



# **DATA STRUCTURES**

---

## **LAB FILE**

**SUBMITTED BY - VANSR RANA**

**ROLL NO - 2401420011**

**PROGRAM - B.Tech CSE (Data Science)**

**SEMESTER - III**

**SUBMITTED TO -**

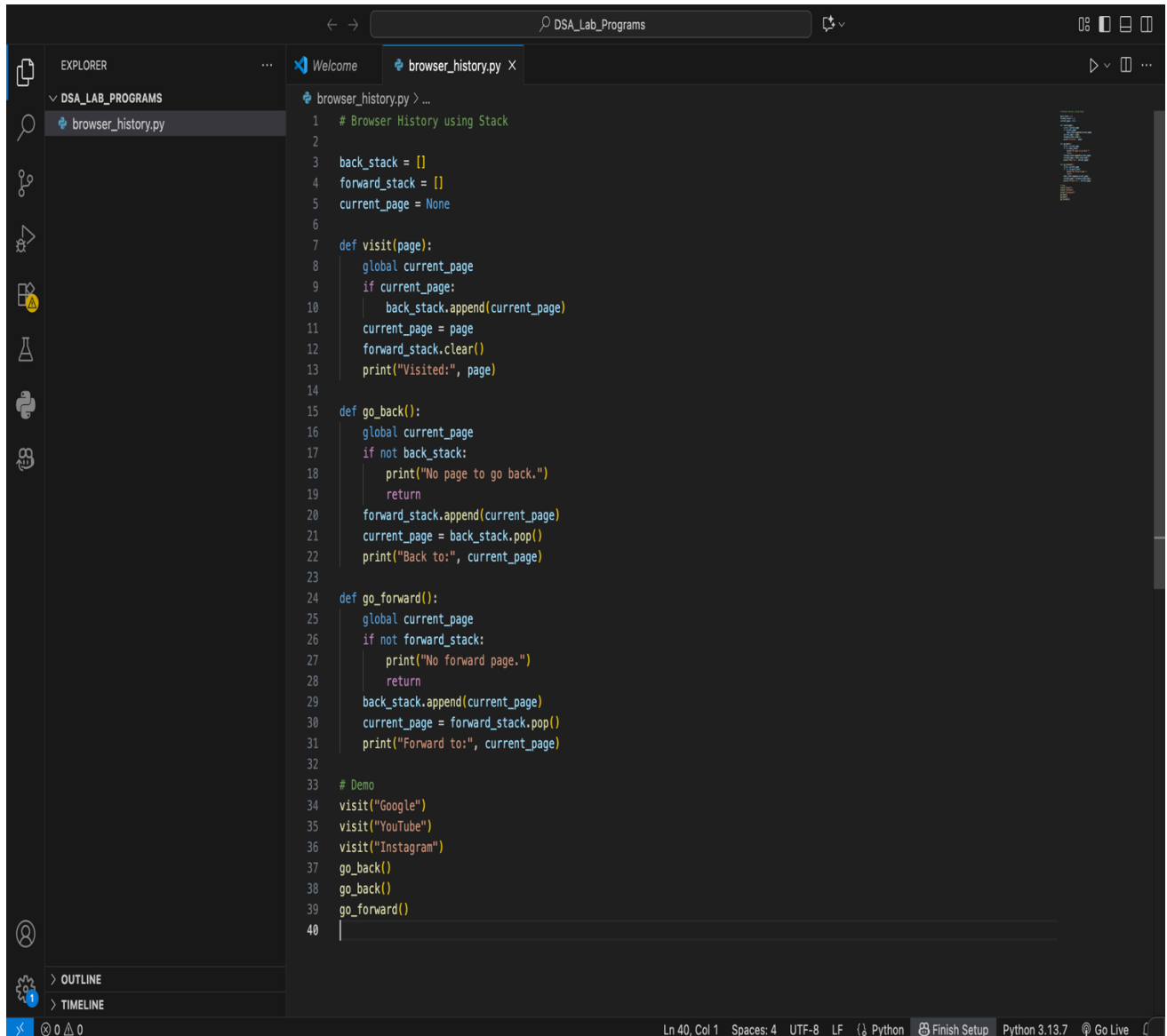
## INDEX

<b>S. NO.</b>	<b>Experiment Title</b>	<b>Page No.</b>
<b>1.</b>	Browser History Navigation System (Using Stack Concept)	
<b>2.</b>	Ticketing System Using Queue (Linear Queue Implementation)	
<b>3.</b>	Singly Linked List Operations (Insert, Delete, Search, Display)	
<b>4.</b>	Circular Singly Linked List (Insert, Search, Delete, Display)	
<b>5.</b>	Reverse a String Using Stack	
<b>6.</b>	Check Balanced Parentheses Using Stack	
<b>7.</b>	Lab Project: Inventory Stock Management System	

