsed between two actions. 4. Formulate a 'difference' variable to calculate the difference between 11 seconds and the 'taken' time. 5. Implement code to display the result and provide feedback to the user.

Week 5

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 6

Lesson: Chase the Dot

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

In this lesson, students will create a game called 'Chase the Dot' using Microbit. They will learn to create a new project, define variables, create a function, and use gestures to control movements. The game involves two dots, a target and a chaser. The aim is for the chaser dot to catch the target dot, which moves to a random position each time it's caught. The lesson involves coding for dot creation, movement, scoring, and game initiation.

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 Microbit projects on makecode.com. 2. Understand and apply the concept of variables in coding to create game sprites. 3. Learn to create and use functions for repetitive tasks in a game scenario. 4. Develop skills in using gestures to control game elements in a Microbit project. 5. Understand and implement the concept of scoring and round systems in game development.	1. Create and manipulate variables to store and control game sprites in a Microbit project. 2. Develop a function to position a sprite at a random location on the edge of the screen. 3. Implement a countdown timer and sound effects to enhance game play. 4. Programme the Microbit to respond to tilt gestures to control sprite movement. 5. Design a scoring system that responds to sprite interactions and triggers new rounds of play.

Week 7

Lesson: Microbit - Invaders

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to guide students through creating a Microbit project on the Microbit website. They'll learn to create a countdown timer, and establish variables for Bullets, Players, and Enemy. Students will then create and move player and enemy sprites, incorporating logic for movement and game-ending conditions. They'll also create a bullet sprite, with movement and enemy-hit detection. Lastly, they'll add conditions for enemy sprites hitting the player, ending the game.

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 a Microbit project using the online platform. 2. Create a countdown timer using basic show number and pause blocks. 3. Define and utilise variables for game elements such as Bullets, Players and Enemy. 4. Design and implement player and enemy sprites in the game. 5. Enable player sprite movement using A and B button inputs.	1. Create a Microbit project using the Microbit website. 2. Develop a countdown timer using the show number and pause blocks in the Basic category. 3. Create three variables: Bullets, Players, and Enemy. 4. Design a player sprite and position it on the left side of the screen. 5. Program the player sprite to move left or right in response to the A and B buttons. 6. Create an enemy sprite, position it at the top of the screen, and program it to move downwards and end the game if it touches the player sprite. 7. Create a bullet sprite that moves upwards from the player's position when button A+B is pressed. 8. Implement a condition to check if the bullet sprite is touching the enemy sprite, and if so, make the enemy sprite disappear and add a point to the player's score. 9. Add a condition to check if the enemy sprite is touching the player sprite, and if so, end the game.

Lesson: Microbit: Invaders

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Embark on an exciting journey to create your own 'Invaders' game using a Microbit. Learn to create and control sprites, detect collisions, and implement game logic. By the end, you'll have a fully functional game to play and share with your friends. Let's get started!

Learning Goals	Learning Outcomes
1. Understand how to create a project using Microbit. 2. Learn to create and utilise variables in a Microbit project. 3. Develop skills in positioning and moving sprites. 4. Gain knowledge on creating and controlling bullet sprites. 5. Learn to implement collision detection between sprites.	1. Create a Microbit project using the Microbit website. 2. Define and utilise three variables: Bullets, Players, and Enemy. 3. Position the player sprite in the middle of the screen. 4. Implement logic for moving the player sprite using the A and B buttons. 5. Create an enemy sprite that starts at a random position at the top of the screen and moves downwards. 6. Create a bullet sprite that moves upwards from the player's position when button A+B is pressed. 7. Implement a condition to detect if the bullet sprite is touching the enemy sprite, causing the enemy sprite to disappear and the player to score a point. 8. Implement a condition to detect if the enemy sprite is touching the player sprite, causing the game to end.

Week 8

Lesson: Microbit Lab

Advanced	60 mins	Teacher led

Prepare to introduce the Microbit Lab lesson, demonstrating a simple Microbit project to inspire students. Divide students into groups for brainstorming and project creation. Facilitate brainstorming, feedback, and project creation sessions, ensuring students keep their ideas simple and achievable. Encourage constructive feedback and teamwork. Finally, organise a 'Show and Tell' session for groups to present their projects, fostering a supportive learning environment and reinforcing 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 a simple Microbit project using basic blocks. 2. Work effectively in groups, contributing ideas and making collective decisions. 3. Present and explain a project idea, including its components and envisioned outcome. 4. Give and receive constructive feedback, and incorporate it into project plans. 5. Code a Microbit project, demonstrating problem-solving and creativity.	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. Incorporate feedback from peers and teacher into the project plan. 4. Create a Microbit project based on the brainstormed idea and feedback received. 5. Present the final Microbit project to the class, explaining the coding process, changes made, and learnings from the process.

Module: Coding with JavaScript

This module introduces students to JavaScript, a popular textual programming language. Teachers should explain the differences between JavaScript and block-based languages, guide students through creating and modifying code, and cover common syntax errors. The module progresses from basic concepts to more complex ones, such as variables, data types, operators, and conditional and switch statements. The final lesson allows students to apply their skills in a self-guided project. Encourage creativity, problem-solving, and thorough testing and debugging throughout The module.

Duration	**Equipment**
8 weeks	Students can use any of these devices: • Chromebook/Laptop/PC Required Equipment: • Microbit
Module Goals	**Module Outcomes**
1. Understand and apply the fundamentals of JavaScript, including syntax, code execution, and debugging. 2. Develop proficiency in creating and manipulating JavaScript variables and data types. 3. Master the use of JavaScript operators for arithmetic, string, assignment, comparison, and logical operations. 4. Gain competency in controlling code flow using JavaScript conditional and switch statements. 5. Apply learned JavaScript skills to create an innovative project using the MakeCode Microbit editor.	1. Understand and apply the basic concepts of JavaScript, including code execution sequence and debugging syntax errors. 2. Create and manipulate variables in JavaScript, understanding their role in storing and calculating data. 3. Identify and utilise different JavaScript data types, including strings, numbers, booleans, arrays, and objects. 4. Apply JavaScript operators and conditional statements to control the flow of code and perform operations between operands. 5. Design and implement a unique project using JavaScript and the MakeCode Microbit editor, demonstrating a comprehensive understanding of the language's fundamentals.

Week 1

Lesson: Introduction to JavaScript

Intermediate	60 mins	Teacher/Student led	Student Quiz

This lesson introduces students to JavaScript, a popular programming language. Teachers should explain the concept of programming languages and how JavaScript is used to make websites interactive. The lesson progresses through a sequence of steps, starting with basic JavaScript code and gradually introducing more complex elements. Students will have the opportunity to write their own JavaScript code, switch between block and JavaScript views, and correct errors in their code. The lesson concludes with a challenge to program a button to play music. Teachers should be prepared to assist students and answer questions throughout the lesson.

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

- Chromebook/Laptop/PC

Learning Goals	Learning Outcomes
1. Understand the basic concept and importance of JavaScript in web development. 2. Learn about the sequence of execution in JavaScript programming. 3. Gain hands-on experience in writing and executing JavaScript code. 4. Develop skills to troubleshoot and correct syntax errors in JavaScript. 5. Apply JavaScript knowledge to create interactive functions, such as programming buttons and displaying messages.	1. Understand the basic concept and purpose of JavaScript as a textual programming language. 2. Recognise the sequence in which JavaScript code is executed. 3. Write and modify simple JavaScript code using the Makecode Microbit project editor. 4. Identify and correct common syntax errors in JavaScript code. 5. Program interactive functions in JavaScript, such as button presses and music playback.

Week 2

Lesson: JavaScript - Exactly 11

Intermediate	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to guide students through creating a game using JavaScript, focusing on variables, functions, and mathematical operations. The game, 'Exactly 11', challenges players to guess when exactly 11 seconds have passed. Students will learn to create and use variables such as 'starttime', 'taken', and 'difference', and to use functions like 'input.onButtonPressed' and 'Math.abs'. They will also learn to add comments to their code for clarity. The lesson concludes with testing the game in the Microbit simulator.

