**Data Structure**

Data Structure is a way of collecting and organising data in such a way that we can perform operations on these data in an effective way. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage. For example, we have some data which has, player's **name** "Virat" and **age** 26. Here "Virat" is of **String** data type and 26 is of **integer** data type.

We can organize this data as a record like **Player** record, which will have both player's name and age in it. Now we can collect and store player's records in a file or database as a data structure. **For example**: "Dhoni" 30, "Gambhir" 31, "Sehwag" 33

If you are aware of Object Oriented programming concepts, then a class also does the same thing, it collects different type of data under one single entity. The only difference being, data structures provides for techniques to access and manipulate data efficiently.

In simple language, Data Structures are structures programmed to store ordered data, so that various operations can be performed on it easily. It represents the knowledge of data to be organized in memory. It should be designed and implemented in such a way that it reduces the complexity and increases the efficiency.

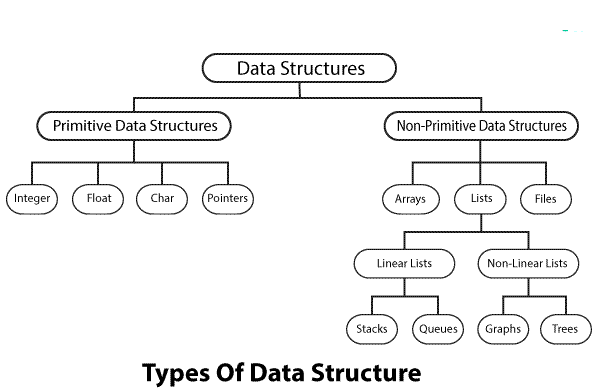
Basic types of Data Structures

As we have discussed above, anything that can store data can be called as a data structure, hence Integer, Float, Boolean, Char etc, all are data structures. They are known as **Primitive Data Structures**.

Then we also have some complex Data Structures, which are used to store large and connected data. Some example of **Abstract Data Structure** are :

* Linked List
* Tree
* Graph
* Stack, Queue etc.

All these data structures allow us to perform different operations on data. We select these data structures based on which type of operation is required. We will look into these data structures in more details in our later lessons.



Primitive Data Types :

* Integer
* Float
* Char
* Pointers

Non-Primitive Data Structures :

1. Arrays :- An array is a finite group of data, which is allocated contiguous (i.e. sharing a common border) memory locations, and each element within the array is accessed via an index key (typically numerical, and zero based).

The name assigned to an array is typically a pointer to the first item in the array. Meaning that given an array identifier of arr which was assigned the value ["a", "b", "c"], in order to access the "b" element you would use the index 1 to lookup the value: arr[1].

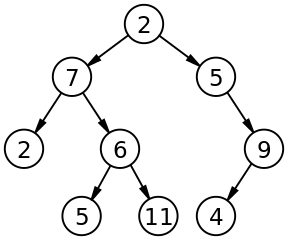
1. [Stacks](https://whatis.techtarget.com/definition/stack) :- A stack stores a collection of items in the linear order that operations are applied. This order could be last in first out (LIFO) or first in first out ([FIFO](https://whatis.techtarget.com/definition/FIFO-first-in-first-out)).
2. [Queues](https://whatis.techtarget.com/definition/queue) :- A queue stores a collection of items similar to a stack; however, the operation order can only be first in first out.
3. Linked lists :- A linked list is different to an array in that the order of the elements within the list are not determined by a contiguous memory allocation. Instead the elements of the linked list can be sporadically placed in memory due to its design, which is that each element of the list (also referred to as a ‘node’) consists of two parts:

* the data
* a pointer

The data is what you’ve assigned to that element/node, whereas the pointer is a memory address reference to the next node in the list.

https://www.integralist.co.uk/images/linked-list.png

1. Trees :- The concept of a ‘tree’ in its simplest terms is to represent a hierarchical tree structure, with a root value and subtrees of children (with a parent node), represented as a set of linked nodes.



A tree contains “nodes” (a node has a value associated with it) and each node is connected by a line called an “edge”. These lines represent the ***relationship*** between the nodes.

1. Graphs :- A graph is an abstract data type intended to guide the implementation of a data structure following the principles of [graph theory](https://en.wikipedia.org/wiki/Graph_theory).

The data struture itself is non-linear and it consists of:

**nodes**: points on the graph (also known as ‘vertices’).

**edges**: lines connecting each node.

The following image demonstrates a ‘directed’ graph (notice the edges have arrows indicating the direction and flow):