# EXPERIMENT 1

## Browser History Navigation System (Using Stack Concept)

CODE:-



The screenshot shows a code editor with a dark theme. The Explorer panel on the left shows a project named 'DSA\_LAB\_PROGRAMS' containing a file 'browser\_history.py'. The main editor area displays the following Python code:

```
1 # Browser History using Stack
2
3 back_stack = []
4 forward_stack = []
5 current_page = None
6
7 def visit(page):
8     global current_page
9     if current_page:
10         back_stack.append(current_page)
11     current_page = page
12     forward_stack.clear()
13     print("Visited:", page)
14
15 def go_back():
16     global current_page
17     if not back_stack:
18         print("No page to go back.")
19         return
20     forward_stack.append(current_page)
21     current_page = back_stack.pop()
22     print("Back to:", current_page)
23
24 def go_forward():
25     global current_page
26     if not forward_stack:
27         print("No forward page.")
28         return
29     back_stack.append(current_page)
30     current_page = forward_stack.pop()
31     print("Forward to:", current_page)
32
33 # Demo
34 visit("Google")
35 visit("YouTube")
36 visit("Instagram")
37 go_back()
38 go_back()
39 go_forward()
40
```

The status bar at the bottom indicates the cursor is at line 40, column 1, with 4 spaces, UTF-8 encoding, and LF line endings. It also shows the Python interpreter is set to Python 3.13.7.

OUTPUT:-

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

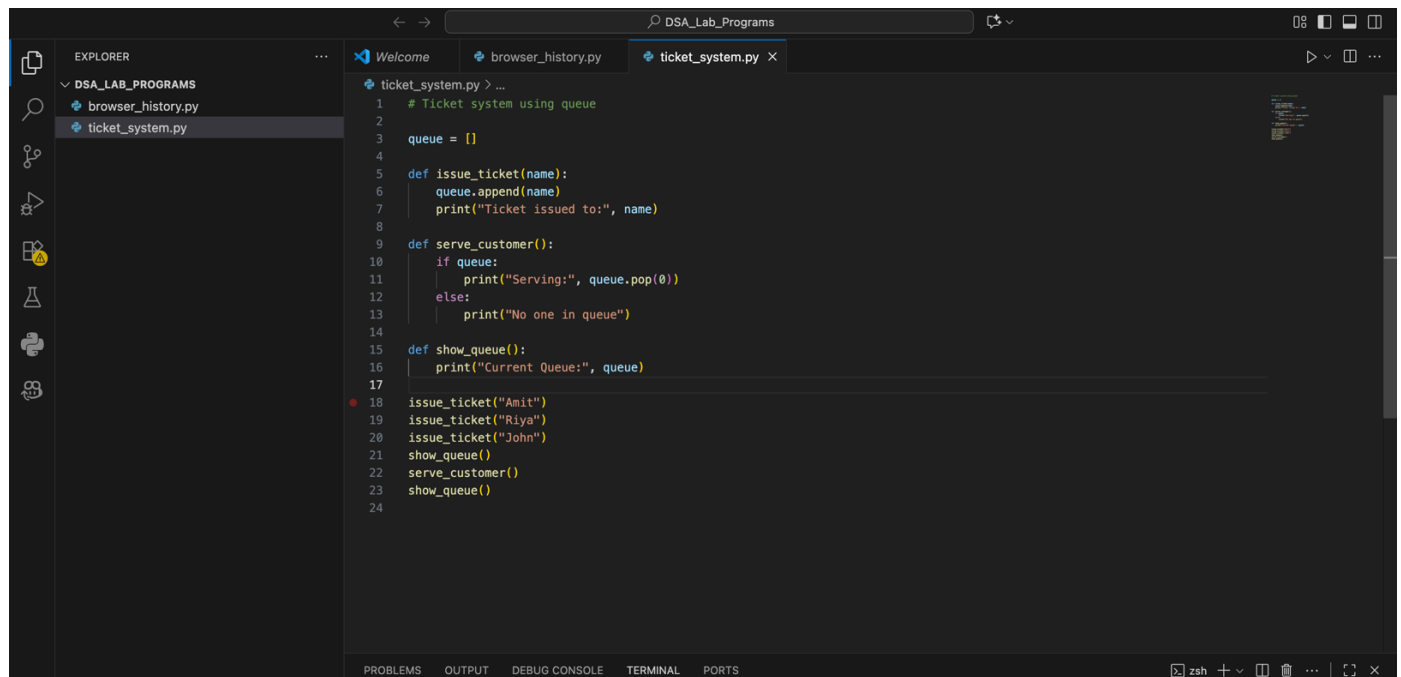
(base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs % python3 filename.py
ms/filename.py': [Errno 2] No such file or directory
● (base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs % python3 browser_history.py

