**Primary Computing - Progression of Skills and Assessment Profile – Coding and Computational Thinking**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Age Related Expectations** | | | | | | | |
| **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. | Purple MashI can explain that an algorithm is a set of **precise** step-by-step instructions to achieve a particular task.  **(Units 1.4, 1.5, 1.7)** | Purple MashI understand that algorithms are implemented on digital devices as programs and can identify examples of each.  **(Unit 2.1)** | Purple MashI can make a real-life situation into an algorithm for a program.  **(Unit 3.1)** | Purple MashI can turn a real-life situation to solve into an algorithm, using a diagram to express solutions.  **(Units 4.1, 4.5)** | Purple MashI can make more complex real-life problems into algorithms for a program.  **(Unit 5.1)** | Purple MashI can turn a complex programming task into an algorithm.  **(Unit 6.1)** | Learn how to write code using a text- based language (e.g. Python, Java, HTML). |
| I understand what an algorithm is. | Purple MashI know that an algorithm written for a computer is called a program.  **(Units 1.4, 1.7)** | Purple MashI know I need to carefully plan my algorithm so it will work when I make it into code.  **(Unit 2.1)** | Purple MashI can design an algorithm carefully, thinking about what I want it to do and how I can turn it into code.  (**Unit 3.1)** | Purple MashI 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.  **(Unit 4.1)** | Purple MashI can test and debug my programs as I work.  **(Units 5.1, 5.5)** | Purple MashI can identify the important aspects of a programming task (abstraction).  **(Unit 6.1)** | Describe different error types (*syntax and logical bugs).* |
| I can demonstrate an ability to following an algorithm. | Purple MashI can work out what is wrong when the steps are out of order in instructions.  **(Units 1.4, 1.5)** | Purple MashI can design a simple program *(e.g. using 2Code)* that achieves a purpose.  **(Unit 2.1)** | Purple MashI can design a program thinking logically about the sequence of steps required.  **(Unit 3.1)** | Purple MashI can use **timers** within my program designs more accurately to create repetition effects.  **(Unit 4.1)** | Purple MashI can convert (translate) algorithms that contain sequence, selection and repetition into code that works.  **(Unit 5.1)** | Purple MashI can decompose important aspects of a programming task in a logical way, identifying appropriate coding structures that would work.  **(Unit 6.1)** | Uses a range of operators and expressions e.g. Boolean and applies them in the context of program control. |
| I can design simple algorithms. | Purple MashI can say that if something does not work how it should, it is because my code is incorrect.  **(Unit 1.7)** | Purple MashI can find and correct some errors in my program ***(debugging).***  **(Unit 2.1)** | Purple MashI can experiment with **timers** in my programs.  **(Unit 3.1)** | Purple MashI 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.  **(Units 4.1)** | Purple MashI can use sequence, selection, repetition, and some other coding structures in my code.  **(Unit 5.1)** | Purple MashI can test and debug my program as I work on it and use logical methods to identify a cause of a bug.  **(Unit 6.1)** |  |
| I can detect and corrects errors (debugging) in simple algorithms. | Purple MashI can try and fix my code if it isn’t working properly (debugging).  **(Unit 1.7)** | Purple MashI can say what will happen in a program.  **(Unit 2.1)** | Purple MashI can experiment with the effect of using repeat commands.  **(Unit 3.1)** | Purple MashI can use variables within my program and know how to change the value of variables.  **(Unit 4.1)** | Purple MashI can organise my code carefully for example, naming variables and using tabs. I know this will help me debug more efficiently.  **(Unit 5.1)** | Purple MashI can identify a specific line of code that is causing a problem in my program and attempt a fix.  **(Unit 6.1)** |  |
|  | Purple MashI can make good guesses (logical reasoning) of what is going to happen in a program. For example, where the Bee-Bot might go.  **(Units 1.5, 1.7)** | Purple MashI can spot something in a program that has an action or effect (does something).  **(Unit 2.1)** | Purple MashI can identify the difference in using the effect of a timer or repeat command in my code.  **(Unit 3.1)** | Purple MashI can use the user inputs and output features within my program, such as ***‘Print to screen’***.  **(Unit 4.1)** | Purple MashI can use logical methods to identify the cause of any bug with support to identify the specific line of code.  **(Unit 5.1)** | Purple MashI can translate algorithms that include sequence, selection and repetition into code and nest these structures within each other.  **(Unit 6.1)** |  |
|  |  |  | Purple MashI can identify an error in my program and fix it.  **(Unit 3.1)** | Purple MashI can identify errors in my code by using different methods, such as steeping through lines of code and fixing them.  **(Unit 4.1)** |  | Purple MashI can use inputs and outputs within my coded programs such as sound, movement and buttons and represent the state of an object.  **(Units 6.1, 6.7)** |  |
|  |  |  | Purple MashI can read programs with several steps and predict what it will do.  **(Unit 3.1)** | Purple MashI can read programs that contain several steps and predict the outcomes with increasing accuracy.  **(Unit 4.1)** |  | Purple MashI 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.  **(Unit 6.1)** |  |

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

***Skills based on progression documents from 2Simple Software, CAS (Computing at School), Animate 2 Educate Ltd and Simon Haughton Computing***