SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENTOF COMPUTER SCIENCE ENGINEERING	
ProgramName:M. Tech/MCA/MSC		**AssignmentType: Lab**	**AcademicYear:**2025-2026
CourseCoordinatorName		Venkataramana Veeramsetty	
CourseCode		**CourseTitle**	AI Assisted Problem Solving Using Python
Year/Sem	II/I	**Regulation**	R24
DateandDay of Assignment	Week5- Monday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	
AssignmentNumber:11.3(Presentassignmentnumber)/**24**(Totalnumberofassignments)			

Q.No.	Question	*ExpectedTime to complete*
1	**Lab 11 – Data Structures with AI: Implementing Fundamental Structures** **Lab Objectives** • To implement fundamental data structures with the assistance of AI tools. • To understand how AI suggests different implementations and optimizations. • To analyze the readability, correctness, and performance of AI-generated code. • To reinforce problem-solving skills using AI-powered coding assistance. **Learning Outcomes** After completing this lab, students will be able to: 1. Implement stack, queue, and linked list using Python with AI support. 2. Use AI tools to optimize and refactor basic data structure operations. 3. Compare multiple AI-suggested implementations for the same structure. 4. Apply AI assistance to generate test cases for verifying data structure behavior. 5. Demonstrate understanding of trade-offs in AI-generated solutions. **Task Description #1 – Stack class implementation** Task: Ask AI to implement a stack class with push(), pop(), peek() and is_empty() methods **Task Description #2 – Queue Implementation**	Week5 - Monday

	Task: Use AI to generate a Queue class with enqueue(), dequeue(), and is_empty(). **Task Description #3 – Linked List Implementation** Task: Ask AI to create a singly linked list with insert_at_end(), insert_at_beginning(), and display(). **Task Description #4 – Binary Search Tree (BST)** Task: Ask AI to generate a simple BST with insert() and inorder_traversal().	

Task Description #1 – Stack class implementation

Task: Ask AI to implement a stack class with push(), pop(), peek() and is_empty() methods

PROMPT: Write a Python program that takes user input to operate a Stack class. Implement methods: push(), pop(), peek(), and is_empty(). Let the user enter values to push and choose operations.

task 2.py | class Stack: Untitled-1 | TASK 4.py

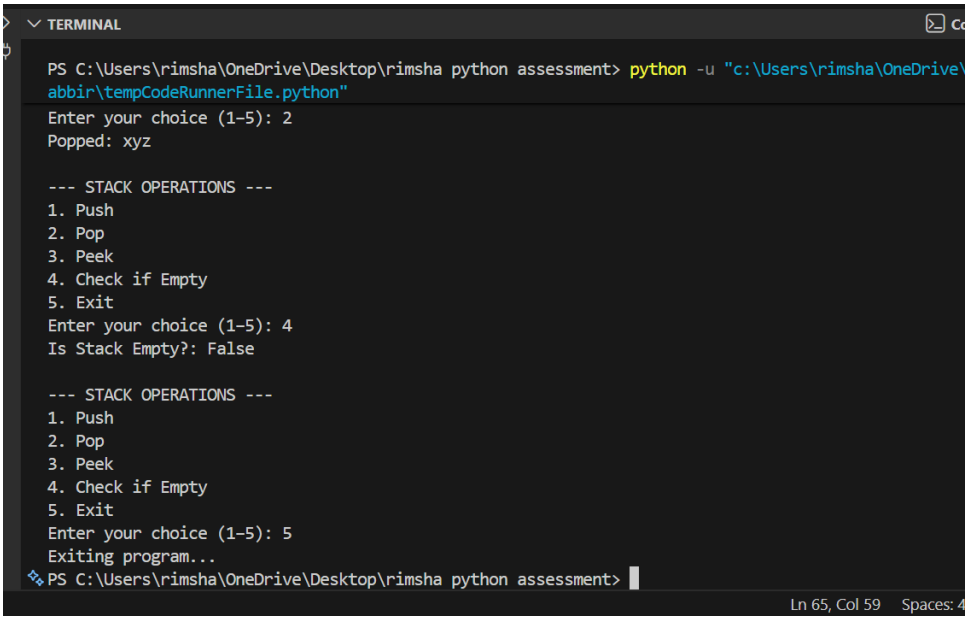
```python
class Stack:
    """
    A simple Stack implementation using Python list.
    Supports push, pop, peek, and is_empty operations.
    """

    def __init__(self):
        """Initialize an empty stack."""
        self.items = []

    def push(self, item):
        """Push an item onto the stack."""
        self.items.append(item)

    def pop(self):
        """Pop and return the top item of the stack."""
        if self.is_empty():
            return "Stack is empty. Cannot pop."
        return self.items.pop()

    def peek(self):
        """Return the top item without removing it."""
        if self.is_empty():
            return "Stack is empty. Nothing to peek."
        return self.items[-1]

    def is_empty(self):
        """Check if the stack is empty."""
        return len(self.items) == 0
```