Visited: Google
Visited: YouTube
Visited: Instagram
Back to: YouTube
Back to: Google
Forward to: YouTube
○ (base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs %
```

## EXPERIMENT 2

### Ticketing System Using Queue (Linear Queue Implementation)

CODE:-



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder named 'DSA\_LAB\_PROGRAMS' containing two files: 'browser\_history.py' and 'ticket\_system.py'. The code editor is open to 'ticket\_system.py' and displays the following Python code:

```
1 # Ticket system using queue
2
3 queue = []
4
5 def issue_ticket(name):
6     queue.append(name)
7     print("Ticket issued to:", name)
8
9 def serve_customer():
10     if queue:
11         print("Serving:", queue.pop(0))
12     else:
13         print("No one in queue")
14
15 def show_queue():
16     print("Current Queue:", queue)
17
18 issue_ticket("Amit")
19 issue_ticket("Riya")
20 issue_ticket("John")
21 show_queue()
22 serve_customer()
23 show_queue()
24
```

OUTPUT:-

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● (base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs % python3 ticket_system.py

Ticket issued to: Amit
Ticket issued to: Riya
Ticket issued to: John
Current Queue: ['Amit', 'Riya', 'John']
Serving: Amit
Current Queue: ['Riya', 'John']
○ (base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs % █
```

## EXPERIMENT 3

**Ticketing System Using Queue (Linear Queue Implementation)**

CODE:-

```
1 class Node:
2     def __init__(self, data):
3         self.data = data
4         self.next = None
5
6 class List:
7     def __init__(self):
8         self.head = None
9
10    def insert_first(self, data):
11        n = Node(data)
12        n.next = self.head
13        self.head = n
14
15    def delete_value(self, val):
16        temp = self.head
17        if temp and temp.data == val:
18            self.head = temp.next
19            return
20
21        prev = None
22        while temp and temp.data != val:
23            prev = temp
24            temp = temp.next
25        if temp:
26            prev.next = temp.next
27
28    def search(self, val):
29        t = self.head
30        while t:
31            if t.data == val:
32                return True
33            t = t.next
34        return False
35
36    def display(self):
37        t = self.head
38        while t:
39            print(t.data, end=" -> ")
40            t = t.next
41        print("None")
42
43 lst = List()
44 lst.insert_first(5)
```

```
6 class List:
7     def __init__(self):
8         self.head = None
9
10    def insert_first(self, data):
11        n = Node(data)
12        n.next = self.head
13        self.head = n
14
15    def delete_value(self, val):
16        temp = self.head
17        if temp and temp.data == val:
18            self.head = temp.next
19            return
20
21        prev = None
22        while temp and temp.data != val:
23            prev = temp
24            temp = temp.next
25        if temp:
26            prev.next = temp.next
27
28    def search(self, val):
29        t = self.head
30        while t:
31            if t.data == val:
32                return True
33            t = t.next
34        return False
35
36    def display(self):
37        t = self.head
38        while t:
39            print(t.data, end=" -> ")
40            t = t.next
41        print("None")
42
43 lst = List()
44 lst.insert_first(5)
45 lst.insert_first(15)
46 lst.insert_first(25)
47 lst.display()
48 lst.delete_value(15)
49 lst.display()
50 print("Search 25:", lst.search(25))
51
```

