Computing Skills and Knowledge Progression

Computational Skills:

Computing systems and networks

	EYFS	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6
Computational thinking	Logic - Anticipating and explaining. Pattern - Grouping things, comparing, spotting similarities and differences, working out rules. Abstraction - naming and labelling, working out what is important, summarising. Algorithms and decomposition - Responding to instructions, ordering and sequencing, breaking down problems.	To identify that technology is man-made and designed to help people by listing different types. To identify and sort examples of technology in the immediate environment and explain their purpose.	Sort school IT by what it is used for and how it can help the school function. Identify that some IT can be used in more than one way. Demonstrate how IT devices can work together.	Explain that devices accept inputs and produce outputs. To follow a simple process to describe how a device works. Compare digital and non-digital processes for the same task. Suggest some differences and similarities in how they are used.	Demonstrate how information is shared across the internet. To separate web-browsing from other internet activities like online gaming or streaming videos.	Explain that computer systems communication with other devices. Explain the benefits of a computer system. Use online or interactive maps to track how a barcode scanner or smart traffic light passes data through a local system. Relate a search term to the search engines index. Explain that a search engine follows rules to rank results. Give examples of criteria used by search engines to rank results. To order a list by rank.	Explain that internet devices have addresses. Describe how computers use addresses to access websites.
Programming and debugging	Algorithms and decomposition - Responding to instructions, ordering and sequencing, breaking down problems. Creating - Checking and fixing things.			Identify the correct steps to take if a device begins acting strangely after a download.			
Digital creation	Tinkering - Playing and exploring.	Typing individual characters to write their name.	To use IT for different types of activities.	To design a digital device.	Use a web browser to type in a web address (URL), click on links and		Identify different ways of working together online

	Creating - Checking and fixing things. Collaboration - Playing and working collaboratively.	Identify and use the CAPS lock/shift button to create capital letters. To use the backspace to delete letters. Making changes to a text by highlighting and using the backspace to delete.			find specific information on a page. Describe how to access websites on the World Wide Web.		Explain how the internet enables effective collaboration. Choose methods of communication to suit particular purposes.
Data & information handling	Pattern - Grouping things, comparing, spotting similarities and differences. Abstraction - naming and labelling, summarising.	Understand that when work is saved it can be retrieved from a specific place. To locate a saved file on a desktop and open it by double clicking on the mouse.	Sort school IT by what it is used for and how it can help the school function. Find examples of IT. Sort IT by where it if found.	Demonstrate how information can be passed between devices. Explain that devices accept inputs and produce outputs.	Compare information from an unknown website against a trusted source to see if it matches. Identify clues that show a website might not be honest such as missing author names, fake headlines.	Recognise the role of web crawlers in creating an index. Relate a search term to the search engines index. Explain why we need tools to find things online. To refine a web search. Make use of a web search to find specific information.	Send information over the internet in different ways. Explain that the internet allows different media to be shared. Recognise how to access shared files stored online. Identify and explain the main parts of a data packet. Explain that data is transferred over networks in packets. Explain that all data transferred over the internet is in packets.
Digital literacy & Online safety	Collaboration - Playing and working collaboratively. Logic - Anticipating and explaining. Creating - Checking and fixing things.	Develop the ability to seek help and advice when something online feels wrong. Recognise specific online challenges such as mean comments or sharing too much information.	Develop the habit of asking 'Who made this?' and 'Why did they make it?' Identify clues that a website might not be trustworthy, such as strange addresses, spelling mistakes or offers on products. To talk about different rules for using IT. To talk about how rules can keep us safe online. Identify the choices made when using IT.	Recognise a fake email by checking the sender's email address and the tone of a message before trusting its content. Evaluate whether a digital offer seems too good to be true and decide on the safest action. Identify the correct steps to take if a device begins acting strangely after a download. Explain what makes a secure password.	Find and name the person, group or company responsible for making content on the internet. Recognise common signs of ownership on a webpage, such as the copyright symbol or an author's name. Compare information from an unknown website against a trusted source to see if it matches. Identify clues that show a website might not be honest such as missing author names, fake headlines.	Recognise trustworthy websites. Compare results from different search engines. Recognise some of the limitations of search engines. Describe some of the ways that search results can be influenced. Explain how search engines make money.	Decide when information should and should not be shared online. Explain that communication on the internet may not be private. Explain how to report inappropriate content online. Recognise that working together on the internet can be public or private.

					Explain the safe rules of the internet and how they can be enforced.		

Creating media

	EYFS	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6
Computational thinking	Logic - Anticipating and explaining. Creating - Checking and fixing things.	Pressing keys (input) creates letters and words on the document (output). Practice putting a device to sleep or turning it off when given instruction to do so. Navigate a paint program independently. To articulate the reasoning behind a choice. Compare and contrast the two experiences, identifying which they prefer and why. Observe that digital screens use pixels to form images, whereas paper uses pigment from paints, pencils and pens.	Explain why a photograph looks better in portrait or landscape format. Identify the focal point or main subject in an image. Explain why a picture may be unclear. Explain the choice to use a specific tool to change an image. Identify that music is a sequence of notes. Select specific notes to build a short, repeating sequence. To add a sequence of notes to a rhythm. Explain that music can be played in different ways.	To recognise that movement is many static images. Predict how many photographs might be needed for each part of the story. Create a storyboard that translates an idea into a series of planned frames. Work to ensure the scene or set is not accidentally moved between shots. Use the 'Onion Skinning' tool to make precise adjustments to characters or props. Select and open a relevant template for a specific task i.e., a magazine cover. Differentiate between text and image objects on a digital screen.	View 3D shapes from different perspectives. Move 3D shapes relative to one another. Accurately size 3D objects. Show that placeholders can create holes in 3D objects. Choose objects to use in a 3D model based on a design brief. Analyse a 3D model. Use photo editing software to crop an image.	Open, pause, skip and navigate a digital media player to identify production elements. Compare features in different videos. Identify and find features on a digital video recording device. Outline the scenes of a video. Recognise that choices made when making a video will impact the quality of the final outcome. Evaluate a video and share opinions about it. Compare digital vector drawings against traditional paper drawings. Zoom in to edit intricate boundaries accurately and centre shapes. Determine when to use a specific tool to fit a design requirement.	Discuss the different types of media used on websites. Recognise the common features of a web page. Suggest media to include on a page. Plan appropriate content for a podcast. Explain how sounds can be combined to make a podcast more engaging. Discuss what sounds can be added to a podcast. Identify the input and output devices used to record and play sound.
Programming and debugging	Algorithms and decomposition - Responding to instructions, ordering and sequencing, breaking down problems.	To understand that the undo button can revert text and the backspace can get rid of letters/words.	Improve a photograph by retaking it. To review a piece of work.	Play back a sequence to identify glitches or frames that do not fit the flow. Edit the timeline by removing or re-ordering	Explain how a 3D model could be improved. Modify a 3D model to improve it.	Review how effective a video is. Explain how to improve a video by reshooting and editing.	Evaluate and suggest/make edits to a web page. Preview what the web page will look like.