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

- Chromebook/Laptop/PC

Learning Goals	Learning Outcomes
1. Develop understanding of variables, functions, and mathematical operations in JavaScript. 2. Gain proficiency in creating and manipulating variables such as 'starttime', 'taken', and 'difference'. 3. Learn to use 'input.runningTime()' to measure time in JavaScript. 4. Understand the use of 'Math.abs()' for calculating absolute values. 5. Apply conditional statements to display different outcomes based on user input.	1. Create a new Microbit project using JavaScript. 2. Define and initialise 'starttime' variable in JavaScript. 3. Implement a function to set the start time using 'input.onButtonPressed' method. 4. Create and initialise 'taken' and 'difference' variables in JavaScript. 5. Program a function to calculate the time taken and the difference from 11 seconds. 6. Display the result of the game using conditional statements and 'basic.showIcon' method.

Week 3

Lesson: JavaScript Variables

Intermediate	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to introduce students to JavaScript variables, explaining their role as containers for data values. Use practical examples such as a game score to illustrate this concept. Guide students through the process of creating, adding to, and displaying variables. Progress to discussing number and string variables, demonstrating how they can be added or concatenated. Explain the importance of unique identifiers for variables and the rules for constructing them. Facilitate an exercise where students create their own variables, and guide them through programming buttons to display these variables. Finally, simulate a birthday scenario to demonstrate how variable values can be updated.

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

- Chromebook/Laptop/PC

Learning Goals	Learning Outcomes
1. Understand and apply the concept of variables in JavaScript. 2. Perform arithmetic operations with number variables. 3. Concatenate string variables to form new strings. 4. Identify and follow the rules for naming JavaScript variables. 5. Write JavaScript code to create and manipulate variables.	1. Define and initialise JavaScript variables. 2. Perform arithmetic operations with number variables. 3. Concatenate string variables. 4. Understand and apply rules for JavaScript identifiers. 5. Manipulate variables through button-press and gesture-based events in JavaScript code.

Week 4

Lesson: JavaScript Data Types

Intermediate	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to guide students through understanding JavaScript data types, including strings, numbers, booleans, arrays, and objects. Emphasise the importance of correct syntax and the role of quotes in defining strings. Highlight the use of booleans in conditional statements. Explain how arrays store multiple values and how objects are collections of properties. Discuss the concept of 'undefined' and its implications. Finally, facilitate hands-on exercises to reinforce learning, using the makecode.microbit.org platform for practical application.

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

- Chromebook/Laptop/PC

Learning Goals	Learning Outcomes
1. Understand and apply JavaScript data types including strings, numbers, booleans, arrays, and objects. 2. Recognise and handle undefined variables in JavaScript. 3. Manipulate strings using single and double quotes. 4. Use booleans in conditional statements. 5. Create and manipulate arrays and objects, including accessing specific elements and properties.	1. Identify and differentiate between various JavaScript data types including strings, numbers, booleans, arrays, objects, and undefined. 2. Write and manipulate JavaScript strings using single and double quotes. 3. Understand and utilise JavaScript numbers with or without decimal points. 4. Implement JavaScript booleans in conditional statements. 5. Create and manipulate JavaScript arrays using square brackets and commas. 6. Construct JavaScript objects using curly braces, and understand the concept of name:value pairs. 7. Recognise and handle undefined variables in JavaScript. 8. Apply learned JavaScript data types in practical coding exercises. 9. Understand and use array indexing to access specific elements in a JavaScript array. 10. Manipulate array indexes using button-pressed functions in JavaScript.

Week 5

Lesson: JavaScript Operators

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to introduce students to JavaScript operators, starting with the basics of what an operator is. Use practical examples to explain different types of operators including arithmetic, string, assignment, comparison, and logical operators. Encourage students to try out each operator type with hands-on coding exercises on the MakeCode website. Ensure students understand the concept of operands and how operators are used to perform operations on these operands.

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

- Chromebook/Laptop/PC

Learning Goals	Learning Outcomes
1. Understand and apply JavaScript operators including arithmetic, string, assignment, comparison, and logical operators. 2. Perform arithmetic operations using JavaScript arithmetic operators. 3. Manipulate strings using JavaScript string operators. 4. Assign and modify variable values using JavaScript assignment operators. 5. Compare values and make decisions using JavaScript comparison and logical operators.	1. Identify and explain the different types of JavaScript operators. 2. Perform arithmetic operations using JavaScript arithmetic operators. 3. Concatenate strings and perform operations on variables using JavaScript string and assignment operators. 4. Compare values using JavaScript comparison operators. 5. Apply logical operators to determine the logic between variables or values in JavaScript.

Week 6

Lesson: JavaScript Conditional Statements

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

This lesson delves into JavaScript Conditional Statements, starting with an introduction to 'if', 'else if', and 'else' statements. Teachers should guide students through the process of writing and understanding these statements, using practical examples such as setting movie ticket prices based on age. The lesson then progresses to more complex scenarios involving multiple conditions. Students will get hands-on experience by writing their own conditional statements to display different greeting messages based on the time of day. Teachers should ensure students understand the importance of correct syntax and the use of operators in these statements.

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

- Chromebook/Laptop/PC

Learning Goals	Learning Outcomes
1. Understand and apply JavaScript conditional statements including 'if', 'else if', and 'else'. 2. Use conditional statements to test conditions and execute different code blocks based on the results. 3. Combine multiple conditions in a single 'if' statement using logical operators. 4. Write and implement custom 'if', 'else if', and 'else' statements in practical scenarios. 5. Debug and improve code using conditional statements to enhance program functionality.	1. Understand the concept and purpose of JavaScript conditional statements including 'if', 'else if', and 'else'. 2. Apply 'if' statement in JavaScript to test a condition and execute a piece of code if the condition is true. 3. Utilise 'else if' and 'else' statements in JavaScript to test multiple conditions and execute different pieces of code based on the results. 4. Write a JavaScript 'if' statement to set a variable based on a condition. 5. Develop a JavaScript program using 'if', 'else if', and 'else' statements to set a variable based on multiple conditions.

Week 7

Lesson: JavaScript Switch Statements

Advanced	25 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to guide students through the concept of JavaScript Switch Statements. Begin with explaining what a switch statement is, using the provided examples. Highlight the importance of the 'break' keyword and its role in the flow of code execution. Introduce the 'default' keyword and its use when none of the case conditions are true. Discuss how the same code can be run for different case conditions. Finally, encourage students to write their own switch statement and include a default case.

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

- Chromebook/Laptop/PC

Learning Goals	Learning Outcomes
1. Understand and apply JavaScript switch statements. 2. Use the 'break' keyword effectively within switch statements. 3. Implement the 'default' keyword in switch statements for unmatched conditions. 4. Apply multiple cases to the same code within switch statements. 5. Write and modify switch statements to manipulate variable values.	1. Understand and apply JavaScript switch statements to select code to run based on specific conditions. 2. Utilise the 'break' keyword to exit a switch statement after a condition has been met. 3. Implement the 'default' keyword to specify code to run when no case conditions are met. 4. Apply multiple case conditions to the same piece of code within a switch statement. 5. Write and modify a switch statement, including a default case, to manipulate variable values based on conditions.

Week 8

Lesson: Microbit Innovation Project

Advanced	60 mins	Teacher/Student led

This lesson encourages students to innovate using Microbit. They'll brainstorm a project, plan their code, and translate it into JavaScript using the makecode microbit editor. Testing and debugging are integral parts of the process. Finally, students will reflect on their work, identifying successes, challenges, and areas for future improvement.

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 thinking and problem-solving skills through brainstorming and planning a unique Microbit project. 2. Understand and apply coding concepts such as variables, data types, operators, conditional statements, and switch statements in JavaScript. 3. Gain practical experience in using the makecode microbit editor for coding and debugging. 4. Test and refine the functionality of the created project, demonstrating perseverance and resilience. 5. Reflect on the process and outcomes of the project, identifying successes, challenges, and areas for future improvement.	1. Generate and sketch innovative project ideas using the Microbit platform. 2. Develop a comprehensive plan for coding, including pseudocode or flowcharts, with appropriate use of variables, data types, operators, conditional and switch statements. 3. Translate planned code into JavaScript using the makecode microbit editor, demonstrating resilience through trial and error. 4. Test and debug the coded project, ensuring it functions as intended. 5. Reflect on the project process, identifying successes, challenges, and areas for future improvement.

