

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING		
Program Name: M. Tech/MCA/MSC		Assignment Type: Lab		
Course Coordinator Name		Venkataramana Veeramsetty		
Course Code		Course Title	AI Assisted Problem Solving Using Python	
Year/Sem	II/I	Regulation	R24	
Date and Day of Assignment	Week 5- Monday	Time(s)		
Duration	2 Hours	Applicable to Batches		
Assignment Number: 11.3(Present assignment number)/ 24 (Total number of assignments)				

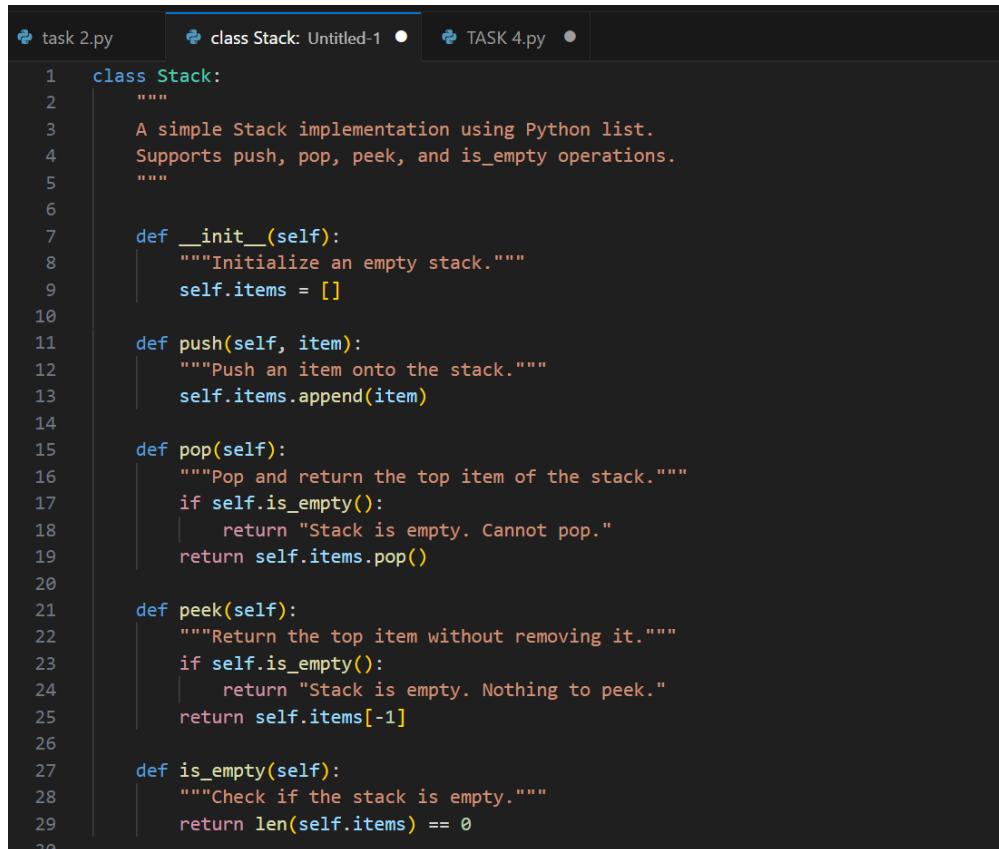
Q.No.	Question	Expected Time to complete
1	<p>Lab 11 – Data Structures with AI: Implementing Fundamental Structures</p> <p>Lab Objectives</p> <ul style="list-style-type: none"> To implement fundamental data structures with the assistance of AI tools. To understand how AI suggests different implementations 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>	Week 5 - Monday

	<p>Task: Use AI to generate a Queue class with enqueue(), dequeue(), and is_empty().</p> <p>Task Description #3 – Linked List Implementation</p> <p>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)</p> <p>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
```

