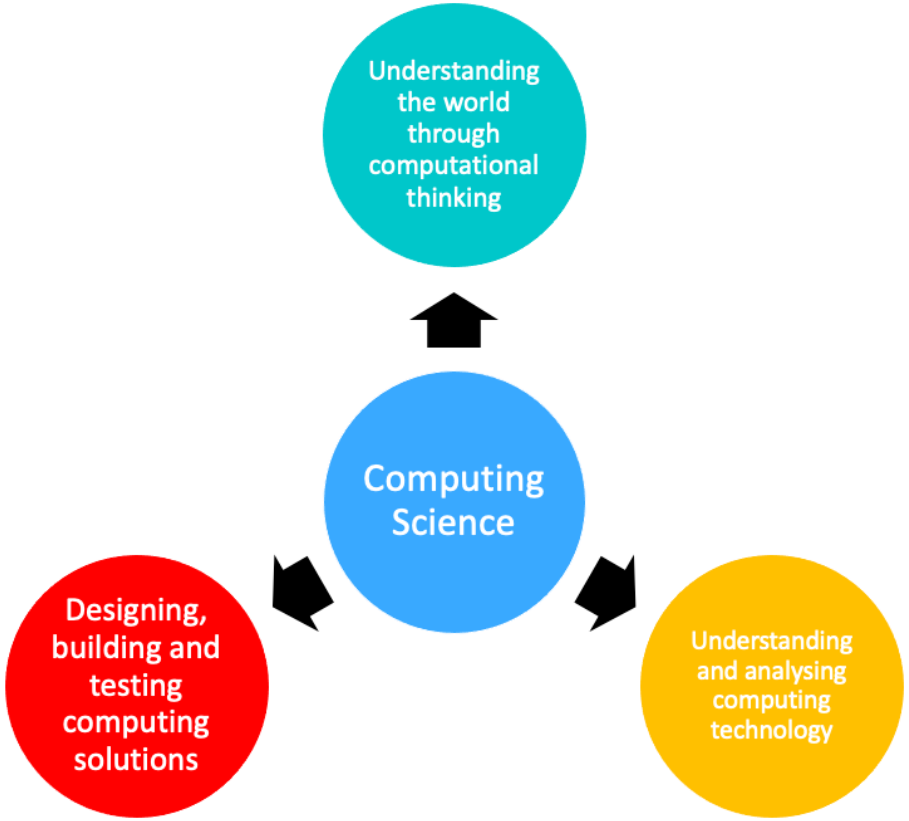
Computing Science First Level



First Level Computing Science

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding the world through computational thinking	I can explore and comment on processes in the world around me making use of core computational thinking concepts and can organise information in a logical way. TCH 1-13a	• Follows sequences of instructions/algorithms from everyday situations for example, recipes or directions, including those with selection and repetition. • Identifies steps in a process and describes precisely the effect of each step. • Makes decisions based on logical thinking including IF, AND, OR and NOT for example, collecting balls in the gym hall but NOT basketballs, line up if you are left-handed OR have green eyes. • Collects, groups and orders information in a logical, organised way using my own and others‟ criteria (MNU 1-20a and b).
Understanding and analysing computer technology	I understand the instructions of a visual programming language and can predict the outcome of a program written using the language. TCH 1-14a I understand how computers process information. TCH 1-14b	• Demonstrates an understanding of the meaning of individual instructions when using a visual programming language (including sequences, fixed repetition and selection). • Explains and predicts what a program in a visual programming language will do when it runs for example, what audio, visual or movement effect will result. • Demonstrates an understanding that computers take information as input, process and store that information and output the results.
Designing, building and testing computing solutions	I can demonstrate a range of basic problem solving skills by building simple programs to carry out a given task, using an appropriate language. TCH 1-15a	• Simplifies problems by breaking them down into smaller more manageable parts. • Constructs a sequence of instructions to solve a task, explaining the expected output from each step and how each to contributes towards solving the task. • Creates programs to carry out activities (using selection and fixed repetition) in an visual programming language. • Identifies when a program does not do what was intended and can correct errors/bugs. • Evaluates solutions/programs and suggests improvements.

First Level Computational Thinking

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding the world through computational thinking	I can explore and comment on processes in the world around me making use of core computational thinking concepts and can organise information in a logical way. TCH 1-13a	• Follows sequences of instructions/algorithms from everyday situations for example, recipes or directions, including those with selection and repetition. • Identifies steps in a process and describes precisely the effect of each step. • Makes decisions based on logical thinking including IF, AND, OR and NOT for example, collecting balls in the gym hall but NOT basketballs, line up if you are left-handed OR have green eyes. • Collects, groups and orders information in a logical, organised way using my own and others' criteria (MNU 1-20a and b).