Module: Dynamic Web Design with HTML, CSS & JS

This module provides a comprehensive introduction to dynamic web design using HTML, CSS, and JavaScript. Teachers should utilise a hands-on approach, guiding students through setting up their development environment, understanding how these languages interact, manipulating the DOM, and integrating external libraries and APIs. The module culminates in the creation of an interactive quiz game and a weather web app. The final week allows students to showcase their work, providing an opportunity for peer review and constructive feedback.

Duration	**Equipment**
8 weeks	Students can use any of these devices: • Chromebook/Laptop/PC • iPad/Tablet
Module Goals	**Module Outcomes**
1. Master the interaction of HTML, CSS, and JavaScript to create dynamic web pages. 2. Set up and effectively utilise web development tools including code editors, browser developer tools, and debugging consoles. 3. Develop advanced scripting skills for DOM manipulation, including creating, deleting, and modifying HTML elements. 4. Implement dynamic form validation with JavaScript, providing real-time feedback and validating various input types. 5. Integrate external libraries and APIs into web development projects to enhance functionality and data dynamism.	1. Understand and apply the interaction between HTML, CSS, and JavaScript to create dynamic web pages. 2. Set up and effectively use a web development environment, including code editors, browser developer tools, and debugging consoles. 3. Manipulate the Document Object Model (DOM) using JavaScript, including creating, deleting, and modifying HTML elements based on certain conditions or inputs. 4. Implement dynamic form validation using JavaScript, providing real-time feedback and validating different input types. 5. Integrate external libraries such as jQuery and APIs to pull dynamic data into web pages. 6. Create an interactive quiz game using HTML, CSS, and JavaScript, incorporating features such as timers, score trackers, and dynamic question loading. 7. Develop a Weather Web App that fetches and displays real-time weather data based on a location, making it interactive and visually appealing. 8. Present a web development project in a showcase, demonstrating mastery of HTML, CSS, and JavaScript.

Week 1

Lesson: Overview of how HTML, CSS, and JavaScript Interact

Beginner	20 mins	Teacher/Student led	Student Quiz

Prepare to explain the analogy of web development to building a house, with HTML, CSS, and JavaScript as the structure, design, and functionality respectively. Ensure understanding of the basic structures of HTML, CSS, and JavaScript, including their syntax and usage. Highlight the synergy of these three languages in creating dynamic web pages. Be ready to discuss real-life applications, such as creating a web-based quiz, to illustrate their interactivity. Conclude by emphasising the importance of proficiency in all three components for effective web development.

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

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Understand the roles of HTML, CSS, and JavaScript in web development. 2. Comprehend the basic structure and syntax of HTML, CSS, and JavaScript. 3. Apply CSS styles in different ways: inline, internal, and external. 4. Recognise common uses of JavaScript in enhancing web interactivity. 5. Appreciate the synergy of HTML, CSS, and JavaScript in creating dynamic web pages.	1. Understand the roles of HTML, CSS, and JavaScript in web development. 2. Identify the basic structure and elements of an HTML document. 3. Apply CSS to control the appearance of a webpage. 4. Use JavaScript to add interactivity to web pages. 5. Integrate HTML, CSS, and JavaScript to create dynamic web pages.

Lesson: Setting up Essential Tools

Beginner	60 mins	Teacher/Student led	Student Quiz	Student Challenge

This lesson guides students through setting up essential web development tools. They'll learn about code editors, browser developer tools, and the console for debugging. Students will explore CodePen, an online code editor, and create a basic webpage using HTML, CSS, and JavaScript. They'll also add a button with an onclick event. The lesson concludes with a wrap-up and encouragement for further practice.

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

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Gain proficiency in using web development environments and essential tools such as code editors, browser developer tools, and the console for debugging. 2. Understand the features and benefits of using CodePen as an online code editor. 3. Develop skills to inspect, debug, and optimise code using browser developer tools. 4. Learn to use the console for identifying and resolving issues in JavaScript code. 5. Apply knowledge to create a basic webpage on CodePen, incorporating HTML, CSS, and JavaScript, and adding interactive elements like buttons with onclick events.	1. Identify and utilise essential web development tools including code editors, browser developer tools, and the console for debugging. 2. Select and use CodePen as an online code editor for writing and previewing HTML, CSS, and JavaScript in real-time. 3. Inspect and debug code using browser developer tools and the console. 4. Create and edit a basic webpage on CodePen using HTML, CSS, and JavaScript. 5. Add interactive elements to a webpage, such as a button with an onclick event, using JavaScript.

Week 2

Lesson: Scripting and DOM Manipulation

Intermediate	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to guide students through a hands-on exploration of scripting and DOM manipulation. They'll set up a CodePen project, create HTML structures, and add JavaScript functions. They'll learn to use 'onclick' attributes, event listeners, and manipulate text colour and size. The lesson concludes with challenges to create small text and remove elements, reinforcing their understanding of DOM manipulation.

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 concept of DOM manipulation using JavaScript. 2. Develop skills in creating and modifying HTML elements dynamically. 3. Gain proficiency in handling events using JavaScript, including click and mouseover events. 4. Learn to use JavaScript to alter CSS properties of HTML elements. 5. Apply problem-solving skills to complete coding challenges related to DOM manipulation.	1. Set up and utilise CodePen for HTML and JavaScript scripting. 2. Create and manipulate HTML structure using JavaScript. 3. Implement JavaScript functions to dynamically add elements to a webpage. 4. Utilise event listeners to trigger JavaScript functions. 5. Manipulate CSS properties of HTML elements through JavaScript.

Week 3

Lesson: Dynamic Form Validation with JavaScript

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare for a hands-on lesson on dynamic form validation using JavaScript. Familiarise yourself with the CodePen environment, as students will be setting up their projects there. The lesson will guide students through creating a form, styling it with CSS, and adding JavaScript for validation. They will learn to validate fields for name, email, and password, ensuring the correct length and format. The lesson concludes with a challenge to add an age field and validate it.

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

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Understand how to set up a project on CodePen for HTML, CSS, and JavaScript development. 2. Create and style a form using HTML and CSS. 3. Implement JavaScript code to validate form fields for specific requirements. 4. Test form validation and handle form submission using JavaScript. 5. Extend JavaScript validation to new form fields, demonstrating adaptability of skills.	1. Set up a project environment in CodePen. 2. Create a form with name, email, and password fields using HTML. 3. Style the form using CSS for better visual appeal. 4. Implement JavaScript code to prevent form submission and enable validation. 5. Validate the name field to ensure it is at least 3 characters long. 6. Validate the email field to ensure it contains '@' and '.' characters. 7. Validate the password field to ensure it is at least 8 characters long and contains at least one number and one letter. 8. Test the form by entering different values and checking if the validation works as expected. 9. Add an age field and validate it to ensure the person is over 13.

Week 4

Lesson: Integrating External Libraries and APIs

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to guide students through the process of integrating external libraries and APIs into a project. The lesson involves setting up a project on CodePen, adding jQuery, creating an HTML structure, understanding jQuery syntax, selectors, and events, adding a click event listener, fetching weather data with an API, displaying the weather data, and extending the functionality of the weather app. The lesson concludes with wrapping up the weather app and encouraging students to explore more advanced features and APIs.

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 setting up a new project on CodePen. 2. Gain understanding and practical skills in integrating jQuery into a project. 3. Master the creation of HTML structures and the application of jQuery syntax and selectors. 4. Learn to handle jQuery events and implement event listeners. 5. Acquire skills in fetching data from external APIs and integrating it into a web application.	1. Set up a new project using CodePen and integrate jQuery library. 2. Create a basic HTML structure for a web application. 3. Understand and apply jQuery syntax to select and manipulate HTML elements. 4. Handle user interactions using jQuery event listeners. 5. Fetch and display real-time weather data from an external API.

