

---

---

# **Data Structures Using Python**

---

---

# What are Data Structures and Algorithm?

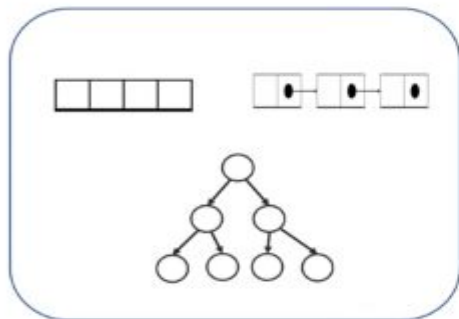
**Data:** Data is a collection of values

**Structure:** It is the way of organizing.

**Algorithms:** A step by step procedure or formula for solving a problem.



+ Home building instructions =



+ Code instructions =



**Data structures and Algorithm:** There are essential components that help organize and store data efficiently in computer memory. They provide a way to manage and manipulate data effectively, enabling faster access.

Data structures and algorithms make it simple for people to find and deal with the information they need.

# Why to learn?

## 1. Problem-Solving Skills:

- Think logically and solve problems methodically
- Choose the right approach to solve a problem efficiently
- Write optimized and efficient code that performs well

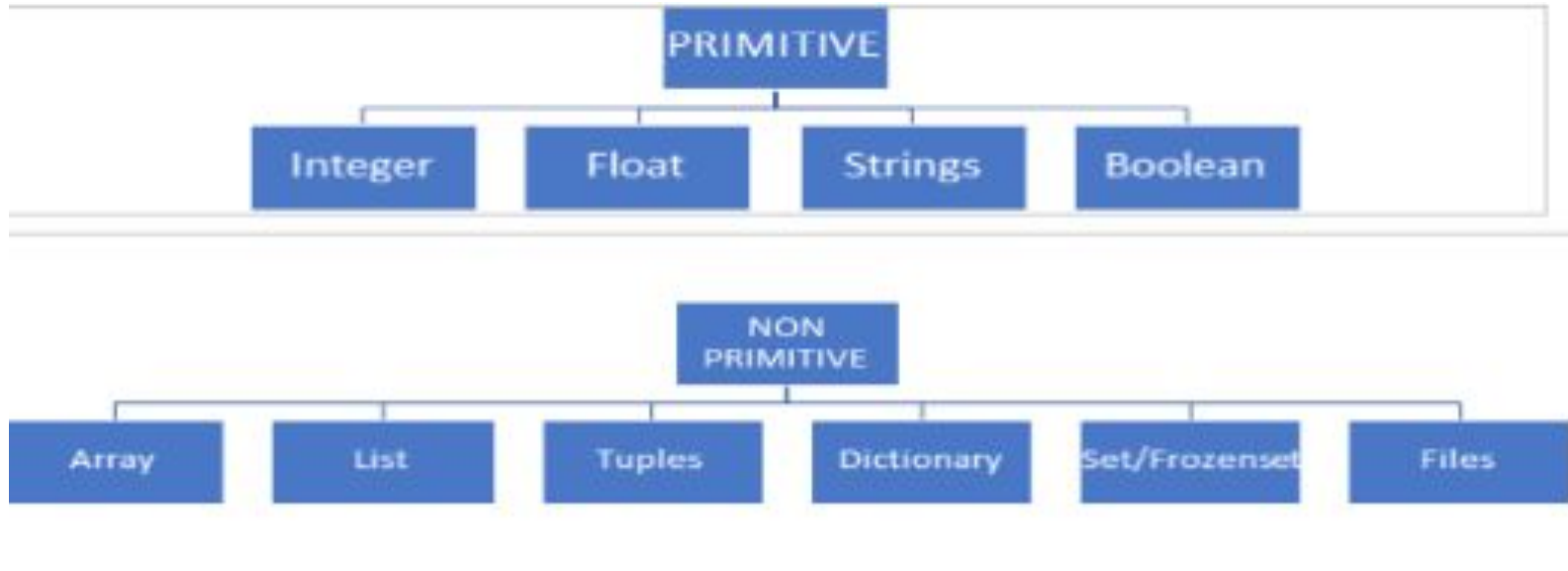
**2. Building Applications:** Many real-world applications require efficient data processing. For example, search engines, social media platforms, and financial systems rely heavily on efficient data structures and algorithms.

**3. Interview Preparation:** Many technical job interviews focus heavily on DSA concepts. Companies like Google, Amazon, Facebook, and Microsoft often ask candidates to solve problems that require a good understanding of data structures and algorithms.

