

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF 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	<p>Lab 11 – Data Structures with AI: Implementing Fundamental Structures</p> <p>Lab Objectives</p> <ul style="list-style-type: none"> 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. <p>Learning Outcomes</p> <p>After completing this lab, students will be able to:</p> <ol style="list-style-type: none"> Implement stack, queue, and linked list using Python with AI support. Use AI tools to optimize and refactor basic data structure operations. Compare multiple AI-suggested implementations for the same structure. Apply AI assistance to generate test cases for verifying data structure behavior. Demonstrate understanding of trade-offs in AI-generated solutions. <p>Task Description #1 – Stack class implementation</p> <p>Task: Ask AI to implement a stack class with push(), pop(), peek() and is_empty() methods</p> <p>Task Description #2 – Queue Implementation</p>		Week5 - Monday

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

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
1  class Stack:
2      """
3      A simple Stack implementation using Python list.
4      Supports push, pop, peek, and is_empty operations.
5      """
6
7      def __init__(self):
8          """Initialize an empty stack."""
9          self.items = []
10
11      def push(self, item):
12          """Push an item onto the stack."""
13          self.items.append(item)
14
15      def pop(self):
16          """Pop and return the top item of the stack."""
17          if self.is_empty():
18              return "Stack is empty. Cannot pop."
19          return self.items.pop()
20
21      def peek(self):
22          """Return the top item without removing it."""
23          if self.is_empty():
24              return "Stack is empty. Nothing to peek."
25          return self.items[-1]
26
27      def is_empty(self):
28          """Check if the stack is empty."""
29          return len(self.items) == 0
30
```

```
task 2.py  class Stack: Untitled-1  TASK 4.py
31
32 # ----- MENU DRIVEN USER INPUT -----
33
34 stack = Stack()
35
36 while True:
37     print("\n--- STACK OPERATIONS ---")
38     print("1. Push")
39     print("2. Pop")
40     print("3. Peek")
41     print("4. Check if Empty")
42     print("5. Exit")
43
44     choice = input("Enter your choice (1-5): ")
45
46     if choice == "1":
47         item = input("Enter item to push: ")
48         stack.push(item)
49         print("Pushed:", item)
50
51     elif choice == "2":
52         print("Popped:", stack.pop())
53
54     elif choice == "3":
55         print("Top Element:", stack.peak())
56
57     elif choice == "4":
58         print("Is Stack Empty?", stack.is_empty())
59
60     elif choice == "5":
61
62
63
64
65     elif choice == "3":
66         print("Top Element:", stack.peak())
67
68     elif choice == "4":
69         print("Is Stack Empty?", stack.is_empty())
70
71     elif choice == "5":
72         print("Exiting program...")
73         break
74
75     else:
76         print("Invalid choice! Please enter between 1-5.")
```

PRACTICAL OUTPUT:

```
> ▼ TERMINAL
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\rimsha\OneDrive\
abbir\tempCodeRunnerFile.python"
Enter your choice (1-5): 2
Popped: xyz

--- STACK OPERATIONS ---
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Exit
Enter your choice (1-5): 4
Is Stack Empty?: False

--- STACK OPERATIONS ---
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Exit
Enter your choice (1-5): 5
Exiting program...
❖ PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> |
```

Ln 65, Col 59 Spaces: 4

```
> ▼ TERMINAL
❖ PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> python -u "c:\Users\rims
abbir\tempCodeRunnerFile.python"

--- STACK OPERATIONS ---
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Exit
Enter your choice (1-5): 2
Popped: xyz

--- STACK OPERATIONS ---
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Exit
Enter your choice (1-5): 4
Is Stack Empty?: False

--- STACK OPERATIONS ---
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Exit
Enter your choice (1-5): 5
Exiting program...
❖ PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment> |
```

Ln 65, Co

Task Description #2 – Queue Implementation

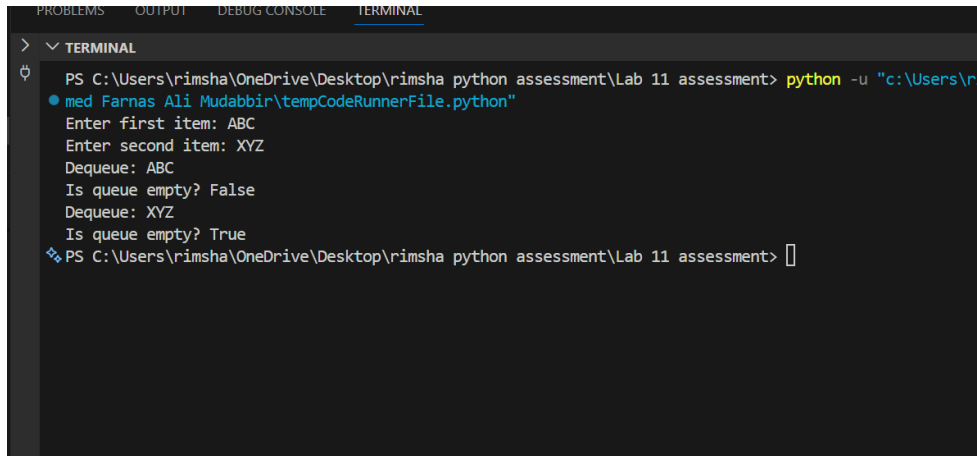
Task: Use AI to generate a Queue class with enqueue(), dequeue(), and is_empty().

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.

