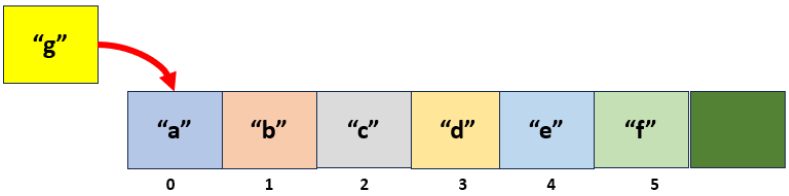
DAILY DSA | DAY-2 | ARRAYS – PART2|
-GOPALKRISHNA A

We went through the concept, types, representation, and memory footage.
We will go through the **Array operations**.

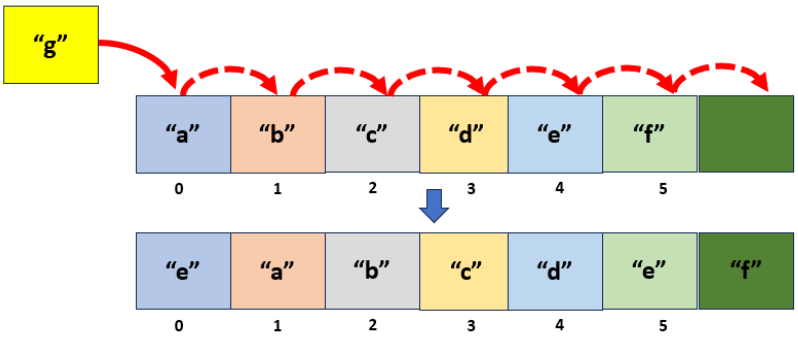
1. **Insert:** We can insert one or more items into the array at the start, end, or any given index.

   Syntax: array_Name.insert(index, value)

   We want to insert an element at the beginning of the array (index 0)

   We move all the elements located to the right of the insertion index, one index to the right. Element "a" will now be at index 1 and "b" will be at index 2

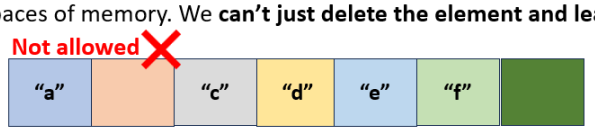
   

2. **Deletion:** For deletion, we can delete an item from an array by its value. After the removal of the element array items are re-arranged and indices are re-assigned.

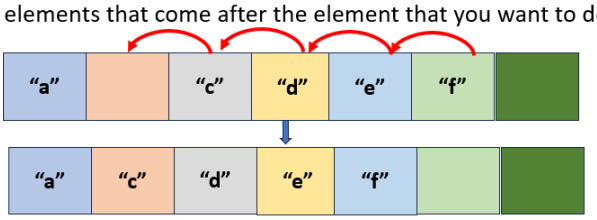
   Syntax: array_Name.remove(value)

We want to delete an element from the array (element "b")

To maintain the efficiency of accessing element in a faster way, elements must be stored in contiguous spaces of memory. We **can't just delete the element and leave the space empty**



We move the elements that come after the element that you want to delete one index the left



3. **Search:** This method is non-destructive i.e. it has no effect on the array values. In this case, we can search an item in the array by its value.

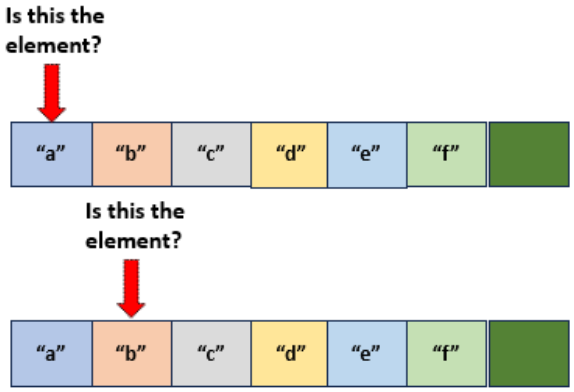
   Syntax: array_Name.index(value)

   Three options to find an element in an array:

   If you know where it's located, use the index

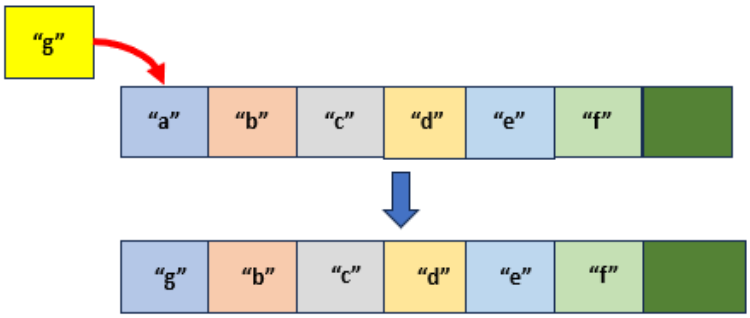
   If you don't know where it's located & data is sorted, use sorting algorithms, BST

   if you don't know where it's located and your data is not sorted, need to search through the element in the array and check if the current element is the element you are looking for



4. **Update:** This method works similarly to the insertion operation, except that we are replacing an existing value at the giving index with a new one.

   Syntax: arrayName.update(index, value)



Summary:

- **Arrays are extremely powerful data structures** that store elements of the same type. The type of elements and the size of the array are fixed and defined when you create it
- **Memory is allocated immediately** after the array is created and it's empty until values are assigned.
- **Elements are located in contiguous locations in memory**, so they can be accessed very efficiently (O(1) = constant time) using indices
- **Indices start at 0**, not 1

Practice problems:

Find the difference between Arrays: Write a function diff(arrA, arrB) that accepts two arrays and returns a new array that contains all values that are not contained in both input arrays. The order of numbers in the result array does not matter.

Find the duplicates: Given an array a[] of size N which contains elements from 0 to N-1, you need to find all the elements occurring more than once in the given array