K. R. MANGALAM UNIVERSITY, GURUGRAM, HARYANA, INDIA

Practical File

Data Structure Report file

Name: Sourabh Suman

Course: B-Tech CSE (Data Science)

Semester: 3rd

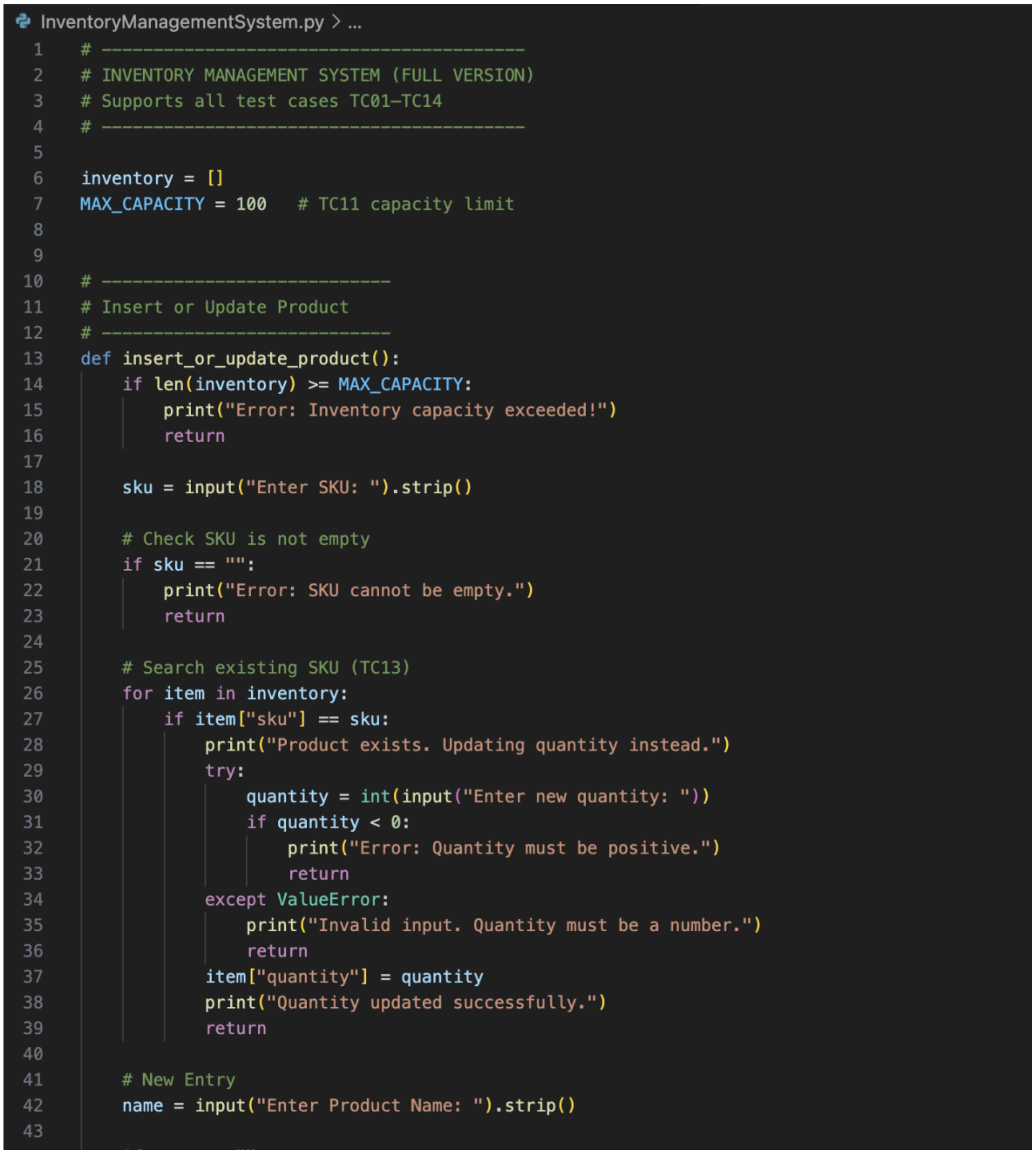
Batch: 2024-2028

Submitted to- **Signature**

Dr Swati Gupta

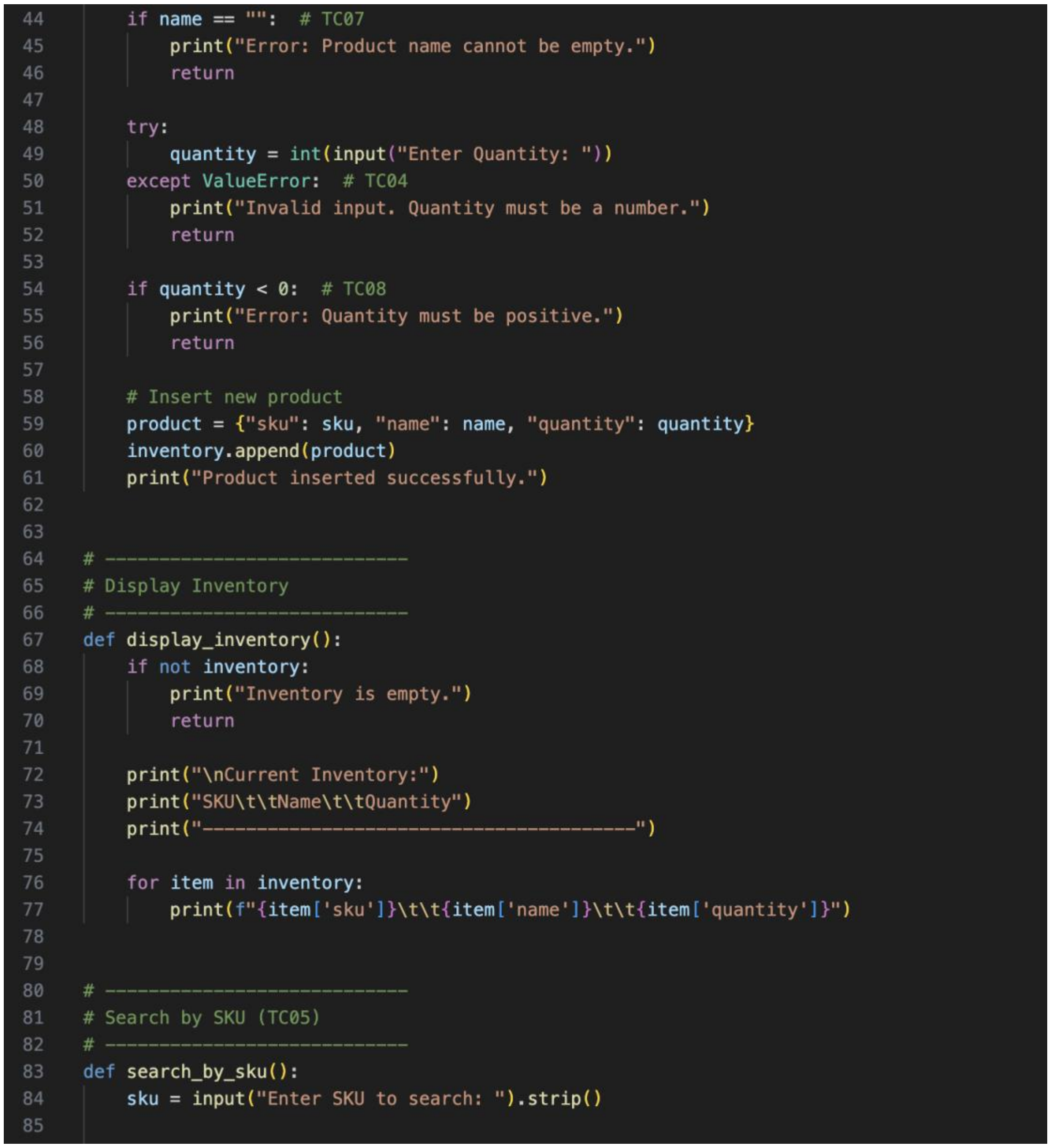
1. Inventory Management System

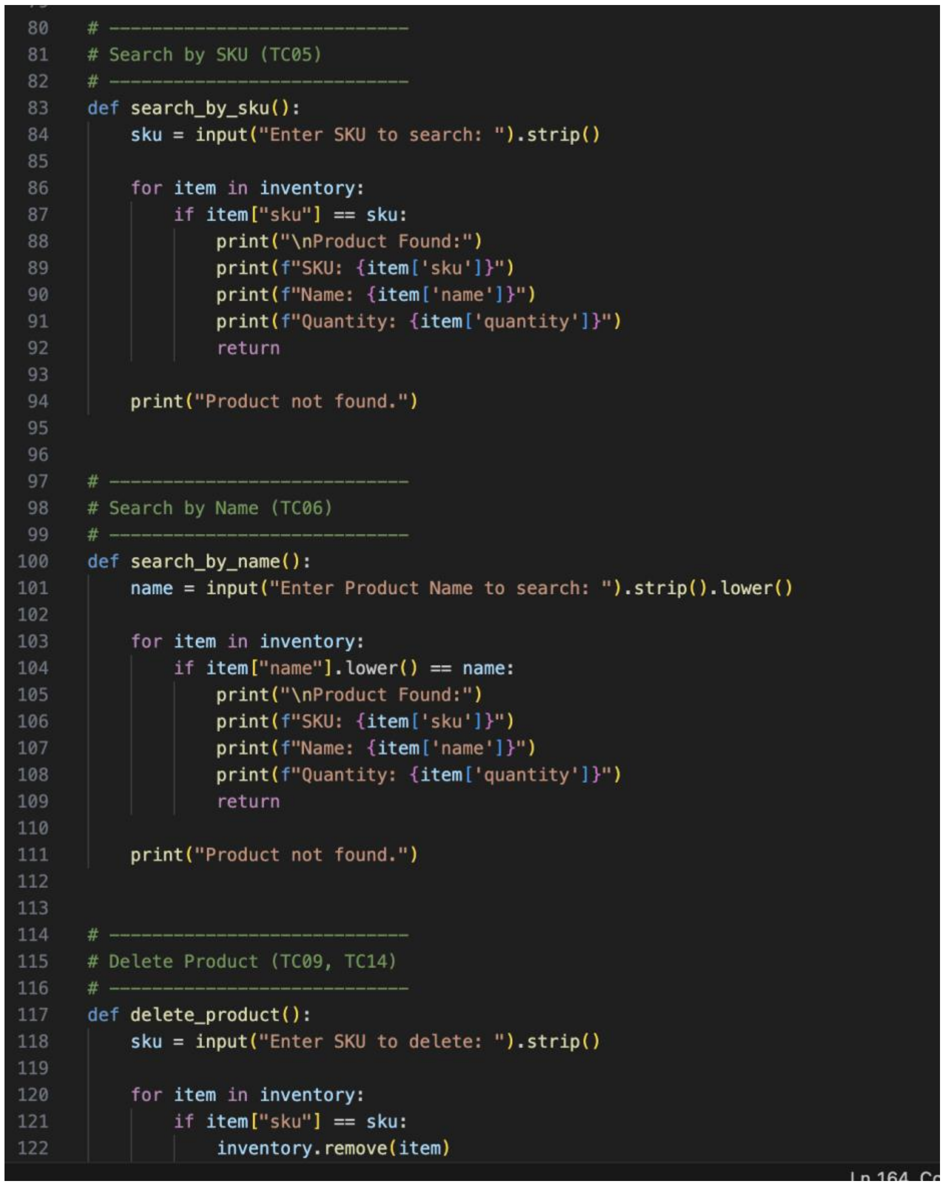
Code-



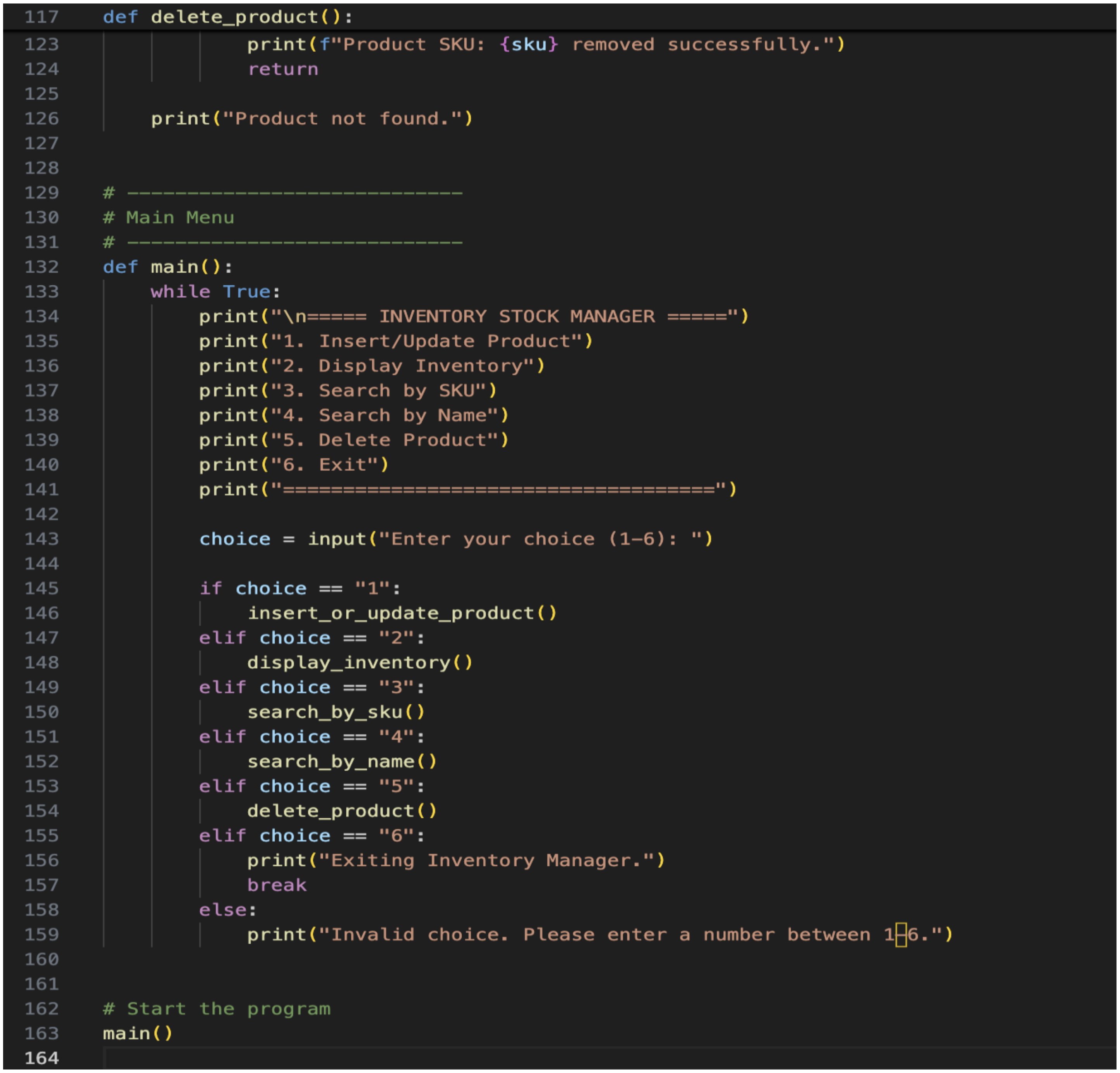
InventoryManagementSystem.py > ...

```python
# -----------------------------------------
# INVENTORY MANAGEMENT SYSTEM (FULL VERSION)
# Supports all test cases TC01–TC14
# -----------------------------------------

inventory = []
MAX_CAPACITY = 100   # TC11 capacity limit


# ----------------------------
# Insert or Update Product
# ----------------------------
def insert_or_update_product():
    if len(inventory) >= MAX_CAPACITY:
        print("Error: Inventory capacity exceeded!")
        return

    sku = input("Enter SKU: ").strip()

    # Check SKU is not empty
    if sku == "":
        print("Error: SKU cannot be empty.")
        return

    # Search existing SKU (TC13)
    for item in inventory:
        if item["sku"] == sku:
            print("Product exists. Updating quantity instead.")
            try:
                quantity = int(input("Enter new quantity: "))
                if quantity < 0:
                    print("Error: Quantity must be positive.")
                    return
            except ValueError:
                print("Invalid input. Quantity must be a number.")
                return
            item["quantity"] = quantity
            print("Quantity updated successfully.")
            return

    # New Entry
    name = input("Enter Product Name: ").strip()

```


```python
    if name == "":  # TC07
        print("Error: Product name cannot be empty.")
        return

    try:
        quantity = int(input("Enter Quantity: "))
    except ValueError:  # TC04
        print("Invalid input. Quantity must be a number.")
        return

    if quantity < 0:  # TC08
