Common linked list questions include:

- Reversing a linked list
- Detecting a cycle in a linked list
- Removing duplicates from a sorted linked list
- Checking if a linked list represents a palindrome

As an example, below we reverse a linked list. Said another way, given the input linked list $4 \rightarrow 1 \rightarrow 3 \rightarrow 2$, we want to write a function which returns $2 \rightarrow 3 \rightarrow 1 \rightarrow 4$. To implement this, we first start with a basic node class:

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

Then we create the LinkedList class, along with the method to reverse its elements. The reverse function iterates through each node of the linked list. At each step, it does a series of swaps between the pointers of the current node and its neighbors.

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

  def reverse(self):
    prev = None
```

```
    curr = self.head
    while curr:
      next = curr.next
      curr.next = prev
      prev = curr
      curr = next
    self.head = prev
```

Like array interview questions, linked list problems often have an obvious brute-force solution that uses O(n) space, but then also a more clever solution that utilizes the existing list nodes to reduce the memory usage to O(1). Another commonality between array and linked list interview solutions is the prevalence of off-by-one errors. In the linked list case, it's easy to mishandle pointers for the head or tail nodes.