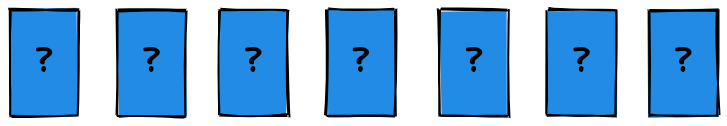
***Data structure and Algorithm in Python***

* *Why we should learn Data structures and Algorithms?*
* *To solve programming problems like a linked list or balancing a binary tree*
* *In software development, use it for solving programming problems like-*

1. *We can* ***think about a problem systematically*** *and solve it systematically step-by-step.*
2. ***Can envision different inputs, output, and edge cases*** *for program we write.*
3. *Most importantly, we* ***can convert our thoughts and ideas into a working code*** *that is also readable.*

* *A systematic strategy we shall apply for solving problems :---*
* *State the problem clearly. Identify the input & output formats.*
* *Come up with some example inputs, outputs. Try to cover all edges cases.*
* *Come up with a correct solution for the problems .state in plain English.*
* *Analyze the algorithm’s complexity and identify inefficiencies, if any.*
* Problems

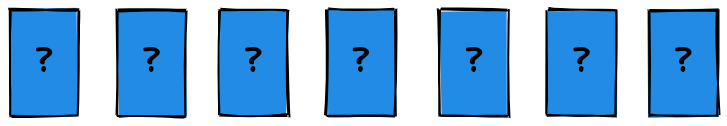
***QUESTION 1:****Alice has some cards with numbers written on them. She arranges the cards in decreasing order, and lays them out face down in a sequence on a table. She challenges Bob to pick out the card containing a given number by turning over as few cards as possible. Write a function to help Bob locate the card.*



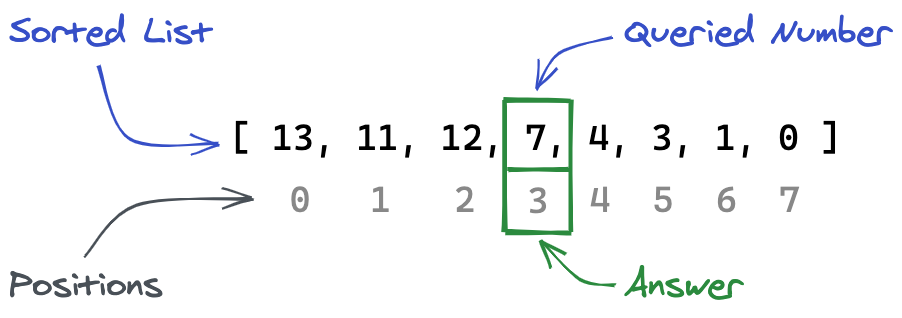
* Solution

**1. State the problem clearly. Identify the input & output formats.**

You will often encounter detailed word problems in coding challenges and interviews. The first step is to state the problem clearly and precisely in abstract terms.



In this case, for instance, we can represent the sequence of cards as a list of numbers. Turning over a specific card is equivalent to accessing the value of the number at the corresponding position the list.



* ***Problem***

We need to write a program to find the position of a given number in a list of numbers arranged in decreasing order. We also need to minimize the number of times we access elements from the list.

* ***Input***

1. Cards: A list of numbers sorted in decreasing order. E.g. [13, 11, 10, 7, 4, 3, 1, 0]
2. Query: A number, whose position in the array is to be determined. E.g. 7

* Output

Position: The position of query in the list cards. E.g. 3 in the above case the

above case (counting from 0).

* The \_space complexity\_ is some constant `c'` (independent of `N`), since we just need a single variable `position` to iterate through the array, and it occupies a constant space in the computer's memory (RAM).

> \*\*Big O Notation\*\*: Worst-case complexity is often expressed using the Big O notation. In the Big O, we drop fixed constants and lower powers of variables to capture the trend of relationship between the size of the input and the complexity of the algorithm i.e. if the complexity of the algorithm is `cN^3 + dN^2 + eN + f`, in the Big O notation it is expressed as \*\*O(N^3)\*\*

* Thus, the time complexity of linear search is \*\*O(N)\*\* and its space complexity is \*\*O(1)\*\*.