Data Science & AI
&
NIC - Param

Python-For Data Science

Linked List

Lecture No.- 03

By- Pankaj Sharma Sir

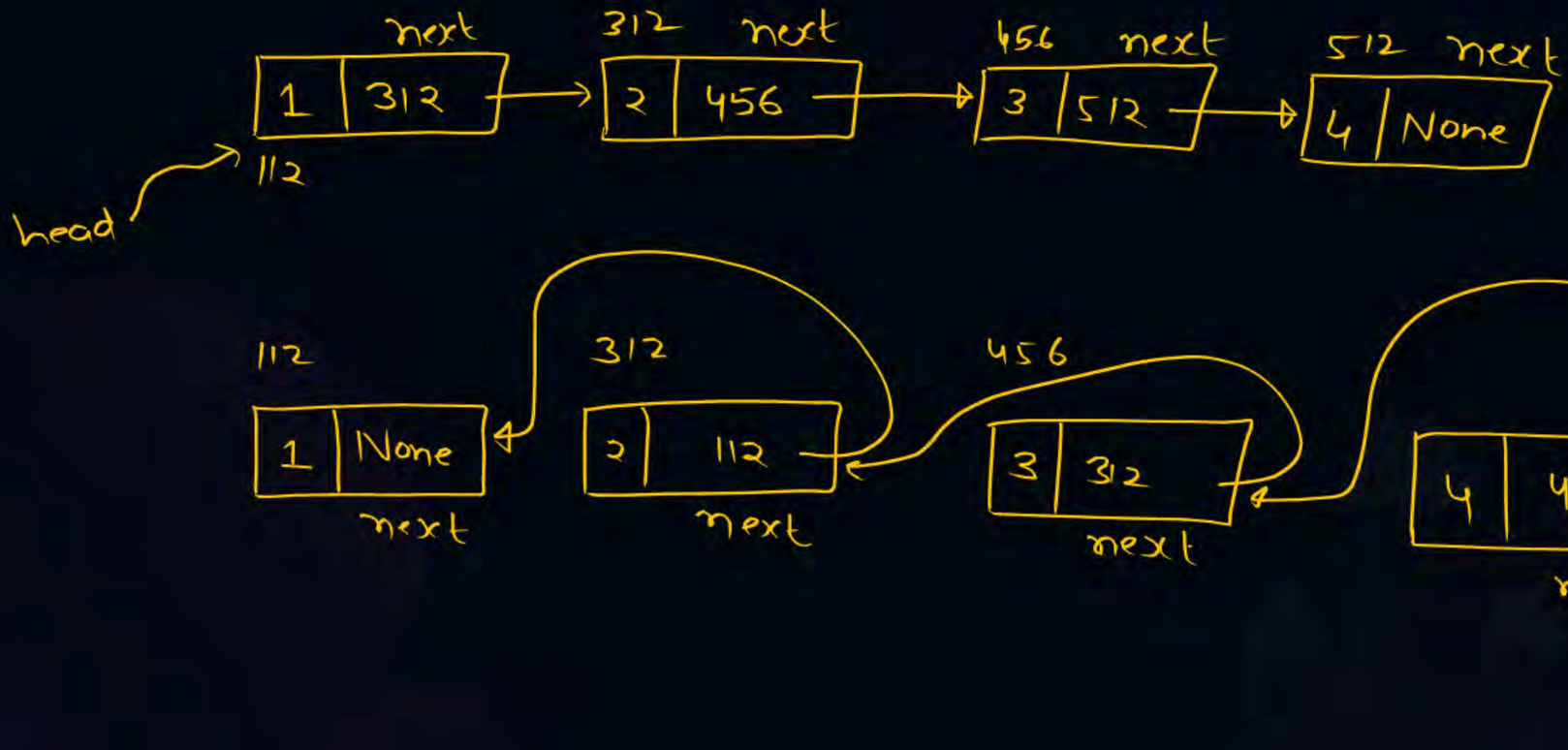
Recap of Previous Lecture

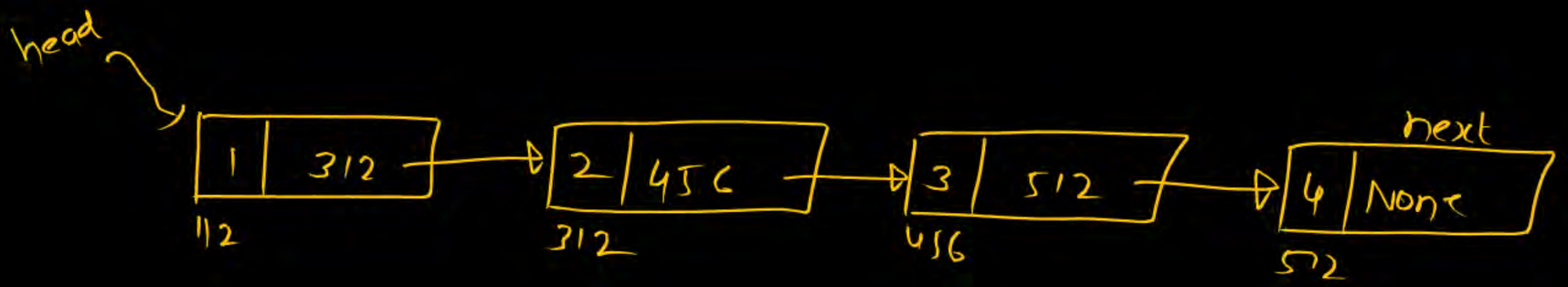
Topic Linked List Part-02

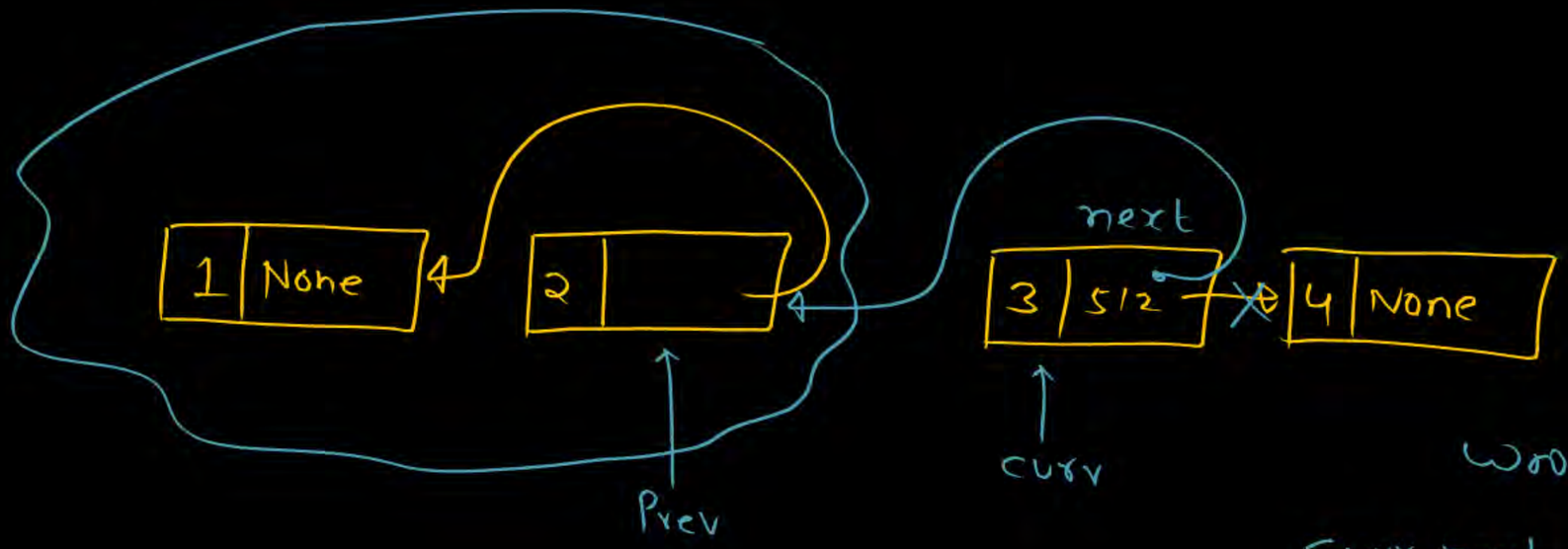
Topics to be Covered

Topic Linked List Part-03

Topic : Linked List

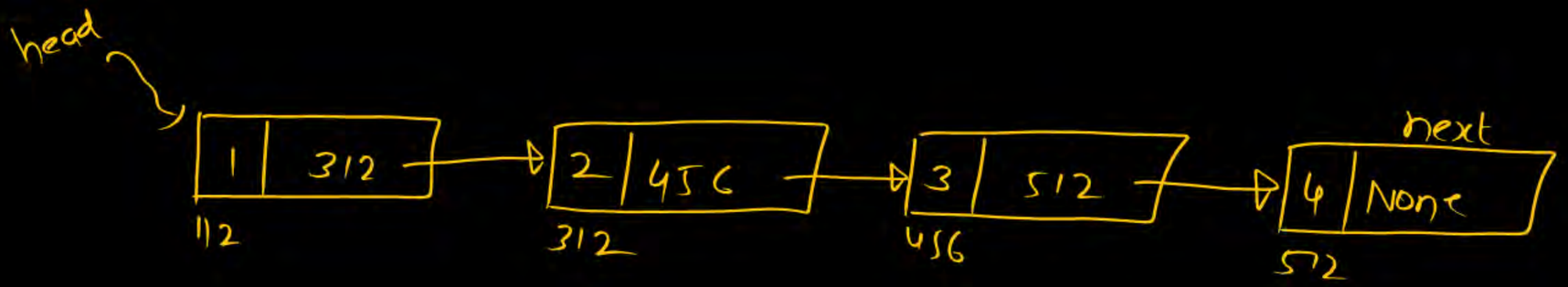


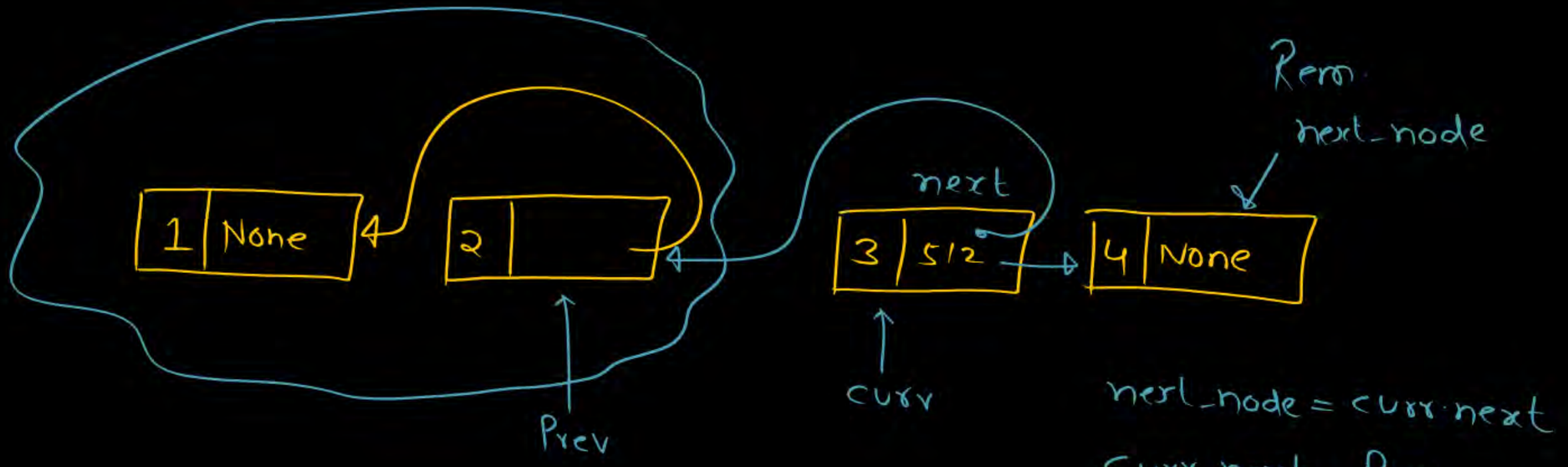




Wrong ✗

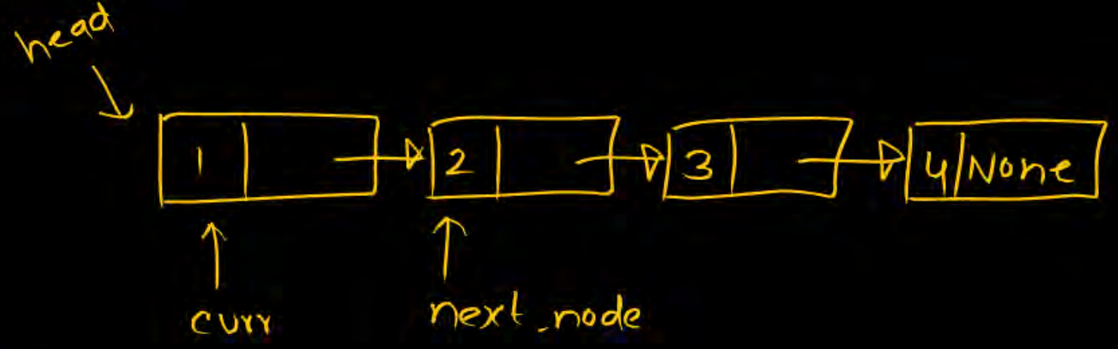
curr.next = Prev





next_node = curr.next

