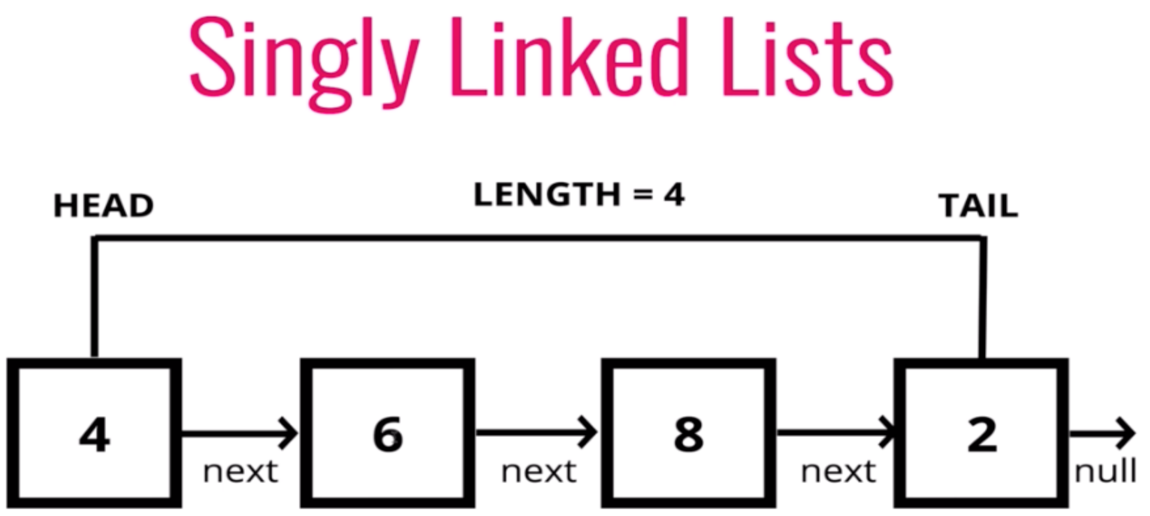
📌 Day 4 of 50 Days of DSA in Python – Singly Linked List 🔗

👋 **Welcome to Day 4** of our **50 Days of DSA in Python** challenge! 🚀
Today, we're diving into **Singly Linked Lists (SLL)** – one of the most important **linear data structures** used in programming. 🏗️

🔗 What is a Singly Linked List?

A **Singly Linked List (SLL)** is a collection of **nodes**, where each node consists of two parts:
1 **Data** 🏷️ – The actual value stored in the node.
2 **Next Pointer** 🔗 – A reference to the next node in the sequence.

Unlike arrays, linked lists are **not stored in contiguous memory**. Instead, each node points to the next one, allowing **dynamic memory allocation**.



