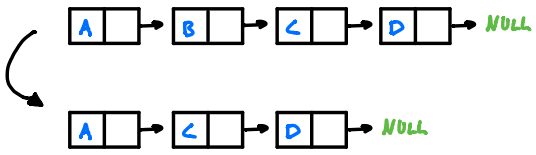
Singular Linked Lists Deletion

Given a key (data field) delete node with this field

Assume elements in the linked list are unique

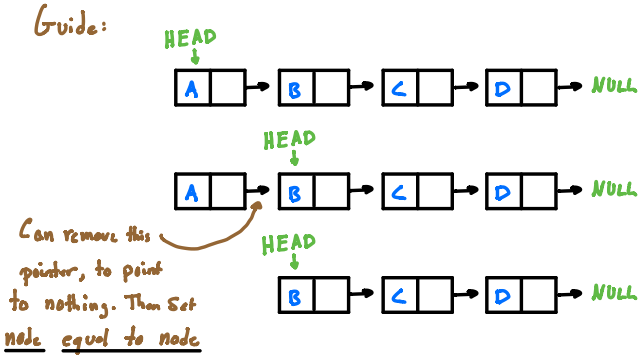
Example: Delete node with data field “B”



2 Cases:

- Node to be deleted is head
- Node to be deleted is not head

Node to be deleted is head




```
def delete_node(self, key):

    current_node = self.head

    if current_node and current_node == key:

        self.head = current_node.next

        current_node = None

        return

    prev = None

    while current_node and current_node.data != key:
        prev = current_node
        current_node = current_node.next

    if current_node is None:
        return

    prev.next = current_node.next

    current_node = None
```

Case 2

Check if the head node, is the node to be deleted. (which is set equal to the head of the list)

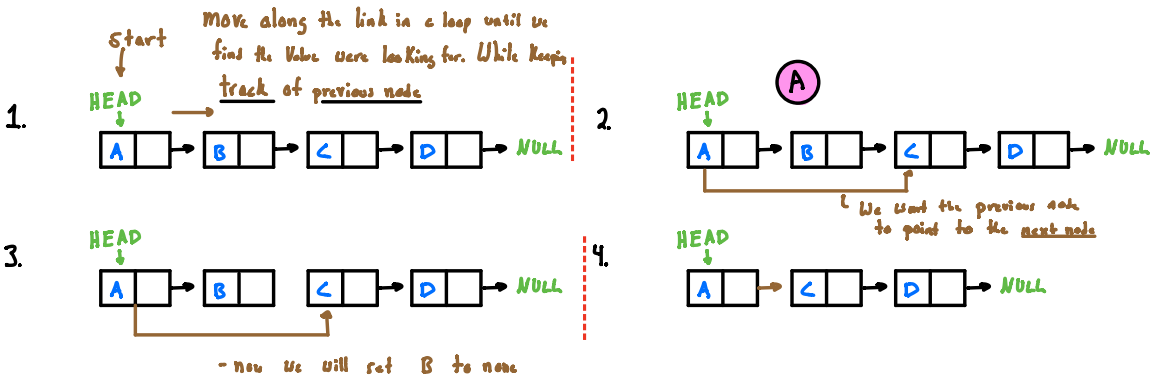
Check if list is not empty
Change head to point to node in the list

We're essentially moved Head to next element in the list.

Node to be deleted is not head

Delete node with data field "B"

Case 2:




```
def delete_node(self, key):
    current_node = self.head
    if current_node and current_node == key:
        self.head = current_node.next
        current_node = None
        return
    prev = None
    while current_node and current_node.data != key:
        prev = current_node
        current_node = current_node.next
    if current_node is None:
        return
    A prev.next = current_node.next
    current_node = None
```

Case 1

Case 2

iterate through list, while the head node is not none and while the data fields of the nodes we encounter are not equal to the key were looking for, keep moving along

Keeping track of previous node

move head pointer along

If we looped through the list and the element was not found, then we must return since it does not exist.

Set previous node to point to the next node (node after the current node, which will be deleted).

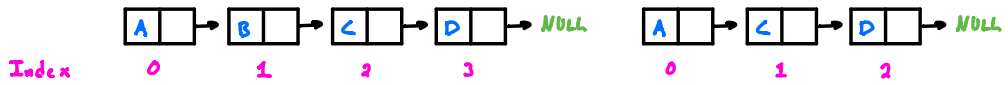
- now we will set B to none, which will remove the element from the list

Delete Node at Position

Given a position, delete node with this position

Assume elements in linked list are unique.

Example: Delete node with position 1




```
def delete_node_at_pos(self, pos):

    cur_node = self.head

    if pos == 0:
        self.head = cur_node.next
        cur_node = None
        return

    prev = None
    count = 0
    while cur_node and count != pos:
        prev = cur_node
        cur_node = cur_node.next
        count += 1

    if cur_node is None:
        return

    prev.next = cur_node.next
    cur_node = None
```

Set current node to the head

if position to be removed is the head node

We need to go thru list and see if its equal to the position

Keep track

Moving us along in list

Position Given does not exist. Check

Set previous node to point to the next

Set element to be removed to none

Full Code:

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


class LinkedList:
    def __init__(self):
        self.head = None

    def print_list(self):
        cur_node = self.head
        while cur_node:
            print(cur_node.data)
            cur_node = cur_node.next

    def append(self, data):
        new_node = Node(data)

        if self.head is None:
            self.head = new_node
            return

        last_node = self.head
        while last_node.next:
            last_node = last_node.next
        last_node.next = new_node

    def prepend(self, data):
        new_node = Node(data)

        new_node.next = self.head
        self.head = new_node

    def insert_after_node(self, prev_node, data):

        if not prev_node:
            print("Previous node is not in the list")
            return

        new_node = Node(data)

        new_node.next = prev_node.next
        prev_node.next = new_node

    def delete_node(self, key):

        cur_node = self.head

        if cur_node and cur_node.data == key:
            self.head = cur_node.next
            cur_node = None
            return

        prev = None
        while cur_node and cur_node.data != key:
            prev = cur_node
            cur_node = cur_node.next

        if cur_node is None:
            return

        prev.next = cur_node.next
        cur_node = None

    def delete_node_at_pos(self, pos):

        cur_node = self.head

        if pos == 0:
            self.head = cur_node.next
            cur_node = None
            return

        prev = None
        count = 0
        while cur_node and count != pos:
            prev = cur_node
            cur_node = cur_node.next
            count += 1

        if cur_node is None:
            return

        prev.next = cur_node.next
        cur_node = None

llist = LinkedList()
llist.append("A")
llist.append("B")
llist.append("C")
llist.append("D")

llist.delete_node("B")
llist.delete_node_at_pos(2)

llist.print_list()
```

→ A C