Digital Programmer

	Exploratory Learner will	Developing Learner will	Confident Learner will	Independent Learner will	Transformational Learner will
Awareness	Experience a variety of simple games, simulations, and apps (including online resources). Express an opinion about a game, simulation or app.	Understand the main differences in using different platforms (PC, mobile, online) and that they work in different ways. Be able to give reasons why they like or dislike a computer program or app. Be able to explain, in simple terms, key vocabulary in programming, including algorithm, program, code/ instruction.	Use a number of different computing platforms with confidence, including PC/Mac, online resources, and mobile platforms (e.g. iPad, Android). Be able to give detailed criticisms of the design (user interface) of a specific program. Be able to explain key vocabulary in programming, and give specific examples.	Be able to critically evaluate the user interface of apps, web services, and computer programmes, using criteria such as ease-of-use, layout, colour and navigation. Understand the need for efficiency, organisation and commenting in computer programmes. Understand key algorithms for common processes in programming; e.g. sorting and searching.	Be able to critique application and interface design with reference to best practice and the 'design language' of major platforms. Be able to criticise the style, logic, structure and layout of existing computer code. Engage with online communities to stay up-to-date with best practice.
Planning	Order simple directional instructions (eg for a Beebot). Contribute to class decisions about the design of a collaborative program or game.	Work independently or with other learners to design simple programs or games to be created in age-appropriate software (Kodu/Scratch) and hardware (e.g. Beebots). Be able to predict the behaviour of a simple program, and to test that prediction.	Work independently, and in teams, to design programs or games in response to a given problem or challenge. Predict the behaviour of a program, giving clear reasons for that prediction.	Use a range of programming and data structures (including lists and arrays) to design and develop modular programs based on procedures or functions.	Explore a variety of programming languages and app platforms and be able to select the best language and tools for a given task. Design programs or apps which solve real-world problems for real users. Give careful consideration to the design of user interfaces.
Skills	Program or control an interactive toy/robot to follow simple instructions.	Create a simple computer program that includes multiple instructions. Test a simple program, notice bugs and make changes to improve it.	Use sequences, selection and repetition in programs. Be able to detect and correct most errors in programs. Simplify a program by breaking it into some smaller parts e.g.	Use two or more programming languages, at least one of which is text-only (e.g. Python, JavaScript) to solve computational problems. Use computer programs to model the state and behaviour	Use and modify code from a variety of sources (online communities, professionals) giving full credit to the creator. Write code which seeks to maximise efficiency and readability.

		Programme and control more sophisticated programmable toys to achieve specific tasks e.g. negotiate a course.	using procedures / functions. Begin to use software to control programmable toys or robots to achieve tasks e.g. to follow a white line.	of real-world problems or systems. Debug, trouble-shoot and optimise programs to improve their performance. Work with others using suitable software to program an interactive robot e.g. Easy C or Flowol 4.	
Sharing	Show other children the result of simple programs they have helped to create.	Describe to other children how they created a program, and what they hoped to achieve.	Share programs within the school community e.g. via the learning platform.	Share programs within and beyond the school community e.g. via a school website.	Distribute code, apps or programs to users beyond the school, using learning platforms, app stores, and online forums.
Feedback / Evaluation	Be able to describe what happened when a set of instructions was run, and whether it achieved the desired outcome. Suggest simple suggestions why a program did not work correctly.	Suggest specific reasons why a program did not work correctly, and work, with support, to solve issues. Give feedback to other children about their programs or games.	Give feedback to other learners about their programs or games, and suggest positive steps or ideas for improvement. Suggest specific reasons why a program did not work correctly, and attempt to debug or troubleshoot.	Give detailed feedback to other learners about their programming, including commentary on the structure, organisation and efficiency of programs.	Seek feedback from online communities about the design of apps or games, and make on-going changes or improvements based on this feedback.