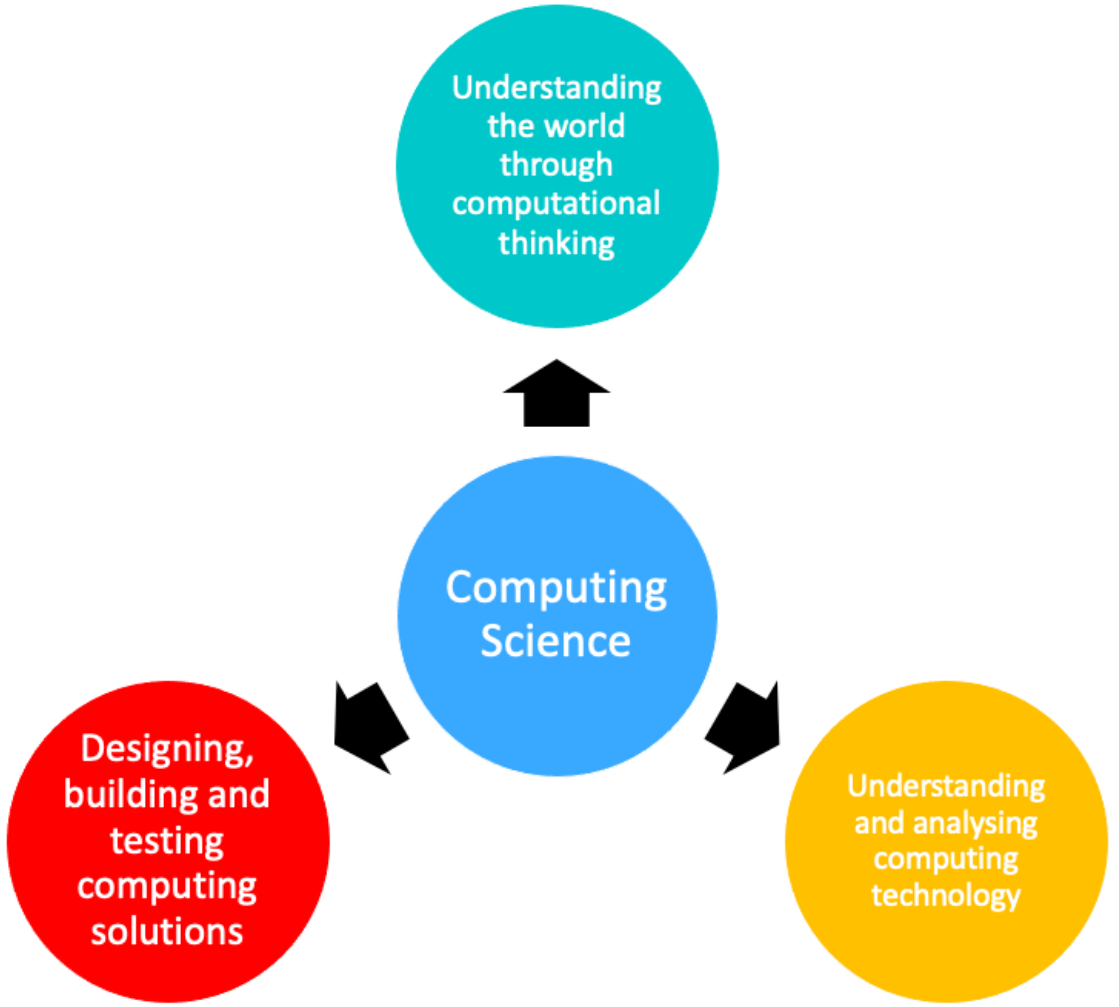
Fife Digital Learning and Literacy Progression – Computing Science

COPYRIGHT AND DISCLAIMER

CONTENTS



Digital Learning and Literacy in Fife

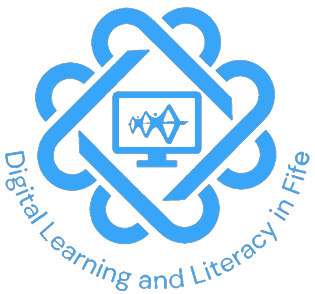
ACKNOWLEDGEMENT

This programme is extensively based on the guidance and recommendations of the Curriculum for Excellence and Benchmarks set out by Education Scotland.

These progressions allow for a fluidity of approaches that facilitate the opportunity for learners to think creatively and independently, whilst at the same time supporting practitioners to plan in a structured and coherent way. Using these shared standards and expectations within Schools, Clusters and Local Improvement Forums across Fife will ensure that all learners have an equitable experience to develop skills for their learning, life, and work.

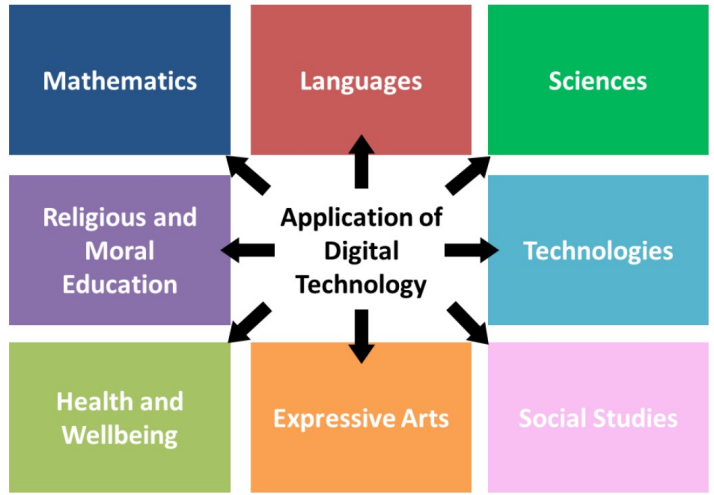
The Fife Digital Learning and Literacy Progression has been developed from the PICT (Progression in Information Communication Technology) in collaboration with the Fife Digital Learning Team and BTS Education staff. This programme supersedes the previous PICT v3 content to reflect the developments since its launch, both in terms of technology and their applications. Consideration has been given to sustainable resources that are also GDPR compliant, with scope for future changes.

