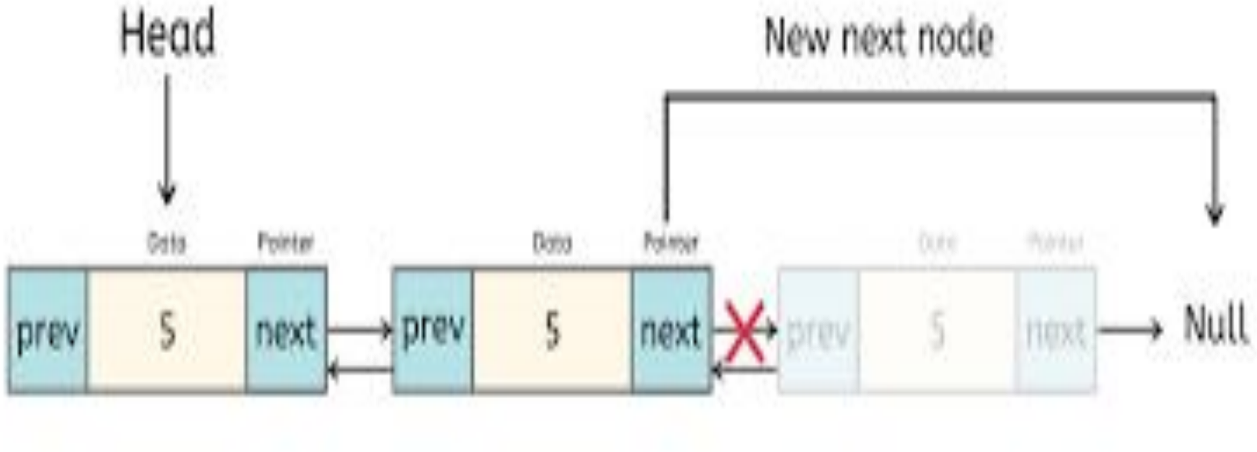
Deletion in Doubly Linked List (from end)

- **Deletion from end**



Deletion in Doubly Linked List (from end)

```
# 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
```

Deletion in Doubly Linked List (from end) (contd..)

```
#Delete last node of the list
  def del_end(self):
      if(self.head == None):
          print("Underflow-Link List is empty")

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

        prev.next=None
        print("The deleted element is", temp.data)
        temp = None
```

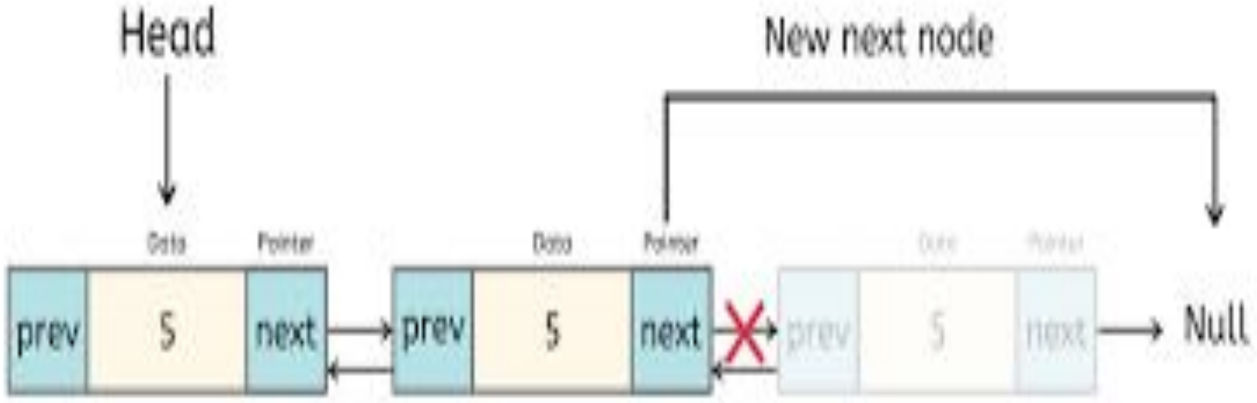
```
# print method for the linked list
    def printLL(self):
        current = self.head
        if(current!=None):
            print("The List
      Contains:",end="\n")
            while(current!=None):
                print(current.data)
                current = current.next
        else:
     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

Deletion in Doubly Linked List (from end) (contd..)

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

Deletion in Doubly Linked List (from position)

- **Deletion from end**



Deletion in Doubly Linked List (from position)

```
# 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
```

Deletion in Doubly Linked List (from position) (contd..)

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

    else:
        temp=self.head
        for i in range(1, pos):
            if(temp!=None):
                current=temp
                temp=temp.next
        if(temp!=None):
            current.next=temp.next
            temp.next.prev=current
            print("the deleted element is", temp.data)
            temp=None

        else:
            print("\nThe position does not exist in link list.")
```

Deletion in Doubly Linked List (from position) (contd..)

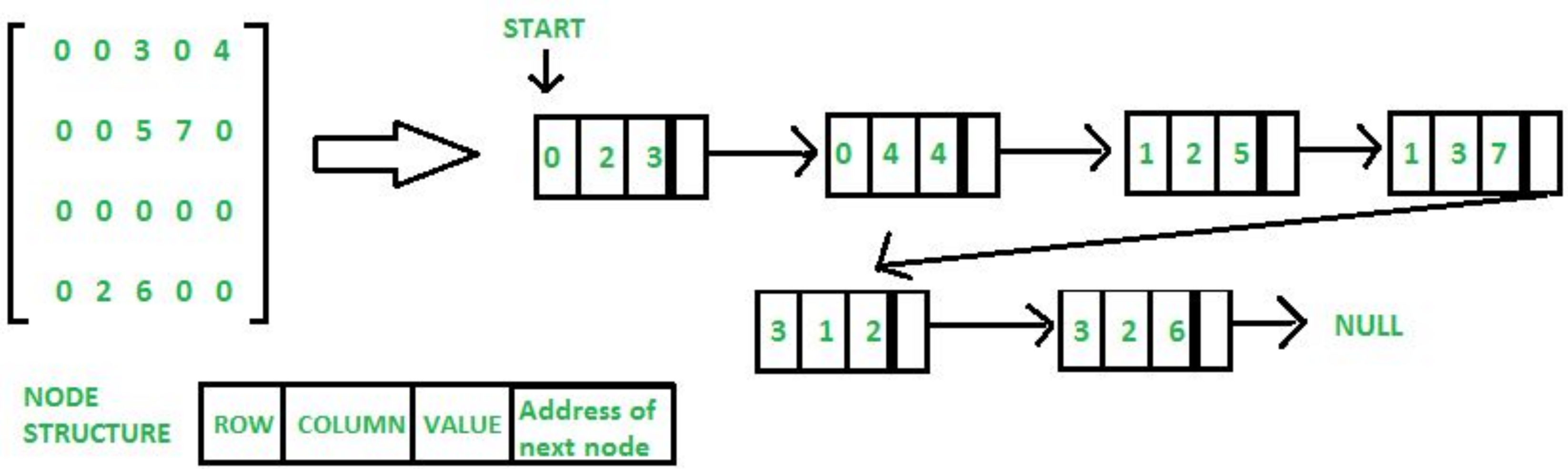
```
# print method for the linked list
def printLL(self):
    current = self.head
    if(current!=None):
        print("The List Contains:",end="\n")
        while(current!=None):
            print(current.data)
            current = current.next
    else:
        print("List is Empty.")
```

```
# Doubly Linked List with creation, deletion and print methods
LL = DoublyLinkedList()
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()
```

Linked Representation of Sparse Matrix

In linked list, each node has four fields. These four fields are defined as:

- **Row:** Index of row, where non-zero element is located
- **Column:** Index of column, where non-zero element is located
- **Value:** Value of the non zero element located at index – (row , column)
- **Next node:** Address of the next node



$$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix}$$

Polynomial Representation

Polynomials

Polynomials are the algebraic expressions which consist of exponents and coefficients.