	Creating - Checking and fixing things.			frames to improve the narrative. Evaluate whether the media helped the audience understand the story better. Evaluate a layout based on whether it effectively communicates its intended message. Use the undo function to reverse changes. Make changes to content after being added. Evaluate and improve layout.	Use feedback to guide making changes. Remove parts of an image using cloning.	Select the correct tools to make edits to a video. Make edits to a video and improve the final outcome. Explain how alignment girds and resize handles can be used to improve consistency. Reuse a group of objects to further develop a vector drawing. Reflect on the skills used and why they have been used.	Listen to an audio recording to identify its strengths. Suggest improvements to an audio recording. Choose appropriate edits to improve a podcast. Review the quality of a recording. Improve a voice recording. Re-record a voice to improve a recording. Inspect the soundwave view to know where to trim the recording. Save my project so the different parts remain editable.
Digital creation	Tinkering - Playing and exploring. Creating - Checking and fixing things. Collaboration - Playing and working collaboratively.	To open a word processor and understand its purpose for writing. Pressing keys (input) creates letters and words on the document (output). To apply formatting such as Bold, Italic and Underline. To double-click and hold the left click to drag and rearrange text. To identify that digital writing can be shared or printed for others to read. To use a mouse to click, drag and 'paint' marks on a screen.	To hold a device steady, look at the screen/viewfinder, and press the shutter. Decide on how to hold the device based on the subject (horizontal or vertical). Identify the focal point or main subject in an image. Experiment with different light sources. Use a tool to achieve a desired effect. Apply a range of photography skills to capture a photograph. Create a rhythm pattern. Play an instrument following a rhythm pattern.	To create a physical flipbook. Capture a series of photographs using a digital device where an object moves incrementally. Create a storyboard that translates an idea into a series of planned frames. Select and import appropriate audio or text overlays that match the story. Apply formatting tools to change font size and colour within software like Microsoft publisher or Canva. Toggle between portrait and landscape page settings.	Add 3D shapes to a project. Resize an object. Rotate 3D objects. Duplicate 3D objects. Recolour a 3D object. Lift and lower 3D objects. Combine objects in a design. Create a project that is a combination of other images. Combine text and images to complete a project. Experiment with different colour effects.	Experiment with different camera angles. Make use of a microphone. Capture video using a range of filming techniques. Create and save video content. Decide which filming techniques to use. Experiment with selecting, drawing, changing line weights, and applying fill colours to individual shapes. Click, drag, resize, rotate and duplicate matching geometric shapes to construct a composite vector illustration.	Add content to a web page. Make multiple web pages and link them using hyperlinks. Create hyperlinks to link to another person's work. Draw a web page layout that suits a purpose. Use a computer to record audio. Re-record a voice to improve a recording. Record content following a plan. Arrange multiple sounds to create the desired effect.

		Select a 'line' or 'shape' tool and use it in specific forms. Purposefully change the colour and thickness of a brush before making a mark. Use the 'fill' tool to add colour to a closed shape effectively. Demonstrate an ability to combine various tools to create a final piece (in the style of an artist).	Use a computer to experiment with pitch. Select specific notes to build a short, repeating sequence. To add a sequence of notes to a rhythm. Combine musical elements to communicate an idea to an audience.	Manipulate the position of objects on a page to improve visual appeal.		Use layering to create an image.	
Data & information handling	Abstraction - naming and labelling, working out what is important, summarising	To use the save function to save work to a specific place. Observe that digital screens use pixels to form images, whereas paper uses pigment from paints, pencils and pens.	To identify which photographs are real and which ones have been changed. To identify simple differences in pieces of music. Describe music using adjectives. Connect images with sounds.	Predict how many photographs might be needed for each part of the story. Save and share digital files.	Group 3D objects.	Store, retrieve and export a recording to a computer. Recognise when objects need to be grouped or ungrouped.	Recognise the common features of a web page. Explain the difference between saving a project and exporting an audio file.
Digital literacy & Online safety	Tinkering - Playing and exploring. Collaboration - Playing and working collaboratively.	To practice using kind words in digital conversation. Recognise physical sensations of unease when encountering tricky situations online.	To advise others how to respect copyright and ownership online. To identify which photographs are real and which ones have been changed.	Assess whether a picture is appropriate to share by considering how it might make others feel before clicking 'send'. Identify practical steps to take if an image is shared without permission, such as asking for it to be removed or putting the device down. Evaluate online profiles or message threads to judge whether they show a positive or negative reputation. Use a 'think before you post' check to decide if a piece of work, comment			Explain why copyright-free images should only be used. Find copyright-free images. Use technology safely and responsibly when online. Explore a website. Explain that the person who records a sound can say who is allowed to use it.

				or an image is safe and kind enough to share online. Use emojis appropriately and respectfully.			

Programming

	EYFS	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6
Computational thinking	Logic - Anticipating and explaining. Creating - Checking and fixing things.	To move a physical distance in response to a single spoken command. Use instruction cards or drawing arrows on paper to represent the robot's journey before pressing the buttons. Compare two different sequences that reach the same goal and identifying which one is faster or more efficient. Verbally explaining or drawing a plan (the algorithm) for what they want their animation to do. Testing their code and comparing the animation on screen to their original plan.	Predict the outcome of a given sequence of blocks before running it. Match different command sequences to their corresponding outcomes. Plan quiz questions with multiple outcomes. Break down a simple task into separate, clear steps. Give clear instructions. Use vocabulary like 'forward', 'backward' and 'turn' to describe movements. To look at a program and know where the robot will end up by tracing its instructions, before starting. Plan a route to show how a robot should travel from one point to another. Design a mat for a floor robot.	Predict the outcome of a snippet of code. Evaluate the effectiveness of the repeated sequences used in a program. Identify which parts of a loop can be changed. Select key parts of a given project to use in a design. Develop a design explaining what the project will do. Refine the algorithm in a design. Evaluate the steps followed when building a project. Translate a physical drawing or sketch into a written step-by-step algorithm. Trace code by following it step by step, to predict what the turtle will draw. Break down a complex drawing task into its component shapes. Plan an algorithm before typing code.	Create a program following a design brief. Match coding blocks to their actions. Order notes into a sequence. Combine sound commands. Choose blocks to set up a program. Choose which keys to use for actions and explain why it was chosen. Plan the logical route a sprite must take to move through a path. Consider the real world when design choices. Identify additional features from a given set of blocks. Make design choices and justify them. Evaluate a project.	I can explain that a condition is either true or false. Explain that a condition being met can start an action. Identify a condition and an action in a project. Identify real world examples of a condition starting an action. Describe what a project will do. Design the flow of a program that contains 'if…then…else…' Identify the outcome of user input in an algorithm. Outline a given task. Use a design format to outline a project. Identify the setup code needed in a program.	Identify examples of information that is variable. Look at a game and accurately categorize which information is completely fixed and which is variable. Complete a data-tracking sequence to model how a variable update and replaces its memory. Plan how variables will track goals or limits within the project. Identify examples of conditions in the real world. Explain that checking a variable doesn't change its value. Explain the importance of the order of conditions in 'else…if…' statements. Decide what variable to include in a project.

