Lecture 16, 9 October 2025

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming and Data Structures with Python

Linked Lists

Class List

append

if empty – set value to v

Iterate next till end of list

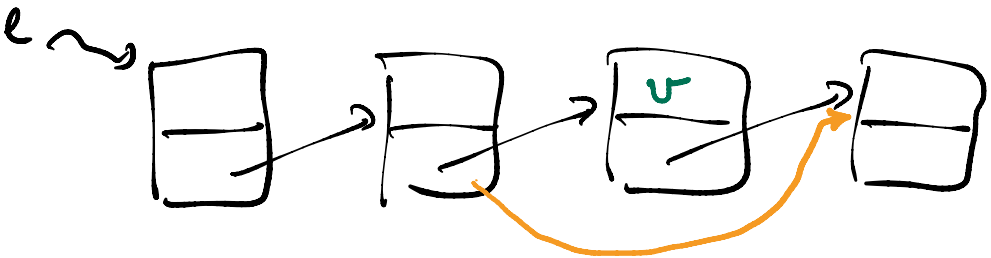
add new node at the end

Insert

delete(v)

No value v? — Do nothing

Multiple values v? - Remove first one



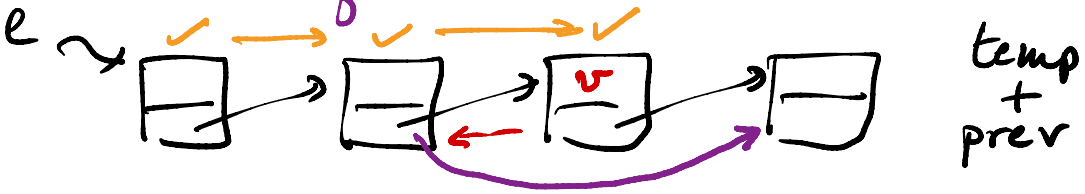
l.delete(v)

L.delete(v)

empty – return

not empty, first value is not v

iteratively follow next till we find v or reach end of list



Recursion & induction

fn defined inductively

f(n) depends on f(k), k<n

compute f(n) recursively

Base case

$\ell \to x$

$x \to \ell$

Inductive data structures

Lists

Base case: Empty list

Inductive stuff

build(v,ℓ)

v:ℓ

cons(v,ℓ)

List: (non empty)

First item - value

Rest - list (could be empty)

v_0 $[v_1, v_2]$

Recursive append(v)

$(x : l).append(v) = x : (l.append(v))$

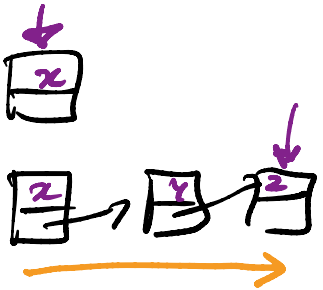
Base Case

value = None → value = v

Otherwise (value ≠ None)

next = None : create a new node with v, make next point to it

next ≠ None → recursively append v to Next



Recursive delete(v)

Base Case

Return

Otherwise

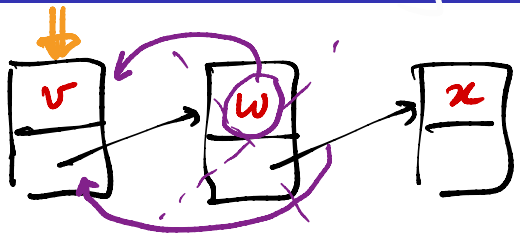
Easy case: current value is not v

Recursively next.delete(v)

(If next == None, return)

What if current value is v?

L.delete(v) — interesting case

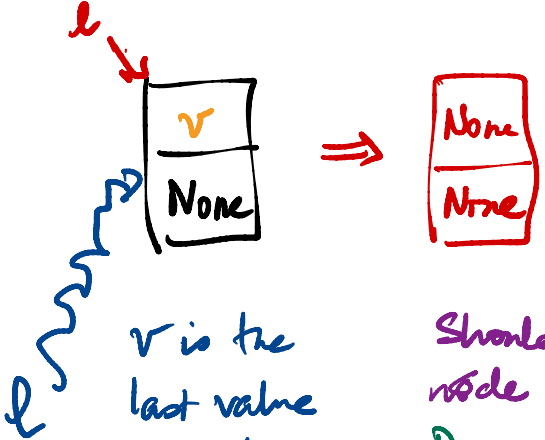


Cannot move
(like insert)

if next != None
copy next.value
next.next

if no x:
w nodes next = None
⇒ first node next = None

No next nodes ?



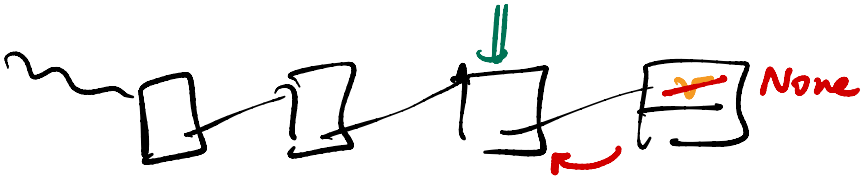
Singleton → Empty

v is the last value in a list

length ≥ 2

Should remove this node

Done at node that made recursive call



next.delete(v)

Check if next.value == None

If so, set next = None

Handling errors

Why should we deal with errors?

Interaction

User input → Run something


```
x = input()
```

```
x = input("Message: ")
```

Read an integer

```
x = input()
y = int(x)
```

Exception handling