UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 6

# Singly Linked Lists

*Submitted by:*
Ampong, J-kevin L.

*Instructor:*
Engr. Maria Rizette H. Sayo

August, 23, 2025

## I. Objectives

Introduction

A linked list is an organization of a list where each item in the list is in a separate node. Linked lists look like the links in a chain. Each link is attached to the next link by a reference that points to the next link in the chain. When working with a linked list, each link in the chain is called a Node. Each node consists of two pieces of information, an item, which is the data associated with the node, and a link to the next node in the linked list, often called next.

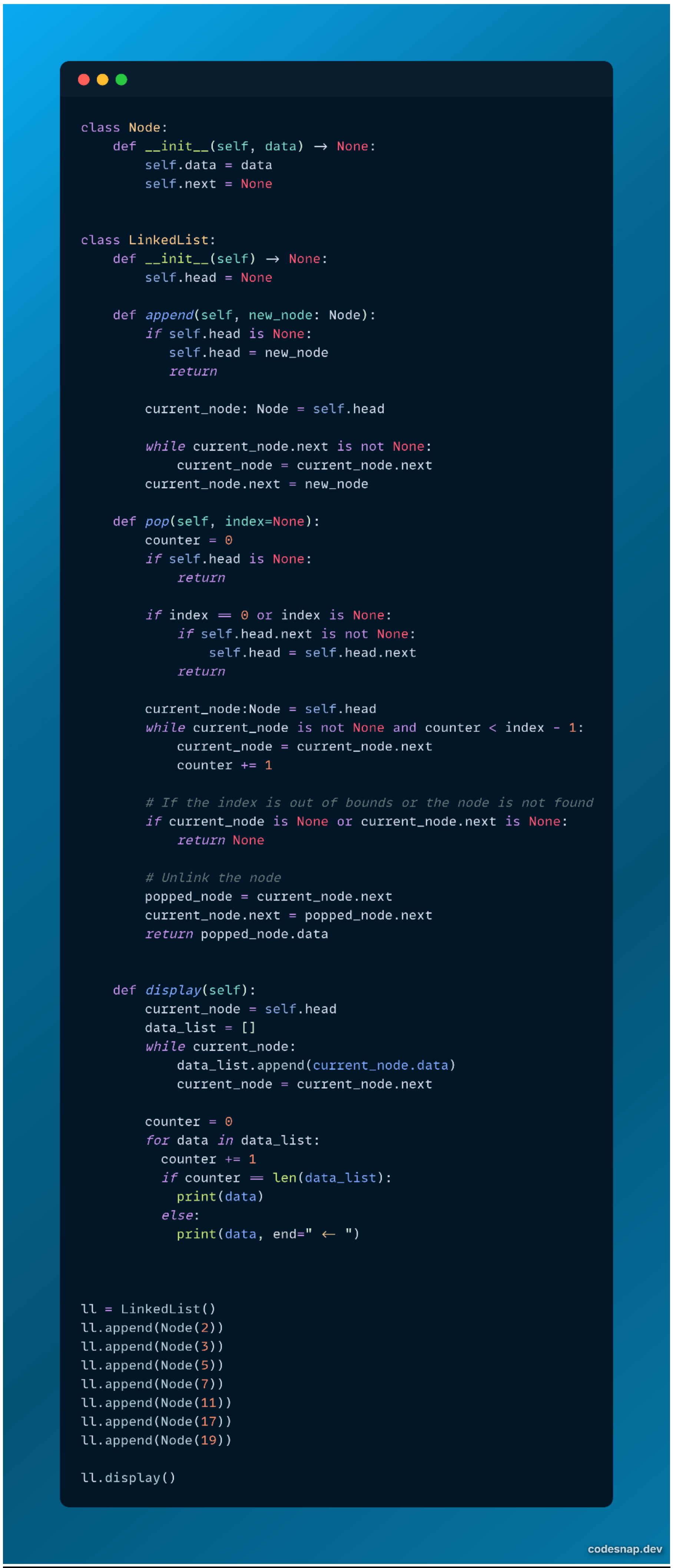
This laboratory activity aims to implement the principles and techniques in:

- Writing algorithms using Linked list
- Writing a python program that will perform the common operations in a singly linked list

## II. Methods

- Write a Python program to create a singly linked list of prime numbers less than 20. By iterating through the list, display all the prime numbers, the head, and the tail of the list. (using Google Colab)
- Save your source codes to GitHub

## III. Results



```python
class Node:
    def __init__(self, data) -> None:
        self.data = data
        self.next = None


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

    def append(self, new_node: Node):
        if self.head is None:
           self.head = new_node
           return

        current_node: Node = self.head

        while current_node.next is not None:
            current_node = current_node.next
        current_node.next = new_node

    def pop(self, index=None):
        counter = 0
        if self.head is None:
            return

        if index == 0 or index is None:
            if self.head.next is not None:
                self.head = self.head.next
            return

        current_node:Node = self.head
        while current_node is not None and counter < index - 1:
            current_node = current_node.next
            counter += 1

        # If the index is out of bounds or the node is not found
        if current_node is None or current_node.next is None:
            return None

        # Unlink the node
        popped_node = current_node.next
        current_node.next = popped_node.next
        return popped_node.data


    def display(self):
        current_node = self.head
        data_list = []
        while current_node:
            data_list.append(current_node.data)
            current_node = current_node.next

        counter = 0
        for data in data_list:
          counter += 1
          if counter == len(data_list):
            print(data)
          else:
            print(data, end=" <- ")



ll = LinkedList()
ll.append(Node(2))
ll.append(Node(3))
ll.append(Node(5))
ll.append(Node(7))
ll.append(Node(11))
ll.append(Node(17))
ll.append(Node(19))

ll.display()
```

Figure 1 Screenshot of program

Conclusion

LinkedList is another kind of list where all its collection is node, and each node contains two parameters such as the data itself and the next which is a pointer to another node and only has two states either a node or none which is a Null in another programming language. Ther node property is unique in design its purpose is to be connected to another node but has its difficult level to be implemented since it’s not primitive type unlike the list, dictionary and tuple.

While the advantages of Node and LinkedList are different from another array type it is only useful in certain areas and it’s not commonly used in small to medium projects that don’t deals on relational system

## References

[1] Co Arthur O.. "University of Caloocan City Computer Engineering Department Honor Code," UCC-CpE Departmental Policies, 2020.

[2] GeeksforGeeks, "Python Linked List," GeeksforGeeks, Jul. 23, 2025. https://www.geeksforgeeks.org/python/python-linked-list/