curr.next = Prev

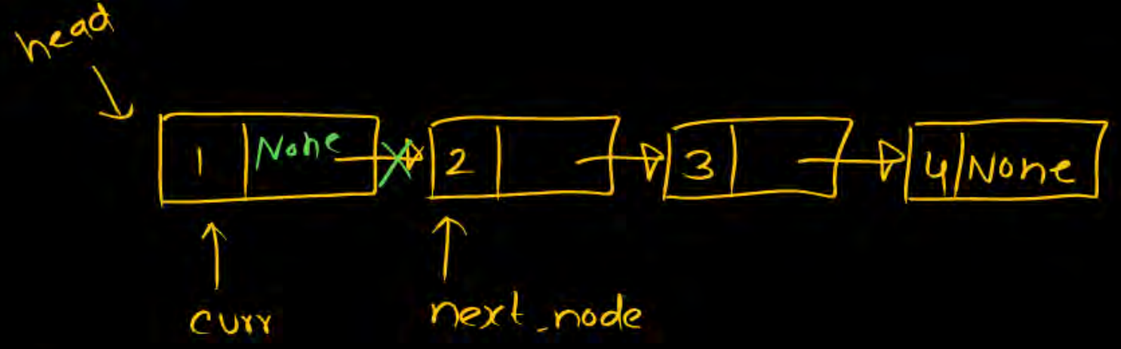


Prev = None

Curr = head

(i) next_node = curr.next

(ii) curr.next = Prev

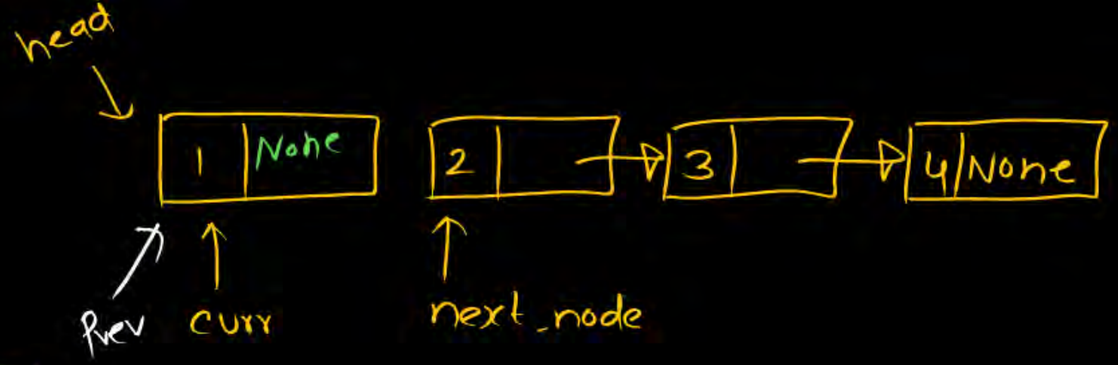


Prev = None

Curr = head

(i) next_node = curr.next

(ii) curr.next = Prev



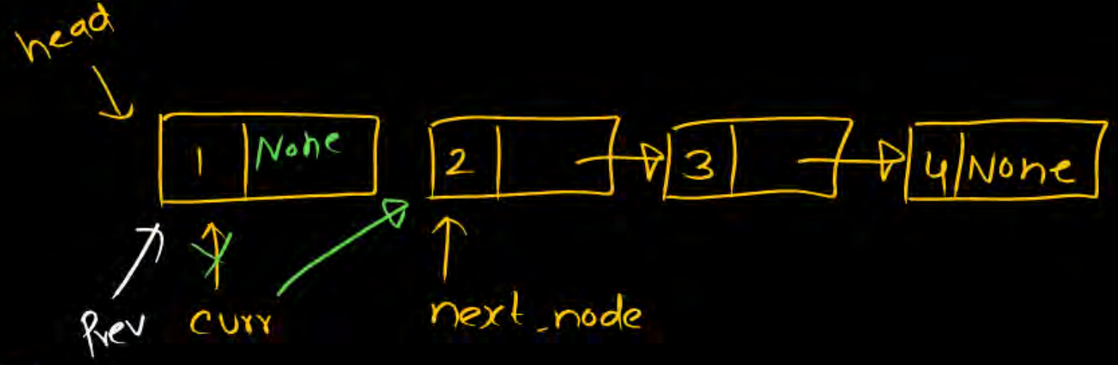
Prev = None

curr = head

(i) next_node = curr.next

(ii) curr.next = Prev

(iii) Prev = curr



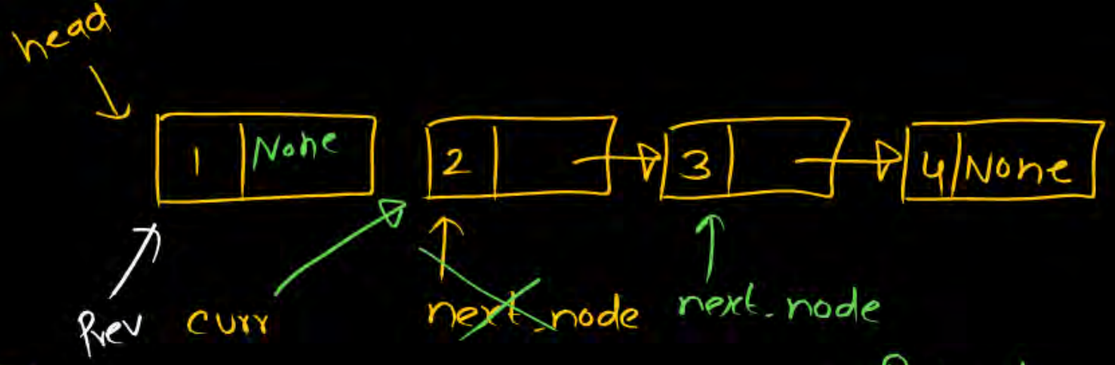
Prev = None
curr = head

(i) next_node = curr.next

(ii) curr.next = Prev

(iii) Prev = curr

(iv) curr = next_node