We would like to acknowledge the efforts of Fife practitioners in supporting the development of the Fife Computing Science Progression through engaging with consultations and providing feedback on the programme. This has been crucial in ensuring that the programme is cohesive, progressive and practical.



GUIDANCE

The Technologies Curriculum has been split into five curricular organisers, two of which are related to digital learning. These are Digital Literacy (including Cyber Security and Internet Safety) and Computing Science.



This progression pathway is intended to provide a framework for practitioners as they plan and deliver Computing Science to ensure all learners have the skills and experiences required to use digital tools safely to support their learning across the curriculum.

Within the Progression Pathways, developmental stages of learning are clearly outlined. These amalgamate both Experiences and Outcomes with the national Benchmarks. These are not prescribed in a hierarchical way but rather enable practitioners to be responsive and flexible in their planning of learning pathways as appropriate to the needs of their learners. Though knowledge and skills at the base are often required for further learning to be built upon, these are not aligned to any particular year group nor always the starting point for all. Learners will progress through their learning pathways within each curricular organiser as appropriate to their developmental needs.

The national Benchmarks, which support practitioners' professional judgement of achievement of a level, are embedded within each of the Progression Pathways. These are emboldened for ease of identification.

Effective use of the documentation is best supported by engaging with the whole school culture to identify how it can be used consistently for planning and assessment purposes. Ongoing professional learning will be a core element for all practitioners to meet the ever-evolving requirements.

In order to ensure the Pathways reflect any future changes, we recommend visiting the documentation site regularly to check for updates or changes in resources. You can click the QR code on this page or scan it to take you to this site.

IMPLEMENTATION

“Computer science encompasses the theory, design, development and use of computer systems. It is a broad field which includes, but is certainly not limited to, the development of computer components, the development of computer systems and networks, and programming. Computer science makes close links with subjects such as maths, logic and science.”

Read more about Computing Science from Barefoot here and explore their resources to support the teaching of computing science.

To support the implementation of the progression pathway, use the Fife Computing Science Progression site for exemplification of the pathway and resources for whole setting/school planning.

See the links below for further support.

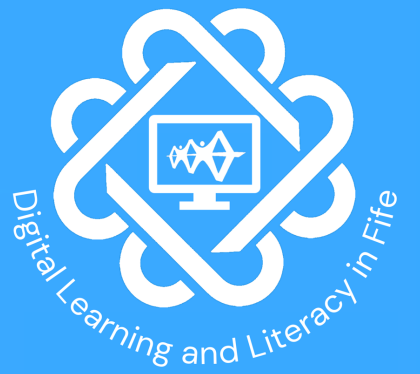
Fife Digital Learning
Visit the Fife Digital Learning site for further support on teaching computing science. This site also contains help guides and resources around the hardware and software supported in Fife.

Education Scotland Resources
Education Scotland have produced a series of documents and resources to support computing science and they can be used alongside the Fife Computing Science Progression. Click the links below to access the documents.
What Digital Learning Might Look Like – Early to Second Level, this exemplar has been developed to support practitioners when they are planning learning and teaching of the digital literacy and computing science experiences and outcomes.
Computing Science from DigiLearnScot – dedicated site from Education Scotland about computing science. Also has links to professional learning on offer from the Education Scotland Digital Learning Team.

Computing At School Scotland Guides
Computing At School Scotland have published guides to support practitioners with subject knowledge for computing science. Click the links below to access the documents.
Teach Computing Science – a guide for Early Years and Primary practitioners
Quick Start Computing Scotland – Subject Knowledge for Primary Teachers
Quick Start Computing Scotland – Subject Knowledge covering the transition from Primary to Secondary
Teach Computing Science – a guide for Secondary practitioners

Digital Learning and Literacy in Fife

Computing Science Overview
Early to Fourth Level



Understanding the world through computational thinking

Experiences and Outcomes	Benchmarks
I can explore computational thinking processes involved in a variety of everyday tasks and can identify patterns in objects or information. TCH 0-13a	• Identifies and sequences the main steps in an everyday task to create instructions/an algorithm for example, washing hands. • Classifies objects and groups them into simple categories for examples, groups toy bricks according to colour. • Identifies patterns, similarities and differences in objects or information such as colour, size and temperature and simple relationships between them.
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).
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.
I can describe different fundamental information processes and how they communicate and can identify their use in solving different problems. TCH 3-13a I am developing my understanding of information and can use an information model to describe particular aspects of a real world system. TCH 3-13b	• Recognises and describes information systems with communicating processes which occur in the world around me. • Explains the difference between parallel processes and those that communicate with each other. • Demonstrates an understanding of the basic principles of compression and encryption of information. • Identifies a set of characteristics describing a collection of related items that enable each item to be individually identified. • Identifies the use of common algorithms such as sorting and searching as part of larger processes.
I can describe in detail the processes used in real world solutions, compare these processes against alternative solutions and justify which is the most appropriate. TCH 4-13a I can informally compare algorithms for correctness and efficiency. TCH 4-13b	• Identifies the transfer of information through complex systems involving both computers and physical artefacts, for example, airline check-in, parcel tracking and delivery. • Describes instances of human decision making as an information process, for example, deciding which check-out queue to pick, which route to take to school, how to prepare family dinner / a school event. • Compares alternative algorithms for the same problem and understands that there are different ways of defining “better” solutions depending on the problem context for example, is speed or space more valuable in this context?

