Data Science & AI
&
NIC - Param

Python-For Data Science

Linked List

Lecture No.- 02

By- Pankaj Sharma Sir

Recap of Previous Lecture

Topic Linked List Part-01

Topics to be Covered

Topic Linked List Part-02

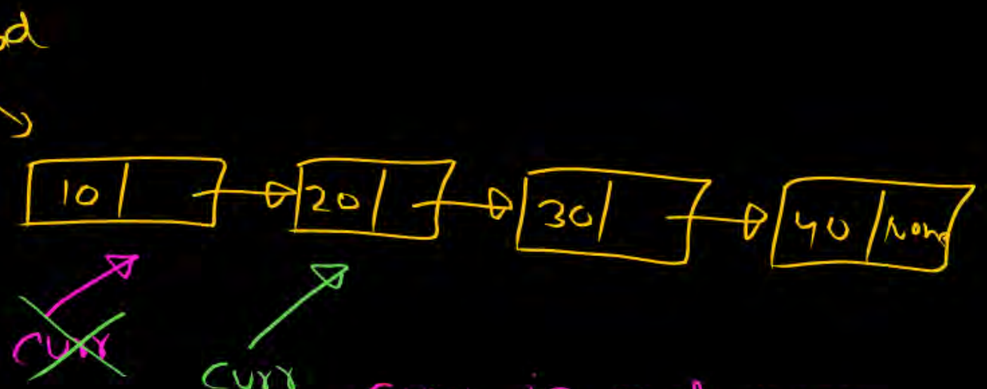
Topic : Linked List

```
def takinginput():
    head = None
    .
    .
    .
    .
    .
    .
    return head
```

```
r = takinginput()
```

Given a LL, print it

↳ head



if curr = head

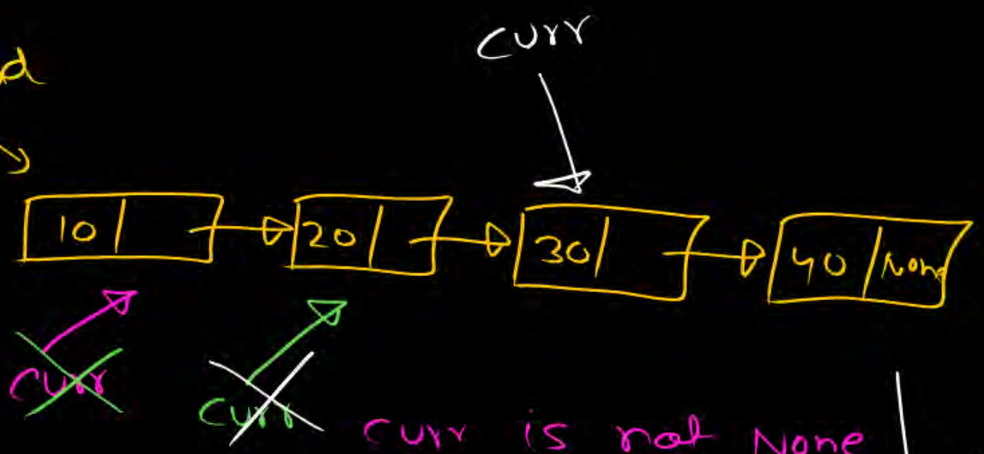
curr is not None

→ print(curr.data)

curr = curr.next

Given a LL, print it

→ head



if curr = head

curr is not None

→ print(curr.data)