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. • Use IF, THEN, ELSE, AND, OR , NOT expressions using both code and real-life examples e.g. IF it is raining OR snowing, THEN wear wellies outside, ELSE wear shoes. • Carry out sorting activities to develop classification skills. • 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.	• **Abstraction** - Simplifying things; identifying what is important without worrying too much about the detail. Abstraction allows us to manage complexity • **Selection** - A programming construct in which one section of code or another is executed depending on whether a particular condition is met • **Sequence** - Arrange things in a particular order (computer programs are built up of sequences of instructions) For the full glossary at First Level, click here.

First Level Computational Thinking

Makes decisions based on logical thinking including IF, AND, OR and NOT for example, collecting balls in the gym hall but NOT basketballs, line up if you are left-handed OR have green eyes.

Follows sequences of instructions/algorithms from everyday situations for example, recipes or directions, including those with selection and repetition.

Identifies when steps require a decision and explores how this can be represented by using IF, AND, OR, ELSE, THEN and NOT.

Identifies when steps are repeated and explores how to use loops to represent repetition.

Explores how to read code as part of a sequence.

Identifies steps in a process and describes precisely the effect of each step.

Explains why processes must be carried out in a specific order.

Collects, groups and orders information in a logical, organised way using my own and others' criteria (MNU 1-20a and b).

Sorts and classifies a group of items in different ways to meet different conditions e.g. colour, size.

Sorts and classifies a group of items by asking simple yes / no questions.

First Level Analysing Computing Technology

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding and analysing computer technology	I understand the instructions of a visual programming language and can predict the outcome of a program written using the language. TCH 1-14a I understand how computers process information. TCH 1-14b	• Demonstrates an understanding of the meaning of individual instructions when using a visual programming language (including sequences, fixed repetition and selection). • Explains and predicts what a program in a visual programming language will do when it runs for example, what audio, visual or movement effect will result. • Demonstrates an understanding that computers take information as input, process and store that information and output the results.

What the learning may look like in Fife	Glossary of terms
• Play with programmable devices such as Beebots, etc. • Experience giving and following instructions, e.g. using songs, PE activities, daily routines. • Use visual programming languages such as ScratchJr and Scratch to read and build algorithms. • Use resources from Barefoot Computing to support exploration of Computing Science. Visit the Computing Science Progression site for further ideas and resources.	• **Input** - Data transferred from the outside world into a computer system via some kind of input device such as a keyboard, scanner or storage device • **Output** - The data actively transmitted from within the computer to an external device such as a monitor, storage device or printer • **Predict** – To make known in advance • **Process** - An instance of a computer program that is being run • **Selection** - A programming construct in which one section of code or another is executed depending on whether a particular condition is met For the full glossary at First Level, click here.

First Level Analysing Computing Technology

Explains and predicts what a program in a visual programming language will do when it runs for example, what audio, visual or movement effect will result.

Demonstrates an understanding of the meaning of individual instructions when using a visual programming language (including sequences, fixed repetition and selection).

Explores the output of selection in algorithms.

Explores the output of fixed repetition in algorithms.

Explores the output of sequences in algorithms.

Explores how to read code as part of an algorithm.

Demonstrates an understanding that computers take information as input, process and store that information and output the results.

Explores the difference between inputs and outputs.

First Level Designing, Building and Testing

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Designing, building and testing computing solutions	I can demonstrate a range of basic problem solving skills by building simple programs to carry out a given task, using an appropriate language. TCH 1-15a	• Simplifies problems by breaking them down into smaller more manageable parts. • Constructs a sequence of instructions to solve a task, explaining the expected output from each step and how each to contributes towards solving the task. • Creates programs to carry out activities (using selection and fixed repetition) in a visual programming language. • Identifies when a program does not do what was intended and can correct errors/bugs. • Evaluates solutions/programs and suggests improvements.

What the learning may look like in Fife	Glossary of terms
• Play with programmable devices such as Beebots, etc. • 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. • Use resources from Barefoot Computing to support exploration of Computing Science. Visit the Computing Science Progression site for further ideas and resources.	• **Abstraction** - Simplifying things; identifying what is important without worrying too much about the detail. Abstraction allows us to manage complexity • **Debugging** - Errors in algorithms and code are called 'bugs', and the process of finding and fixing these is called debugging • **Decomposing/Decomposition** - Breaking problems or systems down into smaller, more manageable parts making it easier to manage complexity For the full glossary at First Level, click here.

First Level Designing, Building and Testing

Evaluates solutions/programs and suggests improvements.

Demonstrates resilience when experiencing challenge during a testing phase.

Creates programs to carry out activities (using selection and fixed repetition) in a visual programming language.

Explores using fixed repetition in algorithms and identifies how these appear in different visual programming languages.

Explores using selection in algorithms and identifies how these appear in different visual programming languages.

Identifies when a program does not do what was intended and can correct errors/bugs.

Uses vocabulary such as decomposition and debugging to explain how problems and programs are solved and edited.

Simplifies problems by breaking them down into smaller more manageable parts.

Constructs a sequence of instructions to solve a task, explaining the expected output from each step and how each to contributes towards solving the task.

Explores the designing, building and testing processes when creating an algorithm.