**SR UNIVERSITY**

**ASSIGNMENT N0 # 11**

**NAME :VASANTA**

**HALL TICKET NUMBER:2403A54127**

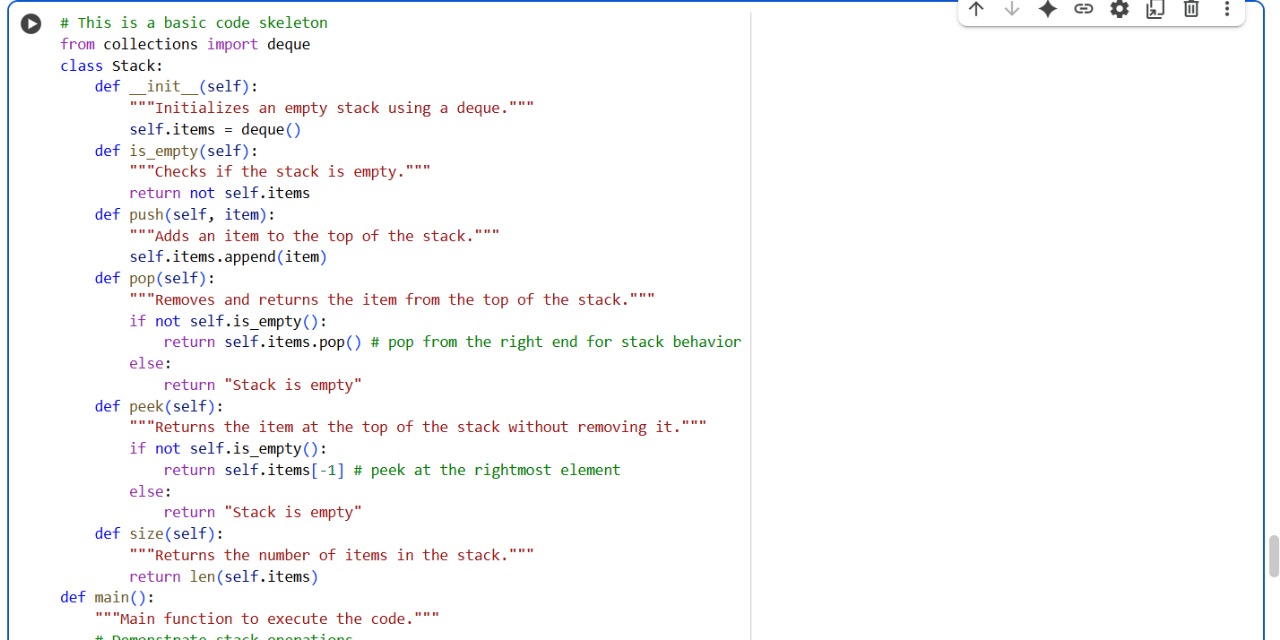
**BATCH NO:4(DS)**

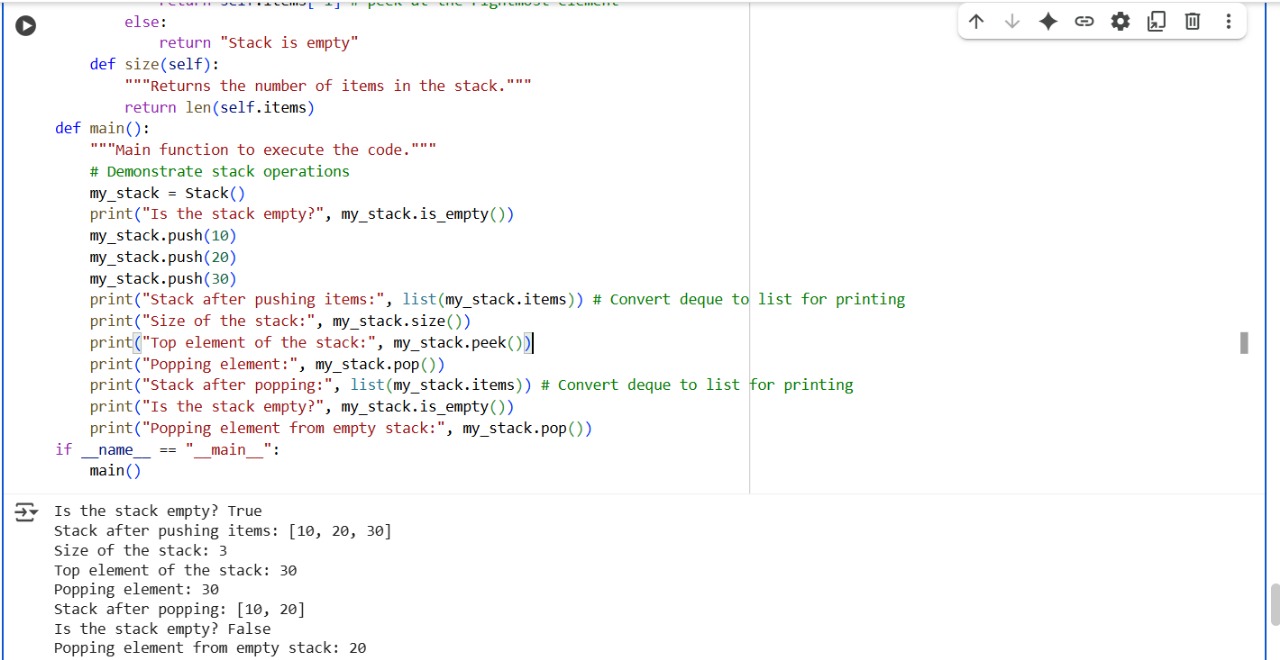
**Task 1: Implementing a Stack (LIFO)**

**PROMPT:**

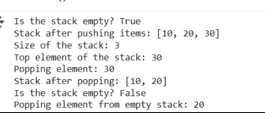
**write a Python Stack class with push(), pop(), peek(), and is\_empty() methods. Include a code skeleton with Google-style docstrings. After that, help me test it with some sample data. Then suggest optimizations or alternatives like using collections.deque.**