Programming and debugging	Algorithms and decomposition - Responding to instructions, ordering and sequencing, breaking down problems. Creating - Checking and fixing things.	Identify the symbols for 'forwards', 'backwards', 'left turn' and 'right turn'. To understand that a robot only follows specific instructions. Press a series of buttons and then 'go' to execute the program. Use all four directional keys to navigate a simple path. Incorporate turn commands i.e., turn right 90 degrees. Use instruction cards or drawing arrows on paper to represent the robot's journey before pressing the buttons. Debug programs by identifying and correcting incorrect instructions. Plan, create and debug their own program to achieve a given purpose. To drag a block into the 'script area' to make a sprite perform one action, such as moving forward. Navigate a menu or palette to find specific blocks. Use drag-and-drop to move blocks onto a workspace. Physically snapping blocks together locks into a single script. Selecting a number field within a block to edit it.	Identify and use a 'start' block to initiate a sequence of commands. Demonstrate how to 'run' a program once it has been built. Translate a simple algorithm or visual plan into a working program. To choose the correct blocks to match the steps in a design. Edit an existing program or plan to change its coding. Build a sequence of blocks needed to make the quiz function. Test the complete program to see if it works as intended. Identify specific areas for improvement, such as fixing a bug or adding a new level to the quiz. Use an algorithm to program a sequence on a floor robot. Input instructions into a robot and if there is a bug, identify what it is and fix it. Use the same instructions to create different algorithms. Show the difference in outcome between two sequences that consist of the same instruction.	Use a 'Repeat' block in Scratch to move a sprite across the screen. Modify a snippet of code to create a given outcome. Choose when to use a count-controlled and an infinite loop. Decide between when to use a 'Repeat' block vs a 'Forever' block based on the task. Choose which action will be repeated for each object. Build a Scratch project where one loop controls a timer whilst another controls the character's movement. Combine a 'forever' loop an 'if', 'then', block and a 'sensing' block to create the core game mechanic. Debug a loop that isn't working by checking the sequence of blocks. To re-use existing code snippets on new sprites. Identify and correct simple syntax errors in text-based code. Type commands accurately into a program. Input a sequence of individual commands to move and draw on the screen.	Create a program following a design brief. Match coding blocks to their actions. Navigate the programming environment by clicking and dragging blocks. Click on individual blocks to observe their effect i.e., a sprite moving or making a sound. Start a program in different ways. Create a sequence of connected commands. Build a sequence of commands. Fix mistakes within a sequence. Implement algorithm as code. Order notes into a sequence. Drag, drop and snap blocks together in Scratch to link a keyboard event to a movement block. Build more sequences of commands to make a design work. Edit the size value box of a sprite and use the duplicate tool to quickly copy code scripts across four triggers. To use a programming extension.	Program a microcontroller to make an LED switch on. Connect more than one output component to a microcontroller. Use a count-controlled loop to control outputs. Explain what an infinite loop does. Design sequences that use count-controlled loops. Design a conditional loop. Program a microcontroller to respond to an input. Use selection ('if', 'then' statements) to direct the flow of a program. Use selection to produce an intended outcome. Test and debug a project. To recall how conditions are used in selection. Identify conditions in a program. Modify a condition in a program. Use selection in an infinite loop to check a condition. Identify the condition and outcomes in an 'if'… then… else…' statement. Create a program that uses selection to produce different outcomes.	Use Scratch to manually create, name and display a new functional variable block. Program an existing game to update its scoreboard dynamically by adding a 'change score' block inside an interaction script. Predict the outcome of moving a data change block to different sections of an event loop. Make a functional multi-sprite game matching your initial storyboard structure. Check the software to find and debug errors. Suggest functional upgrades or advances ways to expand the game. Apply knowledge of programming to a new environment. Test a program on an emulator. Transfer a program to a controllable device. Use a variable in an 'if…then…else' statement to select the flow of a program. Determine the flow of a program using selection. Use a condition to change a variable. Use an operand in an 'if…then…' statement.

		Using an on-screen keypad or keyboard to input a new digit. Toggling between different sprites in the "sprite list" or selector to view their individual code. Adding or deleting sprites to a project. To input a specific command for Sprite and for it to follow that command. Testing their code and comparing the animation on screen to their original plan.		Translate a written algorithm into formal code. Identify sections of code where commands are repeated and could be replaced by a loop. Modify the count in a loop to draw different regular polygons. Change the values in an existing repeat command to fix a broken shape. Write and test a full program that uses at least one repeat loop. Debug the program to fix errors.	Choose suitable keys to turn on additional features. Test a program against a given design. Modify a program using a design. Match a piece of code to an outcome. Implement a design.	Explain that program flow can branch according to a condition. Show that a condition can direct program flow in one of two ways. Implement an algorithm to create the first section of a program. Test a program. Extend a program further.	Modify a program to achieve a different outcome. Test a program against a design. Use a range of approaches to find and fix bugs. .
Digital creation	Tinkering - Playing and exploring. Creating - Checking and fixing things. Collaboration - Playing and working collaboratively.	To use basic paint tools (brush, bucket, rubber) to create or edit a sprite or backdrop. Labelling or naming objects within a design. To input a specific command for Sprite and for it to follow that command.	Plan quiz questions with multiple outcomes. Build a sequence of blocks needed to make the quiz function.	Build a Scratch project where one loop controls a timer whilst another controls the character's movement. Combine a 'forever' loop, an 'if then' block and a 'sensing' block to create the core game mechanic. Develop a design explaining what the project will do. Input a sequence of individual commands to move and draw on the screen. Modify the count in a loop to draw different regular polygons. Write and test a full program that uses at least one repeat loop.	Build a complete musical instrument project by combining sounds and movement. Add visuals to a project based on a brief. Combine sound commands. Implement a design. Build more sequences of commands to make a design work. Choose suitable keys to turn on additional features. Make design choices and justify them.	Create a detailed drawing of a project. Write an algorithm that describes what the model will do. Create a simple circuit and connect it to a microcontroller. Create a program that uses selection to produce different outcomes. Implement an algorithm to create the first section of a program. Use a design format to outline a project.	Sketch out a clear storyboard layout mapping game flow, user targets and scoring. Plan how variables will track goals or limits within the project. Make a functional multi-sprite game matching your initial storyboard structure. Design the algorithm for a project. Design the program flow for a project. Create a program based on a design. Modify a program to achieve a different outcome.
Data & information handling	Abstraction - naming and labelling, working out what is important, summarising	Labelling or naming objects within a design.	Show the difference in outcome between two sequences that consist of the same instruction.	Predict the outcome of a snippet of code.	Match a piece of code to an outcome.	Program a microcontroller to respond to an input.	Identify examples of information that is variable.

