**ASSIGNMENT-11.4**

NAME:-E.HARINI,

BRANCH: -CSE,

ROLLNUMBER: 2503A51L40, COURSE: -AI ASSISTED CODING.

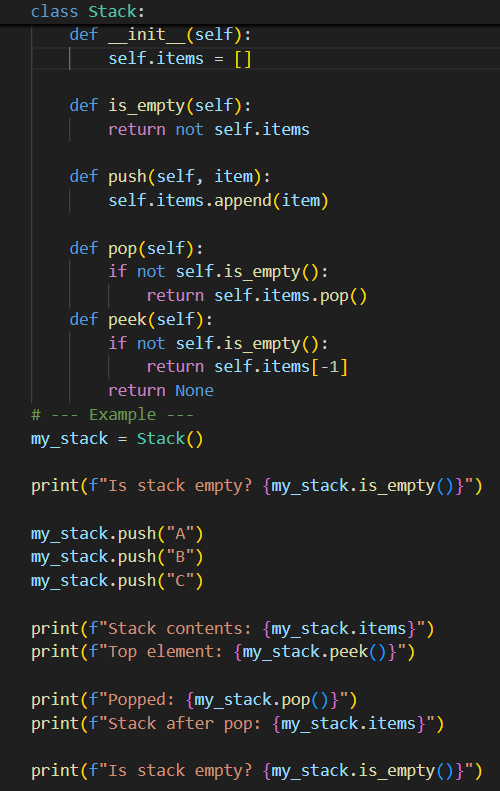
**TASK1:**

Implementing a Stack (LIFO)  
• Task:Use AI to help implement a Stack class in Python with the  
following operations: push(), pop(), peek(), and is\_empty().

**1.PROMPT:-**

Implement a Stack class in Python with methods push(), pop(), peek(), and is\_empty(). Add Google-style docstrings and comments explaining each part in simple language.

**2.code:**



**3.output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**4.OBSERVATION:**

The code demonstrates the **Last-In, First-Out (LIFO)** . When you push elements, they are added to the end of the list. When you pop or peek, you interact with the last element added, effectively treating the end of the list as the "top" of the stack.

**TASK2:**

Queue Implementation with Performance Review  
• Task: Implement a Queue with enqueue(), dequeue(), and is\_empty()  
methods.

**1.PROMPT:**

Illustrate the behavior of the Queue class by enqueuing items and then dequeuing them, observing the order of operations.

**2.Code:**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**3.output:**

A black background with white text

AI-generated content may be incorrect.

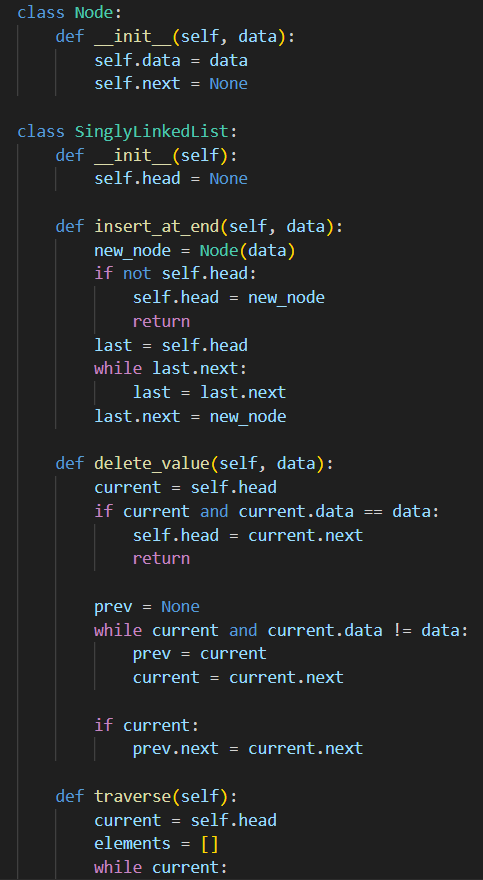
**4.observation:** Queues (FIFO) can be implemented using Python lists. The enqueue() operation adds elements to the end of the list.

**Task3**: Singly Linked List with Traversal  
• Task: Implement a Singly Linked List with operations:  
insert\_at\_end(), delete\_value(), and traverse().

**1.Prompt**:

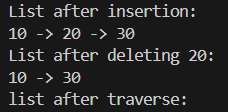
Write a Python program to implement a singly linked list.

**2.Code:**

 A screen shot of a computer program

AI-generated content may be incorrect.

**3.output:**

****

**4.observation:**

The required implementation involves creating a Node class and a SinglyLinkedList class with specific methods.

**Task4:** Binary Search Tree (BST)  
• Task: Implement a Binary Search Tree with methods for insert(),  
search(), and inorder\_traversal()

**1.Prompt**:

Write a Python program to implement a Binary Search Tree (BST).

**2.Code:**

A screen shot of a computer program

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.

**3.output:**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**4.observation:**

The user's request is for a standard BST implementation. The observation will detail the structure and behavior of the BST as the provided operations are performed.

**Task5**: Graph Representation and BFS/DFS Traversal  
• Task: Implement a Graph using an adjacency list, with traversal  
methods BFS() and DFS()

**1.Prompt**:

Implement Breadth-First Search (BFS) and Depth-First Search (DFS) traversal algorithms.

**2.Code:**

A screen shot of a computer program

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.

**3.output:**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**4.observation:**

It requires two core components an adjacency list representation and two distinct traversal algorithms.