**📘 Singly Linked List (SLL)**

**🔹 Definition:**

A **Singly Linked List** is a linear data structure in which each element (called a **node**) contains:

1. **Data** – the actual value
2. **Next** – a pointer/reference to the next node in the list

The **last node’s next** is always None, indicating the end of the list.

**🔹 Node Structure:**

[ Data | Next ] → [ Data | Next ] → [ Data | None ]

**🔹 Key Components:**

* **Head**: The first node in the list
* **Next**: A reference to the next node
* **None**: Marks the end of the list

**🔹 Operations:**

**✅ 1. Traversal / Display**

Go through each node starting from the head and print its data.

**✅ 2. Insertion**

* At beginning
* At end
* At a specific position

**✅ 3. Deletion**

* From beginning
* From end
* From specific position

**🔹 Python Code Example**

python

# Define a node class

class Node:

def \_\_init\_\_(self, data):

self.data = data # data part

self.next = None # pointer to next node

# Define Singly Linked List

class SinglyLinkedList:

def \_\_init\_\_(self):

self.head = None # initially list is empty

# Append a node at the end

def append(self, data):

new\_node = Node(data)

if not self.head: # if list is empty

self.head = new\_node

return

curr = self.head

while curr.next: # traverse to the last node

curr = curr.next

curr.next = new\_node # link last node to new node

# Display the list

def display(self):

curr = self.head

while curr:

print(curr.data, end=" → ")

curr = curr.next

print("None")

**🔹 Example Usage**

python

sll = SinglyLinkedList()

sll.append(10)

sll.append(20)

sll.append(30)

sll.display()

**🔹 Output**

10 → 20 → 30 → None

**🔹 Advantages:**

* Dynamic size (no need to define size at start)
* Efficient insertion/deletion (compared to arrays)

**🔹 Disadvantages:**

* No backward traversal
* Random access not possible (must traverse nodes sequentially)