Unit 2: Problem solving

Key terms

Term	Definition
Computational thinking	Methods and techniques for analysing a problem aimed at producing a solution that can be developed as a computer program.
Decomposition	The technique of breaking down a large program into a series of more manageable sub-problems.
Pattern recognition	Identifying how similar problems have been solved previously.
Abstraction	The reduction of a sub-problem to its simplest set of characteristics that are most relevant to producing a solution.
Algorithm	A step-by-step design of a solution to a sub-problem, or the rules to follow to solve the sub-problem.
Variables	Memory locations used to store data that can change during program execution. Variables should be given identifiers (names) to reflect the data being stored.
Constants	Memory locations used to store data that does not change during program execution. Constants should be given identifiers, in UPPERCASE, to reflect the data being stored.

"Programmers can only write programs for computers to solve problems if they understand how to solve the problem themselves."

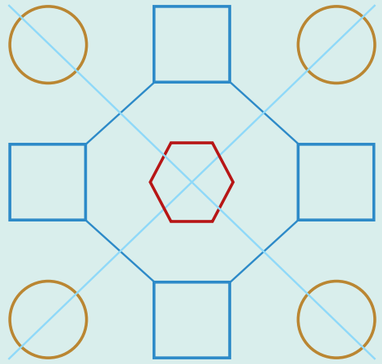
Computational thinking

1. DECOMPOSITION

A problem that seemed confusing will make a lot more sense when broken down into smaller steps that need to be taken. This can help to decide what to do first and allows for a systematic approach.

2. PATTERN RECOGNITION

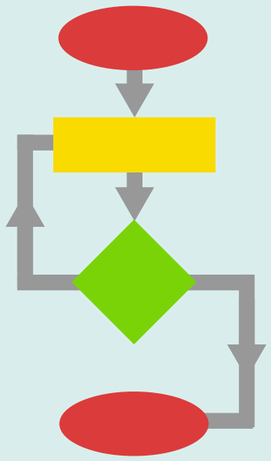
Recognising patterns in complex problems can help to solve them more efficiently. If a problem requires the same steps to be taken over and over again, it makes sense to identify that early on so that the process of solving them becomes more manageable.



3. ABSTRACTION

Abstraction is about looking at the problem and the patterns and working out what is important and what can be ignored. Unnecessary details are removed so that a model can be formed that will allow a solution to be created.

4. ALGORITHM



The algorithm is the list of steps that need to be taken to solve the problem. Used to tell computers what to do, it can also be a useful approach to use with humans.

It is the explanation of input steps to be taken, in the order they need to be taken. This will produce the desired output result.

Variables

A variable is a holder for an item of data (that can be changed) of a specific type and length. Data types include character, integer, real and Boolean.

Variables must be defined in a program by specifying type and length and assigning a self-identifying name, indicative of the data stored.

Variables are assigned a memory location when the program is loaded into memory. The data in a variable may be created, edited and deleted whilst the program is running.

Local and global variables

A variable declared in a sub-procedure has **'local scope'** because it can only be accessed from within that sub-procedure.

A variable declared in the main program has **'global scope'** because it can be accessed from all parts of the program.

Static and dynamic variables

Each time a program is run, **static variables** are stored in a location in memory and have a lifespan that lasts the entire time that program is running.

Each time a **dynamic variable** is declared, it is assigned a new location in memory and has a lifespan that ends when the sub-procedure ends.

Constants

A constant is a holder for an item of data that will not be changed whilst the program is running.

Constants should be declared in the main program by specifying a value and assigning a self-identifying name. Any change in the value will involve editing the value in the source code.