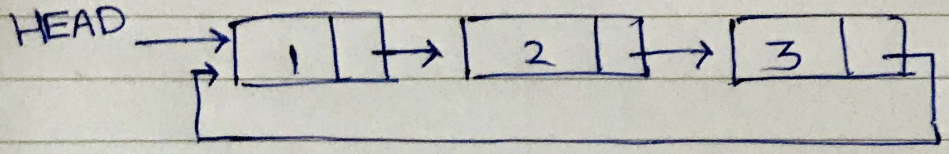
Circular Linked List

- Circular Linked is a variation of a Linked List in which the last element is linked to the first element.




```
Struct node *head
struct node * one = NULL;
struct node * two = NULL;
```
} Initialize

```
one = malloc (size of (struct node));
two =      //
three =    //
```
} Memory Allocation

```
one -> data = 1;
two -> data = 2;
three -> data = 3;
```
} Assign Data values

```
one -> next = two;
two -> next = three;
three -> next = one;
```
} Connect node → Last node List to the first node

```
head = one
```
→ First nod address in head.

Advantage :-
i) Any node can be a starting point
ii) Implementation of queue
iii) Circular double linked list used in implementation of Advance D.S.

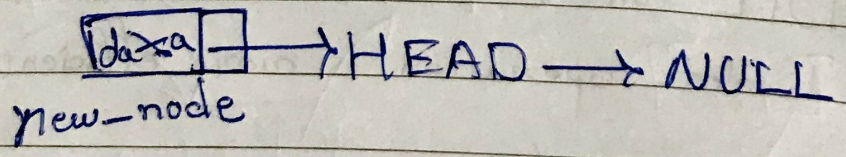
Push Operation on Circular Linked List

→ if HEAD is None :-

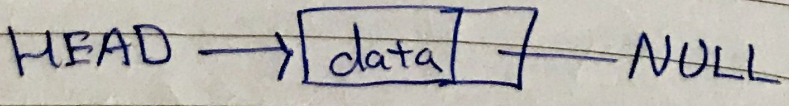
HEAD → NULL

```
def push(self, data):
    new_node = node(data)
    temp = self.head
    new_node.next = self.head
```

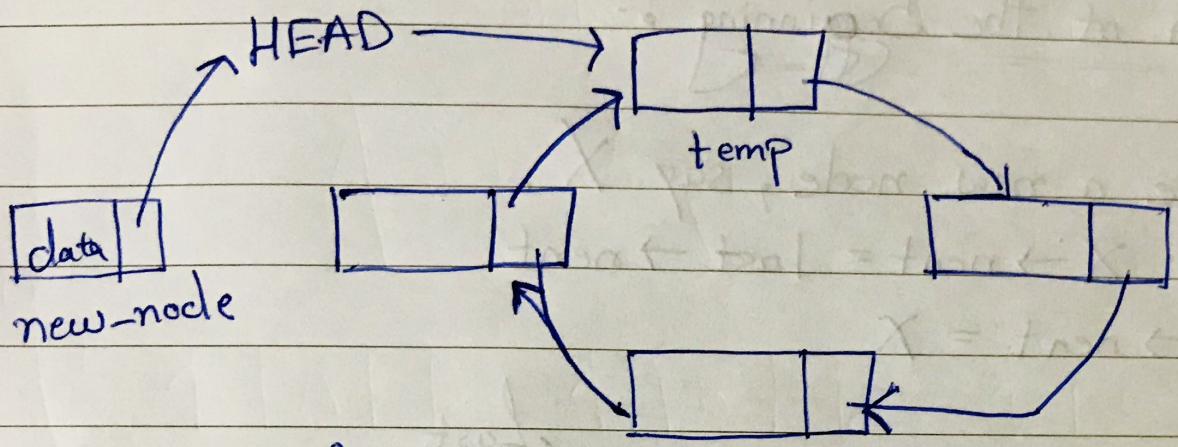
} Base Condition



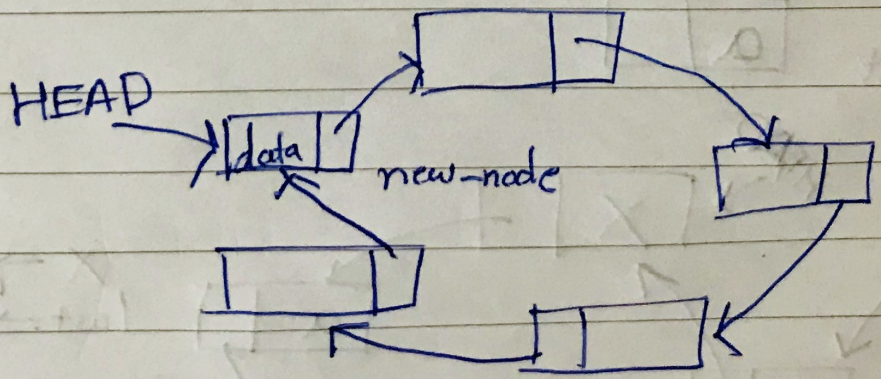

```
if self.head is None:
    new_node.next = new_node
self.head = new_node
```



→ If head is not None :-




```
if head is not None:
    while (temp.next != HEAD):
        temp = temp.next
    temp.next = new_node
self.head = new_node.
```



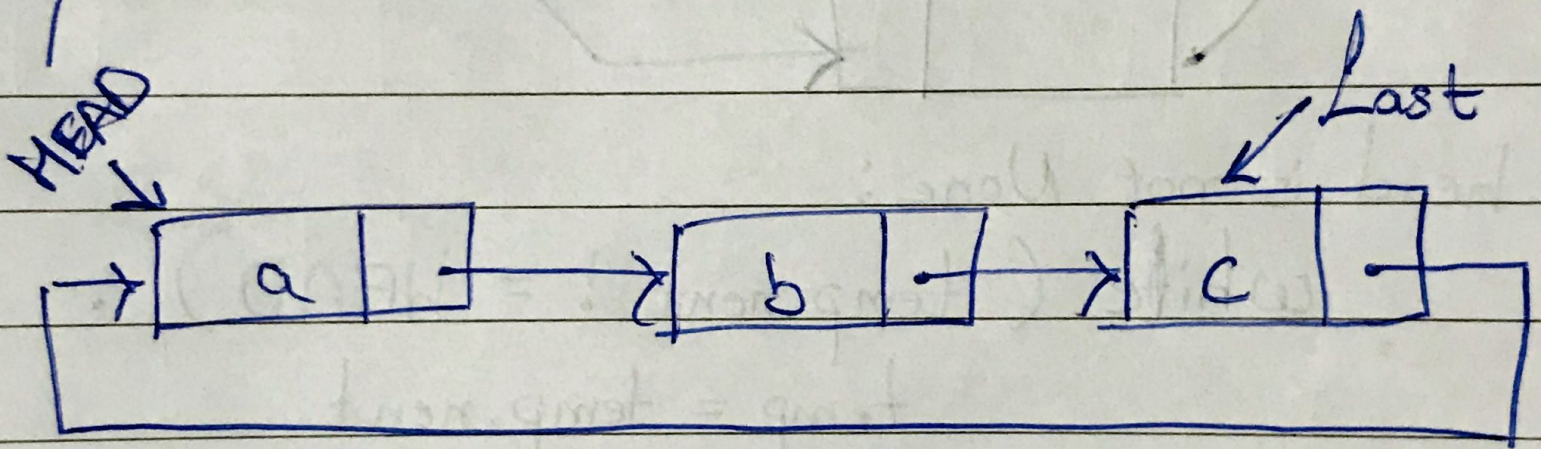
Circular Linked List :-

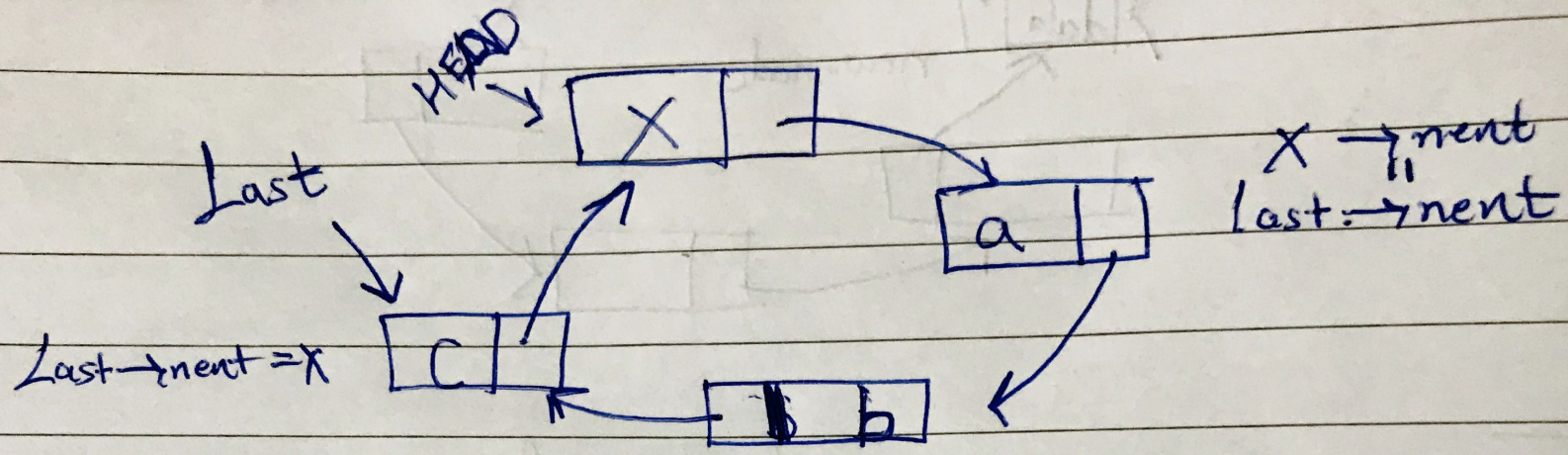
⇒ Insertion at the beginning :-

i) Create a new node, say X.

ii) Make X → next = last → next

iii) last → next = X

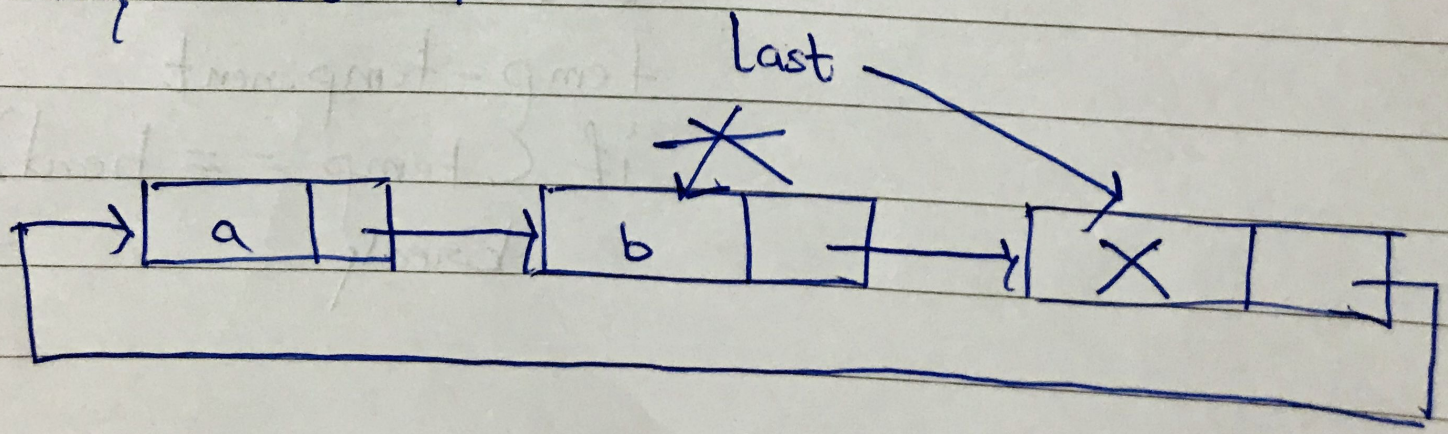




=> Insertion at the end :-

i) Create a ned node, say X.
ii) Make X->next = last->next
iii) last->next = X
iv) last = T

} Same as Insertion at beginning



⇒ Insertion in between the nodes :

1. Create a node, say T
2. Search the node before T, say P
3. Make T→next = P→next
4. P→next = T