			To look at a program and know where the robot will end up by tracing its instructions.	Evaluate the effectiveness of repeated sequences. Identify sections of code where commands are repeated and could be replaced by a loop. Match coding blocks to their actions. Click on individual blocks to observe their effect i.e., a sprite moving or making a sound.		Explain that a condition being met can start an action. Identify the outcome of user input in an algorithm. Identify the condition and outcomes in an 'if…then…else…' statement.	Look at a game and accurately categorize which information is completely fixed and which is variable. Complete a data-tracking sequence to model how a variable update and replaces its memory. Program an existing game to update its scoreboard dynamically by adding a 'change score' block inside an interaction script. Use a variable in an 'if…then…else' statement to select the flow of a program. Use a condition to change a variable. Explain that checking a variable doesn't change its value.
Digital literacy & Online safety	Tinkering - Playing and exploring. Collaboration - Playing and working collaboratively.	Question the reliability of AI. Spot potential harmful content or chatbots who have negative intent. Develop safe choices by applying safe filtering. Identify specific emotions in response to digital content. Observe and differentiate between 'fun' videos and those that might cause uneasiness. Practice verbalising a worry to an adult.	Link the story of Hanni to personal uses of devices at home and school. Identify the type of content that makes us feel happy verses scary or worried. Draw or list their personal safety network. Practice using specific language to stop unkindness i.e., "That isn't funny, let's stop". Evaluate a message of 'joke' before it is sent to decide if it is helpful, kind or necessary.	Develop strategies to cool down or take a break if a game becomes frustrating or upsetting. Work with peers to resolve conflicts that arise during multiplayer games without escalating the situation. Evaluate the pros and cons of sharing information or making a purchase within a game environment.	Consider the real world when design choices.	Identify real world examples of a condition starting an action. Share a program with others and receive feedback.	Export or share the operational game file to a class studio. Identify examples of conditions in the real world.

Data and Information

	EYFS	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6
Computational thinking	Logic - Anticipating and explaining. Creating - Checking and fixing things.	Determine the most appropriate label for a collection of diverse objects. Explain that a computer cannot group or count objects until a human provides the input and rules. Formulate a 'rule' for sorting and apply it across a dataset. Explain how the same set of data can be reorganised into different structures by changing the property focus. Select the correct grouping method to solve a specific problem.	To choose a suitable attribute to compare people. Compare totals in a tally chart. To answer 'more than'/'less than' questions.	Decompose a large group of objects into smaller groups. Identify which data is relevant for a specific purpose. Select the most effective attribute to split objects into two groups. Compare two different branching structures to identify which one is more direct and easier to follow.	Choose a data set to answer a given question. Suggest questions that can be answered during a given data set. Propose a question that can be answered using a data logger. Plan how to collect data using a data logger. To sort data to find information. Interpret data that has been collected using a data logger. Draw conclusions from the data.	Choose which field and value are required to answer a given question. I can outline how 'and' and 'or' can be used to refine data selection. Choose multiple criteria to answer a given question. Ask questions that will need more than one field to answer. Refine a search in a real-world context.	Suggest when to use a table or a chart. Suggest how to structure data. Construct a formula in a spreadsheet. Explain which data types can be used in calculations. Create a formula which includes a range of cells. Apply a formula to calculate the data I need to answer questions.
Programming and debugging	Algorithms and decomposition - Responding to instructions, ordering and sequencing, breaking down problems. Creating - Checking and fixing things.	Formulate a 'rule' for sorting and apply it across a dataset. Explain that a computer cannot group or count objects until a human provides the input and rules.	Organising data systematically (tally → pictogram). Following structured processes to represent data	Test questions to ensure they are not ambiguous. Check and correct the tool by having another peer use it. Correctly order questions so each level of the tree leads towards a single object.	Plan how to collect data using a data logger. Use a data logger to collect data. Successfully load an existing data file or export a log into a table.	I can outline how 'and' and 'or' can be used to refine data selection. Choose multiple criteria to answer a given question. Refine a chart by selecting a particular filter.	Construct a formula in a spreadsheet. Apply a formula to multiple cells by duplicating it. Calculate data using different operations.
Digital creation	Tinkering - Playing and exploring. Creating - Checking and fixing things. Collaboration - Playing and working collaboratively.		Create a pictogram and draw conclusions from it.	Use a digital tool to build a functional database. Create a complete database for a specific topic.		Create a database using cards.	Produce a chart.
Data & information handling	Abstraction - naming and labelling, working out what is important, summarising	To identify and assign name labels to digital images. Match digital objects to specific group labels using 'drag-and-drop' actions.	To record data in a tally chart. Represent a tally count as a total. Compare totals in a tally chart.	Investigate and formulate simple questions about a collection of objects, where the answer is only 'yes' or 'no'. Test questions to ensure they are not ambiguous.	Choose a data set to answer a given question. Suggest questions that can be answered during a given data set.	Explain how information can be recorded. Order, sort and group data cards. Explain what a field and a record is in a database.	Collect data. Enter data into a spreadsheet. Suggest how to structure data.

		Determine the most appropriate label for a collection of diverse objects. Use a digital tool to count and verify the total number of objects in a pre-labelled group. Identify similar properties (attributes) across different digital objects to justify grouping. Describe an object by its digital properties to make it searchable for a computer. Formulate a 'rule' for sorting and apply it across a dataset. Explain how the same set of data can be reorganised into different structures by changing the property focus. Select the correct grouping method to solve a specific problem i.e., 'which group has the most objects?' Compare digital groups to identify similarities and differences in the data results. Share and record digital findings with peers explaining the logic behind the data structure.	To use a computer program to present information in different ways. Organise data in a tally chart. Tally objects using a common attribute. To use a tally chart to create a pictogram. Explain what a pictogram represents. Create a pictogram to arrange objects by an attribute. To use pictograms to answer simple questions about objects. To answer 'more than'/'less than' and 'most'/'least' questions. To choose a suitable attribute to compare people. Collect data. Create a pictogram and draw conclusions from it. To use a computer program to present information in different ways. To enter data into a computer. To use a computer to view data in a different format.	Observe a set of objects and list their physical and functional attributes. Select the most effective attribute to split objects into two groups. Use a digital tool to build a functional database. Correctly order questions so each level of the tree leads towards a single object. Compare two different branching structures to identify which one is more direct and easier to follow. Create a complete database for a specific topic. Check and correct the tool by having another peer use it.	Identify data that can be gathered over time. To use data from a sensor to answer a given question. To identify how much time has passed between each sample. To sort data to find information. Successfully load an existing data file or export a log into a table. Propose a question that can be answered using a data logger. Plan how to collect data using a data logger. Use a data logger to collect data. Interpret data that has been collected using a data logger. Draw conclusions from the data. Present findings to a partner or adult using the data to back up their claims.	Navigate a flat-file database to compare different views of information. Choose which field to sort data by to answer a given question. Explain that data can be grouped using chosen values. Group information using a database. Combine grouping and sorting to answer specific questions. Choose which field and value are required to answer a given question. Choose multiple criteria to answer a given question. Ask questions that will need more than one field to answer. Refine a search in a real-world context.	Choose an appropriate format for a cell. Apply an appropriate format to a cell. Construct a formula in a spreadsheet. Explain which data types can be used in calculations. Calculate data using different operations. Create a formula which includes a range of cells. Apply a formula to multiple cells by duplicating it. Use a spreadsheet to answer questions. Explain why data should be organised. Apply a formula to calculate the data I need to answer questions. Produce a chart. Use a chart to show the answer to a question.
Digital literacy & Online safety	Tinkering - Playing and exploring.	To apply a strong password recipe by combining capital letters,	Demonstrate how to check and change basic			Refine a search in a real-world context.	Explain why data should be organised.