        print("Error: Quantity must be positive.")
        return

    # Insert new product
    product = {"sku": sku, "name": name, "quantity": quantity}
    inventory.append(product)
    print("Product inserted successfully.")


# ----------------------------
# Display Inventory
# ----------------------------
def display_inventory():
    if not inventory:
        print("Inventory is empty.")
        return

    print("\nCurrent Inventory:")
    print("SKU\t\tName\t\tQuantity")
    print("----------------------------------------")

    for item in inventory:
        print(f"{item['sku']}\t\t{item['name']}\t\t{item['quantity']}")


# ----------------------------
# Search by SKU (TC05)
# ----------------------------
def search_by_sku():
    sku = input("Enter SKU to search: ").strip()

```

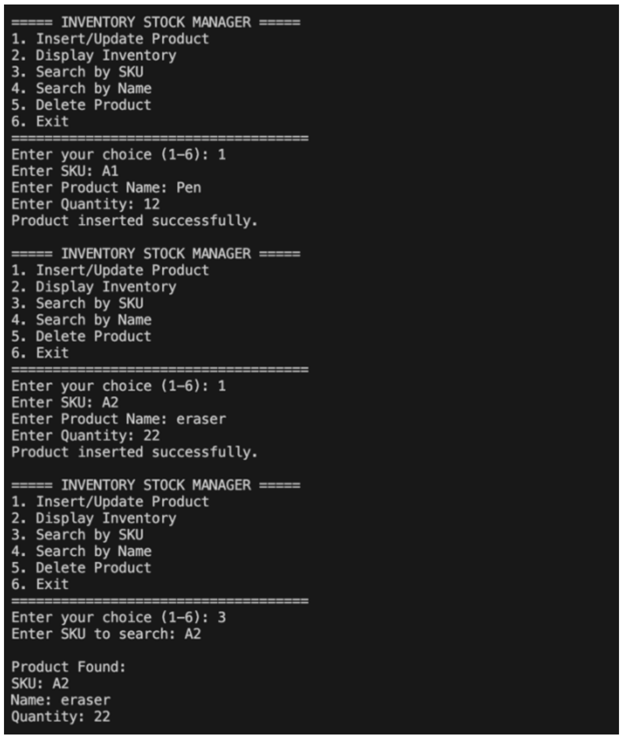


```python
# ----------------------------
# Search by SKU (TC05)
# ----------------------------
def search_by_sku():
    sku = input("Enter SKU to search: ").strip()

    for item in inventory:
        if item["sku"] == sku:
            print("\nProduct Found:")
            print(f"SKU: {item['sku']}")
            print(f"Name: {item['name']}")
            print(f"Quantity: {item['quantity']}")
            return

    print("Product not found.")


# ----------------------------
# Search by Name (TC06)
# ----------------------------
def search_by_name():
    name = input("Enter Product Name to search: ").strip().lower()

    for item in inventory:
        if item["name"].lower() == name:
            print("\nProduct Found:")
            print(f"SKU: {item['sku']}")
            print(f"Name: {item['name']}")
            print(f"Quantity: {item['quantity']}")
            return

    print("Product not found.")


# ----------------------------
# Delete Product (TC09, TC14)
# ----------------------------
def delete_product():
    sku = input("Enter SKU to delete: ").strip()

    for item in inventory:
        if item["sku"] == sku:
            inventory.remove(item)
```

