Circular Singly Linked List

In a circular Singly linked list, the last node of the list contains a pointer to the first node of the list. We can have circular singly linked list as well as circular doubly linked list.

We traverse a circular singly linked list until we reach the same node where we started. The circular singly liked list has no beginning and no ending. There is no null value present in the next part of any of the nodes.

The following image shows a circular singly linked list.

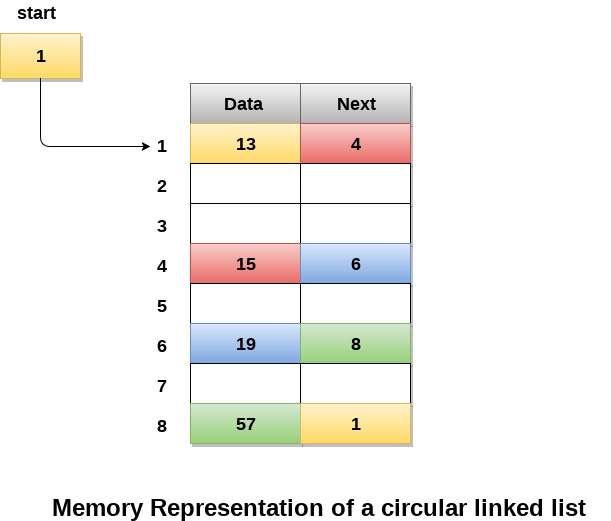


Circular linked list are mostly used in task maintenance in operating systems. There are many examples where circular linked list are being used in computer science including browser surfing where a record of pages visited in the past by the user, is maintained in the form of circular linked lists and can be accessed again on clicking the previous button.

## **Memory Representation of circular linked list:**

In the following image, memory representation of a circular linked list containing marks of a student in 4 subjects. However, the image shows a glimpse of how the circular list is being stored in the memory. The start or head of the list is pointing to the element with the index 1 and containing 13 marks in the data part and 4 in the next part. Which means that it is linked with the node that is being stored at 4th index of the list.

However, due to the fact that we are considering circular linked list in the memory therefore the last node of the list contains the address of the first node of the list.



We can also have more than one number of linked list in the memory with the different start pointers pointing to the different start nodes in the list. The last node is identified by its next part which contains the address of the start node of the list. We must be able to identify the last node of any linked list so that we can find out the number of iterations which need to be performed while traversing the list.

## **Operations on Circular Singly linked list:**

### **Insertion**

|  |  |  |
| --- | --- | --- |
| **SN** | **Operation** | **Description** |
| 1 | [Insertion at beginning](https://www.javatpoint.com/insertion-in-circular-singly-list-at-beginning) | Adding a node into circular singly linked list at the beginning. |
| 2 | [Insertion at the end](https://www.javatpoint.com/insertion-in-circular-singly-linked-list-at-end) | Adding a node into circular singly linked list at the end. |

### **Deletion & Traversing**

|  |  |  |
| --- | --- | --- |
| **SN** | **Operation** | **Description** |
| 1 | [Deletion at beginning](https://www.javatpoint.com/deletion-in-circular-singly-linked-list-at-beginning) | Removing the node from circular singly linked list at the beginning. |
| 2 | [Deletion at the end](https://www.javatpoint.com/deletion-in-circular-singly-linked-list-at-end) | Removing the node from circular singly linked list at the end. |
| 3 | [Searching](https://www.javatpoint.com/searching-in-circular-singly-linked-list) | Compare each element of the node with the given item and return the location at which the item is present in the list otherwise return null. |
| 4 | [Traversing](https://www.javatpoint.com/traversing-in-circular-singly-linked-list) | Visiting each element of the list at least once in order to perform some specific operation. |

# Python program to demonstrate

# circular linked list traversal

# Structure for a Node

class Node:

    # Constructor to create  a new node

    def \_\_init\_\_(self, data):

        self.data = data

        self.next = None

class CircularLinkedList:

    # Constructor to create a empty circular linked list

    def \_\_init\_\_(self):

        self.head = None

    # Function to insert a node at the beginning of a

    # circular linked list

    def push(self, data):

        ptr1 = Node(data)

        temp = self.head

        ptr1.next = self.head

        # If linked list is not None then set the next of

        # last node

        if self.head is not None:

            while(temp.next != self.head):

                temp = temp.next

            temp.next = ptr1

        else:

            ptr1.next = ptr1 # For the first node

        self.head = ptr1

    # Function to print nodes in a given circular linked list

    def printList(self):

        temp = self.head

        if self.head is not None:

            while(True):

                print "%d" %(temp.data),

                temp = temp.next

                if (temp == self.head):

                    break

# Driver program to test above function

# Initialize list as empty

cllist = CircularLinkedList()

# Created linked list will be 11->2->56->12

cllist.push(12)

cllist.push(56)

cllist.push(2)

cllist.push(11)

print "Contents of circular Linked List"

cllist.printList()

**Output:**

Contents of Circular Linked List

11 2 56 12