# Programming Glossary

| Term | Definition |
|---|---|
| Algorithm | A sequence of steps to perform a task. |
| Flowchart | A graphical representation of an algorithm that uses flow lines and shapes to represent the operations. |
| Pseudocode | Pseudocode looks a bit like a programming language but, unlike a real programming language, it does not require a strict syntax. This makes it particularly useful for designing programs.<br>Because it is not an actual programming language, pseudocode cannot be compiled into executable code. |
| Structured English | Similar to pseudocode, structured English is a way of describing an algorithm using English words to describe each step of the process. |
| Program code | The set of instructions that form a program written in a particular programming language. |
| | |
| Data type | The classification of data as being a string, integer, float or Boolean (logical). |
| Boolean | Boolean or logical data that can only have one of two values, i.e. True or False (1 = True and 0 = False). |
| Float | A number with a fractional part, e.g. 34.67. Also known as a real number. |
| Integer | A whole number, e.g. 45. |
| String | A sequence of characters that can include letters, spaces, symbols, e.g. “This is a name £££46 “ |
| | |
| Variable | A named location in a computer’s memory where data is stored. |
| Global variable | A variable assignment that is available throughout the whole program. |
| Local variable | A variable assignment that is only available within a subprogram. |
| Type declaration | When a variable is allocated a particular data type (integer, string, Boolean, float). |
| Variable declaration | When a value is assigned to a variable. |
| Constant | A variable assignment that stays the same throughout a program. |
| Data structure | A group of related data items. Examples include strings, arrays and lists. |
| | |
| Command sequence | A block of program commands that are executed sequentially, i.e. one after the other. |
| Loop | A piece of code that keeps repeating until a certain condition is reached. |
| Output | Data that is sent out from a program to a screen, a file or a printer. |
| Indentation | Positioning a block of code, e.g. a for loop or an if statement, further from the margin than the main program.<br>The use of indentation makes programs easier to read.<br>In Python correct indentation is important, since indenting starts a block and unindenting ends it. |

| Term | Definition |
|---|---|
| Comment | Text included in code that is for the human reader and is ignored by the program.<br>In Python, comments are indicated by the # symbol at the start of a line. |
| Structural components | The parts of a program such as variable declarations, data structures or subprograms. |
| | |
| Subprogram | A small computer program that runs within another computer program. Subprograms are used to split up a program into a number of smaller programs, with each subprogram performing a specific function. Subprograms can be called in any order any number of times. |
| Function | A subprogram that returns a value. |
| Procedure | A subprogram that does not return a value. |
| Return value | The value returned by a subprogram. |
| Parameter | The names that appear in a function definition when passing data to a function. |
| Argument | A piece of information/value that is required by a function to perform a task, e.g. function(argument1, argument2) |
| Built-in subprograms | Pre-existing libraries of subprograms that are built into the programming language. |
| Library subprograms | Pre-existing libraries of sub-programs that can be imported and used in the programming language. |
| | |
| Mathematical operators | Mathematical operators take two operands and perform a calculation on them.<br>Examples of mathematical operators are: ‘+’ (addition), ‘-‘ (subtraction), ‘*’ (multiplication), ‘/’ (division). |
| Precedence | The order in which values are calculated. |
| Integer division | Division in which the remainder is discarded, e.g. 10//3 = 3. |
| Modulus | The absolute (non-negative) value of a number. The modulus of -3 is 3. |
| Relational operators | Relational operators take two operands and compare them.<br>Examples of relational operators are: < (less than) < (greater than), == (equal to), != (not equal to).<br>Relational operators are used in conditional statements, especially in loops, where the result of the comparison decides whether execution should proceed. |
| Logical operators | Logical operators take two operands and compare them, returning True or False depending on the outcome.<br>Examples of logical operators are: AND, OR, NOT. |
| | |
| Logic error | An error in the design of a program, such as the use of a wrong programming control structure or the wrong logic in a condition statement.<br>This type of error may not produce an error message, just the wrong outcome. |
| Runtime error | An error detected during program execution, often due to a mistake in the algorithm or in the type of data used. |

| Term | Definition |
|---|---|
| Syntax error | An error that occurs when a program statement cannot be understood because it does not follow the rules of the programming language. |
| Trace table | A technique used to test for logical errors in a program. Each column of the table is used to represent a variable and each row to show the values of the variables at different stages of the program. |
| | |
| Break point | A point in the code when a program is halted so that the programmer can investigate the values of variables to help locate errors. |
| Bug | An error in the code that prevents a program from running properly, or from running at all. |
| Debugger | A tool that helps to detect, locate and correct faults (or bugs) in programs. |
| Single step | When a program is executed one statement at a time under user control. |
| Watchers | Allow the programmer to watch for certain events, such as variables changing as the program is running. |
| | |
| Test plan | A list of tests to be carried out, designed to test a program in the widest possible range of situations to make sure it is working. |
| Test data | Data used in a test plan to test what happens when the input data is valid, invalid or extreme. |
| Valid test data | Test data that is within the normal range and should be accepted by the program. |
| Invalid test data | Test data that is outside the normal range and should be rejected by the program. |
| Extreme test data | Test data that is still valid, but is on the absolute limits of the normal range and should be accepted by the program. |
| | |
| Textual user interface | A type of interface in which the user interacts with the computer program by using a keyboard to enter commands and make selections. |
| Graphical User Interface (GUI) | A type of user interface that uses windows, icons, buttons and menus. A pointer is used to make selections. |
| Validation | Validation is designed to ensure that a program only operates on appropriate data, e.g. that it has an appropriate format or value. It involves checking the data that a user enters or that is read in from a file and rejecting data that does not meet specified requirements.<br>However, validation can only prove that the data entered is appropriate, not that it is correct. |
| | |
| Integrated development environment (IDE) | A text editor with useful built in tools to help programmers develop programs, e.g. IDLE. |
| | |
| Cartesian co-ordinates | A co-ordinate system that specifies a point uniquely using (x,y) co-ordinates. |