📌 Day 4 of 50 Days of DSA in Python – Singly Linked List 🔗

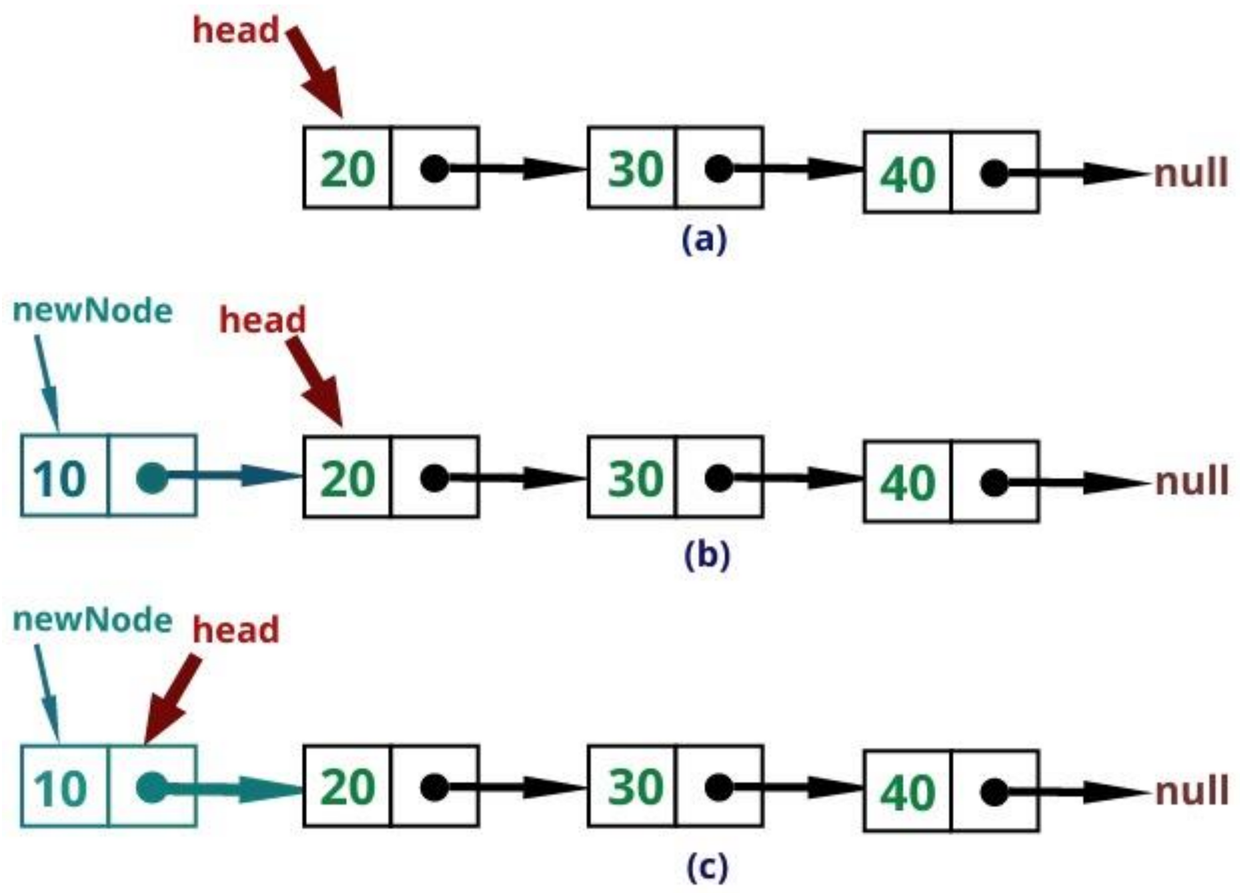
📌 Why Use Singly Linked Lists?

🔹 **Dynamic Size** – No need to predefine the size like arrays.
🔹 **Efficient Insertions & Deletions** – No shifting required like arrays.
🔹 **Better Memory Utilization** – Nodes are allocated as needed.

Drawbacks of Singly Linked Lists:
Extra Memory for Pointers – Each node stores an additional pointer.
Sequential Access Only – No direct access like arrays; must traverse.