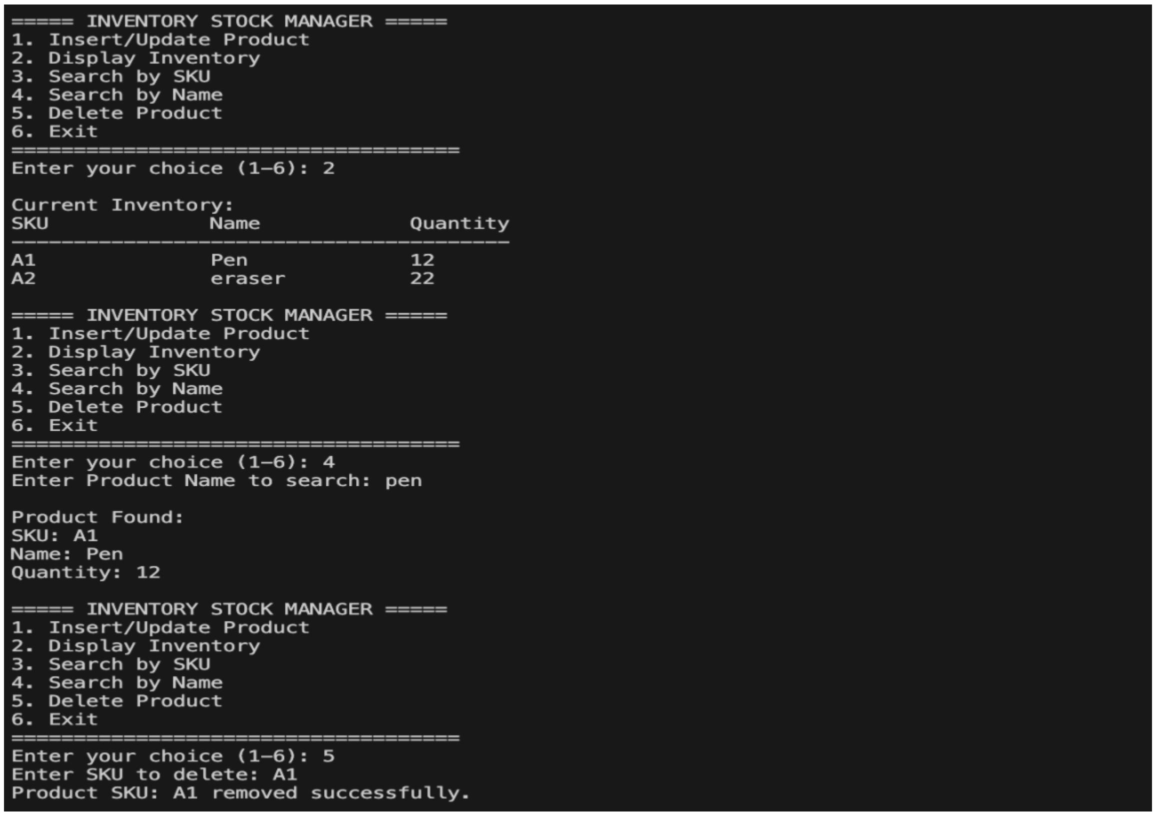


```python
def delete_product():
            print(f"Product SKU: {sku} removed successfully.")
            return

    print("Product not found.")


# ---------------------------
# Main Menu
# ---------------------------
def main():
    while True:
        print("\n===== INVENTORY STOCK MANAGER =====")
        print("1. Insert/Update Product")
        print("2. Display Inventory")
        print("3. Search by SKU")
        print("4. Search by Name")
        print("5. Delete Product")
        print("6. Exit")
        print("====================================")

        choice = input("Enter your choice (1-6): ")

        if choice == "1":
            insert_or_update_product()
        elif choice == "2":
            display_inventory()
        elif choice == "3":
            search_by_sku()
        elif choice == "4":
            search_by_name()
        elif choice == "5":
            delete_product()
        elif choice == "6":
            print("Exiting Inventory Manager.")
            break
        else:
            print("Invalid choice. Please enter a number between 1□6.")


# Start the program
main()
```

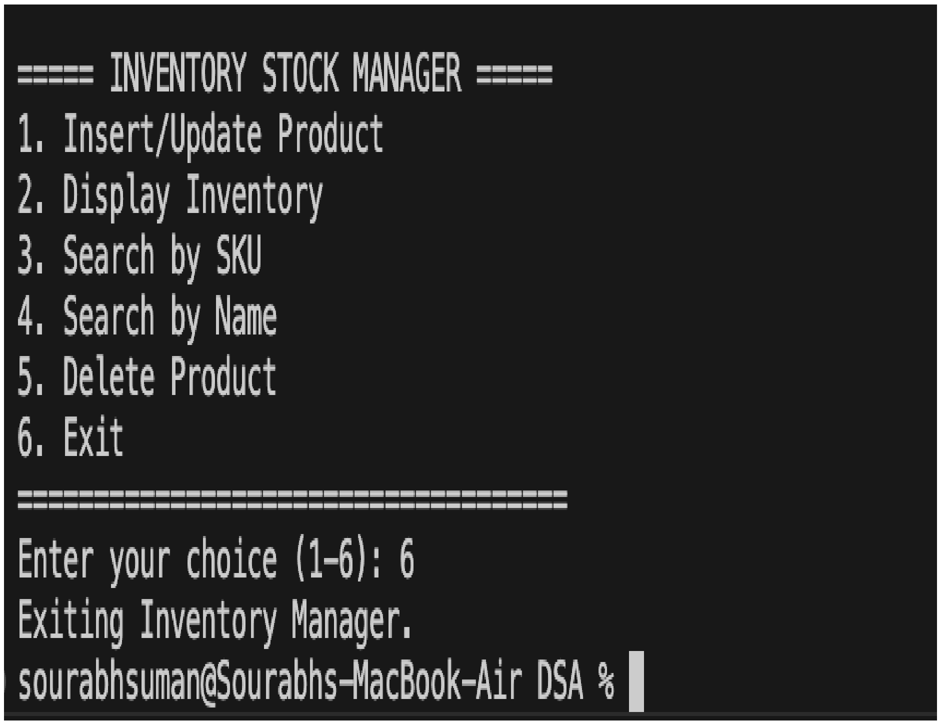
Output



```
===== INVENTORY STOCK MANAGER =====
1. Insert/Update Product
2. Display Inventory
3. Search by SKU
4. Search by Name
5. Delete Product
6. Exit
====================================
Enter your choice (1-6): 1
Enter SKU: A1
Enter Product Name: Pen
Enter Quantity: 12
Product inserted successfully.

===== INVENTORY STOCK MANAGER =====
1. Insert/Update Product
2. Display Inventory
3. Search by SKU
4. Search by Name
5. Delete Product
6. Exit
====================================
Enter your choice (1-6): 1
Enter SKU: A2
Enter Product Name: eraser
Enter Quantity: 22
Product inserted successfully.

===== INVENTORY STOCK MANAGER =====
1. Insert/Update Product
2. Display Inventory
3. Search by SKU
4. Search by Name
5. Delete Product
6. Exit
====================================
Enter your choice (1-6): 3
Enter SKU to search: A2

Product Found:
SKU: A2
Name: eraser
Quantity: 22
```



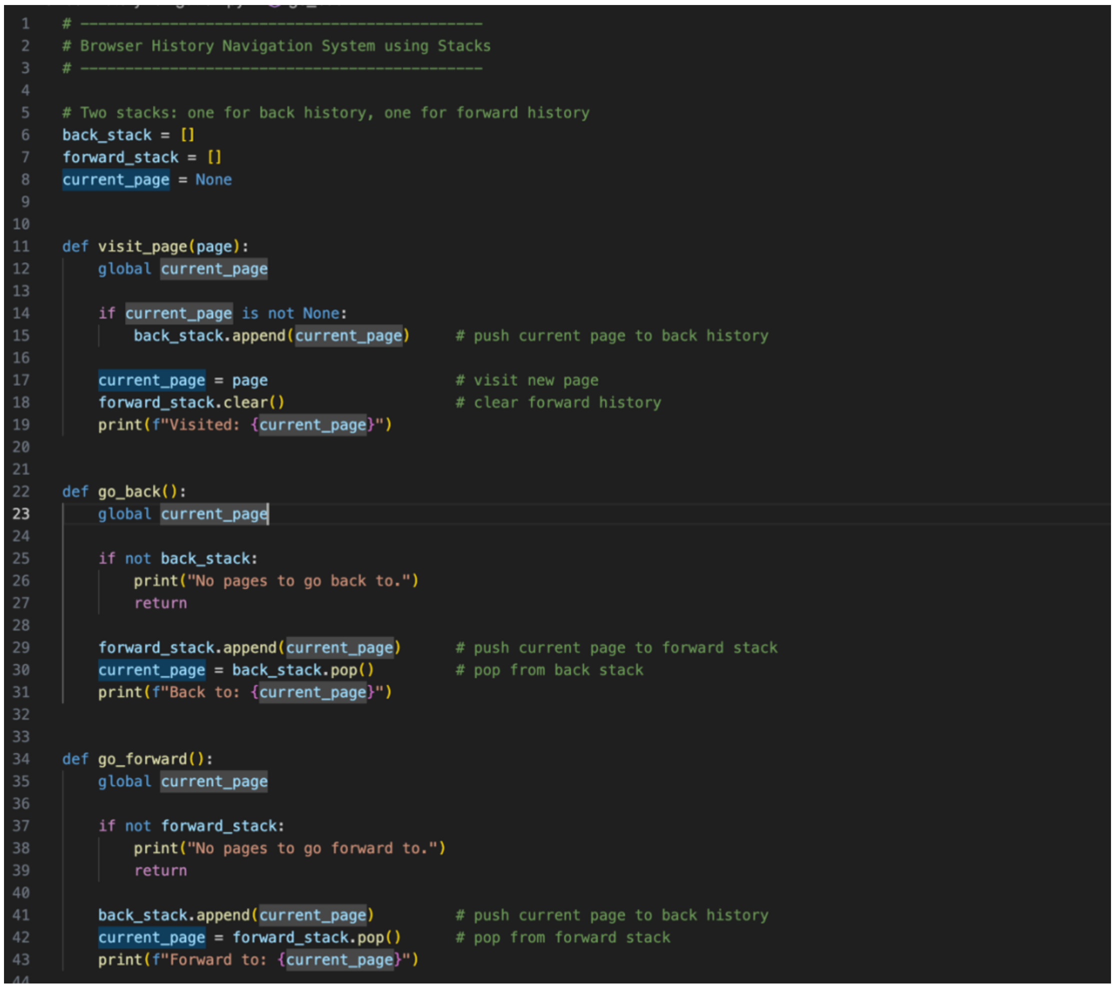
```
===== INVENTORY STOCK MANAGER =====
1. Insert/Update Product
2. Display Inventory
3. Search by SKU
4. Search by Name
5. Delete Product
6. Exit
====================================
Enter your choice (1-6): 2

Current Inventory:
SKU             Name            Quantity
----------------------------------------
A1              Pen             12
A2              eraser          22

===== INVENTORY STOCK MANAGER =====
1. Insert/Update Product
2. Display Inventory
3. Search by SKU
4. Search by Name
5. Delete Product
6. Exit
====================================
Enter your choice (1-6): 4
Enter Product Name to search: pen

Product Found:
SKU: A1
Name: Pen
Quantity: 12

===== INVENTORY STOCK MANAGER =====
1. Insert/Update Product
2. Display Inventory
3. Search by SKU
4. Search by Name
5. Delete Product
6. Exit
====================================
Enter your choice (1-6): 5
Enter SKU to delete: A1
Product SKU: A1 removed successfully.
```



