|  |
| --- |
| class ListNode: def \_\_init\_\_(self, val=0, next=None): self.val = val self.next = next  def rotate\_right(head, k): if not head or not head.next or k == 0: return head    *# Count the nodes*  count = 1 curr = head while curr.next: curr = curr.next count += 1    *# Adjust k if it's too big*  k = k % count if k == 0:  return head    *# Find the new tail*  new\_tail\_pos = count - k - 1 curr = head for \_ in range(new\_tail\_pos): curr = curr.next    *# Reconnect the list*  new\_head = curr.next  curr.next = None *# Break the list* |

# Balram Mandal (30)

# Assignment No-03

Implement linked list, stacks, queues and dequeues data structures

1. Given the head of a linked list, rotate the list to the right by k places.

Example 1

Input: head = [1,2,3,4,5], k = 2 Output: [4,5,1,2,3]

Example 2

Input: head = [0,1,2], k = 4 Output: [2,0,1]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| last = new\_head while last.next:  last = last.next  last.next = head *# Connect the tail to the head*    return new\_head  *# Example usage (Example 1)* head1 = ListNode(1) head1.next = ListNode(2) head1.next.next = ListNode(3) head1.next.next.next = ListNode(4) head1.next.next.next.next = ListNode(5) rotated\_head1 = rotate\_right(head1, 2)  *# Print the rotated list* while rotated\_head1: print(rotated\_head1.val, end=" -> ") rotated\_head1 = rotated\_head1.next print("None")  *# Example usage (Example 2)* head2 = ListNode(0) head2.next = ListNode(1) head2.next.next = ListNode(2) rotated\_head2 = rotate\_right(head2, 4)  *# Print the rotated list* while rotated\_head2: print(rotated\_head2.val, end=" -> ") rotated\_head2 = rotated\_head2.next print("None")   |  |  | | --- | --- | | 4 -> 5 -> 1 -> 2 -> 3 -> None | | | 2 -> 0 -> 1 -> None |  | |