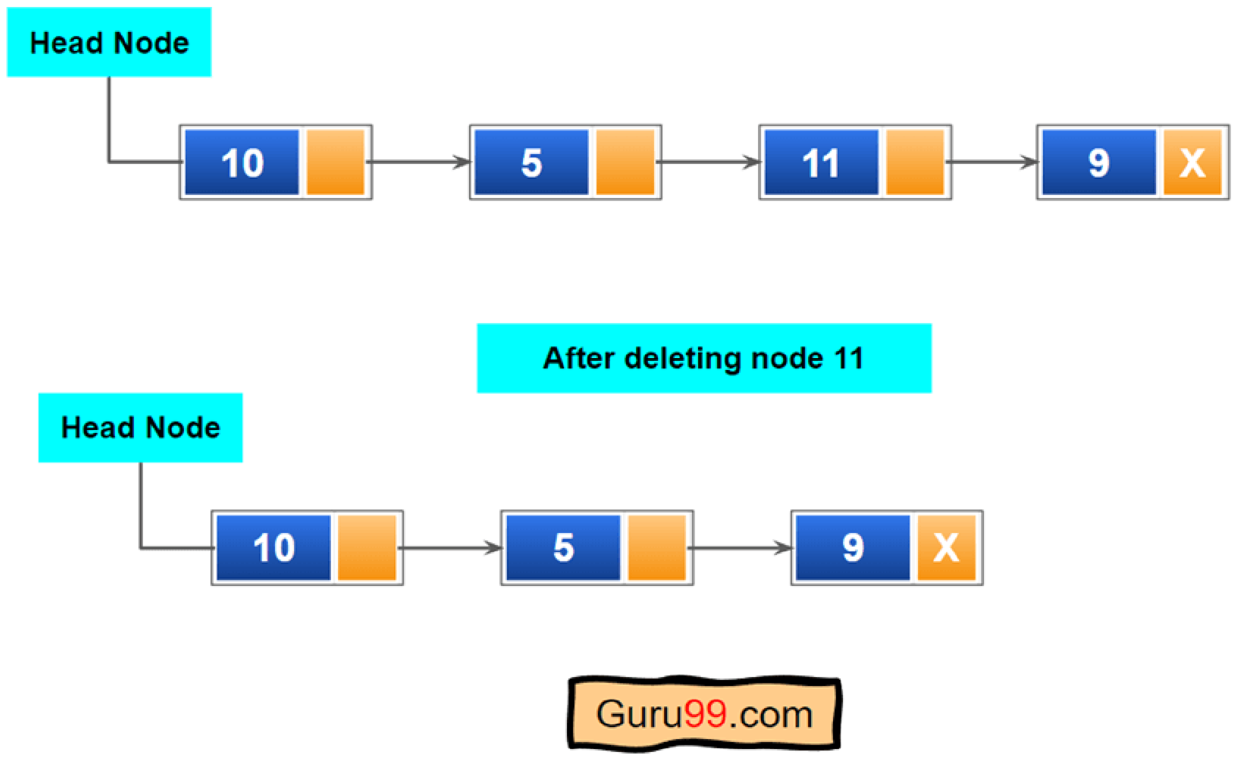
🛠 Basic Operations in SLL

📍 **Insertion** – Add elements at the beginning, end, or a specific position.

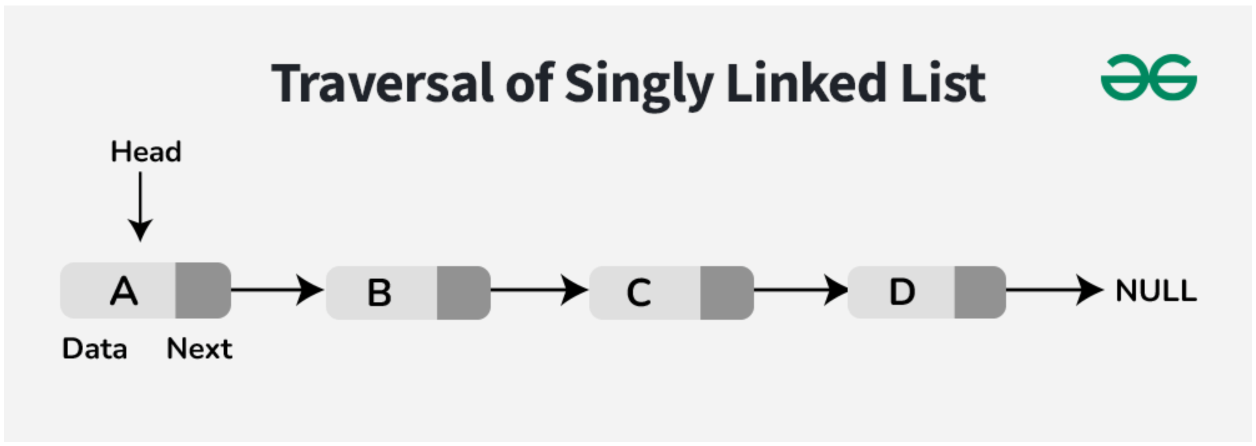


📌 Day 4 of 50 Days of DSA in Python – Singly Linked List 🔗

Deletion – Remove elements from the beginning, end, or any position.



Traversal – Go through all nodes to access elements.



Searching – Find an element by traversing the list.

📌 Day 4 of 50 Days of DSA in Python – Singly Linked List 🔗

🚀 How Singly Linked List Works?

Imagine a **train** 🚆 where each **coach (node)** is connected to the next one. The **engine (head node)** pulls the entire train forward by pointing to the next coach.

- ✅ **Head (First Node)** – Always points to the start of the list.
- 🔗 **Each node** connects to the next one, forming a chain.
- ❌ **No Backward Navigation** – Unlike Doubly Linked Lists, you can't move backward.

🎯 Key Takeaways

✅ **Best for dynamic memory allocation** where elements are frequently inserted or deleted.
✅ **Used in various applications** like implementing stacks, queues, and graphs.
✅ **Not suitable when random access is needed**, as traversal is sequential.

🔔 Stay Tuned for Day 5!

Tomorrow, we will explore **Doubly Linked Lists (DLL)** – an upgraded version of SLL that allows **bi-directional traversal**! 🔄📚

🔥 Keep coding, keep learning! 🚀💡