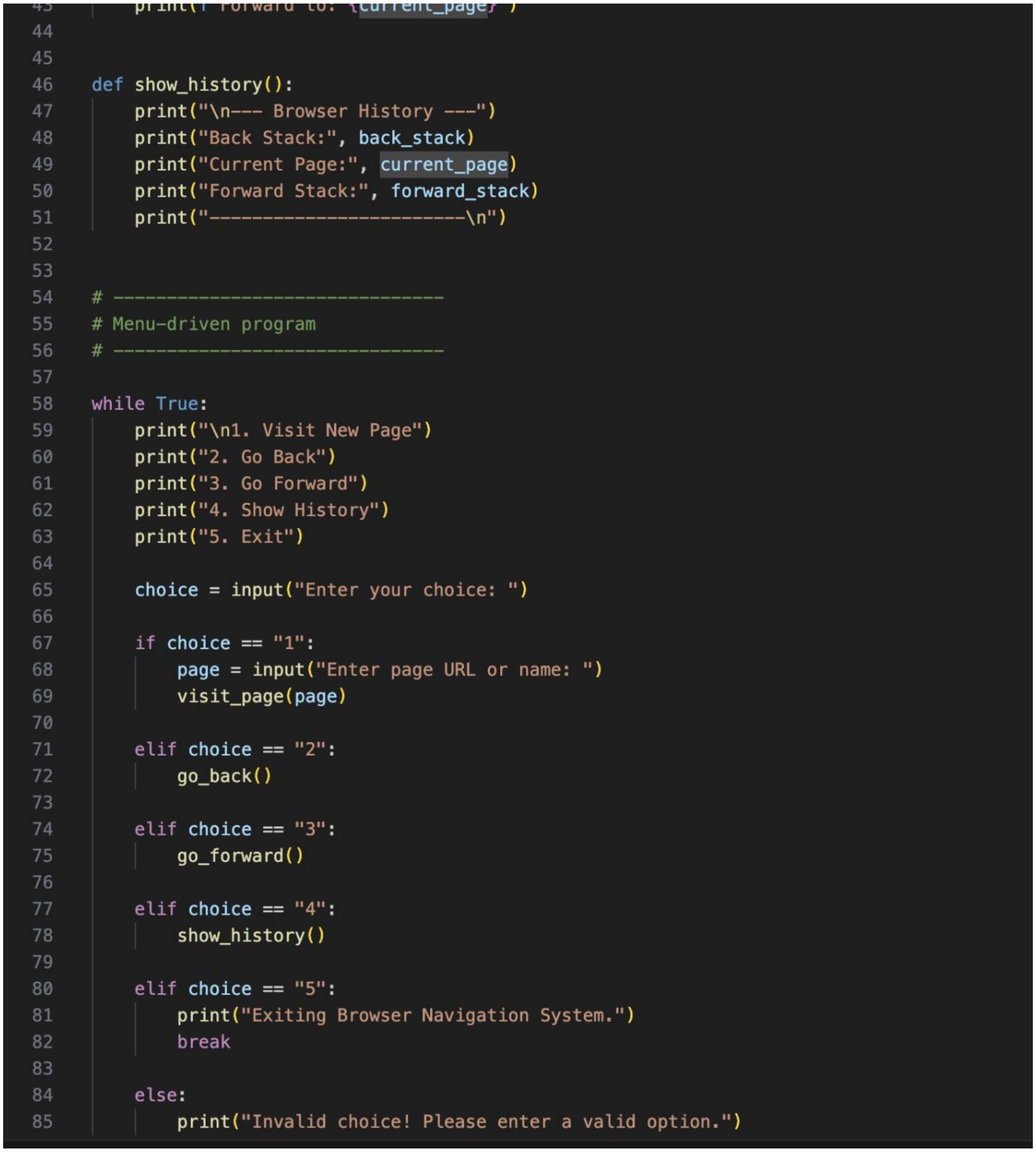
```
===== INVENTORY STOCK MANAGER =====
1. Insert/Update Product
2. Display Inventory
3. Search by SKU
4. Search by Name
5. Delete Product
6. Exit
====================================
Enter your choice (1-6): 6
Exiting Inventory Manager.
sourabhsuman@Sourabhs-MacBook-Air DSA %
```

2. Browsing Navigation

Code-

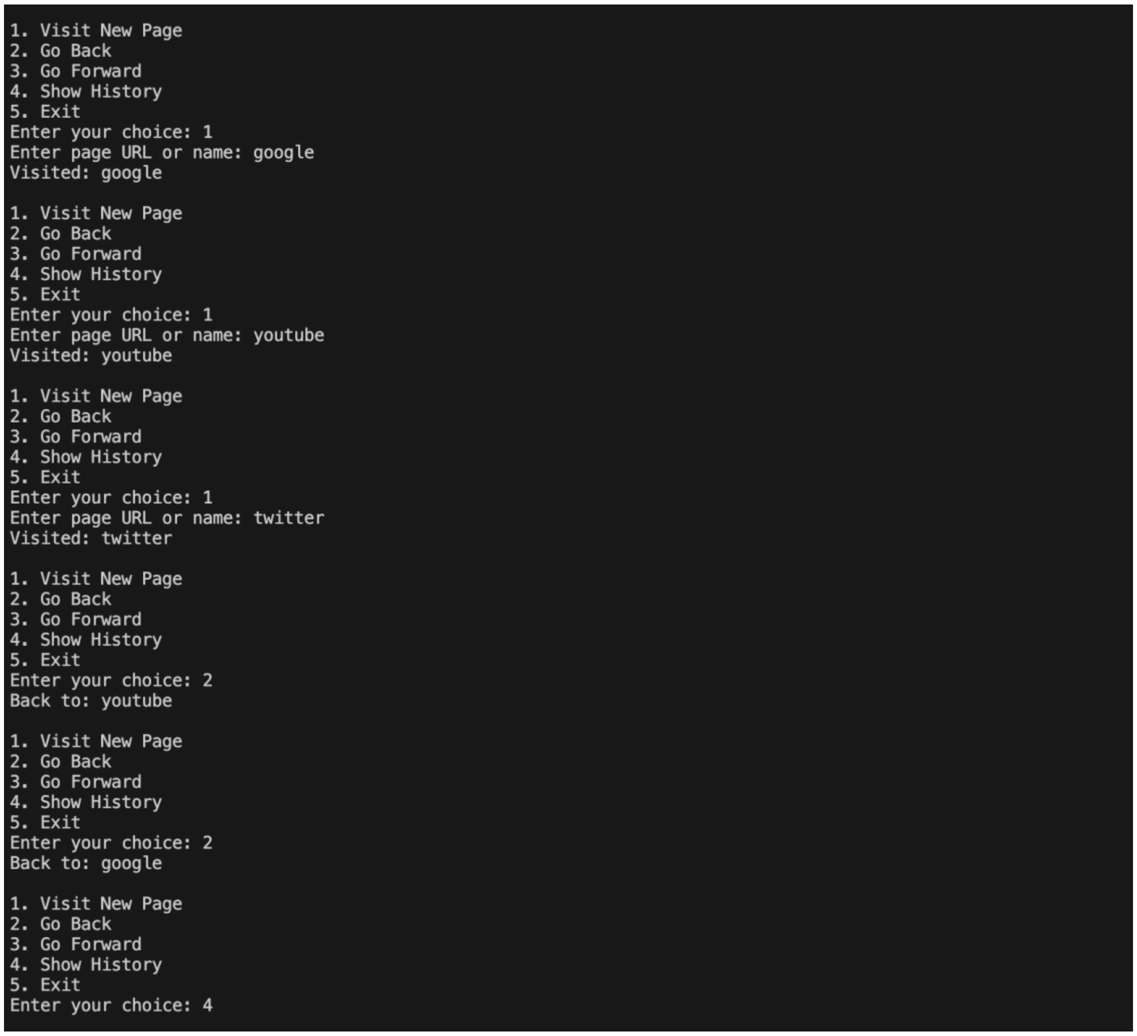


```python
# ---------------------------------------------
# Browser History Navigation System using Stacks
# ---------------------------------------------

# Two stacks: one for back history, one for forward history
back_stack = []
forward_stack = []
current_page = None


def visit_page(page):
    global current_page

    if current_page is not None:
        back_stack.append(current_page)     # push current page to back history

    current_page = page                     # visit new page
    forward_stack.clear()                   # clear forward history
    print(f"Visited: {current_page}")


def go_back():
    global current_page

    if not back_stack:
        print("No pages to go back to.")
        return

    forward_stack.append(current_page)      # push current page to forward stack
    current_page = back_stack.pop()         # pop from back stack
    print(f"Back to: {current_page}")


def go_forward():
    global current_page

    if not forward_stack:
        print("No pages to go forward to.")
        return

    back_stack.append(current_page)         # push current page to back history
    current_page = forward_stack.pop()      # pop from forward stack
    print(f"Forward to: {current_page}")
```

