**AI ASSISTED CODING**

ASSIGNMENT-11.1

Name: S. Vrindha Reddy

Hall Number.: 2403A51255

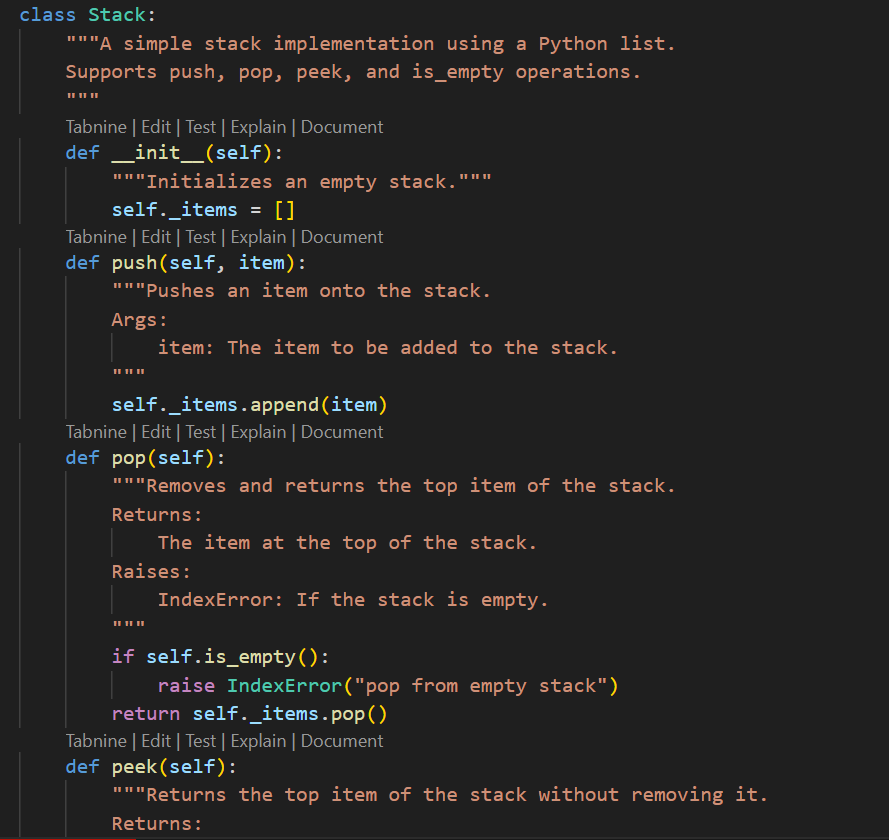
Batch No.:11

**Task 1: 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().  
• Instructions:  
o Ask AI to generate code skeleton with docstrings.  
o Test stack operations using sample data.  
o Request AI to suggest optimizations or alternative  
implementations (e.g., using collections.deque).  
**• Expected Output:**o A working Stack class with proper methods, Google-style  
docstrings, and inline comments for tricky parts.

**Prompt:**

Write a Python Stack class with methods push (), pop (), peek (), and is\_empty (). Include Google-style docstrings and inline comments. Provide a simple test using sample data. Finally, suggest possible optimizations or alternative implementations (e.g., using collections. deque.

**Code:**

****

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Task 2: Queue Implementation with Performance Review**• Task: Implement a Queue with enqueue(), dequeue(), and is\_empty()  
methods.  
• Instructions:  
o First, implement using Python lists.  
o Then, ask AI to review performance and suggest a more  
efficient implementation (using collections.deque).  
**• Expected Output:**o Two versions of a queue: one with lists and one optimized with  
deque, plus an AI-generated performance comparison.

**Prompt:**

Implement a Python singly linked list with Node and LinkedList classes. Include methods: insert\_at\_end (), delete\_value (), and traverse (). Add inline comments to explain pointer updates clearly. Suggest test cases to validate all operations.

**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.**

**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.**

**Output:**

**A computer screen with blue text

AI-generated content may be incorrect.**

**Task 3: Singly Linked List with Traversal**• Task: Implement a Singly Linked List with operations:  
insert\_at\_end(), delete\_value(), and traverse().  
• Instructions:  
o Start with a simple class-based implementation (Node,  
LinkedList).  
o Use AI to generate inline comments explaining pointer updates  
(which are non-trivial).  
o Ask AI to suggest test cases to validate all operations.  
**• Expected Output:**o A functional linked list implementation with clear comments  
explaining the logic of insertions and deletions.

**Prompt:**

Implement a Python singly linked list with Node and LinkedList classes. Include methods: insert\_at\_end (), delete\_value (), and traverse (). Add inline comments to explain pointer updates clearly. Suggest test cases to validate all operations

**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.**

**A computer screen shot of a program code

AI-generated content may be incorrect.**

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Task 4: Binary Search Tree (BST)**• Task: Implement a Binary Search Tree with methods for insert (),  
search (), and inorder\_traversal ().  
• Instructions:  
o Provide AI with a partially written Node and BST class.

o Ask AI to complete missing methods and add docstrings.  
o Test with a list of integers and compare outputs of search () for  
present vs absent elements.  
**• Expected Output:**o A BST class with clean implementation, meaningful docstrings,  
and correct traversal output.

**Prompt:**

Complete a Python BinarySearchTree with Node and BST classes. Implement methods: insert (), search (), and inorder\_traversal (). Add Google-style docstrings for all methods. Test the BST with a list of integers and compare search () results for both present and absent elements.

**Code:**

**A computer screen shot of a program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A computer screen shot of a program code

AI-generated content may be incorrect.**

**Output:**

**A black screen with white text

AI-generated content may be incorrect.**

**Task 5: Graph Representation and BFS/DFS Traversal**• Task: Implement a Graph using an adjacency list, with traversal  
methods BFS() and DFS().  
• Instructions:  
o Start with an adjacency list dictionary.  
o Ask AI to generate BFS and DFS implementations with inline  
comments.  
o Compare recursive vs iterative DFS if suggested by AI.  
**• Expected Output:**o A graph implementation with BFS and DFS traversal methods,  
with AI-generated comments explaining traversal steps.

**Prompt:**

Implement a Python Graph using an adjacency list (dictionary). Add traversal methods BFS () and DFS () with inline comments explaining each step. If possible, compare recursive vs iterative DFS approaches. Provide example usage with sample graph data.

**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.

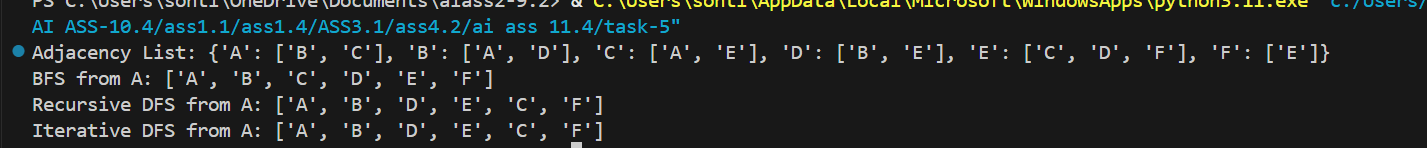
A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program code

AI-generated content may be incorrect.

**Output:**

****