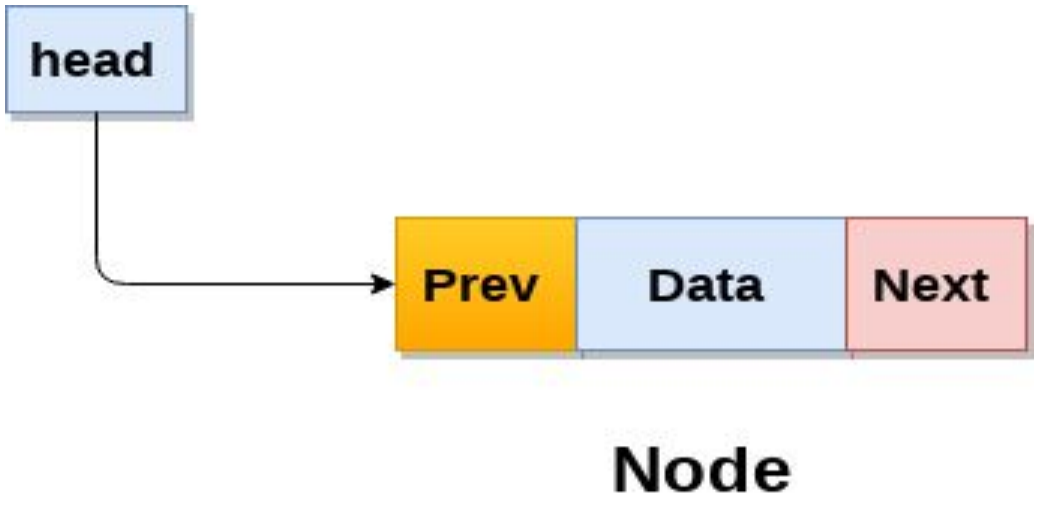
# Doubly Linked List

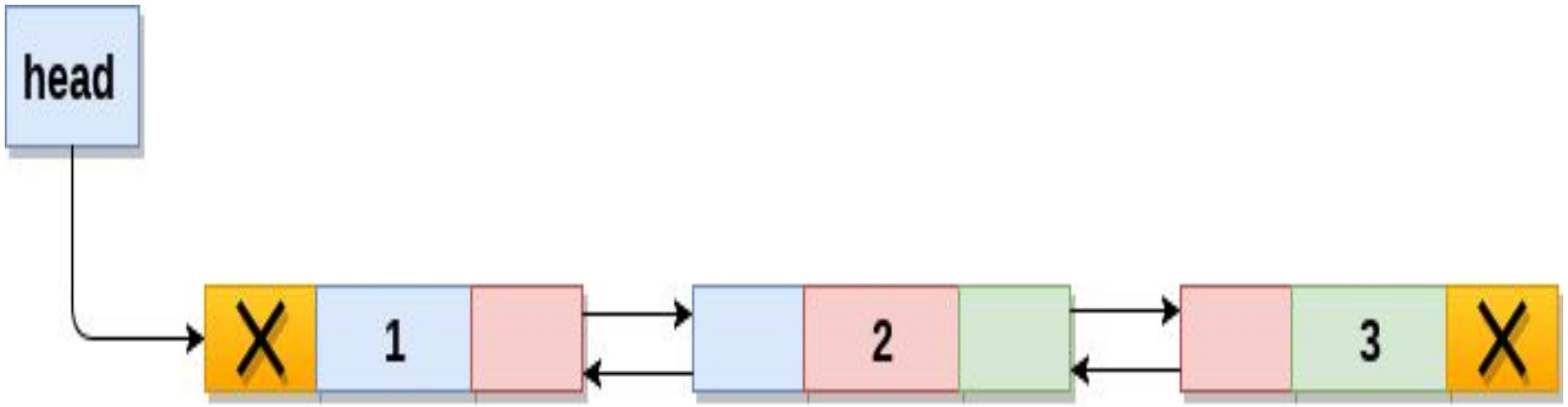
- Doubly linked list is a complex type of linked list in which a node contains a pointer to the previous as well as the next node in the sequence.
- Therefore, in a doubly linked list, a node consists of three parts: node data, pointer to the next node in sequence (next pointer) , pointer to the previous node (previous pointer).
- A sample node in a doubly linked list is shown in the figure.




Node

# Doubly Linked List

A doubly linked list containing three nodes is shown in the following image.




Doubly Linked List

# Creating a Node of Doubly Linked List

- Create a class for creating a node in a doubly linked list, with three attributes: the data, previous pointer and next pointer. The code looks like this:

```
class Node:
    def __init__(self, data):
        self.prev = None
        self.item = data
        self.next = None
```

# Creating Doubly Linked List Class

Create a doublyLinkedList class, that contains different functions to insert, delete and display elements of doubly linked list.

```
class doublyLinkedList:
    def __init__(self):
        self.start_node = None
```

# Creating a Doubly linked list with single node

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

class DoublyLinkedList:
        def __init__(self):
                self.head = None

LL = DoublyLinkedList()
LL.head = Node(3)
print(LL.head.data)
```

# Creation and Traversal of Doubly linked list

