Year 9			***Curriculum Checkpoints: What do students know and what can they do?***			
Computer Science			Developing	Securing	Mastering	Excelling
AF1	Practical Programming Skills	Knowledge	Can define key terms: Inputs, outputs variables, logic and syntax errors. Can identify errors in code.	Can identify and define programming constructs. Know the purpose of arithmetic and comparison operators in Python.	Understands the use of a txt file in python. Knows the purpose of read, write, open and close commands within Python.	Can explain the importance of validation and how it can be used in input sanitation. Can differentiate between a function and procedure.
		Practical Skills	Codes basic inputs, outputs and variables' in Python. Can correct syntax errors in Python code and can use the correct data types when coding.	Implements selection and iteration in Python using arithmetic and comparison operators.	Can read, write, open and close a txt file in Python.	Confidently codes algorithms in Python using correct syntax and logic using functions and regular expressions.
AF2	R094 NEA Visual Identity and Digital Graphics	2.1.1 Computational Thinking	Can define the terms abstraction, decomposition and algorithmic design.	Have a good understanding of abstraction, decomposition and how and why they are used to define and refine problems	Is able to apply computational thinking techniques in real life scenarios.	Apply computational thinking techniques to solve scenarios and use these skills in other aspects of Computer Science.
		2.1.2 Designing, creating and refining algorithms	Knows the purpose of each flowchart symbol. Can define syntax and logic errors. Understands the purpose of a trace table.	Understands where processes, inputs and outputs take place within an algorithm. Can start to trace basic algorithms using a trace table.	Can create flowcharts and pseudocode programs using correct logic based on a scenario. Can independently trace algorithms using a trace table.	Can complete, write and refine complex algorithms using flowcharts, pseudocode and reference language. Can trace a complex algorithm working out the outputs using a trace table.
		2.1.3 Searching and Sorting Algorithms	Understands the main steps of each algorithm.	Understands any pre-requisites of an algorithm. Know the pros and cons of each algorithm.	Can apply searching and sorting algorithms to a data set.	Can confidently implement searching and soring algorithms in a given scenario. Identifies searching and sorting algorithms if given the code or pseudocode for it.
AF3	R094 NEA Visual Identity and Digital Graphics	2.2.1 Programming Fundamentals	Defines the terms variables, constants, inputs and outputs. Can make use of some comparison, arithmetic and Boolean operators. Can define the terms sequence, selection and iteration.	Creates basic programs using inputs, outputs, variables using comparison operators.	Creates programs using inputs, outputs, variables using comparison and arithmetic operators in a given scenario.	Can practically develop programs using the programming constructs as well as confidently implementing comparison, arithmetic and Boolean operators in given scenarios.
		2.2.2 Data Types	Knows the uses of each data type. Can give a suitable example for each data type.	Has to ability to choose suitable data types in a given scenario.	Understands the importance of using data types in programming and can justify why a data type has been used.	Has practical use of the data types in a high-level language within the classroom. Understands the use of casting and can apply this in a given scenario.
		2.2.3 Additional Programming Techniques	Understands the purpose of read, write, open and close commands in Python. Can define what sub-programs are and the types.	Can start to implement read, write, open and close commands in Python. Can discuss some benefits of sub-programs and how the two types differ.	Can independently implement read, write, open and close commands in Python using a txt file. Can discuss a range of benefits of sub-programs and how the two types differ. Can start to implement subprograms in Python. Knows what the SQL commands mean.	Has practical experience of additional programming techniques such as; reading, writing, opening and closing a file, implementing sup-programs in Python. Can confidently use SQL commands in a given scenario.

AF4	2.3 Producing Robust Programs	2.3.1 Defensive Design	Defines the terms authentication and validation with examples. Can define some maintainability techniques.	Justifies why authentication, validation and maintainability techniques are used. Can list and explain a range of maintainability techniques.	Applies authentication, validation and maintainability techniques in a given scenario.	Has practical experience of designing input validation and simple authentication (e.g. username and password). Can implement maintainability techniques in program development.
		2.3.2 Testing	Defines and identifies logic and syntax errors in a program.	Knows the difference between testing modules of a program during development and testing the program at the end of production.	Defines and applies normal, boundary, invalid and erroneous test data in a given scenario.	Confidently identifies suitable test data for a given scenario and create/complete a test plan
AF5	2.4 Boolean Logic	2.4.1 Boolean Logic	Recognises and draws AND, OR NOT gates.	Can recognise the symbol for each logic gate as well as complete the basic truth table.	Understanding of how to create, complete or edit logic diagrams and truth tables for given scenarios	Ability to work with more than one gate in a logic diagram and complete the truth tables of compiled logic gates.
AF6	2.5 Programming Languages and Integrated Development Environments	2.5.1 Languages	Knows the difference between high- and low-level programming languages	Explains the characteristics and purpose of high- and low-level programming languages	Knows the purpose of translators; Compilers and Interpreters	Compares the differences, benefits and drawbacks of using a compiler or an interpreter
		2.5.2 Integrated Development Environments	Can list at least three Integrated Development Environment tools.	Can list and explain a range of Integrated Development Environment tools.	Knows how each of the tools and facilities listed can be used to help a programmer develop a program.	Has practical experience of using a range of these tools within Python.