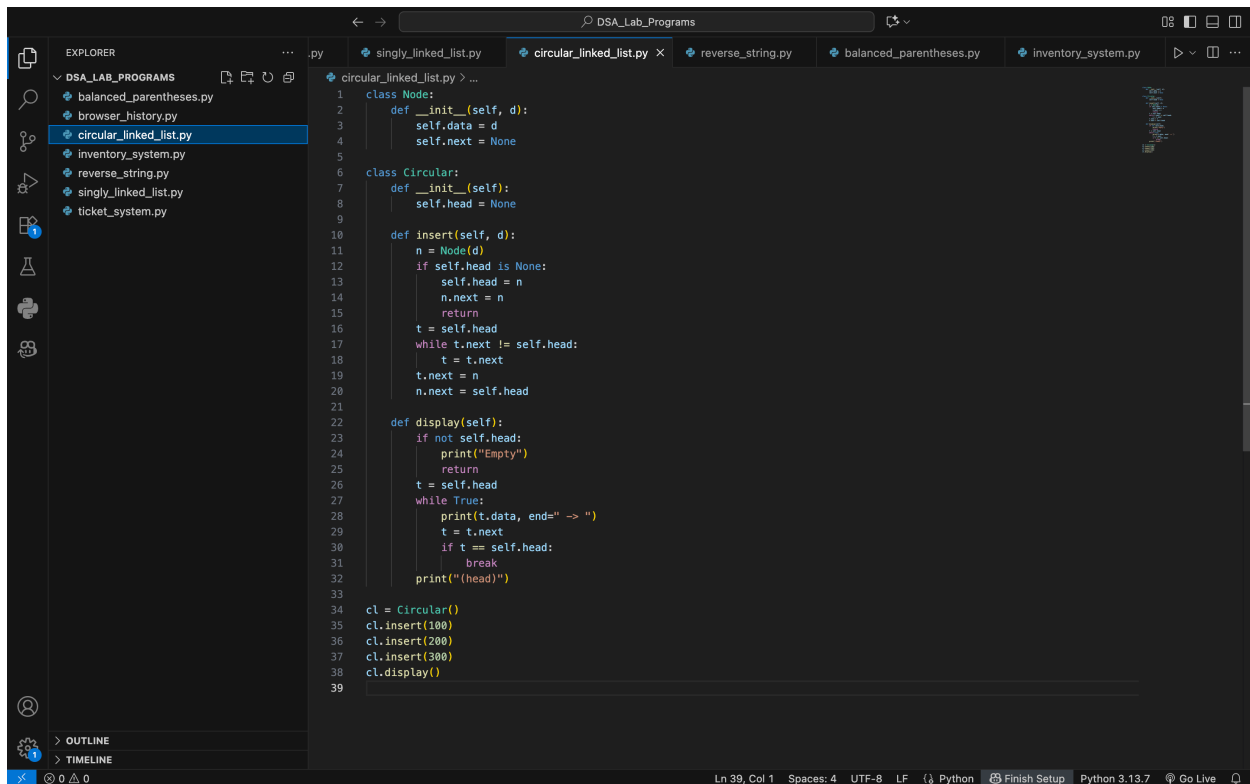
## OUTPUT:-

```
(base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs % python3 singly_linked_list.py
25 -> 15 -> 5 -> None
25 -> 5 -> None
Search 25: True
(base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs %
```

## **EXPERIMENT 4**

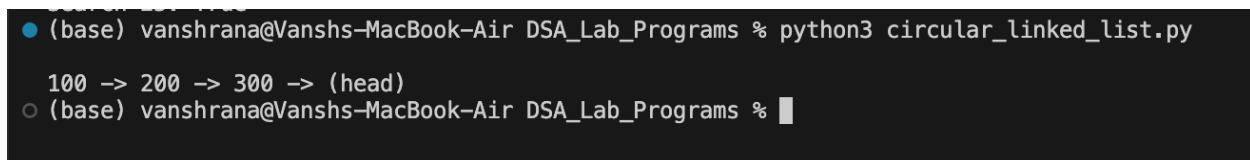
**Circular Singly Linked List (Insert, Search, Delete, Display)**

CODE:-



```
1 class Node:
2     def __init__(self, d):
3         self.data = d
4         self.next = None
5
6 class Circular:
7     def __init__(self):
8         self.head = None
9
10    def insert(self, d):
11        n = Node(d)
12        if self.head is None:
13            self.head = n
14            n.next = n
15            return
16        t = self.head
17        while t.next != self.head:
18            t = t.next
19        t.next = n
20        n.next = self.head
21
22    def display(self):
23        if not self.head:
24            print("Empty")
25            return
26        t = self.head
27        while True:
28            print(t.data, end=" -> ")
29            t = t.next
30            if t == self.head:
31                break
32        print("(head)")
33
34 cl = Circular()
35 cl.insert(100)
36 cl.insert(200)
37 cl.insert(300)
38 cl.display()
39
```