```
# A single node of a doubly linked list
class Node:
    def __init__(self, data):
        self.prev = None
        self.data = data
        self.next = None

# A Linked List class with a single head node
class DoublyLinkedList:
    def __init__(self):
        self.head = None
```

```
# creation method for the doubly linked list
    def create(self, data):
        newNode = Node(data)
        if(self.head==None):
            self.head = newNode

        else:
            temp=self.head
            while(temp.next!=None):
                temp=temp.next

            temp.next=newNode
            newNode.prev=temp
```

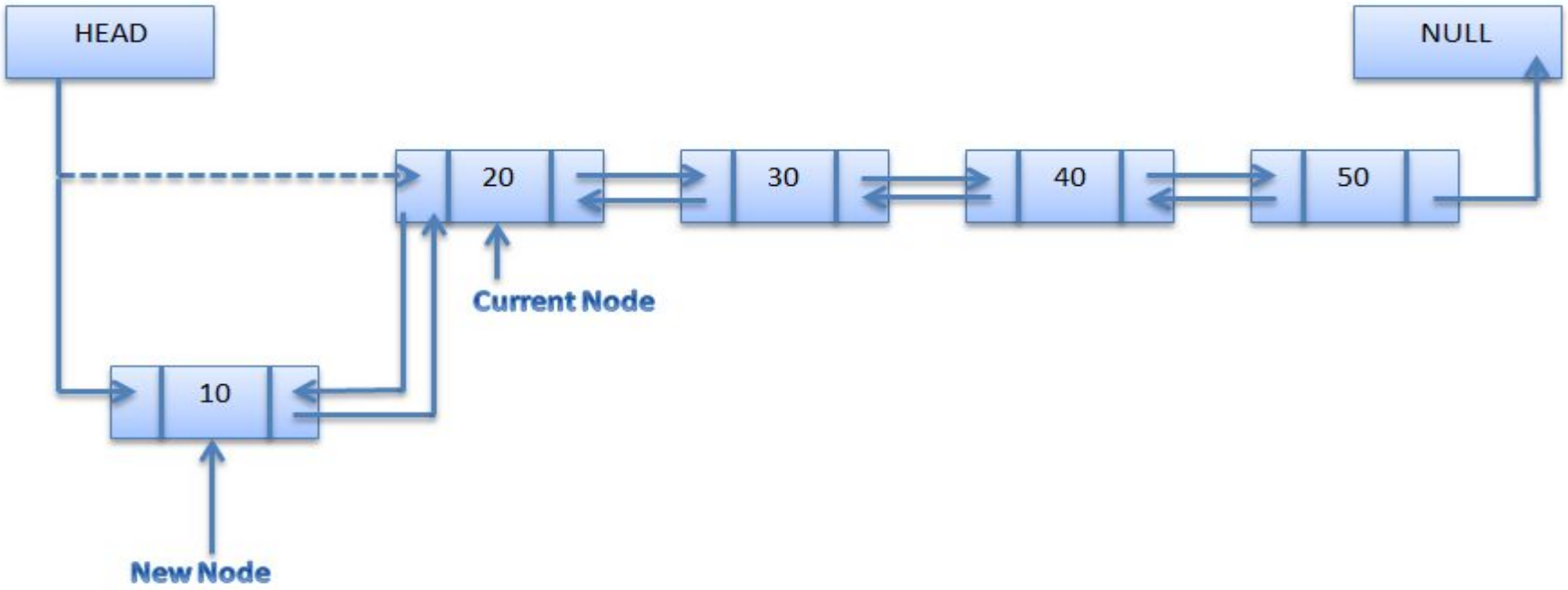
# Creation and Traversal of doubly linked list (contd..)

```
# print method for the linked list
    def printLL(self):
        current = self.head
        if(current!=None):
            print("The List Contains:",end="\n")
            while(current!=None):
                print(current.data)
                current = current.next
        else:
            print("List is Empty.")
```

```
# Singly Linked List with creation and
      print methods
LL = DoublyLinkedList()
LL.create(3)
LL.create(4)
LL.create(5)
LL.create(6)
LL.printLL()
```

# Insertion at the Beginning Linked List

- Insertion at the Beginning of Doubly Linked List



# Insertion at Beginning in Doubly linked list

```
# A single node of a doubly linked list
class Node:
    def __init__(self, data):
        self.prev = None
        self.data = data
        self.next = None

# A Linked List class with a single head node
class DoublyLinkedList:
    def __init__(self):
        self.head = None
```

```
# Insertion method for the doubly linked list at  beginning
    def insert_beg(self, data):
        newNode = Node(data)
        if(self.head==None):
            self.head = newNode

        else:
            newNode.next=self.head
            self.head.prev=newNode
            self.head=newNode
```

# Insertion at Beginning in Doubly linked list (contd..)

```
# print method for the linked list
    def printLL(self):
        current = self.head
        if(current!=None):
            print("The List Contains:",end="\n")
            while(current!=None):
                print(current.data)
                current = current.next
        else:
            print("List is Empty.")
```

```
# Singly Linked List with creation and
      print methods
LL = DoublyLinkedList()
LL.insert_beg(6)
LL.insert_beg(5)
LL.insert_beg(4)
LL.insert_beg(3)
LL.printLL()
```