**Task Description #1 – Stack Implementation**

Task: Use AI to generate a Stack class with push, pop, peek, and is\_empty methods.

Sample Input Code:

class Stack:

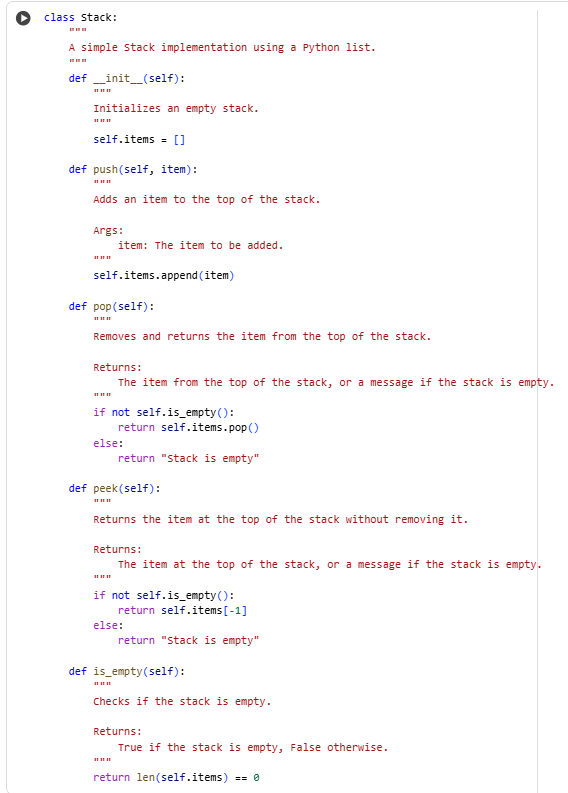
pass

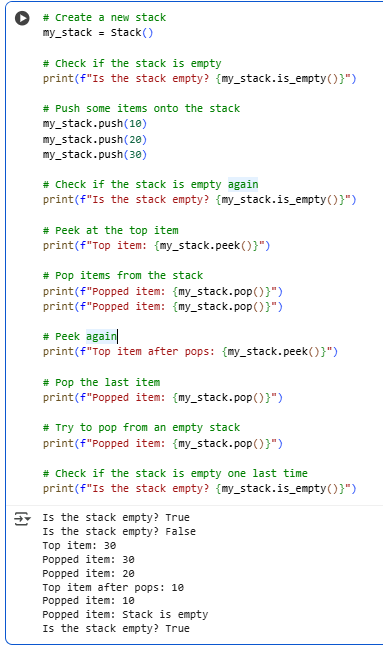
Expected Output:

* A functional stack implementation with all required methods and docstrings.

PROMPT :

generate a Stack class with push, pop, peek, and is\_empty methods using an example.





**Task Description #2 – Queue Implementation**

Task: Use AI to implement a Queue using Python lists.

Sample Input Code:

class Queue:

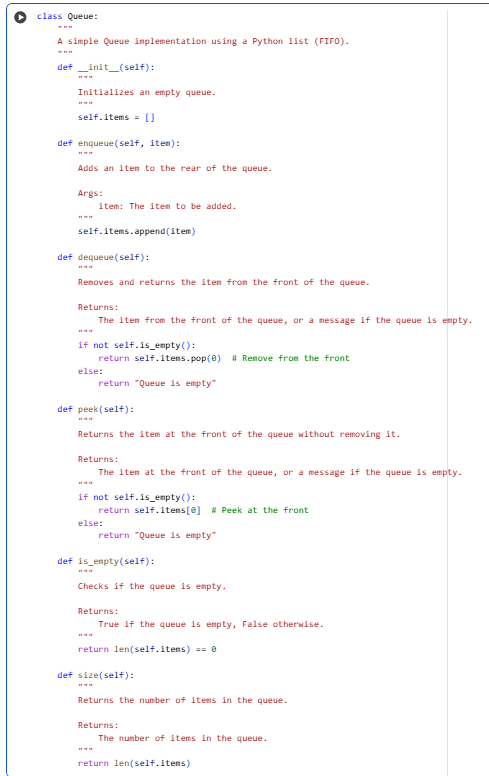
pass

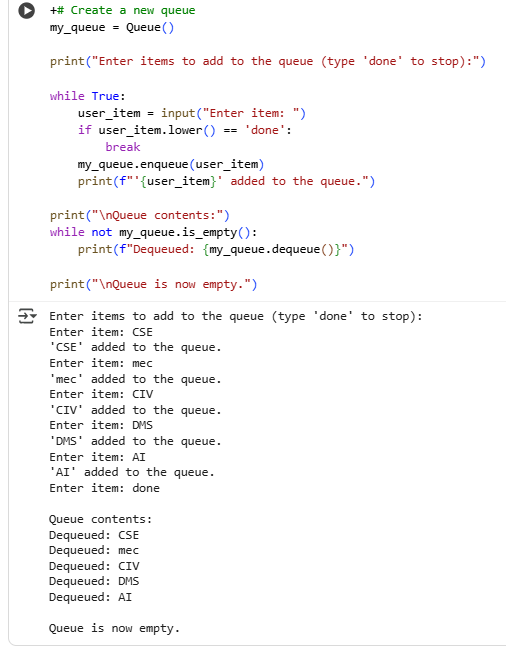
Expected Output:

* FIFO-based queue class with enqueue, dequeue, peek, and size methods.

PROMPT:

generate a FIFO-based queue using lists class with enqueue, dequeue, peek, and size methods





**Task Description #3 – Linked List**

Task: Use AI to generate a Singly Linked List with insert and display methods.

Sample Input Code:

class Node:

pass

class LinkedList:

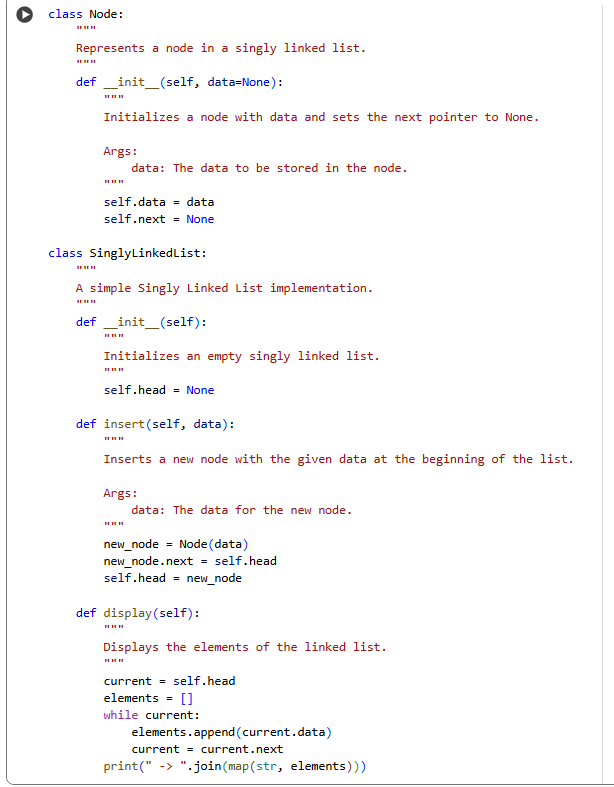
pass

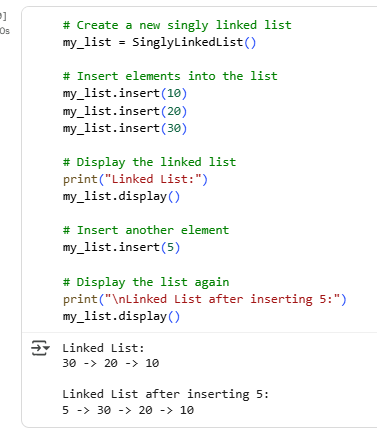
Expected Output:

* A working linked list implementation with clear method documentation.

PROMPT:

generate a Singly Linked List with insert and display methods.





**Task Description #4 – Binary Search Tree (BST)**

Task: Use AI to create a BST with insert and in-order traversal methods.

Sample Input Code:

class BST:

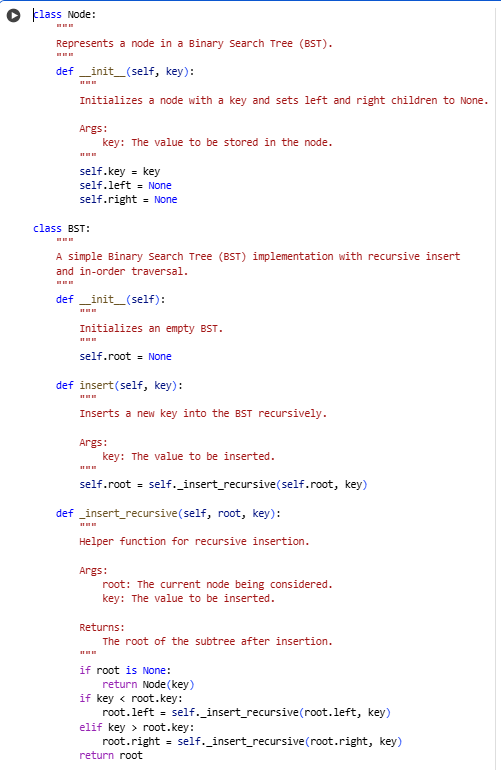
pass

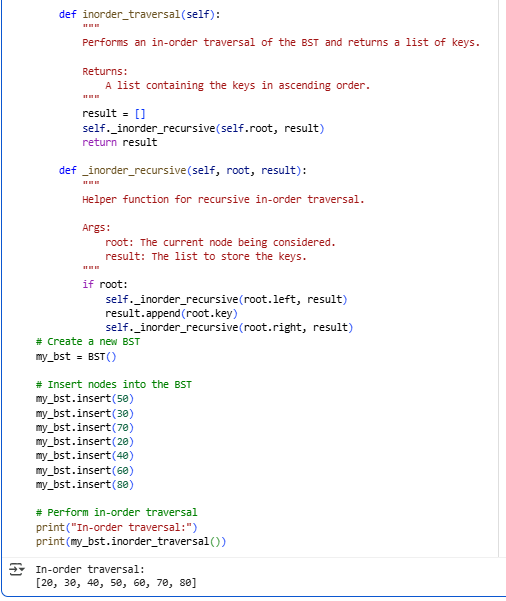
Expected Output:

* BST implementation with recursive insert and traversal methods.

PROMPT:

create a BST with recursive insert and in-order traversal methods.





**Task Description #5 – Hash Table**

Task: Use AI to implement a hash table with basic insert, search, and delete methods.

Sample Input Code:

class HashTable:

pass

Expected Output:

Collision handling using chaining, with well-commented methods.

PROMPT:

generate a hash table with basic insert, search, and delete methods.