```
# ----- 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":
        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:

```
> ▾ 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\rimsha\OneDrive\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

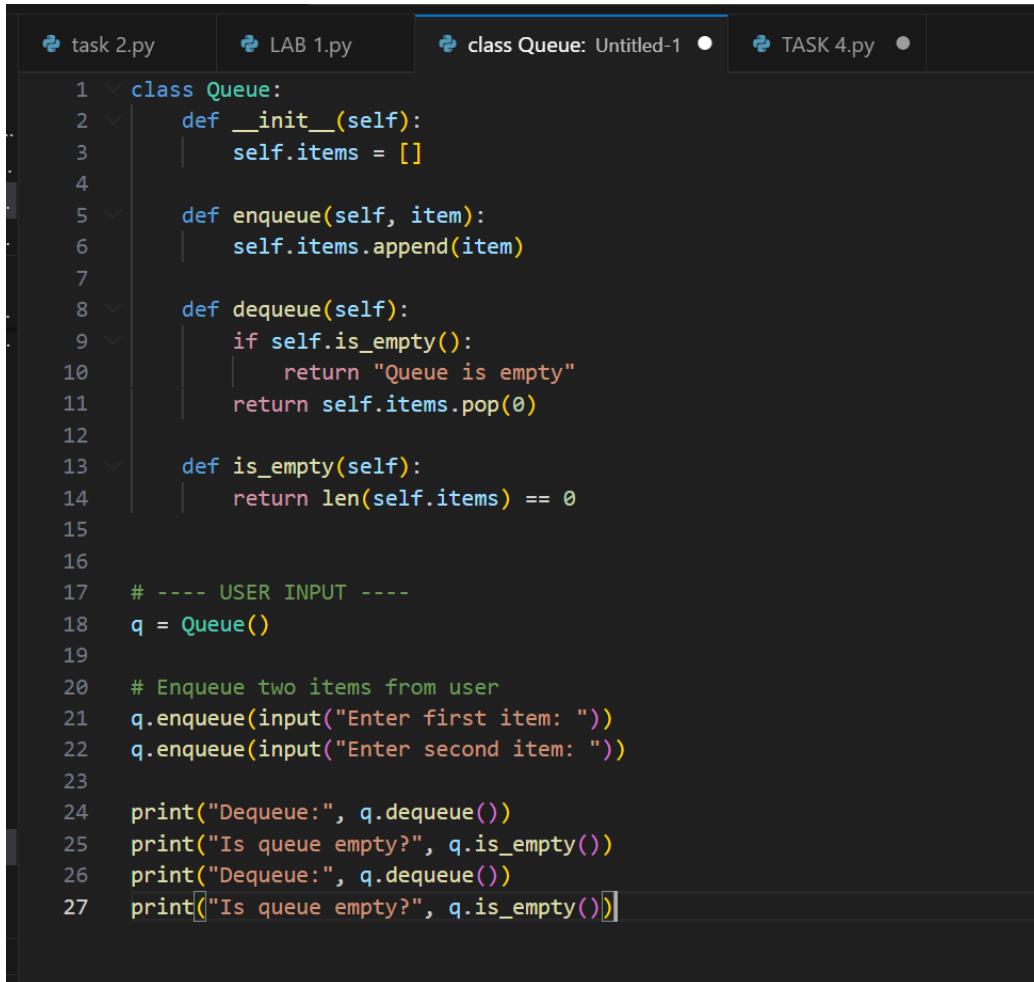
--- STACK OPERATIONS ---
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Exit
Enter your choice (1-5): 4
Is Stack Empty?: False

--- 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, Col 59   Spaces: 4
```

