How Fire Tech Python Courses cover the coding component of GCSE Computing Qualifications

AQA GCSE Computer Science and ICT Specification 8525			Teen Coding with Python 1	Teen Coding with Python 2
3.2 Programming	**3.2.1** Data Types	Understand the concept of a data type Understand and use the following appropriately: ~ integer ~ real ~ Boolean ~ character ~ string	☑	☑
3.2 Programming	**3.2.2** Programming concepts	Use, understand and know how the following statement types can be combined in programs: ~ variable declaration ~ constant declaration ~ assignment ~ iteration ~ selection ~ subroutine (procedure / function)	☑	☑
3.2 Programming	**3.2.2** Programming concepts	Use definite (count controlled) and indefinite (condition controlled) iteration, including indefinite iteration with the condition(s) at the start or the end of the iterative structure.	☑	☑
3.2 Programming	**3.2.2** Programming concepts	Use nested selection and nested iteration structures.	☑	☑
3.2 Programming	**3.2.2** Programming concepts	Use meaningful identifier names and know why it is important to use them.	☑	☑
3.2 Programming	**3.2.3** Arithmetic operations in a programming language	Be familiar with and be able to use: ~ addition ~ subtraction ~ multiplication ~ real division ~ integer division, including remainders.	☑	☑
3.2 Programming	**3.2.4** Relational operations in a programming language	Be familiar with and be able to use: ~ equal to ~ not equal to ~ less than ~ greater than ~ less than or equal to ~ greater than or equal to.	☑	☑
3.2 Programming	**3.2.5** Boolean operations in a programming language	Be familiar with and be able to use: ~ NOT ~ AND ~ OR	☑	☑
3.2 Programming	**3.2.6** Data structures	Understand the concept of data structures	☑	☑
3.2 Programming	**3.2.6** Data structures	Use arrays (or equivalent) in the design of solutions to simple problems.	☑	☑
3.2 Programming	**3.2.6** Data structures	Use records (or equivalent) in the design of solutions to simple problems.	☑	☑
3.2 Programming	**3.2.7** Input/output	Be able to obtain user input from the keyboard	☑	☑
3.2 Programming	**3.2.7** Input/output	Be able to output data and information from a program to the computer display.	☑	☑
3.2 Programming	**3.2.8** String handling operations in a programming language	Understand and be able to use: ~ length ~ position ~ substring ~ concatenation ~ convert character to character code ~ convert characetr code to character ~ string conversion operations	☑	☑
3.2 Programming	**3.2.9** Random number generation in a programming language	Be able to use random number generation.	☑	☑
3.2 Programming		Understand the concept of subroutines	☑	☑
3.2 Programming		Explain the advantages of using subroutines in programs	☑	☑

Pearson Edexcel GCSE Computer Science Specification 1CP2			Teen Coding with Python 1	Teen Coding with Python 2
6.1 Develop code	**6.1.1**	be able to use decomposition and abstraction to analyse, understand and solve problems	☑	☑
6.1 Develop code	**6.1.2**	be able to read, write, analyse and refine programs written in a high-level programming language	☑	☑
6.1 Develop code	**6.1.3**	be able to convert algorithms (flowcharts, pseudocode*) into programs	☐	☐
6.1 Develop code	**6.1.4**	be able to use techniques (layout, indentation, comments, meaningful identifiers, white space) to make programs easier to read, understand and maintain	☑	☑
6.1 Develop code	**6.1.5**	be able to identify, locate and correct program errors (logic, syntax, runtime)	☑	☑
6.1 Develop code	**6.1.6**	be able to use logical reasoning and test data to evaluate a program's fitness for purpose and efficiency (number of compares, number of passes through a loop, use of memory)	☐	☑
6.2 Constructs	**6.2.1**	understand the function of and be able to identify the structural components of programs (constants, variables, initialisation and assignment statements, command sequences, selection, repetition, iteration, data structures, subprograms, parameters, input/output)	☑	☑
6.2 Constructs	**6.2.2**	be able to write programs that make appropriate use of sequencing, selection, repetition (count-controlled, condition-controlled), iteration (over every item in a data structure) and single entry/exit points from code blocks and subprograms	☑	☑
6.3 Data types and structures	**6.3.1**	be able to write programs that make appropriate use of primitive data types (integer, real, Boolean, char) and one and two dimensional structured data types (string, array, record)	☑	☑
6.3 Data types and structures	**6.3.2**	be able to write programs that make appropriate use of variables and constants	☑	☑
6.3 Data types and structures	**6.3.3**	be able to write programs that manipulate strings (length, position, substrings, case conversion)	☑	☑
6.4 Input/output	**6.4.1**	be able to write programs that accept and respond appropriately to user input	☑	☑
6.4 Input/output	**6.4.2**	be able to write programs that read from and write to comma separated value text files	☑	☑
6.4 Input/output	**6.4.3**	understand the need for and be able to write programs that implement validation (length check, presence check, range check, pattern check)	☐	☐
6.4 Input/output	**6.4.4**	understand the need for and be able to write programs that implement authentication (ID and password, lookup)	☐	☐
6.5 Operators	**6.5.1**	be able to write programs that use arithmetic operators (addition, subtraction, division, multiplication, modulus, integer division, exponentiation)	☑	☑
6.5 Operators	**6.5.2**	be able to write programs that use relational operators (equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to)	☑	☑

