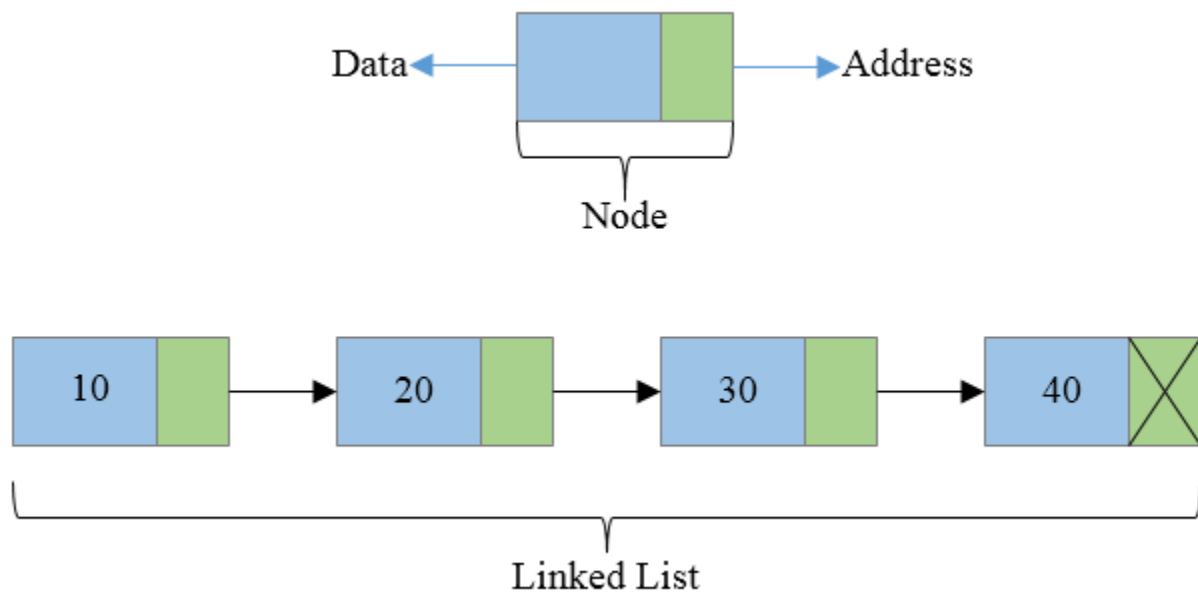


Linked List: Day 3 of 50 Days of DSA in Python



What is a Linked List?

A linked list is a linear data structure where elements are stored in nodes. Each node contains two parts:

1. Data - The value stored in the node.
2. Pointer - A reference to the next node in the sequence.


Key Characteristics:

- Dynamic in size (no fixed array size).
 - Efficient insertions and deletions compared to arrays.
-

Linked List: Day 3 of 50 Days of DSA in Python

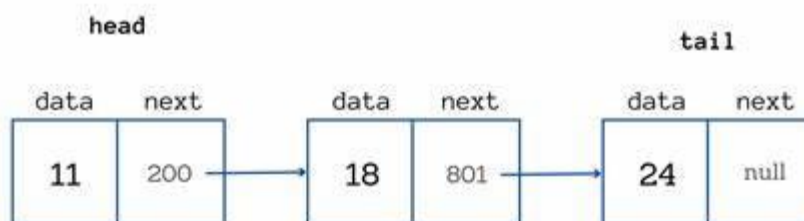
Comparison with Arrays:

- Arrays: Fixed size, continuous memory, $O(1)$ access time.
- Linked Lists: Dynamic size, scattered memory, $O(n)$ access time.

masai.	
Array [...]	Linked List 
Elements are stored in contiguous memory locations	Elements are connected using pointers
Supports random access to elements	Only supports sequential access to elements
Insertions and deletions are tricky: elements need to be shifted	Insertions and deletions can be done efficiently without shifting
Fixed memory: static memory allocation	Dynamic memory allocation at runtime
Elements are independent of each other	Each node points to the next node or both the next and the previous node

Components of a Linked List:

- Node: A structure holding data and a pointer.
- Pointer: Links one node to another.
- Head: The starting point of the list.



Linked List: Day 3 of 50 Days of DSA in Python

Types of Linked Lists:

1. Singly Linked List: Nodes are connected in one direction.
 2. Doubly Linked List: Nodes are connected in both directions.
 3. Circular Linked List: The last node connects to the first, forming a circle.
-


Why Use Linked Lists?


Advantages:

- Dynamic size allocation.
- Faster insertions/deletions.

