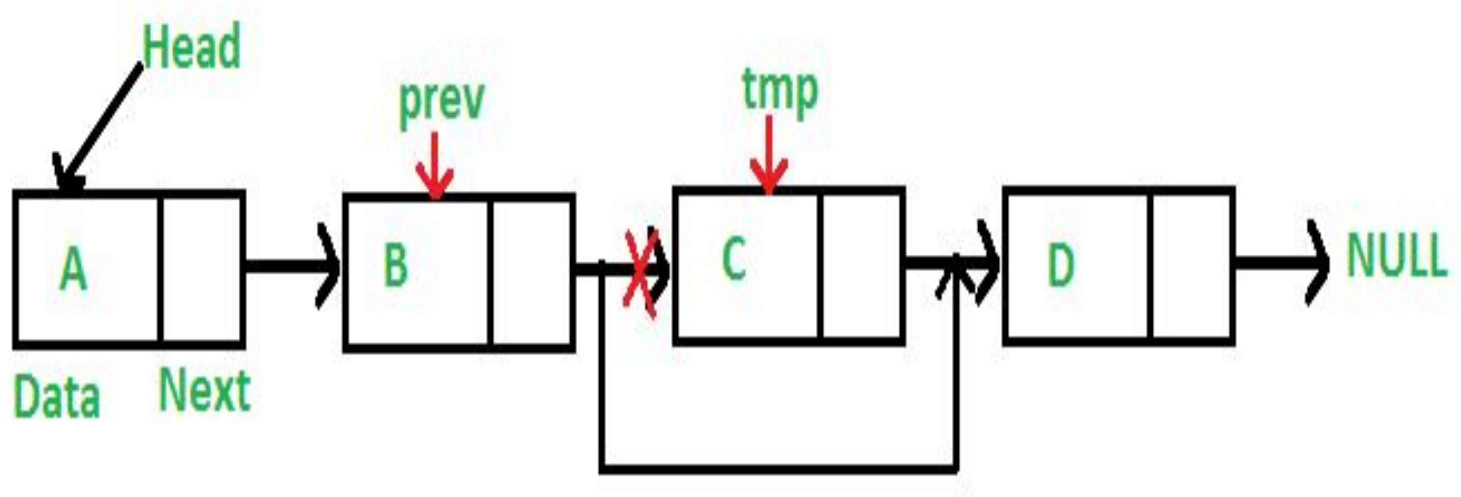
Deletion in Single Linked List (from position)

- **Deletion from position**



Deletion in Single Linked List (from position)

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

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

```
# create method for the linked list
  def create(self, data):
    newNode = Node(data)
    if(self.head):
      current = self.head
      while(current.next):
        current = current.next
      current.next = newNode
    else:
      self.head = newNode
```

Deletion in Single Linked List (from position)

```
# Deletion method from the linked list at
  given position
 def del_position(self, pos):
  if(pos<1):
    print("\nPosition should be >=1.")


  elif(pos==1):
    temp = self.head
    self.head = self.head.next
    print("the deleted element is",
  temp.data)
    temp = None
```

```
else:
  temp=self.head
  for i in range(1, pos):
    if(temp!=None):
      prev=temp
      temp=temp.next

  if(temp!=None):
  prev.next=temp.next
  print("the deleted element
 is", temp.data)
  temp=None
  else:
    print("\nThe position does not
  exist in link list.")
```

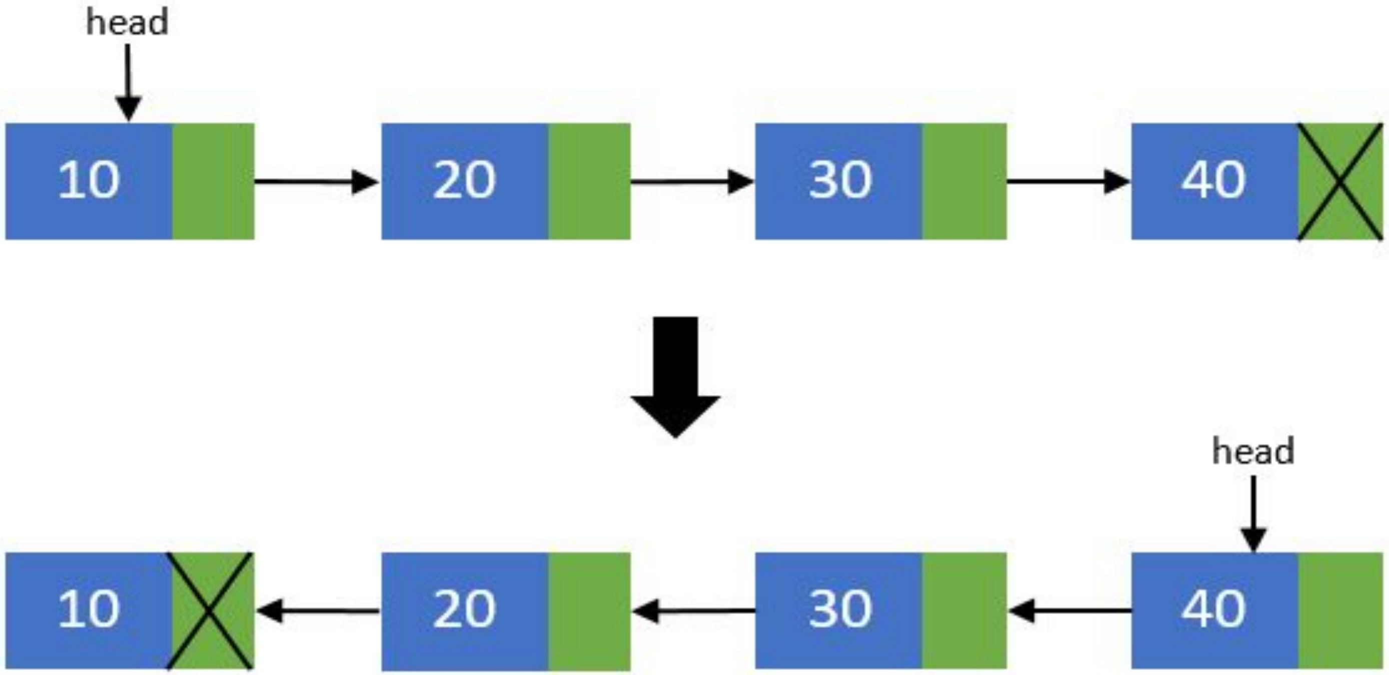
Deletion in Single Linked List (from position)

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

```
# Singly Linked List with deletion and print methods
LL = LinkedList()
LL.create(3)
LL.create(4)
LL.create(5)
LL.create(6)
LL.create(7)
LL.create(8)
LL.printLL()
LL.del_position(4)
LL.printLL()
```

Reverse of a Single Linked List

If the linked list has two or more elements, we can use three pointers to implement an iterative solution..



Reverse of a Single Linked List

```
# Method to Reverse the linked list
def reverse(self):
    if(self.head==None):
        print("List is Empty.")

    elif(self.head.next==None):
        print("Only one node is present in list")
    else:
        temp1 = self.head
        temp2=temp1.next
        temp3=temp2.next
        temp1.next=None
        while(temp3!=None):
            temp2.next=temp1
            temp1=temp2
            temp2=temp3
            temp3=temp3.next

        temp2.next=temp1
        self.head=temp2
```