Week 5

Lesson: Interactive Quiz Game

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

This lesson guides students through creating an interactive quiz game using HTML, CSS, and JavaScript. They'll learn how to set up a project on CodePen, structure HTML for the game, style it with CSS, and add functionality with JavaScript. The lesson includes adding a jQuery library, setting up questions, variables, and functions to display questions, check answers, and display scores. It concludes with challenges to add a timer and different difficulty levels 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 setting up a coding project using CodePen. 2. Understand and apply HTML structure to create an interactive quiz game. 3. Apply CSS styling to enhance the visual presentation of the quiz game. 4. Utilise jQuery library to manipulate HTML elements and handle user interactions. 5. Create and manipulate JavaScript arrays and objects to store quiz questions and answers.	1. Create a new project on CodePen and add HTML structure for an interactive quiz game. 2. Apply CSS styling to HTML elements for visual enhancement. 3. Integrate the jQuery library into the project for dynamic features. 4. Set up an array of question objects and variables for tracking quiz progress. 5. Display questions and answer options dynamically, and check user's answers for correctness. 6. Implement functionality to move to the next question after an answer is selected. 7. Display the user's score after all questions have been answered. 8. Enhance the quiz game with a timer for each question. 9. Add different difficulty levels to the quiz game for varied user experience.

Week 6

Lesson: Weather Web App

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

In this lesson, students will create a Weather Web App using HTML, CSS, and JavaScript. They will learn how to fetch and display real-time weather data using APIs and jQuery. The lesson will guide them through setting up the project, adding jQuery and Fontawesome, creating the HTML structure, styling the structure and headings, initializing JavaScript, fetching and displaying weather data, adding a unit toggle, and testing the app. They will also be encouraged to enhance their app by adding additional features such as a search bar and displaying more weather information.

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

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Develop a functional weather web application using HTML, CSS, and JavaScript. 2. Utilise jQuery for efficient manipulation of HTML documents. 3. Integrate and use external libraries such as Fontawesome for enhanced visual appeal. 4. Fetch and display real-time weather data using APIs. 5. Implement a feature to toggle between Celsius and Fahrenheit temperature units.	1. Develop a Weather Web App using HTML, CSS, and JavaScript. 2. Integrate jQuery and Fontawesome libraries into a web project. 3. Construct HTML structure to display weather data. 4. Style the web app using CSS for an engaging user interface. 5. Fetch and display real-time weather data from an API using JavaScript.

Week 7

Lesson: Web Showcase

Advanced	60 mins	Teacher/Student led

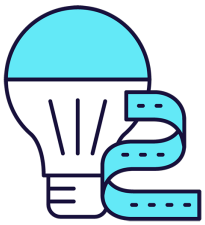
Prepare for a hands-on session where students will brainstorm, design, and code their own web page. Encourage creativity in content and design, emphasising the importance of HTML structure, CSS styling, and JavaScript interactivity. Guide them through project setup, content addition, styling, and refining their work. Ensure they test their work across different screen sizes and browsers, and foster a collaborative environment for feedback sharing.

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

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Generate creative and unique ideas for a web page design and content. 2. Establish a basic HTML structure for a web project using Codepen. 3. Implement diverse HTML elements to structure and add content to a web page. 4. Apply CSS for styling, enhancing visual appeal and readability of the web page. 5. Integrate JavaScript to add interactivity to the web page and ensure its functionality. 6. Review, refine, and optimise the web page for different screen sizes and browsers, while seeking and incorporating feedback.	1. Generate and articulate creative ideas for a web page design. 2. Establish a web project using Codepen, incorporating the basic structure of an HTML document. 3. Implement HTML to structure and add diverse content to a web page. 4. Apply CSS to enhance the visual appeal of the web page, experimenting with colours, fonts, and layouts. 5. Integrate JavaScript to add interactivity to the web page, ensuring functionality through thorough testing. 6. Evaluate and refine the web page, ensuring code cleanliness, cross-browser compatibility, and responsiveness, and seek peer feedback.

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):

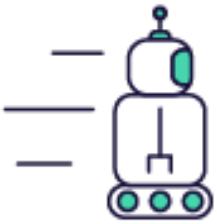
- Chromebook/Laptop/PC

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.

Module: Designing and Building for the Future



This module guides students through the process of designing and building future technologies, starting with assembling and programming traffic lights using a Microbit Traffic Lights Kit. Students will then create a traffic light reaction game, a pedestrian crossing simulation, and a Move Motor Sensor Car. They will learn to program the car to follow a line track, use ultrasonic sensors, and be controlled by a Microbit remote. The module concludes with a lesson on coding a set of traffic lights and a robot car to communicate. Teachers should ensure they are familiar with the MakeCode editor, Microbit, and the various extensions used throughout The module.

Duration	**Equipment**
8 weeks	Students can use any of these devices: • Chromebook/Laptop/PC • Microbit Required Equipment: • Microbit • Move Motor Car • Phillips Screwdriver • Traffic Lights Kit
Module Goals	**Module Outcomes**
1. Master the assembly and programming of Microbit Traffic Lights. 2. Develop skills in creating interactive games using Microbit and STOP:bit Traffic Lights. 3. Understand and apply the principles of pedestrian crossing simulations using MakeCode editor and micro:bit. 4. Gain proficiency in building and programming a Move Motor Sensor Car. 5. Learn to code a car to follow a line track and use ultrasonic sensors for object detection and avoidance.	1. Assemble and operate a Microbit Traffic Lights Kit. 2. Program a sequence of traffic lights using on/off and state methods. 3. Create a traffic light reaction game, incorporating variables and reaction time measurements. 4. Construct a pedestrian crossing simulation, incorporating button press detection and traffic light sequencing. 5. Assemble and code a Move Motor Sensor Car, exploring its various sensors and capabilities. 6. Program a Move Motor Car to follow a line track, adjusting code for optimal performance. 7. Utilise ultrasonic sensors to enable a Move Motor Car to follow an object and avoid obstacles. 8. Control a Move Motor Car using a Microbit as a remote controller, based on tilt detection. 9. Code a set of traffic lights and a robot car to communicate and respond to each other's states.

Week 1

Lesson: Build your Traffic Lights

Beginner	10 mins	Teacher/Student led	Student Quiz

Ensure students have all necessary materials, including the Microbit Traffic Lights Kit, a Microbit, and a Phillips head screwdriver. Guide them through opening the package and assembling the stand. Assist them in correctly positioning the Microbit on the traffic lights, ensuring they align the holes correctly. Supervise as they use the screwdriver to secure the Microbit. Celebrate their accomplishment once completed.

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

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- Traffic Lights Kit
- Phillips Screwdriver

Learning Goals	Learning Outcomes
1. Identify and gather necessary components for the Microbit Traffic Lights Kit. 2. Understand and execute the process of unpacking and preparing the kit. 3. Develop skills in assembling the stand for the traffic lights. 4. Apply knowledge of Microbit to correctly align and attach it to the traffic lights. 5. Demonstrate the ability to follow step-by-step instructions to complete a technical task.	1. Identify and gather necessary components for the Microbit Traffic Lights Kit. 2. Unpack and organise the Microbit Traffic Lights package contents. 3. Assemble the stand from the provided parts in the kit. 4. Align and attach the Microbit to the traffic lights using the correct hole configuration. 5. Successfully complete the assembly of the Microbit Traffic Lights.

Lesson: Microbit Traffic Lights

Beginner	40 mins	Teacher/Student led	Student Quiz	Student Challenge

In this lesson, students will create a new Microbit project on makecode.com, add the Stopbit extension, and test all the lights. They will learn about sequences in coding and apply this knowledge to program a traffic light sequence using on/off and state methods. Students will need to check the correct display of lights in each sequence.

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

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- Traffic Lights Kit

Learning Goals	Learning Outcomes
1. Understand and apply the process of creating a new Microbit project. 2. Learn to add and utilise the Stopbit extension for programming traffic lights kit. 3. Gain skills in testing and troubleshooting the functionality of the lights. 4. Comprehend the concept of 'sequence' in coding and apply it to program traffic lights. 5. Develop proficiency in programming the sequence of traffic lights using on/off and state methods.	1. Create and manage a new Microbit project on makecode.com. 2. Add and utilise the "stopbit" extension to the Microbit project. 3. Test and troubleshoot the functionality of each light on the Microbit. 4. Understand and apply the concept of 'sequence' in coding to program traffic lights. 5. Program the sequence of traffic lights using on/off and state methods.

Week 2

Lesson: Traffic Light Reaction Game

Beginner	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to guide students in creating a traffic light reaction game using a micro:bit and STOP:bit Traffic Lights. Ensure students understand how to attach the traffic lights to the micro:bit, create a new project on MakeCode, and add the "stopbit" extension. Explain the purpose of the 'startTime', 'endTime', and 'reactionTime' variables. Walk them through the process of setting up the code for button A and B presses, displaying the reaction time, and testing the game. Encourage students to challenge their peers and improve their reaction times.