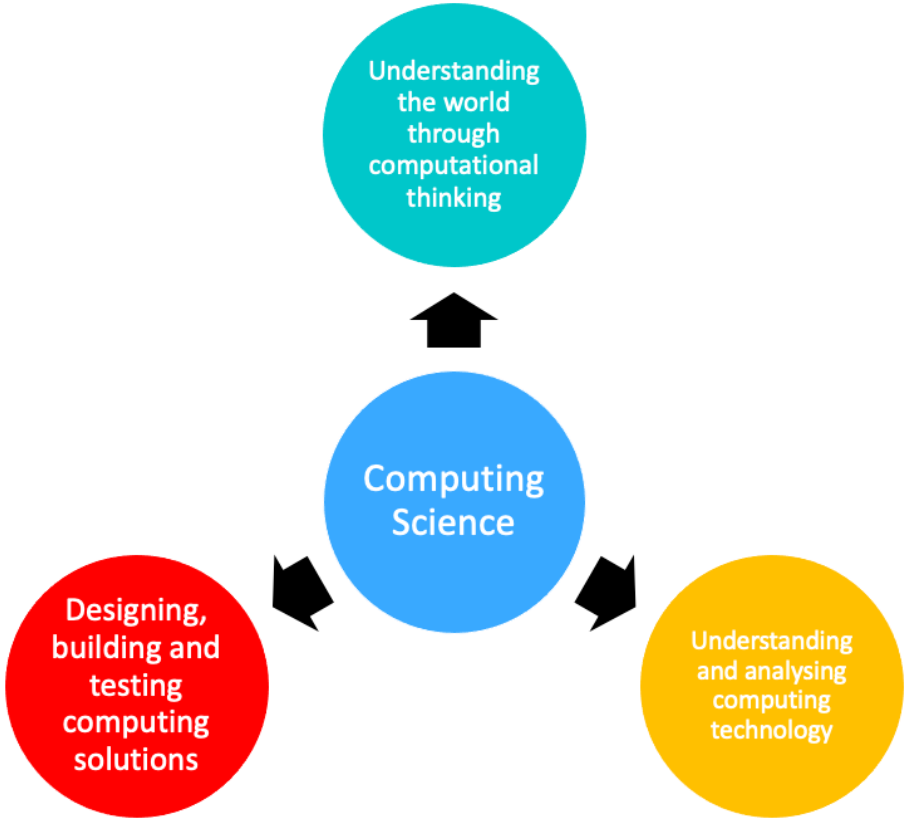
Understanding and analysing computer technology

Experiences and Outcomes	Benchmarks
I understand that sequences of instructions are used to control computing technology. TCH 0-14a I can experiment with and identify uses of a range of computing technology in the world around me. TCH 0-14b	• Demonstrates an understanding of how symbols can represent process and information. • Predicts what a device or person will do when presented with a sequence of instructions for example, arrows drawn on paper. • Identifies computing devices in the world (including those hidden in appliances and objects such as automatic doors).
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.
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.
I understand language constructs for representing structured Information. TCH 3-14a I can describe the structure and operation of computing systems which have multiple software and hardware levels that interact with each other. TCH 3-14b	• Understands that the same information could be represented in more than one representational system. • Understands that different information could be represented in exactly the same representation. • Demonstrates an understanding of structured information in programs, databases or webpages. • Describes the effect of mark-up language on the appearance of a webpage, and understand that this may be different on different devices. • Demonstrates an understanding of the von Neumann architecture and how machine code instructions are stored and executed within a computer system. • Reads and explains code extracts including those with variables and data structures. • Demonstrate an understanding of how computers communicate and share information over networks including the concepts of sender, receiver, address and packets. • Understands simple compression and encryption techniques used in computing technology.
I understand constructs and data structures in a textual programming language. TCH 4-14a I can explain the overall operation and architecture of a digitally created solution. TCH 4-14b I understand the relationship between high level language and the operation of computer. TCH 4-14c	• Understands basic control constructs such as sequence, selection repetition, variables and numerical calculations in a textual language. • Demonstrates an understanding of how visual instructions and textual instructions for the same construct are related. • Identifies and explains syntax errors in a program written in a textual language. • Demonstrates an understanding of representations of data structures in a textual language. • Demonstrates an understanding of how computers represent and manipulate information in a range of formats. • Demonstrates an understanding of program plans expressed in accepted design representations for example pseudocode, storyboarding, structure diagram, data flow diagram, flow chart. • Demonstrates an understanding of the underling technical concepts of some specific facets of modern complex technologies for example, online payment systems and satnav. • Demonstrates an understanding that computers translate information processes between different levels of abstraction.

Designing, building and testing computing solutions

Experiences and Outcomes	Benchmarks
I can develop a sequence of instructions and run them using programmable devices or equivalent. TCH 0-15a	• Designs a simple sequence of instructions/algorithm for programmable device to carry out a task for example, directional instructions: forwards/backwards. • Identifies and corrects errors in a set of instructions.
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.
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.
I can select appropriate development tools to design, build, evaluate and refine computing solutions based on requirements. TCH 3-15a	• Designs and builds a program using a visual language combining constructs and using multiple variables. • Represents and manipulates structured information in programs, or databases for example, works with a list data structure in a visual language, or a flat file database. • Interprets a problem statement, and identifies processes and information to create a physical computing and/or software solution. • Can find and correct errors in program logic. • Groups related instructions into named subprograms (in a visual language). • Writes code in which there is communication between parallel processes (in a visual language). • Writes code which receives and responds to real world inputs (in a visual language). • Designs and builds web pages using appropriate mark-up languages.
I can select appropriate development tools to design, build, evaluate and refine computing solutions to process and present information whilst making reasoned arguments to justify my decisions. TCH 4-15a	• Analyses problem specifications across a range of contexts, identifying key requirements. • Writes a program in a textual language which uses variables and constructs such as sequence, selection and repetition. • Creates a design using accepted design notations for example, pseudocode storyboarding, structure diagram, data flow diagram, flow chart. • Develops a relational database to represent structured information. • Debugs code and can distinguish between the nature of identified errors e.g. syntax and logic. • Writes test and evaluation reports. • Can make use of logical operators – AND, OR, NOT. • Writes a program in a textual language which uses variables within instructions instead of specific values where appropriate. • Designs appropriate data structures to represent information in a textual language. • Selects an appropriate platform on which to develop a physical and/or software solution from a requirements specification. • Compares common algorithms for example, those for sorting and searching, and justify which would be most appropriate for a given problem. • Design and build web pages which includes interactivity.

