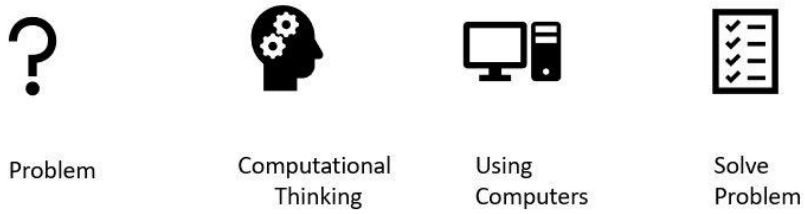
1.1.1 Computational Thinking

Computational Thinking is in fact a thought process, rather than a specific study area on a particular device or language. It is a universally applicable attitude and a set of skills achievable by anybody, not just computer scientists.



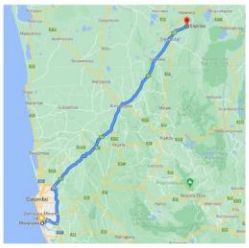
Any problem that seems difficult or unsolvable can easily be tackled by breaking it down into manageable pieces. Each of these subproblems can be dealt with individually by referring to how similar situations have been handled previously solving a complex problem that seemed unsolvable.

The 4 major components of computational thinking,

- **Decomposition** - break down complex problems into smaller, simpler subproblems.
- **Pattern recognition** - make connections between similar problems and experiences
- **Abstraction** - Make one solution applicable to multiple situations by ignoring unrelated, irrelevant information.
- **Algorithm Design** - Design simple achievable steps to realize a solution.

For example, let's say you want to go on a trip to Sigiriya a very interesting place to visit. The ancient rock fortress was built by a Sri Lankan King. For some planning the trip could be a problem. When do you go? How do you go? With whom do you go ? What do you do there? Where do you eat? All these are questions that need to be answered before you go. Lets see how we can use computational thinking to solve these problems.

Going on A Trip : A Problem ?



Let's look at each of the components of computational thinking and also how we can use Computational Thinking for a example problem.

1. Decomposition

When a problem is decomposed, or broken down into manageable pieces, it is much easier to solve. The problem can be more closely examined in further details, making finding a solutions an easier task.

In our example we can break the trip up into 3 parts rather than trying to think of the entire trip at once.

Rather than trying to think about what we are going to do for the entire day, we can break it down in to morning, noon, and afternoon. Another way is distance, the first 70 Km the next 70Km and the final 62 Km.

Also we can break it down into sub parts of what food to eat what sites to visit and what transports to use.

When breaking down the problem we can even further breakdown the sub problems to make them easier to solve. Breaking a bigger problem into smaller problems for the purpose of solvingthe problem is decomposition.

In each sub problems we can go into further details of the specific things that we need to solve and concentrate on solving each part. Once we complete solving all the parts we have the solution for the entire problem.

2. Pattern Recognition

Pattern recognition is the analysis of similar objects or experiences and the identification of commonalities. By finding what these objects or experiences have in common, we can develop an understanding of patterns and make predictions based on them.

Once a problem is decomposed, we can find patterns among sub problems we created, and use prior experience and knowledge to solve each subproblem.

If we take the same example of solving the problem of where to visit in our trip to Sigiriya we can check to see if we have gone to Sigiriya before, and we went last time. Are there other people who have gone to Sigiriya? Where did they stop along the way? Rather than looking at all the interesting places in Sri Lanka we can look at where others have gone and based on their experiences slightly modify what they have done to suit ourselves.

If we wish would decide on a time to leave we can look at the time we left on the last trip. The last rip many have not been to Sigiriya. What about the time we went to Kandy or Badulla? Was it early in the morning or later in the day? May be earlier in the day is better time to leave because of less traffic and more time to spend in the places we visited.

What mode of transport did we use? Was it public transport or a private bus? We may not even realize patterns of the past and solutions pattens we used that can solve problems in the present.

3. Abstraction

Abstraction is focusing on relevant, important information in identified patterns. It involves removing specific details and making generalizations. Abstraction helps create a general idea of the problem and how to solve it.

In our example, the entire trip can be looked upon as going from point A to point B without going into details about what cities we pass on the way.

Also, abstraction helps us produce a common solution that not only works for the problem at hand, but related problems as well. For example, if we want to go to Matara next year we can follow a similar itinerary as the trip to Sigiriya.

4. Algorithm Design

An algorithm is a stepwise approach to solve a problem. It involves creating a process or set of rules to be followed. It should also be noted that there can be more than one way to solve a problem.

In daily life we follow algorithms to achieve tasks at hand. From tying shoelaces to making tea to preparing dinner, we follow step by step processes to achieve each task.

In our example when we go on the trip we will visit multiple sites. So what do we do there? Coming up with a set of instructions on what to do would be an algorithm.

When we visit a site we may want to share our itinerary like first take a head count , then get down from the vehicle, visit the site, stop for refreshments, visit the gift shop, return to the vehicle and finally take a headcount before departing. The collection of theses steps is an algorithm. We can use the same set of steps or algorithm for all the sites that we visit.

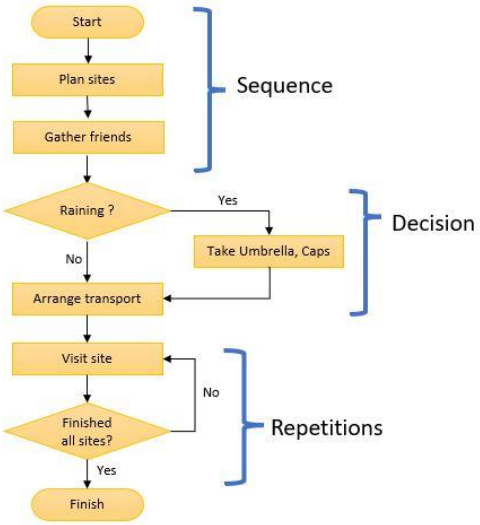
Algorithms can be presented as flowcharts or pseudocodes, specifying each step precisely.

