



	EYFS	Year 1	Year 2	Year 3
Programming	- To know that being able to follow and give simple instructions is important in computing. - To understand that it is important for instructions to be in the right order. - To understand why a set of instructions may have gone wrong. - To know that you can program a Bee-Bot with some simple commands. - To understand that debugging means how to fix some simple programming errors. - To understand that an algorithm is a set of clear and precise instructions.	- To understand that an algorithm is when instructions are put in an exact order. - To know that input devices get information into a computer and that output devices get information out of a computer. - To understand that decomposition means breaking a problem into manageable chunks and that it is important in computing. - To know that we call errors in an algorithm 'bugs' and fixing these 'debugging'. - To understand the basic functions of a Bee-Bot. - To know that you can use a camera/tablet to make simple videos. - To know that algorithms move a bee-bot accurately to a chosen destination. - To know that humans need to give robots instructions to follow and that they will carry out these instructions exactly, even if they are wrong. - To know that humans need to give instructions in the correct language for the robot to understand. - To know that instructions (algorithms) must give every step of a task. - To know that an algorithm must give clear, sequenced instructions. - To know that there may be an error if a set of instructions (an algorithm) does not give the expected result. - To know that errors could result from sequencing, unclear instructions or missing steps.	- To understand what machine learning is and how that enables computers to make predictions. - To know that loops in programming are where you set a certain instruction (or instructions) to be repeated multiple times. - To know that abstraction is the removing of unnecessary detail to help solve a problem. - To know that coding is writing in a special language so that the computer understands what to do - To understand that the character in ScratchJr is controlled by the programming blocks. - To know that you can write a program to create a musical instrument or tell a joke. - To know that programming a computer or device involves giving it instructions to perform specific tasks - That video games, phones, websites and apps are all created using programming. - To know that different devices and programs use different programming languages or 'codes' - To know that an algorithm becomes a program when it is coded. - That programs execute the exact instructions they are given, even if they are incorrect. - That a program is a series of instructions (algorithms) that are written for a computer to follow. - That a person can program a device by giving it an algorithm/algorithms to follow.	- To know that Scratch is a programming language and some of its basic functions. - To understand how to use loops to improve programming. - To understand how decomposition is used in programming. - To understand that you can remix and adapt existing code. - To know that websites have all been programmed, often using HTML. - To know that programming languages can be visual (Scratch) or text-based (HTML). - To know that 'coding' is the process of turning an algorithm into a programming language. - To know that HTML tells a browser how to display text, images and multimedia on a webpage. - To know that 'decomposition' is the process of breaking down a task or problem into smaller parts. - To know that breaking down a problem into smaller parts makes it easier to solve the problem. - To know that 'abstraction' is identifying the important detail and ignoring irrelevant information. - To know that loops are used to save time when writing code by reducing repetition. - To know that a variable is a container or holder for storing information that can change, e.g. numbers or text. - To know that conditional statements tell the computer what to do next based on a user's input. - To know that it is important to identify where the mistake is in

			• That there must be an error if a program does not execute as expected. • That an error in a computer program is known as a 'bug' and fixing errors is known as 'debugging'. •	the programming as part of the debugging process. • To know that errors in a program could result from sequencing errors, coding errors or missing code. •
Data Handling		• To know that computers understand different types of 'input'.	• To understand what steps you need to take to create an algorithm.	• To know that different visual representations of data can be made on a computer.
Computer Systems and Networks			• To know that computers often work together.	• To know the components that make up a network (Wireless access point/WAP, Network switch, Router, Server and devices). • To know that a server is central to a network and responds to requests made. • To know that the internet connects all the networks around the world. • To know that a router connects us to the internet. • To know what a packet is and why it is important for website data transfer. • To know the roles that inputs and outputs play on computers. • To know what some of the different components inside a computer are e.g. CPU, RAM, hard drive, and how they work together.

Progression of Skills

	EYFS	Year 1	Year 2	Year 3
Programming	• Following instructions as part of practical activities and games. • Learning to give simple instructions. • Experimenting with programming a Bee-bot/Blue- bot and learning how to give simple commands.	• Making suggestions for how to fix errors in algorithms. • Beginning to identify errors in algorithms. • Using terms like 'start,' 'end' and 'next' to describe the steps in algorithms.	• Using logical thinking to explore software, predicting, testing and explaining what it does. • Using an algorithm to write a basic computer program. • Using loop blocks when programming to repeat an instruction more than once.	• Making reasonable suggestions for how to debug their own and others' code. • Continuing existing code. • Incorporating loops to make code more efficient. • Using logical thinking to explore more complex software;

	Learning to debug instructions, with the help of an adult, when things go wrong.	- Writing clear, sequenced algorithms for familiar tasks - Explaining what they are trying to achieve with their algorithms. - Recognising that robots are programmed by humans. - Learning to debug an algorithm in an unplugged scenario. - Using programming language to explain how a floor robot works. - Learning to debug instructions when things go wrong. - Programming a floor robot to follow a planned route. - Changing their instructions or algorithms into code that the robot understands.		predicting, testing and explaining what it does. - Recognising visual and text-based programming languages. - Recognising further examples of computers being programmed by humans to perform simple tasks. - Working towards a given goal that a program needs to accomplish. - Breaking down what they want to achieve into smaller, manageable parts. - Using logic, pattern recognition and decomposition to solve simple problems. - Tinkering with an existing text-based code to see how it affects a program (website). - Remixing code to alter and add to an existing program. - Recognising repeating patterns in a program or code. - Creating loops to make code more efficient in block-based programs. - Beginning to use variables in block-based programming languages to make programs more interactive. - Including a conditional statement in block-based programming languages. - Recognising the relationship between what is happening in a program and the written (block) code. - Working backwards, beginning to identify the code they think a program uses. - Running small chunks of code at a time to find the error or 'bug.'
Hardware	- Learning how to operate a camera to take photographs of meaningful creations or moments. - Learning how to explore and tinker with hardware to develop familiarity and introduce relevant vocabulary.	- Learning how to explore and tinker with hardware to find out how it works. - Recognising that some devices are input devices and others are output devices. - Learning where keys are located on the keyboard.	- Understanding what a computer is and that it's made up of different components. - Recognising that buttons cause effects and that technology follows instructions.	- Understanding what the different components of a computer do and how they work together. - Learning about the purpose of routers. - Drawing comparisons across different types of computers.

	- Recognising and identifying familiar letters and numbers on a keyboard. - Developing basic mouse skills such as moving and clicking.	Learning how to operate a camera to take photos and videos.	- Learning how we know that technology is doing what we want it to do via its output. - Using greater control when taking photos with cameras, tablets or computers. - Developing confidence with the keyboard and the basics of touch typing.	
Computational Thinking	Using logical reasoning to understand simple instructions and predict the outcome	- Learning that decomposition means breaking a problem down into smaller parts. - Using decomposition to solve unplugged challenges. - Using logical reasoning to predict the behaviour of simple programs. - Developing the skills associated with sequencing in unplugged activities. - Following a basic set of instructions. - Assembling instructions into a simple algorithm.	- Articulating what decomposition is - Decomposing a game to predict the algorithms used to create it. - Learning that there are different levels of abstraction. - Explaining what an algorithm is. - Following an algorithm. - Learning that programs execute by following precise instructions. - Incorporating loops within algorithms. - Creating a clear and precise algorithm.	- Forming algorithms independently. - Explaining the purpose of an algorithm. - Using logical reasoning to explain how simple algorithms work. - Using repetition in programs. - Using decomposition to explore the code behind an animation. - Using decomposition to explain the parts of a laptop computer.