Computing Science Early Level



Early Level Computing Science

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding the world through computational thinking	I can explore computational thinking processes involved in a variety of everyday tasks and can identify patterns in objects or information. TCH 0-13a	• Identifies and sequences the main steps in an everyday task to create instructions/an algorithm for example, washing hands. • Classifies objects and groups them into simple categories for examples, groups toy bricks according to colour. • Identifies patterns, similarities and differences in objects or information such as colour, size and temperature and simple relationships between them.
Understanding and analysing computer technology	I understand that sequences of instructions are used to control computing technology. TCH 0-14a I can experiment with and identify uses of a range of computing technology in the world around me. TCH 0-14b	• Demonstrates an understanding of how symbols can represent process and information. • Predicts what a device or person will do when presented with a sequence of instructions for example, arrows drawn on paper. • Identifies computing devices in the world (including those hidden in appliances and objects such as automatic doors).
Designing, building and testing computing solutions	I can develop a sequence of instructions and run them using programmable devices or equivalent. TCH 0-15a	• Designs a simple sequence of instructions/algorithm for programmable device to carry out a task for example, directional instructions: forwards/backwards. • Identifies and corrects errors in a set of instructions.

Early Level Computational Thinking

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding the world through computational thinking	I can explore computational thinking processes involved in a variety of everyday task and can identify patterns in objects or information. TCH 0-13a	• Identifies and sequences the main steps in an everyday task to create instructions/an algorithm for example, washing hands. • Classifies objects and groups them into simple categories for examples, groups toy bricks according to colour. • Identifies patterns, similarities and differences in objects or information such as colour, size and temperature and simple relationships between them.

What the learning may look like in Fife	Glossary of terms
• Reorganise a list of steps in a logical order to complete a task. • Sort objects such as beads or Lego together into groups of similar colour and size. • 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.	• **Algorithm** - A list of instructions that describe how to do a particular task • **Computational Thinking** - Looking at a problem in a way that a computer does to help us to solve it • **Direction** - A course along which someone or something move • **Instructions** - A single operation of a processor defined by the processor instruction set • **Sorting** – Grouping by class or kind or size For the full glossary at Early Level, click here.

Early Level Computational Thinking

Creates and describes a pattern to sequence a given series of objects or information.

Identifies patterns, similarities and differences in objects or information such as colour, size and temperature and simple relationships between them.

Sequences a series of objects or information to match a given pattern e.g. by size, colour, etc.

Justifies how objects have been sorted and explains the categories.

Classifies objects and groups them into simple categories for examples, groups toy bricks according to colour.

Describes how objects can be sorted e.g. by colour, size, shape and offers other categories of their own.

Identifies and sequences the main steps in an everyday task to create instructions/an algorithm for example, washing hands.

Identifies what happens first, next and last when observing the order and sequence of everyday processes.

Observes and explores the order of everyday processes in the world around me e.g. getting dressed, making a sandwich.

Early Level Analysing Computing Technology

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding and analysing computer technology	I understand that sequences of instructions are used to control computing technology. TCH 0-14a I can experiment with and identify uses of a range of computing technology in the world around me. TCH 0-14b	• Demonstrates an understanding of how symbols can represent process and information. • Predicts what a device or person will do when presented with a sequence of instructions for example, arrows drawn on paper. • Identifies computing devices in the world (including those hidden in appliances and objects such as automatic doors).

What the learning may look like in Fife	Glossary of terms
• Create a list of common symbols throughout the school e.g. Fire Exits, Boardmaker symbols, toilets, etc. • Play with programmable devices such as Beebots, etc. • Experience giving and following instructions, e.g. using songs, PE activities, daily routines. • Create a tinker area within the classroom that learners can explore technology through free play. • Use resources from Barefoot Computing to support exploration of Computing Science. Visit the Computing Science Progression site for further ideas and resources.	• **Instructions** - A single operation of a processor defined by the processor instruction set • **Internet of Things** - A network of internet-connected devices such as fridge freezers/smartphones/medical devices all able to collect and exchange data using embedded sensor • **Process** - An instance of a computer program that is being run For the full glossary at Early Level, click here.

Early Level Analysing Computing Technology

Predicts what a device or person will do when presented with a sequence of instructions for example, arrows drawn on paper.

Creates instructions and follows a sequence of using arrows.

Explores how symbols are used to control technology e.g. on a phone, programmable device, computer, etc.

Discusses why symbols are important e.g. in school and the world around me.

Demonstrates an understanding of how symbols can represent process and information.

Identifies symbols and signs in the school environment e.g. the toilet, coat pegs, classrooms.

Identifies computing devices in the world (including those hidden in appliances and objects such as automatic doors).

Early Level Designing, Building and Testing

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Designing, building and testing computing solutions	I can develop a sequence of instructions and run them using programmable devices or equivalent. TCH 0-15a	• Designs a simple sequence of instructions/algorithm for programmable device to carry out a task for example, directional instructions: forwards/backwards. • Identifies and corrects errors in a set of instructions.

What the learning may look like in Fife	Glossary of terms
• Play with programmable devices such as Beebots, etc. • Use cards/printed symbols/written symbols to design an algorithm. • Environmental displays to share vocabulary. • 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.	• **Algorithm** - A list of instructions that describe how to do a particular task • **Direction** - A course along which someone or something move • **Instructions** - A single operation of a processor defined by the processor instruction set For the full glossary at Early Level, click here.

Early Level Designing, Building and Testing

Designs a simple sequence of instructions/algorithm for programmable device to carry out a task for example, directional instructions: forwards/backwards.

Designs a simple algorithm describing a regular route e.g. how to navigate around the school/playground

Uses vocabulary such as forward, backward, turn, left and right in play and when giving instructions.

Demonstrates resilience when experiencing challenge during a testing phase.

