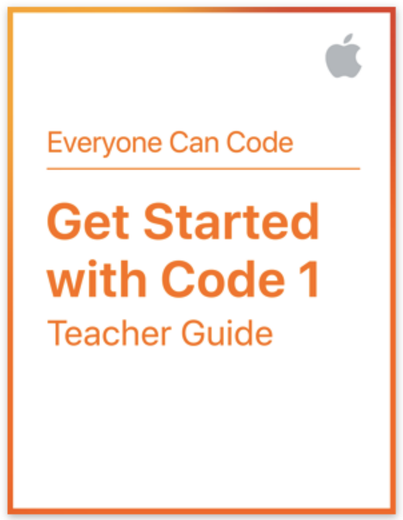
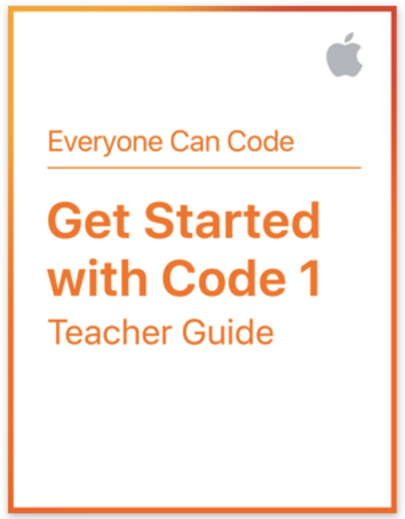
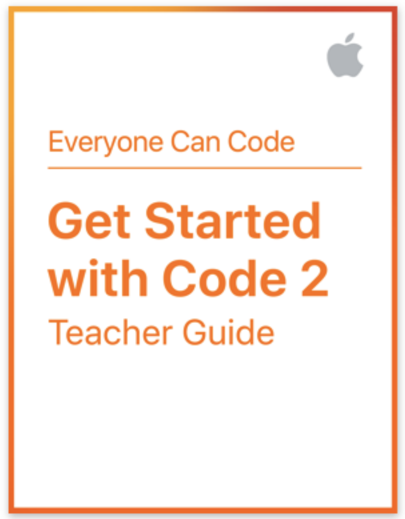
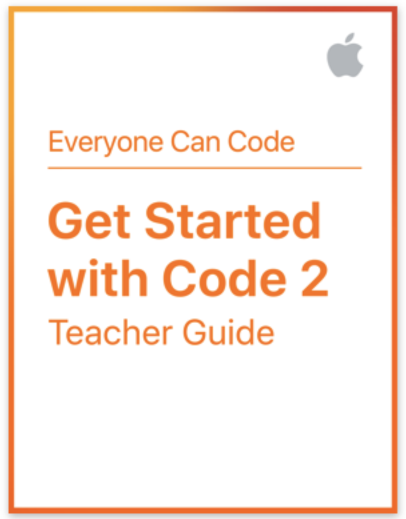
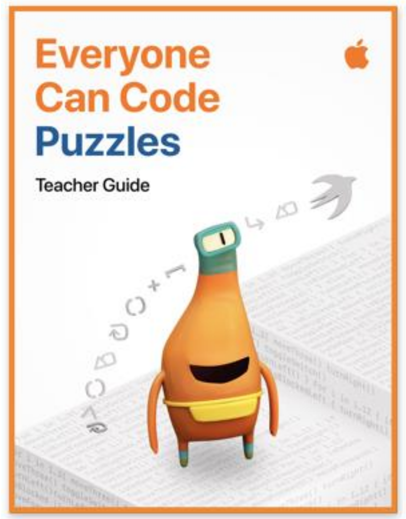
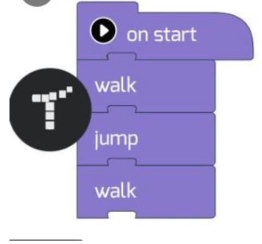
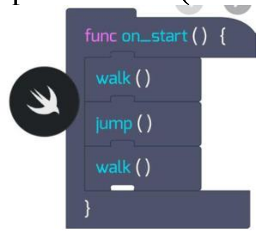
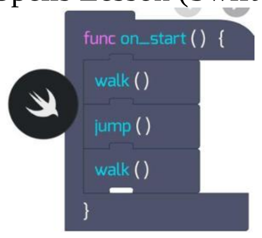
Everyone Can Code: a primary progression for programming

Key Concepts, Skills and Approaches to Programming

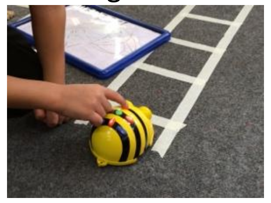
Computational Thinking Skills For Every Lesson

Each lesson from the Everyone Can Code Teacher Guides has an 'unplugged' activity which develops these thinking skills in a real life problem. The second activity applies these thinking skills to coding skills through the use of Codespark, Tynker or Swift Playgrounds software.