Task Description #2 – Queue Implementation

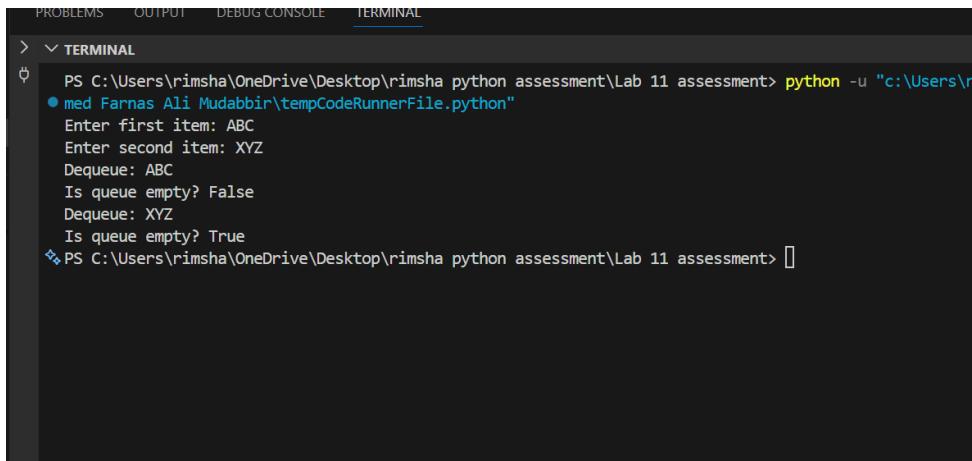
Task: Use AI to generate a Queue class with enqueue(), dequeue(), and is_empty().

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:

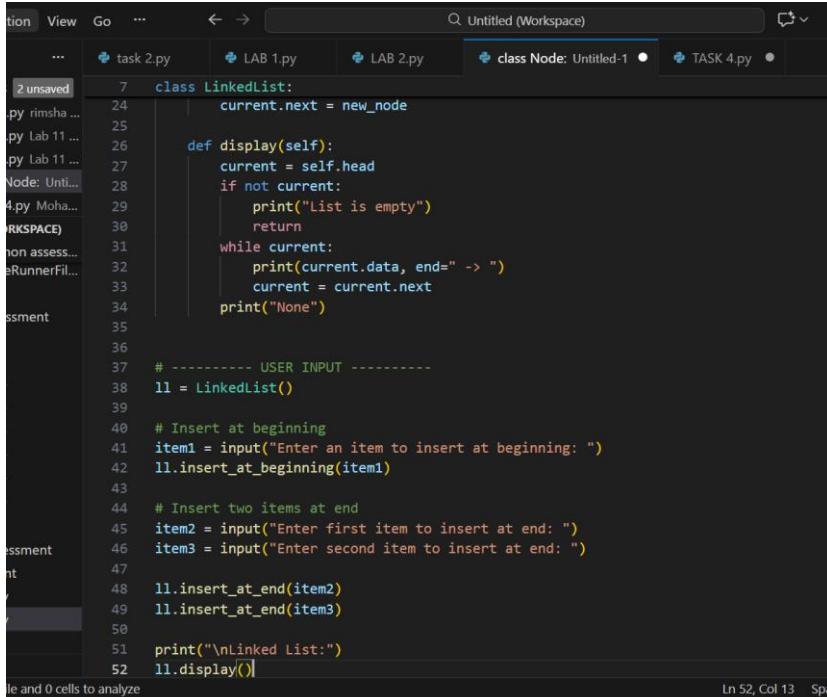


```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
> < TERMINAL
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\Lab 11 assessment> python -u "c:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\Lab 11 assessment\tempCodeRunnerFile.python"
● med Farnas Ali Mudabbir\tempCodeRunnerFile.python"
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

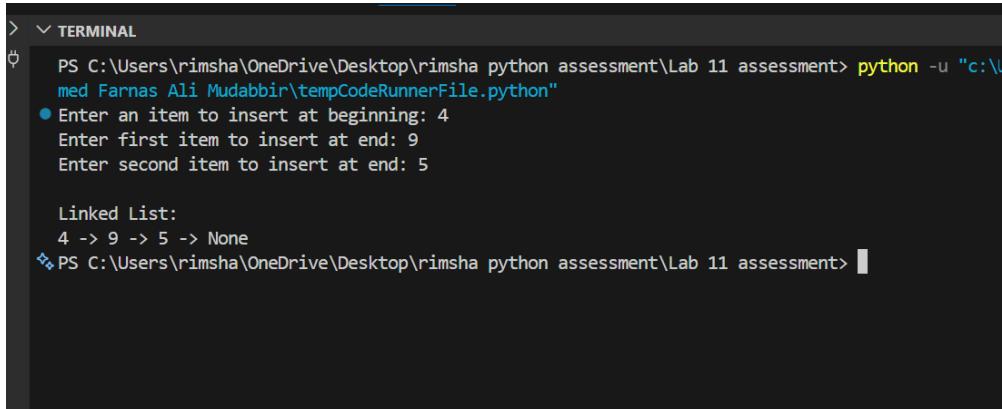
Task: Ask AI to create a singly linked list with insert_at_end(), insert_at_beginning(), and display().

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.