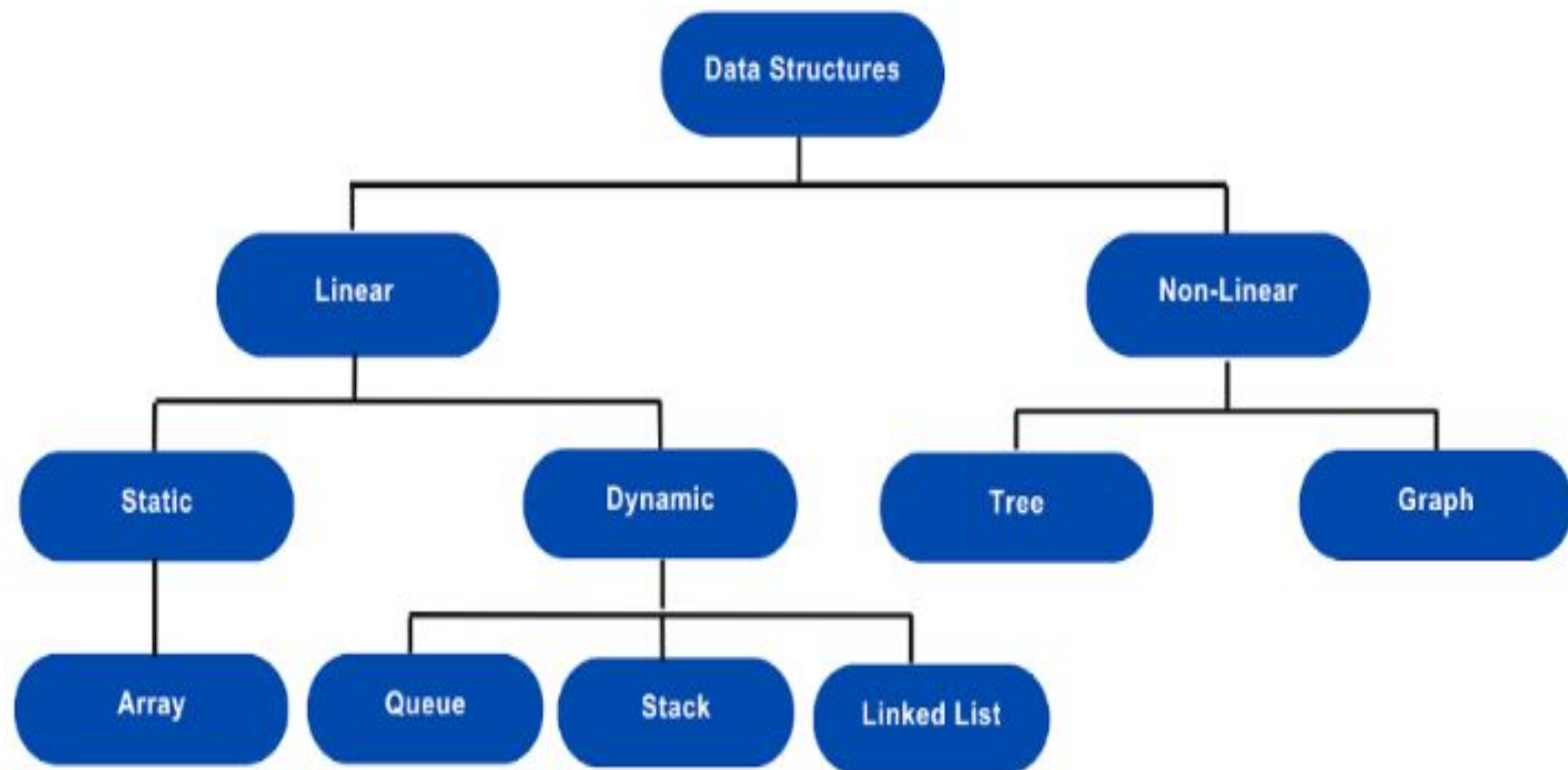
**4. Performance:** Efficient data structures and algorithms are essential for writing code that runs quickly and uses resources (memory, CPU) effectively. This is crucial for applications that need to scale or handle large volumes of data.

