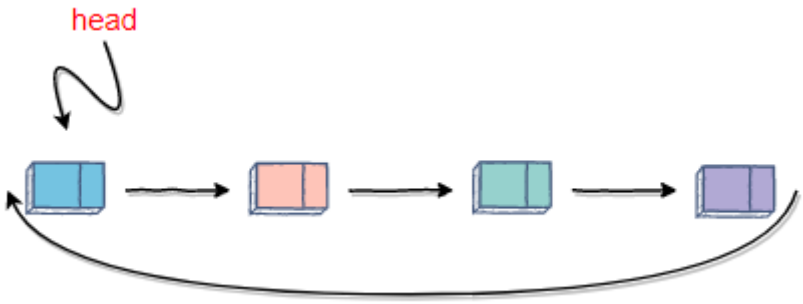
Circular LL

February 25, 2021



[3]:
```
class Node():

    def __init__(self,data):
        self.data=data
        self.next=None
```

[4]:
```
class CircularLL():
    def __init__(self,node=None):
        self.head=node
        if node:
            node.next=self.head #Establish a loop head node


    def is_empty(self):
        return self.head==None

    def length(self):
        if self.is_empty():
            return 0

        count=1
        cur=self.head
        while cur.next!=self.head:
            count+=1
            cur=cur.next
        return count
```

```
    def travel(self):
        if self.is_empty():
            return
        #Creating a cursor equals the starting node
        cur = self.head
        while cur.next != self.head:
            print(cur.data)
            cur = cur.next
        print(cur.data)

    def HeadInsert(self,num):
        node=Node(num)
        if self.is_empty():
            self.head=node
            node.next=node
        else:
            cur=self.head
            while cur.next!=self.head:
                cur=cur.next
            node.next=self.head
            self.head=node
            cur.next=self.head

    def TailInsert(self,num):
        node=Node(num)
        if self.is_empty():
            self.head=node
            node.next=self.head
        else:
            cur=self.head
            while cur.next!=self.head:
                cur=cur.next
            cur.next=node
            node.next=self.head


    def NodeInsert(self,index,num):
        #Point to the address of self. head, which is not a header element
        # For the next element, so Preis the next element
        if index <= 0:
            #Think of it as head insertion.
            self.HeadInsert(num)
        elif index > (self.length()-1):
            self.TailInsert(num)
        else:
            pre_cur = self.head
            count = 0
```

```
            while count < (index-1):
                count+=1
                pre_cur = pre_cur.next
            node = Node(num)
            node.next = pre_cur.next
            pre_cur.next = node
```

[5]:
```
a=CircularLL()
for i in range(6):
    a.TailInsert(i)
```

[6]:
```
a.travel()
```

```
0
1
2
3
4
5
```

[7]:
```
a.NodeInsert(10,102)
```

[9]:
```
a.travel()
```

```
0
1
2
3
4
5
102
```

0.0.1 Interface for deletion of nodes can be tried out.

[ ]:
```

```