Prev = None

curr = head

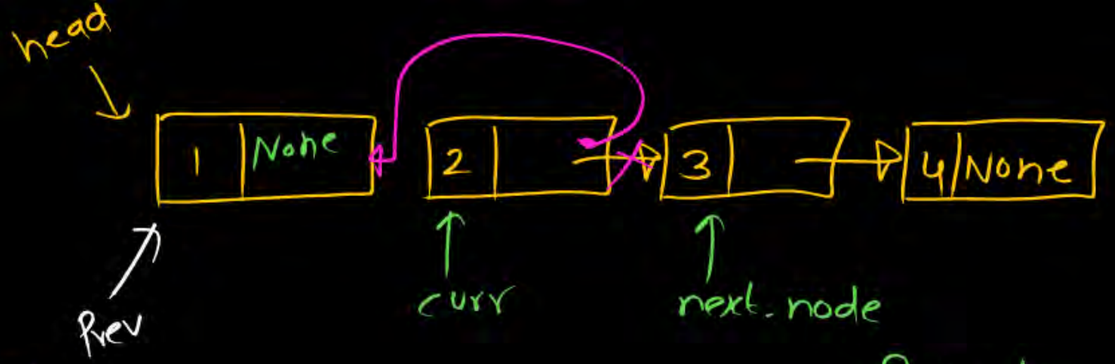
Repeat

(i) next_node = curr.next

(ii) curr.next = Prev

(iii) Prev = curr

(iv) curr = next_node

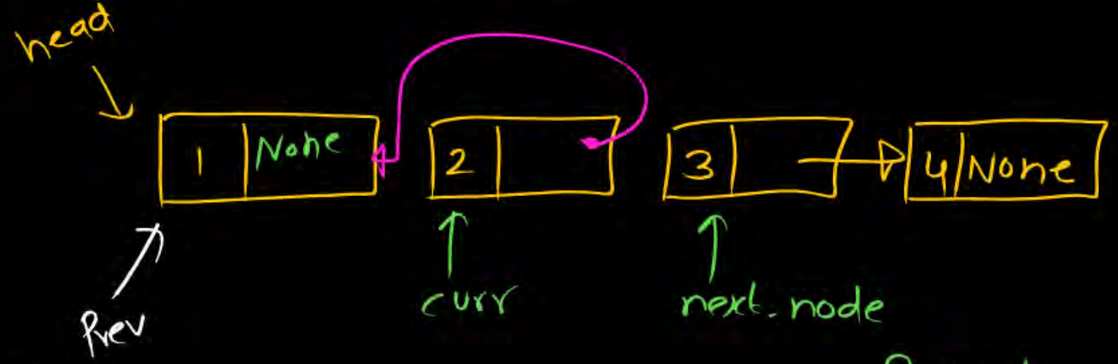


Prev = None
curr = head

(i) next_node = curr.next
(ii) curr.next = Prev
(iii) Prev = curr
(iv) curr = next_node

Repeat

(i) next_node

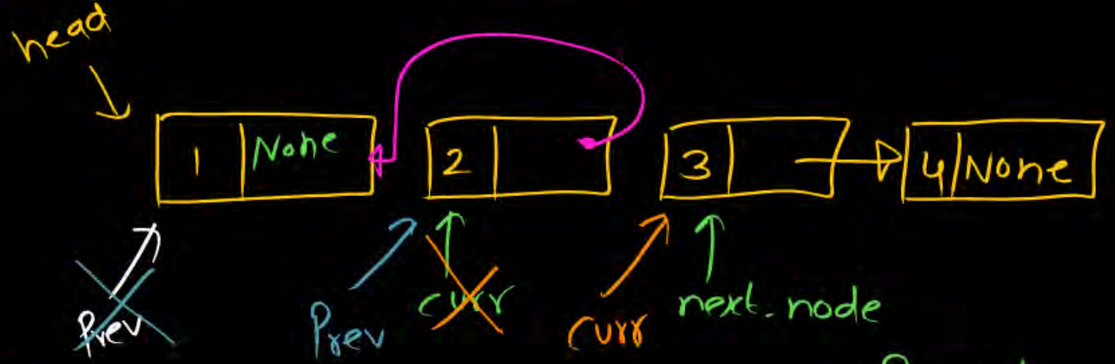


Prev = None
Curr = head

(i) next_node = curr.next
(ii) curr.next = Prev
(iii) Prev = curr
(iv) curr = next_node

Repeat

(i) next_node
(ii)