	Collaboration - Playing and working collaboratively.	lowercase letters, numbers and symbols. Develop the ability to spot 'red flags' such as emails with poor spelling or requests for passwords from strangers.	privacy settings on a familiar device. Spot the clues that might accidentally give away private information to strangers. Give simple examples of why information should not be shared.				

Knowledge:

Computing systems and networks

EYFS	Year 1: Technology around us	Year 2: Information Technology around us	Year 3: Connecting computers	Year 4: The Internet	Year 5: Systems and Searching	Year 6: Communication and Collaboration
	Technology is something man-made that helps us solve a problem or do a job.	Understand that Information Technology (IT) refers to any equipment that allows us to create, store or communicate digital information.	Define a 'network' as a group of devices connected together to share data.	The internet is a network of networks.	A computer system is a collection of connected hardware parts and software applications working together to perform a task.	Every device connected to the internet has a unique numerical internet protocol (IP) address.
	Technology is everywhere; at home, in school and in shops.	IT is found in many public places, such as shops (tills/scanners), hospitals (monitoring equipment) and libraries.	Understand that the internet is used to provide many services.		Many everyday infrastructure tasks are managed by hidden, large-scale computer systems.	The Domain Name System (DNS) acts like a phone book, translating human-friendly web addresses into numerical IP addresses that computers understand.
	A computer is a specific type of technology that processes information.	Know that many different devices contain computers, such as tablets, laptops and smartphones.	Know that digital devices have a computer inside programmed for specific tasks.	Recognise that the World Wide Web contains websites and web pages.	Connected systems send electronic or wireless signals to share data with other peripheral devices, such as a network server or printer.	Files, images and messages do not travel whole; they are broken down into tiny, numbered pieces called packets.
	A computer is a specific type of technology that processes information. Different parts of technology have different jobs. Some parts let us talk to the computer and others let the computer talk to us.	Know that devices often work together.	Identify key parts of a network i.e., switch, server and wireless access point. Know that devices can connect using physical cables or invisible radio waves i.e., WIFI or 46/5G etc.	Know where websites are stored when uploaded to the World Wide Web. Everyone can add content to the World Wide Web.	A search engine is specialised software tool designed to hunt for information across the World Wide Web.	Packets take different paths across networks to reach their destination efficiently and safely. Cloud platforms allow multiple users to edit, and contribute to the same document in real time from anywhere in the world. Files can be stored on remote internet server (the cloud) rather than locally on a single machine.
	Pressing a key sends a digital signal to the computer to display the character.		Understand the input-process-output model.		Systems are not entirely independent, they require human elements to design,	

					input data and use the system and override errors.	
	The internet is a fun and useful place for learning and play when use safely.	There are rules for using IT to keep us safe and healthy, such as taking breaks from screens and not sharing personal information.	Understand that malware is bad software designed to damage or take control of a device. Recognise that you should never open attachments of click links from people you don't know.	There are rules to protect content. Not everything on the World Wide Web is true.	Search results can be altered or personalised based on a user's location, browsing history or profile. Sponsored results or advertisements are organisations paying for the search engines to promote their website.	Online communication spaces require strict e-safety protocols to protect privacy and mental-health.

Creating media (1)

EYFS	Year 1: Digital writing	Year 2: Digital music	Year 3: Desktop publishing	Year 4: 3D modelling	Year 5: Vector graphics	Year 6: Web-page creation
	Computers are tools that help us create digital documents.	Understand that a computer is a tool for creating and manipulating sounds.	Identify the uses of desktop publishing in the real world.	Construct a complete project independently based on an initial design.	Creating a final illustration relies on combining, planning, layering, grouping and colouring to meet a goal.	A website is a collection of interconnected web pages hosted on a web server under a single domain name.
	A keyboard is an input device used to send letters to the screen. Opening a specific program and identifying that the cursor shows where the next letter will appear.	Digital music is created by drawing notes.	Understand the concept of a 'text box' to begin typing anywhere on the page.	Know how to use specific handles to alter a shape's size across one, two or three dimensions. Understand how to change an object's elevation to lift or lower it off the floor grid.	Advanced software features like alignment guides and snapping grids assist designers in sizing and placing objects relative to each other.	A hyperlink is a digital link that connects one web page to another when clicked.
	Digital text is not permanent and can easily be changed or undone. The backspace key deletes the character to the left of the cursor. Digital text is flexible; you can change its appearance after you have written it.	Some software allows you to change a sound's pitch, volume or instrument type instantly.	Text can be changed to communicate more clearly.	Digital objects possess properties that can be changed. Check a digital build from multiple camera views to spot design flaws and adjust dimensions or placements.	Grouped objects are not permanently fused, they can be ungrouped at any time to edit the smaller properties inside. Every line, circle or polygon added to a canvas is treated by the computer as a standalone, independent object.	Most online media assets belong to the person who made them and are protected by copyright rules. Creators must use creative commons or public domain markers to let others borrow their work safely.
	A font is the style or shape of letters. Size refers to how big or small the text is. Colour can be changed using a palette tool.	Recognise different musical elements i.e., 'tempo' and 'pitch'.	Know the terms for font properties: size, colour and style i.e., bold, italic and underline.	2D shapes have two dimensions (width and height), whilst 3D objects have three (width, height and depth). 3D shapes can be rotated around three different axes to orient them correctly.	A vector drawing is a digital image made entirely from mathematically defined lines and 2D shapes, rather than rows of tiny square pixels. Vector drawings can be scaled up or down infinitely without ever losing their quality or getting blurry.	The homepage acts as the main entry hub, serving as the starting point for visitors to access all other sub-pages.
	Design choices affect how easy a document is to read.	Realise that music is created with a specific goal in mind	Create a template for a specific purpose.	Choose the most accurate 3D shape to execute a design.	Complex real-world objects can be broken down into a	Sites are designed with targeted audiences and