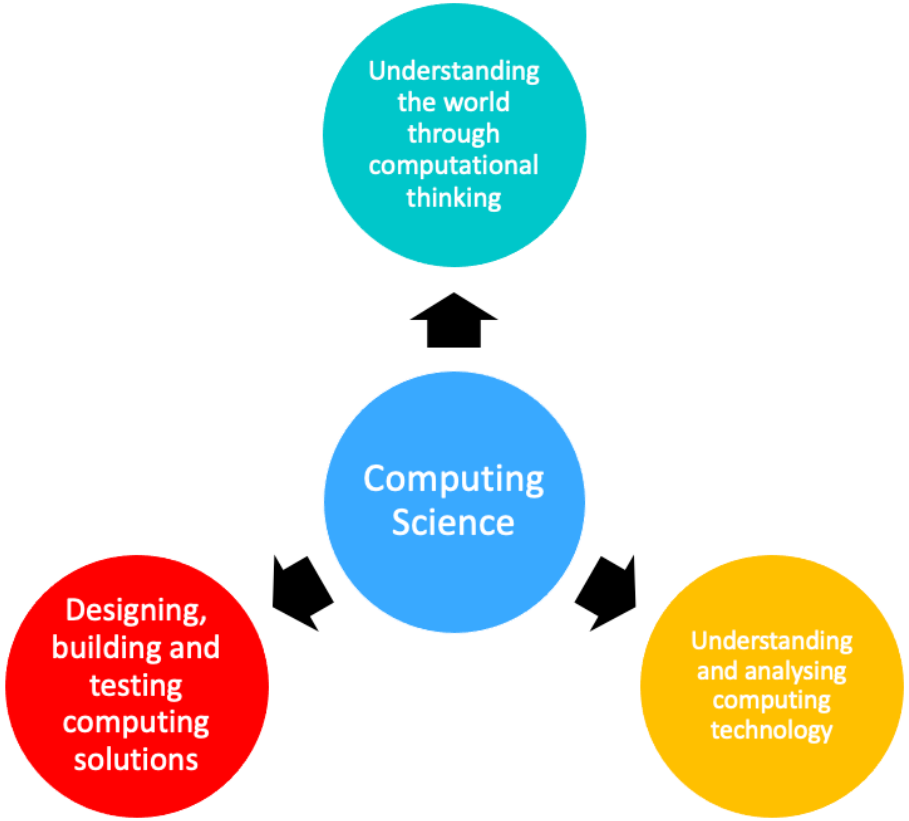
Identifies and corrects errors in a set of instructions.

Explains the importance of testing an algorithm

Recognises that directional instructions are affected by the orientation of the device/person carrying them out.

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

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.

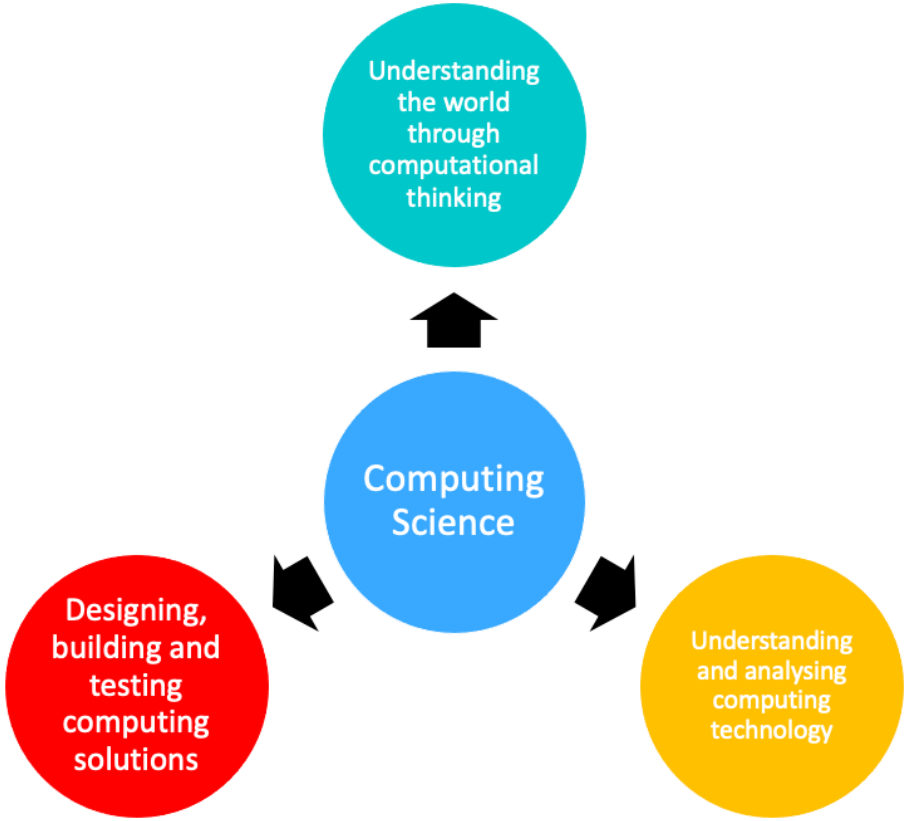
Uses vocabulary such as decomposition and debugging to explain how problems and programs are solved and edited.

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.

Computing Science Second Level



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.