curr = curr.next

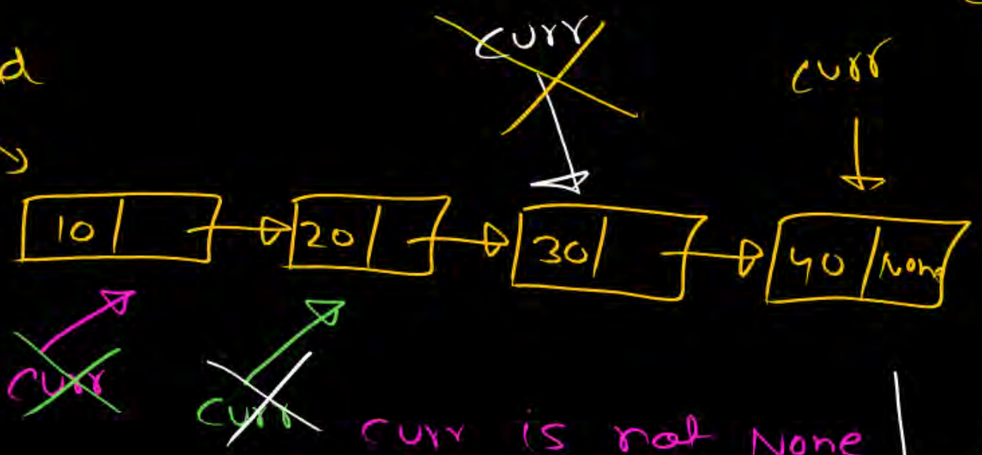
curr is not None

print(curr.data)

curr = curr.next

Given a LL, print it

→ head



if curr = head

curr is not None

→ print(curr.data)

curr = curr.next

curr is not None

print(curr.data)

curr = curr.next

curr is not None

print(curr.data)

curr = curr.next

Given a LL, print it.

→ head

curr is not None

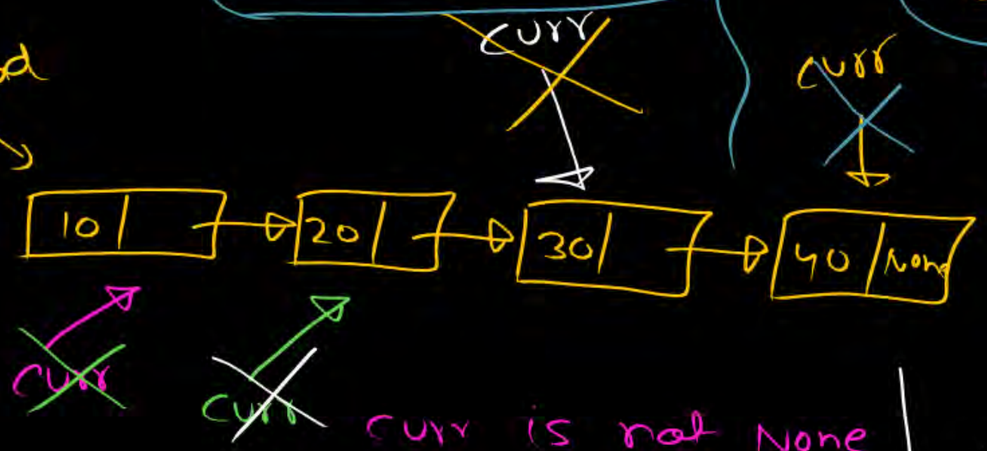
print(curr.data)

curr = curr.next

curr is not None

print(curr.data)

curr = curr.next



if curr = head

curr is not None

→ print(curr.data)

curr = curr.next

curr is not None

print(curr.data)

curr = curr.next

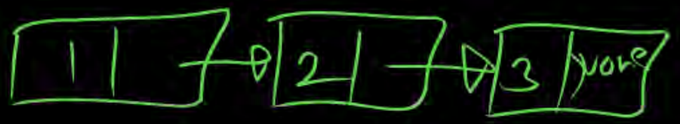
PrintLL का head

```
def PrintLL(head):
    curr = head
    while curr is not None:
        print(curr.data)
        curr = curr.next
```

PrintLL(head) main

PrintLL(head)

```
def PrintLL(head):
    curr = head
    while curr is not None:
        print(curr.data, "->", end="")
        curr = curr.next
    print(None)
```