Progression of Knowledge

	Year 4	Year 5	Year 6
Programming	- To understand that a variable is a value that can change (depending on conditions) and know that you can create them in Scratch. - To know what a conditional statement is in programming. - To understand that variables can help you to create a quiz on Scratch. - To know that combining computational thinking skills can help you to solve a problem. - To understand that pattern recognition means identifying patterns to help them work out how the code works. - To understand that algorithms can be used for a number of purposes e.g. animation, games design etc - To know that websites have all been programmed, often using HTML. - To know that programming languages can be visual (Scratch) or text-based (HTML).	- To know that a soundtrack is music for a film/video and that one way of composing these is on programming software. - To understand that using loops can make the process of writing music simpler and more effective. - To know how to adapt their music while performing. - To understand and recognise coding structures including variables. - To know what techniques to use to create a program for a specific purpose (including decomposition). - To know that code may sometimes need to be downloaded onto a physical system/device. - To know that devices with sensors (such as pedometers, security systems, thermostats and light sensors) are often programmed to perform specific tasks in reaction to the input from the sensors.	- To know that there are text-based programming languages such as Logo and Python. - To know that nested loops are loops inside of loops. - To understand the use of random numbers and remix Python code. - To know code may sometimes need to be downloaded onto a physical system/device. - To know programmers often save time when creating code by taking code from one program and turning it into another. - To know nested loops are loops within loops. - To know it is important to follow the syntax rules in a programming language so that the computer understands what we are trying to tell it, but that we do not need to remember all these rules. - To know many text-based coding languages use brackets or indentation to show which code belongs to a particular function or loop.

	- To know that 'coding' is the process of turning an algorithm into a programming language. - To know that HTML tells a browser how to display text, images and multimedia on a webpage. - To know that 'decomposition' is the process of breaking down a task or problem into smaller parts. - To know that breaking down a problem into smaller parts makes it easier to solve the problem. - To know that 'abstraction' is identifying the important detail and ignoring irrelevant information. - To know that loops are used to save time when writing code by reducing repetition. - To know that a variable is a container or holder for storing information that can change, e.g. numbers or text. - To know that conditional statements tell the computer what to do next based on a user's input. - To know that it is important to identify where the mistake is in the programming as part of the debugging process. - To know that errors in a program could result from sequencing errors, coding errors or missing code.	- To know that programmers often save time when creating code by taking code from one program and turning it into another. - To know that nested loops are loops within loops. - To know that it is important to follow the syntax rules in a programming language so the computer understands what we are trying to tell it but that we do not need to remember all these rules. - To know that computers use pixels to measure length in digital visuals. - To know that positions are often understood by a computer as x and y coordinates. - To know that turns or rotations are often described as degrees. - To know that many text-based coding languages use brackets or indentation to show which code belongs to a particular function or loop. - To know that running a program to identify errors should be done before checking the code. - To know that errors in a program could be as a result of forgetting to 'end' a loop. - To know that typing and spacing are very important in text-based languages and can cause errors in code if used incorrectly.	- To know typing and spacing are very important in text-based languages and can cause errors in code if used incorrectly. - To know running a program to identify errors should be done before checking the code. - To know errors in a program could be as a result of forgetting to 'end' a loop.
Computer Systems and Networks		- To know how search engines work. - To know that web crawlers are computer programs that crawl through the internet. - To know the difference between ROM and RAM	- To know that AI is trained on data to recognise patterns and generate outputs. - To know that AI can help generate basic HTML code to create the structure and layout of a website.
Data Handling		- To know what numbers using binary code look like and be able to identify how messages can be sent in this format. - To understand that RAM is Random Access Memory and acts as the computer's working memory.	- To know that data contained within barcodes and QR codes can be used by computers. - To know that infrared waves are a way of transmitting data. - To know that Radio Frequency Identification (RFID) is a more private way of transmitting data. - To know that data is often encrypted so that even if it is stolen it is not useful to the thief. - To know that data can become corrupted within a network but this is less likely to happen if it is sent in 'packets'.