	3.2.10 Structured programming and subroutines (procedures and functions)	Describe the use of parameters to pass data within programs.	☑	☑
		Use subroutines that return values to the calling routine.	☑	☑
		Know that subroutines may declare their own variables, called local variables, an that local variables usually: ~ only exist while the subroutine is executing ~ are only accessible within the subroutine	☑	☑
		Use local variable and explain why it is good practice to do so	☐	☐
		Describe the structured approach to programming	☐	☐
		Explain the advantages of the structured approach	☐	☐

	6.5.3	be able to write programs that use logical operators (AND, OR, NOT)	☑	☑
6.6 Subprograms	**6.6.1**	be able to write programs that use pre-existing (built-in, library) and user-devised subprograms (procedures, functions)	☑	☑
	6.6.2	be able to write functions that may or may not take parameters but must return values, and procedures that may or may not take parameters but do not return values	☑	☑
	6.6.3	understand the difference between and be able to write programs that make appropriate use of global and local variables	☑	☑

OCR GCSE Computer Science Specification J277			**Teen Coding with Python 1**	**Teen Coding with Python 2**
2.2 Programming fundamentals	**2.2.1** Programming fundamentals	The use of variables, constants, operators, inputs, outputs and assignments	☑	☑
		The use of the three basic programming constructs used to control the flow of a program: o Sequence o Selection o Iteration (count- and condition-controlled loops)	☑	☑
		The common arithmetic operators	☑	☑
		The common Boolean operators AND, OR and NOT	☑	☑
	2.2.2 Data Types	The use of data types: o Integer o Real o Boolean o Character and string o Casting	☑	☑
	2.2.3 Additional programming techniques	The use of basic string manipulation	☑	☑
		The use of basic file handling operations: o Open o Read o Write o Close	☑	☑
		The use of records to store data	☑	☑
		The use of SQL to search for data	☐	☐
		The use of arrays (or equivalent) when solving problems, including both one-dimensional (1D) and two-dimensional arrays (2D)	☑	☑
		How to use sub programs (functions and procedures) to produce structured code	☑	☑
		Random number generation	☑	☑
	2.3.1 Defensive Design	Defensive design considerations: o Anticipating misuse o Authentication	☐	☑
		Input validation	☐	☑
		Maintainability: o Use of sub programs o Naming conventions o Indentation o Commenting	☑	☑
	2.3.2 Testing	The purpose of testing	☐	☑
		Types of testing: o Iterative o Final/terminal	☐	☐
		Identify syntax and logic errors	☑	☑
		Selecting and using suitable test data: o Normal o Boundary o Invalid/Erroneous	☐	☐

WJEC Eduqas GCSE in COMPUTER SCIENCE			**Teen Coding with Python 1**	**Teen Coding with Python 2**
Component 2	Investigation	use a systematic approach to problem solving including the use of decomposition and abstraction	☑	☑
		use abstraction effectively to model selected aspects of the external world in an algorithm or program	☑	☑
		use abstraction effectively to appropriately structure programs into modular parts with clear, well-documented interfaces	☑	☑
		analyse a set of requirements	☑	☑
		program a solution that meets a set of requirements.	☑	☑
	Design	design and document the input and output facilities required to produce an effective user interface	☐	☐
		design and document all required data structures	☐	☐
		using pseudo code, design and document the following routines: • validation and verification • data handling and processing • authentication.	☐	☐
	Implementatio n design, write, test and refine Python 3 code using the following skills:	create new and extend existing functions or methods	☑	☑
		create new and edit existing objects	☑	☑
		create new and extend existing Python 3 libraries	☑	☑
		use variables (labels), operators, inputs, outputs and assignment	☑	☑
		use a variety of data types, including integer, Boolean, real, character and string	☑	☑
		use programming constructs to control the flow of a program, including: • iteration (condition and counter controlled loops) • selection • sequence	☑	☑
		use basic file handling, including: • open a file • read from a file into a variable • read from a file into an array • write to a file • write to a file from an array • close a file	☑	☑
		create new and extend data structures and fixed length records to store data	☐	☑
		use string manipulation	☑	☑
		create new and extend lists, tuples and dictionaries (arrays)	☐	☑
		use slicing	☐	☑

		Refining algorithms	☐	☐

		use mathematical and logical operations	✓	✓
		create new and modify existing intuitive graphical user interfaces using the Python 3 built in libraries including: · text boxes · buttons · forms	☐	☐
		create and modify data validation and verification routines	☐	☐
		create and modify authentication management routines	☐	☐
		create code for the solution that is self-documenting and uses meaningful identifiers	✓	✓
		use a programming style that is consistent, including indentation and appropriate use of white space	✓	✓
		use subroutines with well-defined interfaces	☐	☐
		annotate code so that it is accessible to a competent third party	✓	✓
		explain the solution or changes made to a solution.	✓	✓
	Testing	design and document an effective testing strategy that will ensure that the final solution meets the requirements	✓	✓
		describe how the outcomes of the testing process can be used to inform further development of the solution	☐	☐
		design test data to include examples of typical, extreme and erroneous content	☐	☐
		implement a test plan using typical, extreme and erroneous data where appropriate	☐	☐
		present test outcomes with detailed and informed commentaries	☐	☐
		demonstrate testing and refinement of code during development or in response to change in the requirements provided	✓	✓
		explain the outcome of testing using given data or data that the candidate has designed.	☐	☐