**CODE:**





**OUTPUT:**



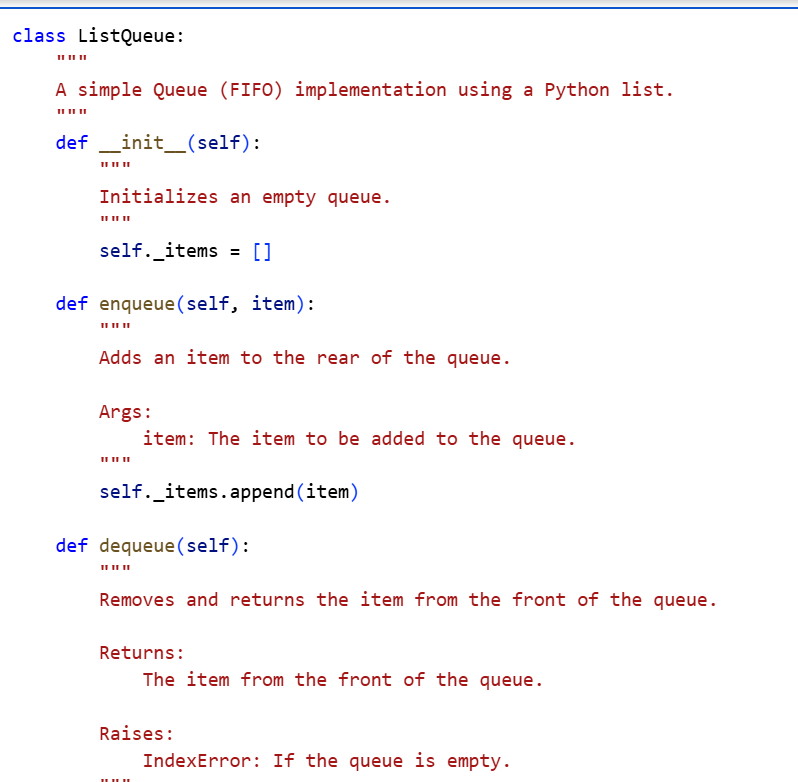
**EXOLANATION:**

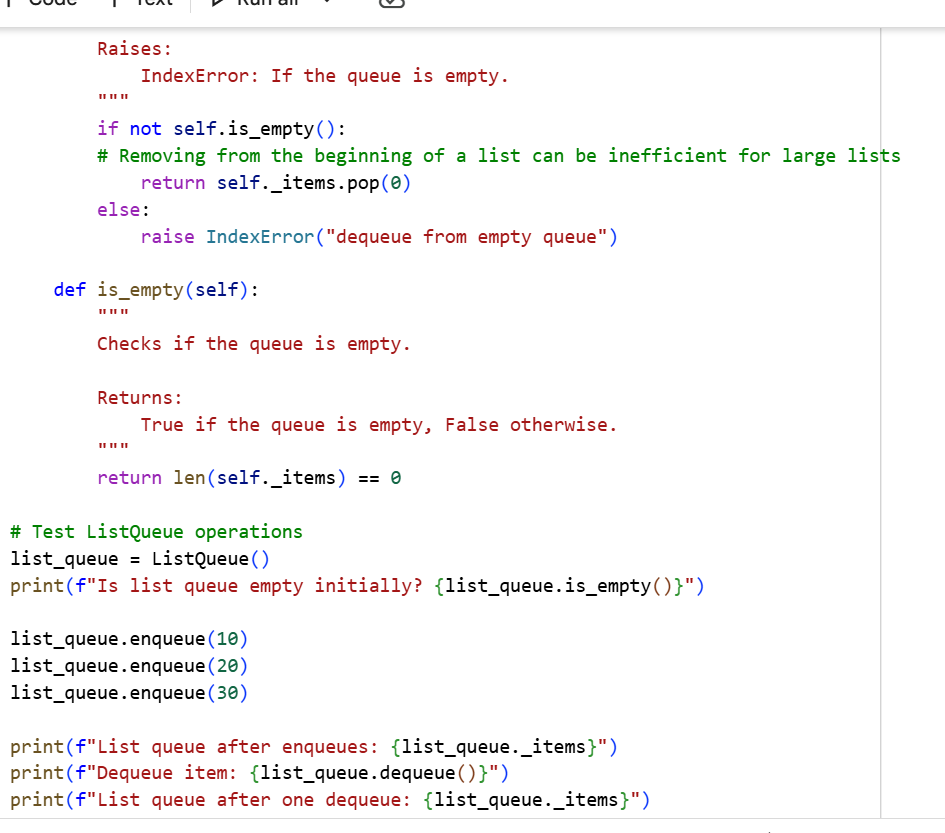
**Task 2: Queue Implementation with Performance Review**

