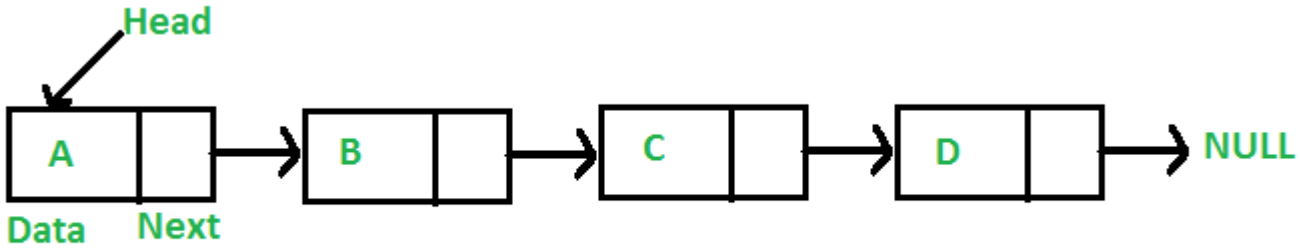
Linked List

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:



In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

In an array, each and every element is adjacent to each other.

0	1	2	3	4	5
21	3	5	4	1	8
#58	#59	#60	#61	#62	#63

A ‘linked list’ is similar to any array and it has box, which is called node. This node has a value a pointer or reference to the next node. Similarly, this will be a value and a reference to the next node. The last node will not have a reference, it would be *NULL*

The address of the node can be different places or random.

0	1	2	3	4	5
21	3	5	4	1	8
#55	#590	#65	#62	#78	#13

The first node of a linked list is called **‘head’** and the last node is called **‘tail’**. Tail of the linked list will not have a reference value. Arrays and linked list of one very important property. In array you do direct indexing i.e., if we have an array A,

A = [5 8 10 1 2 3]

A [3] will give us the value at the 3rd index.

But in a linked list, **we only know the head of the linked list** and we cannot do direct indexing, we can only simulate indexing.

How to represent real life in CODE?

To represent a ‘Node’, we use classes. Node has a data/value

Linked List in Implementation:

****

Linked List will have a data part and a next part. So, we need to define a node. We can write a `class node`, which will have `def __init__ (self, x):`

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

We have set the data and we have set the data to None.

`next` is the part of the node which will become a path to the next node in the linked list.

If we say, `head = node (10)`, this means that we have created a head and it’s value is 10.

If we say `head.next = node (20)`, then, the next which was none, will now become a reference point for the next node with its value 20.

`head.next = node (50)`, the node at value 20 which was none, will now become a reference point for the next node with value 50.

CODE:

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

if __name__ == "__main__":
    head = Node(100)
    head.next = Node(50)
    head.next.next = Node(200)

    print(head.data)
    print(head.next.data )
    print(head.next.next.data)
```

Operations:

How to add an element at the end?

let's suppose we have a linked list as

50| 100| 20| 15|Null

To add 500|X at the end, we need to know the tail node. So, we need to put the pointer at head. We can say, cur = cur.next

Since the next node is also not None, the current will move to the next node until it comes to the tail node were the next is None. Once we go to the tail node, we have to say that tail node will be your next node (Node(50)).

```
head = None
def addNodeAtEnd(x):
    global head

    if head is None:
        head = Node(x)
        return

    cur = head
    # as long as cur.next is none, the current will move forward
    while cur.next != None:
        cur = cur.next

    cur.next = Node(x)

if __name__ == "__main__":

    head = None
    addNodeAtEnd(10)
    addNodeAtEnd(100)
    addNodeAtEnd(200)
    addNodeAtEnd(500)

    print("The head of thee linked list contains,", head.data)
    print("The head of thee linked list contains,", head.next.data)
    print("The head of thee linked list contains,", head.next.next.data)
    print("The head of thee linked list contains,", head.next.next.next.
data)
```

How to print a linked-list

1| 2| 3| 4|

Using while loop

```
def printLinkedList():
    global head

    cur = head
    while cur != None:
        print(cur.data)
        cur = cur.next
if __name__ == "__main__":

    printLinkedList()
```

How to add node at the head position?

if we want to add a new node at the head position, node = Node(1)

if your head is None,

```
def addNodeatBegining(x):
    global head

    if head is None:
        head = Node(x)
        return

    node = Node(x)
    node.next = head
    head = node

if __name__ == "__main__":

    addNodeatBegining(5)
    addNodeatBegining(25)
```

How to add something in the middle of the linked list.

1 2 3 4

If we want to add something after n = 2,

New linked list = 1 2 5 3 4

head

Taking count = 1, we can put a ‘cur’ at head (cur = head). Whenever we move cur forward, we will also keep increasing the count. So, if cnt is not equal to n, we can move the ‘cur’ forward.

If we create a node (100), after 2, then we can say cur.next = n and at the same time, the new node should point to 3 as well. So,

next = cur.next
cur.next = n
n.next = next

```
def addNodeInMiddle(n, x):
    if n == 0:
        return

    global head

    cnt = 1
    cur = head
    while cnt != n:
        cur = cur.next
        cnt += 1

    cur.neighbor = cur.next
    new_node = Node(x)

    cur.next = new_node
    new_node.next = cur.neighbor


if __name__ == "__main__":

    head = None
```

```
    addNodeatBegining(5)
    addNodeatBegining(25)

    addNodeAtEnd(10)
    addNodeAtEnd(100)
    addNodeAtEnd(200)
    addNodeAtEnd(500)

    addNodeInMiddle(2, 80)
    addNodeInMiddle(6, 80)
```

How to delete the tail of a linked list.

For any linked list if you want to delete the tail node, taking two pointers, ‘cur’ and ‘pre’ and then check if ‘cur’ is None or not. If ‘cur’ is not None, we will move current forward and the cur will move to the next node and ‘pre’ would move forward along with ‘cur’.

When the cur is None, the ‘pre.next will be None and delete the last node.

```
def deleteTailNode():
    global head
    prev = Node
    cur = head

    while cur.next != None:
        prev = cur
        cur = cur.next

    prev.next = None

if __name__ == "__main__":

    addNodeatBegining(5)
    addNodeatBegining(25)

    addNodeAtEnd(10)
    addNodeAtEnd(100)
    addNodeAtEnd(200)
    addNodeAtEnd(500)

    printLinkedList()
```

```
    print("before deletion")

    deleteTailNode()
    deleteTailNode()
    deleteTailNode()
    print("after deletion")
```

How to delete the head of the linked list.

We can say head = head.next

```
def deleteAtHead():
    global head
    if head is None:
        return

    head = head.next
```

H.W:

How to delete node in the middle?

https://www.youtube.com/watch?v=M3E_y4XNdA8