Disadvantages:

- Sequential access makes searching slower.
- Extra memory for storing pointers.

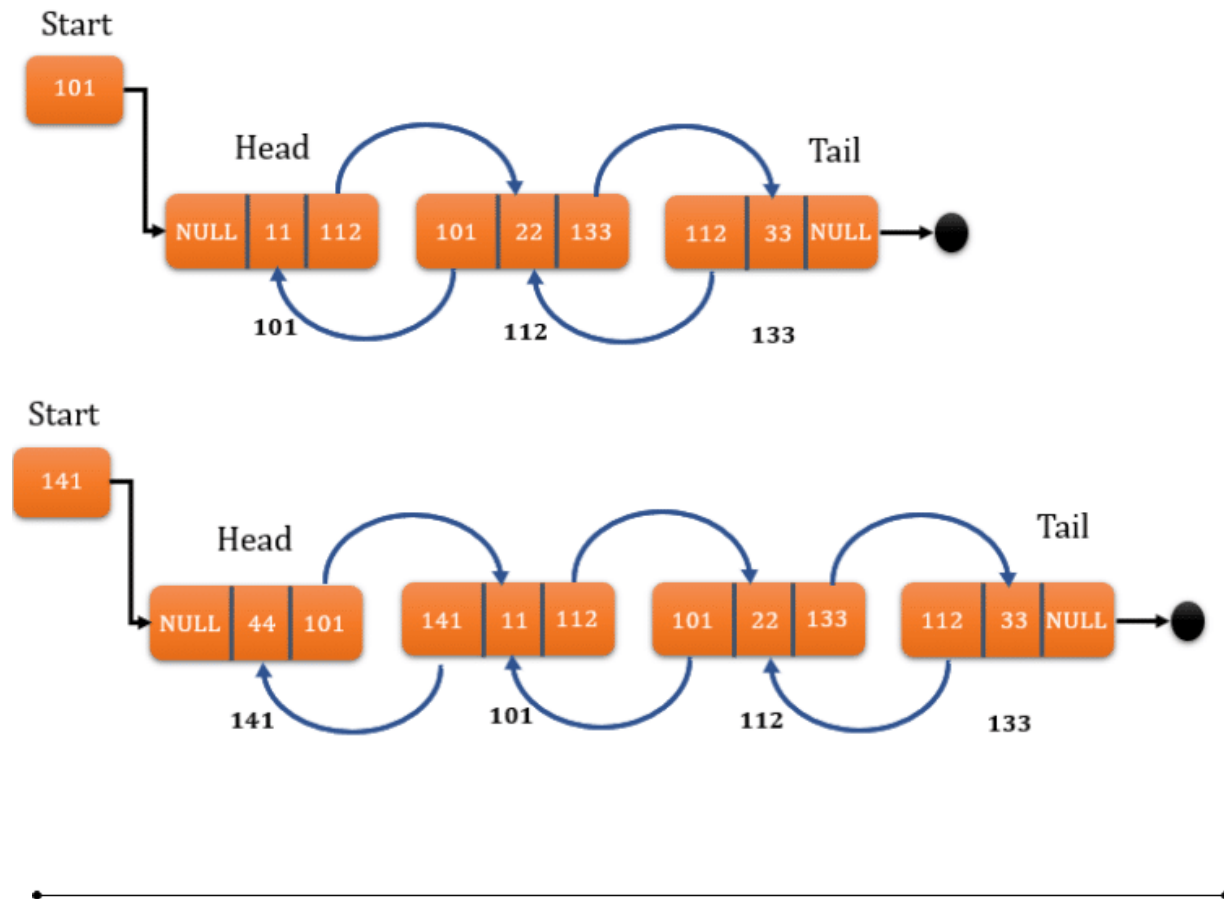
 Linked Lists	
Advantages	Disadvantages
<ul style="list-style-type: none">• Dynamic growth/shrinkage• Easy implementation• No space overhead• Easy insertion and deletion	<ul style="list-style-type: none">• More consumption of memory• Difficult traversal• Random access not possible• Reverse traversing causes memory wastage



Linked List: Day 3 of 50 Days of DSA in Python

Operations in Linked Lists:

1. Insertion: Add a node at the beginning, end, or specific position.
2. Deletion: Remove a node from the list.
3. Traversal: Go through each node to access or process data.



Applications of Linked Lists:

- Dynamic memory allocation (e.g., operating systems).
- Implementation of stacks and queues.
- Undo functionality in text editors