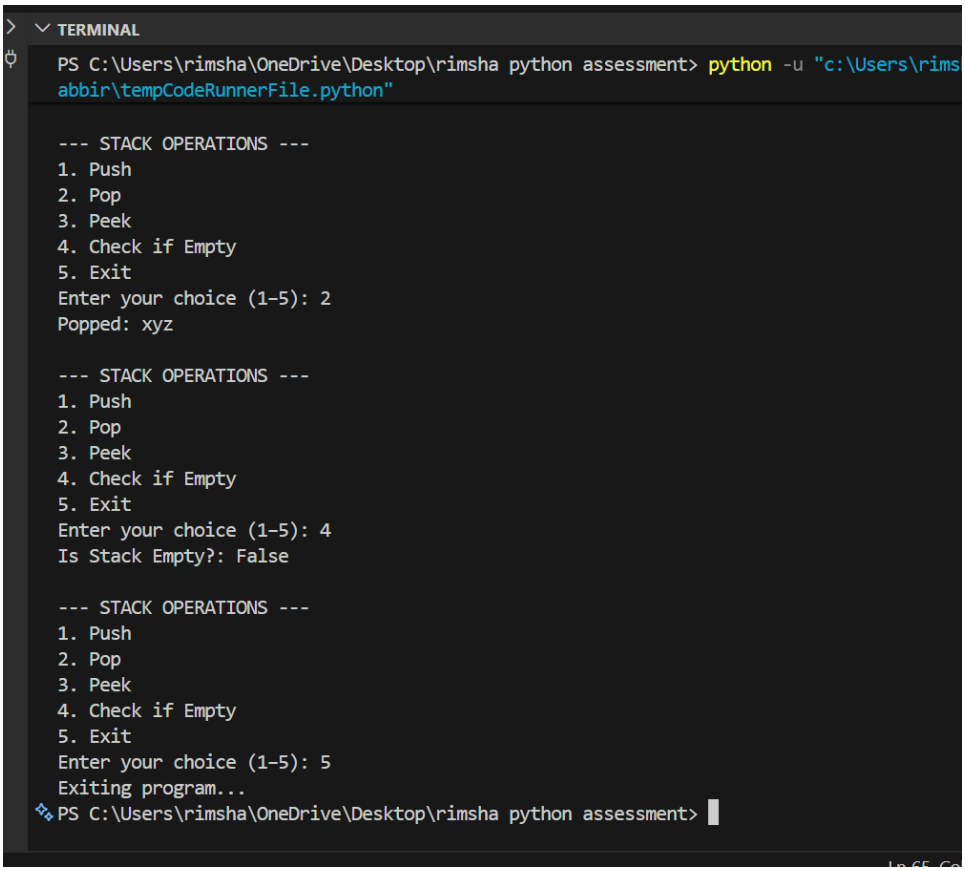
task 2.py | class Stack: Untitled-1 | TASK 4.py

```python

# -------- MENU DRIVEN USER INPUT --------

stack = Stack()

while True:
    print("\n--- STACK OPERATIONS ---")
    print("1. Push")
    print("2. Pop")
    print("3. Peek")
    print("4. Check if Empty")
    print("5. Exit")

    choice = input("Enter your choice (1-5): ")

    if choice == "1":
        item = input("Enter item to push: ")
        stack.push(item)
        print("Pushed:", item)

    elif choice == "2":
        print("Popped:", stack.pop())

    elif choice == "3":
        print("Top Element:", stack.peek())

    elif choice == "4":
        print("Is Stack Empty?:", stack.is_empty())

    elif choice == "5":
```

```python
    elif choice == "3":
        print("Top Element:", stack.peek())

    elif choice == "4":
        print("Is Stack Empty?:", stack.is_empty())

    elif choice == "5":
        print("Exiting program...")
        break

    else:
        print("Invalid choice! Please enter between 1-5.")
```

