Reverse

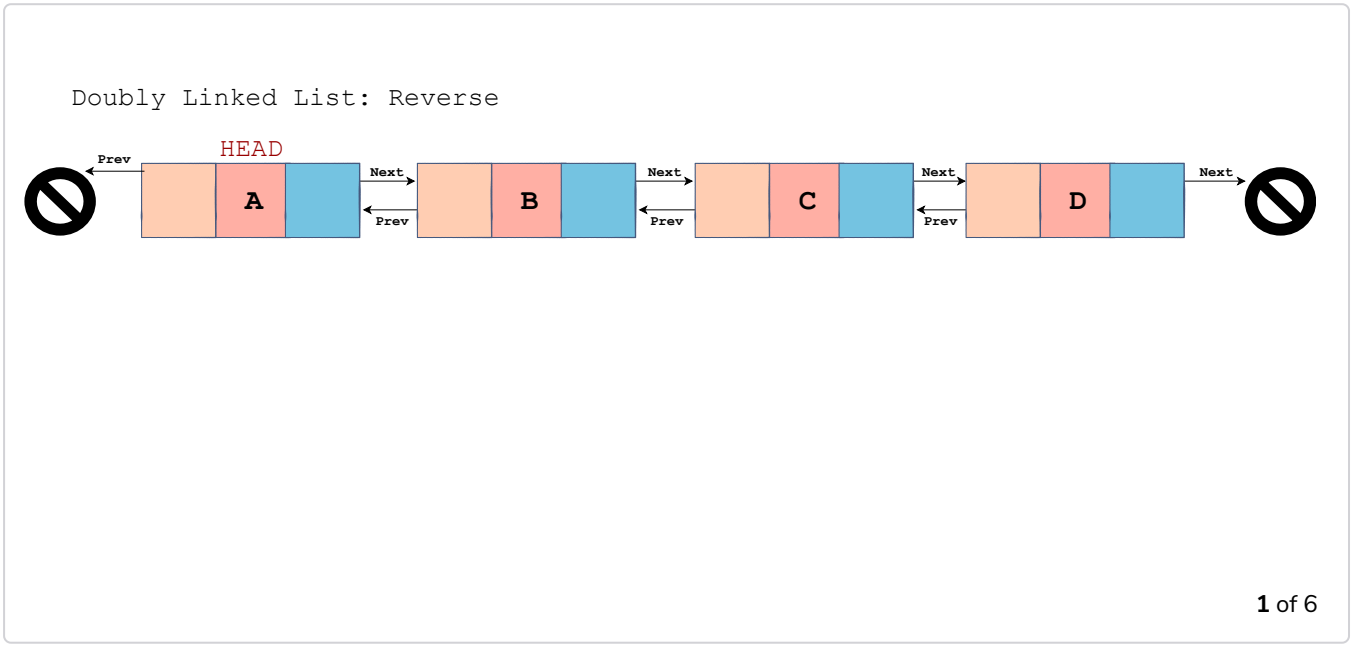
In this lesson, you will learn how to reverse a doubly linked list.

We'll cover the following

In this lesson, we consider how to reverse the nodes in a doubly linked list. Once we cover the concept of how to perform this action, we follow through with a Python implementation.

Algorithm

To reverse a doubly linked list, we need to switch the next and the previous pointers of every node. Also, we need to switch the last node with the head node of the linked list. Check out the illustration below for more clarity:



As we traverse the linked list, we swap the previous pointer with the next pointer and eventually, we make the last node of the original linked list the head node of the reversed linked list.

Implementation

Now let's go ahead and see how this algorithm is implemented in Python:

```
1  def reverse(self):
2    tmp = None
3    cur = self.head
4    while cur:
5      tmp = cur.prev
6      cur.prev = cur.next
7      cur.next = tmp
8      cur = cur.prev
9    if tmp:
10     self.head = tmp.prev
```

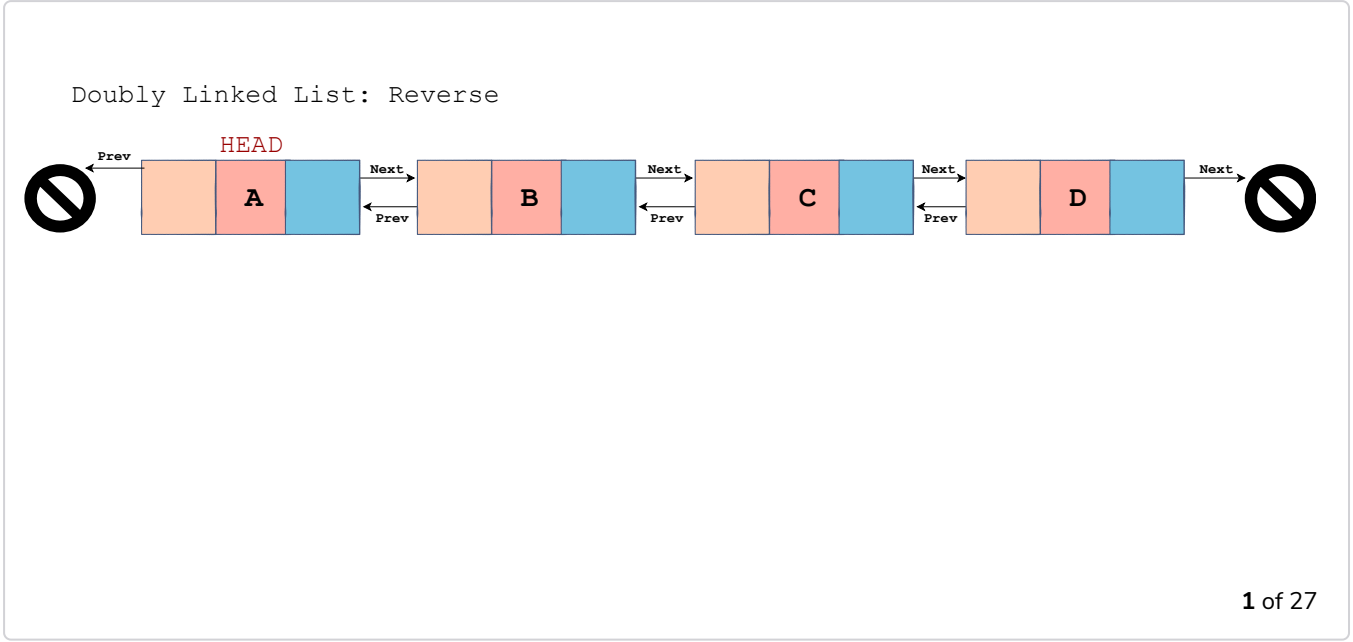
reverse(self)

