UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 6

Singly Linked Lists

Submitted by:
Vasig, Yuan Hessed O.

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:

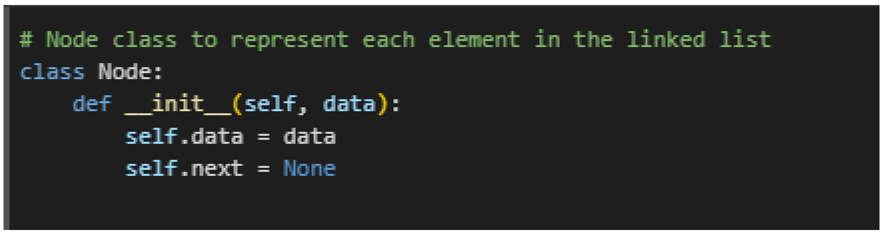
- Writing algorithms using Linked list
- Writing a python program that will perform the common operations in a singly linked list

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

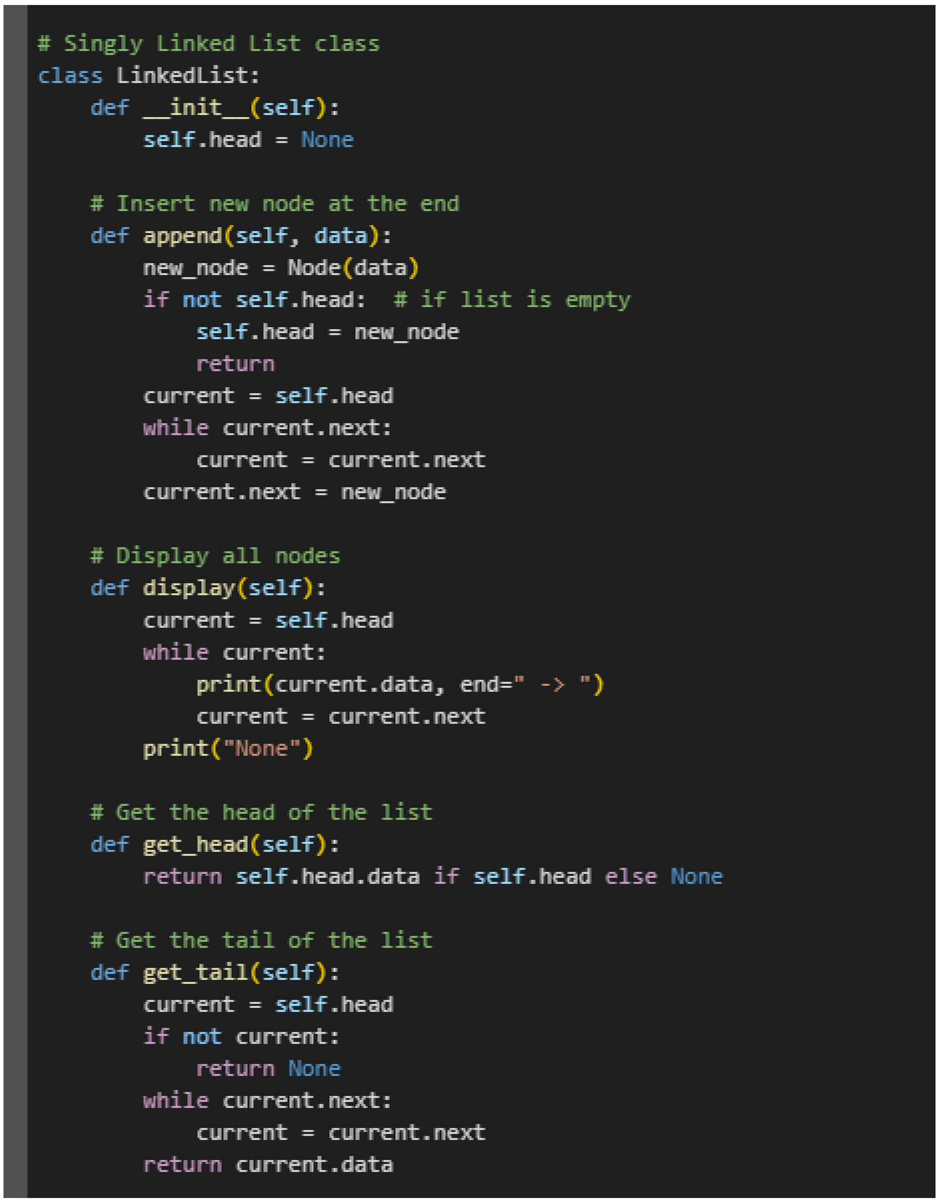
Source Code



```python
# Node class to represent each element in the linked list
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
```

Figure 1 Source Code(Class Node)