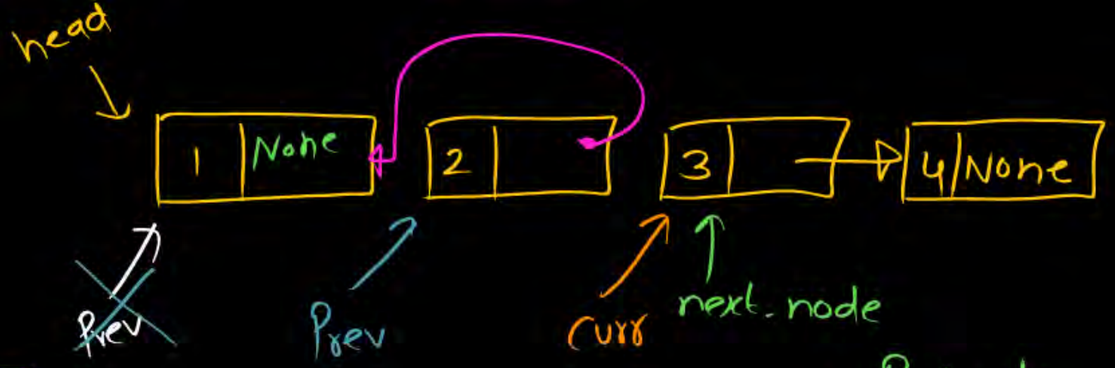


Prev = None
curr = head

(i) next_node = curr.next
(ii) curr.next = Prev
(iii) Prev = curr
(iv) curr = next_node

Repeat

(i) next_node
(ii)
(iii)
(iv)

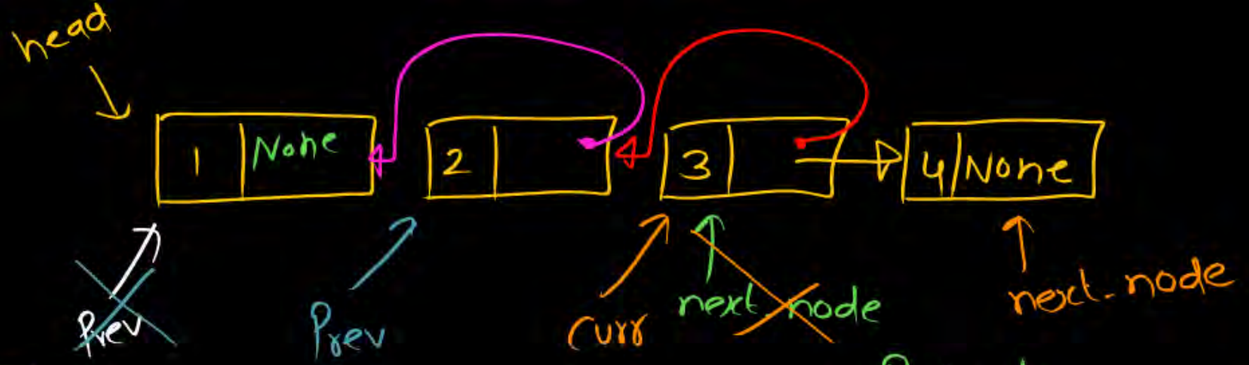


Prev = None
curr = head

(i) next_node = curr.next
(ii) curr.next = Prev
(iii) Prev = curr
(iv) curr = next_node

Repeat

(i) next_node
(ii)
(iii)
(iv)



Prev = None
curr = head

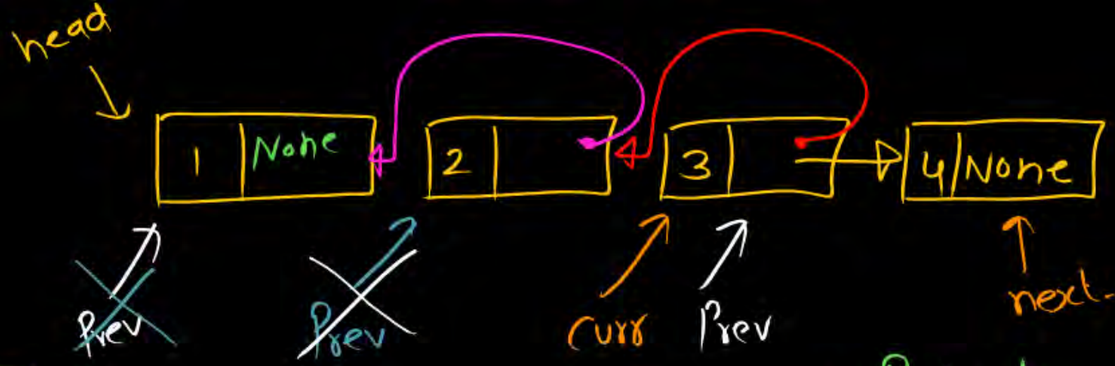
(i) next_node = curr.next
(ii) curr.next = Prev
(iii) Prev = curr
(iv) curr = next_node

Repeat

(i) next_node
(ii)
(iii)
(iv)

Repeat

(i)
(ii)



Prev = None
curr = head

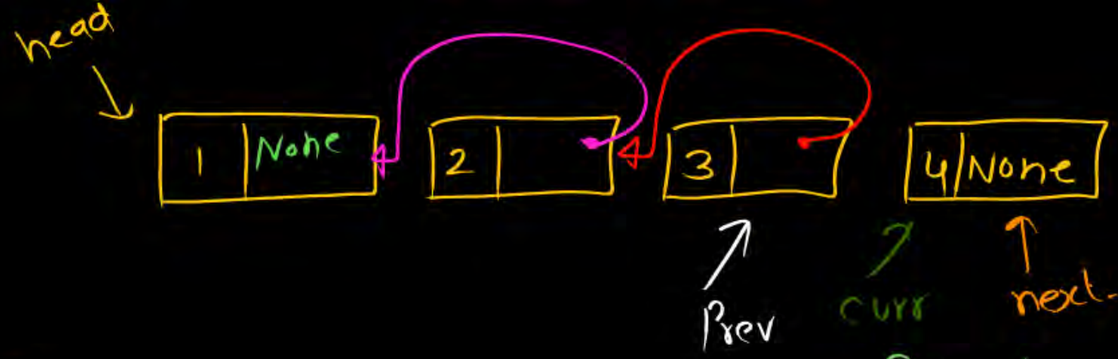
(i) next_node = curr.next
(ii) curr.next = Prev
(iii) Prev = curr
(iv) curr = next_node

Repeat

(i) next_node
(ii)
(iii)
(iv)

Repeat

(i)
(ii)
(iii)



Prev = None
curr = head

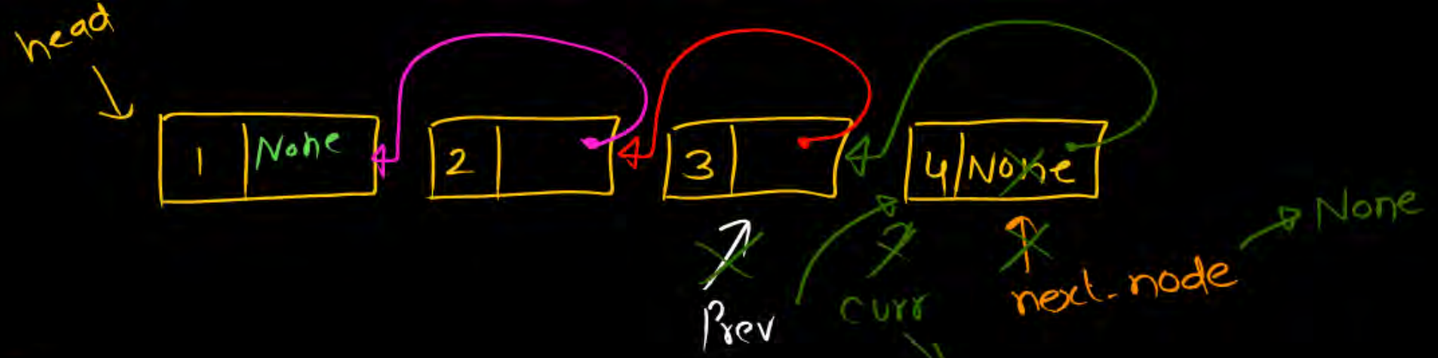
(i) next_node = curr.next
(ii) curr.next = Prev
(iii) Prev = curr
(iv) curr = next_node

Repeat

(i) next_node
(ii)
(iii)
(iv)

Repeat

(i)
(ii)
(iii)
(iv)



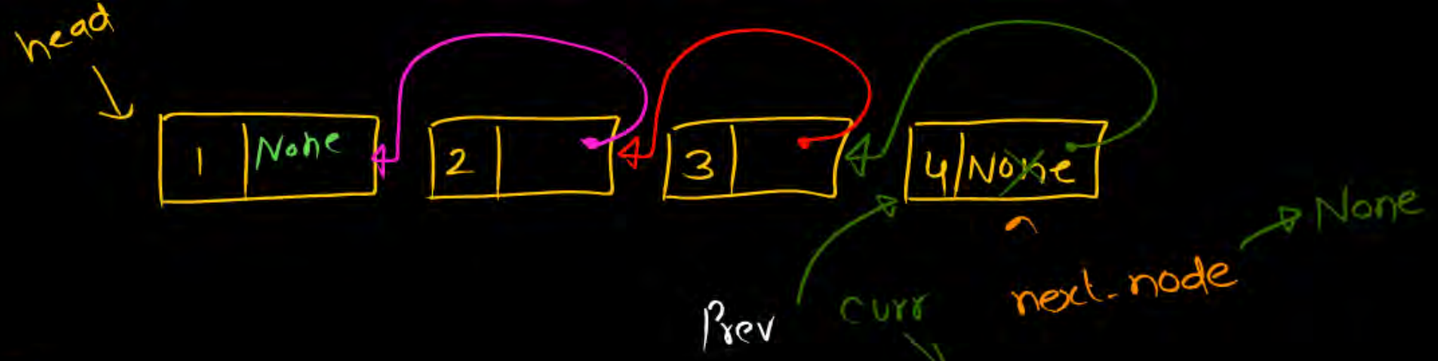
Prev = None
curr = head

(i) next_node = curr.next

(ii) curr.next = Prev

(iii) Prev = curr

(iv) curr = next_node



Prev = None
curr = head

(i) next_node = curr.next

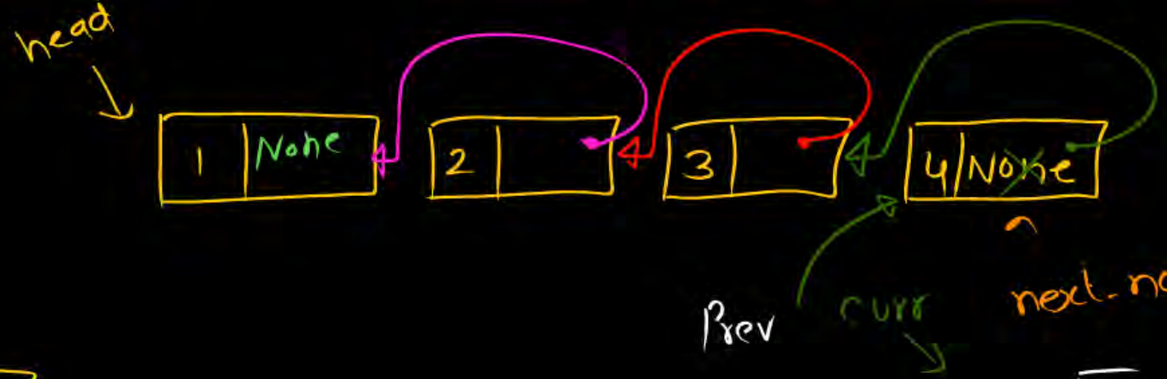
(ii) curr.next = Prev

(iii) Prev = curr

(iv) curr = next_node

(I)

(II)



Prev = None
curr = head

(i) next_node = curr.next
(ii) curr.next = Prev
(iii) Prev = curr
(iv) curr = next_node

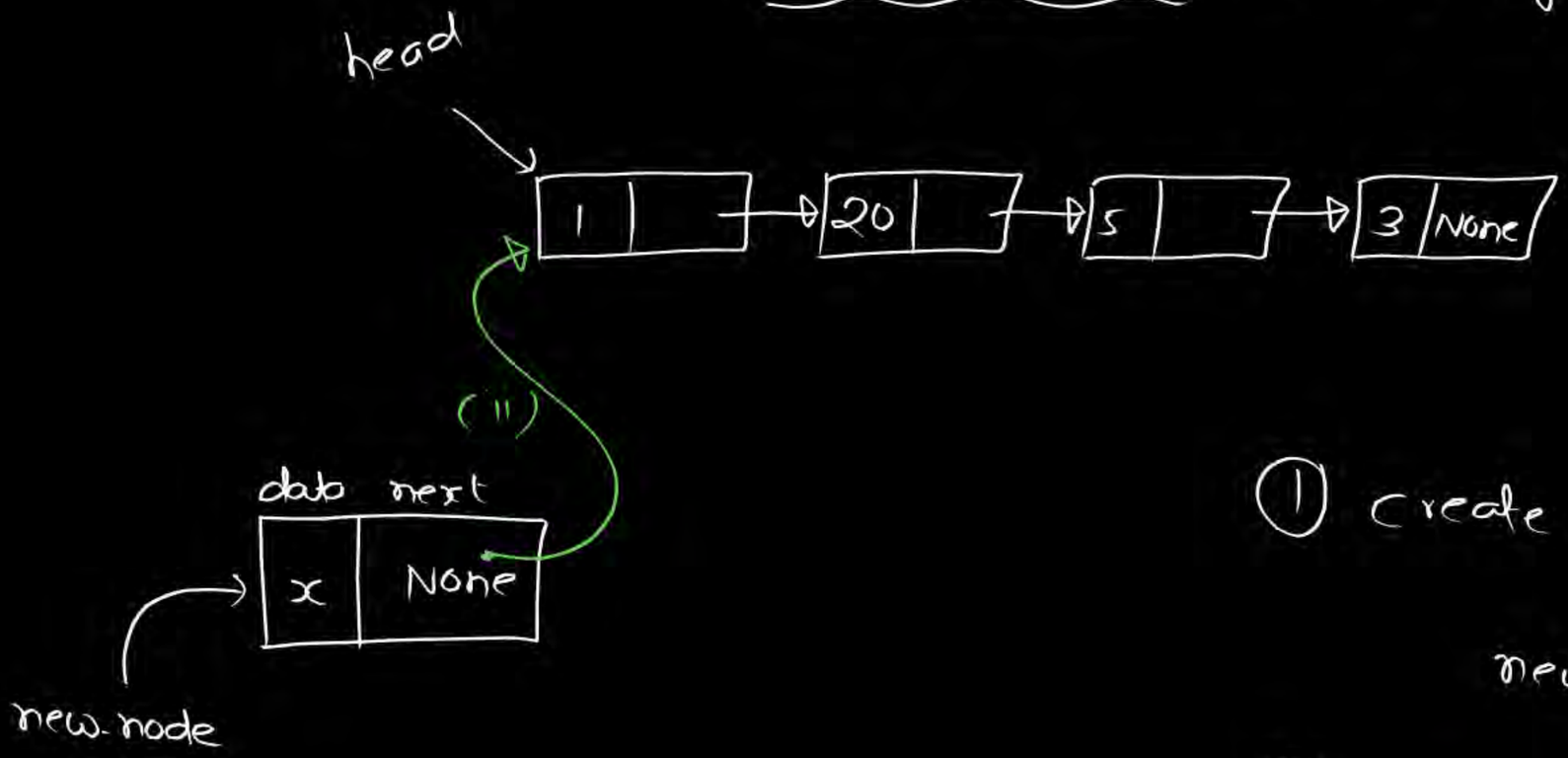
head = Prev

```
Prev = None
curr = head

while current :

    next_node = curr.next
    curr.next = Prev
    Prev = curr
    curr = next_node
return Prev
```

Insert an element ⇒ in beginning of LL

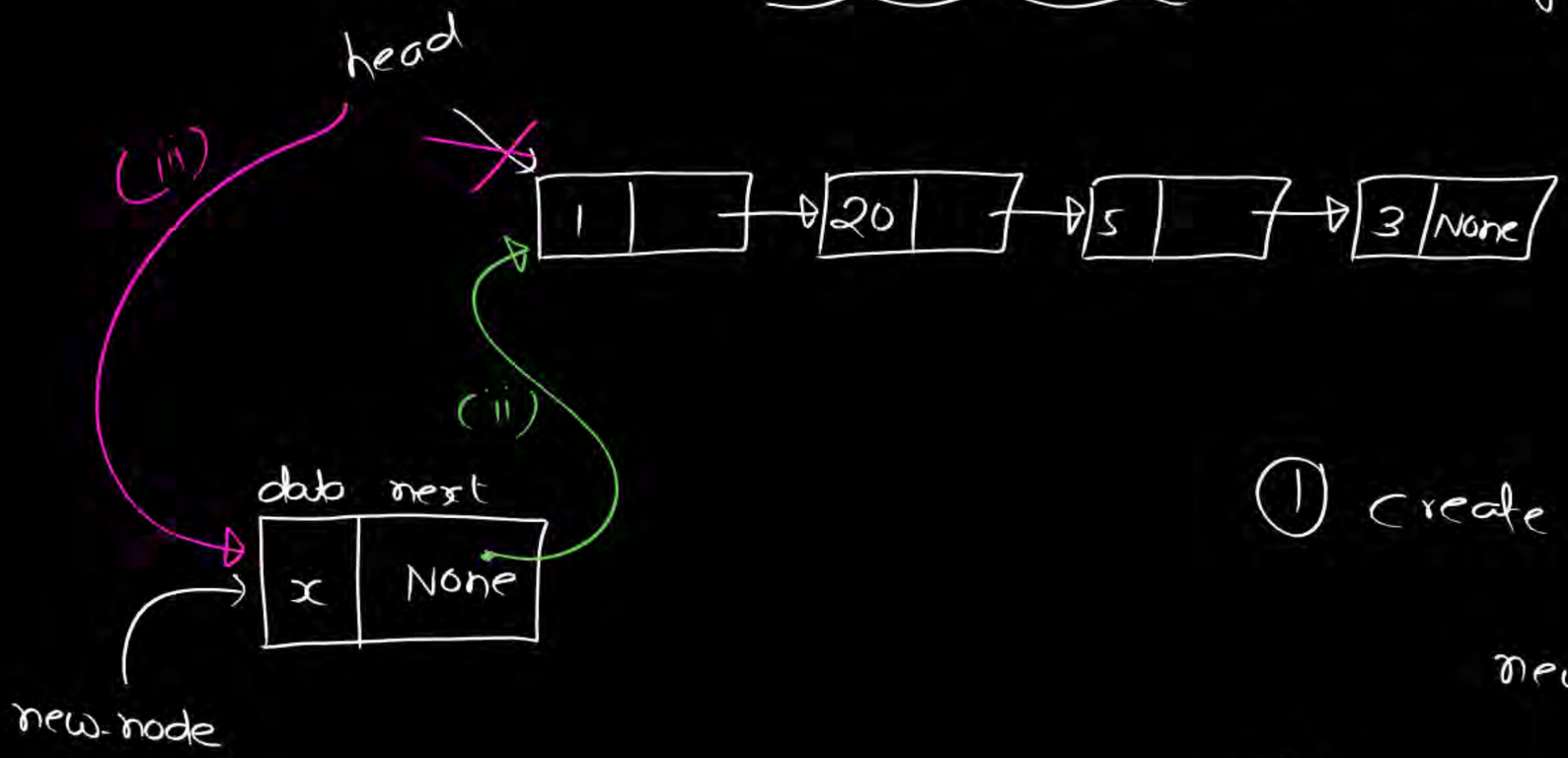


① Create a node with data x

```
new_node = Node(x)
```

② new_node.next = head

Insert an element ⇒ in beginning of LL



① Create a node with data x

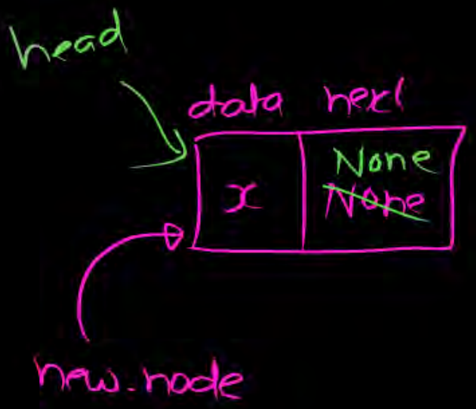
new_node = Node(x)

② new_node.next = head

③ head = new_node

Insert an element ⇒ in beginning of LL

LL is Empty head = None



① Create a node with data x

new_node = Node(x)

② new_node.next = head

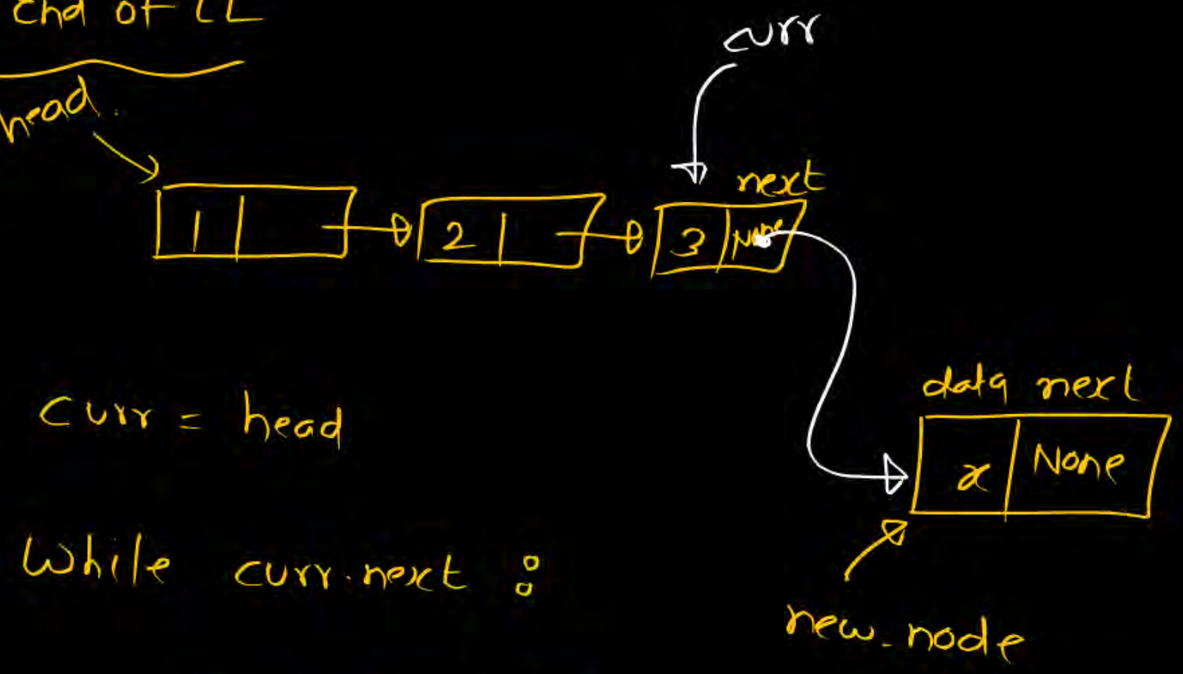
③ head = new_node

Insert at End of LL

Case1 LL is Empty

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

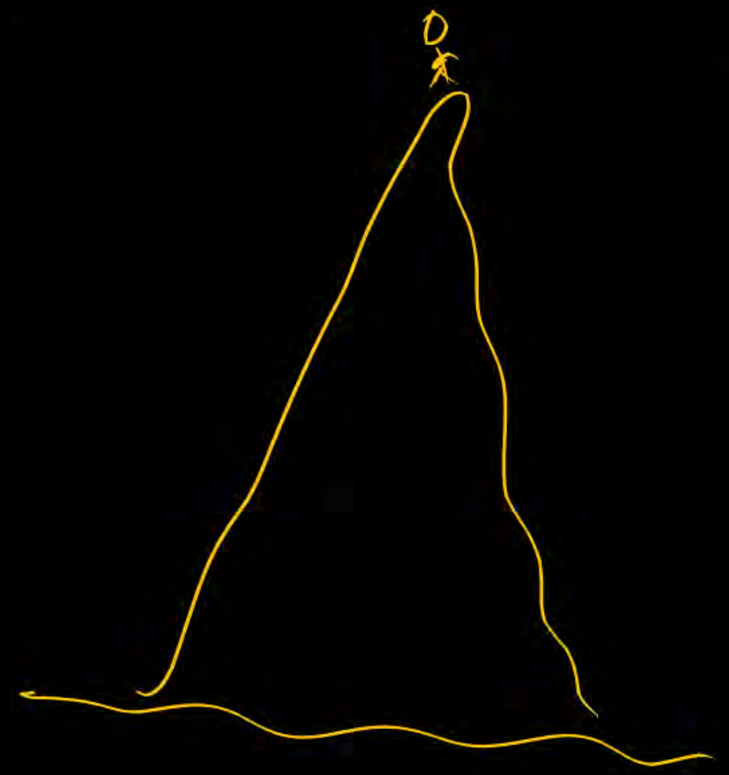
```
curr = head
While curr.next :
    curr = curr.next
curr.next = new_node
return head
```



