CELEBAL WEEK 2 ASSIGNMENT

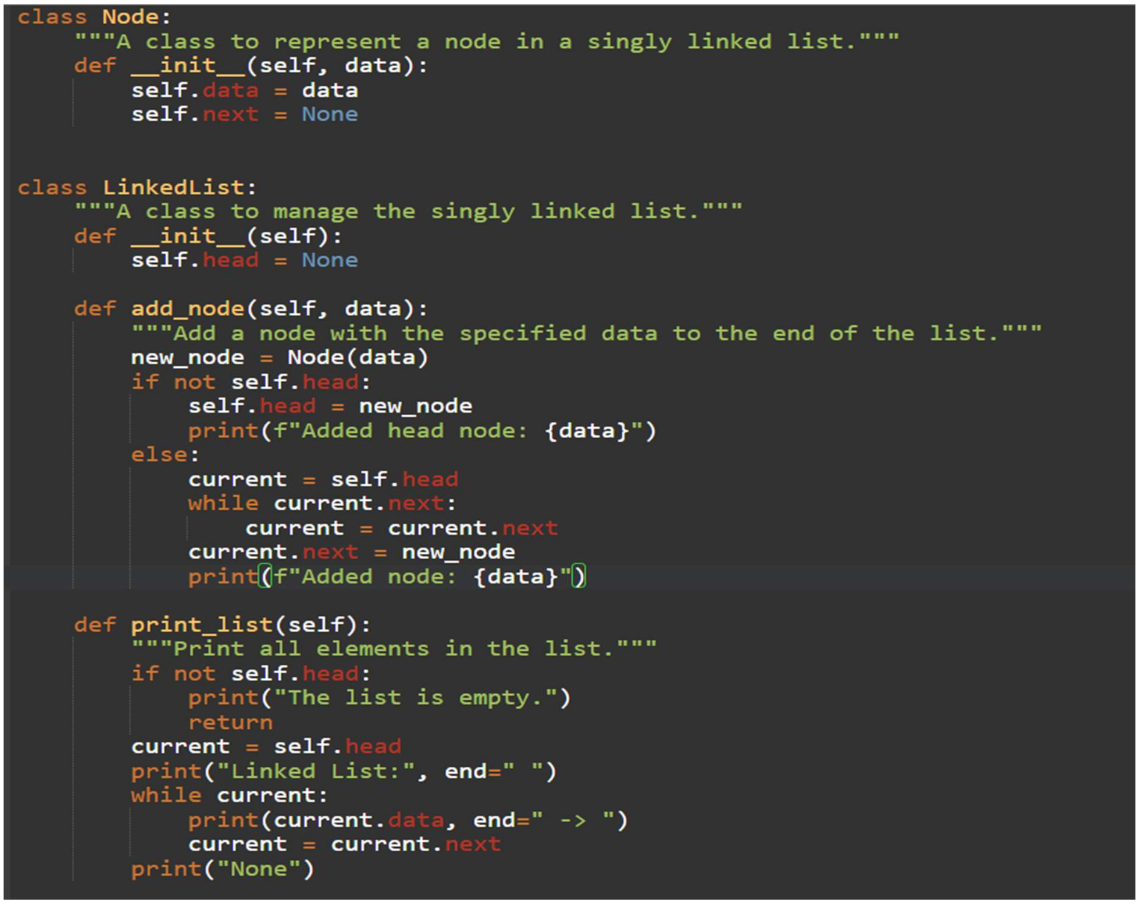
Create a Python program that implements a singly linked list using Object-Oriented Programming (OOP) principles. Your implementation should include the following: A Node class to represent each node in the list. A LinkedList class to manage the nodes, with methods to: Add a node to the end of the list Print the list Delete the nth node (where n is a 1-based index) Include exception handling to manage edge cases such as: Deleting a node from an empty list Deleting a node with an index out of range Test your implementation with at least one sample list.