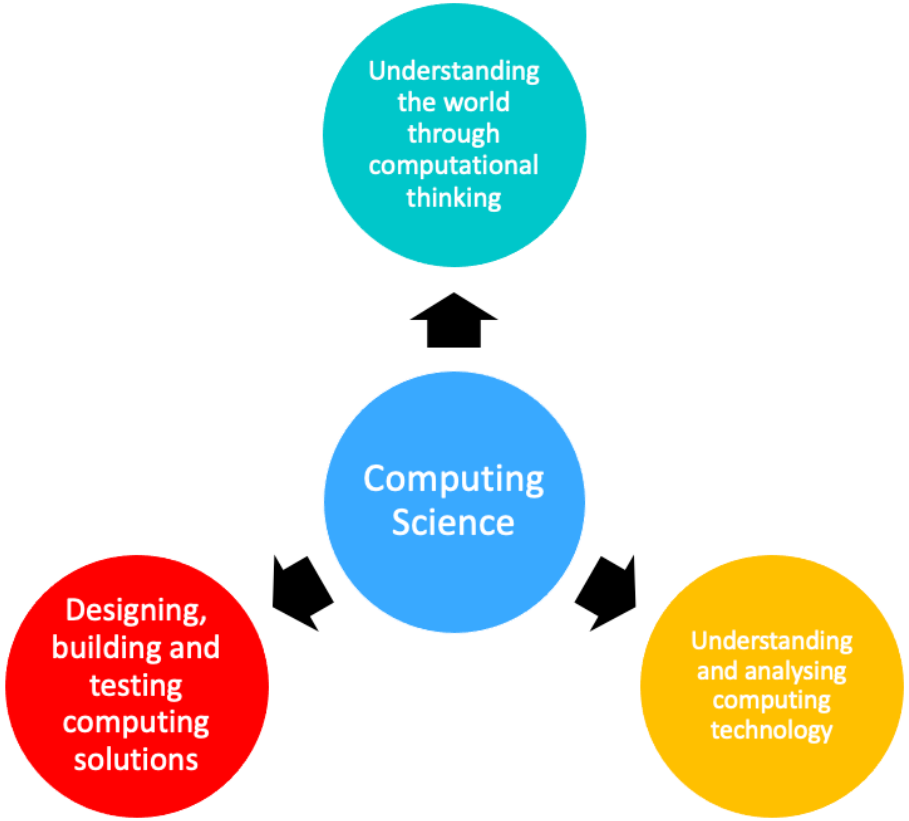
Identifies patterns in problem solving and reuses aspects of previous solutions appropriately for example, reuse code for a timer, score counter or controlling arrow keys.

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.

Computing Science Third Level



Third Level Computing Science

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding the world through computational thinking	I can describe different fundamental information processes and how they communicate and can identify their use in solving different problems. TCH 3-13a I am developing my understanding of information and can use an information model to describe particular aspects of a real world system. TCH 3-13b	• Recognises and describes information systems with communicating processes which occur in the world around me. • Explains the difference between parallel processes and those that communicate with each other. • Demonstrates an understanding of the basic principles of compression and encryption of information. • Identifies a set of characteristics describing a collection of related items that enable each item to be individually identified. • Identifies the use of common algorithms such as sorting and searching as part of larger processes.
Understanding and analysing computer technology	I understand language constructs for representing structured Information. TCH 3-14a I can describe the structure and operation of computing systems which have multiple software and hardware levels that interact with each other. TCH 3-14b	• Understands that the same information could be represented in more than one representational system. • Understands that different information could be represented in exactly the same representation. • Demonstrates an understanding of structured information in programs, databases or webpages. • Describes the effect of mark-up language on the appearance of a webpage, and understand that this may be different on different devices. • Demonstrates an understanding of the von Neumann architecture and how machine code instructions are stored and executed within a computer system. • Reads and explains code extracts including those with variables and data structures. • Demonstrate an understanding of how computers communicate and share information over networks including the concepts of sender, receiver, address and packets. • Understands simple compression and encryption techniques used in computing technology.
Designing, building and testing computing solutions	I can select appropriate development tools to design, build, evaluate and refine computing solutions based on requirements. TCH 3-15a	• Designs and builds a program using a visual language combining constructs and using multiple variables. • Represents and manipulates structured information in programs, or databases for example, works with a list data structure in a visual language, or a flat file database. • Interprets a problem statement, and identifies processes and information to create a physical computing and/or software solution. • Can find and correct errors in program logic. • Groups related instructions into named subprograms (in a visual language). • Writes code in which there is communication between parallel processes (in a visual language). • Writes code which receives and responds to real world inputs (in a visual language). • Designs and builds web pages using appropriate mark-up languages.

Third Level Computational Thinking

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding the world through computational thinking	I can describe different fundamental information processes and how they communicate and can identify their use in solving different problems. TCH 3-13a I am developing my understanding of information and can use an information model to describe particular aspects of a real world system. TCH 3-13b	• Recognises and describes information systems with communicating processes which occur in the world around me. • Explains the difference between parallel processes and those that communicate with each other. • Demonstrates an understanding of the basic principles of compression and encryption of information. • Identifies a set of characteristics describing a collection of related items that enable each item to be individually identified. • Identifies the use of common algorithms such as sorting and searching as part of larger processes.

What the learning may look like in Fife	Glossary of terms
Visit the Computing Science Progression site for further ideas and resources.	For the full glossary at Early Level, click here.

Third Level Computational Thinking

Tiles to support

Identifies the use of common algorithms such as sorting and searching as part of larger processes.

Identifies a set of characteristics describing a collection of related items that enable each item to be individually identified.

Demonstrates an understanding of the basic principles of compression and encryption of information.

Explains the difference between parallel processes and those that communicate with each other.

Recognises and describes information systems with communicating processes which occur in the world around me.

Third Level Analysing Computing Technology

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding and analysing computer technology	I understand language constructs for representing structured Information. TCH 3-14a I can describe the structure and operation of computing systems which have multiple software and hardware levels that interact with each other. TCH 3-14b	• Understands that the same information could be represented in more than one representational system. • Understands that different information could be represented in exactly the same representation. • Demonstrates an understanding of structured information in programs, databases or webpages. • Describes the effect of mark-up language on the appearance of a webpage, and understand that this may be different on different devices. • Demonstrates an understanding of the von Neumann architecture and how machine code instructions are stored and executed within a computer system. • Reads and explains code extracts including those with variables and data structures. • Demonstrate an understanding of how computers communicate and share information over networks including the concepts of sender, receiver, address and packets. • Understands simple compression and encryption techniques used in computing technology.

What the learning may look like in Fife	Glossary of terms
Visit the Computing Science Progression site for further ideas and resources.	For the full glossary at Early Level, click here.

Third Level Analysing Computing Technology

Tiles to support

Reads and explains code extracts including those with variables and data structures.

Demonstrates an understanding of the von Neumann architecture and how machine code instructions are stored and executed within a computer system.

Describes the effect of mark-up language on the appearance of a webpage, and understand that this may be different on different devices.

