**Code: -**

class Node:

    def \_\_init\_\_(self, data):

        self.data = data  # The value of the node

        self.next = None  # The pointer to the next node (initially None)

class SinglyLinkedList:

    def \_\_init\_\_(self):

        self.head = None  # Initialize the head as None (empty list)

    def is\_empty(self):

        return self.head is None  # Return True if the list is empty

    def insert\_at\_end(self, data):

        new\_node = Node(data)

        if self.is\_empty():

            self.head = new\_node  # If the list is empty, make the new node the head

            return

        last\_node = self.head

        while last\_node.next:  # Traverse to the last node

            last\_node = last\_node.next

        last\_node.next = new\_node  # Link the last node to the new node

    def insert\_at\_beginning(self, data):

        new\_node = Node(data)

        new\_node.next = self.head  # Make the new node point to the current head

        self.head = new\_node  # Set the new node as the head

    def delete\_node(self, key):

        current = self.head

        if current and current.data == key:  # If the node to be deleted is the head

            self.head = current.next

            current = None

            return

        prev = None

        while current and current.data != key:  # Find the node with the given key

            prev = current

            current = current.next

        if current is None:  # If the key was not found in the list

            print("Node with value", key, "not found.")

            return

        prev.next = current.next  # Unlink the node to be deleted

        current = None  # Free memory (optional in Python)

    def search(self, key):

        current = self.head

        while current:

            if current.data == key:  # Node with the key found

                return True

            current = current.next

        return False  # Node with the key not found

    def display(self):

        if self.is\_empty():

            print("The list is empty.")

            return

        current = self.head

        while current:

            print(current.data, end="   - > ")

            current = current.next

        print("None")  # Indicate the end of the list

    def reverse(self):

        prev = None

        current = *self*.head

        while current:

            next\_node = current.next  *# Save the next node*

            current.next = prev  *# Reverse the current node's pointer*

            prev = current  *# Move prev to the current node*

            current = next\_node  *# Move to the next node*

*self*.head = prev  *# Set the new head to the last node*

*# Example Usage*

linked\_list = SinglyLinkedList()

linked\_list.insert\_at\_end(10)

linked\_list.insert\_at\_end(20)

linked\_list.insert\_at\_end(30)

linked\_list.insert\_at\_beginning(5)

print("Linked List after insertion:")

linked\_list.display()  *# Display the list*

linked\_list.delete\_node(20)

print("\nLinked List after deleting node with value 20:")

linked\_list.display()  *# Display the list after deletion*

found = linked\_list.search(30)

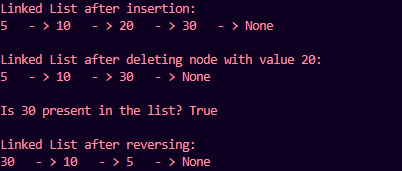
print("\nIs 30 present in the list?", found)

linked\_list.reverse()

print("\nLinked List after reversing:")

linked\_list.display()  *# Display the reversed list*

**OUTPUT:**

****