Flow charts and Pseudocode

Flowcharts are pictorial representation of algorithms, while pseudocodes are more of high level, textual representations.

We can also think of the problem of scheduling the trip as an algorithm.

We plan the sites. We collect all the people going on the trip. We check to see if it is raining. We stay in the bus. We take our umbrellas and caps. If it is not a rainy day we don't take them. We find a vehicle and visit the sites. Until we have sites we repeat and finish all the sites. This algorithm can be presented as a flow chart.



We can also represent the solution as pseudocode. Pseudocode is a way of writing a solution to a problem in an informal way so we remember how we solved the problem and for other people to understand how we solved the problem. Pseudocode is not a programming language but we can easily convert pseudocode to programming languages like Python or Java.

In Pseudocode

```
BEGIN
PLAN SITES
GATHER FRIENDS
IF RAINING
   TAKE UMBRELLA, CAPS
ARRANGE TRANSPORT
WHILE THERE ARE SITES
   VISIT NEXT SITE
FINISH
```

Now that we have a way to solve problems and a way to represent them. so others can also solve them. we can think of machines solving problems for us. we can build a machines to take each step and do what we ask it to do.

1.1.2 Computer Systems

History of Computing

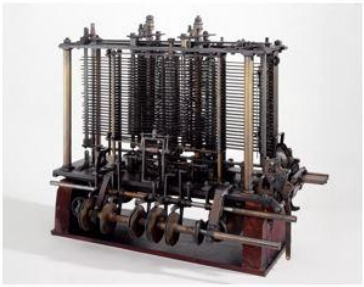
In earlier times, computers were only used for simple calculations. The Abacus is considered as one of the first computing devices ever invented.

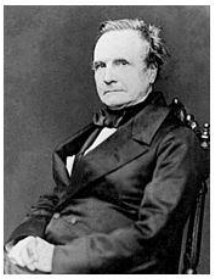
An abacus uses several rows of beads to represent numbers.

With time, automating calculations helped accomplish more advanced and tedious tasks.

In 1642, a French Mathematician named Blaise Pascal invented the first mechanical calculating machine, the Pascaline to perform addition and subtraction.

Later during the early 1800s, British Mathematician Charles Babbage originated the concept of programmable computers. He invented the Difference Engine a special purpose machine , to compile mathematical tables. Later he proposed the Analytical Engine, the first general purpose mechanical computer. Babbage is distinguished as the Father of Computing due to the recognizable inventions he did at the time.





Ada Augustus Lovelace, the first computer programmer in the world, worked with Babbage on his designs and inventions. She wrote the first program while translating a paper on Babbage's Analytical Engine from French into English. She had mentioned stepwise descriptions, an algorithm, to calculate Bernoulli numbers an the Analytical Engine.

The early programming language Ada was named after her, recognizing her contribution to the computing world.

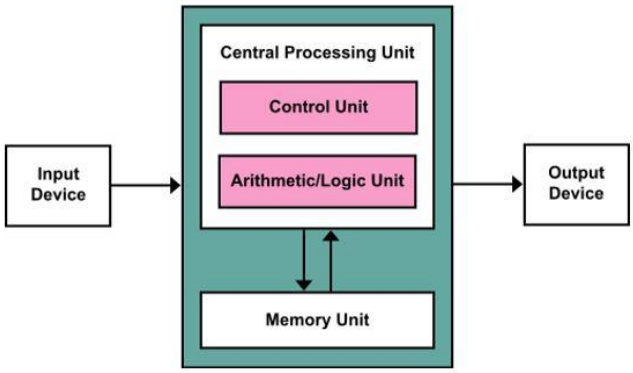
In 1945, the first automatic, electronic, general purpose, digital and programmable computer was invented by two professors at the University of Pennsylvania, John Mauchly and J. Presper Ecker. It could solve numerical problems through reprogramming. The computer was named ENIAC, Electronic Numerical Integrator and Computer.

Components of a Computer

The design of most of the General Purpose computers are based on the **Von Neumann Architecture** (the Princeton Model). The concept was presented by John von Neumann, a Hungarian-American Mathematician in 1945.

Von Neumann Architecture is based on the stored-program concept in which the program and data are stored in accessible memory.

This model introduces key components of the Computer as, the Central Processing Unit where control is centralized in a single component, a storage, primary memory where both data and instructions reside, and input/output devices.



The CPU executes instructions once they have been fetched into it from the main memory. There is a built-in addressing procedure for the main Memory , which the CPU refers to whenever instructions and data are fetched. The execution of these instructions is done in a sequential manner. The CPU also contains the Arithmetic and Logic Unit which is capable of arithmetic and logical operations.

The Processor comprises several key registers including,

- Accumulator, where the results of ALU calculations are stored .
- Program Counter (PC), keeps track of the address of the next instruction to be fetched and executed after the current instruction.
- Memory Address Register (MAR), stores the memory locations of instructions that will be fetched from memory or sent and stored in memory.
- Memory Data Register (MDR),stores instructions fetched from memory or any data that is to be sent to, and stored in, memory.
- Current Instruction Register (CIR),stores the most recently fetched instructions .

And more.

Programs or data can be read into main memory from the input devices or secondary storage. And the Output devices are to output the information from a computer.