PRACTICAL OUTPUT:

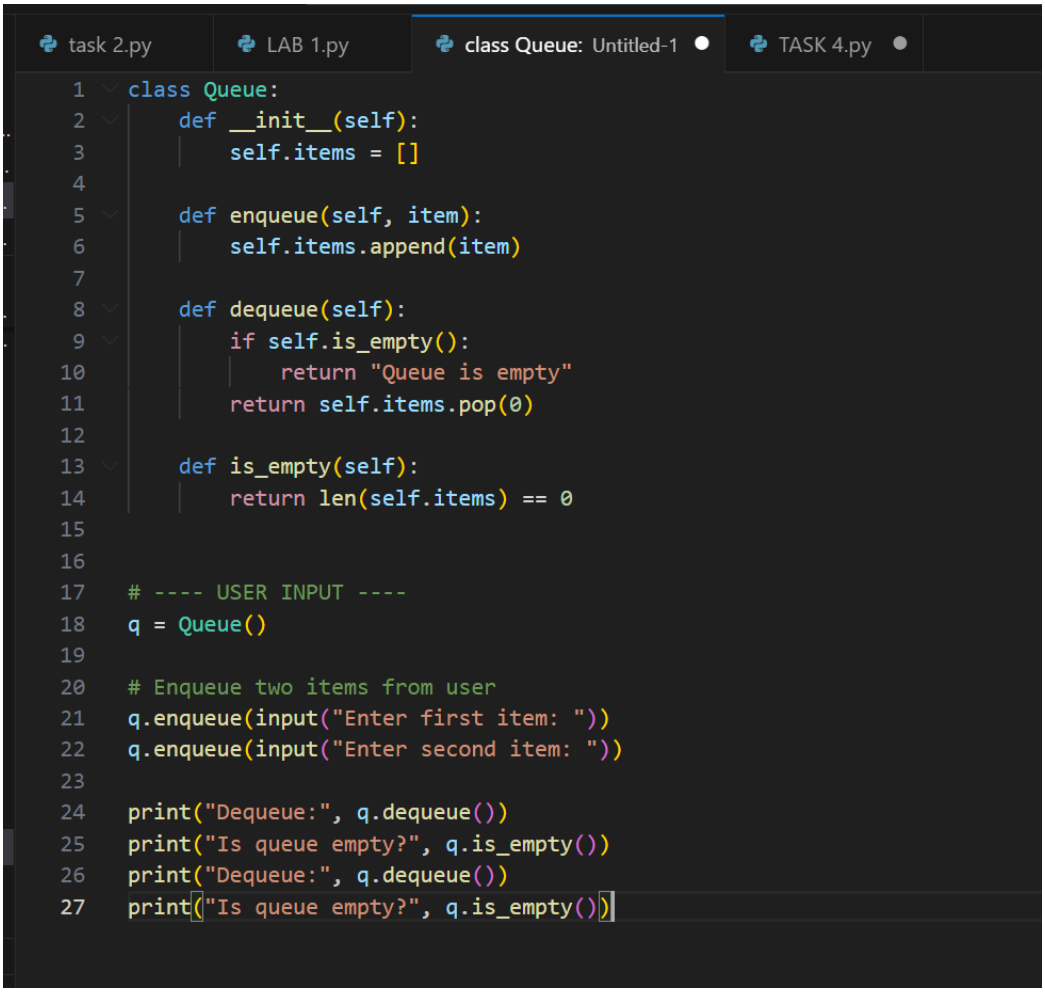




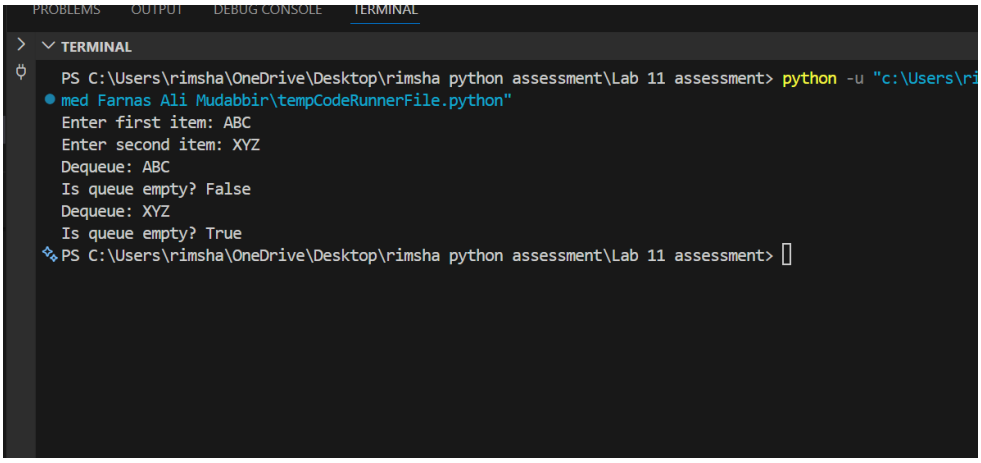
Task Description #2 – Queue Implementation

Task: Use AI to generate a Queue class with enqueue(), dequeue(), and is_empty().