head

None

Insert at End



def ____

if

X return ____

else

None

deletion

- Remove 1st node
- Remove last node

shayan

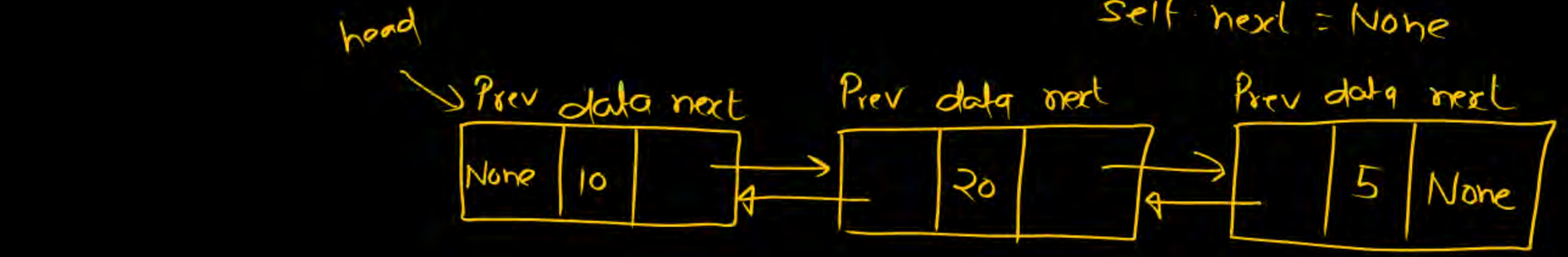
Statement :

Types of LL

① Singly LL

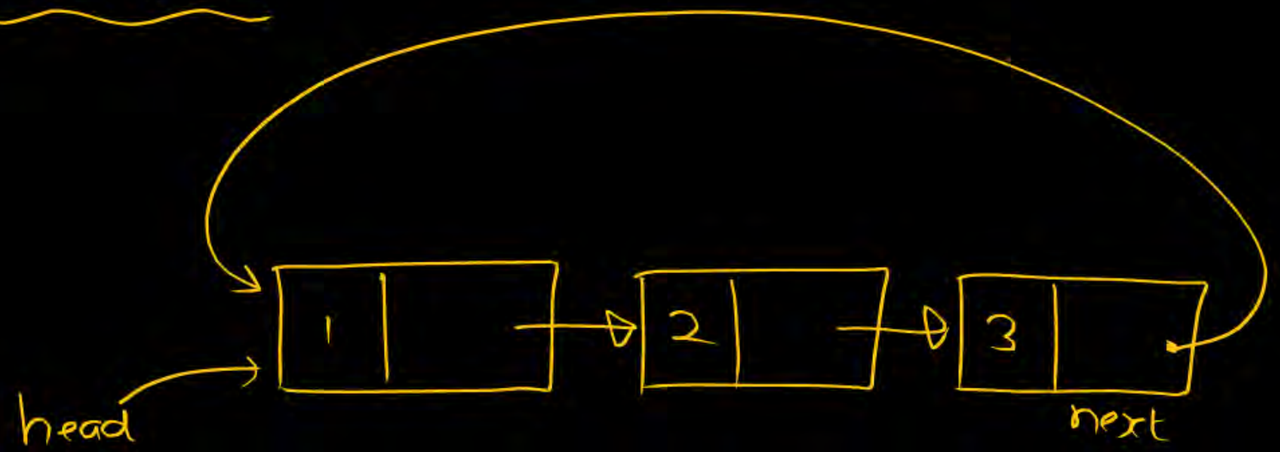
② Doubly LL

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

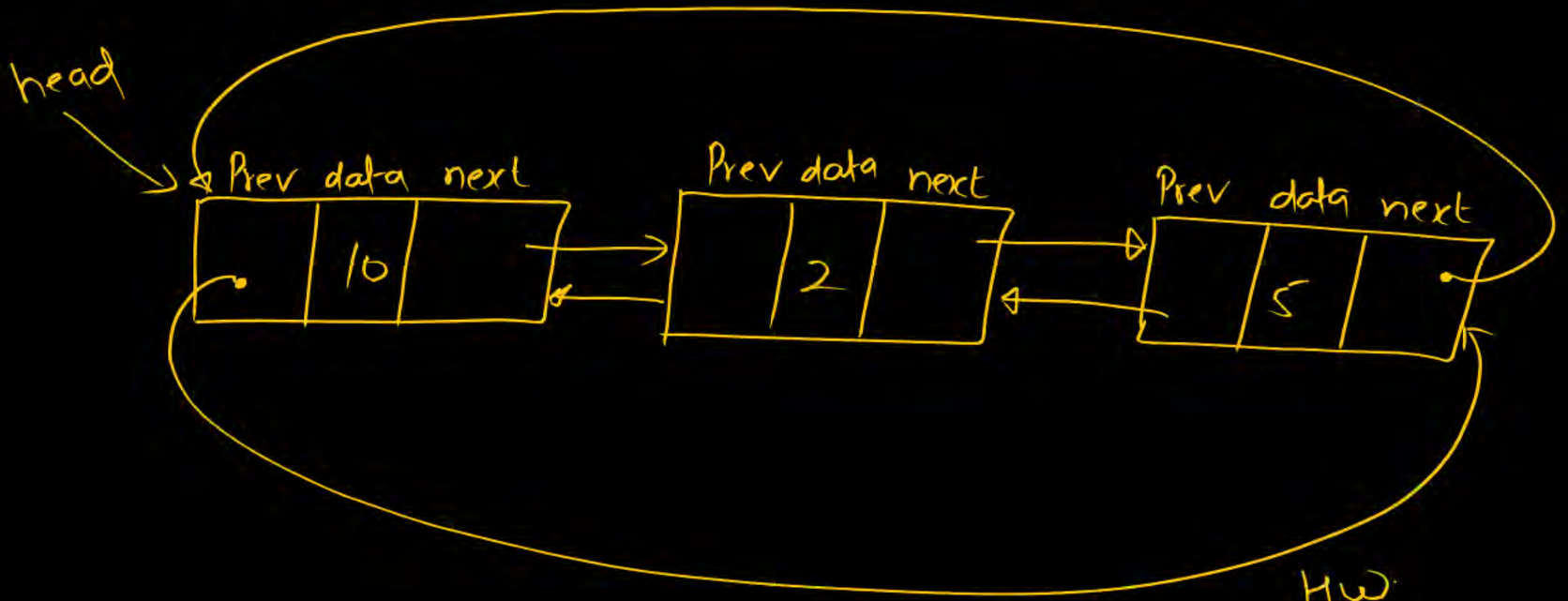


Adv

③ Circular LL



4) Doubly Circular LL



HW

[deletion first
deletion last]

linked list

Time complexity

H.W

TC

1) Search —

2) Print —

3) Insert at beg —

4) Insert at end —

THANK - YOU