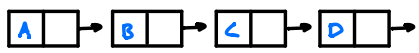
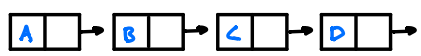
Singly Linked Lists- Node Swap

Node Swap. Two Cases: (Assume data entries are unique)

1. Node 1 and Node 2 are not head Nodes
2. Either Node 1 or Node 2 are head Nodes



Swap Nodes Function




```
def swap_nodes(self, key_1, key_2):

    if key_1 == key_2:
        return

    prev_1 = None
    curr_1 = self.head
    while curr_1 and curr_1.data != key_1:
        prev_1 = curr_1
        curr_1 = curr_1.next

    prev_2 = None
    curr_2 = self.head
    while curr_2 and curr_2.data != key_2:
        prev_2 = curr_2
        curr_2 = curr_2.next

    if not curr_1 or not curr_2:
        return
```

Check if we were given the same nodes

Start at head node

loop until we get to node we want to swap

- While the head pointer is not at the end of the list (not none) and while the data element is not equal to the data member we seek

Keep updating nodes while we loop until we find the values we wish to swap.

Checking to see if our keys exists

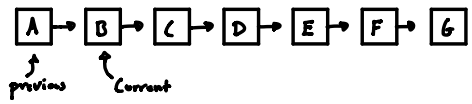
Do exact same thing for the second key

Quick test of the Node Swap

★ Case 1 Example ★

```
def swap_nodes(self, key_1, key_2):

    if key_1 == key_2:
        return

    prev_1 = None
    curr_1 = self.head
    while curr_1 and curr_1.data != key_1:
        prev_1 = curr_1
        curr_1 = curr_1.next

    print(prev_1.data) → A
    print(curr_1.data) → B
```

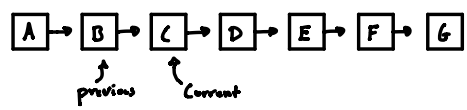
For key 1: We have it as our current and its previous node




```
llist.swap_nodes("B","C")
```

```
def swap_nodes(self, key_1, key_2):

    if key_1 == key_2:
        return

    prev_2 = None
    curr_2 = self.head
    while curr_2 and curr_2.data != key_2:
        prev_2 = curr_2
        curr_2 = curr_2.next

    print(prev_2.data) → B
    print(curr_2.data) → C
```

For key 2: We have it as our current and its previous node



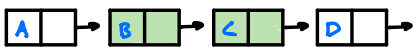

```
llist.swap_nodes("B","C")
```

Case 1: Node 1 and Node 2 are not head Nodes

```
① if prev_1:
      prev_1.next = curr_2
  else:
      self.head = curr_2

② if prev_2:
      prev_2.next = curr_1
  else:
      self.head = curr_1

③ curr_1.next, curr_2.next = curr_2.next, curr_1.next
```

We check if the previous node has a node

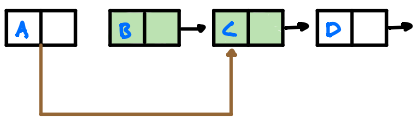
If it has a node it essentially telling us it's <u>not</u> a head node.



Were trying to swap node B and C

Variable Explorer: prev_1.data = A , <u>curr_1.data = B</u> , prev_2.data = B , <u>curr_2.data = C</u>

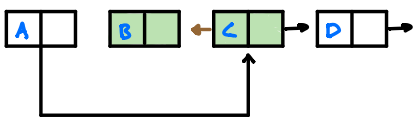
①



prev_1.next = Curr_2

prev_1.next = C

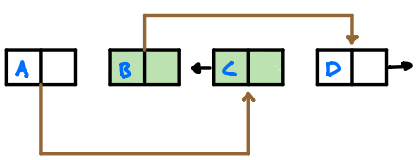
②



prev_2.next = Curr_1

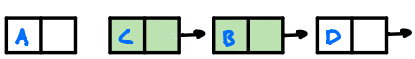
prev_2.next = B

③



Curr_1.next, curr_2.next = Curr_2.next, curr_1.next

Swapped :

