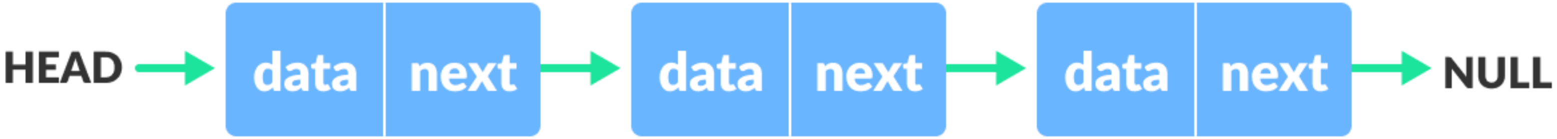
15CSE374
INTRODUCTION TO DATA STRUCTURES AND ALGORITHMS

Sarath tv

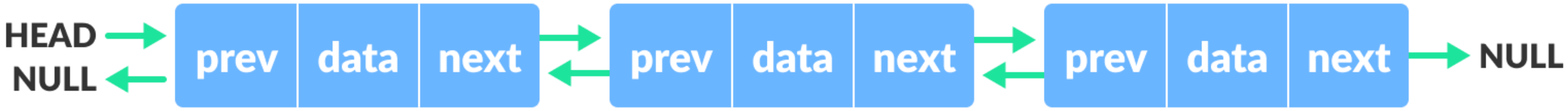
Last Lecture

- Singly LL.
- Interfaces.
- Implementation in python.

Doubly LL



- In singly linked list each node of the list has two components,
  - the actual value of the node and
  - the reference to the next node in the linked list.
- In the doubly linked list, each node has three components:
  - the value of the node,
  - the reference to the previous node, and
  - the reference to the next node.
- For the start node of the doubly linked list, the reference to the previous node is null.
- For the last node in the doubly linked list, the reference to next node is null.



HEAD

NULL

prev data next

prev data next

prev data next

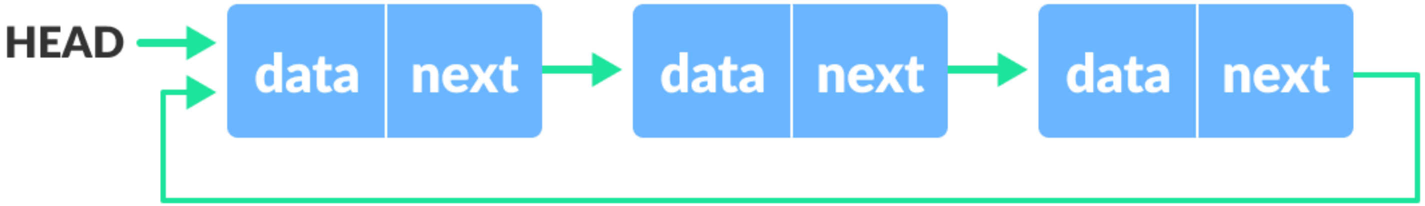
NULL

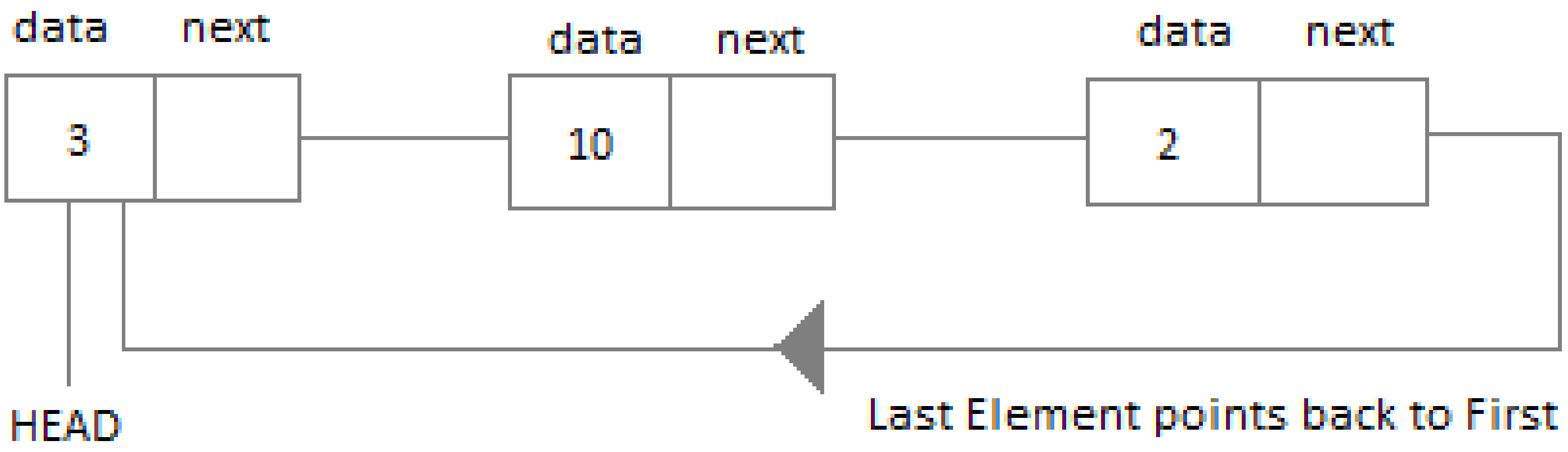
Pros and Cons of a Doubly Linked List

- **Pros**
- Can be traversed and searched in **both directions.**
- Basic **operations** such as insertion and deletion are **easier** to **implement** in the doubly linked lists
- **Cons**
- One of the major drawbacks of the doubly linked list is that you need **more memory space** to store one extra reference for each node.
- A **few additional steps** are required to be performed in order to perform insertion and deletion operations.

Circular Linked List

- ***Circular linked list*** *is a linked list where all nodes are connected to form a circle.*
- *There is no NULL at the end*.
- *A circular linked list can be a singly circular linked list or doubly circular linked list.*
- In the circular linked list the previous element stores the address of the next element and **the last element stores the address of the starting element**.
- The elements points to each other in a circular way which forms a circular chain.



