Order of delivery	11 - 2.1 Algorithms	11 - 2.2 Programing Techniques	11 - 2.3 Producing rfobust programs	11 - 2.4 Computational Logic	11 - 2.5 Translators and Facilities of Languages	11- 2.6 Data Representation
Key Questions	2.1 What are the 3 main components of computational thinking? What is a binary search? What is a linear search? What is a bubble sort? What is a merge sort? What is an insertion sort? What is pseudocode? What is a flowdiagram?	2.2. What is the difference between a variabke and a constant? What is sequence/selection/iteration? What are the 4 main file handling operations? What is SQL? What is an array? What is the difference between a function and a procedure? What are the main data types? What is casting?	2.3 What are the 4 main defensive design considerations? What are the 2 main areas of maintainability? What is the purpose of testing? What types of testing exist? What is the difference between a logic error and a syntax error?	2.4 Why is data represented in binary form? What is a logic diagram? What is a truth table? What are the 7 most common mathematical symbols?	2.5 What are the different levels of programming language? What is a translator? What is an assembler? What is a compiler? What is an interpreter? What are common tools and facilities available in a IDE?	2.6 What are the main units of measurement? How we convert from binary to denary and vice versa? What is a binary shift? How do we convert a denary value to a hexadecimal value and vice versa? How do we add binary numbers? What is a check digit? How does a computer understand a letter or character? What is a character set? How does a computer store an image? What is metadata? How is sound stored as binary? What is compression and why do we apply it?
Knowledge	2.1 To know that abstraction, decomposition & algorithmic thinking are components of computationl thinking. To know how a binary search works. To know how a linear search works. To know how a bubble sort works. To know how a merge sort works. To know how an insertion sort works. To know how to produce a pseudocode. To know how to produce a flow diagram.	2.2 To know that a constant is a variable that doesn't change. To know when to use sequence/selection/iteration to solve a problem. To know that open, read, write and close are the 4 main file handling operations. To know that SQL is used to query databases to extract specific data. To know that an array is like a list of values. To know that a function returns a value and a sub program can be called from the main program. to know that integer, real, Boolean, characte and string are data types.	2.3. To know that input sanitisation/validation, planning for contingencies, anticipating misuse and authentication are the 4 min defensive design considerations. To know that comments and indentation are the 2 main ways of applying maintainability. To know the purpose of testing is to identify any errors in a program. To know what iterative and final/terminal testing are. To know that syntax error (punctuation etc.) will produce an error whereas a logic error won't but it will still mean the output will be wrong.	2.4. To know that switches/logic gates are used to control the flow of electricity in a processor and a 1 represents a pulse and a 0 doesn't. To know that logic diagrams show a representation of the switches in a CPU. To know that truth tables are used to identify the output of a program. To know that exponentiation (square of a number) MOD (is the remainder of a number) and DIV (returns the whole number of a division but not the remainder?.	2.5 To know that there are high level and low level programming languages. To know that a translator changes programming code to machine code. To know that an assembler is a program for converting instructions written in low-level symbolic code into machine code. To know that a compiler is a special program that processes statements written in a particular programming language and turns them into machine language. To know that an interpreter is a program that can analyse and execute a program line by line. To know that editors, error diagnostics, run time environments and translators are the common tools used in an IDE.	2.6 To know what a bit, nibble, byte, kilobyte, megabyte, gigabytle, terabyte, petabyte are. To know how to convert a binary value to a denary value and vice versa. To know that a binary shift is when binary values move to the left or right based on 1 or 2 places. To know how to convert a hexadecimal number to a denary number and vice versa. To know how to add binary numbers together. To know how to apply a check digit. To know that all characters are represented by a unique binary code. to know what a character set is. to know that ASCII, Extened ASCII and Unicode are character sets. To know that an image is stored as a series of binary values. To know that metadata is data about the data e.g author, dimensions, date. To know that sound is stored as binary based on the number of sampling intervals. To know that compression reduces the original file size. to know that lossy loses some data and lossless doesn't lose any of the original data.

Skills	2.1 To be able to apply compuational thinking to a rel life scenario. This scenrio will be emulated through the production of a Python program. To be able to explain how many steps are required to find a specific value using a linear or a binary search. To be able to identify the number of passes to carry out a bubble, merge and insertion sort. To be able to complete a sort from given values. To be able to produce pseudocode for a given scenario. To be able to produce a flow diagram for a given scenario.	2.2 Be able to make a program that includes a variable and a constant. Be able to use to write a program that shows appropriate use of sequence /selection/iteration. To be able to clearly demonstrate theuse of the 4 main file handling operations in a Python program. To be able to enter SQL statements e.g. in SQL Zoo to extract specific data. To be able to add and manipulate data/values in an array. To be able to write a program that uses a function and a sub program. To be able to use the correct data types in a program to produce accurate rsults. to be able to carry out casting when appropriate in a program.	2.3 To be able to clearly explain the 4 defensive design considerations from to a relevant question on the topic. To be able to clearly explain how comments and indentation are relevant to maintainability. To be able to create a test plan that will ensure errors are located/avoided in a program. to be able to clearly explain using a relevany exam question the difference between iterative and final/terminal testing methods. To able to identify a logic and a syntax error from a given question.	2.4 To be aale to expain why computers use binary. To be able to identify the output from a series of linked logic gates. To be able to use a truth table to ientify the output of a program. To be able to use all 7 mathematical operators in a variety of scenarios.	2.5 To be able to explain the difference between a low level and a high level programming language. To be able to explain what a translator does. To be able to explain the difference between an assembler, compiler and interpreter. To be able to clearly explain the 4 main tools in an IDE.	2.6 To be able to convert bits to bytes or relevant. To be able to identify the number of bits in a given unit of data. To be able to convert a binary number to a denary number and vice versa when given examples to solve. To be able to apply a given binary shift to an example to give the correct answer. To be able to convert a hexadecimal number to a denary number and vice versa when given examples to solve. To be abke to add binary numbers together when goven solutions to solve. To be able to determine the result when a check digit is given to determine if the data is corect. to be abe to give the correct binary code when a character is given. To be able to explain the difference between ASCII, Extened ASCII and Unicode. To be able to produce a small representation of an image and how each pixel is either a 1 or a 0. To be able to explain what metadata is. To be able to explain how sound is stored. to be able to explain what compression is and the advantages and disadvantages of using lossy and lossless compression.
Vocab	abstraction, decomposition, algorithmic thinking, binary search, linear search, bubble sort, merge sort, insertion sort, pseudocode, flow	constant, variable, sequence, selection, iteration, SQL, array, Boolean	input sanitisation, planning for contingencies, maintainability, comments, indentation, iterative, logic error, syntax error	binary, logic gates, truth tables,exponentiation, MOD, DIV	translator, assembler, compiler, interpreter, IDE	bit, nibble, byte, kilobyte, megabyte, gigabytle, terabyte, petabyte, binary shift, check digit, ASCII, Extended SCII, Unicode, pixel, colour depth, resolution, sampling interval, compression, lossless,
Assessment	Exam Style Assessment using real world exam questions and a range of teacher created resources. Reinforced using practical programming examples	Exam Style Assessment using real world exam questions and a range of teacher created resources. Reinforced using practical programming examples	Exam Style Assessment using real world exam questions and a range of teacher created resources. Reinforced using practical programming examples	Exam Style Assessment using real world exam questions and a range of teacher created resources. Reinforced using practical programming examples	Exam Style Assessment using real world exam questions and a range of teacher created resources. Reinforced using practical programming examples	End of Unit 2.6 assessment. Exam type assessment.