	Choosing a font that 'matches' the mood of a sentence.	like representing an image or telling a story.	Match a layout to a purpose i.e., newsletter, brochure, magazine poster or invitation.	Map out a building or project prior to working on the software.	layout of simple geometric blocks.	specific goals in mind, which dictates how information in laid out.
	Evaluation is thinking about whether the work meets its purpose. Articulating why a specific change was made.	Reviewing means listening back to the work critically. Understand that the first version of a piece of music is called a 'draft' and can be improved.	Compare work made on desktop publishing to work created by hand.	Check a digital build from multiple camera views to spot design flaws and adjust dimensions or placements.	Assessing digital art involves comparing the final canvas layout directly back against the original design checklist to identify errors.	Web designers must constantly test their pages to fix errors, spelling mistakes or layout bugs before allowing the public access.
	Writing by hand uses a pen/pencil, writing digitally uses a keyboard/screen.	Understand that music is a form of communication.	Text and images can communicate a message clearly. Identify the advantages and disadvantages of using text and images.	Know how to visually break down real-world items and buildings into their basic shapes.		A good design grid keeps pages organised so users can easily read and scan the information.

Creating media (2)

EYFS	Year 1: Digital painting	Year 2: Digital photography	Year 3: Stop-frame animation	Year 4: Photo editing	Year 5: Video production	Year 6: Audio production
	Understand that digital work belongs to the person who has created it.	Someone else's work, like a photo or video, belongs to them.	Recognise that before taking or sharing a photo of someone else, permission must be given first.		When recording files across shared hardware, content must be saved carefully to prevent overwriting or deleting another user's work.	
	Understand that not everything on the internet is true or helpful.	Understand that images seen online may not be real.	Understand that once an image is shared online, it can be viewed by many people and the person who originally shared it loses control over where it goes next.	Explain why certain colour effects have an effect on someone.		
	Know that digital tools like the 'paintbrush' make marks and the 'eraser' removes them. Understand that different tools are designed for different jobs i.e., the 'fill' tool is faster for large areas and the pencil is better for fine details.	Identify that various devices i.e., tablets, phones can capture photos.	Know simple animation techniques.	Cropping cuts away the outer edges of a picture to remove unwanted items, change the focal point or reshape the final image.		
	Know that 'careful choices' include selecting appropriate colours, brush sizes and specific tools to achieve a desired outcome.	Identify the core components of a good quality photograph (lighting, subject and framing)	Understand that animation is a technique where still images are shown quickly to create the illusion of movement. Realise that consistency in lighting and object placement	Background and foreground objects must work together harmoniously. An edit is successful if it directly if it fulfils its intended purpose and looks visually	Effective video clips rely on specific choices like framing, steady tracking, audio and pacing.	Sound recording requires an input device (microphone) to capture physical sound waves and an output device (speakers or headphones) to play it back.

	Understand that different brushes in a program (like Paint) can create different artistic effects, such as thin lines or spray-paint textures.		is vital for a professional-looking result. Identify a criterion for a successful stop-motion clip.	clean, rather than just using as many tools as possible.		Evaluation means measuring the final product against the initial script and checking it for technical clarity and listener enjoyment.
	Know the basic operational steps of digital painting i.e., opening the application and using the 'canvas area'.	Understand the basic sequence of taking a photo.	To explain how an animation/flip book works.		Video is a visual media format created by compiling a fast sequence of photographs.	
	Understand that computers have specific tools for drawing geometric shapes (square, circles) and straight lines.	Understand the difference between landscape and portrait. Know that the choice of orientation depends on the subject of the photograph.	Understand the concept of 'Onion Skinning'; a digital tool that shows a faint image of the previous frame to align the current one.	Rotating turns an image around a fixed central point which can turn a portrait photo into a landscape photo.	Each frame layout must account for the physical shot type, actor positioning and matching the audio to script.	A structured podcast relies on a distinct sequence of pieces, including an introduction (intro), body content, transitions, and an ending (outro).
	Understand that digital paint can be changed or 'undone' using the 'undo' tool if a choice doesn't work out.	Editing software can change how a photograph looks after it has been taken. Know the 'bin' icon is how to delete a photograph.	Understand that deleting or re-taking a frame is often part of the process.		A video editor timeline allows clips to be trimmed, split, reordered or deleted.	Audio software allows users to trim, delete, or rearrange sound fragments without ruining the raw file. Splicing is used to remove mistakes, coughing, or long pauses to make speech sound fluent.
	Recognise that tools require different interactions than freehand drawing.	Recognise common mistake such as fingers over the lens, blurriness or a subject being too far away.	Understand that for movement to look smooth, objects must be moving a small amount each time. Recognise that large gaps between positions result in jumpy movement.	Cut out elements from separate files will naturally have different sized and must be resized to fit their new surroundings.	Successful videography requires managing background noise, using microphones and ensuring consistent lighting.	Good audio capture relies on low background noise and proximity to the person speaking.
		Recognise that photographs can be manipulated for a specific purpose.	Understand that sound effects, background music and titles can change the mood or feel of an animation. Recognise how a change in background can change the context of a character's actions.	Review images against a given criteria. Different colour effects make you think and feels different things.	Different shot lengths are chosen to convey emotion or reveal context.	

Programming A

EYFS	Year 1: Moving a robot	Year 2: Robot algorithms	Year 3: Repetition in shapes	Year 4: Sequencing sounds	Year 5: Selection in physical computing	Year 6: Variables in games
	An algorithm is a precise set of ordered instructions.	Understand that an algorithm is a precise set of instruction to perform a task.	Write an algorithm to achieve a specific outcome, considering the order and structure of instructions.	Recognise that commands in Scratch are represented as programmable blocks.	Physical computing involves connecting software code on a computer to real-world	A variable is a named space or container in a computer's memory that stores information that can vary or