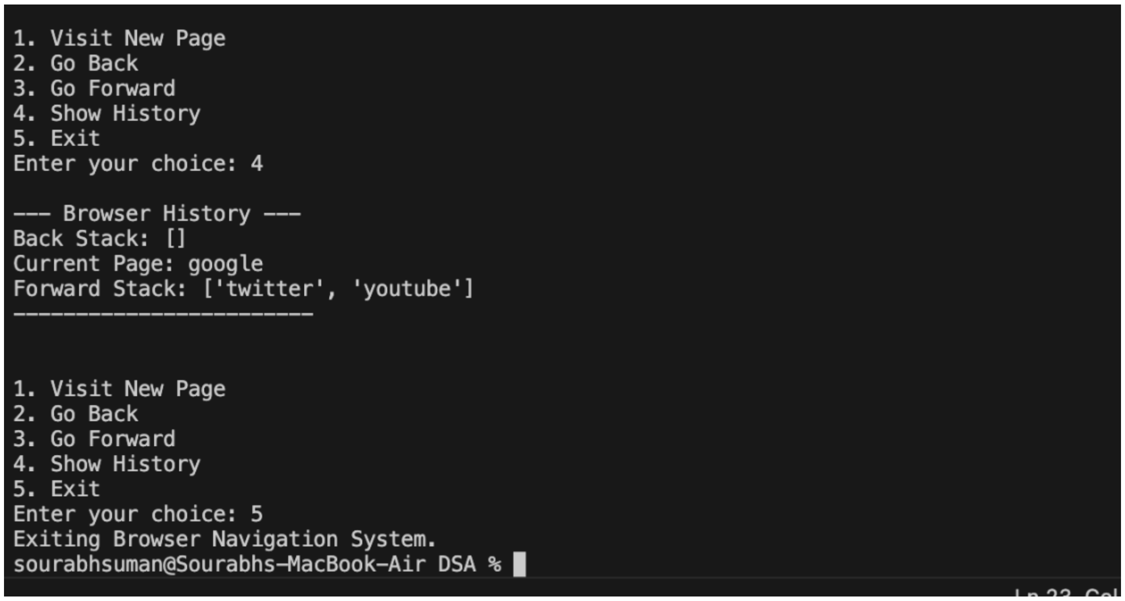


```python
    print(f"Forward to: {current_page}")


def show_history():
    print("\n--- Browser History ---")
    print("Back Stack:", back_stack)
    print("Current Page:", current_page)
    print("Forward Stack:", forward_stack)
    print("------------------------\n")


# ------------------------------
# Menu-driven program
# ------------------------------

while True:
    print("\n1. Visit New Page")
    print("2. Go Back")
    print("3. Go Forward")
    print("4. Show History")
    print("5. Exit")

    choice = input("Enter your choice: ")

    if choice == "1":
        page = input("Enter page URL or name: ")
        visit_page(page)

    elif choice == "2":
        go_back()

    elif choice == "3":
        go_forward()

    elif choice == "4":
        show_history()

    elif choice == "5":
        print("Exiting Browser Navigation System.")
        break

    else:
        print("Invalid choice! Please enter a valid option.")
```

Output-



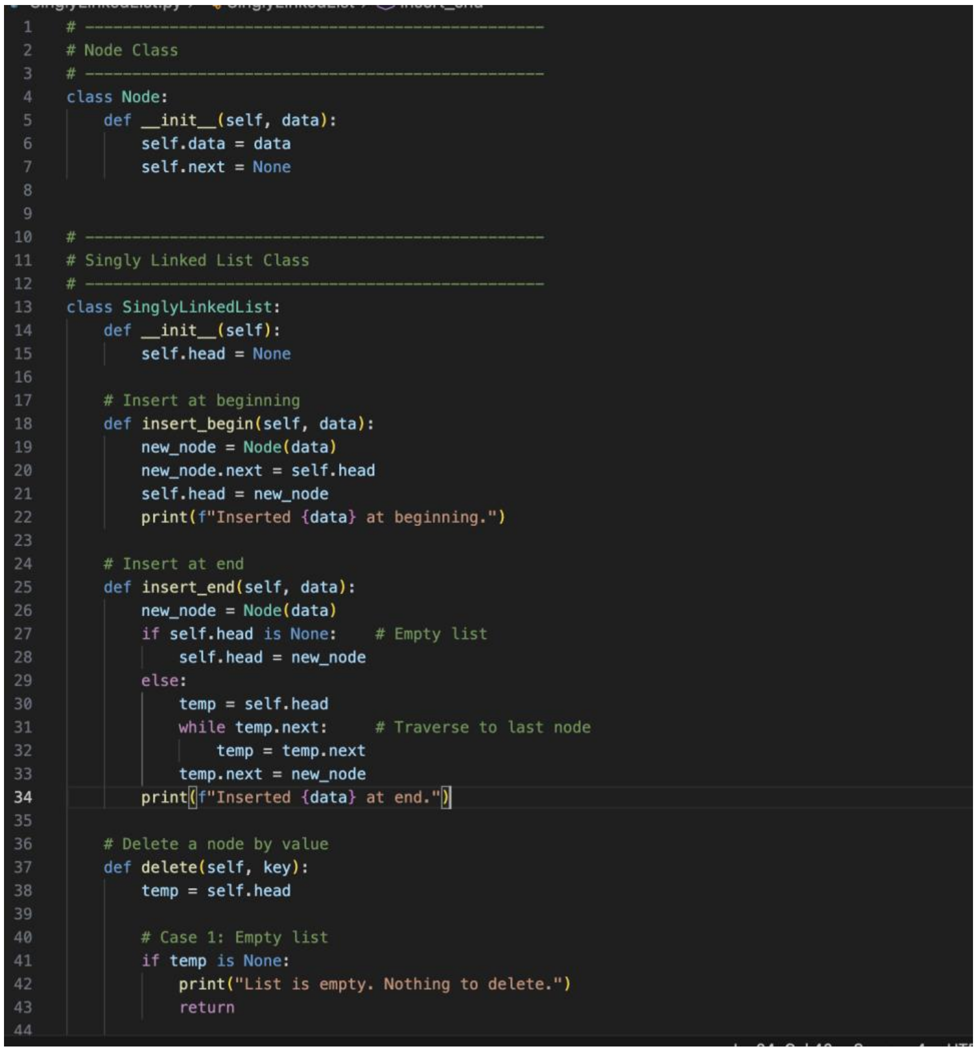
```
1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 1
Enter page URL or name: google
Visited: google

1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 1
Enter page URL or name: youtube
Visited: youtube

1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 1
Enter page URL or name: twitter
Visited: twitter

1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 2
Back to: youtube

1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 2
Back to: google

1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 4
```



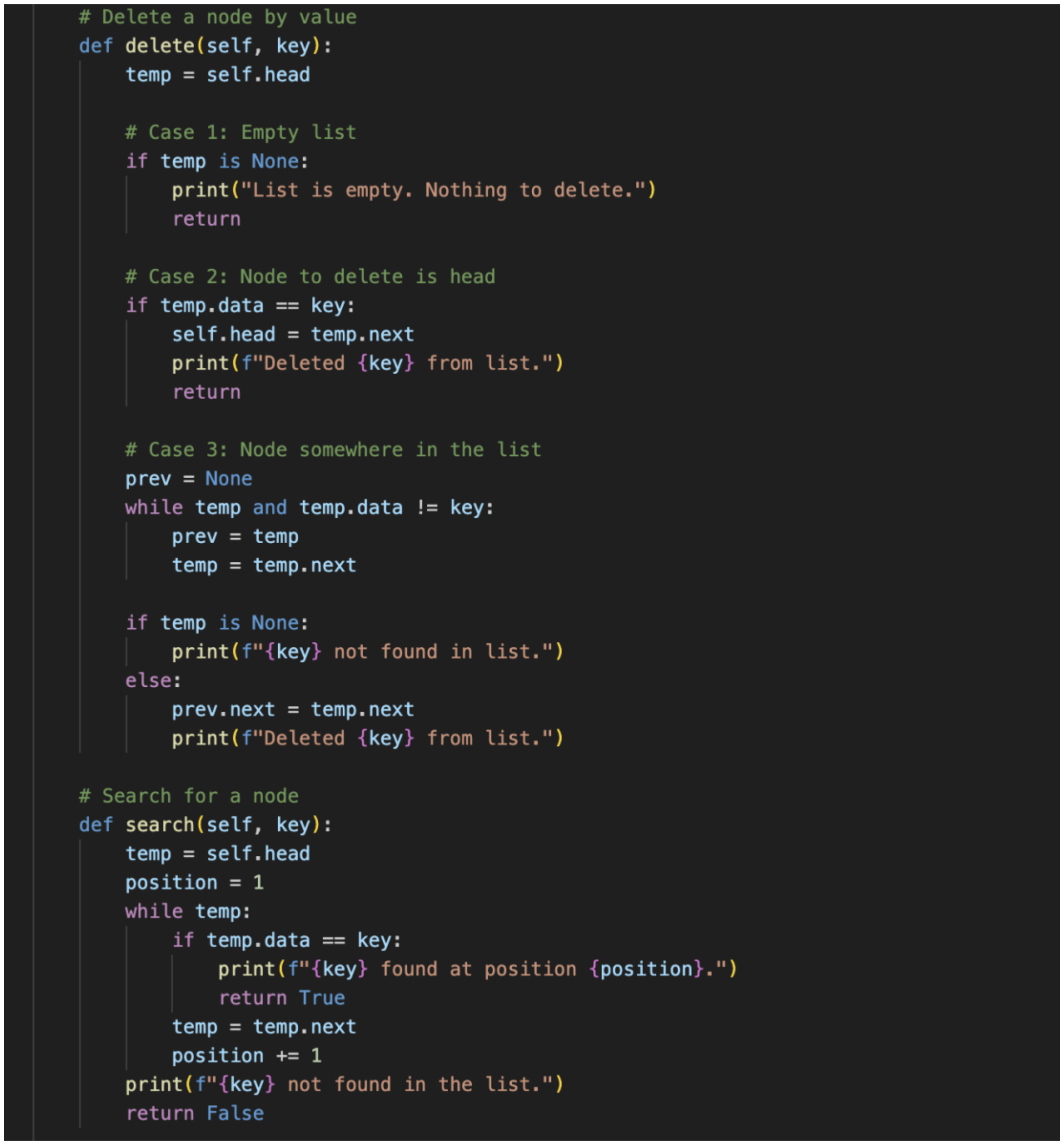
```
1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 4

--- Browser History ---
Back Stack: []
Current Page: google
Forward Stack: ['twitter', 'youtube']
------------------------


1. Visit New Page
2. Go Back
3. Go Forward
4. Show History
5. Exit
Enter your choice: 5
Exiting Browser Navigation System.
sourabhsuman@Sourabhs-MacBook-Air DSA %
```