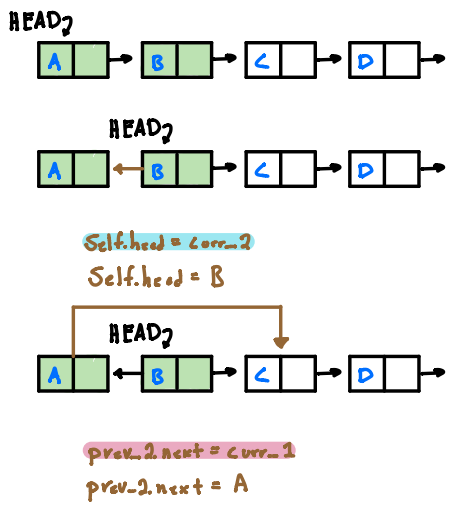


Case 2: Either Node 1 or Node 2 are head Nodes

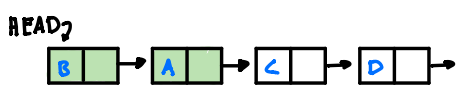
```
if prev_1:
    prev_1.next = curr_2
else:
    self.head = curr_2

if prev_2:
    prev_2.next = curr_1
else:
    self.head = curr_1

curr_1.next, curr_2.next = curr_2.next, curr_1.next
```

```
llist.swap_nodes("A", "B")
```

Variable Explorer: prev_1.data = none, curr_1.data = A , prev_2.data = A , curr_2.data = B



curr_1.next, curr_2.next = curr_2.next, curr_1.next



swapped

Full Code:

```python
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 = 1
        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

    def len_iterative(self):

        count = 0
        cur_node = self.head

        while cur_node:
            count += 1
            cur_node = cur_node.next
        return count

    def len_recursive(self, node):
        if node is None:
            return 0
        return 1 + self.len_recursive(node.next)
```

```python
    '''swap by changing the next attribute of node'''
    def swap_nodes(self, key_1, key_2):

        if key_1 == key_2:
            return

        prev_1 = None
        curr_1 = self.head
        while curr_1 and curr_1.data != key_1:
            prev_1 = curr_1
            curr_1 = curr_1.next

        prev_2 = None
        curr_2 = self.head
        while curr_2 and curr_2.data != key_2:
            prev_2 = curr_2
            curr_2 = curr_2.next

        if not curr_1 or not curr_2:
            return


        if prev_1:
            prev_1.next = curr_2
        else:
            self.head = curr_2

        if prev_2:
            prev_2.next = curr_1
        else:
            self.head = curr_1

        curr_1.next, curr_2.next = curr_2.next, curr_1.next

    ''' Alternate swap node function , swap by changing the data attribute of node '''
    def swap_nodes_alt(self, key_1, key_2):
        if key_1 == key_2:
            return
        curr  = self.head
        x , y = None , None # Assign None to avoid reference error
        while curr :
            if curr.data == key_1:
                x = curr # key_1 found
            if curr.data == key_2:
                y =curr # key_2 found
            curr = curr.next

        if x and y: # Check if both key's exist
            x.data , y.data = y.data , x.data
        else :
            return
```

```python
llist = LinkedList()
llist.append("A")
llist.append("B")
llist.append("C")
llist.append("D")
llist.append("E")
llist.append("F")
llist.append("G")

print("Initial list")
llist.print_list()

print(" swap by changing next attribute \n")
llist.swap_nodes("A", "B")

llist.print_list()

print(" swap by changing data attribute \n ")
llist.swap_nodes_alt("B", "A")

llist.print_list()
```

Results:

```
Initial list
A
B
C
D
E
F
G
 swap by changing next attribute

B
A
C
D
E
F
G
 swap by changing data attribute

A
B
C
D
E
F
G
```