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

- Microbit

Required equipment for this lesson:

- Microbit
- Traffic Lights Kit
- Phillips Screwdriver

Learning Goals	Learning Outcomes
1. Develop understanding of micro:bit and STOP:bit Traffic Lights for creating a reaction game. 2. Learn to add and utilise the "stopbit" extension in the MakeCode toolbox. 3. Understand and apply the concept of variables to measure reaction time. 4. Develop skills to program micro:bit buttons for specific actions. 5. Learn to display data on the micro:bit's LED matrix and interpret the results.	1. Develop a new project using MakeCode for micro:bit. 2. Add the 'stopbit' extension to the toolbox for programming the traffic lights kit. 3. Create and utilise variables 'startTime', 'endTime', and 'reactionTime' to measure reaction time. 4. Program button A to initiate the traffic light sequence and record the start time. 5. Program button B to record the end time and calculate the reaction time.

Week 3

Lesson: Pedestrian Crossing

Beginner	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare for this interactive lesson by familiarising yourself with the MakeCode editor for micro:bit and the STOP:bit traffic lights. Understand how to add the 'stopbit' extension and create a 'seconds' variable. Be ready to guide students through coding a traffic light sequence, including red light display, button press detection, and the full traffic light sequence. Ensure you know how to download the code to a micro:bit for testing.

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

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- Traffic Lights Kit
- Phillips Screwdriver

Learning Goals	Learning Outcomes
1. Understand and apply the use of MakeCode editor for micro:bit in creating a new project. 2. Learn to add and utilise the "stopbit" extension for programming the STOP:bit traffic lights kit. 3. Develop skills in creating and manipulating variables, specifically the 'seconds' variable in this context. 4. Gain proficiency in coding for specific outcomes such as displaying the red light and detecting button press on the micro:bit. 5. Master the sequence of traffic lights and their corresponding symbols on the micro:bit, and successfully download and test the code.	1. Develop a new project and attach STOP:bit traffic lights to the micro:bit. 2. Add the "stopbit" extension to the MakeCode editor toolbox. 3. Create a variable named 'seconds' for the countdown function. 4. Program the micro:bit to display a red light and an 'X' symbol at the start. 5. Implement button press detection to simulate a pedestrian waiting to cross. 6. Code a traffic light sequence with green, yellow, and red lights, along with appropriate wait times. 7. Download and test the code on the micro:bit.

Week 4

Lesson: Build your Move Motor Sensor Car

Intermediate	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Ensure all materials are ready, including the Microbit and 4 AA batteries. Guide students through the step-by-step instructions provided in the yellow booklet, ensuring they understand each stage of assembly, connection to Makecode, and adding the Move Motor Extension. Facilitate their understanding of coding the motors, using the buzzer, Zip LEDs, line following sensors, and the distance sensor. Encourage exploration and experimentation once the Move Motor Sensor Car is built.

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

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- Move Motor Car

Learning Goals	Learning Outcomes
1. Develop practical skills in assembling a Move Motor Sensor Car. 2. Understand how to connect the Move Motor Sensor Car to Makecode. 3. Acquire coding skills for controlling the motors, buzzer, and LEDs of the Move Motor Sensor Car. 4. Learn to utilise the line following and distance sensors for navigation. 5. Encourage exploration and creativity in coding for different movements and LED usage.	1. Identify and organise components of the Move Motor Sensor Car kit. 2. Assemble the Move Motor Sensor Car following the provided instructions. 3. Connect the assembled car to Makecode and add the Move Motor Extension. 4. Code the motors, buzzer, Zip LEDs, line following sensors, and distance sensor of the car. 5. Apply learned skills to explore and create new movements and LED patterns.

Week 5

Lesson: Line Following Car

Intermediate	60 mins	Teacher/Student led	Student Quiz	Student Challenge

In this lesson, students will program a Move Motor Car to follow a line track using a Microbit. They will create a new project on the MakeCode website, add the kitronik-move-motor extension, and create variables for the left and right line sensors and their difference. Students will then program the car to turn right, left, and move forward based on these sensor readings. After testing their code on a track, students can tweak the code to improve the car's speed and performance on more complex tracks.

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

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- Move Motor Car

Learning Goals	Learning Outcomes
1. Understand and apply the concept of programming a Microbit to control a Move Motor Car. 2. Create and manipulate variables to store sensor values and control the car's movements. 3. Implement conditional logic to guide the car's movements based on sensor readings. 4. Use LEDs for visual feedback and enhance the functionality of the car. 5. Experiment with code modifications to optimise the car's performance on different tracks.	1. Programme the Move Motor Car to follow a line track using a Microbit. 2. Create a new project on the https://makecode.microbit.org website. 3. Add the kitronik-move-motor extension to the project and utilise the custom blocks to program the Move Motor car. 4. Create and utilise variables to store values of the left and right line sensors and their difference. 5. Set up the LEDs on the Move Motor car to light up different colours depending on the car's direction. 6. Programme the car to turn right when the left sensor reads a higher darker value than the right sensor. 7. Programme the car to turn left when the right sensor reads a higher darker value than the left sensor. 8. Programme the car to move forwards when the left and right sensors have similar readings. 9. Test the programmed car on a track and observe its autonomous driving. 10. Tweak the code to improve the car's speed and performance on different tracks.

Week 6

Lesson: Car Distance Sensors

Intermediate	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to introduce students to the concept of ultrasonic sensors and how they function. Guide them through creating a new project on the MakeCode website, adding the kitronik-move-motor extension. Assist them in programming the sensor to measure distance and display it on the Microbit. Progress to programming the car to maintain a 10cm distance from an object, including reversing. Finally, challenge students to improve the code, adding lights and randomised movement to enhance the car's obstacle avoidance.

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

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- Move Motor Car

Learning Goals	Learning Outcomes
1. Understand the function and operation of ultrasonic sensors. 2. Develop skills in creating a new project using the kitronik-move-motor extension. 3. Acquire the ability to program a sensor to measure distance. 4. Learn to code a car to maintain a specific distance from an object. 5. Enhance problem-solving skills by programming the car to reverse and maintain distance.	1. Understand the function and operation of an ultrasonic sensor. 2. Create a new project on the MakeCode Microbit website and add the necessary extension. 3. Program the sensor to measure and display the distance to an object. 4. Modify the code to make the car maintain a distance of 10cm from an object. 5. Enhance the code to reverse the car until it is exactly 10cm away from an object. 6. Program the car to free roam and avoid objects by stopping, reversing, and turning right when an object is detected within 10cm. 7. Improve the code for better navigation and add lights for visual feedback.

Week 7

Lesson: Tilt Remote Control Car

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

In this lesson, students will learn to control a Move Motor car using a Microbit as a remote controller. They will create two code projects: one for the remote control and another for the car. The lesson involves programming the Microbit to detect tilts in different directions and send corresponding messages to the car. The students will also add code to stop the car and to light up the LEDs on the car in different colours. Ensure each remote and car set uses a different radio group to avoid crossed signals.

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

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- Move Motor Car

Learning Goals	Learning Outcomes
1. Understand and apply the concept of radio communication between two Microbits. 2. Programme a Microbit to send specific messages based on different gestures. 3. Develop the ability to programme a Move Motor car to respond to different messages received. 4. Test and debug the code to ensure the car responds correctly to the remote control. 5. Extend the project by adding additional features such as LED light changes.	1. Programme a Microbit as a remote control to send directional commands. 2. Programme a Microbit to receive and execute directional commands in a Move Motor car. 3. Test and debug the code to ensure correct functioning of the remote-controlled car. 4. Download and implement the code onto the Microbits. 5. Extend the code to include LED light changes in response to different commands as an additional challenge.

Week 8

Lesson: Traffic Lights and Car Communication

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

This lesson involves coding a set of traffic lights and a robot car using Microbits. Students will program the traffic lights to display a sequence and broadcast the light being shown. The robot car will receive this broadcast and decide whether to stop or go. The lesson involves creating two code projects, adding a 'stopbit' extension, programming a sequence, broadcasting the state, programming the car, receiving the message, downloading the code, and an additional challenge. Teachers should ensure they have the necessary equipment and familiarise themselves with the coding platforms used.

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

- Chromebook/Laptop/PC

Required equipment for this lesson:

- Microbit
- Traffic Lights Kit
- Move Motor Car

Learning Goals	Learning Outcomes
1. Understand and apply the concept of radio communication between Microbits. 2. Program a sequence of traffic light signals using code blocks. 3. Develop skills to broadcast and receive specific messages based on traffic light states. 4. Control the movement of a robot car based on received messages. 5. Enhance problem-solving skills by modifying the code to respond based on the proximity of the car to the traffic lights.	1. Code a set of traffic lights to run through a sequence and broadcast the displayed light. 2. Program a robot car to receive the broadcast and decide whether to stop or go based on the traffic light signal. 3. Use the "stopbit" extension to create custom code blocks for programming the traffic lights kit. 4. Program the car to move at different speeds or stop, depending on the received message from the traffic lights. 5. Modify the code to make the car respond to the traffic lights based on its proximity to them.

Module: Discovering Artificial Intelligence

This module explores the fascinating world of artificial intelligence (AI), starting with an introduction to AI models, their types, applications, and limitations. Students will gain hands-on experience creating image and pose models using Google's Teachable Machine, and applying these models in interactive games using Scratch. The module culminates in a project where students conceptualise, plan, and build their own AI Scratch project, applying their newfound knowledge and skills. Teachers should familiarise themselves with the tools and concepts, and be prepared to guide students through each step, encouraging creativity and problem-solving throughout.

Duration	**Equipment**
4 weeks	Students can use any of these devices: • Chromebook/Laptop/PC • iPad/Tablet Required Equipment: • Webcam/camera
Module Goals	**Module Outcomes**
1. Understand the fundamentals of AI models, their types, applications, limitations, and ethical considerations. 2. Develop an image model using Google's Teachable Machine and apply it in a practical project. 3. Create an interactive game using Scratch and Google Teachable Machine, incorporating elements of randomisation and conditionals. 4. Design and develop a pose model using Google's Teachable Machine, and apply it in a space game project. 5. Conceptualise, plan, and execute an original AI Scratch project, demonstrating creativity, problem-solving, and application of AI knowledge.	1. Understand and explain the function, types, applications, and limitations of AI models, including ethical considerations. 2. Create an image model using Google's Teachable Machine for a rock, paper, scissors game. 3. Develop a Rock, Paper, Scissors game using Scratch and Google Teachable Machine, incorporating variables, randomisation, and conditionals. 4. Create a pose model using Google's Teachable Machine for a space game, understanding the importance of testing and adjusting the model. 5. Conceptualise, plan, and build a unique AI Scratch project, demonstrating creativity, problem-solving, and the ability to seek and incorporate feedback.

Week 1

Lesson: An Introduction to AI Models

Beginner	20 mins	Teacher/Student led	Student Quiz

Prepare to introduce students to AI models, explaining their function and various types. Discuss different learning methods such as supervised, unsupervised, and reinforcement learning. Explore the diverse applications of AI models, from speech recognition to autonomous vehicles. Discuss the limitations of AI models, including data quality and computational resources. Finally, delve into the ethics of AI models, discussing responsibility, privacy, transparency, and fairness.

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

- Chromebook/Laptop/PC
- iPad/Tablet

Learning Goals	Learning Outcomes
1. Understand the concept and purpose of AI models. 2. Identify different types of AI models and their learning methods. 3. Recognise various applications of AI models in real-world scenarios. 4. Appreciate the limitations and challenges associated with AI models. 5. Reflect on the ethical considerations in the use of AI models.	1. Identify and describe the different types of AI models: Supervised Learning, Unsupervised Learning, and Reinforcement Learning. 2. Explain the various applications of AI models, including speech recognition, image recognition, natural language processing, recommendation systems, and autonomous vehicles. 3. Discuss the limitations of AI models, focusing on data quality, computational resources, transparency, privacy, and security. 4. Understand the ethical considerations related to AI models, including responsibility, privacy, transparency, and fairness. 5. Demonstrate a basic understanding of how AI models function, their uses, limitations, and ethical implications.

Lesson: Create an Image Model

Intermediate	20 mins	Teacher/Student led	Student Quiz

Familiarise yourself with Google's Teachable Machine tool before the lesson. Ensure students understand the concept of machine learning and how it applies to image recognition. Encourage students to take clear images for their classes and emphasise the importance of quality over quantity. Guide them through the process of training, testing, and exporting their models. Reinforce the practical application of these skills in future projects.

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

- Chromebook/Laptop/PC
- iPad/Tablet

Required equipment for this lesson:

- Webcam/camera

Learning Goals	Learning Outcomes
1. Understand and utilise Google's Teachable Machine to create an image model. 2. Create and define classes within an image model project. 3. Add and manage image samples to each class for effective model training. 4. Train, test, and refine the image model to ensure accurate gesture recognition. 5. Export and save the created image model for future use in projects.	1. Utilise Google's Teachable Machine to create an image model. 2. Create and categorise classes within an image model project. 3. Add and record images to each class using a webcam. 4. Train the image model using the added images and understand the process of machine learning. 5. Test the model's performance, make necessary adjustments, and export the model for future use.

Week 2

Lesson: Scratch AI Rock, Paper, Scissors Game

Intermediate	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to guide students through creating a Rock, Paper, Scissors game using Scratch and Google Teachable Machine. Ensure they understand the use of variables, randomisation, and conditionals. They'll need to set up Scratch and TM2Scratch, add a sprite, create variables, and load a Teachable Machine Model. They'll also learn to set a confidence threshold, get player choice, determine game outcomes, and add enhancements. Encourage creativity and problem-solving throughout.

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

- Chromebook/Laptop/PC
- iPad/Tablet

Required equipment for this lesson:

- Webcam/camera

Learning Goals	Learning Outcomes
1. Develop a Rock, Paper, Scissors game using Scratch and Google Teachable Machine. 2. Understand and apply the use of variables in Scratch for storing player's choice, computer's choice, and the result of the game. 3. Implement randomisation in Scratch to simulate the computer's choice in the game. 4. Integrate Google Teachable Machine Image models in Scratch projects for gesture recognition. 5. Understand and adjust the confidence threshold for AI model to improve accuracy of gesture recognition.	1. Develop a Rock, Paper, Scissors game using Scratch and Google Teachable Machine. 2. Set up Scratch and TM2Scratch for the game development. 3. Create and utilise variables to store player's choice, computer's choice, and the game result. 4. Implement randomisation for computer's choice in the game. 5. Load and use a Teachable Machine Image model for hand gesture recognition. 6. Set and adjust the confidence threshold for the AI model. 7. Recognise and interpret player's choice through hand gestures. 8. Develop game logic to determine the game result: draw, win, or lose. 9. Improve the game by enhancing the image model, adding new features like sound effects, and improving user interaction.

Week 3

Lesson: Create a Pose Model

Intermediate	20 mins	Teacher/Student led	Student Quiz

Prepare to guide students through creating a pose model using Google's Teachable Machine. Familiarise yourself with the tool and the process of creating classes, adding images, and training the model. Be ready to troubleshoot any issues with webcam permissions or image quality. Ensure students understand the importance of testing their model and making necessary adjustments. Finally, assist them in exporting their model for future use in projects like an AI-powered space game.

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

- Chromebook/Laptop/PC
- iPad/Tablet

Required equipment for this lesson:

- Webcam/camera

Learning Goals	Learning Outcomes
1. Develop an understanding of Google's Teachable Machine and its application in creating pose models. 2. Acquire skills to create and categorise classes within a pose model. 3. Learn to add and manage image samples for each class to train the model. 4. Gain proficiency in training and testing the model for different poses. 5. Master the process of exporting the model for future use in other projects.	1. Operate Google's Teachable Machine to create a pose model. 2. Define and create classes for the pose model. 3. Add and categorise images into the respective classes: Tilt Left, Tilt Right, and No Tilt. 4. Train the pose model using the categorised images and test its performance. 5. Export the created pose model and obtain a shareable link for future use.

Lesson: Scratch AI Pose Space Game

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to guide students in creating a Scratch AI Pose Space Game. They'll learn to use Scratch and Google Teachable Machine to control a spaceship with tilt poses. They'll set up Scratch, add a rocketship sprite, load a Teachable Machine Model, display pose labels, set a confidence threshold, and make the spaceship move. They'll also add a star sprite, make stars fall, and face a challenge to improve their game. Ensure students understand each step, and encourage creativity in the challenge.

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

- Chromebook/Laptop/PC
- iPad/Tablet

Required equipment for this lesson:

- Webcam/camera

Learning Goals	Learning Outcomes
1. Develop skills in using Scratch and Google Teachable Machine to create a game. 2. Understand how to control a sprite using pose models. 3. Learn to set up and adjust the confidence threshold for an AI model. 4. Gain knowledge on how to create and manipulate clones of sprites in Scratch. 5. Apply creativity to enhance and personalise the game with additional features.	1. Create a Space game using Scratch and Google Teachable Machine. 2. Set up Scratch and TMPose2Scratch for a new project. 3. Integrate a Teachable Machine Pose model into the Scratch project. 4. Control a sprite's movement using pose labels and confidence thresholds. 5. Enhance the game by adding falling sprites and scoring mechanisms.

Week 4

Lesson: Crafting Your Own AI Project

Advanced	60 mins	Teacher/Student led

In this lesson, students will utilise their knowledge of AI and Scratch to create their own AI project. They will brainstorm ideas, focusing on real-life routines or challenges that could be enhanced with AI. After selecting their favourite idea, they will create a project proposal, seek feedback, refine their idea, and plan their project. They will then prototype and code their project, before presenting and demonstrating their work. Finally, they will reflect on their learning journey and the process of creating their project.

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

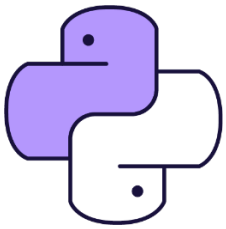
- Chromebook/Laptop/PC
- iPad/Tablet

Required equipment for this lesson:

- Webcam/camera

Learning Goals	Learning Outcomes
1. Develop the ability to conceptualise and plan an AI project. 2. Enhance brainstorming skills and generate creative AI project ideas. 3. Gain proficiency in creating and refining a project proposal. 4. Acquire skills in prototyping and coding an AI project using Scratch and Google Teachable Machine. 5. Improve presentation skills and ability to reflect on the learning process and project outcomes.	1. Generate and evaluate 3-5 AI project ideas, drawing inspiration from daily routines or challenges. 2. Formulate a detailed project proposal, including project name, purpose, required features, and necessary components. 3. Seek and incorporate feedback from peers or teachers to refine the project idea and plan. 4. Code, prototype, and test the core features of the AI project, using problem-solving skills to overcome any issues. 5. Present and demonstrate the final AI project, reflecting on the challenges faced, solutions found, and lessons learned.

Module: Introduction to Python



This module provides an introduction to Python programming, starting with basic syntax and the Microbit Python editor. Teachers should guide students through setting up their first project, creating a simple program, and introducing code sequence. The module progresses to cover variables, loops, conditional statements, operators, arrays, and functions. Each module includes a practical project to reinforce learning. The module culminates in a final project where students apply their skills to create a unique MicroPython project. Teachers should encourage experimentation and provide regular feedback.

Duration	Equipment
10 weeks	Students can use any of these devices: • Chromebook/Laptop/PC Required Equipment: • Microbit
Module Goals	**Module Outcomes**
1. Understand and apply basic Python syntax and programming concepts using the Micro:bit Python editor. 2. Master the use of variables, including declaration, assignment, and manipulation in Python. 3. Comprehend and implement different types of loops and conditional statements in Python programming. 4. Learn about and apply comparison operators, logical operators, and conditional Booleans in Python. 5. Gain proficiency in working with arrays, including creating, manipulating, and applying advanced array tactics in Python.	1. Understand and apply basic Python syntax and use the Micro:bit Python editor to create simple programs. 2. Declare, assign, and manipulate variables in Python, culminating in the creation of a higher or lower game. 3. Understand and implement different types of loops in Python, including while loops, for loops, and nested loops, and apply these in a reaction time game. 4. Use conditional statements in Python to make decisions in code and apply these concepts in a Dice Roller project. 5. Understand and use comparison operators, logical operators, and conditional Booleans in Python, and apply these in a Temperature Indicator project. 6. Work with arrays in Python, including creating, manipulating, and retrieving elements from a list, and apply these skills in an LED light pattern project. 7. Perform advanced operations with arrays in Python, including sorting, finding the length of a list, and counting occurrences, and apply these in a strong password generator project. 8. Understand the differences between procedures and functions in Python and apply this knowledge in a weather station project. 9. Understand the distinctions between local and global variables, understand variable scope, and apply these concepts in a Micro:bit temperature logger project. 10. Conceptualize, plan, and build a unique project using Python and the Micro:bit, applying all the skills and knowledge acquired throughout the course.

Week 1

Lesson: An Introduction to Python

Intermediate	30 mins	Teacher/Student led	Student Quiz

Prepare to introduce Python as a beginner-friendly programming language, highlighting its use in various fields. Familiarise yourself with the Micro:bit Python editor for practical application. Discuss Python's syntax, particularly the importance of indentation and comments. Guide students through writing their first Python program and adding comments for clarity. Explain the sequence of code execution using a simple program. Encourage further practice and exploration post-lesson.

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 basics of Python programming language. 2. Utilise the Micro:bit Python editor for code writing and testing. 3. Comprehend and implement Python's indentation rules to define code blocks. 4. Use Python comments for code explanation and documentation. 5. Write, run, and debug simple Python programs using Micro:bit.	1. Understand and explain the basics of Python programming language and its application in various fields. 2. Access and navigate the Micro:bit Python editor for writing and testing Python code. 3. Apply Python indentation rules to define code blocks correctly. 4. Use Python comments to add notes and explanations to the code. 5. Write, run, and debug a simple Python program using the Micro:bit Python editor.

Week 2

Lesson: Mastering Variables

Intermediate	60 mins	Teacher/Student led	Student Quiz	Student Challenge

This lesson guides students through the process of mastering variables in Python. They will learn about variable declaration, assignment, types, and naming conventions. The lesson also includes practical exercises such as creating a higher or lower game using the Microbits Python editor. Teachers should ensure students understand the concept of variables, their types, and how to manipulate them. They should also facilitate the game creation exercise, helping students apply their knowledge in a practical context.

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 concept of variables in Python, including their declaration, assignment, and types. 2. Learn and apply good variable naming conventions in Python. 3. Manipulate variable values through operations such as incrementing, decrementing, and string concatenation. 4. Apply knowledge of variables in creating a simple higher or lower game using the Microbits Python editor. 5. Gain familiarity with importing and using libraries in Python, specifically for game development.	1. Understand and apply the concept of variables in Python, including declaration, assignment, and types. 2. Manipulate variable values through incrementing, decrementing, and string concatenation. 3. Adhere to good naming conventions for variables, specifically snake_case and camelCase. 4. Import and utilise libraries in Python, specifically for the creation of a higher or lower game. 5. Develop a simple higher or lower game using variables, loops, conditionals, and libraries in Python.

Week 3

Lesson: Looping Around

Intermediate	60 mins	Teacher/Student led	Student Quiz	Student Challenge

This lesson will guide students through understanding loops in Python, focusing on while loops, for loops, and nested loops. They will learn how Python uses indentation to define code blocks and how to break out of loops. The lesson includes practical exercises to reinforce learning, such as creating a project to measure reaction times. Teachers should ensure students understand the importance of consistent indentation and how loops can be used to control the flow of a program.

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
• Understand the importance of indentations in Python and how they define code blocks. • Learn how to use 'while' loops and 'for' loops to repeat code execution based on conditions. • Explore nested loops and how they can be used to iterate through multiple dimensions. • Learn how to break out of loops using the 'break' statement when a specific condition is met. • Apply the knowledge of loops and control structures to create a simple reaction time game. • Develop problem-solving skills by modifying and extending the provided code examples.	• By the end of this lesson, students will be able to identify and differentiate between while loops, for loops, and nested loops in Python. • Students will be able to demonstrate the use of proper indentation in Python code to define code blocks. • Students will be able to create and manipulate while loops and for loops to execute a block of code multiple times. • Students will be able to implement nested loops to control the flow of their program through multiple levels of iteration. • Students will be able to use the 'break' statement to exit a loop prematurely based on a specific condition. • Students will be able to apply their knowledge of loops and control structures to create a simple reaction time game using the micro:bit's LED matrix and buttons.

Week 4

Lesson: Making Decisions

Intermediate	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to guide students through understanding conditional statements in MicroPython, including 'if', 'elif', and 'else'. They will create a simple project to reinforce their understanding, and then apply these concepts to a Dice Roller project. Ensure students understand how to use the Microbit Python Editor, and are comfortable with concepts such as loops, conditions, and using the micro:bit's accelerometer. Encourage experimentation with different conditions and scenarios to deepen their understanding.

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. Comprehend and apply conditional statements in coding, including 'if', 'elif', and 'else'. 2. Develop a basic project using 'if' statements to demonstrate understanding. 3. Understand and implement 'elif' and 'else' statements to create complex decision-making structures. 4. Grasp the concept of nested 'if' statements and their application in coding. 5. Create a Dice Roller project utilising 'if', 'elif', and 'else' statements, demonstrating the ability to make decisions in code based on specific conditions.	1. Apply conditional statements in Python code, specifically `if`, `elif`, and `else` statements. 2. Construct a simple `if` statement to check a condition and execute a block of code. 3. Create complex decision-making structures using `elif` and `else` statements in conjunction with `if` statements. 4. Utilise nested `if` statements to check multiple conditions within a single `if` statement. 5. Develop a Dice Roller project using the micro:bit's accelerometer, random number generation, and conditional statements to display different outcomes.

Week 5

Lesson: Operators Decoded

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to guide students through understanding comparison and logical operators, and conditional Booleans in MicroPython. They'll apply these concepts in a practical project, creating a temperature indicator using the Microbit's online editor. Ensure they understand how to use these operators in 'if' and 'elif' statements. Encourage experimentation with different values to see how it affects conditions. The final project will involve using comparison and logical operators to determine temperature ranges and display appropriate images.

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 comparison operators in MicroPython. 2. Understand and apply logical operators in MicroPython. 3. Understand and utilise conditional Booleans in MicroPython. 4. Create a temperature indicator project using MicroPython and Micro:bit. 5. Apply knowledge of operators and conditional Booleans in practical coding scenarios.	1. Understand and apply comparison operators in MicroPython. 2. Utilise logical operators to combine conditional statements in MicroPython. 3. Implement conditional Booleans to make decisions based on the result of a condition. 4. Create a temperature indicator project using the built-in temperature sensor of Micro:bit. 5. Apply comparison and logical operators to determine temperature range and display appropriate images on Micro:bit.

Week 6

Lesson: Array Essentials

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

In this lesson, teachers will guide students through the basics of working with arrays in MicroPython, using the micro:bit Python editor. Students will learn what an array is, how to create a list, retrieve and change list elements, add and remove elements from a list. The lesson culminates in a project where students will use arrays to create patterns of lights on the micro:bit LED display. Teachers should ensure students understand each step before moving on to the next.

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 arrays in MicroPython. 2. Create, retrieve, and modify elements in a list. 3. Add and remove elements from a list. 4. Use arrays to create patterns of lights on the micro:bit LED display. 5. Combine and manipulate multiple arrays to create complex data structures.	1. Understand and define arrays in MicroPython. 2. Create, retrieve, and manipulate elements in a list. 3. Add and remove elements from a list using append(), extend(), remove(), and pop() methods. 4. Use arrays to store and manage data in MicroPython programs. 5. Apply array manipulation skills to create an LED light pattern project on a micro:bit.

Week 7

Lesson: Advanced Array Tactics

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to guide students through advanced operations on Python lists using MicroPython. The lesson covers sorting lists in ascending and descending order, finding the length of a list, counting occurrences in lists, and applying these skills to create a strong password generator. Ensure students understand the use of sort(), len(), and count() methods, and how to use loops and random.choice() function. Encourage them to experiment with the code in their own projects.

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. Master advanced operations on lists in Python, including sorting in ascending and descending order. 2. Understand how to determine the length of a list using the len() function. 3. Learn to count the occurrences of a specific item in a list using the count() method. 4. Apply the learned concepts in a practical project to create a strong password generator. 5. Develop skills in manipulating and analysing data stored in lists.	1. Sort a list in ascending order using the sort() method in MicroPython. 2. Sort a list in descending order using the sort(reverse=True) method in MicroPython. 3. Determine the length of a list using the len() function in MicroPython. 4. Count occurrences of a specific item in a list using the count() method in MicroPython. 5. Generate a strong, random password using a combination of character sets in MicroPython.

Week 8

Lesson: Function Junction

Advanced	60 mins	Teacher/Student led	Student Quiz	Student Challenge

In this lesson, students will delve into Python programming, focusing on procedures and functions. They will learn the difference between the two, create simple procedures and functions using the micro:bit, and understand the use of parameters in functions. The lesson culminates in a project where students design a simplified weather station using their new skills. This hands-on approach will help reinforce their understanding of procedures and functions in Python.

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 difference between procedures and functions in Python programming. 2. Create and utilise procedures in Python code. 3. Create and utilise functions in Python code, including those that return values. 4. Develop functions with parameters to enhance flexibility and functionality. 5. Apply knowledge of procedures and functions to create a simple weather station project using MicroPython and Micro:bit.	1. Differentiate between procedures and functions in Python programming. 2. Create and utilise a procedure in Python to display a smiley face on the micro:bit's display. 3. Develop a function in Python that returns the square of a number and display the result on the micro:bit. 4. Construct a function with parameters in Python that takes two numbers and returns their sum, displaying the result on the micro:bit. 5. Design and implement a simplified weather station on the micro:bit using procedures and functions.

Week 9

Lesson: Scope Showdown: Local vs. Global

Expert	60 mins	Teacher/Student led	Student Quiz	Student Challenge

Prepare to guide students through understanding the concept of local and global variables in programming. Start with an introduction to the term 'scope', followed by a detailed explanation of local variables using Python code. Then, introduce global variables and their usage. Discuss best practices for using global variables. The lesson culminates in a practical project where students create a temperature logger using MicroPython, applying their understanding of local and global variables. Finally, wrap up the lesson by reinforcing the importance of variable scope in coding projects.

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 concept of variable scope in programming. 2. Distinguish between local and global variables and their usage. 3. Apply the concept of local and global variables in Python programming. 4. Adhere to best practices when using global variables. 5. Develop a Micro:bit temperature logger project using both local and global variables.	1. Understand and differentiate between local and global variables in programming. 2. Identify the scope of a variable and its accessibility within a program. 3. Apply the concept of local variables within a function, demonstrating their limited scope. 4. Utilise global variables appropriately within a program, demonstrating their wider scope. 5. Combine the use of local and global variables in a practical project, demonstrating understanding of best practices.

Week 10

Lesson: Python Showcase

Expert	60 mins	Teacher/Student led

Prepare to guide students through a creative process of conceptualising, planning, and building a MicroPython project. Encourage brainstorming, idea selection, and project proposal creation. Facilitate feedback sessions and assist in refining ideas. Support students during the coding process, ensuring they test their code regularly. Finally, help students prepare a presentation to demonstrate their project, followed by a reflection on their learning journey.

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 problem-solving skills through conceptualising and planning a MicroPython project. 2. Enhance brainstorming abilities and apply previous knowledge to generate project ideas. 3. Improve communication skills by presenting project proposals and seeking feedback. 4. Strengthen coding skills through prototyping and building a MicroPython project. 5. Reflect on the learning experience, identifying challenges faced and strategies used to overcome them.	1. Generate and refine project ideas related to MicroPython and Micro:bit. 2. Develop a comprehensive project proposal including purpose, features, and required components. 3. Seek and incorporate feedback to improve the project concept. 4. Code, test, and debug a MicroPython project on the Micro:bit. 5. Present the completed project, demonstrating its features and discussing the development process.

© 2025 Coding Ireland. All rights reserved.

This learning plan and its contents are provided exclusively for use with the Digital Skills Curriculum and may not be reproduced, distributed, or shared without prior written permission from Coding Ireland. For more information, please visit www.codingireland.ie.