LOGICAL REASONING **Predicting and analysing** If you set up two computers in the same way, give them the same instructions (the program) and the same input, you can pretty much guarantee the same output. This means that they are predictable. Because of this we can use logical reasoning to work out why something happens. Part of logical reasoning is the ability to use existing knowledge to make reliable predictions about future behaviour of a system.	**PATTERN SPOTTING** **Spotting and using similarities** Patterns are everywhere, for example, we use weather patterns to create weather forecasts. By identifying patterns we can make predictions, create rules and solve more general problems. Children need to be able to identify repeating patterns in a list of commands to understand how this could be made more efficient using a repeat loop.	**DECOMPOSITION** **Breaking down into parts** The process of breaking down a problem into smaller manageable parts is known as decomposition. Decomposition helps us solve complex problems and manage large projects.
DEBUGGING **Finding and fixing errors** Errors in algorithms and code are called 'bugs', and the process of finding and fixing these is called 'debugging'. Getting pupils to take responsibility for thinking through their algorithms and code, to identify and fix errors is an important part of learning to think and work like a programmer. 1. Predict what should happen. 2. Test -find out -exactly what happens when a program is run 3. Work out where something has gone wrong. 4. Fix it.	**EVALUATING** **Making judgements** Evaluation is about making judgements, in an objective and systematic way where possible. Children need to evaluate the effectiveness of their programs in solving a specific task. Pupils should be encouraged to reflect on the quality of the work that they produce – is it fit for purpose?	**TINKERING** We want to develop in children a willingness to experiment and explore a new app or new software. Children should be encouraged to 'play' with a new piece of software, sharing what they discover about it to one another, rather than always coming to the teacher for the answers. Pupils can explore how to use others' code as a starting point for their own programming projects. Tinkering should help develop independence and perseverance when working with technology.

Key Stage 1	Year 3 & 4	Year 5	Year 6
Teacher Guide:  Everyone Can Code Get Started with Code 1 Teacher Guide	**Teacher Guide:**  Everyone Can Code Get Started with Code 1 Teacher Guide  Everyone Can Code Get Started with Code 2 Teacher Guide	**Teacher Guide:**  Everyone Can Code Get Started with Code 2 Teacher Guide	**Teacher Guide:**  Everyone Can Code Puzzles Teacher Guide
Lesson Sequence: 0 1 2 3 4 5 6	**Lesson Sequence:** Year 3: Get Started With Code 1: Start at Lesson 7, 8 Get Started With Code 2: Lessons 1 2 3 4 Year 4: Get Started With Code 2 Lessons 1 2 3 4 5 6	**Lesson Sequence:** Get Started With Code 2: 7 8 9 10	**Lesson Sequence:** 0 1 2 3 4 5
Apps: Year 1 Codespark: The Foos Create class accounts **Year 2** Tynker, Space Cadet Lessons	**Apps:** **Year 3** Start with Space Cadets, then Dragon Spells Lessons (Regular Blocks)  on start / walk / jump / walk **Year 4** Dragon Spells Lesson (Swift Blocks)  `func on_start() {` `walk()` `jump()` `walk()` `}`	**Apps:** **Year 5** Dragon Spells Lesson (Swift Blocks)  `func on_start() {` `walk()` `jump()` `walk()` `}`	**Apps:** Download the Learn To Code 1 Playground in the app. Child titles the playground with their name and returns to the same iPad each lesson
Teach the same lesson but apply the computational thinking skills	Teach the same lesson in the same app but have Year 3 use Regular Blocks and Year 4 use Swift Blocks in the same level.		

to coding using the year group specific app.			