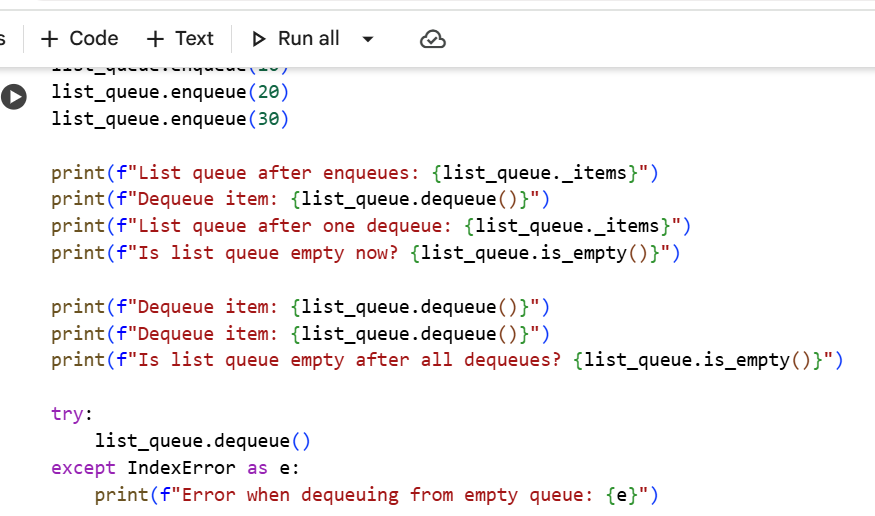
**PROMPT:**

**Implement a Queue class in Python with enqueue(), dequeue(), and is\_empty() using a list.**

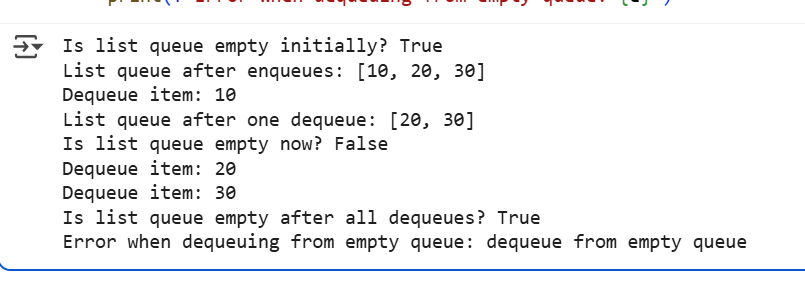
**CODE:**

****

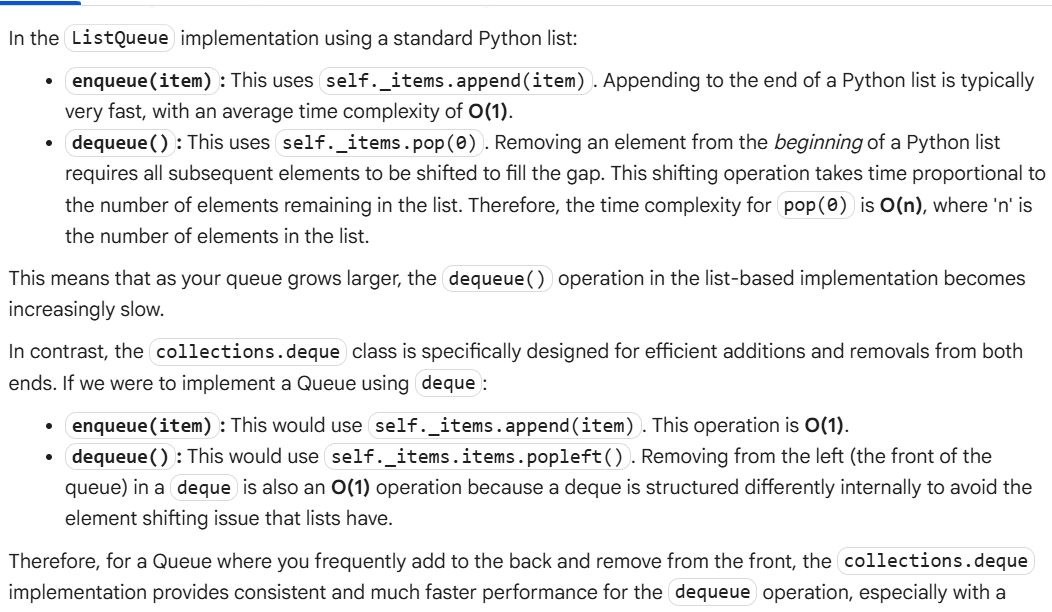
****

****

**OUTPUT:**

****

**EXPLANATION:**

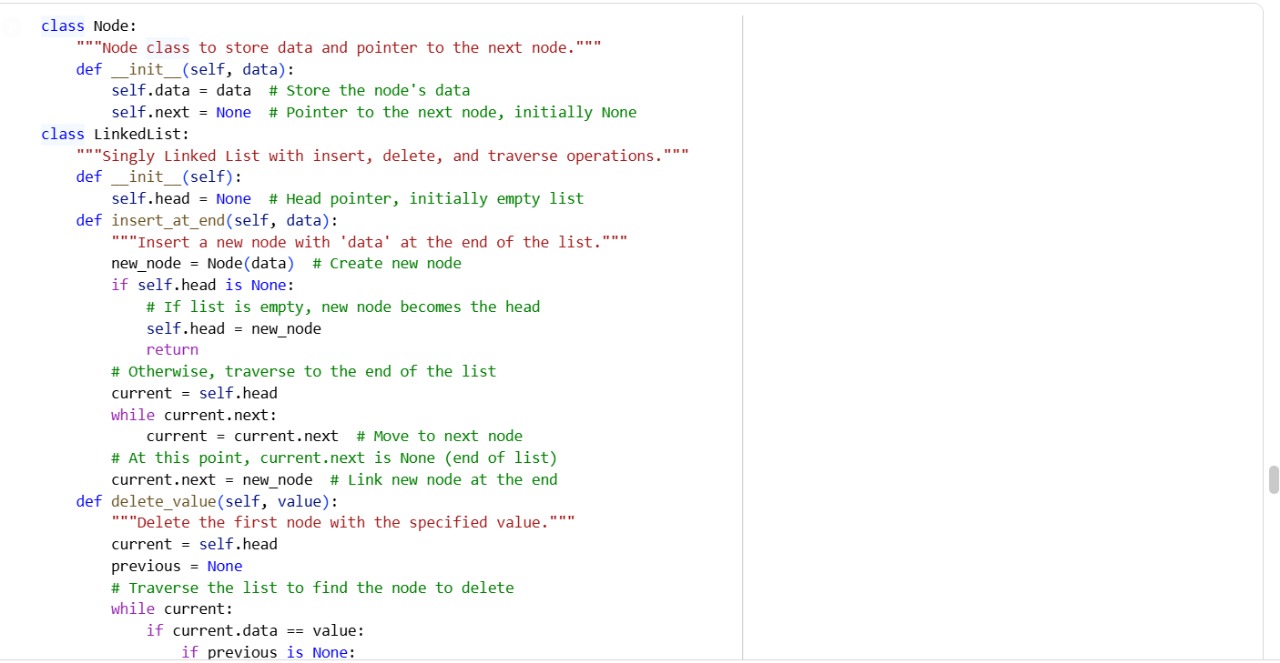


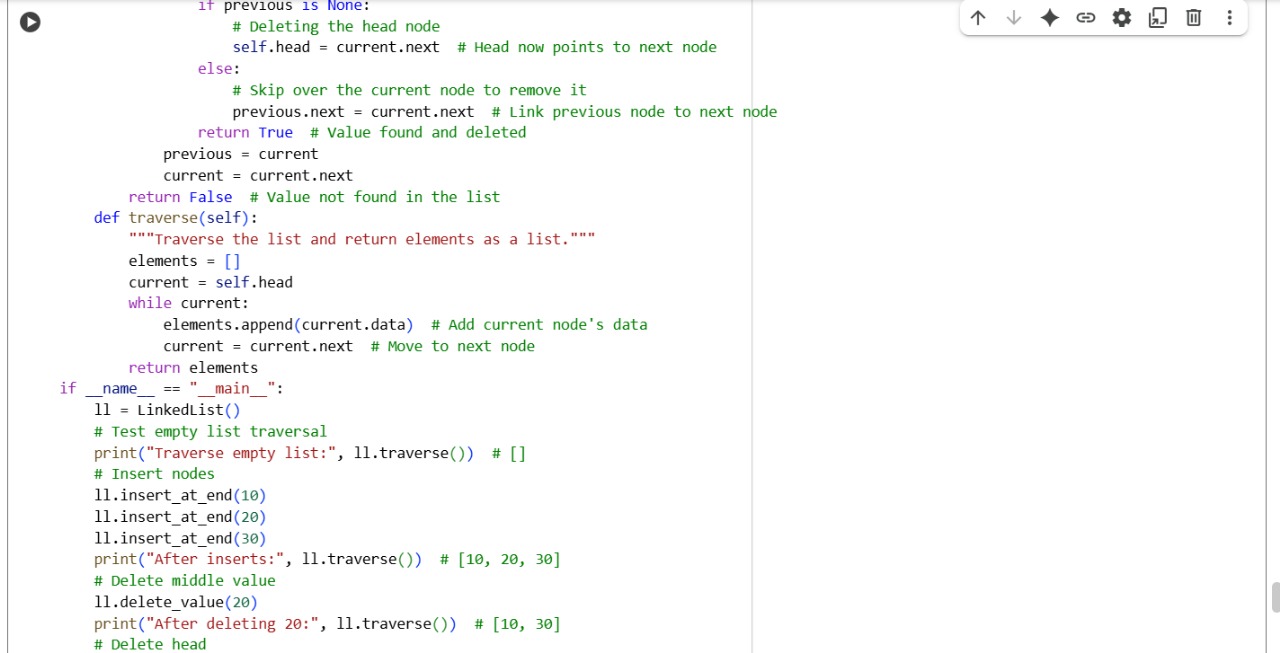
**Task 3: Singly Linked List with Traversal**

**PROMPT:**

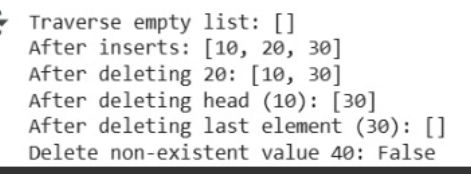
**Implement a singly linked list in Python with insert\_at\_end(), delete\_value(), and traverse() methods. Use a class-based approach with Node and LinkedList classes. Add inline comments to explain pointer updates during insertions and deletions. Also, suggest test cases to validate each operation**