Demonstrates an understanding of structured information in programs, databases or webpages.

Understands that different information could be represented in exactly the same representation.

Understands that the same information could be represented in more than one representational system.

Understands simple compression and encryption techniques used in computing technology.

Demonstrate an understanding of how computers communicate and share information over networks including the concepts of sender, receiver, address and packets.

Third Level Designing, Building and Testing

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Designing, building and testing computing solutions	I can select appropriate development tools to design, build, evaluate and refine computing solutions based on requirements. TCH 3-15a	• Designs and builds a program using a visual language combining constructs and using multiple variables. • Represents and manipulates structured information in programs, or databases for example, works with a list data structure in a visual language, or a flat file database. • Interprets a problem statement, and identifies processes and information to create a physical computing and/or software solution. • Can find and correct errors in program logic. • Groups related instructions into named subprograms (in a visual language). • Writes code in which there is communication between parallel processes (in a visual language). • Writes code which receives and responds to real world inputs (in a visual language). • Designs and builds web pages using appropriate mark-up languages.

What the learning may look like in Fife	Glossary of terms
Visit the Computing Science Progression site for further ideas and resources.	For the full glossary at Early Level, click here.

Third Level Designing, Building and Testing

Tiles to support

Writes code in which there is communication between parallel processes (in a visual language).

Groups related instructions into named subprograms (in a visual language).

Can find and correct errors in program logic.

Interprets a problem statement, and identifies processes and information to create a physical computing and/or software solution.

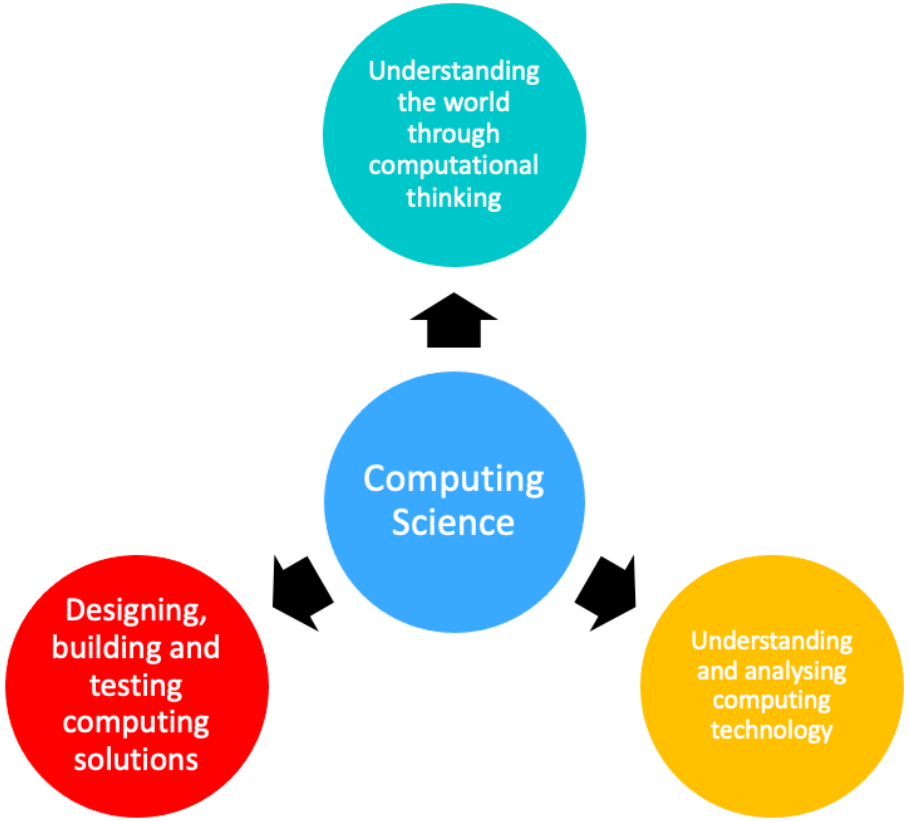
Represents and manipulates structured information in programs, or databases for example, works with a list data structure in a visual language, or a flat file database.

Designs and builds a program using a visual language combining constructs and using multiple variables.

Writes code which receives and responds to real world inputs (in a visual language).

Designs and builds web pages using appropriate mark-up languages.

Computing Science Fourth Level



Fourth Level Computing Science

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding the world through computational thinking	I can describe in detail the processes used in real world solutions, compare these processes against alternative solutions and justify which is the most appropriate. TCH 4-13a I can informally compare algorithms for correctness and efficiency. TCH 4-13b	• Identifies the transfer of information through complex systems involving both computers and physical artefacts, for example, airline check-in, parcel tracking and delivery. • Describes instances of human decision making as an information process, for example, deciding which check-out queue to pick, which route to take to school, how to prepare family dinner / a school event. • Compares alternative algorithms for the same problem and understands that there are different ways of defining “better” solutions depending on the problem context for example, is speed or space more valuable in this context?
Understanding and analysing computer technology	I understand constructs and data structures in a textual programming language. TCH 4-14a I can explain the overall operation and architecture of a digitally created solution. TCH 4-14b I understand the relationship between high level language and the operation of computer. TCH 4-14c	• Understands basic control constructs such as sequence, selection repetition, variables and numerical calculations in a textual language. • Demonstrates an understanding of how visual instructions and textual instructions for the same construct are related. • Identifies and explains syntax errors in a program written in a textual language. • Demonstrates an understanding of representations of data structures in a textual language. • Demonstrates an understanding of how computers represent and manipulate information in a range of formats. • Demonstrates an understanding of program plans expressed in accepted design representations for example pseudocode, storyboarding, structure diagram, data flow diagram, flow chart. • Demonstrates an understanding of the underling technical concepts of some specific facets of modern complex technologies for example, online payment systems and satnav. • Demonstrates an understanding that computers translate information processes between different levels of abstraction.
Designing, building and testing computing solutions	I can select appropriate development tools to design, build, evaluate and refine computing solutions to process and present information whilst making reasoned arguments to justify my decisions. TCH 4-15a	• Analyses problem specifications across a range of contexts, identifying key requirements. • Writes a program in a textual language which uses variables and constructs such as sequence, selection and repetition. • Creates a design using accepted design notations for example, pseudocode storyboarding, structure diagram, data flow diagram, flow chart. • Develops a relational database to represent structured information. • Debugs code and can distinguish between the nature of identified errors e.g. syntax and logic. • Writes test and evaluation reports. • Can make use of logical operators – AND, OR, NOT. • Writes a program in a textual language which uses variables within instructions instead of specific values where appropriate. • Designs appropriate data structures to represent information in a textual language. • Selects an appropriate platform on which to develop a physical and/or software solution from a requirements specification. • Compares common algorithms for example, those for sorting and searching, and justify which would be most appropriate for a given problem. • Design and build web pages which includes interactivity.