3. Singly Linked List

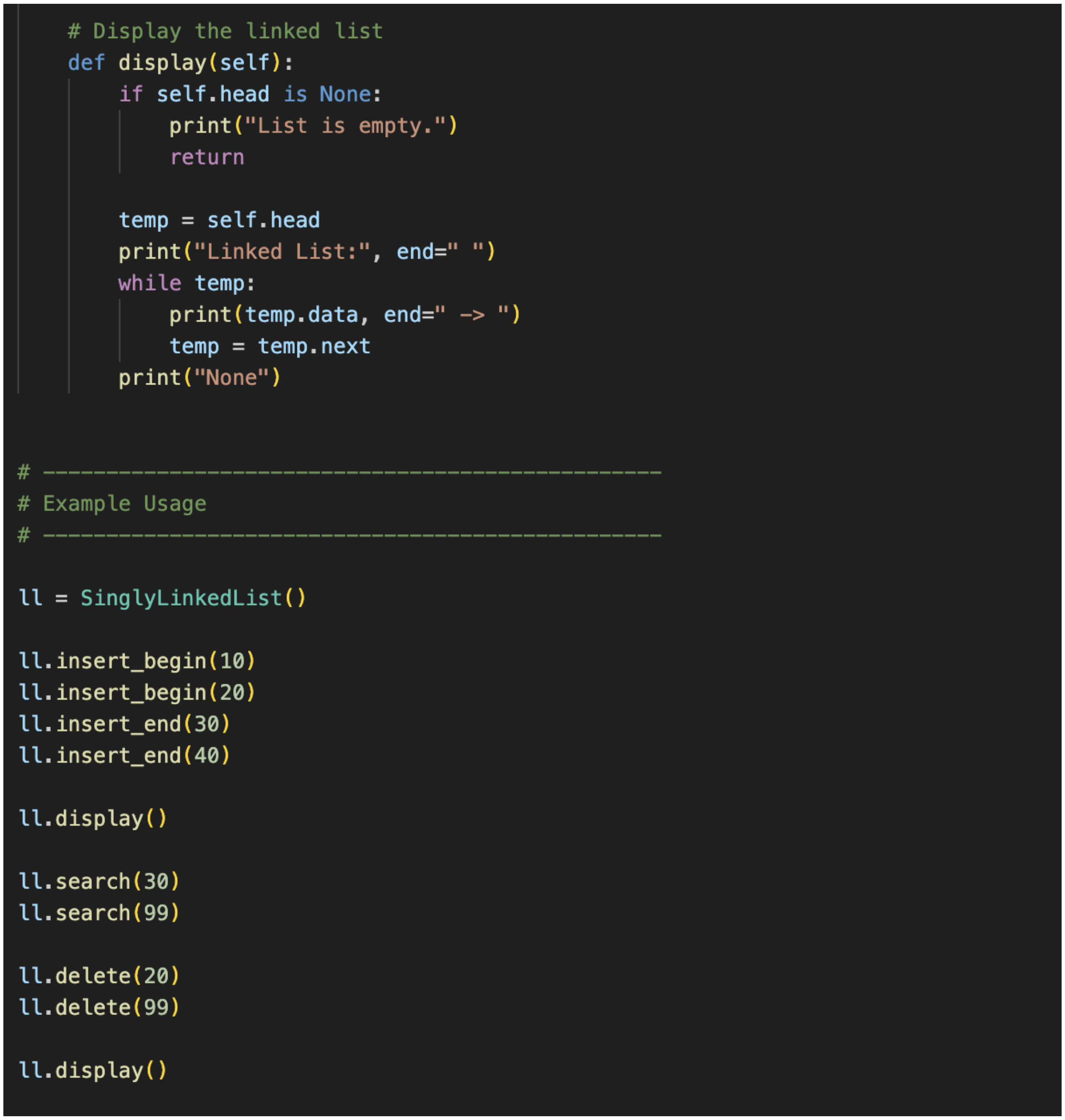
Code-



```python
# ------------------------------------------------
# Node Class
# ------------------------------------------------
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None


# ------------------------------------------------
# Singly Linked List Class
# ------------------------------------------------
class SinglyLinkedList:
    def __init__(self):
        self.head = None

    # Insert at beginning
    def insert_begin(self, data):
        new_node = Node(data)
        new_node.next = self.head
        self.head = new_node
        print(f"Inserted {data} at beginning.")

    # Insert at end
    def insert_end(self, data):
        new_node = Node(data)
        if self.head is None:    # Empty list
            self.head = new_node
        else:
            temp = self.head
            while temp.next:     # Traverse to last node
                temp = temp.next
            temp.next = new_node
        print(f"Inserted {data} at end.")

    # Delete a node by value
    def delete(self, key):
        temp = self.head

        # Case 1: Empty list
        if temp is None:
            print("List is empty. Nothing to delete.")
            return

```

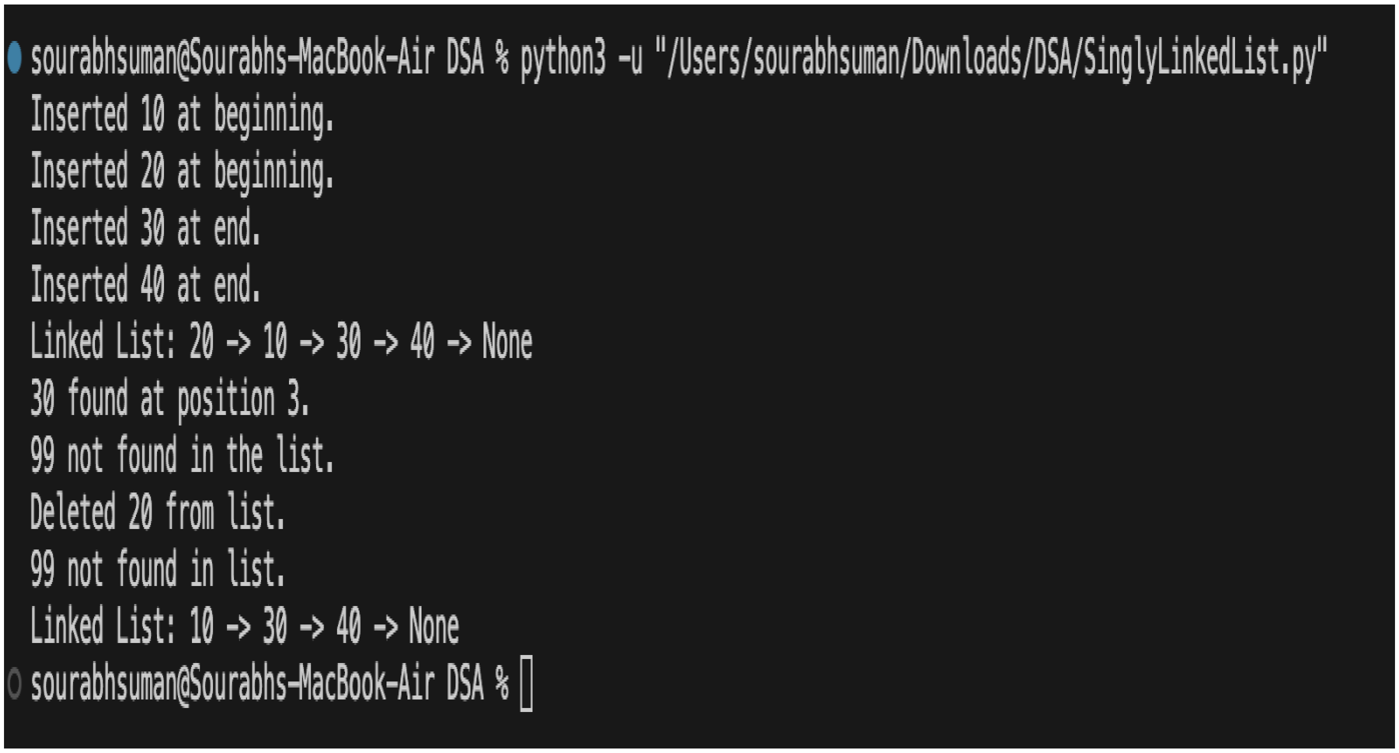


```python
    # Delete a node by value
    def delete(self, key):
        temp = self.head

        # Case 1: Empty list
        if temp is None:
            print("List is empty. Nothing to delete.")
            return

        # Case 2: Node to delete is head
        if temp.data == key:
            self.head = temp.next
            print(f"Deleted {key} from list.")
            return

        # Case 3: Node somewhere in the list
        prev = None
        while temp and temp.data != key:
            prev = temp
            temp = temp.next

        if temp is None:
            print(f"{key} not found in list.")
        else:
            prev.next = temp.next
            print(f"Deleted {key} from list.")

    # Search for a node
    def search(self, key):
        temp = self.head
        position = 1
        while temp:
            if temp.data == key:
                print(f"{key} found at position {position}.")
                return True
            temp = temp.next
            position += 1
        print(f"{key} not found in the list.")
        return False
```



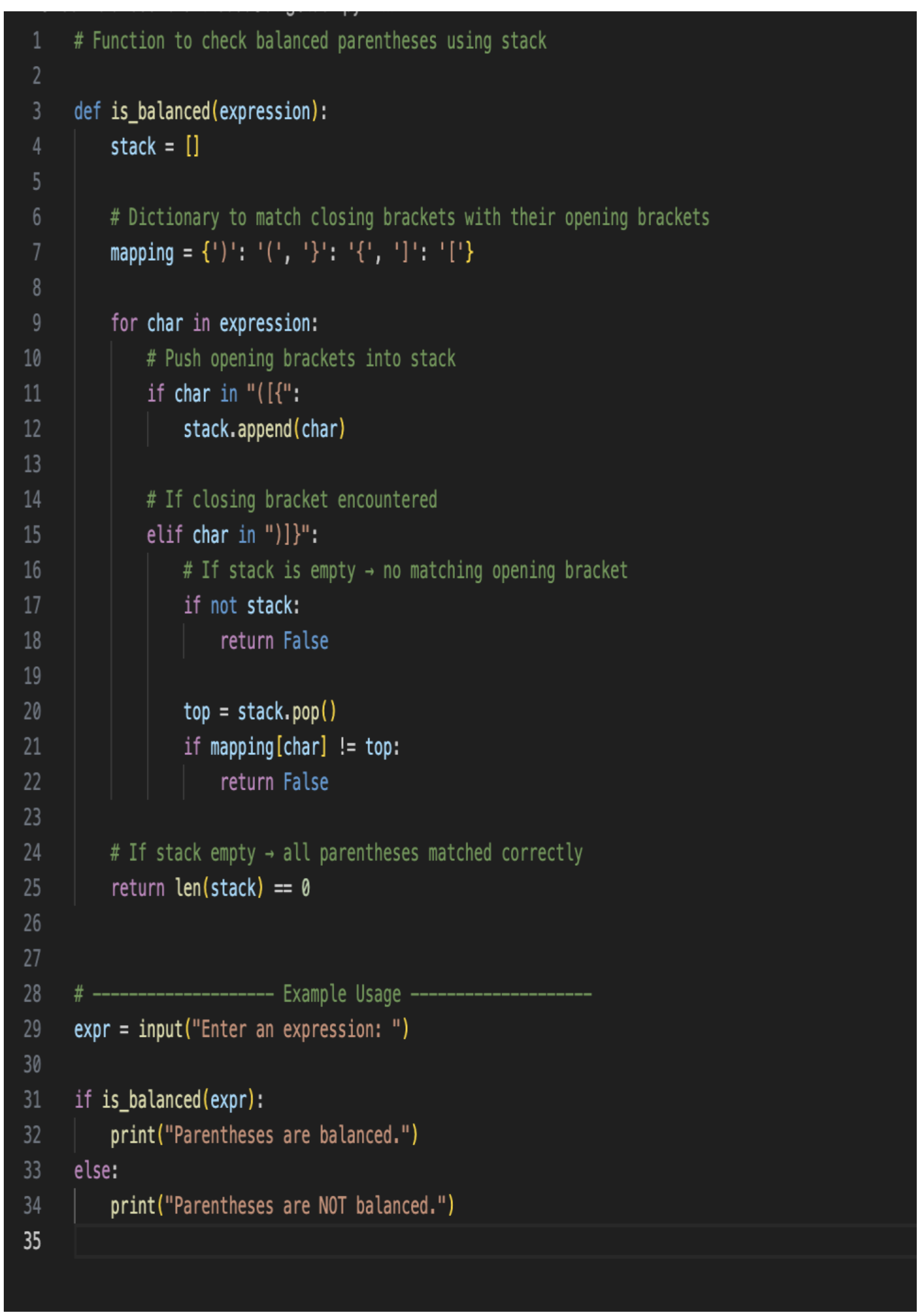
```python
    # Display the linked list
    def display(self):
        if self.head is None:
            print("List is empty.")
            return

        temp = self.head
        print("Linked List:", end=" ")
        while temp:
            print(temp.data, end=" -> ")
            temp = temp.next
        print("None")


# --------------------------------------------------
# Example Usage
# --------------------------------------------------

ll = SinglyLinkedList()

ll.insert_begin(10)
ll.insert_begin(20)
ll.insert_end(30)
ll.insert_end(40)

ll.display()

ll.search(30)
ll.search(99)

ll.delete(20)
ll.delete(99)

ll.display()
```