Example -
$10x^2 + 26x$, here 10 and 26 are coefficients and 2, 1 is its exponential value.

Polynomial can be represented in the various ways. These are:

- By the use of arrays
- By the use of Linked List

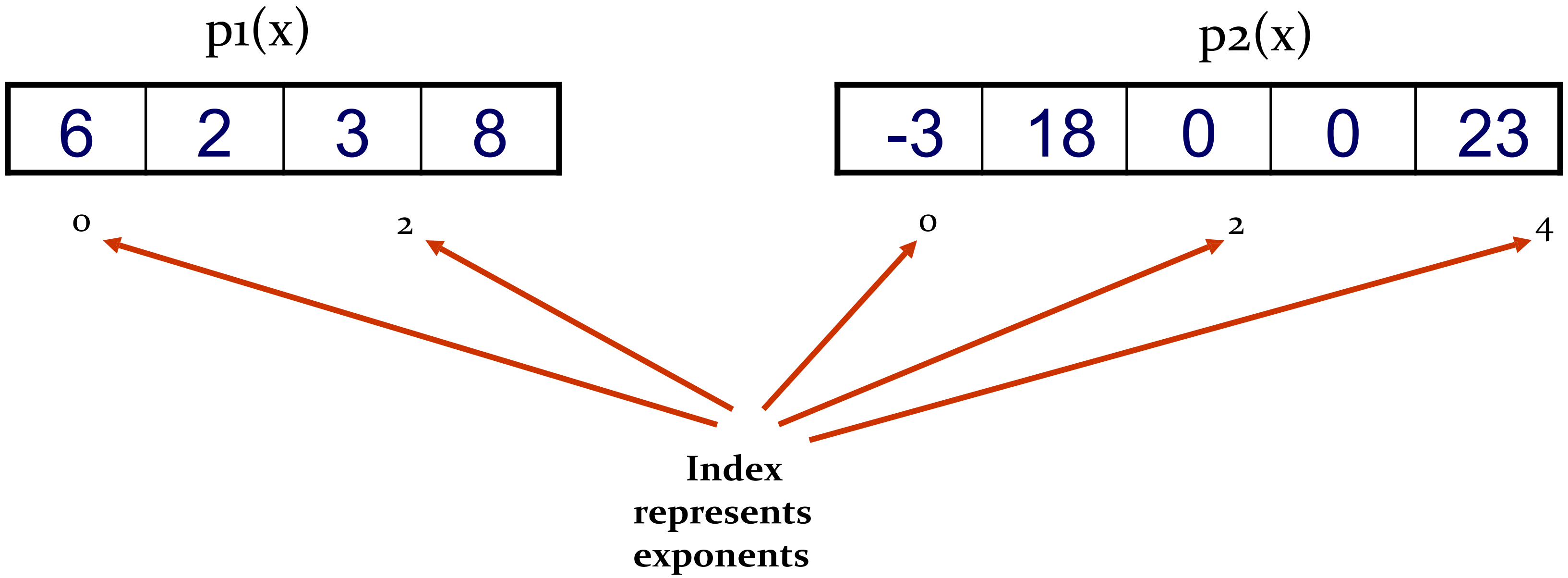
Polynomial Representation

Polynomial can be represented

- By the use of arrays
- By the use of Linked List

Polynomial (Array Representation)

- Array Representation:
  - $p1(x) = 8x^3 + 3x^2 + 2x + 6$
  - $p2(x) = 23x^4 + 18x - 3$



p1(x)

6	2	3	8
0		2	

p2(x)

-3	18	0	0	23
0		2		4

Index represents exponents

Polynomial (Array Representation)

- This is why arrays aren’t good to represent polynomials:

- $p3(x) = 16x^{21} - 3x^5 + 2x + 6$

6	2	0	0	-3	0	…………	0	16

WASTE OF SPACE!