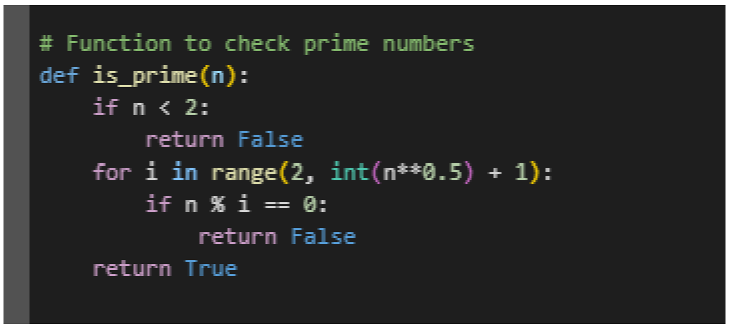
I first created a class specifically for initializing the nodes for the linked list. Which contains self.data, which contains the value of the nodes and self.next which is a pointer to the nodes.



```
# Singly Linked List class
class LinkedList:
    def __init__(self):
        self.head = None

    # Insert new node at the end
    def append(self, data):
        new_node = Node(data)
        if not self.head:  # if list is empty
            self.head = new_node
            return
        current = self.head
        while current.next:
            current = current.next
        current.next = new_node

    # Display all nodes
    def display(self):
        current = self.head
        while current:
            print(current.data, end=" -> ")
            current = current.next
        print("None")

    # Get the head of the list
    def get_head(self):
        return self.head.data if self.head else None

    # Get the tail of the list
    def get_tail(self):
        current = self.head
        if not current:
            return None
        while current.next:
            current = current.next
        return current.data
```

Figure 2 Source Code(Class ‘LinkedList’)

Here I created a class named “LinkedList”. First I initialize self.head, meaning to point to the first node in the list if it’s empty then it’s None. I then created the function **append** to add a new node at the end of the list. Then the function **display**, then prints the contents of the list in order. And lastly, the **get_head** and **get_tail** function which then returns the head or the first node, and the tail which is the last in the list.



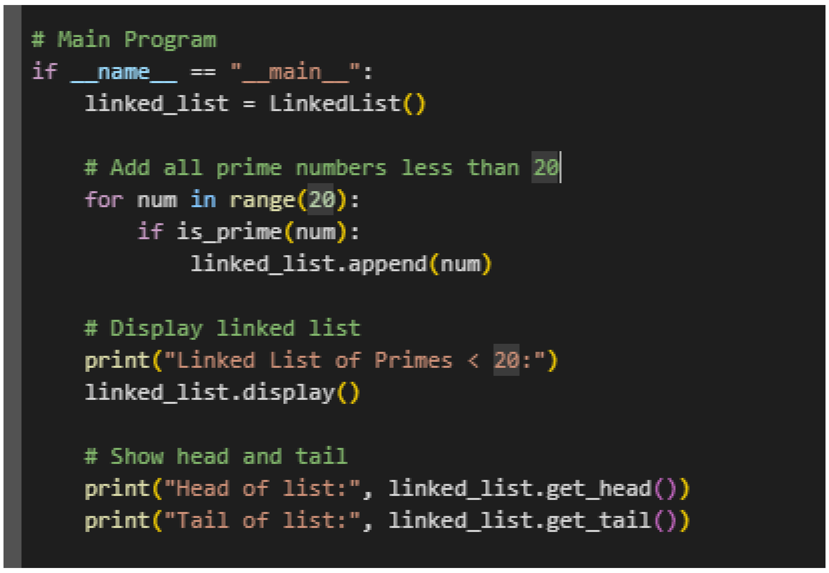
```
# Function to check prime numbers
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
```

Figure 3 Function 'is_prime'

- **Purpose**: Checks if a number n is prime.
- **Logic**:
    1. Numbers < 2 are not prime.
    2. Check divisibility from 2 up to √n.
    3. If divisible, not prime → return False.
    4. Otherwise, return True.

Example:

- is_prime(7) → True
- is_prime(9) → False



```
# Main Program
if __name__ == "__main__":
    linked_list = LinkedList()

    # Add all prime numbers less than 20
    for num in range(20):
        if is_prime(num):
            linked_list.append(num)

    # Display linked list
    print("Linked List of Primes < 20:")
    linked_list.display()

    # Show head and tail
    print("Head of list:", linked_list.get_head())
    print("Tail of list:", linked_list.get_tail())
```

Figure 4 Main Program

Steps:

1. Create a new empty linked list.
2. Loop through numbers from 0 to 19.

   If a number is prime, append it to the list.
3. Display the whole list.
4. Show the head (first element) and tail (last element).

IV. Conclusion

In conclusion, I've learned how I can utilize a single linked list in the future, by being able to manipulate how your data structure flows in a system. In this case, primes is an example of a use for a single linked list.

References

[1] G. Refsnes, *Python Linked Lists*, W3Schools, 2025. [Online]. Available: https://www.w3schools.com/python/python_dsa_linkedlists.asp. [Accessed: 23-Aug-2025].

[2] Google, *Welcome to Colaboratory*, Google Colab Documentation. [Online]. Available: https://colab.research.google.com/. [Accessed: 23-Aug-2025].

[3] Python Software Foundation, *Python Documentation*. [Online]. Available: https://www.python.org/doc/. [Accessed: 23-Aug-2025].

[4] G. H. Gonnet and R. Baeza-Yates, *Handbook of Algorithms and Data Structures: In Pascal and C*, 2nd ed. Reading, MA, USA: Addison-Wesley, 1991. (For general algorithms including doubly linked lists).