1 → 2 → 3 → None

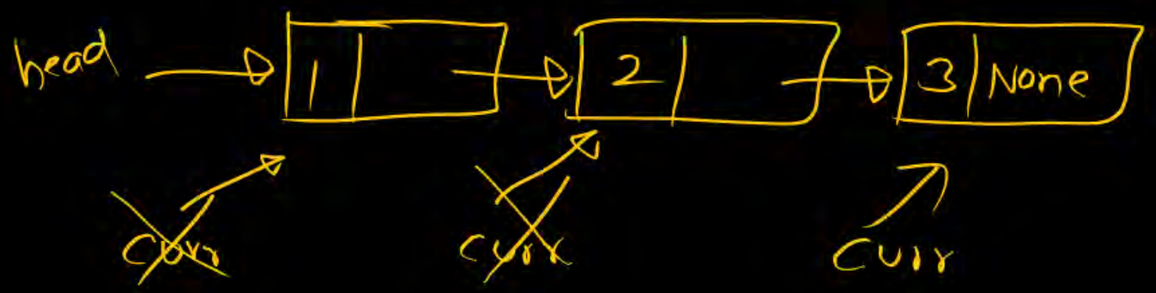
PrintLL(head) main

worst

None. ___

Given a LL, print last node data

Incomplete




```
While curr.next is not None :
    curr = curr.next
```

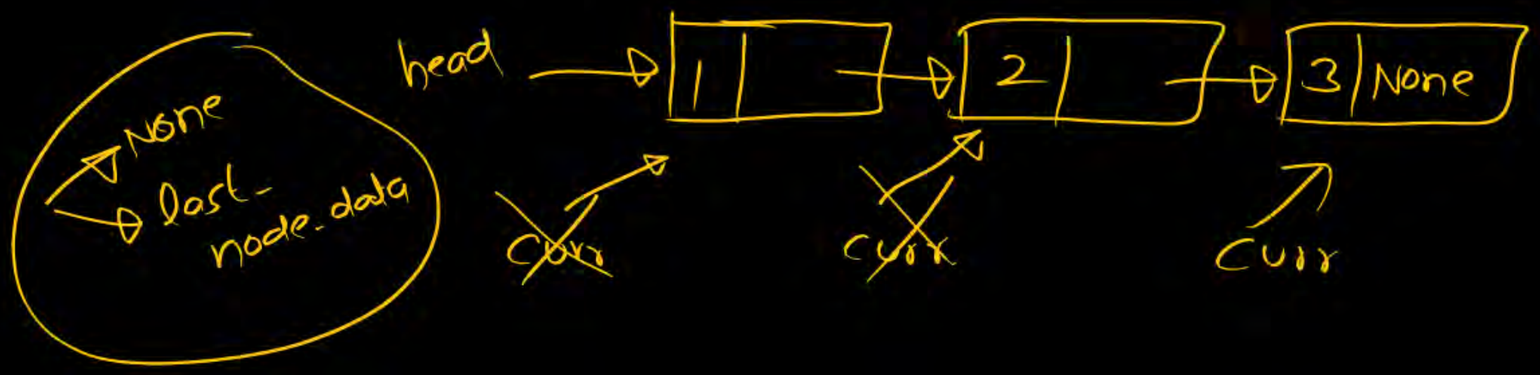
1) Is curr ⇒ last node

Move to next node

curr = curr.next

Given a LL, print last node data

Incomplete




```
while curr.next is not None:
    curr = curr.next
```

if curr is None:

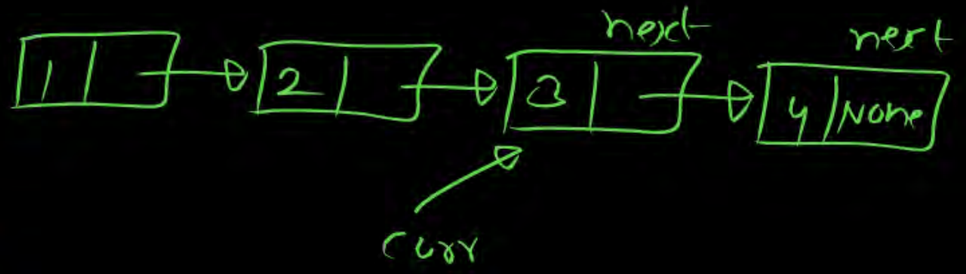
return None

r ←

```
x = fun(r)
print(x)
```

```
def fun(head):
    curr = head
    Special [ if curr is None:
                  return None
    while curr.next is not None:
        curr = curr.next
    return curr.data
```

Second last node

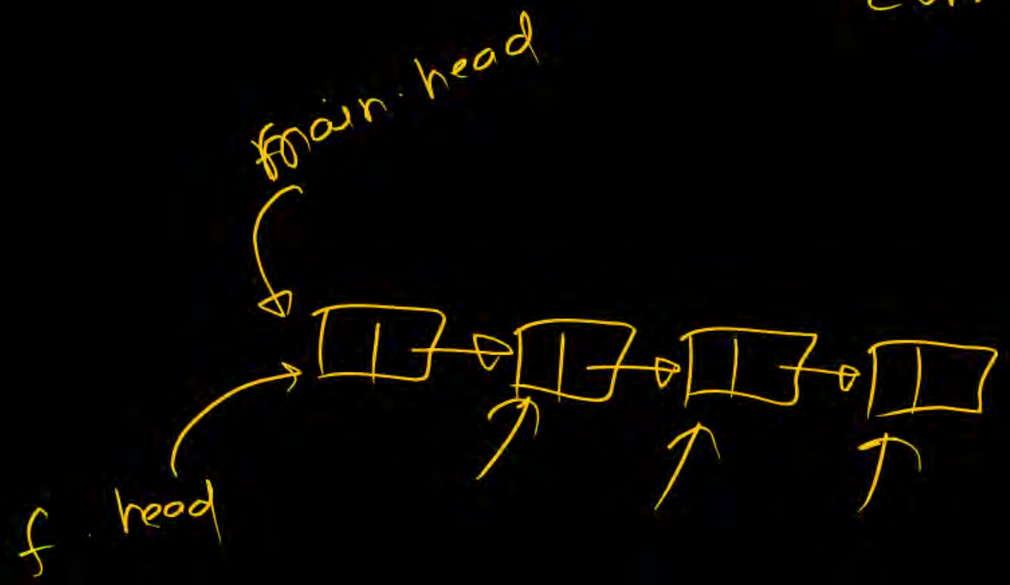


Ensure that atleast 2 node exist

curr.next.next ⇒ None

```
while curr.next.next is not None:
    curr = curr.next
```

curr = head



p = f(head)

if head is None or
head.next is None:
return None

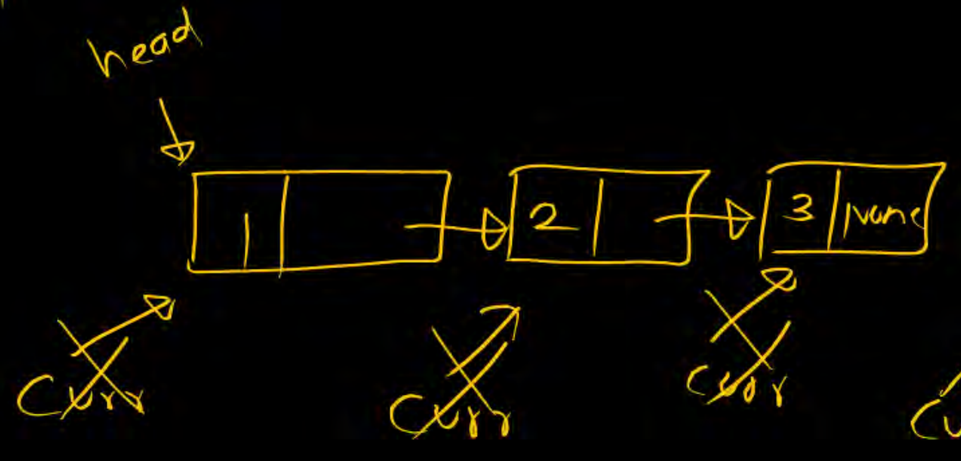
while head.next.next is not None:
head = head.next

