Topic: Unit 2.2 Programming Fundamentals

Key vocabulary:			**Core knowledge questions**	**Powerful knowledge crucial to commit to long term memory**	**Links to previous and future topics**
Input Output Assignment Pseudocode Flowchart Identifier Variable CONSTANT Concatenation RAM Memory Locations Arithmetic Boolean String Substring Delimiter SPLIT Left Mid Right Addition + Subtraction - Division / Multiplication Modular Division Integer Division Negation AND OR NOT Equals = Greater Than > Less Than < Equal to == Not Euqal to != Modulus % Quotient // Exponentiation **	Sequence/Algorithm Instructions Efficiency IF ELSE IF ELSE SELECT CASE Boolean Nested Selection Condition Criteria True / False While For Counter Condition Iteration Break Nested Iteration Exit Condition Stepping Increment Decrement Larger/Greater Character Escape Character EOF – end of file EOL – end of line New Line Character File Mode Commit/Append Overwrite Tabulation/Table Serialise Deserialise	Delimeter SQL Index Elements/Items Dimensions Non-Sequential Zero-Based Empty Array Plural Naming Overflow NULL Random Out of Index Error Column Instructions Efficiency Subprograms Procedures Functions Structured Variable Scope Local Variable Global Variable Parameter Calling/Triggering Abstraction Top Down Approach Integer Real Boolean Casting Data Types Character Sets ASCII Unicode	1. How to use Variables, Constants, Operators, Inputs, Outputs and Assignments 2. How to use the three basic programming constructs used to control the flow of a program: • Sequence • Selection • Iteration 3. Know the common Arithmetic Operators 4. Know the common Boolean Operators 5. The correct use of Data Types: • Integer • Real • Boolean • Character / String • Casting 6. The use of basic string manipulation 7. The use of basic file handling operations: • Open • Read • Write • Close 8. The use of records to store data 9. The use of SQL to search for data 10. The use of arrays when solving problems (1D and 2D) 11. How to use sub programs (functions and procedures) to produce structured code 12. Random Number Generation	• Practical use of key programming techniques in Python • Understanding of each technique • Recognise and use relevant operators: ○ == Equal to ○ != Not Equal to ○ < Less Than ○ <= Less Than or Equal To ○ > Greater Than ○ >= Great Than or Equal To ○ + Addition ○ - Subtraction ○ * Multiplication ○ / Division ○ MOD % Modulus ○ DIV // Quotient ○ ^ ** Exponentiation (Power of) • Practical use of the data types in Python • Ability to choose suitable data types for data in a given scenario • Understand that data types may be temporarily changed through casting, and where this may be useful • Practical use of the additional programming techniques in Python • Ability to manipulate strings including: Concatenation & Slicing • Arrays as fixed length static structures • The use of Functions and Procedures and where to use them effectively • SQL commands: SELECT; FROM; WHERE	• Programming using Python and Scratch is covered in Y7 and Y8 • Programming is fundamental to Computer Science and will be revisited throughout the course.

We will develop these skills:

Impressive reading	**Impressive speaking**	**Impressive writing**	**Resilience**	**Numeracy via:**	**Digital Literacy via:**	**Employability via:**
Research using the Internet to find relevant and appropriate	Discussion of research findings.	Recording research findings appropriately. Writing definitions.	Developing ability to consistently amend and refine work.	Use of a variety of data types	Use of the Internet	Teamwork – working in groups Flexibility – taking on opinions of others

information about the fundamentals of programming.	Discussion in groups of programming fundamentals Demonstration of practical activities	Answering exam questions	Listen to others' opinions	Use of arithmetic operators within programs	Use of MS Office Suite Use of Python	Problem Solving – deciphering algorithms for conversion to working programs
SEND						
• Graduated learning tasks (complexity of concepts and programs) • Peer Support - Some students may have more experience of programming – use these students as Lead Students • Differentiated Activities and Tasks, choice of tasks for certain activities, support sheets • Questioning • Flipped Learning resources for students to study either prior to or after lesson • Peer Assessment / Support on programming tasks • Provision of scaffolded scripts to allow learners to experience successful completion of working programs						