Data Structure and Algorithm

Laboratory Activity No. 6

Singly Linked Lists

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Hermosura, Leigh B. | Engr. Maria Rizette H. Sayo |

August 23, 2025

# 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

# 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

# Results

This part will present the results of the laboratory activity, complete with images and explanation for each segment.

The program consists of two classes, the *Node* and *LinkedList*. The *Node* class initializes the data field and pointer field as *self.data* and *self.next* respectively. the *LinkedList* class contains five functions. First is the constructor function that initializes the head of the linked list. The *print\_all* function defines a variable called *current* and sets it equal to *self.head*, it runs through a *while* loop that prints the *current* variable then it sets it to *current.next* which prints the data from start to finish. The *append* function creates a new *Node* object with the *new\_data*. It then checks if the list is empty, if so, it sets the *new\_data* as the head of the list. A *while* loop is used to traverse the list and append the *new\_data* at the end. The *giveHead* function checks if the list has a head node, otherwise it displays that the list is empty. Lastly, the *giveTail* function also checks if the list is empty. If it isn’t, it iterates through every item and prints the last item in the list.

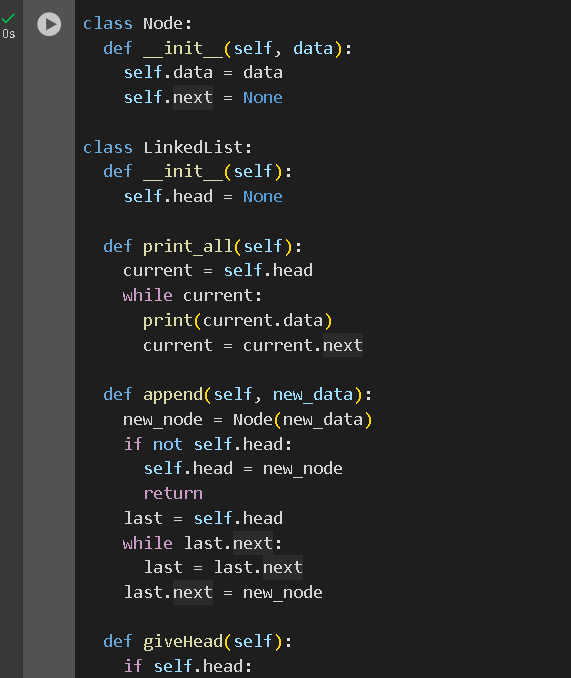


Figure 1 The *Node* and *LinkedList* class

Outside the two classes is a function that generates prime numbers less than 20 called *primeGenerator*. The function runs a nested *for* loop. The outer loop sets the range where prime numbers will be generated, in this case, 2 to 20. The inner loop checks the numbers generated if they have factors other than the number itself and 1. If *num* is divisible by *i*, it’s not a prime number and the loop breaks, otherwise it appends the prime numbers to the linked list.

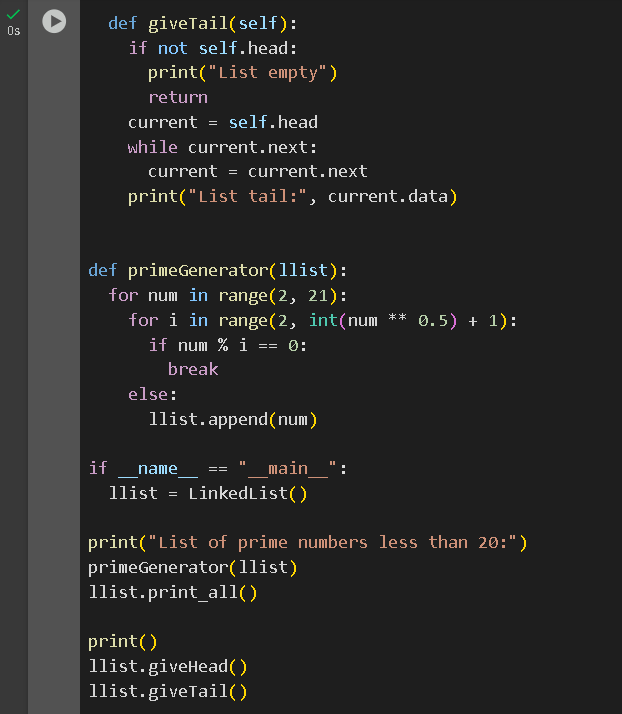


Figure 2 Continuation of the code

The list is then printed along with its head and tail node. The output of the program is provided below.

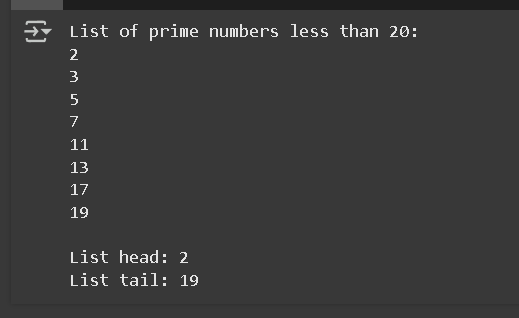


Figure 3 The output of the program

# Conclusion

The laboratory activity presents the structure and functionality of singly linked lists in Python. Through the Node and LinkedList classes, fundamental operations such as appending elements, traversing the list, and accessing the head and tail nodes were implemented. Overall, the program presents the concepts of object-oriented programming and data organization using linked lists.