return head.data

Given a LL, count the no of nodes

1+1+1

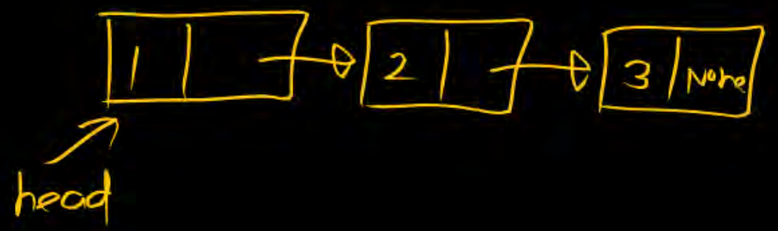
Curr = head




```
count = 0
curr = head
while curr is not None:
    count = count + 1
    curr = curr.next
```

LL create takinginput()

2nd elem ⇒ 1
3rd element ⇒ 2
4th element ⇒ 3
5th elem ⇒ 4
:
nth ele ⇒ (n-1)

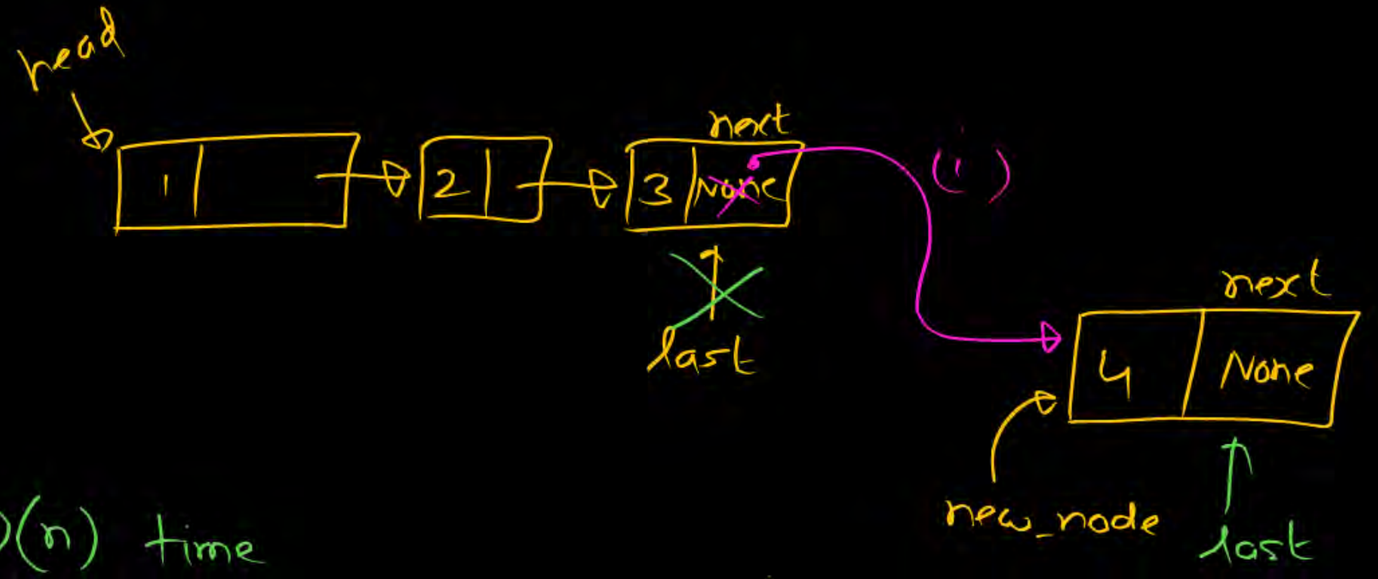


new_element : 4

T.C

⇒ 1+2+3+---+(n-1)