Output-



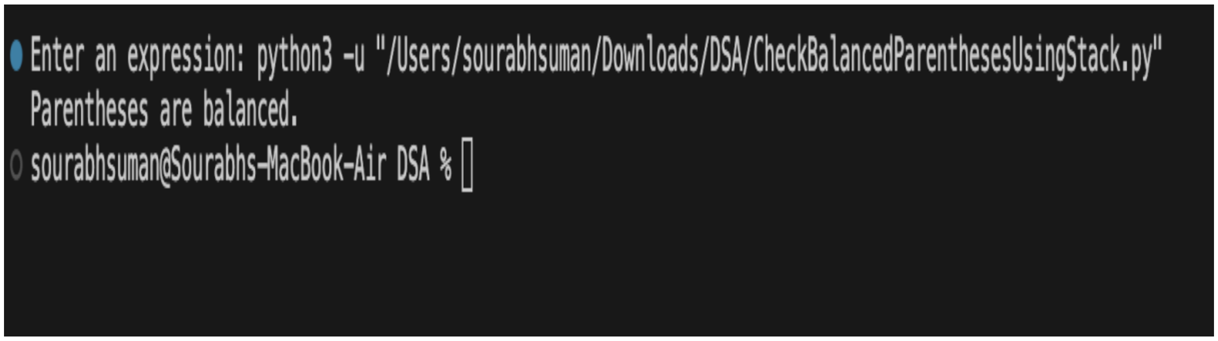
```
sourabhsuman@Sourabhs-MacBook-Air DSA % python3 -u "/Users/sourabhsuman/Downloads/DSA/SinglyLinkedList.py"
Inserted 10 at beginning.
Inserted 20 at beginning.
Inserted 30 at end.
Inserted 40 at end.
Linked List: 20 -> 10 -> 30 -> 40 -> None
30 found at position 3.
99 not found in the list.
Deleted 20 from list.
99 not found in list.
Linked List: 10 -> 30 -> 40 -> None
sourabhsuman@Sourabhs-MacBook-Air DSA %
```

4. Balanced Parentheses Using Stack



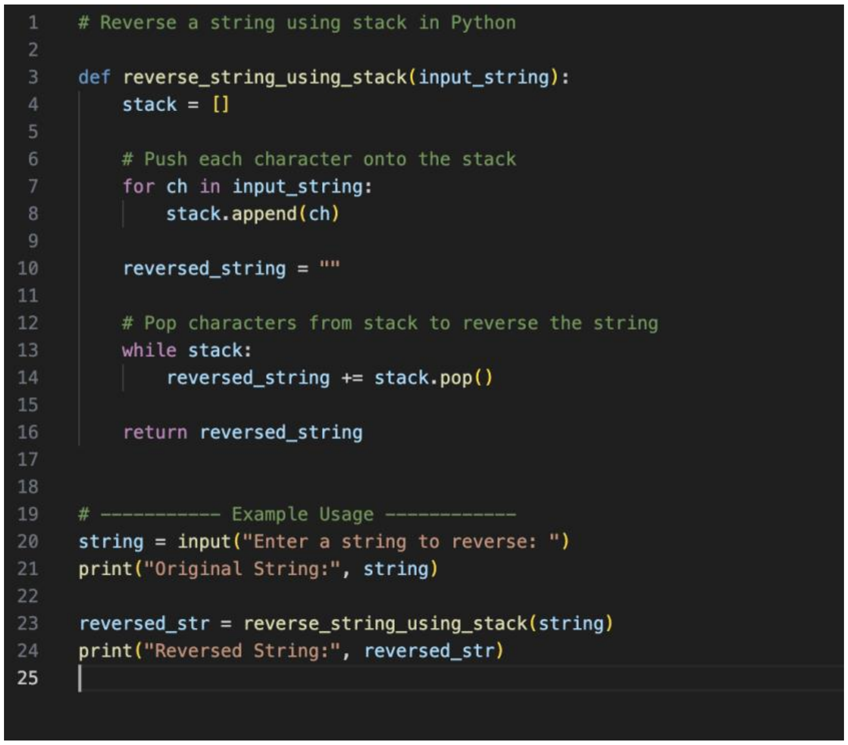
```python
# Function to check balanced parentheses using stack

def is_balanced(expression):
    stack = []

    # Dictionary to match closing brackets with their opening brackets
    mapping = {')': '(', '}': '{', ']': '['}

    for char in expression:
        # Push opening brackets into stack
        if char in "([{":
            stack.append(char)

        # If closing bracket encountered
        elif char in ")]}":
            # If stack is empty → no matching opening bracket
            if not stack:
                return False

            top = stack.pop()
            if mapping[char] != top:
                return False

    # If stack empty → all parentheses matched correctly
    return len(stack) == 0


# -------------------- Example Usage --------------------
expr = input("Enter an expression: ")

if is_balanced(expr):
    print("Parentheses are balanced.")
else:
    print("Parentheses are NOT balanced.")
```

Output-

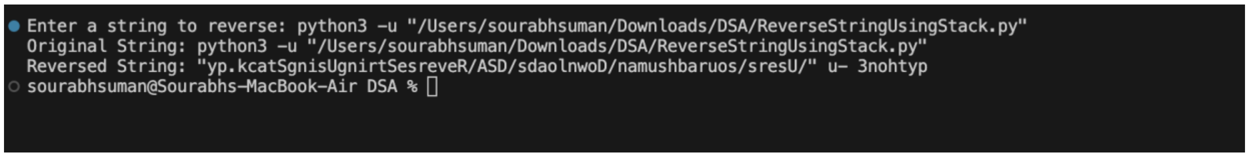

```
Enter an expression: python3 -u "/Users/sourabhsuman/Downloads/DSA/CheckBalancedParenthesesUsingStack.py"
Parentheses are balanced.
sourabhsuman@Sourabhs-MacBook-Air DSA %
```

5. Reverse of String Using Stack

Code-



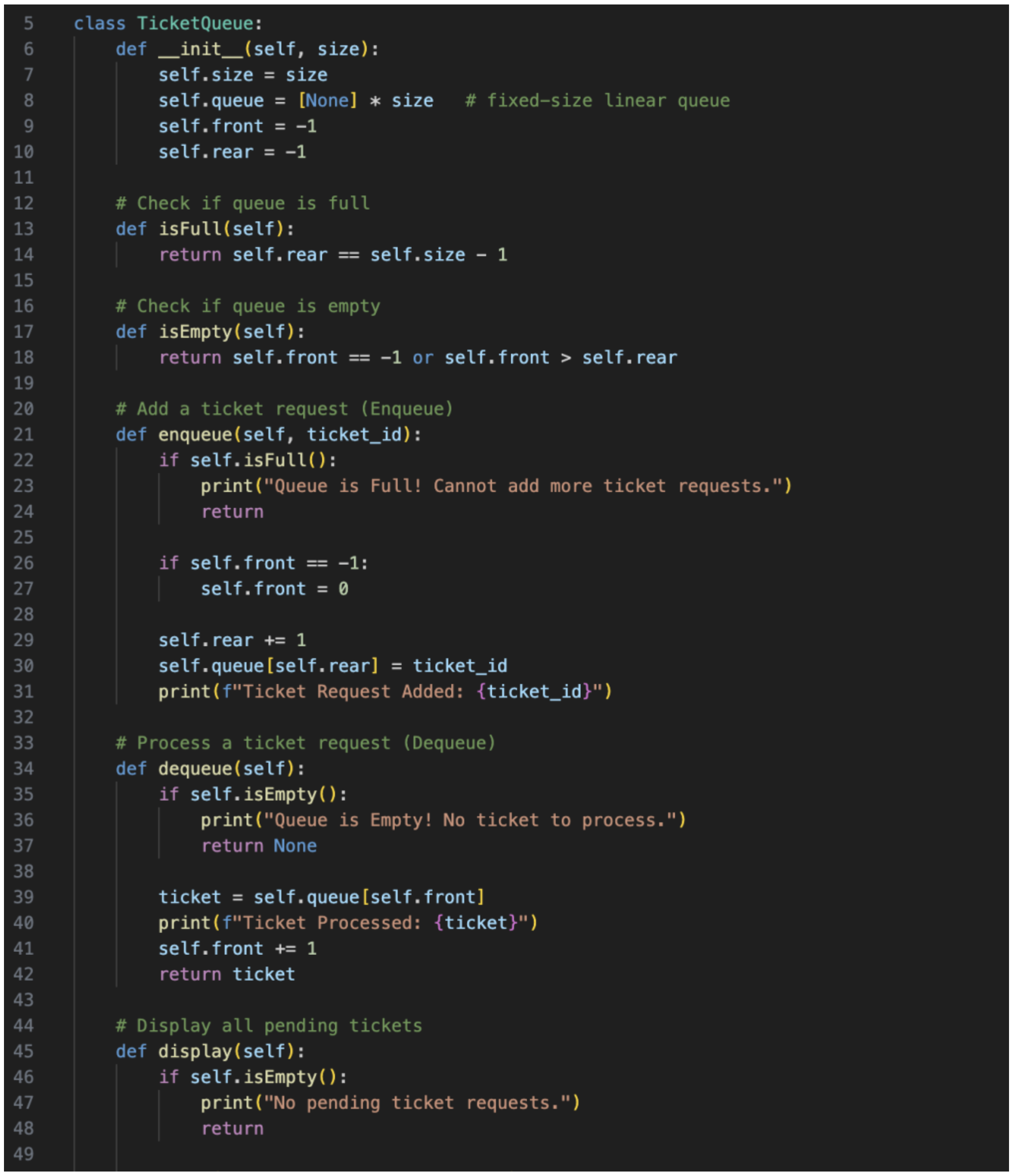
```python
# Reverse a string using stack in Python

def reverse_string_using_stack(input_string):
    stack = []

    # Push each character onto the stack
    for ch in input_string:
        stack.append(ch)

    reversed_string = ""

    # Pop characters from stack to reverse the string
    while stack:
        reversed_string += stack.pop()

    return reversed_string


# ----------- Example Usage ------------
string = input("Enter a string to reverse: ")
print("Original String:", string)

reversed_str = reverse_string_using_stack(string)
print("Reversed String:", reversed_str)