# Types of Data Structures

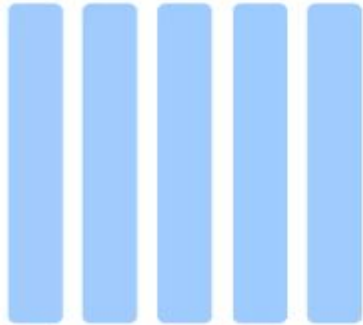
1. **Primitive** : If any keyword (Pre-Define) in the form of data type is called primitive structure.
2. **Non Primitive** : Non primitive structure are those that are derived from primitive.



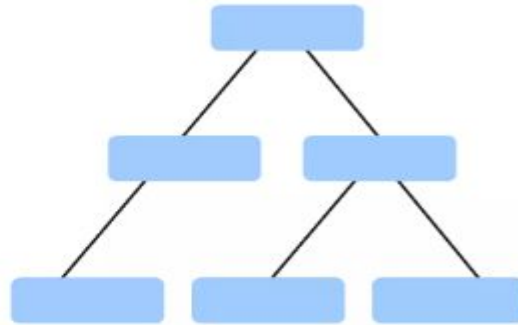
# DATA STRUCTURES CLASSIFICATION



# Linear and Nonlinear



LINEAR DATA STRUCTURE



NON- LINEAR DATA STRUCTURE



# Most common Data structures

1. Array
2. String
3. Linkedlist
4. dictionaries/hashmap
5. stack/queue
6. Trees
7. Graphs

# Array Data Structure

An array is a linear data structure. It is a collection of items of same data type(int, float, boolean, string)

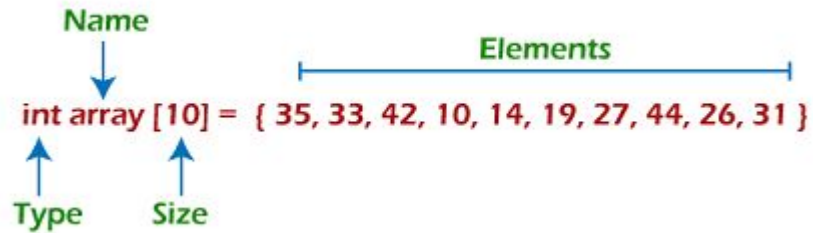
It hold upto N items

.

Memory Address	→	2391	2392	2393	2394	2395
Array Values	→	12	34	68	77	43
Array Index	→	0	1	2	3	4

# Why we use arrays?

To store large number of elements under a single variable name.



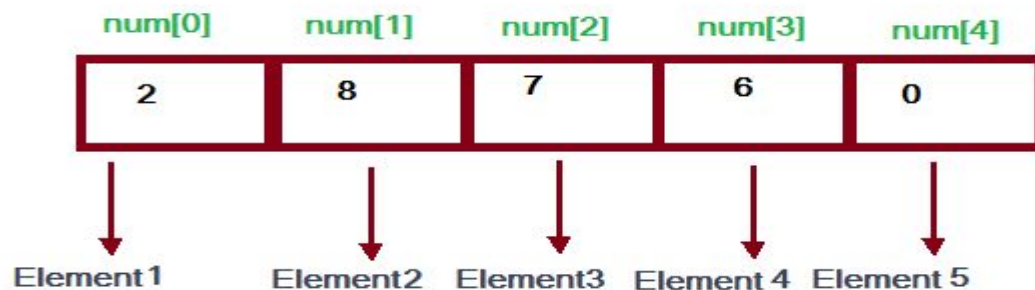
The diagram illustrates the components of the array declaration `int array [10] = { 35, 33, 42, 10, 14, 19, 27, 44, 26, 31 }`. Annotations include: 'Name' pointing to 'array', 'Type' pointing to 'int', 'Size' pointing to '[10]', and 'Elements' spanning the list of values in curly braces.

```
int array [10] = { 35, 33, 42, 10, 14, 19, 27, 44, 26, 31 }
```

# How to access elements in array?

Every element of an array can be accessed through its index.

Index of array always starts with 0 and ends at N-1.



# Array Capacity VS Array Length

Number of element an array can hold is called its capacity of array.

```
Arr[6] = [1,2,3,4,5,6]
```

```
arr= [1,2,3,4,5,6]
```

Number of elements currently in an array is called length of array.

```
Length of array = len(arr)
```

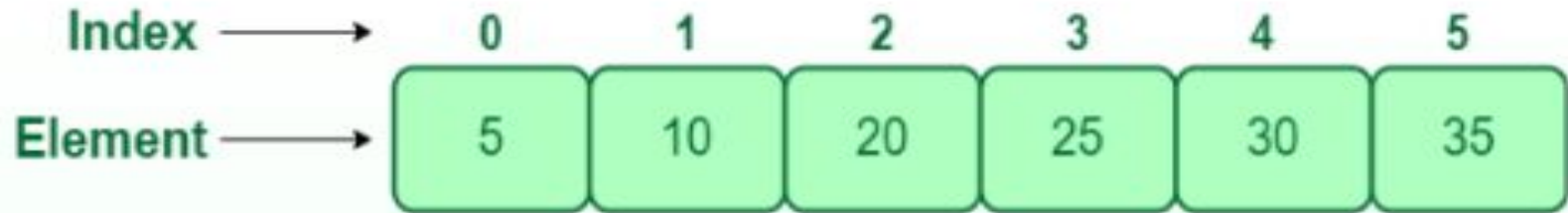
# Types of Arrays

Arrays can be categorized into different types:

1. One dimensional array
2. Two dimensional array
3. Three dimensional array

# 1. One Dimensional Array

You can imagine a 1d array as a row, where elements are stored one after another.



## 2. Two Dimensional Array


$$\begin{bmatrix} 1, 2, 3 \\ 4, 5, 6 \\ 7, 8, 9 \end{bmatrix}$$

2D Array

=


$$\begin{bmatrix} 1, 2, 3 \\ 4, 5, 6 \\ 7, 8, 9 \end{bmatrix}$$

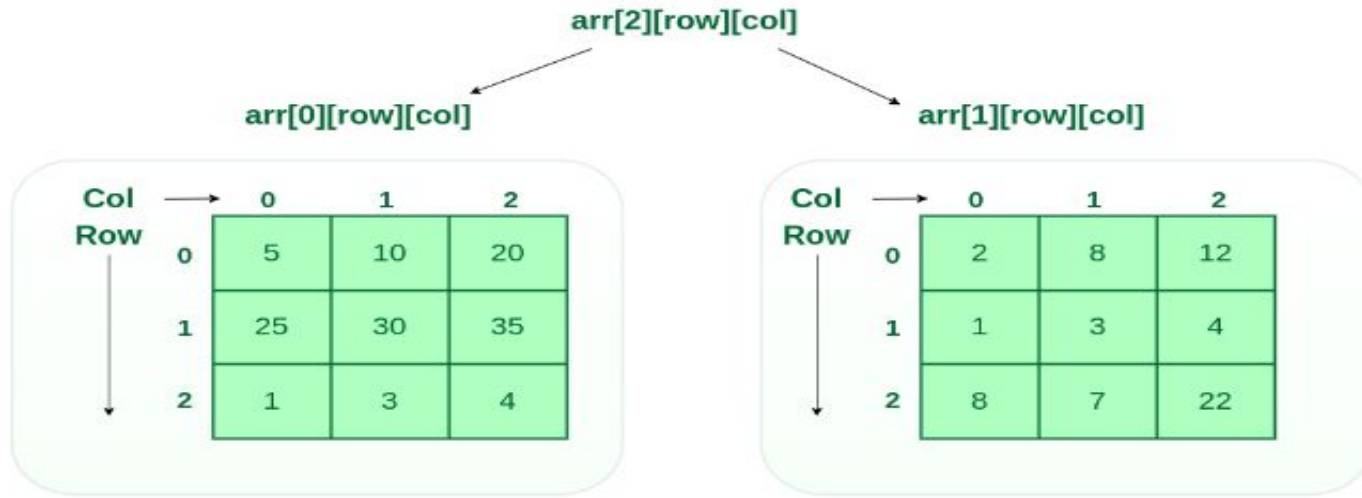
Matrix

SCALER  
Topics



### 3. Three Dimensional Array

A 3-D Multidimensional array contains three dimensions, so it can be considered an array of two-dimensional arrays.