			Implement an algorithm using a text-based programming language.	Identify and describe the key components of a Scratch project, including sprites, backdrops and the stage.	hardware component through an interface board.	change while a program is running. Variables can hold different types of data values, including numbers or letters.
	A sequence is when two or more commands are put together in a specific order.	A sequence is a series of instructions that must be followed one after another, in a specific order to achieve an outcome.	Understand that a count-controlled loop can produce a given outcome.		A finalised automated product depends entirely on code logic coordinating perfectly with stable, physical components.	Real-world games rely on variables to track volatile elements, such as the countdown timer or goal tally in a football match.
	The order of the sequence changes the final result.	Know that changing the order of instructions in a sequence changes the final result.	Recognise when a sequence of instructions could be made more efficient by using repetition.		An electrical circuit must form a complete, continuous loop from a power source, through components like a bulb and back to a ground connection to function. A loop is a programming control that repeats a set block of instructions to make code shorter, cleaner and more efficient.	
	A command is a single instruction for a computer or robot.	Instructions must be completely clear so the robot doesn't get confused.	Understand that a computer follows instructions.	Recognise that commands in Scratch are represented a programmable block. Understand that each sprite follows its own sequence of instructions independently.	Component inputs like light sensors, push buttons or tilt switches, feed digital data back into the system.	A variable is a named space or container in a computer's memory that stores information that can vary or change while a program is running.
	Robots do not think for themselves. They follow the exact button pressed. Buttons on a robot are inputs.	Recognise that robots follow instructions literally, so the input needs to be specific and clear.	Understand that computers execute programs by following sequences of precise instructions exactly as written.	Understand that changing the order of blocks will change the behaviour of the program.		Programmers must give variables clear and descriptive names so the program logic can find and track them easily.
	A turn command (Left or Right) makes the robot spin on the spot but does not move it to a new square.	A sequence is a series of instructions that must be followed one after another, in a specific order to achieve an outcome.	Identify repetition and recurring patterns within a sequence of instructions. Understand that a count-controlled loop repeats a sequence of instructions a fixed number of times.	Explain that programs execute exactly as the blocks are sequenced. Understand that sequencing involves organising blocks in a precise order to achieve a desired outcome.	A condition-controlled loop keeps repeating instructions based on an ongoing rate rather than a static count.	
	Planning happens before touching the robot to avoid mistakes.	Understand that a good design involves identifying the final goal and the steps needed to reach it successfully.	Design a program that uses count-controlled loops to achieve a specific goal. Use a design template or diagram to represent the intended behaviour of a program before coding.	Plan, create and test a program using sequences of blocks. Design the behaviour of multiple sprites within a single project.	Mapping out a system before building ensures that inputs and outputs respond correctly.	Before building a game, designers must step back from coding to map out the visual elements and operational rules separately.

	Breaking a journey into small, individual steps is called decomposition.	Understand that a good design involves identifying the final goal and the steps needed to reach it successfully.	Break down a problem into smaller parts by identifying repeated patterns and grouping actions together.	Translate a task description into a structured program design.	Successful physical computing projects require a balanced design layout that accounts for both physical building materials and software logic constraints	Designing variable-controlled games requires writing clear algorithms that detail how data variable interact with game rules. Turning an algorithm into functional blocks requires translating written layout concepts.
	Debugging is the process of fixing a program that isn't doing what you expected	Debugging is the process of finding and fixing mistakes. Errors may be caused by missing steps, putting instructions in the wrong order or pressing the wrong buttons.	Develop and improve a program by identifying, testing and fixing errors through debugging. Recognise that even a small error in a program can prevent it from running correctly or producing the expected outcome.	Understand that changing the order of blocks will change the behaviour of the program.	When an automated system fails, a programmer must diagnose whether the issue is an error in the software blocks or a mechanical fault in the wiring.	Software evaluation means comparing the final running product back to the original plan.
	There is often more than one 'right way' to reach a destination.	Understand that logical reasoning means using what you have already learnt or know to work out what will happen.	Explain how changing the number of repetitions in a loop affects the overall outcome of a program. Select appropriate values within a loop to control how many times a sequence is repeated.	Understand that different coloured blocks represent different categories of actions (e.g., Motion, Sound, Events).	Successful physical computing projects require a balanced design layout that accounts for both physical building materials and software logic constraints.	Programmers must give variables clear and descriptive names so the program logic can find and track them easily.
		Understand that a program is the implementation of an algorithm.	Understand that an algorithm must be translated into code for a computer to execute it.	Recognise that commands in Scratch are represented as programmable blocks.	A condition is a true/false statement that breaks a loop cycle instantly when its state changes.	Game variables require specific script triggers to alter their value. Variables can hold different types of data values, including numbers or letters.
			Predict the outcome of a program that uses count-controlled loops by tracing the instructions step by step.	Predict how a sequence of blocks will affect the behaviour of a sprite.	Mapping out a system before building ensures that inputs and outputs respond correctly.	Designing variable-controlled games requires writing clear algorithms that detail how data variable interact with game rules.
					To capture real-world changes instantly a program must use an infinite loop to force the computer to evaluate a condition millions of times per second.	The specific slot where a 'change variable' block is dropped in a script dictates exactly how and when a score increases or decreases.

Programming B

EYFS	Year 1: Programming animations	Year 2: Programming quizzes	Year 3: Repetition in games	Year 4: Events and actions in programs	Year 5: Selection in quizzes	Year 6: Sensing movement
	'Debugging' is the process of fixing errors when the program doesn't match the algorithm.	Know that 'debugging' is the process of finding and fixing errors in a program. Understand that a project can be made better by making it more efficient.	Recognise that if a 'Forever' loop is empty or has a mistake, the whole game might freeze or fail to react.	Code needs to be systematically tested against a design brief to see it if acts the way it needs to. A bug is an error or mistake in the code sequence that stops the program from running exactly as it is intended to.	Programs must be checked multiple times by entering both correct and incorrect values to confirm that both branches of code work.	Physical machines require diagnostic hardware checks under separate conditions to verify that sensors and pins react perfectly every time.
	A project is made of both code (instructions) and artwork (sprites/backgrounds).	Know that the outcome is the final result or behaviour produced by a program.	Understand that an infinite loop (e.g., 'Forever' block) repeats a set of instructions indefinitely.	Programming toolsets can be expanded with extension blocks to give sprites entirely new capabilities.	Programs need clear initialisation scripts to ensure it behaves consistently each time is it run.	Creating systems requires constantly checking the active block code against the step-by-step goals written in the initial design algorithm.
	Understand the concept of a sequence. Know that commands run from top to bottom, or left to right.	Understand that changing the order of commands in a sequence will likely change the final outcome.	Realise that an 'if' or 'then' statement is used to trigger the 'buzzer' sound only when the wire is touched.	Programs need a starting state instruction block so the project resets cleanly every single time the start flag is pressed. Blocks of code are read in a sequence from top to bottom, but different sprites run on completely separate scripts.	Code does not always run in a straight, linear sequence. Selection causes the program's path to branch in separate directions. An 'if…then…else' block splits the program flow in exactly two ways, one if the code is true and one if the code is not.	Computers read code rules sequentially from top to bottom.
	That a 'start' block is often needed to trigger a sequence.	Understand that a computer program needs a specific trigger or instruction to begin running. Know that this is often a 'start' block like a green flag.	Recognise that games need infinite loops to constantly listen for player inputs or collisions (like the wire in Operation).	An event is an input or trigger (like pressing a key) that causes an action (like a character moving) to happen in a program.	Selection is the programming concept where a computer system evaluates a specific true/false statement to choose which pathway of code to execute next.	Selection means using conditional logic to branch the direction or path a program takes.
	A value is a number within a command that can be changed (e.g., steps, degrees, or seconds).	Understand that a program's design can be modified to create a different outcome or add new features.	Understand that changing the code inside an infinite loop changes the behaviour of the entire program immediately.	Add features requires assigning memorable control keys so a player can easily interact with the game mechanics.	Programs use the 'ask' block to collect a string of text from a user and temporarily story it inside an 'answer' block.	Physical inputs (like clicking button A, shaking an accelerometer, or touching a pin) can trigger internal data alterations.
	Understanding that a larger number might make a sprite move further or wait longer.	Know that modifying involves purposefully changing or editing specific parts of a plan.	Understand that changing the code inside an infinite loop changes the behaviour of the entire program immediately.	The colour, thickness and style of a digital pen trail can be altered dynamically by changing values within blocks.	Evaluation requires comparing the operational code back against the design brief to measure its success.	User inputs can be programmed to add to, subtract from, or overwrite a data cell directly while the machine runs.
	A sprite is a character or object on the screen that can be programmed.	Recognise that creating a personal design involves making independent choices about characters, backgrounds and the running of the quiz.	Identify that different sprites often need their own separate loops to function independently.	A character moves along an invisible grid where changing directions involves changing the face angle. Sprites must be scaled correctly using percentage sizes so they physically fit		