OUTPUT:-



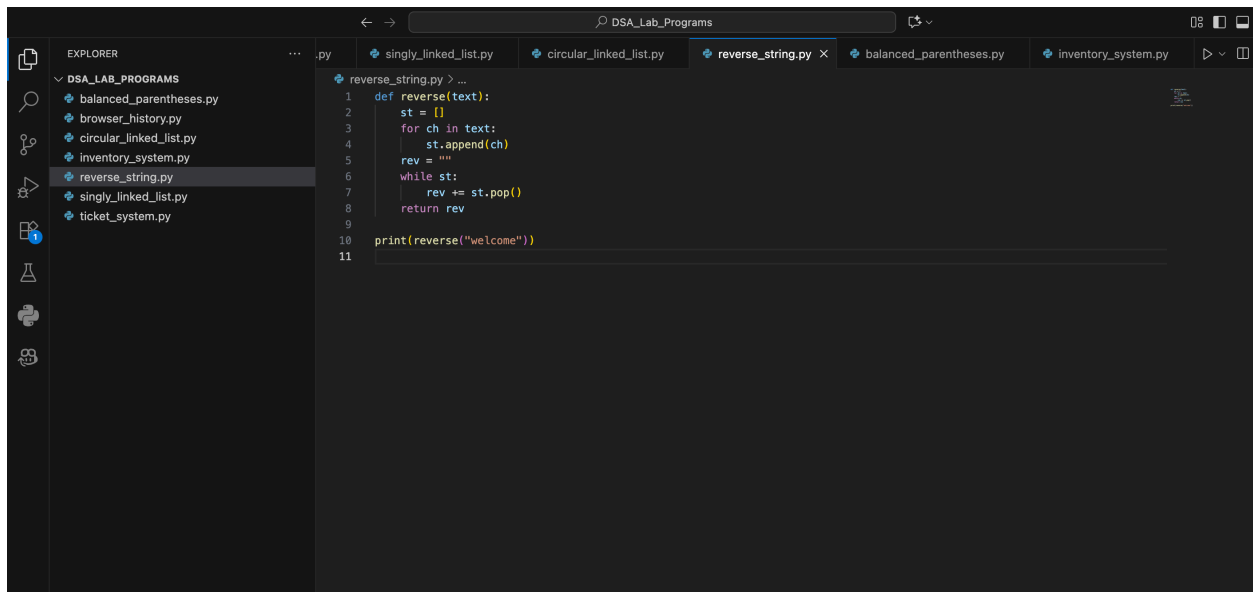
```
• (base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs % python3 circular_linked_list.py
100 -> 200 -> 300 -> (head)
○ (base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs %
```

## EXPERIMENT 5

### Reverse a String Using Stack

CODE:-

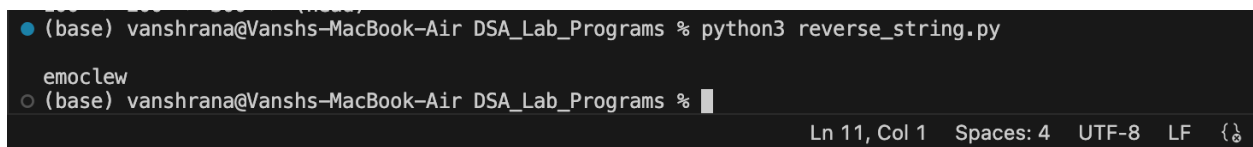




The screenshot shows a code editor with a dark theme. The Explorer panel on the left lists files under 'DSA\_LAB\_PROGRAMS', including 'balanced\_parentheses.py', 'browser\_history.py', 'circular\_linked\_list.py', 'inventory\_system.py', 'reverse\_string.py' (selected), 'singly\_linked\_list.py', and 'ticket\_system.py'. The main editor displays the code for 'reverse\_string.py':

```
1 def reverse(text):  
2     st = []  
3     for ch in text:  
4         st.append(ch)  
5     rev = ""  
6     while st:  
7         rev += st.pop()  
8     return rev  
9  
10 print(reverse("welcome"))  
11
```

OUTPUT:-



The screenshot shows a terminal window with the following output:

```
(base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs % python3 reverse_string.py  
emoclew  
(base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs %
```

The status bar at the bottom indicates 'Ln 11, Col 1', 'Spaces: 4', 'UTF-8', 'LF', and a file encoding icon.