```
task 2.py  LAB 1.py  class Queue: Untitled-1  TASK 4.py

1  class Queue:
2      def __init__(self):
3          self.items = []
4
5      def enqueue(self, item):
6          self.items.append(item)
7
8      def dequeue(self):
9          if self.is_empty():
10             return "Queue is empty"
11             return self.items.pop(0)
12
13     def is_empty(self):
14         return len(self.items) == 0
15
16
17  # ---- USER INPUT ----
18  q = Queue()
19
20  # Enqueue two items from user
21  q.enqueue(input("Enter first item: "))
22  q.enqueue(input("Enter second item: "))
23
24  print("Dequeue:", q.dequeue())
25  print("Is queue empty?", q.is_empty())
26  print("Dequeue:", q.dequeue())
27  print("Is queue empty?", q.is_empty())
```

PRACTICAL OUTPUT:

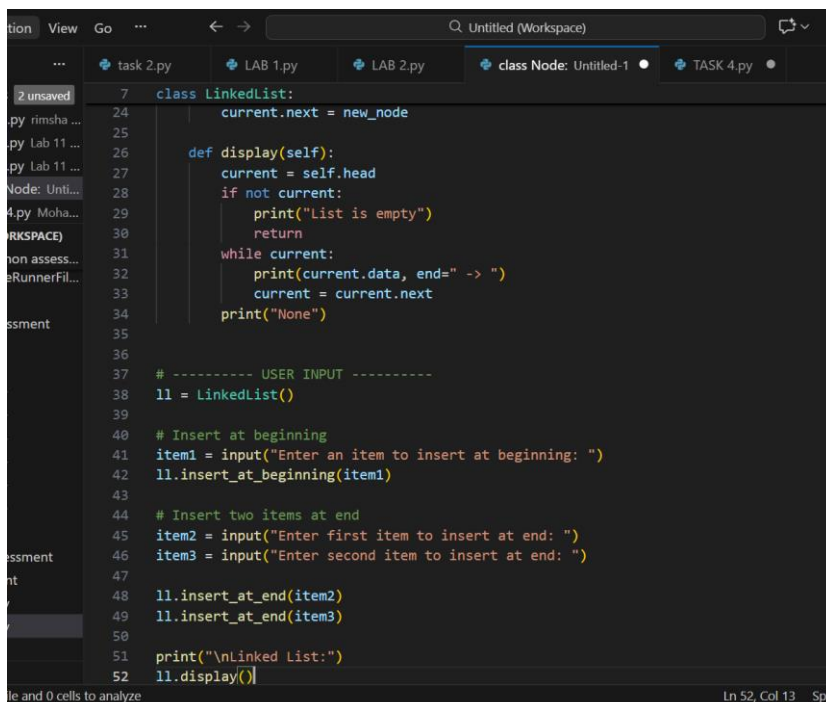


```
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\Lab 11 assessment> python -u "c:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\Lab 11 assessment\tempCodeRunnerFile.python"
Enter first item: ABC
Enter second item: XYZ
Dequeue: ABC
Is queue empty? False
Dequeue: XYZ
Is queue empty? True
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\Lab 11 assessment>
```

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.



```
class LinkedList:
    def __init__(self):
        self.head = None
        self.current.next = new_node

    def display(self):
        current = self.head
        if not current:
            print("List is empty")
            return
        while current:
            print(current.data, end=" -> ")
            current = current.next
        print("None")

# ----- USER INPUT -----
ll = LinkedList()

# Insert at beginning
item1 = input("Enter an item to insert at beginning: ")
ll.insert_at_beginning(item1)

# Insert two items at end
item2 = input("Enter first item to insert at end: ")
item3 = input("Enter second item to insert at end: ")
ll.insert_at_end(item2)
ll.insert_at_end(item3)

print("\nLinked List:")
ll.display()
```

PRACTICAL OUTPUT:

```
> TERMINAL
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\Lab 11 assessment> python -u "c:\u
med Farnas Ali Mudabbir\tempCodeRunnerFile.python"
● Enter an item to insert at beginning: 4
Enter first item to insert at end: 9
Enter second item to insert at end: 5

Linked List:
4 -> 9 -> 5 -> None
❖ PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\Lab 11 assessment> |
```

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.

```
task 2.py x LAB 1.py LAB 2.py LAB 3.py class Node: Untitled-1 TASK 4.py
8 class BST:
22     return node
23
24     def inorder_traversal(self):
25         print("\nInorder Traversal:", end=" ")
26         self._inorder(self.root)
27         print()
28
29     def _inorder(self, node):
30         if node:
31             self._inorder(node.left)
32             print(node.data, end=" ")
33             self._inorder(node.right)
34
35
36 # ----- USER INPUT -----
37 bst = BST()
38
39 n = int(input("How many values do you want to insert? "))
40
41 for _ in range(n):
42     value = int(input("Enter value: "))
43     bst.insert(value)
44
45 bst.inorder_traversal()
```


PRACTICAL OUTPUT:

```
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\Lab 11 assessment> python -u "c:\Users\rimsha\OneDrive\
med Farnas Ali Mudabbir\tempCodeRunnerFile.python"
How many values do you want to insert? 3
Enter value: 22
Enter value: 65
Enter value: 7

Inorder Traversal: 7 22 65
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\Lab 11 assessment>
```