How to reverse the nodes and print them:

Methods added to the LinkedList class:

#reversed  
def print\_reversed(self):  
 self.print\_reversed\_recursive(self.head)  
  
# Recursive Method  
def print\_reversed\_recursive(self, node):  
 if node is not None:  
 self.print\_reversed\_recursive(node.next)  
 print(node.value, end=" ")

In the recursive method, we reach the end of the linked list before we stop the recursive process. Then, we start printing the values in reverse order.

After we write these methods, we insert the nodes:

1. my\_linked\_list.insert\_node(3)
2. my\_linked\_list.insert\_node(5)
3. my\_linked\_list.insert\_node(8)
4. my\_linked\_list.insert\_node(15)
5. my\_linked\_list.insert\_node(26)
6. my\_linked\_list.insert\_node(35)

Finally, we call the method to see the output:

1. my\_linked\_list.print\_reversed()

explanation:

**Printing Linked List in Reverse Order**

Let's see these two methods in more detail:

First, we find the method that will be called from outside the class. We assign a descriptive and concise name to this method. It does not take any parameters.

We pass the head node of the linked list as argument to the recursive method because we want to start the process from the head node.

1. def print\_reversed(self):
2. self.print\_reversed\_recursive(self.head)

Then, (see below) we find the recursive method. This method calls itself repeatedly and it is the building-block of what makes this process work correctly. It takes the instance **self** (the instance of the linked list) and **node**. Initially, this is the head node of the linked list.

1. def print\_reversed\_recursive(self, node):
2. if node is not None:
3. self.print\_reversed\_recursive(node.next)
4. print(node.value, end=" ")

Inside the method, we check if the node is valid (if it is not equal to None).

1. if node is not None:

If the node is valid, we call the method again passing the next node in the linked list as argument.

1. self.print\_reversed\_recursive(node.next)

This recursive call will **pause** the execution of the current method call and start the next one immediately.

This will happen repeatedly once for each node until we reach the end of the linked list.

All the recursive method calls will be paused until the end is reached.

When we reach the end of the linked list, the print statements will run in reverse order because the last recursive call will be completed first and the other recursive calls will be completed one by one from the last one to the first one until the first one is reached.