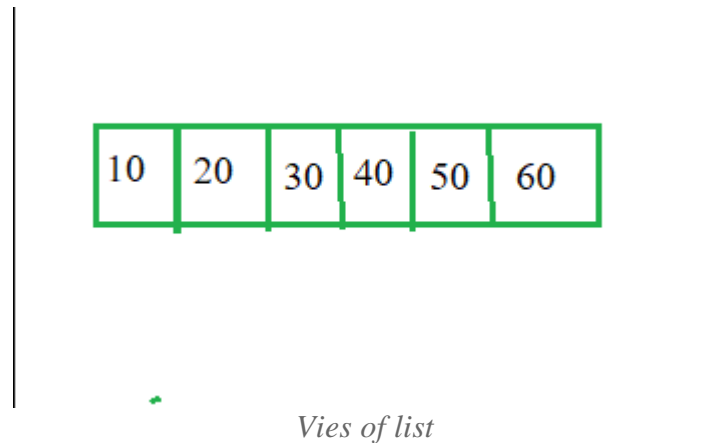


List ADT

Abstract Data type (ADT) is a type (or class) for objects whose behavior is defined by a set of values and a set of operations. The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented. It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations. It is called “abstract” because it gives an implementation-independent view.

List ADT



- The data is generally stored in key sequence in a list which has a head structure consisting of *count*, *pointers* and *address of compare function* needed to compare the data in the list.
- The data node contains the *pointer* to a data structure and a *self-referential pointer* which points to the next node in the list.
- The **List ADT Functions** is given below:
 - `get()` – Return an element from the list at any given position.
 - `insert()` – Insert an element at any position of the list.
 - `remove()` – Remove the first occurrence of any element from a non-empty list.
 - `removeAt()` – Remove the element at a specified location from a non-empty list.
 - `replace()` – Replace an element at any position by another element.
 - `size()` – Return the number of elements in the list.
 - `isEmpty()` – Return true if the list is empty, otherwise return false.
 - `isFull()` – Return true if the list is full, otherwise return false.