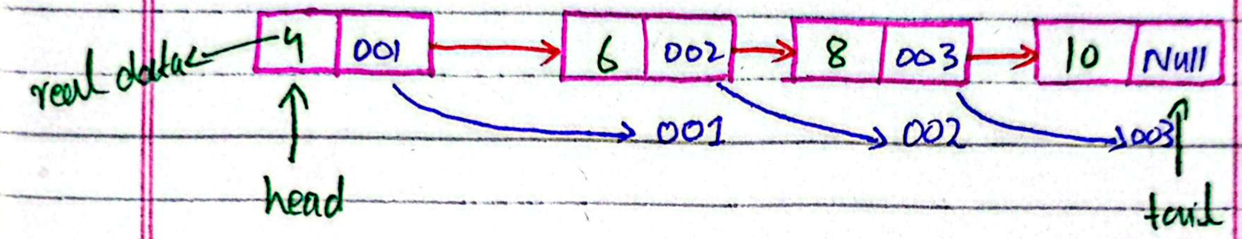
{DSA}

linkedlist

linkedlist store the data ramdomly in memory and conneet with each other therrough pointers. ->

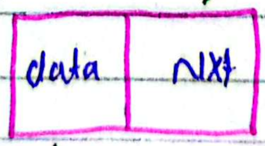


Here each node are conected with other by the help of pointer.

where is Nod:-

The nod is actuly contain data and Next pointer.

so it's Not only data but also nxtptr



• Nod is basically like dict lik.

{
"value" : 50
"next" : None
}

we make seperat NOD class it only make Nod.

It's completly sepert bcz it contain own data & we can connect it everywhere first Node class:

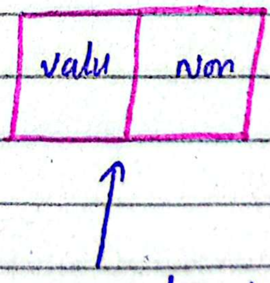
```
class Node:
        def__init__(self, value):
                self.value = value
                self.next = Non
```

this well only create the single Node inside memory.

→Now Creat first Linkedlist class:

```
class  LinkedList :
        def __init__(self, vale):
                new_node = Node(vale)
```

- It create the Linkedlist with only single Node which have valeue and Non in next. {how we know?}
- It works with O(1) ⟶ constant time complexity



we assigne it in Linhedlist class at creation time.

Tip

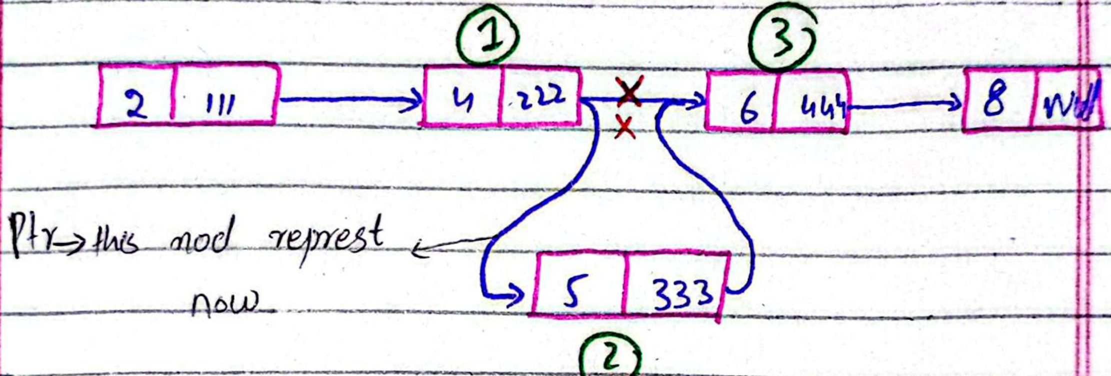
for code and more practical understanding move→ GitHub repo DSA_basic_to_advance

Insertion in singl LL :-

when we insert value in Linkedlist the process is :