⇒ $\frac{(n-1)(n)}{2} = O(n^2)$



O(n) time

4th ele

(i) last.next = new_node

(ii) last = new_node

Singly LL

Searching an element in a given LL

ele

head

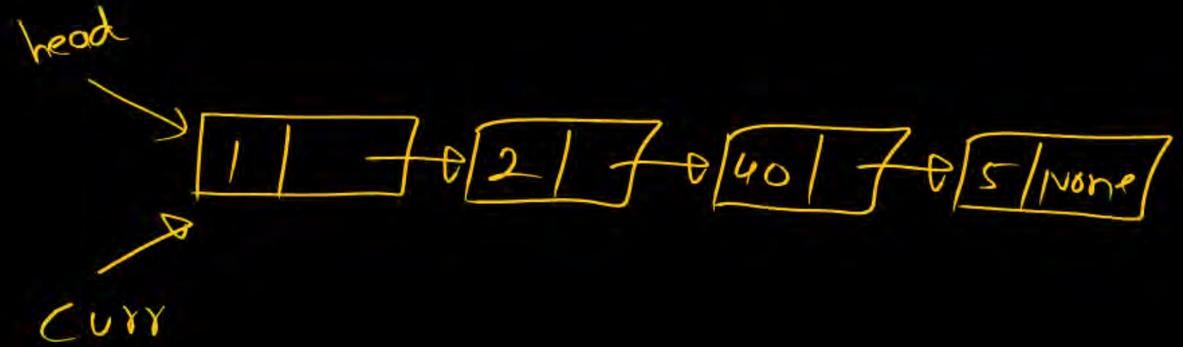
t.me/PWbankajsirP

Singly LL

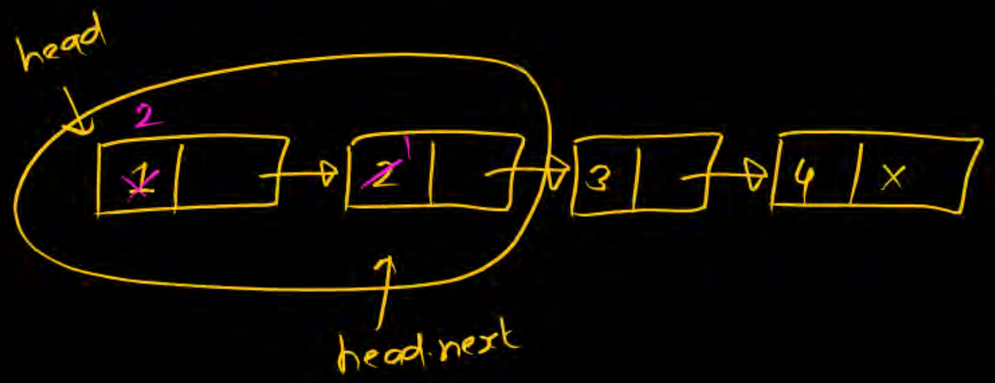
Searching an element in a given LL

ele

head

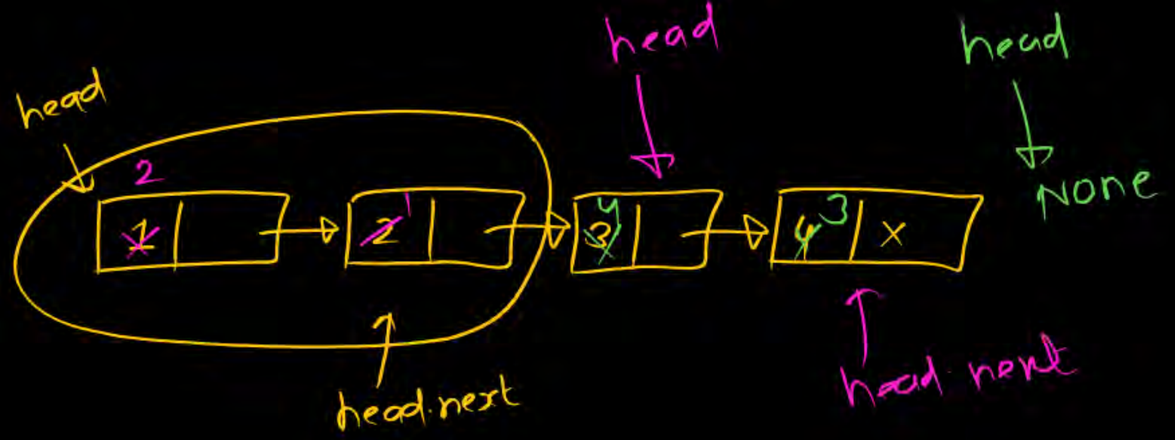



```
curr = head
while curr is not None :
    if curr.data == ele :
        return True
    curr = curr.next
return False
```



1.) swap

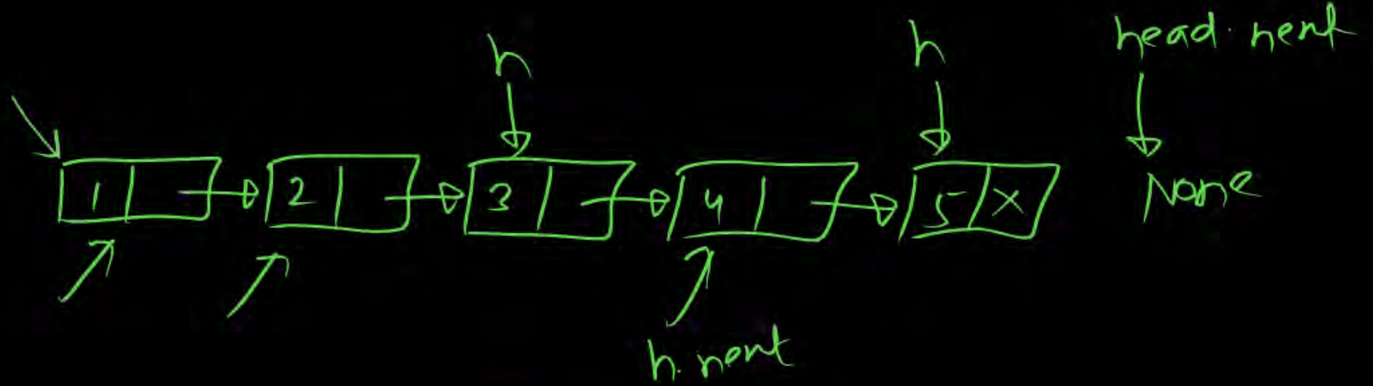
```
head.data, head.next.data = head.next.data, head.data
```



1) swap

```
head.data, head.next.data = head.next.data, head.data
```

2) `head = head.next.next`



h → None

while (h is not None) ~~and~~ or None.next h.next is not None :

None

False

```
While head is not None and head.next is not None:

While head and head.next:
    swap data
    head = head.next.next
```

16 videos

CC → recursion

1. Find loop
2. intersection point
3. reverse

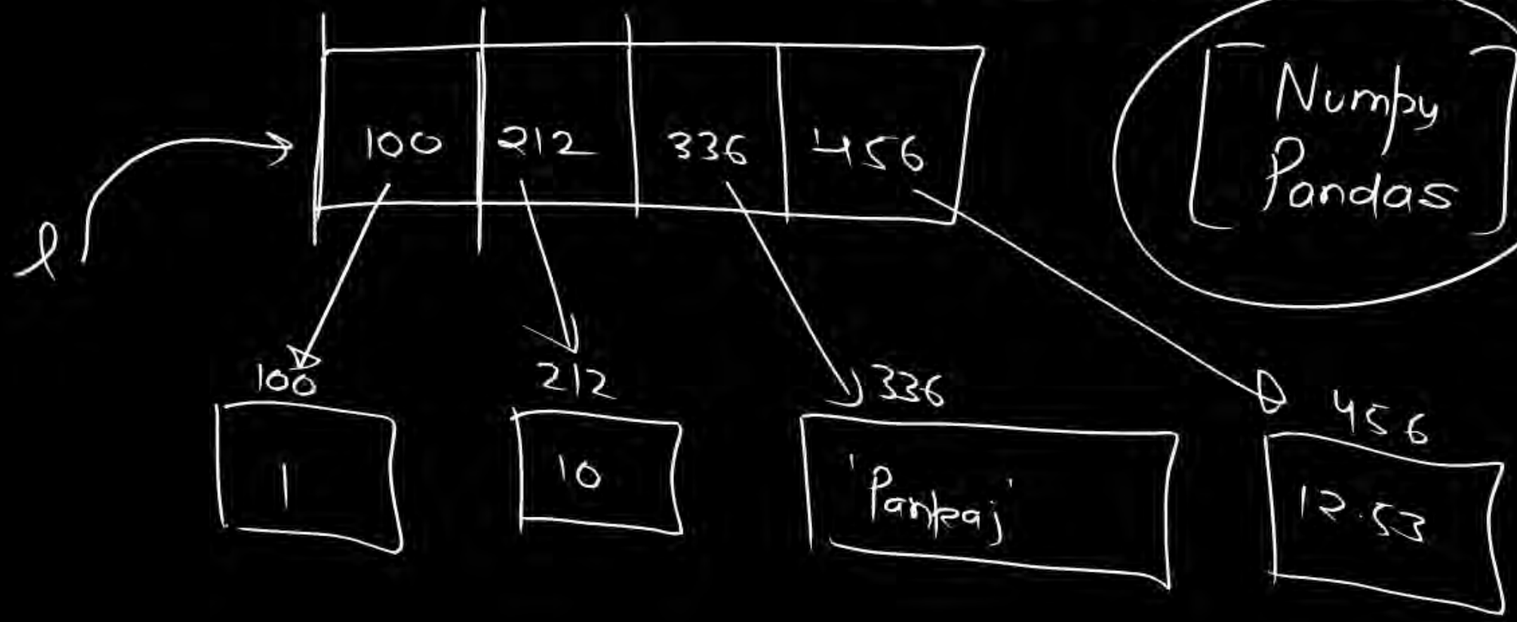
Types of LL

Stack and Queue

4 lecture ... to 5

4 lecture Trees. Tuesday

List



Numpy
Pandas

- No promise

4 Nov

In [8]:
```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
```

In [9]:
```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
def takinginput():
    l=[int(ele) for ele in input().split()]
    head=None
    for i in range(len(l)):
        new_node=Node(l[i])
        if head is None:
            head=new_node
        else:
            curr=head
            while curr.next is not None:
                curr=curr.next
            curr.next=new_node
    return head
```

In [ ]:
```
r=takinginput()
```

In [ ]:
```
print(head,r)
```

In [2]:
```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
def takinginput():
    l=[int(ele) for ele in input().split()]
    head=None
    for i in range(len(l)):
        new_node=Node(l[i])
        if head is None:
            head=new_node
        else:
            curr=head
            while curr.next is not None:
                curr=curr.next
            curr.next=new_node
    return head
def printingLL(head):
    while head is not None:
        print(head.data,"-->",end='')
        head=head.next
    print("None")
```

In [3]:
```
r=takinginput()
```

```
1 2 3 4 5
```

In [4]:
```
printingLL(r)
```

```
1 -->2 -->3 -->4 -->5 -->None
```

In [5]:
```
def last_node_data(head):
    if head is None:
        return head
    while head.next is not None :
        head=head.next
    return head.data
```

In [6]:
```
data=last_node_data(r)
```

In [7]:
```
print(data) #1-->2-->3-->4-->5-->None
```

```
5
```

In [8]:
```
def second_last_node_data(head):
    if head is None or head.next is None:
        return None
    while head.next.next is not None :
        head=head.next
    return head.data
```

In [9]:
```
d=second_last_node_data(r)#1-->2-->3-->4-->5
```

In [10]:
```
print(d)
```

```
4
```

In [11]:
```
def length(head):
    count=0
    while head is not None:
        count=count+1
        head=head.next
    return count
```

In [12]:
```
print(length(r))#1-->2-->3-->4-->5
```

```
5
```

In [13]:
```
def search(head,x):
    while head is not None:
        if x==head.data :
            return True
        head=head.next
    return False
x=search(r,10)
```

In [14]:
```
print(x)
```

```
False
```

In [15]:
```
y=search(r,4)#1-->2-->3-->4-->5-->None
```

In [16]:
```
print(y)
```

```
True
```

In [ ]:

THANK - YOU