Fourth Level Computational Thinking

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding the world through computational thinking	I can describe in detail the processes used in real world solutions, compare these processes against alternative solutions and justify which is the most appropriate. TCH 4-13a I can informally compare algorithms for correctness and efficiency. TCH 4-13b	• Identifies the transfer of information through complex systems involving both computers and physical artefacts, for example, airline check-in, parcel tracking and delivery. • Describes instances of human decision making as an information process, for example, deciding which check-out queue to pick, which route to take to school, how to prepare family dinner / a school event. • Compares alternative algorithms for the same problem and understands that there are different ways of defining "better" solutions depending on the problem context for example, is speed or space more valuable in this context?

What the learning may look like in Fife	Glossary of terms
Visit the Computing Science Progression site for further ideas and resources.	For the full glossary at Early Level, click here.

Fourth Level Computational Thinking

Tiles to support

Compares alternative algorithms for the same problem and understands that there are different ways of defining "better" solutions depending on the problem context for example, is speed or space more valuable in this context?

Describes instances of human decision making as an information process, for example, deciding which check-out queue to pick, which route to take to school, how to prepare family dinner / a school event.

Identifies the transfer of information through complex systems involving both computers and physical artefacts, for example, airline check-in, parcel tracking and delivery.

Fourth Level Analysing Computing Technology

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Understanding and analysing computer technology	I understand constructs and data structures in a textual programming language. TCH 4-14a I can explain the overall operation and architecture of a digitally created solution. TCH 4-14b I understand the relationship between high level language and the operation of computer. TCH 4-14c	• Understands basic control constructs such as sequence, selection repetition, variables and numerical calculations in a textual language. • Demonstrates an understanding of how visual instructions and textual instructions for the same construct are related. • Identifies and explains syntax errors in a program written in a textual language. • Demonstrates an understanding of representations of data structures in a textual language. • Demonstrates an understanding of how computers represent and manipulate information in a range of formats. • Demonstrates an understanding of program plans expressed in accepted design representations for example pseudocode, storyboarding, structure diagram, data flow diagram, flow chart. • Demonstrates an understanding of the underling technical concepts of some specific facets of modern complex technologies for example, online payment systems and satnav. • Demonstrates an understanding that computers translate information processes between different levels of abstraction.

What the learning may look like in Fife	Glossary of terms
Visit the Computing Science Progression site for further ideas and resources.	For the full glossary at Early Level, click here.

Fourth Level Analysing Computing Technology

Tiles to support

Demonstrates an understanding that computers translate information processes between different levels of abstraction.

Demonstrates an understanding of the underling technical concepts of some specific facets of modern complex technologies for example, online payment systems and satnav.

Demonstrates an understanding of representations of data structures in a textual language.

Demonstrates an understanding of program plans expressed in accepted design representations for example pseudocode, storyboarding, structure diagram, data flow diagram, flow chart.

Identifies and explains syntax errors in a program written in a textual language.

Demonstrates an understanding of how visual instructions and textual instructions for the same construct are related.

Understands basic control constructs such as sequence, selection repetition, variables and numerical calculations in a textual language.

Fourth Level Designing, Building and Testing

Curriculum Organiser	Experiences and Outcomes	Benchmarks
Designing, building and testing computing solutions	I can select appropriate development tools to design, build, evaluate and refine computing solutions to process and present information whilst making reasoned arguments to justify my decisions. TCH 4-15a	• Analyses problem specifications across a range of contexts, identifying key requirements. • Writes a program in a textual language which uses variables and constructs such as sequence, selection and repetition. • Creates a design using accepted design notations for example, pseudocode storyboarding, structure diagram, data flow diagram, flow chart. • Develops a relational database to represent structured information. • Debugs code and can distinguish between the nature of identified errors e.g. syntax and logic. • Writes test and evaluation reports. • Can make use of logical operators – AND, OR, NOT. • Writes a program in a textual language which uses variables within instructions instead of specific values where appropriate. • Designs appropriate data structures to represent information in a textual language. • Selects an appropriate platform on which to develop a physical and/or software solution from a requirements specification. • Compares common algorithms for example, those for sorting and searching, and justify which would be most appropriate for a given problem. • Design and build web pages which includes interactivity.

What the learning may look like in Fife	Glossary of terms
Visit the Computing Science Progression site for further ideas and resources.	For the full glossary at Early Level, click here.

Fourth Level Designing, Building and Testing

Tiles to support

Writes test and evaluation reports.

Debugs code and can distinguish between the nature of identified errors e.g. syntax and logic.

Develops a relational database to represent structured information.

Creates a design using accepted design notations for example, pseudocode storyboarding, structure diagram, data flow diagram, flow chart.

Writes a program in a textual language which uses variables and constructs such as sequence, selection and repetition.

Analyses problem specifications across a range of contexts, identifying key requirements.

Compares common algorithms for example, those for sorting and searching, and justify which would be most appropriate for a given problem.

Selects an appropriate platform on which to develop a physical and/or software solution from a requirements specification.

Designs appropriate data structures to represent information in a textual language.

Writes a program in a textual language which uses variables within instructions instead of specific values where appropriate.

Can make use of logical operators – AND, OR, NOT.

Design and build web pages which includes interactivity.