*# Create a Node class to create a node*

**class** **Node**:

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

self.data = data

self.next = **None**

*# Create a LinkedList class*

**class** **LinkedList**:

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

self.head = **None**

*# Method to add a node at begin of LL*

**def** insertAtBegin(self, data):

new\_node = Node(data)

**if** self.head **is** **None**:

self.head = new\_node

**return**

**else**:

new\_node.next = self.head

self.head = new\_node

*# Method to add a node at any index*

*# Indexing starts from 0.*

**def** insertAtIndex(self, data, index):

**if** (index == 0):

self.insertAtBegin(data)

position = 0

current\_node = self.head

**while** (current\_node != **None** **and** position+1 != index):

position = position+1

current\_node = current\_node.next

**if** current\_node != **None**:

new\_node = Node(data)

new\_node.next = current\_node.next

current\_node.next = new\_node

**else**:

print("Index not present")

*# Method to add a node at the end of LL*

**def** insertAtEnd(self, data):

new\_node = Node(data)

**if** self.head **is** **None**:

self.head = new\_node

**return**

current\_node = self.head

**while**(current\_node.next):

current\_node = current\_node.next

current\_node.next = new\_node

*# Update node of a linked list*

*# at given position*

**def** updateNode(self, val, index):

current\_node = self.head

position = 0

**if** position == index:

current\_node.data = val

**else**:

**while**(current\_node != **None** **and** position != index):

position = position+1

current\_node = current\_node.next

**if** current\_node != **None**:

current\_node.data = val

**else**:

print("Index not present")

*# Method to remove first node of linked list*

**def** remove\_first\_node(self):

**if**(self.head == **None**):

**return**

self.head = self.head.next

*# Method to remove last node of linked list*

**def** remove\_last\_node(self):

**if** self.head **is** **None**:

**return**

current\_node = self.head

**while**(current\_node != **None** **and** current\_node.next.next != **None**):

current\_node = current\_node.next

current\_node.next = **None**

*# Method to remove at given index*

**def** remove\_at\_index(self, index):

**if** self.head == **None**:

**return**

current\_node = self.head

position = 0

**if** position == index:

self.remove\_first\_node()

**else**:

**while**(current\_node != **None** **and** position+1 != index):

position = position+1

current\_node = current\_node.next

**if** current\_node != **None**:

current\_node.next = current\_node.next.next

**else**:

print("Index not present")

*# Method to remove a node from linked list*

**def** remove\_node(self, data):

current\_node = self.head

**if** current\_node.data == data:

self.remove\_first\_node()

**return**

**while**(current\_node != **None** **and** current\_node.next.data != data):

current\_node = current\_node.next

**if** current\_node == **None**:

**return**

**else**:

current\_node.next = current\_node.next.next

*# Print the size of linked list*

**def** sizeOfLL(self):

size = 0

**if**(self.head):

current\_node = self.head

**while**(current\_node):

size = size+1

current\_node = current\_node.next

**return** size

**else**:

**return** 0

*# print method for the linked list*

**def** printLL(self):

current\_node = self.head

**while**(current\_node):

print(current\_node.data)

current\_node = current\_node.next

*# create a new linked list*

llist = LinkedList()

*# add nodes to the linked list*

llist.insertAtEnd('a')

llist.insertAtEnd('b')

llist.insertAtBegin('c')

llist.insertAtEnd('d')

llist.insertAtIndex('g', 2)

*# print the linked list*

print("Node Data")

llist.printLL()

*# remove a nodes from the linked list*

print("**\n**Remove First Node")

llist.remove\_first\_node()

print("Remove Last Node")

llist.remove\_last\_node()

print("Remove Node at Index 1")

llist.remove\_at\_index(1)

*# print the linked list again*

print("**\n**Linked list after removing a node:")

llist.printLL()

print("**\n**Update node Value")

llist.updateNode('z', 0)

llist.printLL()

print("**\n**Size of linked list :", end=" ")

print(llist.sizeOfLL())