**CODE:**

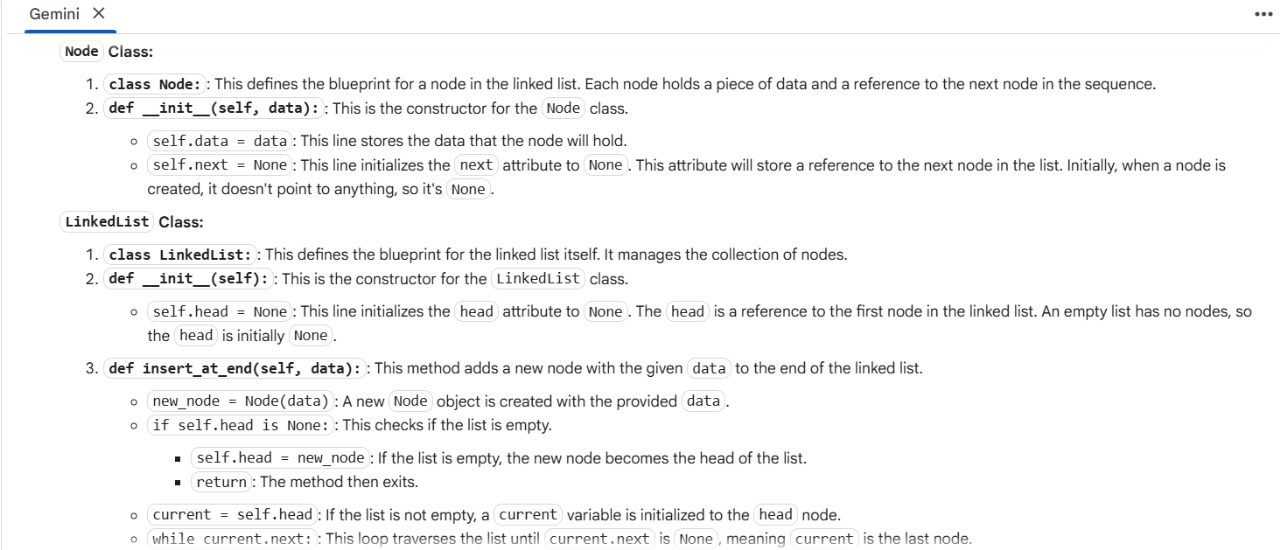


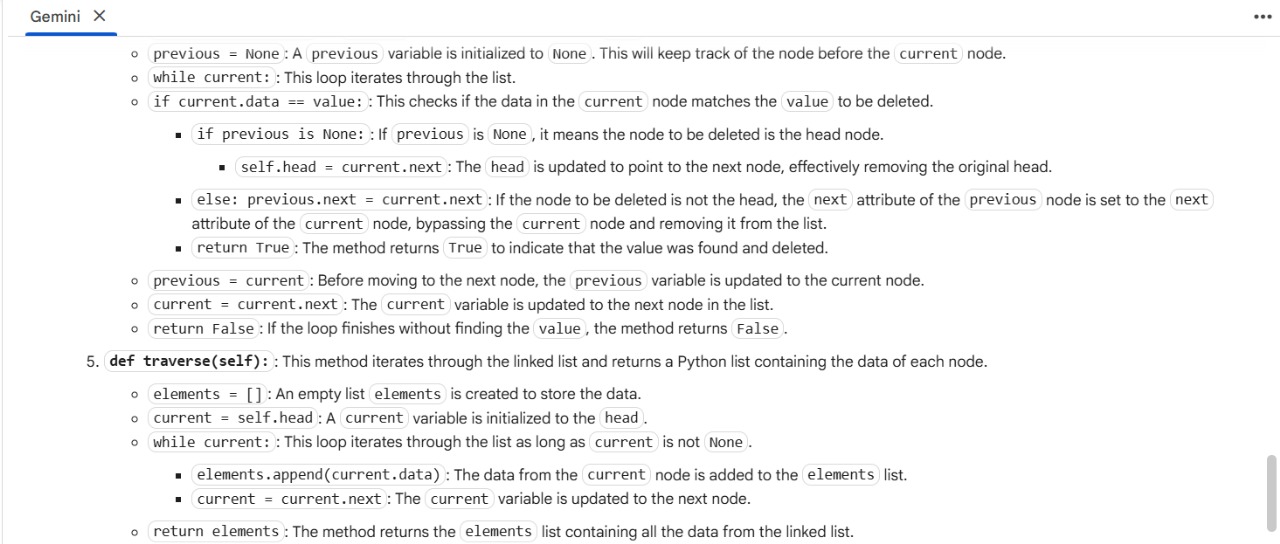


**OUTPUT:**



**EXPLANATION:**



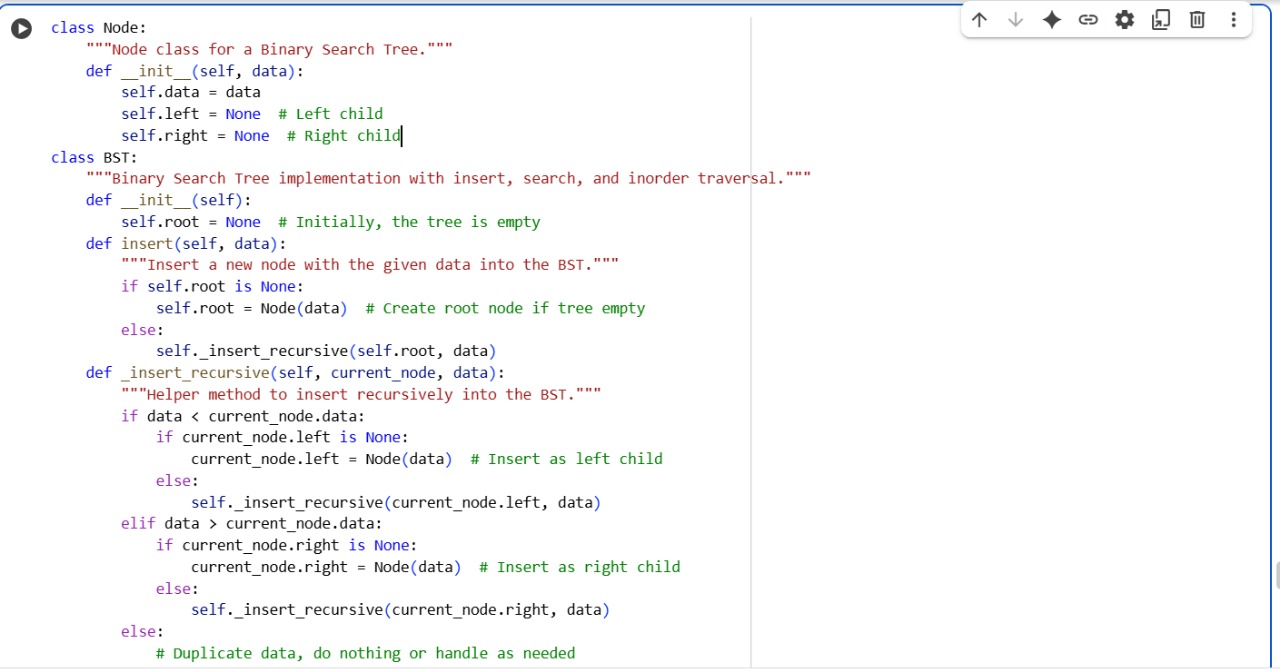


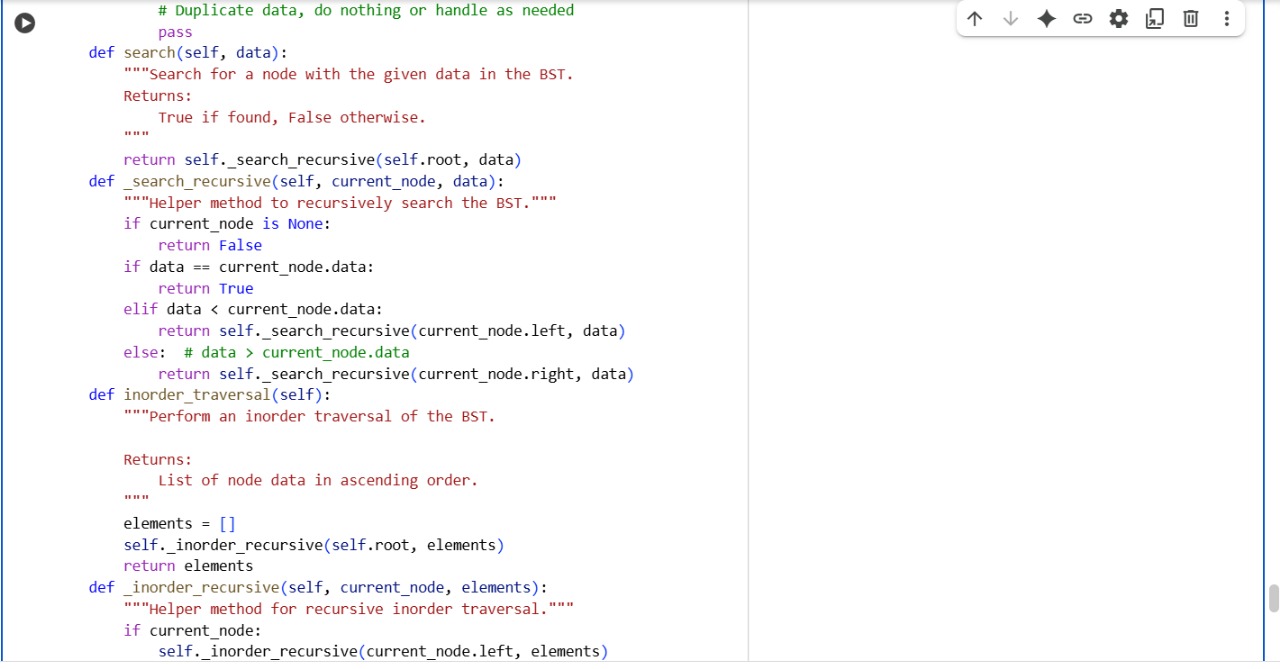
**Task 4: Binary Search Tree (BST)**

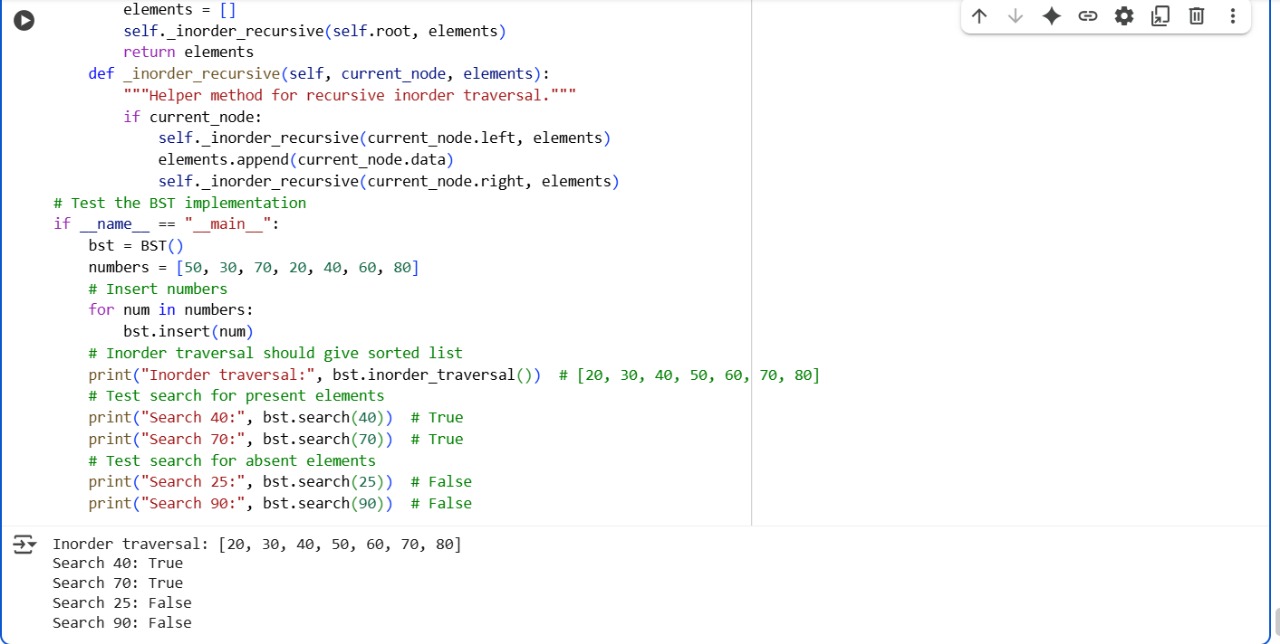
**PROMPT:**

**written Python class for a Binary Search Tree with Node and BST classes. Help me complete the missing methods: insert(), search(), and inorder\_traversal(). Add meaningful Google-style docstrings. Then show how to test the BST with a list of integers and compare the search() results for both present and absent elements.**

**CODE:**



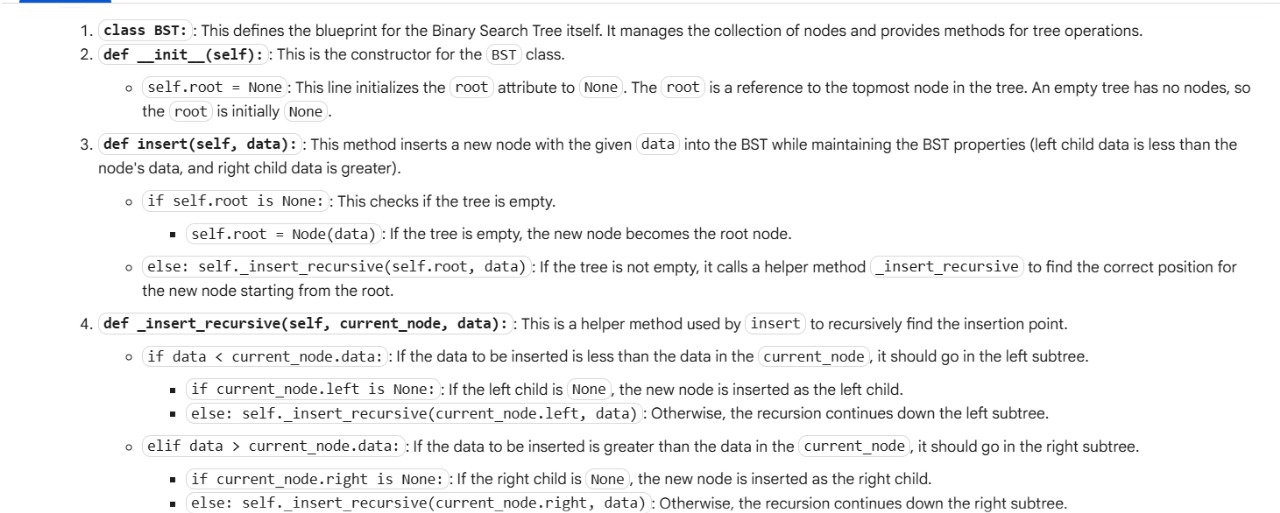


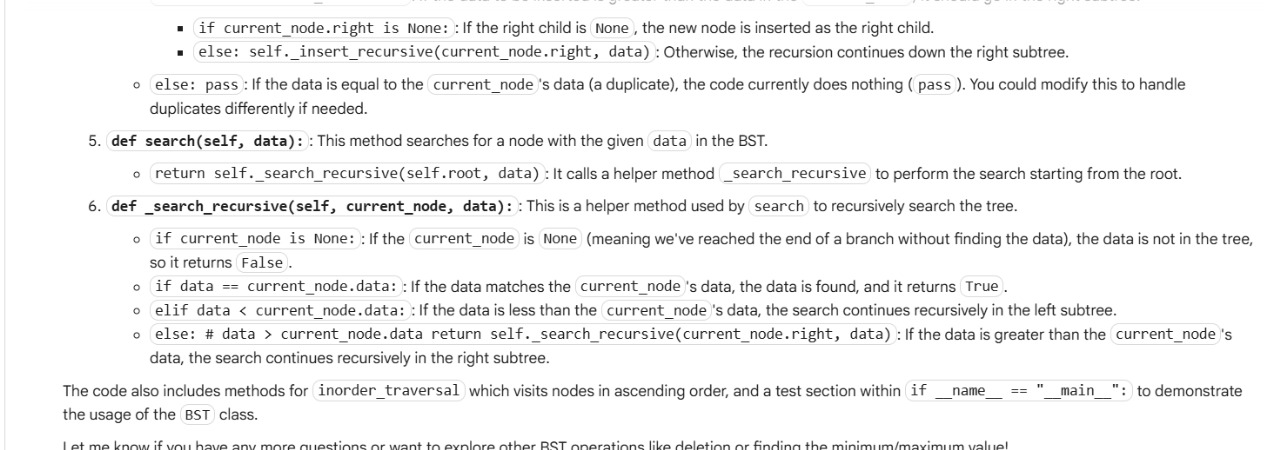


**OUTPUT:**



**EXPLANATION:**



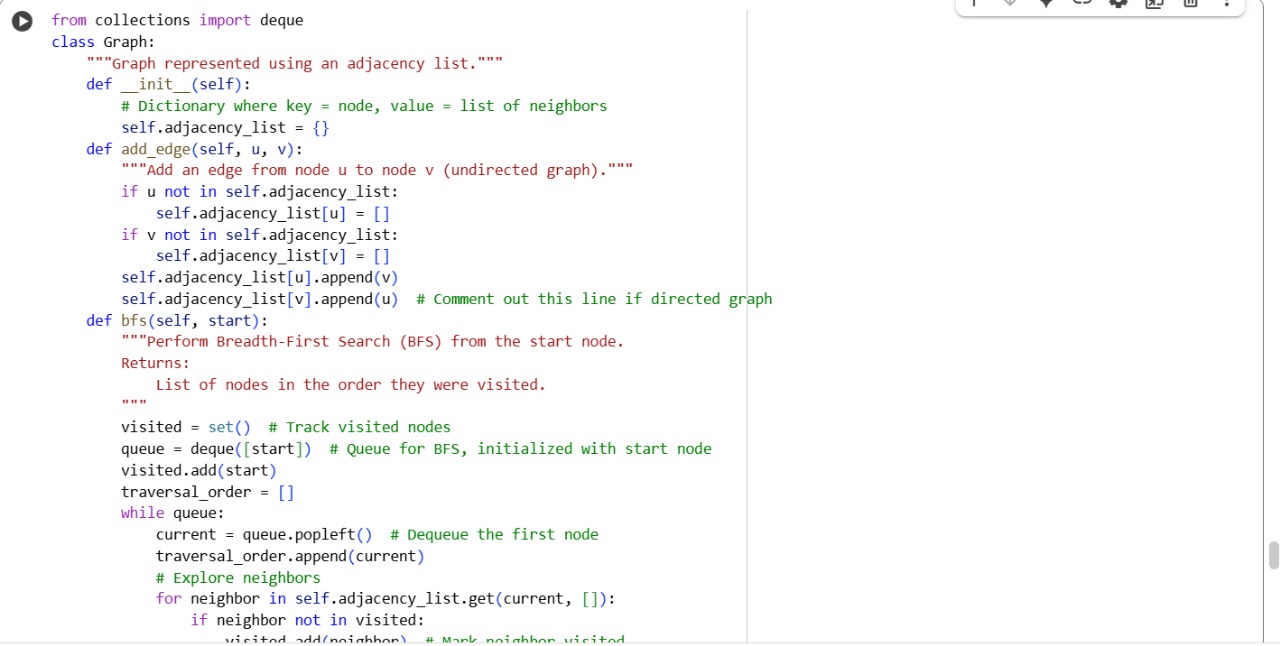


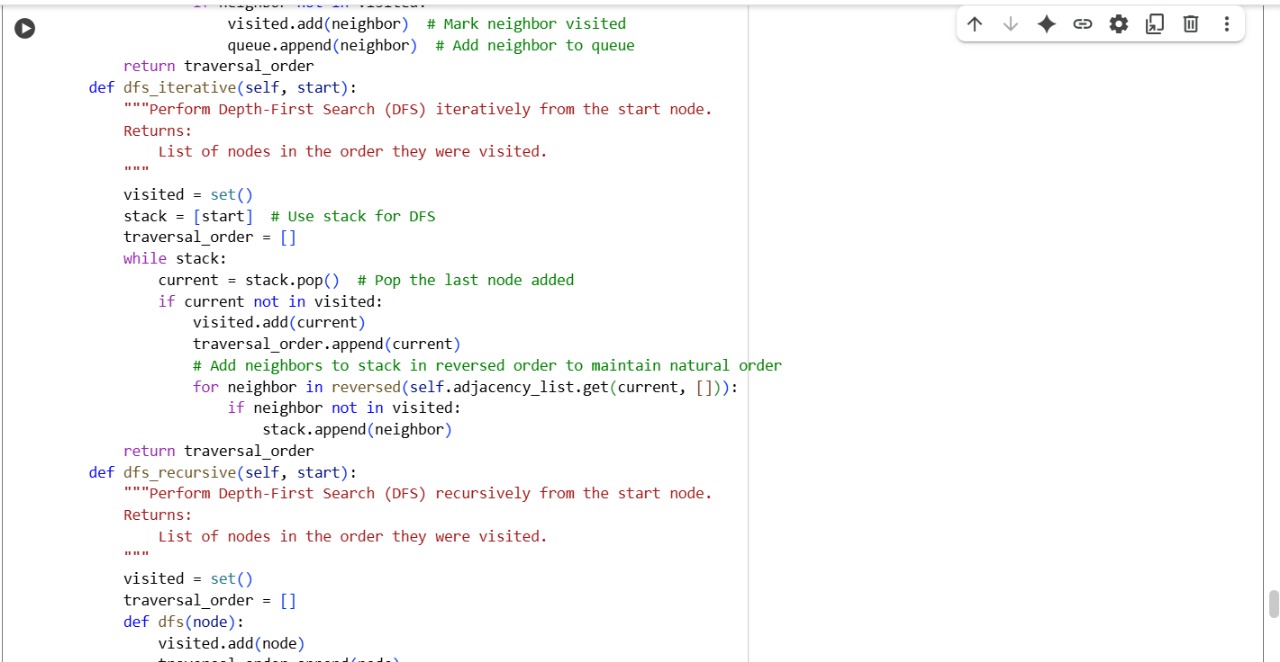
**Task 5: Graph Representation and BFS/DFS Traversal**

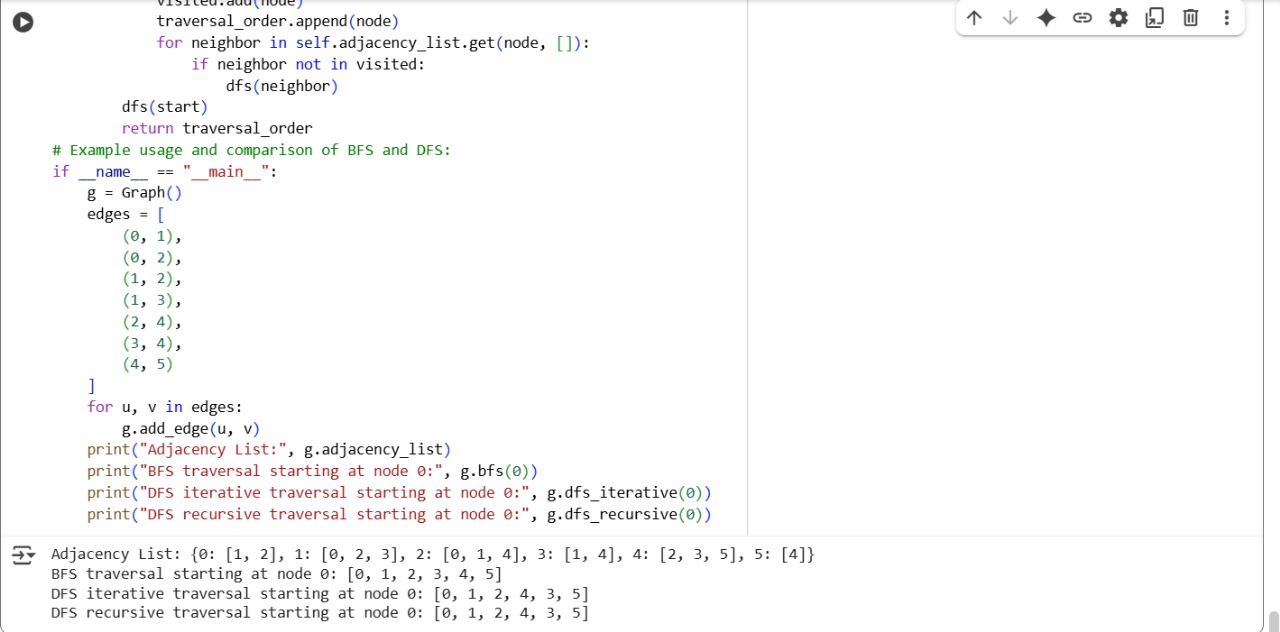
**PROMPT:**

**Python using an adjacency list (dictionary). Then write both BFS() and DFS() traversal methods with inline comments explaining each step. Also, compare recursive vs iterative DFS if there’s a difference in behavior or performance**

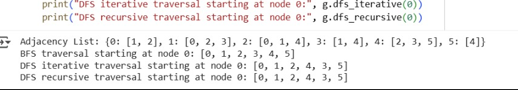
**CODE:**







**OUTPUT:**



**EXPLANATION:**