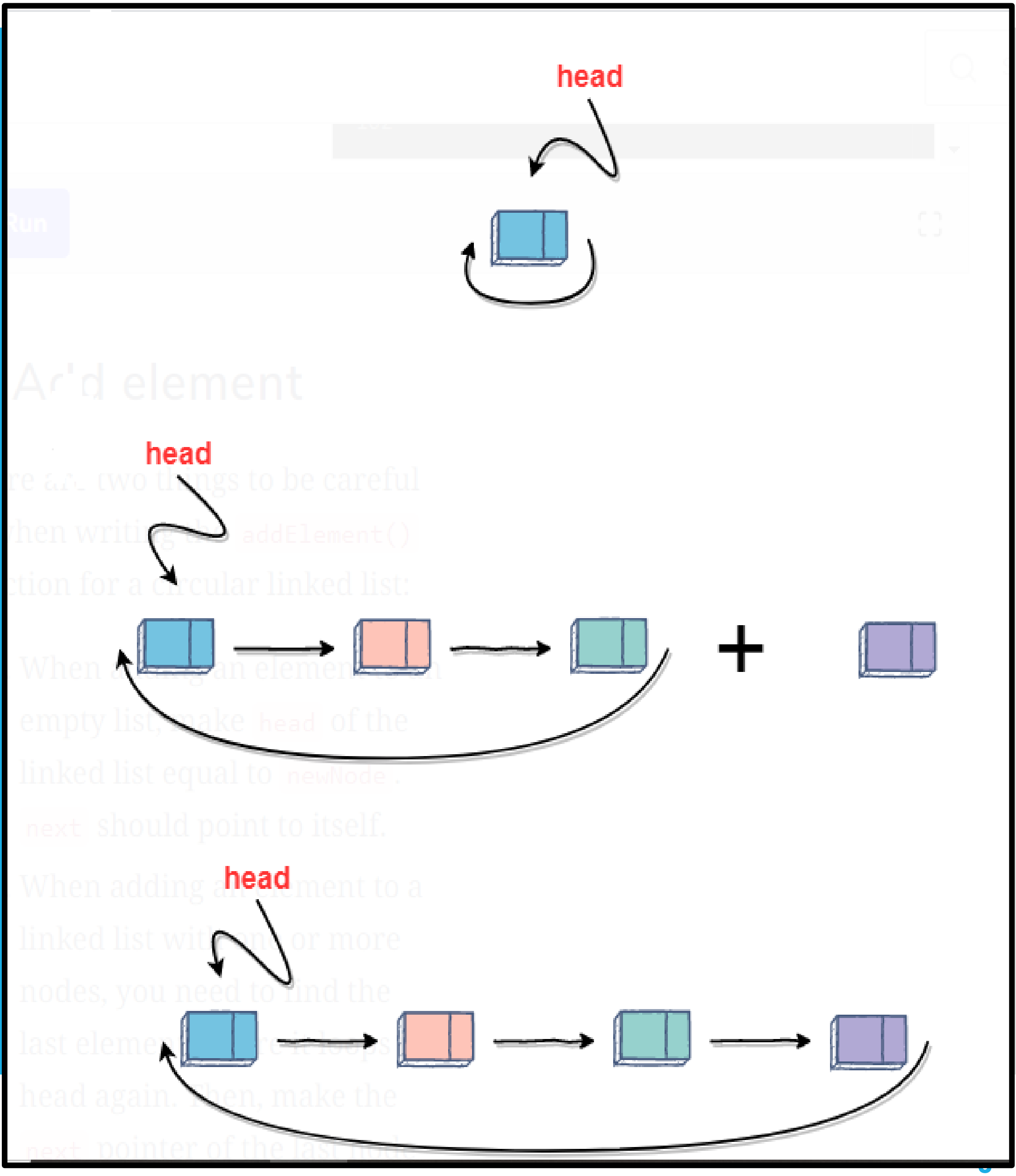

data
next
3
data
next
10
data
next
2
HEAD
Last Element points back to First

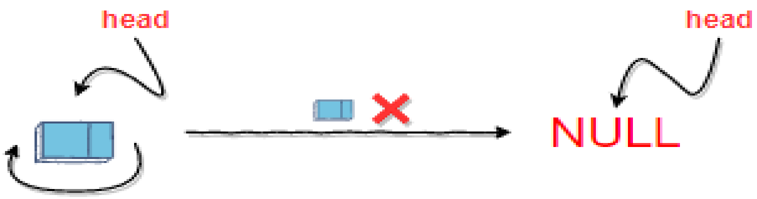
Add element

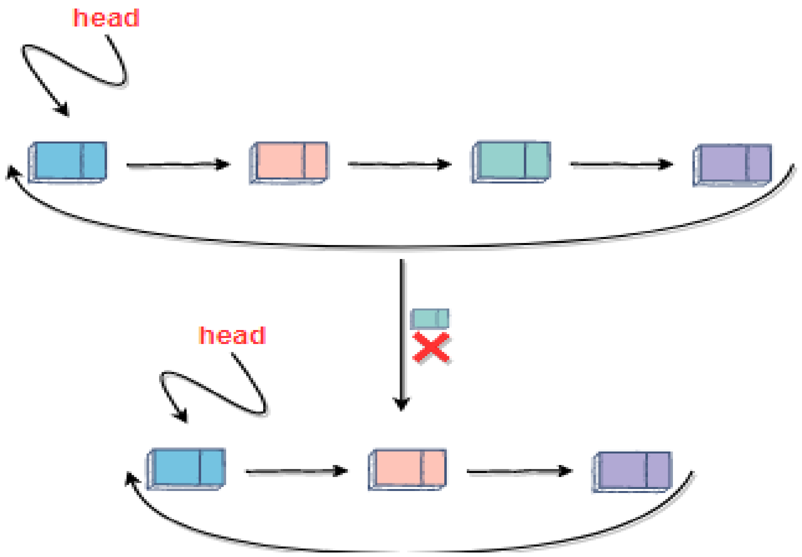
There are two things to be careful of when writing the addElement() function for a circular linked list:

When adding an element to an empty list, make head of the linked list equal to newNode. next should point to itself.

When adding an element to a linked list with one or more nodes, you need to find the last element before it loops to head again. Then, make the next pointer of the last node point to newNode and the next pointer of newNode point to head.





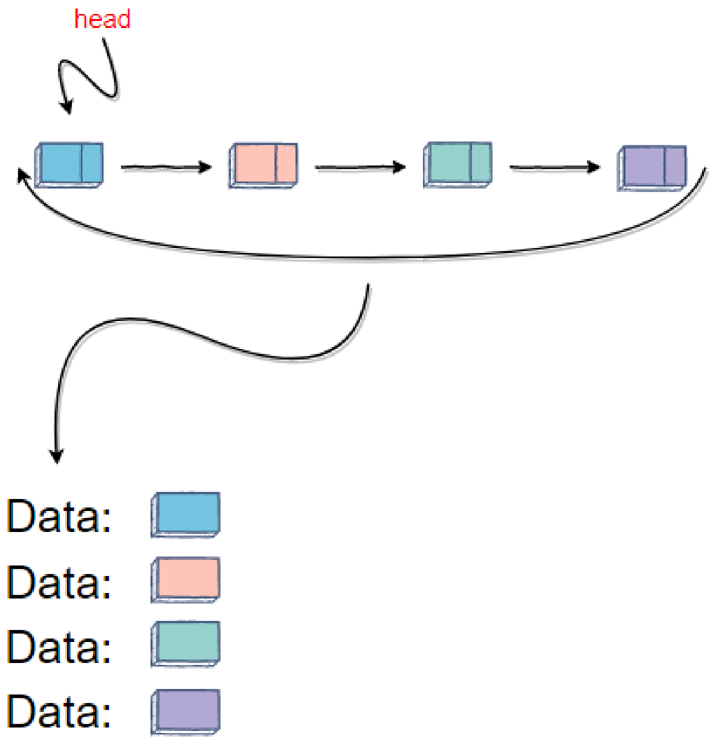


Delete element

When deleting an element, you need to be mindful of two things:

When you are deleting an element from a circular *linked list of size one,* you need to point the head pointer to Null.

When you are deleting an element from a list with two or more elements, you need to keep track of two pointers: current and previous. As soon as the current reaches the target node, make the previous node’s next point to the current node’s next. Then delete the current node.



Display list

Displaying the data of a circular linked list is similar to a normal linked list – you visit each node and print the data.

The only difference between the two lists is in the **method of termination**. When printing data of a normal linked list, the code should start displaying data from the head pointer and terminate as soon as null is hit.

Whereas, in a circular linked list, the code should start displaying data from the head pointer and terminate as soon as head is reached again.

THANK YOU!!!!!