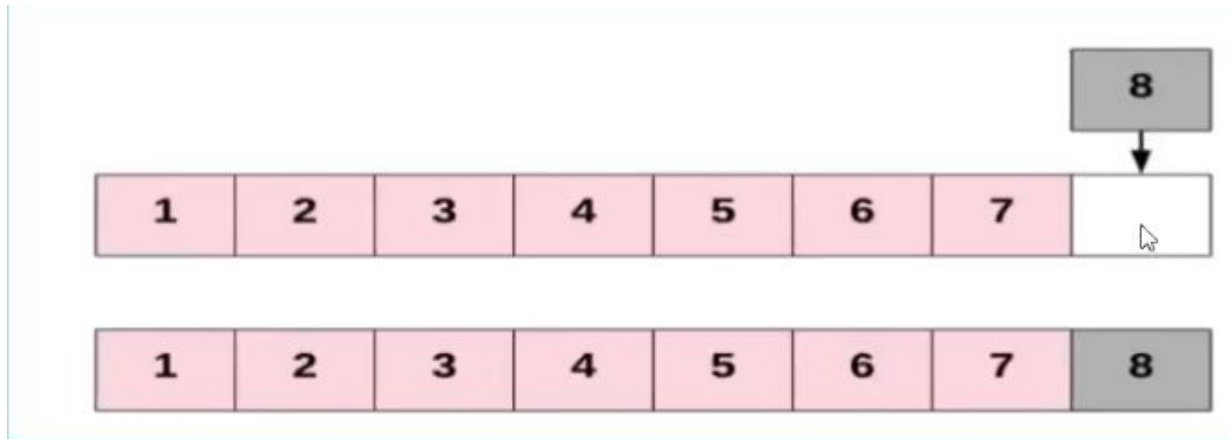
# Basic operations in an Array

- **Traverse** – print all the array elements one by one.
- **Insertion** – Adds an element at the given index.
- **Deletion** – Deletes an element at the given index.
- **Search** – Searches an element using the given index or by the value.
- **Update** – Updates an element at the given index.

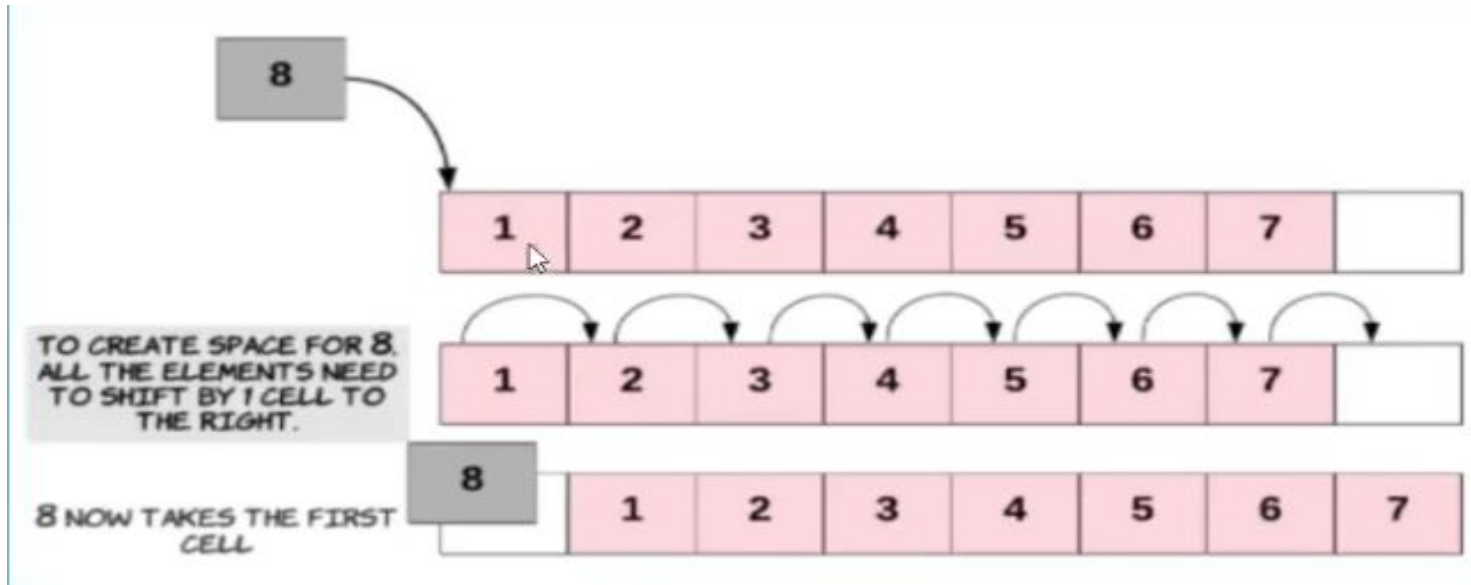
# Array Insertion

Inserting a new element in an array

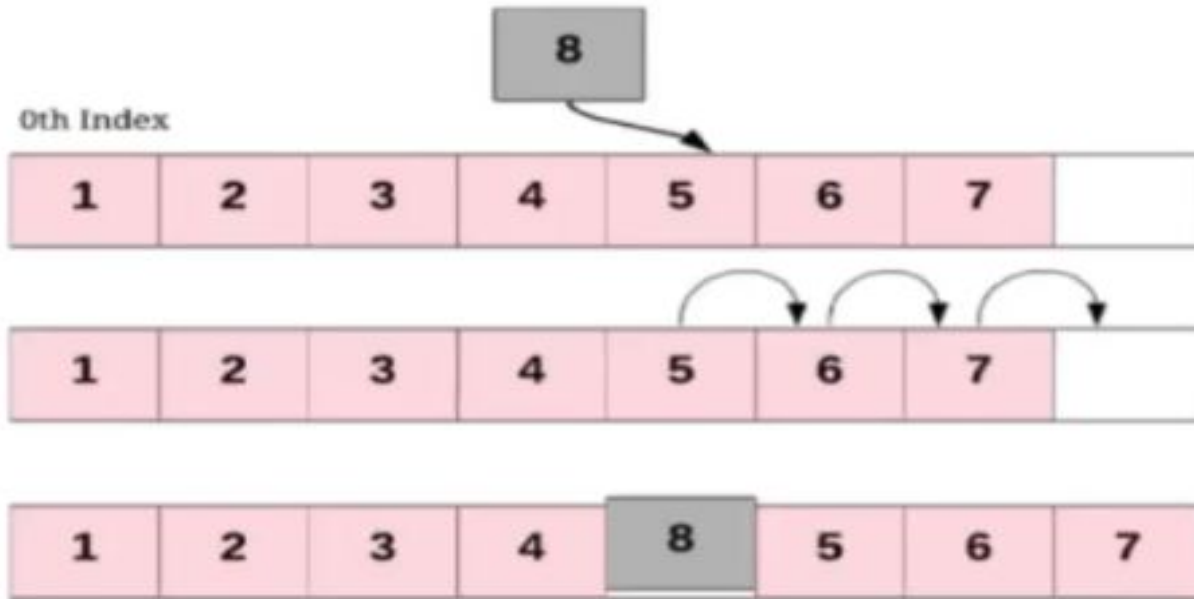
1. Inserting a new element at the end of the array.



## 2. Inserting a new element at the beginning of the array.

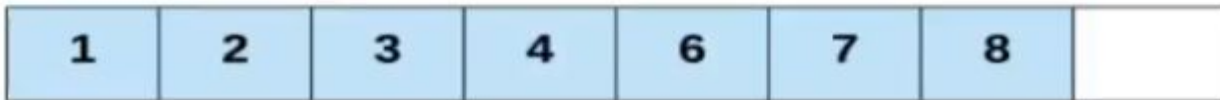
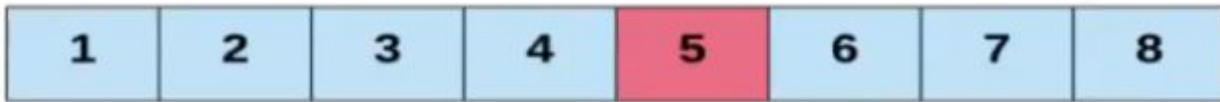


### 3. Inserting a new element at any given index of the array.



# Array Deletion

0th Index

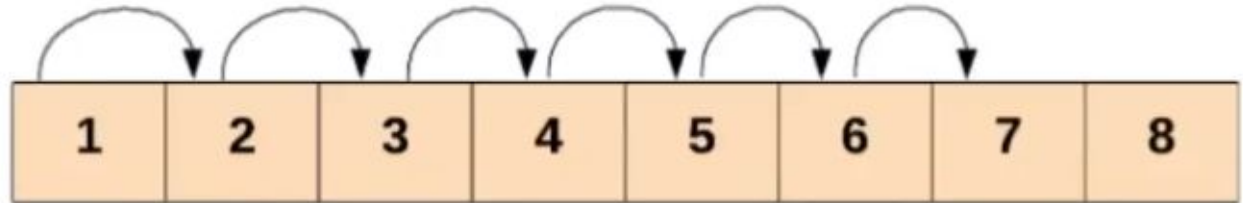


# Array Search

SEARCH FOR 7?



START FROM THE  
BEGINNING AND KEEP  
LOOKING UNTIL 7 IS  
FOUND OR ARRAY ENDS.



# Real time Applications of Array

**Music and Videos:** Have you ever noticed how your phone or computer can play music and videos smoothly? That's because they use arrays to store the individual sounds and images that make up the media.