- first make the new_Node usin Node class then.
- Assign the pointer of previous & nxt accordingly

e.g :-

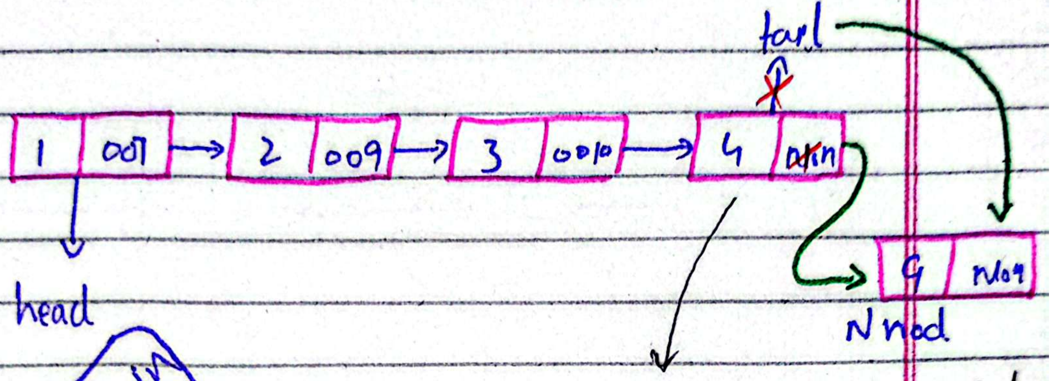


① this Node pointer represent the ③ Node but when first we inset then the pointer of ① Node represent ② Node and the pointer of ② Node ~~③~~ present the ③ Node

TRY to understand

All these work done using append class

If we wanna insert in last the the Next pointer of the tail assigne to newNode and tail—> New Nod :-



Insertion in last

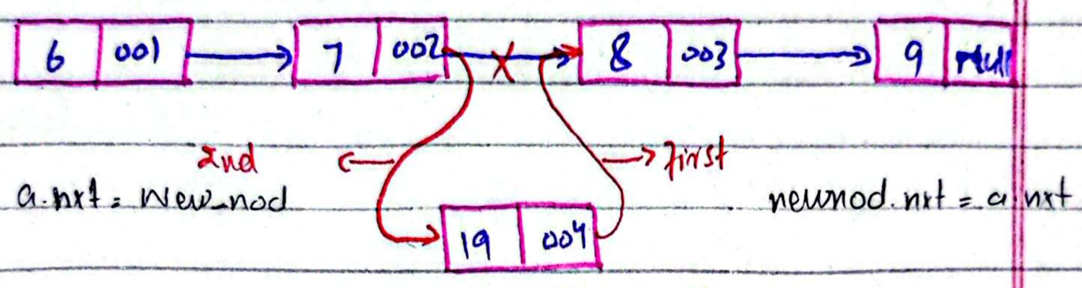
we have the code lik

```
self. tail. nxt = Nnod
self. tail = Nnood
```

- Now just same case for insertion at end/head and same process we use.
- first check likedlist then perform this step.
- if it's empty then assigne head & tail to that Nod.
- just think like real life case....

Insertion at specific index :-

In this case the data and positionll given over jobe is to add in such idx.



Three thing'll be given (self, data, position)
4 3

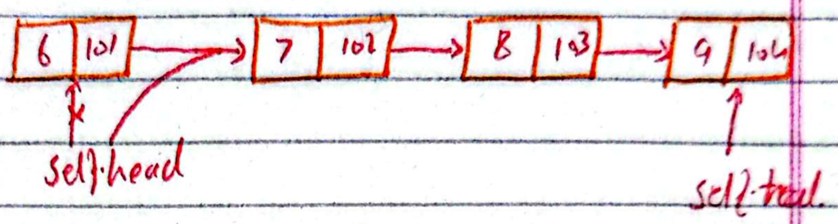
- let start the loop from 1 to position -1 => 3-1 = $\boxed{2}$
- So then add it

```
for i in range(1, position-1)
        a = a.nxt
new_Nod.nxt = a.nxt
a.nxt = New_Node.
```

{find the code in GitHub repo ...}

Deletion at Beginning :-

let say the 1st Nod is self.head. and we'll we'll remove it and assigne head to nxt Node.



- Now we remove 1st Nod so assigne self.head to it nxt
- Take temprary veriable (a) and store 1st Node then a.nxt = Non.
- So the self.head'll start from 2nd Node 1st'll be remove.

```
a = self.head
self.head = a.nxt
a.nxt = Non
```

Disconnect this Node.

Sumeary :-

Assigne self.head to next Node and Disconnect 1st Node that's it

Go to Github for Code