<u>Algorithm:</u>

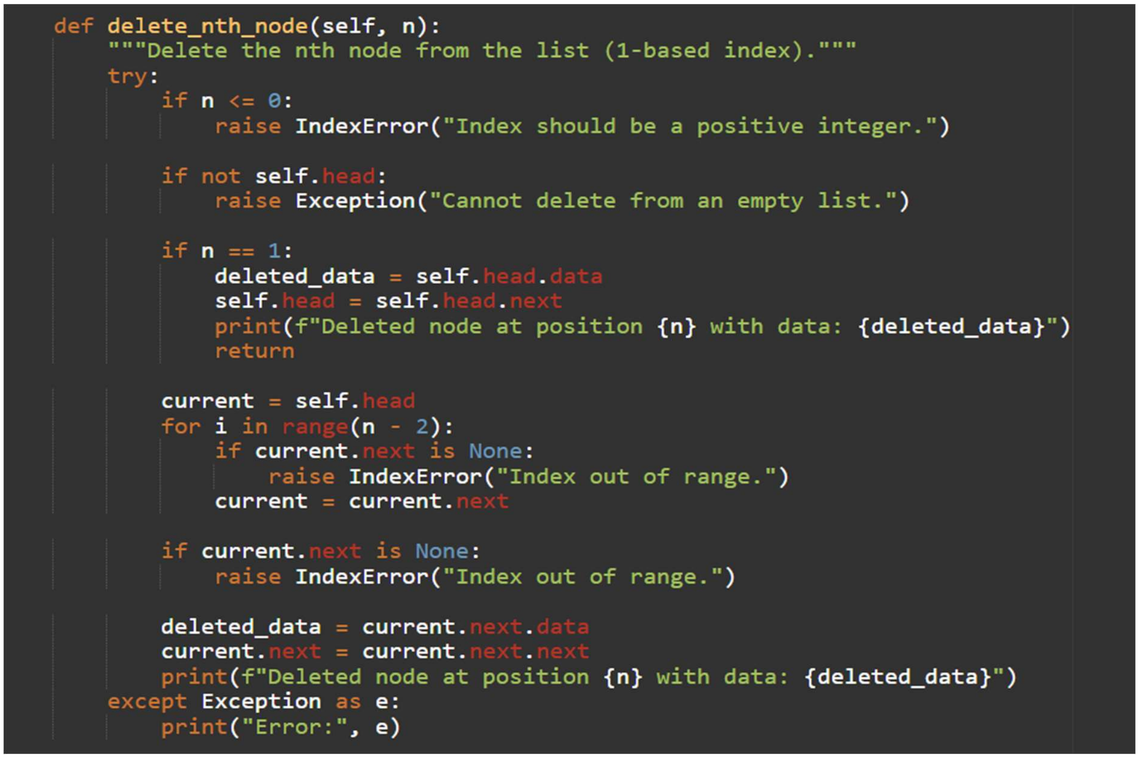
1. **Define Node class** with:
   data: to store value
   next: to store reference to the next node (default None)
2. **Define LinkedList class** with:
   head: initialize as None
3. **Add Node to End**:
   Create a new Node with given data
   If head is None, set head to new node
   Else, traverse to the last node and set its next to new node
4. **Print List**:
   If head is None, print "List is empty"
   Else, start from head and print data of each node until the end (next is None)
5. **Delete Nth Node (1-based index)**:
   If list is empty, raise exception
   If n <= 0, raise index error
   If n == 1, set head = head.next
   Else, traverse to (n-1)th node
   - If next node is None, raise index error
   - Else, set current.next = current.next.next to delete nth node
6. **Handle Exceptions**:
   Use try-except to catch:
   - Invalid index
   - Deleting from empty list
   - Index out of range
7. **Test All Operations**:
   Add sample nodes
   Print the list
   Delete specific nodes
   Test with edge cases (empty list, out-of-range index)

Name: Devanshi Mittal

Code:

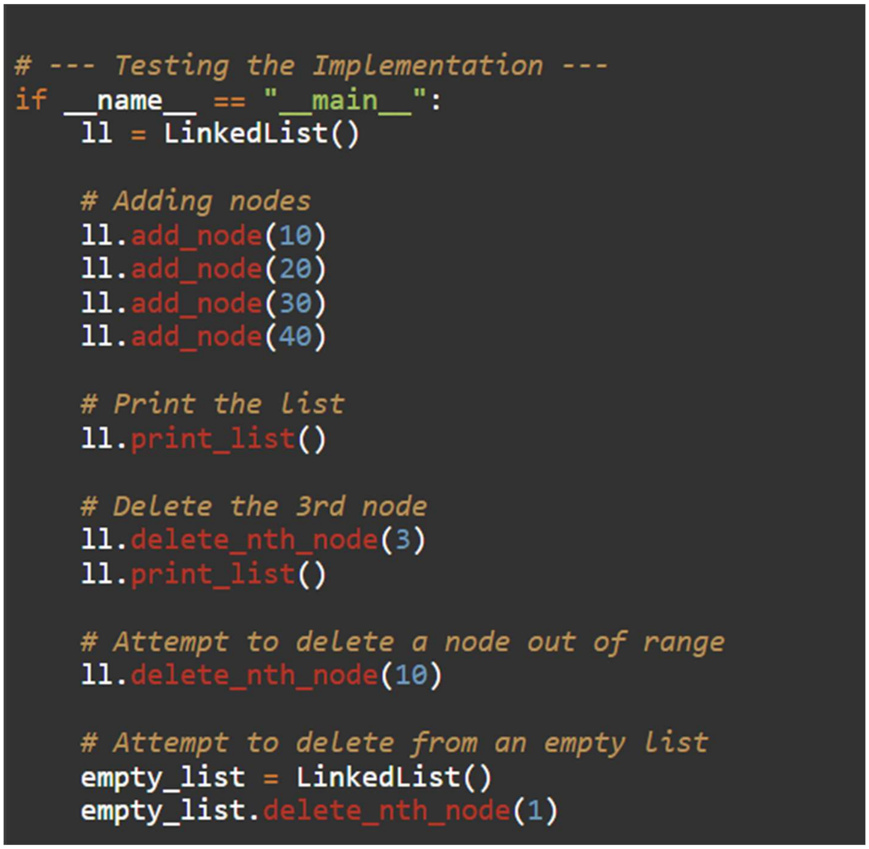


```python
class Node:
    """A class to represent a node in a singly linked list."""
    def __init__(self, data):
        self.data = data
        self.next = None


class LinkedList:
    """A class to manage the singly linked list."""
    def __init__(self):
        self.head = None

    def add_node(self, data):
        """Add a node with the specified data to the end of the list."""
        new_node = Node(data)
        if not self.head:
            self.head = new_node
            print(f"Added head node: {data}")
        else:
            current = self.head
            while current.next:
                current = current.next
            current.next = new_node
            print(f"Added node: {data}")

    def print_list(self):
        """Print all elements in the list."""
        if not self.head:
            print("The list is empty.")
            return
        current = self.head
        print("Linked List:", end=" ")
        while current:
            print(current.data, end=" -> ")
            current = current.next
        print("None")
```



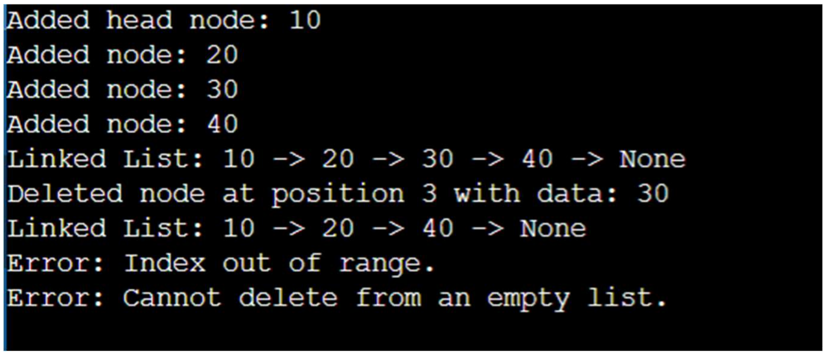
```python
    def delete_nth_node(self, n):
        """Delete the nth node from the list (1-based index)."""
        try:
            if n <= 0:
                raise IndexError("Index should be a positive integer.")

            if not self.head:
                raise Exception("Cannot delete from an empty list.")

            if n == 1:
                deleted_data = self.head.data
                self.head = self.head.next
                print(f"Deleted node at position {n} with data: {deleted_data}")
                return

            current = self.head
            for i in range(n - 2):
                if current.next is None:
                    raise IndexError("Index out of range.")
                current = current.next

            if current.next is None:
                raise IndexError("Index out of range.")

            deleted_data = current.next.data
            current.next = current.next.next
            print(f"Deleted node at position {n} with data: {deleted_data}")
        except Exception as e:
            print("Error:", e)
```

Name: Devanshi Mittal



```python
# --- Testing the Implementation ---
if __name__ == "__main__":
    ll = LinkedList()

    # Adding nodes
    ll.add_node(10)
    ll.add_node(20)
    ll.add_node(30)
    ll.add_node(40)

    # Print the list
    ll.print_list()

    # Delete the 3rd node
    ll.delete_nth_node(3)
    ll.print_list()

    # Attempt to delete a node out of range
    ll.delete_nth_node(10)

    # Attempt to delete from an empty list
    empty_list = LinkedList()
    empty_list.delete_nth_node(1)
```

Output:



```
Added head node: 10
Added node: 20
Added node: 30
Added node: 40
Linked List: 10 -> 20 -> 30 -> 40 -> None
Deleted node at position 3 with data: 30
Linked List: 10 -> 20 -> 40 -> None
Error: Index out of range.
Error: Cannot delete from an empty list.
```

Name: Devanshi Mittal