Progression of Skills

	Year 4	Year 5	Year 6
Hardware	- Using tablets or digital cameras to film a weather forecast. - Understanding that weather stations use sensors to gather and record data which predicts the weather.	- Learning that external devices can be programmed by a separate computer - Learning the difference between ROM and RAM. - Recognising how the size of RAM affects the processing of data. - Understanding the fetch, decode, execute cycle.	- Learning about the history of computers and how they have evolved over time. - Using the understanding of historic computers to design a computer of the future. - Understanding and identifying barcodes, QR codes and RFID. - Identifying devices and applications that can scan or read barcodes, QR codes and RFID. - Understanding how corruption can happen within data during transfer (for example when downloading, installing, copying and updating files). - Identify different types of AI and their applications in everyday life.
Networks and Data Representation	Understanding that computer networks provide multiple services, such as the World Wide Web, and opportunities for communication and collaboration	- Understanding how bit patterns represent images as pixels. - Learning that messages can be sent by binary code, reading binary up to eight characters and carrying out binary calculations. - Relating binary signals (Boolean) to the simple character-based language, ASCII. - Recognising that computers transfer data in binary and understanding simple binary addition. - Learning how the data for digital images can be compressed. - Learning the vocabulary associated with data: data and transmit.	Understanding that computer networks provide multiple services.
Computational Thinking	- Using decomposition to solve a problem by finding out what code was used. - Using decomposition to understand the purpose of a script of code. - Identifying patterns through unplugged activities. - Using abstraction to identify the important parts when completing both plugged and unplugged activities. - Using past experiences to help solve new problems	- Decomposing animations into a series of images. - Writing more complex algorithms for a purpose. - Predicting how software will work based on previous experience. - Decomposing a story to be able to plan a program to tell a story - Decomposing a program without support.	- Analysing the effectiveness of prompts and refine them for improved AI outputs. - Writing increasingly complex algorithms for a purpose. - Using past experiences to help solve new problems. - Decomposing a program into an algorithm.
Programming	- Creating algorithms for a specific purpose. - Coding a simple game. - Using abstraction and pattern recognition to modify code. - Incorporating variables to make code more efficient.	- Iterating and developing their programming as they work. - Confidently using loops in their programming. - Using a more systematic approach to debugging code, justifying what is wrong and how it can be corrected.	- Debugging quickly and effectively to make a program more efficient. - Remixing existing code to explore a problem. - Applying coding skills like decomposition and pattern recognition to interact with AI applications.

	• Remixing existing code. • Recognising visual and text-based programming languages. • Recognising further examples of computers being programmed by humans to perform simple tasks. • Working towards a given goal that a program needs to accomplish. • Breaking down what they want to achieve into smaller, manageable parts. • Using logic, pattern recognition and decomposition to solve simple problems. • Tinkering with an existing text-based code to see how it affects a program (website). • Remixing code to alter and add to an existing program. • Recognising repeating patterns in a program or code. • Creating loops to make code more efficient in block-based programs. • Beginning to use variables in block-based programming languages to make programs more interactive. • Including a conditional statement in block-based programming languages. • Recognising the relationship between what is happening in a program and the written (block) code. • Working backwards, beginning to identify the code they think a program uses. • Running small chunks of code at a time to find the error or 'bug.'	• Amending code within a live scenario. • Using repetition within a program. • Using a range of programming commands. • Writing code to create a desired effect. • Programming an animation. • Making links between different programming interfaces they are faced with. • Planning a program to be downloaded to a physical system/device to solve a particular problem. • Programming a physical system that has the capability for sensory input (e.g. a micro:bit). • Recognising a wider range of text-based programming languages. • Making links between different programming interfaces they are faced with. • Recognising examples of programming elements in real-life applications. • Looking at programming blocks and considering how they could be used in a program. • Decomposing a program independently when given a specific outcome or task to achieve. • Live coding (improvising with code). • Altering existing code with a new, specific outcome in mind. • Independently using loops to make code more efficient in text-based programs. • Using nested loops to make code more efficient. • Recognising real-life applications of conditional statements • Using variables in block-based programming languages and understanding the impact of changing the variables in their code. • Explaining what a program does and how it works, referring to the inputs and outputs. • Becoming more efficient and effective at debugging their programs. • Systematically identifying mistakes, problems or 'bugs' in a program.	• Predicting code and adapting it to a chosen purpose. • Evaluating code to understand its purpose. • Changing a program to personalise it. • Programming using the language Python. • Using and adapting nested loops. • Recognising a wider range of text-based programming languages. • Making links between different programming interfaces that they are faced with. • Recognising examples of programming elements in real-life applications. • Decomposing a program independently when given a specific outcome or task to achieve. • Live coding (improvising with code). • Altering existing code with a new, specific outcome in mind. • Independently using loops to make code more efficient in text-based or block-based programs. • Using nested loops to make code more efficient. • Explaining what a program does and how it works, referring to the inputs and outputs. • Becoming more efficient and effective at debugging their programs. • Systematically identifying mistakes, problems or 'bugs' in a program.
--	---	---	---

Aspiration - Resilience - Community