```
tion  View  Go  ...
...  ← →  Q Untitled (Workspace)
...  task 2.py  LAB 1.py  LAB 2.py  class Node: Untitled-1  ●  TASK 4.py  ●
2 unsaved
py rimsha ...
py Lab 11 ...
py Lab 11 ...
Node: Unt...
4.py Mohan...
(RKSPACE)
non assess...
eRunnerFil...
ssment
7   class LinkedList:
8       current.next = new_node
9
10      def display(self):
11          current = self.head
12          if not current:
13              print("List is empty")
14              return
15          while current:
16              print(current.data, end=" -> ")
17              current = current.next
18          print("None")
19
20
21      # ----- USER INPUT -----
22      ll = LinkedList()
23
24      # Insert at beginning
25      item1 = input("Enter an item to insert at beginning: ")
26      ll.insert_at_beginning(item1)
27
28      # Insert two items at end
29      item2 = input("Enter first item to insert at end: ")
30      item3 = input("Enter second item to insert at end: ")
31
32      ll.insert_at_end(item2)
33      ll.insert_at_end(item3)
34
35      print("\nLinked List:")
36      ll.display()
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
```

PRACTICAL OUTPUT:



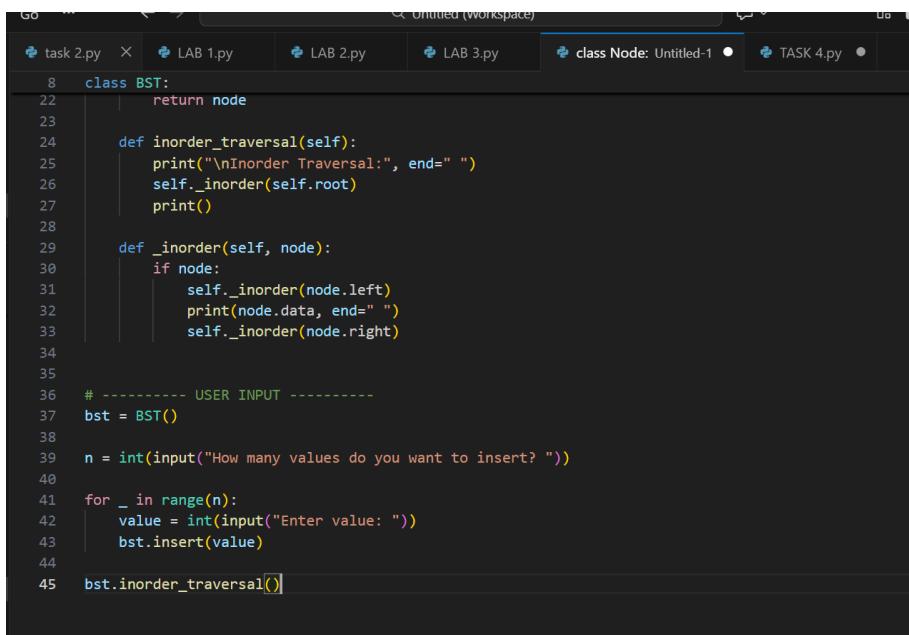
```
> ▾ TERMINAL
PS C:\Users\rimsha\OneDrive\Desktop\rimsha python assessment\Lab 11 assessment> python -u "c:\med Farnas Ali Mudabbir\tempCodeRunnerFile.py"
● Enter an item to insert at beginning: 4
Enter first item to insert at end: 9
Enter second item to insert at end: 5

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.



```
Go ⌂ Untitled (workspace)
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: