# Topic 1: Computational Thinking

## 1 Computational Thinking

**Introduction**

The widespread use of computers has changed the way we solve problems:

1. We can get tech to do the hard work for us. We can formulate our problems to harness the speed and power of computers
2. Formulating a computer based solution for a problem itself clarifies our understanding of a problem. We might not otherwise realise that a problem can be broken down into simpler parts.
3. Understanding how computers store and process data can provide us with powerful analogies for understanding how the world works.

- One of the most powerful benefits of a computational approach is that it encourages us not to be frightened of more complex problems as they can be decomposed.

**Decomposition:** The breaking down of a problem into smaller parts that are easier to solve. The smaller parts can often be solved recursively; that is they can be run again and again until that part of the problem is solved.

**Computational Thinking:** A problem solving approach that borrows techniques from Computer Science, notably abstraction, problem decomposition and the development of algorithms. Computational thinking is applied to a wide variety of problem domains and not just the development of computer systems.

**Structured Programming**

- In structured programming procedures (functions) are packaged off and designed to perform one (or a small amount) of jobs which improves readability.

- It has been shown that any function can be made with three constructs:

1. SEQUENCE - executing one statement after another.
2. SELETION - branching to another place in a program according to a boolean expression.
3. ITERATION - repeating a section of code until a boolean statement is true.

**Object-Orientated Programming:** A program made up of objects (custom made data structures that represent often-used real world entities) that interact.

**Algorithm:** A well-defined set of steps that act on some or a set of values as input and produces another value of set of values as output.

## 2 Elements Of Computational Thinking

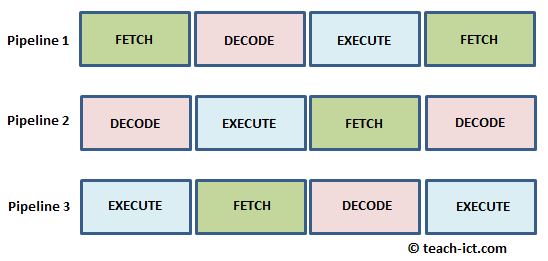
**Computability:** Whether or not a problem can be solved using an algorithm.

**Backtracking:** Algorithmic approach to a problem where partial solutions to a large problem are incrementally built up as a pathway to follow. If the pathway fails the search begins again at the last potentially successful point.

**Data Mining:** Process for trawling through a lot of data to find relationships that may not be immediately obvious.

**Performance Modelling:** Modelling systems before implementing them in the real world to save money, time and increase safety.

**Pipelining:** Situation where the output of one process is used as the input for another. Useful for processors where you can separate the FDE cycle into different pipelines so that your processor is never idle.



**Visualisation:** Visualising data so that is easy to grasp.

## **Ways Of Thinking**

**Thinking Abstractly**

- An abstraction is a concept of reality, it uses simplification (such as symbols) to represent the components of a problem that is easier to understand.

- Teases out what does and what does not matter in a scenario.

**Levels Of Abstraction:**

- It is often useful to construct an abstraction to represent a large problem and to create lower-level abstractions to deal with component parts.

- The powerful part of this is that details in each layer are hidden from other layers so that solving the problem is focused on what’s important at each layer and resources are used efficiently.

**Thinking Ahead**

- Planning is really important in computer systems. You need to know before hand what inputs and outputs are needed and what algorithms are needed to process things in your system.

**Caching:** Storing data that is likely to be used soon or often in memory to avoid fetching from main storage multiple times.

**Pre-fetching:** Instruction or data is fetched from memory before it is required to speed up instruction through-put.

**Re-usability:** Know what code has already been written in libraries/previous projects and writing your code in a way such that minimal code is repeated and minimal time is wasted.

**Thinking Procedurally**

- Knowing and thinking about the order in which things are happening.

**Thinking Logically**

- Understanding the decision points in a problem and how the conditions work in these decision points.

**Thinking Concurrently**

- Often it is possible for different parts of a problem to be solved at the same time as it can speed up the solution.

- Thinking concurrently means understanding what processes can and are happening simultaneously.