PROMPT: Write a Python program that takes user input to create a Queue class with enqueue(), dequeue(), and is_empty() methods. Then let the user add and remove items from the queue.

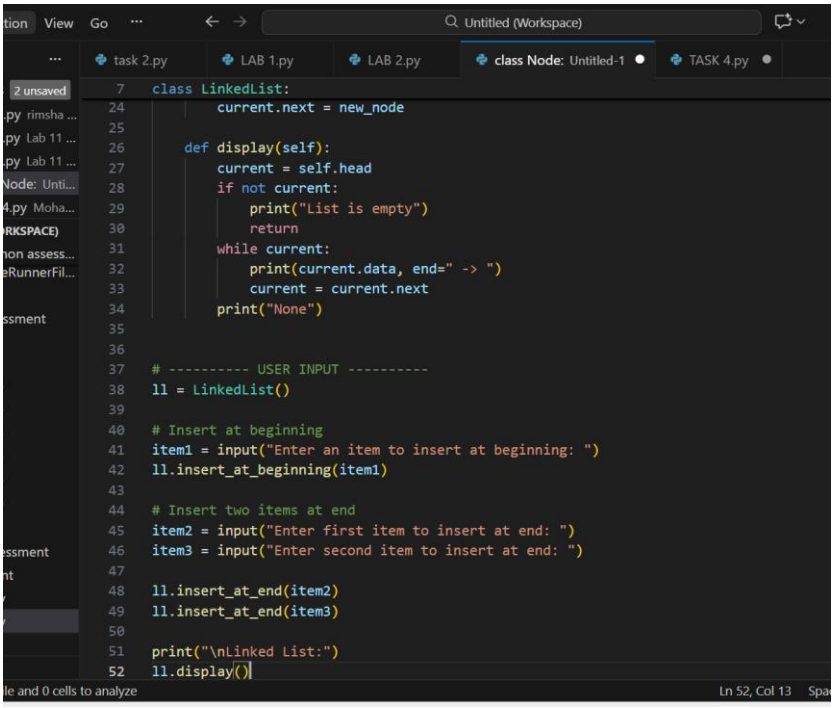


PRACTICAL OUTPUT:

