Problem: Implement a linked list with insert, append, find, delete, length, and print methods.

Constraints

- Can we assume this is a non-circular, singly linked list?
    - Yes
- Do we keep track of the tail or just the head?
    - Just the head
- Can we insert None values?
    - No

Test Cases

Insert to Front

- Insert a None
- Insert in an empty list
- Insert in a list with one element or more elements

Append

- Append a None
- Append in an empty list
- Insert in a list with one element or more elements

Find

- Find a None
- Find in an empty list
- Find in a list with one element or more matching elements
- Find in a list with no matches

Delete

- Delete a None
- Delete in an empty list
- Delete in a list with one element or more matching elements
- Delete in a list with no matches

Length

- Length of zero or more elements

Print

- Print an empty list
- Print a list with one or more elements

Algorithm

Insert to Front

- If the data we are inserting is None, return
- Create a node with the input data, set node.next to head
- Assign the head to the node

Complexity:

- Time: O(1)
- Space: O(1)

Append

- If the data we are inserting is None, return
- Create a node with the input data
- If this is an empty list
    - Assign the head to the node
- Else
    - Iterate to the end of the list
    - Set the final node's next to the new node

Complexity:

- Time: O(n)
- Space: O(1)

Find

- If data we are finding is None, return
- If the list is empty, return
- For each node
    - If the value is a match, return it
    - Else, move on to the next node

Complexity:

- Time: O(n)

- Space: O(1)

Delete

- If data we are deleting is None, return
- If the list is empty, return
- For each node, keep track of previous and current node
    - If the value we are deleting is a match in the current node
        - Update previous node's next pointer to the current node's next pointer
        - We do not have have to explicitly delete in Python
    - Else, move on to the next node
- As an alternative, we could avoid the use of two pointers by evaluating the current node's next value:
    - If the next value is a match, set the current node's next to next.next
    - Special care should be taken if deleting the head node

Complexity:

- Time: O(n)
- Space: O(1)

Length

- For each node
    - Increase length counter

Complexity:

- Time: O(n)
- Space: O(1)

Print

- For each node
    - Print the node's value

Complexity:

- Time: O(n)
- Space: O(1)