Explanation

On **lines 2-3**, `tmp` and `cur` is set to `None` and `self.head`, respectively. In the `while` loop on **line 4**, which terminates when `cur` equals `None`, we swap the next and the previous pointers of `cur`. To swap, we first save the value of `cur.prev` in a temporary variable, `tmp`, on **line 5**. Next, we update `cur.prev` to `cur.next` on **line 6** while `cur.next` is set to `tmp` on **line 7**. `tmp` has the value of `cur.prev` stored before the update on **line 6**. This is pretty much the standard way of swapping the values of two variables in programming.

Now as the swap has taken place, instead of updating `cur` to `cur.next`, we update `cur` to `cur.prev` to iterate to the next node in the original linked list (**line 8**). When the `while` loop terminates, we are almost done with the reversal except for setting the new head of the reverse linked list. Hence, on **lines 9-10**, we check if `tmp` is not `None` and if it is not, we set `self.head` to `tmp.prev` where `tmp.prev` is the last node in the linked list.

Now the above code is tough to visualize, so we have some illustrations for you to go through and visualize what's happening.



In the code widget below, we reverse a doubly-linked list. Go ahead and modify the test cases with your own, so you get hands-on practice with our implementation.

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


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

  def append(self, data):
    if self.head is None:
      new_node = Node(data)
      new_node.prev = None
      self.head = new_node
    else:
      new_node = Node(data)
      cur = self.head
      while cur.next:
        cur = cur.next
      cur.next = new_node
      new_node.prev = cur
      new_node.next = None

  def prepend(self, data):
    if self.head is None:
      new_node = Node(data)
      new_node.prev = None
```

```
    self.head = new_node
  else:
    new_node = Node(data)

    self.head.prev = new_node
    new_node.next = self.head
    self.head = new_node
    new_node.prev = None

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

def add_after_node(self, key, data):
  cur = self.head
  while cur:
    if cur.next is None and cur.data == key:
        self.append(data)
        return
    elif cur.data == key:
        new_node = Node(data)
        nxt = cur.next
        cur.next = new_node
        new_node.next = nxt
        new_node.prev = cur
        nxt.prev = new_node
        return
    cur = cur.next

def add_before_node(self, key, data):
  cur = self.head
  while cur:
    if cur.prev is None and cur.data == key:
      self.prepend(data)
      return
    elif cur.data == key:
      new_node = Node(data)
      prev = cur.prev
      prev.next = new_node
      cur.prev = new_node
      new_node.next = cur
      new_node.prev = prev
      return
    cur = cur.next

def delete(self, key):
  cur = self.head
  while cur:
    if cur.data == key and cur == self.head:
      # Case 1:
      if not cur.next:
        cur = None
        self.head = None
        return

      # Case 2:
      else:
        nxt = cur.next
        cur.next = None
        nxt.prev = None
        cur = None
```

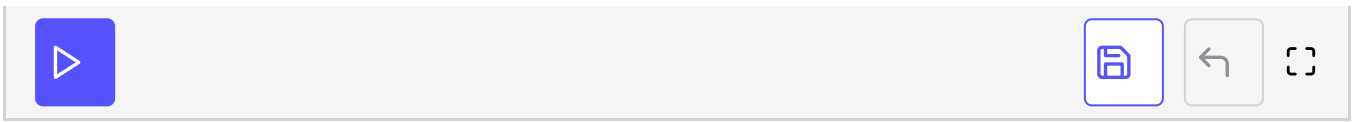
```
            self.head = nxt
            return

        elif cur.data == key:
            # Case 3:
          if cur.next:
              nxt = cur.next
              prev = cur.prev
              prev.next = nxt
              nxt.prev = prev
              cur.next = None
              cur.prev = None
              cur = None
              return

            # Case 4:
          else:
              prev = cur.prev
              prev.next = None
              cur.prev = None
              cur = None
              return
        cur = cur.next

    def reverse(self):
      tmp = None
      cur = self.head
      while cur:
        tmp = cur.prev
        cur.prev = cur.next
        cur.next = tmp
        cur = cur.prev
      if tmp:
        self.head = tmp.prev

dllist = DoublyLinkedList()
dllist.append(1)
dllist.append(2)
dllist.append(3)
dllist.append(4)
dllist.reverse()
dllist.print_list()
```



I hope everything has been clear up until now. Now it’s time for some challenges! See you in the next lesson.