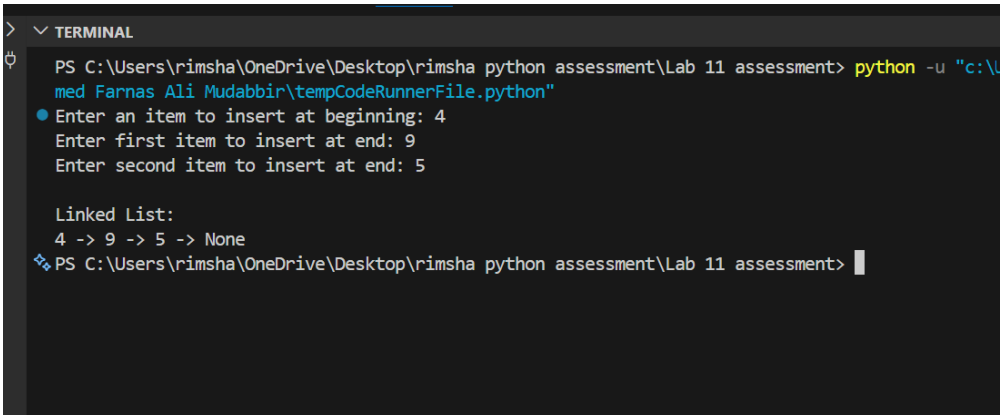


Task Description #3 – Linked List Implementation

Task: Ask AI to create a singly linked list with insert_at_end(), insert_at_beginning(), and display().
PROMPT: Create a Python program for a singly linked list with insert_at_beginning(), insert_at_end(), and display() methods. Take user input to insert and view the list.



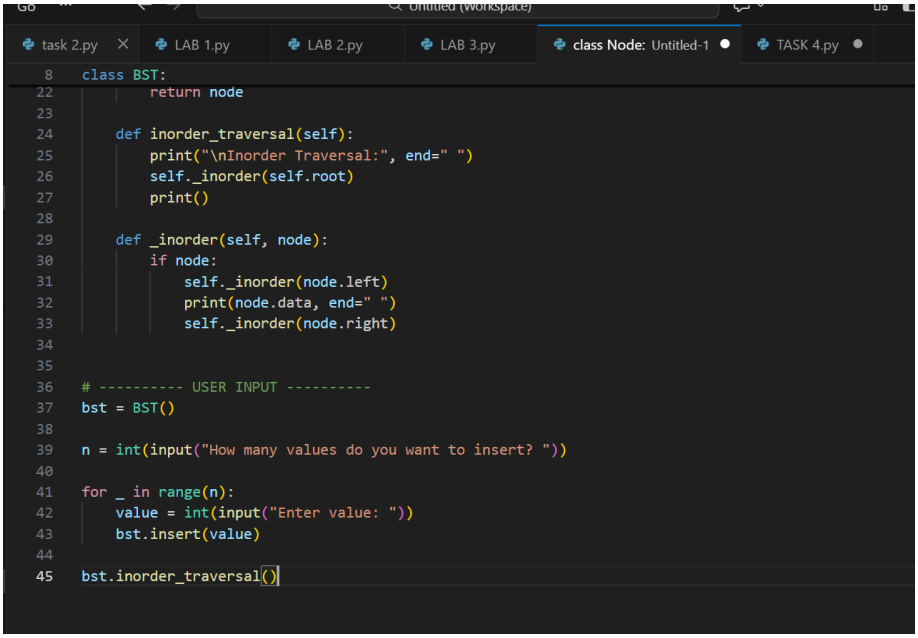
PRACTICAL OUTPUT:



Task Description #4 – Binary Search Tree (BST)

Task: Ask AI to generate a simple BST with insert() and inorder_traversal().

PROMPT: Write a Python program to implement a Binary Search Tree with insert() and inorder_traversal(), and allow user input to add nodes.



PRACTICAL OUTPUT: