KS3 Progress Grid Key Stage 3 Computing **YEAR 9**

	Greater than expected progress.		Expected progress		Less than expected progress	
	Students will be able to know / understand / do:		Students will be able to know / understand / do:		Students will be able to know / understand / do:	
	Mid-year	**End of year**	**Mid-year**	**End of year**	**Mid-year**	**End of Year**
High prior attainment	Understand how instructions can be written efficiently and be able to describe the efficiency of your programs. Be able to test the different modules of your programs as you are developing them, reflect on the results and then improve them. Be able to write programs in a test-based language, like Python ad be able to create your own data structures. Be able to create a simple model for a complex problem Be able to define an outline of a solution in terms of functions and global values.	Be able to show how elements of real life can be represented in programs and the difficulties that sometimes exist when doing this. Make sure that the programs you develop have been written so they are unlikely to crash or cause errors Be able to create your own relational databases and use them in your programs and be able to find, understand and use techniques for specific tasks. Be able to create an accurate detailed model for a complex problem. Be able to analyse real world problems and develop low-level and high-level plans for a solution	Understand how instructions are run inside a computer Be able to develop solutions for problems that are described to you by someone else Correctly use procedures and functions with parameters in your programs Be able to take solutions to one problem and adapt them for similar problems Be able to take a problem and divide it into all its sub-problems and show this is a diagram	Understand how instructions can be written efficiently and be able to describe the efficiency of your programs. Be able to test the different modules of your programs as you are developing them, reflect on the results and then improve them. Be able to write programs in a test-based language, like Python ad be able to create your own data structures. Be able to create a simple model for a complex problem Be able to define an outline of a solution in terms of functions and global values.	Be able to explain why we must be accurate when working with computers Write sequences of instructions and data in a way that a computer will understand. Use selection and repetition correctly in your programs Correctly use variables, lists and simple procedures in your programs. Be able to recognise similarities between simple problems and the ways in which they can be solved. Be able to take a problem and divide it into its main sub-problems	Understand how data, such as numbers, sounds and images are physically stored on a computer system. Be able to plan, create, test and reflect on a solution to a problem that a computer could solve. Correctly use variables, lists and simple procedures in your programs. Be able to recognise similarities between simple problems and the ways in which they can be solved. Be able to take a problem and divide it into its main sub-problems
Middle prior attainment	Understand how instructions are run inside a computer Be able to develop solutions for problems that are described to you by someone else Correctly use procedures and functions with parameters in your programs Be able to take solutions to one problem and adapt them for similar problems Be able to take a problem and divide it into all its sub-problems and show this is a diagram	Understand how instructions can be written efficiently and be able to describe the efficiency of your programs. Be able to test the different modules of your programs as you are developing them, reflect on the results and then improve them. Be able to write programs in a test-based language, like Python ad be able to create your own data structures. Be able to create a simple model for a complex problem Be able to define an outline of a solution in terms of functions and global values.	Understand how data, such as numbers, sounds and images are physically stored on a computer system. Be able to plan, create, test and reflect on a solution to a problem that a computer could solve. Correctly use variables, lists and simple procedures in your programs. Be able to recognise similarities between simple problems ad the ways in which they can be solved. Be able to take a problem and divide it into its main sub-problems	Understand how instructions are run inside a computer Be able to develop solutions for problems that are described to you by someone else Correctly use procedures and functions with parameters in your programs Be able to take solutions to one problem and adapt them for similar problems Be able to take a problem and divide it into all its sub-problems and show this is a diagram	Be able to explain why we must be accurate when working with computers Write sequences of instructions and data in a way that a computer will understand. Use selection and repetition correctly in your programs Be able to trace instructions using variables, selection and repetition and predict what the results will be. Understand what is meant by a computational problem.	Be able to explain why we must be accurate when working with computers Write sequences of instructions and data in a way that a computer will understand. Use selection and repetition correctly in your programs Correctly use variables, lists and simple procedures in your programs. Be able to recognise similarities between simple problems and the ways in which they can be solved. Be able to take a problem and divide it into its main sub-problems
Low prior attainment	Understand how data, such as numbers, sounds and images are physically stored on a computer system. Be able to plan, create, test and reflect on a solution to a problem that a computer could solve. Correctly use variables, lists and simple procedures in your programs. Be able to recognise similarities between simple problems ad the ways in which they can be solved. Be able to take a problem and divide it into its main sub-problems	Understand how instructions are run inside a computer Be able to develop solutions for problems that are described to you by someone else Correctly use procedures and functions with parameters in your programs Be able to take solutions to one problem and adapt them for similar problems Be able to take a problem and divide it into all its sub-problems and show this is a diagram	Be able to explain why we must be accurate when working with computers Write sequences of instructions and data in a way that a computer will understand. Use selection and repetition correctly in your programs Correctly use variables, lists and simple procedures in your programs. Be able to recognise similarities between simple problems and the ways in which they can be solved. Be able to take a problem and divide it into its main sub-problems	Understand how data, such as numbers, sounds and images are physically stored on a computer system. Be able to plan, create, test and reflect on a solution to a problem that a computer could solve. Correctly use variables, lists and simple procedures in your programs. Be able to recognise similarities between simple problems ad the ways in which they can be solved. Be able to take a problem and divide it into its main sub-problems	Understand that computer systems work step-by-step and can only do what we tell them Be able to create a sequence of instructions and improve it if necessary. Use selection and repetition correctly in your programs Be able to trace instructions using variables, selection and repetition and predict what the results will be. Understand what is meant by a computational problem.	Be able to explain why we must be accurate when working with computers Write sequences of instructions and data in a way that a computer will understand. Use selection and repetition correctly in your programs Be able to trace instructions using variables, selection and repetition and predict what the results will be. Understand what is meant by a computational problem.