	FS	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6
SEQUENCING SKILLS	Sequence forwards and turns e.g. with Beebot  **Predict** the outcome of a set of instructions and test the results. Use symbols to represent an instruction e.g. ↑→ for forward and turn. Know how to clear the code **Decompose** by breaking the code down into chunks (1 step at a time) 1) ⬆⬆↩ (clear) 2) ⬆⬆↩ (clear) 3) ⬆⬆↩ (clear) 4) ⬆⬆↩ (clear	**Sequence** commands of forwards, back, left, right using arrow blocks. Understand that a sequence of instructions needs to be clear, precise and unambiguous.	**Sequence** commands including forwards, back and turns more efficiently using blocks. Understand that some steps in a sequence can be reordered but still achieve the same outcome (flexible sequence). Understand that the order in which instructions are given will make a difference to the outcome.	Understand that a sequence of instructions in computing is called an **Algorithm**. Use **decomposition** to break the sequence in to manageable steps. Understand how to approach **debugging** a program or **algorithm**.	Sequence commands in Swift Code blocks Use **abstraction** as a way of making it easier to think about problems. Understand how **functions** help us think more efficiently.	Describe what **commands**, **functions, debugging** and **sequences** are. To read code in Swift Code blocks • Repeat loops • Event handling • Selection Be able to assess success of given instructions and identify and correct any errors that occur.	To sequence an algorithm using written Swift Code. To read **and write** Swift code using: • Repeat loops • Functions • Event handling • Selection • Variables Be able to evaluate the effectiveness of an algorithm written by their peers in class.
RESOURCES		Get Started With Code 1 Use Codespark: The Foos Lesson 1 2 3	Get Started With Code 1 Use Tynker (regular blocks) Lesson 1 2 3	Get Started With Code 2 Tynker (regular blocks) Lesson 1 2 4	Get Started With Code 2 Tynker (Swift blocks) Lesson 1 2 4 5 6	Get Started With Code 2: 7 8 9 10	Puzzles Lesson 1 , 2, 3

REPEAT LOOPS (iteration)		Loop a set of commands by a given amount. Use a number to specify movement rather than repeated commands (e.g. in The Foos enter ↑4 rather than ↑↑↑↑)	Loop a set of commands by a given amount.	Understand what simple **loops** are and how they can make a program more efficient. Identify repeat loops in everyday life	Understand what **loops** are and how they can make a program more efficient. **Pattern spotting** - be able to identify which commands need to be repeated and how many times to achieve a desired end.	Describe what **for loops** are. Use the instruction **repeat until …** Read, write and debug **nested loops** (loops within a loop)	To read and write **loops** in Swift code.
RESOURCES		Get Started With Code 1 Use Codespark: The Foos Lesson 4	Get Started With Code 1 Use Tynker (regular blocks) Lesson 4	Get Started With Code 2 Tynker (regular blocks) Lesson 3	Get Started With Code 2 Tynker (Swift blocks) Lesson 3	Get Started With Code 2 Tynker (Regular Blocks then Swift Blocks) Lesson 8	Puzzles Lesson 3
EVENT HANDLING SKILLS	Know that pressing Go will make the robot move. 	Understand that an **event** is an action that causes something to happen. Sequence an **event** in words ands symbols.	Express an **event** in words and **symbols**.	Be able to create an animation or game using an existing template or scaffold	Be able to create an animation or game using an existing template or scaffold	Be able to create an animation or game **Parallelism** – Allow more than one event to happen at the same time e.g. having more than one set of blocks or instructions running at the same time.	See Sequencing Strand
RESOURCES		Get Started With Code 1 Use Codespark: The Foos Lesson 6	Get Started With Code 1 Use Tynker (regular blocks) Lesson 6	Get Started With Code 1 Use Tynker (Regular Blocks) Lesson 8	Get Started With Code 1 Use Tynker (Swift Blocks) Lesson 8	Get Started With Code 2 Tynker Lesson 10	

CONDITIONAL STATEMENTS SKILLS				Understand that we can make actions occur only under certain conditions. Use IF statements in everyday life and in coding	Understand **conditional statements** as a way of handling different situations (using If, Then, Else commands)	Describe what **Conditionals** are. Read **conditional statements** as Swift code.	Describe what **Conditionals** are. Read and write **conditional statements** as Swift code.
RESOURCES				Get Started With Code **1** Tynker (Regular Blocks) Lesson 7	Get Started With Code **2** Tynker (Swift blocks) Lesson 7	Get Started With Code 2 Tynker (Regular Blocks then Swift Blocks) Lesson 9	Puzzles Lesson 5
VARIABLES SKILLS						Understand **variables** as a way of working with changing values.	Describe what **variables** are and how to use them in Swift code.
RESOURCES						Get Started With Code 2: Lesson 9 – use Tynker and Swift Blocks.	Puzzles Lesson 4
TINKERING OPPORTUNITIES							