**Coding and Computational Thinking – Progression of Skills – Summer 2**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Age Related Expectations – Coding and Computational Thinking** | | | | | | | |
| **EYFS**  **Expectations** | **Year 1**  **Expectations** | **Year 2**  **Expectations** | **Year 3**  **Expectations** | **Year 4**  **Expectations** | **Year 5**  **Expectations** | **Year 6**  **Expectations** | **Beyond Year 6**  **Expectations** |
| I can follow given instructions to program a physical device. | I can explain that an algorithm is a set of **precise** step-by-step instructions to achieve a particular task. | I understand that algorithms are implemented on digital devices as programs and can identify examples of each. | I can make a real-life situation into an algorithm for a program. | I can turn a real-life situation to solve into an algorithm, using a diagram to express solutions. | I can make more complex real-life problems into algorithms for a program. | I can turn a complex programming task into an algorithm. | Learn how to write code using a text- based language (e.g. Python, Java, HTML).  \*Introduced in Y6. |
| I understand what an algorithm is | I know that an alogrithim written for a computer is called a program. | I know I need to carefully plan my algorithm so it will work when I make it into code. | I can design an algorithm carefully, thinking about what I want it to do and how I can turn it into code. | I can use **repetition** in my code. For example, using a **loop** that continues until a condition is met such as the correct answer being entered. | I can test and debug my programs as I work. | I can identify the important aspects of a programming task (abstraction). | Describe different error types (*syntax and logical bugs).* |
| I can demonstrate an ability to following an algorithm. | I can work out what is wrong when the steps are out of order in instructions. | I can design a simple program that achieves a purpose. | I can design a program thinking logically about the sequence of steps required. | I can use **timers** within my program designs more accurately to create repetition effects. | I can convert (translate) algorithms that contain sequence, selection and repetition into code that works. | I can decompose important aspects of a programming task in a logical way, identifying appropriate coding structures that would work. | Uses a range of operators and expressions e.g. Boolean and applies them in the context of program control. |
| I can design simple algorithms. | I can say that if something does not work how it should, it is because my code is incorrect. | I can find and correct some errors in my program ***(debugging).*** | I can experiment with **timers** in my programs. | I can use **selection** *(decision)* in my programming. For example, using an ***‘if statement’*** for a question being asked and the program takes one of two paths. | I can use sequence, selection, repetition, and some other coding structures in my code. | I can test and debug my program as I work on it and use logical methods to identify a cause of a bug. |  |
| I can detect and corrects errors (debugging) in simple algorithms. | I can try and fix my code if it isn’t working properly (debugging). | I can say what will happen in a program. | I can experiment with the effect of using repeat commands. | I can use variables within my program and know how to change the value of variables. | I can organise my code carefully for example, naming variables and using tabs. I know this will help me debug more efficiently. | I can identify a specific line of code that is causing a problem in my program and attempt a fix. |  |
|  | I can make good guesses (logical reasoning) of what is going to happen in a program. For example, where the Bee-Bot might go. | I can spot something in a program that has an action or effect (does something). | I can identify the difference in using the effect of a timer or repeat command in my code. | I can use the user inputs and output features within my program (such as ***‘Print to screen’***). | I can use logical methods to identify the cause of any bug with support to identify the specific line of code. | I can translate algorithms that include sequence, selection and repetition into code and nest these structures within each other. |  |
|  | I know how to give simple instructions one at a time using left, right, forwards and backwards. | I can use repeats and turns within my instructions. | I can identify an error in my program and fix it. | I can identify errors in my code by using different methods, such as steeping through lines of code and fixing them. | I can use **selection** *(decision)* in my programming. For example, using an ***‘if statement’*** for a question being asked and the program takes one of two paths – Building on Y4\* | I can use inputs and outputs within my coded programs such as sound, movement and buttons and represent the state of an object. |  |
|  |  | I can test and change my instructions if needed. | I can read programs with several steps and predict what it will do. | I can read programs that contain several steps and predict the outcomes with increasing accuracy. |  | I can interpret (understand) a program in parts and can make logical attempts to put the separate parts together in an algorithm to explain the program as a whole |  |

\* Children should also **understand and apply the vocabulary related to this strand of the curriculum** for their year group.

Graphical Coding – Hybrid Coding – Leads towards Full Text Coding