				within the boundaries of a background layout.		
	Understanding that an algorithm is a precise plan, while a program is that plan turned into code.	Understand that a design is a plan for what a program should do.	Understand that decomposition is to break down a big game idea into smaller parts. Identify where repetition is needed in a game design.	Successful programming includes knowing the task, designing, writing the code and running the game.	Before typing or assembling code blocks, programmers must plan out the questions, conditions, and branching pathways using pseudo-code or flow diagrams.	Before compiling scripts, systems designers map out data variables and code flows using written algorithms or flowcharts to reduce bugs.
	Understanding that a project should be planned (designed) before it is built.	Recognise that programmers often follow a design to ensure the final product meets a specific goal, such as asking a quiz question. Understand the cycle of design – code – test.	Identify where repetition is needed in a game design.	Code needs to be systematically tested against a design brief to see it if acts the way it needs to.	Translating a paper design into working code requires precise block placement, operator matching and variable tracking.	Reading, checking, or evaluating a variable's state to make a decision does not modify the number held inside it.
	Understand that while the internet is fun, some content can be upsetting or scary. Know that it is not your fault if you see something that makes you feel uneasy.	Understand that computers, tablets and phones are like 'Magic windows' that let us see into different world. Know that what happens behind the screen is not always real, but the feelings it gives us are.	Recognise that being respectful includes following game rules, not cheating and remaining that way whether a player wins or loses. Games are designed to encourage players to spend real money and that this should never be done without permission from an adult.			

Data and information

EYFS	Year 1: Grouping data	Year 2: Pictograms	Year 3: Branching databases	Year 4: Data logging	Year 5: Flat-file databases	Year 6: Introduction to spreadsheets
	To group objects by properties.	Know that data can be sorted into groups based on these shared attributes.	Recognise that to distinguish one object from another, we can look at the attributes they possess.	Data can be collected using sensors.	Data is raw text or numbers without context.	Number formatting changes how the computer reads and interacts with data items (e.g., setting a cell to Currency £, Percentage %, or Date), which is completely different from style formatting like fonts or colours.
	Identify the properties of objects and begin to understand that properties can be used to group objects i.e., colour or size.	Understand that an attribute is a characteristic of an object, such as its colour, size, or shape.	Understand the term attribute as a characteristic, feature, or property of an object i.e., colour, size or number.	A sensor is a type of hardware that reads the physical world and turns it into digital numbers.	An object has descriptive categories called attributes and specific entries called values.	Spreadsheets use specific digital symbols for calculations like an asterisk for multiplication and a forward slash for division.

						A colon symbol is used to define a range of cells spanning across a row or column.
	Understand that objects can be described in different ways. Label a group of objects.	Understand that an attribute is a specific characteristic or quality that describes someone.		There are different ways to view data.	Digital databases present data in a record view or table view.	Spreadsheets arrange data into vertical columns and horizontal rows.
	Name different objects and place them into groups.	Understand that a tally chart is a quick way to record data as it is collected.	Identify the components of a database.	Identify that data from sensors can be recorded.	A flat-file database organises structured data into individual items called records within a single uniform table.	Spreadsheets arrange data into vertical columns and horizontal rows. Different styles of charts communicate different stories (e.g., a bar chart shows distinct comparisons, while a line graph displays changes over time).
	To count groups of objects with the same label.	A single mark represents one item and a diagonal mark through four marks represents five.	Define a branching database as a collection of data organised in a tree structure using yes/no questions.	Data is raw numbers or measurements. It only becomes information when we use it to answer a question.	Information is data that has been organized meaning.	Data headings are critical labels at the top of columns to tell the user and the computer exactly what kind of data is stored underneath them.
	Compare groups by thinking about how they are similar or different.	Know that each picture stands for a specific number of objects.			Combine different records into sets based on shared field similarities or values.	An item of data is a single piece of information sitting inside an individual active cell. A cell reference is a unique alphanumeric code made up of the column letter followed by the row number (e.g., C4) that specifies an exact grid location.
	To explain how the groups have been made.	Recognise that the pictures must be the same size and lined up clearly to allow for easy comparison.	Understand that a well-structured database is efficient because it allows users to find information using the fewest questions.	A data logging plan must run long enough to catch a trend.	Converting data records into visual charts makes it vastly easier for humans to spot patterns, trends and anomalies.	Presenting dense rows of numbers as a chart or graph makes patterns, spikes, and anomalies instantly visible.
	Understand that an object can fit into more than one group depending on the context.	Know that attributes are used to group people when collecting data for a survey.	Recognise that to distinguish one object from another, we can look at the attributes they possess.	To view data at different levels of detail.	Organising a single column of records into an alphabetical or numerical sequence.	Rather than writing out the same formula dozens of times, programmers use features like autofill to clone a formula down an entire column instantly.
	Learn that computers are not intelligent and require input from humans to perform tasks.	Recognise that using a computer to create charts is often faster and neater than drawing by hand.	Understand that computers process information most effectively when it is narrowed down to a choice.	Collecting data over a long period of time makes answering questions more accurate.	Advanced searches use 'and' and 'or' to filter files accurately. Search tools isolate specific fields by looking for an exact	A spreadsheet only knows it is being asked to perform a calculation if the formula starts with an equal's sign.

					value match to strip out unneeded data.	
	Recognise anything shared online can remain there forever.	Know that 'public' means anyone can see your posts and 'private' limits the audience to only people you know and trust.				
		Understand that computers can automatically turn numbers into different visual formats, making information easier for others to read.	Know that an unbalanced database is slower and harder to use.	Reflect on how much harder, slower or less accurate an experiment would be if done by hand.	Digital tools outperform physical sorting because they allow data to be added, edited and accessed instantly without errors.	Formulas use cell references rather than fixed numbers, allowing calculations to update automatically if the original input data changes.
					Real institutions rely entirely on flat-file structures to operate.	Spreadsheets allow people to run hypothetical scenarios to budget costs and model real-world events safely before spending money. Using structured item catalogues allows an event planner to pick, choose, and calculate budget totals cleanly.