## EXPERIMENT 6

### Check Balanced Parentheses Using Stack

CODE:-

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory named 'DSA\_LAB\_PROGRAMS' containing several Python files. The code editor shows the implementation of a function 'balanced(s)' that checks if a string 's' has balanced parentheses. The function uses a stack 'st' and a dictionary 'pairs' to map opening and closing parentheses. It iterates through each character in 's', pushing opening parentheses onto the stack and popping them when a matching closing parenthesis is found. If the stack is empty at the end, the string is balanced.

```
1 def balanced(s):
2     st = []
3     pairs = {'(': ')', '[': ']', '{': '}' }
4
5     for ch in s:
6         if ch in "([{":
7             st.append(ch)
8         elif ch in ")]}":
9             if not st or st.pop() != pairs[ch]:
10                return False
11    return len(st) == 0
12
13 print(balanced("(a+b)*(c+d)"))
14 print(balanced("{}"))
15
16
```

OUTPUT:-

The screenshot shows a terminal window with the following commands and output:

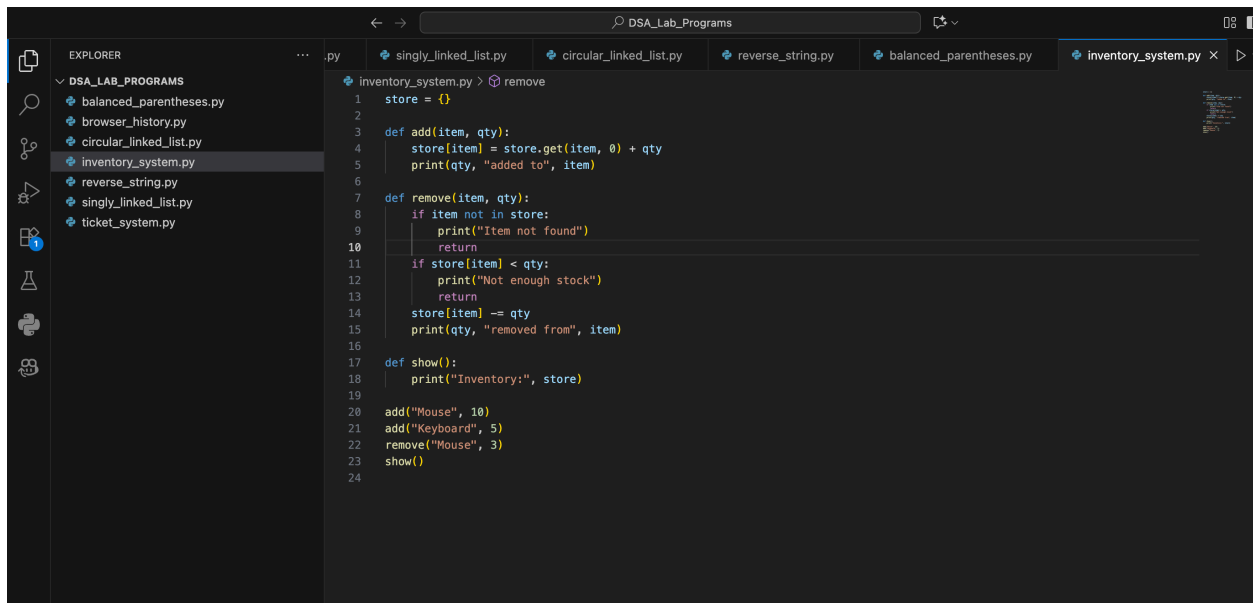
```
(base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs % python3 balanced_parentheses.py
True
False
(base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs %
```

The output shows 'True' for the first test case and 'False' for the second test case.

# EXPERIMENT 7

## Check Balanced Parentheses Using Stack

CODE:-



```
1 store = {}
2
3 def add(item, qty):
4     store[item] = store.get(item, 0) + qty
5     print(qty, "added to", item)
6
7 def remove(item, qty):
8     if item not in store:
9         print("Item not found")
10        return
11    if store[item] < qty:
12        print("Not enough stock")
13        return
14    store[item] -= qty
15    print(qty, "removed from", item)
16
17 def show():
18     print("Inventory:", store)
19
20 add("Mouse", 10)
21 add("Keyboard", 5)
22 remove("Mouse", 3)
23 show()
24
```

## OUTPUT:-

```
● (base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs % python3 inventory_system.py
10 added to Mouse
5 added to Keyboard
3 removed from Mouse
Inventory: {'Mouse': 7, 'Keyboard': 5}
○ (base) vanshrana@Vanshs-MacBook-Air DSA_Lab_Programs %
```

Ln 10, Col 15 Spaces: 4 UTF-8 L