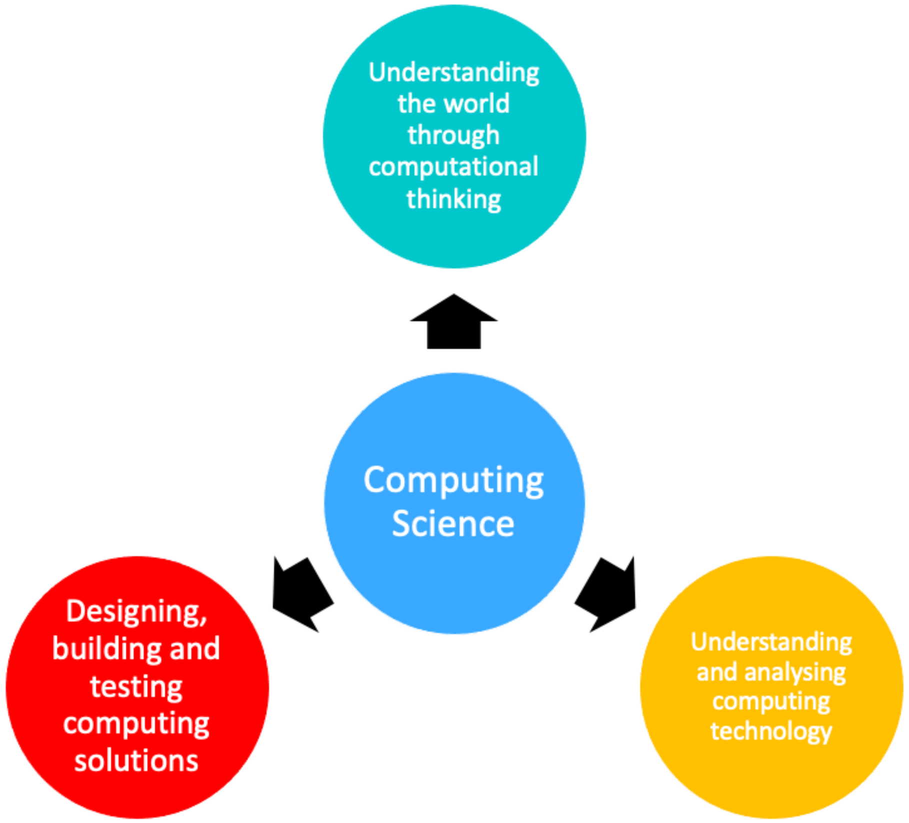
Computing Science Second Level

Digital Learning and Literacy in Fife


Understanding the world through computational thinking
Computing Science
Designing, building and testing computing solutions
Understanding and analysing computing technology

Second Level Computing Science

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding the world through computational thinking	I understand the operation of a process and its outcome. I can structure related items of information. TCH 2-13a	• Compares activities consisting of a single sequence of steps with those consisting of multiple parallel steps, for example, making tomato sauce and cooking pasta to be served at the same time. • Identifies algorithms/instructions that include repeated groups of instructions a fixed number of times and/or loops until a condition is met. • Identifies when a process is not predictable because it has a random element for example, a board game which uses dice. • Structures related items of information for example, a family tree (MNU 2- 20b). • Uses a recognised set of instructions/ an algorithm to sort real worlds objects for examples, books in a library or trading cards.
Understanding and analysing computer technology	I can explain core programming language concepts in appropriate technical language. TCH 2-14a I understand how information is stored and how key components of computing technology connect and interact through networks. TCH 2-14b	• Explains the meaning of individual instructions (including variables and conditional repetition) in a visual programming language. • Predicts what a complete program in a visual programming language will do when it runs, including how the properties of objects for example, position, direction and appearance change as the program runs through each instruction. • Explains and predicts how parallel activities interact. • Demonstrates an understanding that all computer data is represented in binary for example, numbers, text, black and white graphics. • Describes the purpose of the processor, memory and storage and the relationship between them. • Demonstrates an understanding of how networks are connected and used to communicate and share information, for example the internet.
Designing, building and testing computing solutions	I can create, develop and evaluate computing solutions in response to a design challenge. TCH 2-15a	• Creates programs in a visual programming language including variables and conditional repetition. • Identifies patterns in problem solving and reuses aspects of previous solutions appropriately for example, reuse code for a timer, score counter or controlling arrow keys. • Identifies any mismatches between the task description and the programmed solution, and indicates how to fix them.

Second Level Computational Thinking

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding the world through computational thinking	I understand the operation of a process and its outcome. I can structure related items of information. TCH 2-13a	• Compares activities consisting of a single sequence of steps with those consisting of multiple parallel steps, for example, making tomato sauce and cooking pasta to be served at the same time. • Identifies algorithms/instructions that include repeated groups of instructions a fixed number of times and/or loops until a condition is met. • Identifies when a process is not predictable because it has a random element for example, a board game which uses dice. • Structures related items of information for example, a family tree (MNU 2- 20b). • Uses a recognised set of instructions/ an algorithm to sort real worlds objects for examples, books in a library or trading cards.

What the learning may look like in Fife	Glossary of terms
• Reading code and describing what will happen at each step. • Experience of following instructions with recipes, Scottish Country Dancing, during lessons, etc. • Create a data structure to identify the relationship between items e.g. a family tree, library books, types of vehicles. • When exploring sorting and identifying patterns link with Shape, Position & Movement and Data Handling Progression Pathway. • Use resources from Barefoot Computing to support exploration of Computing Science. Visit the Computing Science Progression site for further ideas and resources.	• **Forever Loop (Infinite)** - A piece of code that will run continuously until the program ends as it does not have a functional exit • **Nested Loop** – A loop within a loop • **Parallel Process** - Multiple processes all running at the same time (simultaneously) • **Predict** – To make known in advance • **Simultaneous** – At the same time For the full glossary at Second Level, click here.

Second Level Computational Thinking

Identifies the part of a given algorithm responsible for the creation of a random element/variable.

Identifies when a process is not predictable because it has a random element for example, a board game which uses dice.

Predicts the outcome of algorithms/ instructions that include repeated groups of instructions.

Identifies algorithms/instructions that include repeated groups of instructions a fixed number of times and/or loops until a condition is met.

Explains why some instructions consist of a single sequence of steps running one after the other compared with others that will have parallel processes to carry out more than one activity at the same time.

Compares activities consisting of a single sequence of steps with those consisting of multiple parallel steps, for example, making tomato sauce and cooking pasta to be served at the same time.

Explores activities that require single steps and multiple parallel steps e.g. everyday activities such as getting dressed, cooking.

Predicts, interprets and discusses data sorting results.

Creates an algorithm that will sort objects into a specific order.

Uses a recognised set of instructions/ an algorithm to sort real worlds objects for examples, books in a library or trading cards.

Structures related items of information for example, a family tree (MNU 2- 20b).

Identifies several ways that data or objects can be structured and how they differ from each other i.e. books in a library/ house numbers and postcodes.

Second Level Analysing Computing Technology

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding and analysing computer technology	I can explain core programming language concepts in appropriate technical language. TCH 2-14a I understand how information is stored and how key components of computing technology connect and interact through networks. TCH 2-14b	• Explains the meaning of individual instructions (including variables and conditional repetition) in a visual programming language. • Predicts what a complete program in a visual programming language will do when it runs, including how the properties of objects for example, position, direction and appearance change as the program runs through each instruction. • Explains and predicts how parallel activities interact. • Demonstrates an understanding that all computer data is represented in binary for example, numbers, text, black and white graphics. • Describes the purpose of the processor, memory and storage and the relationship between them. • Demonstrates an understanding of how networks are connected and used to communicate and share information, for example the internet.

What the learning may look like in Fife	Glossary of terms
• Use visual programming languages such as Scratch and MakeCode to read and build algorithms. • Look at different manufacturers e.g. Intel, AMD, Nvidia, etc. and where these products are found throughout the school devices. • Resources such as Hello Ruby to explore computer components. • Use resources from Barefoot Computing to support exploration of Computing Science. Visit the Computing Science Progression site for further ideas and resources.	• **Binary (code)** - A coding system using the binary digits 0 and 1 to represent a letter, digit, or other character in a computer or other electronic device • **Boolean** - Boolean logic is a form of algebra in which all values are reduced to either TRUE or FALSE • **IP Address** - A computer's unique address e.g.192.168.0.127 - This address is used by computers to communicate across a network • **Network** - Two or more computers connected for the purpose of storing, sharing, and managing data i.e. the internet For the full glossary at Second Level, click here.

Second Level Analysing Computing Technology

Predicts what a complete program in a visual programming language will do when it runs, including how the properties of objects for example, position, direction and appearance change as the program runs through each instruction.

Explains and predicts how parallel activities interact.

Explains the meaning of individual instructions (including variables and conditional repetition) in a visual programming language.

Explain what Boolean logic is and identify what the operators are.

Demonstrates an understanding of different types of loops used in programming and when they would be required i.e. forever/conditional/ count controlled.

Explores variables and when they would be used to achieve a required output.

Demonstrates an understanding of how networks are connected and used to communicate and share information, for example the internet.

Explores how computers communicate over a network and identifies key network hardware devices.

Compares the performance of computer components from different devices and how this affects their usage e.g. the CPU within a netbook versus within a PC.

Describes the purpose of the processor, memory and storage and the relationship between them.

Uses correct vocabulary to describe parts of a computer e.g. motherboard, CPU, RAM, VRAM.

Demonstrates an understanding that all computer data is represented in binary for example, numbers, text, black and white graphics.

Second Level Designing, Building and Testing

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Designing, building and testing computing solutions	I can create, develop and evaluate computing solutions in response to a design challenge. TCH 2-15a	• Creates programs in a visual programming language including variables and conditional repetition. • Identifies patterns in problem solving and reuses aspects of previous solutions appropriately for example, reuse code for a timer, score counter or controlling arrow keys. • Identifies any mismatches between the task description and the programmed solution, and indicates how to fix them.

What the learning may look like in Fife	Glossary of terms
• Play with programmable devices to explore more complex programs involving movement and other actions. • Create algorithms for different purposes e.g. solving problems, creating games, creating animations etc. • When creating algorithms and giving instructions link with Shape, Position & Movement Progression Pathway. • Pose questions to an individual within an IT profession about a design challenge they have had to tackle and how they dealt with this. • Use resources from Barefoot Computing to support exploration of Computing Science. Visit the Computing Science Progression site for further ideas and resources.	• **Glitch** - A sudden, usually temporary malfunction or fault of equipment or computer program • **Predict** – To make known in advance • **Sprite** - An icon in a computer game which can be manoeuvred around the screen by means of a joystick, etc. For the full glossary at Second Level, click here.

Second Level Designing, Building and Testing

Identifies any mismatches between the task description and the programmed solution, and indicates how to fix them.

Demonstrates resilience when experiencing challenge during a testing phase.

Uses logical reasoning to detect problems in an algorithm and use problem solving skills to resolve any issues.

Tests and evaluates a program created in response to given criteria in a design challenge.

Identifies patterns in problem solving and reuses aspects of previous solutions appropriately for example, reuse code for a timer, score counter or controlling arrow keys.

Creates programs in a visual programming language including variables and conditional repetition.

Deconstructs a problem into smaller steps and recognise how they may be similar to previous problems.

Designs and plans programs for a game, story/animation, webpage or programmable device before attempting to create these.