```



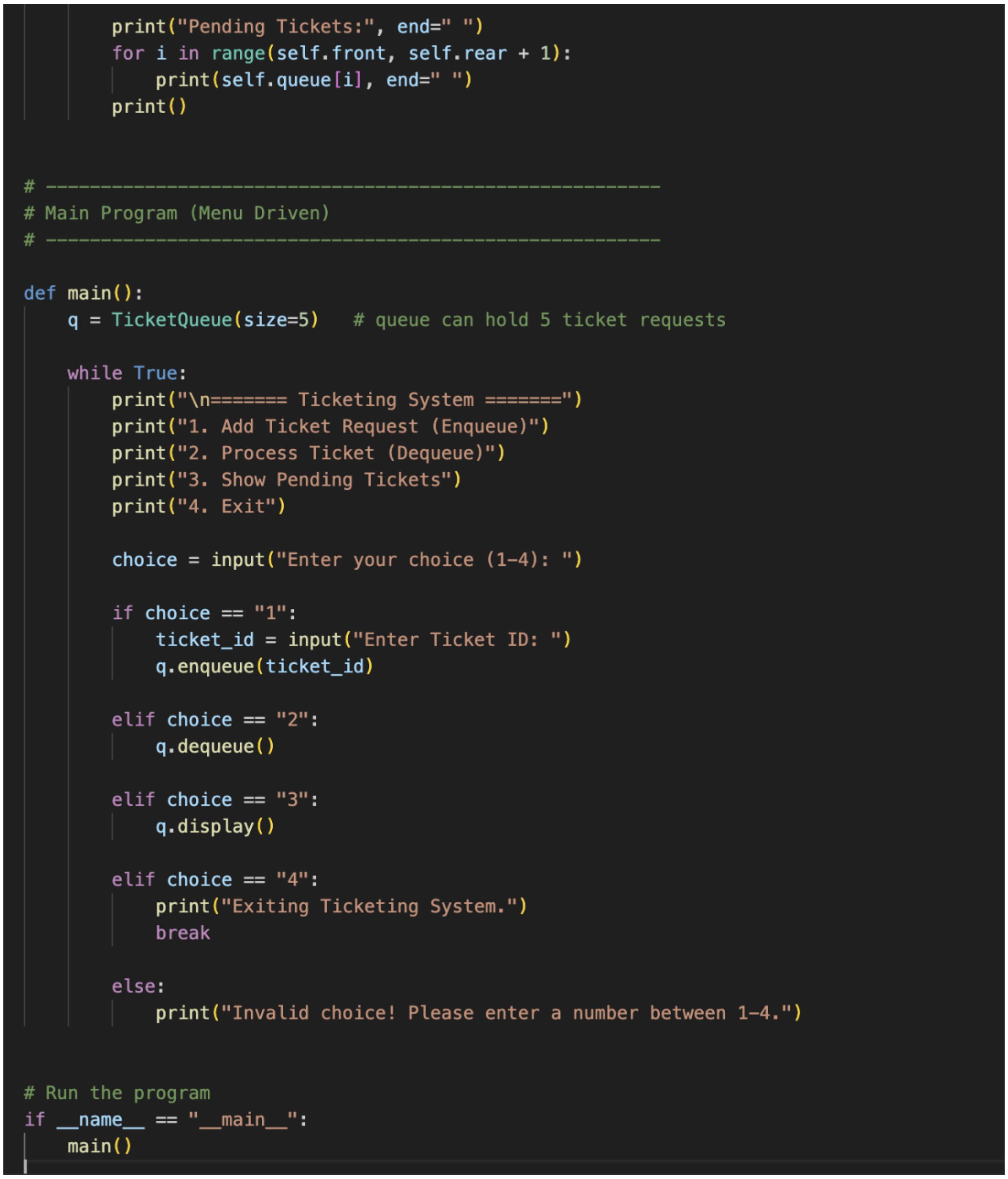
```
● Enter a string to reverse: python3 -u "/Users/sourabhsuman/Downloads/DSA/ReverseStringUsingStack.py"
  Original String: python3 -u "/Users/sourabhsuman/Downloads/DSA/ReverseStringUsingStack.py"
  Reversed String: "yp.kcatSgnisUgnirtSesreveR/ASD/sdaolnwoD/namushbaruos/sresU/" u- 3nohtyp
○ sourabhsuman@Sourabhs-MacBook-Air DSA % []
```

6. Ticket Management System using Linear Queue

Code-

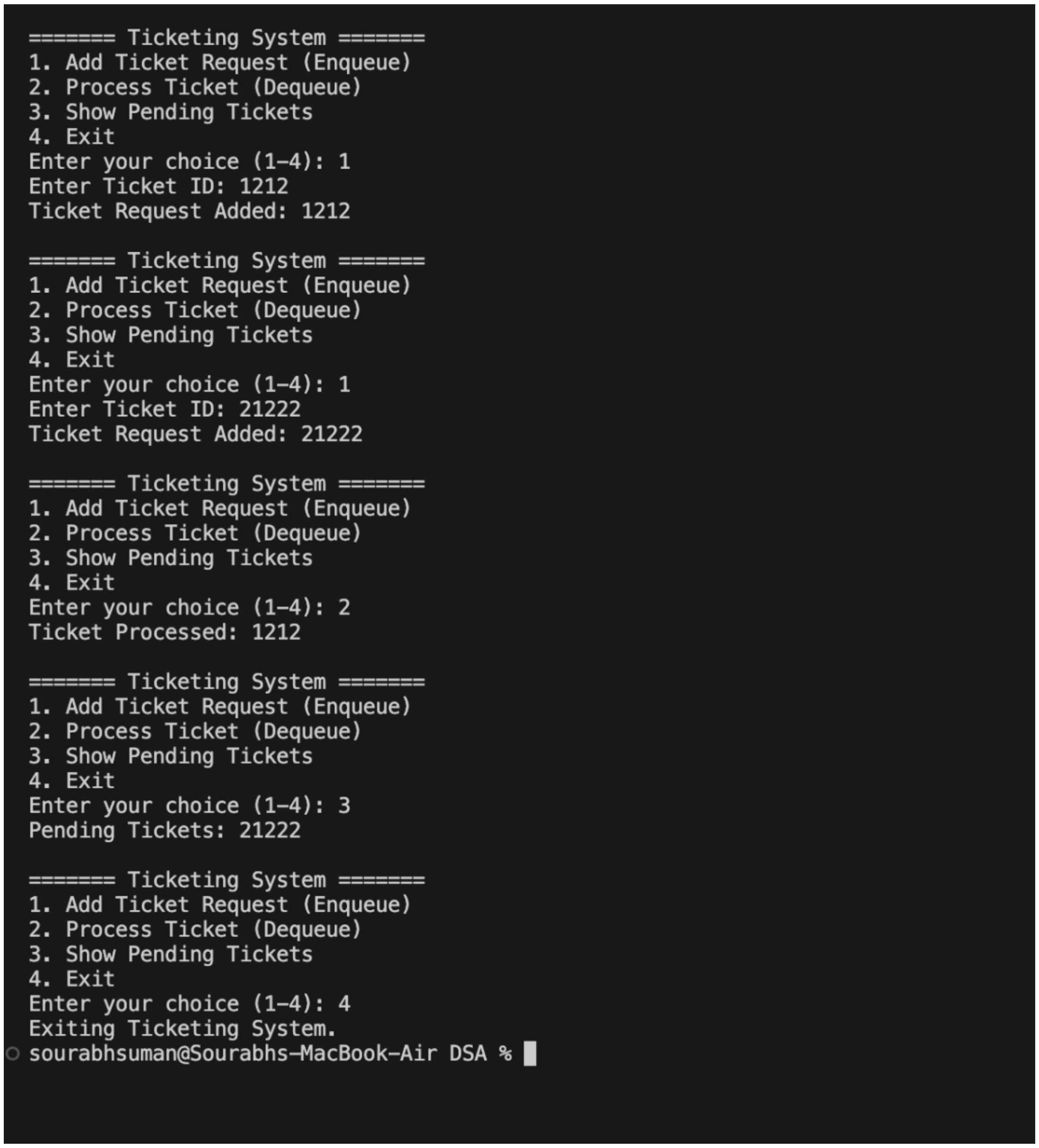


```python
class TicketQueue:
    def __init__(self, size):
        self.size = size
        self.queue = [None] * size   # fixed-size linear queue
        self.front = -1
        self.rear = -1

    # Check if queue is full
    def isFull(self):
        return self.rear == self.size - 1

    # Check if queue is empty
    def isEmpty(self):
        return self.front == -1 or self.front > self.rear

    # Add a ticket request (Enqueue)
    def enqueue(self, ticket_id):
        if self.isFull():
            print("Queue is Full! Cannot add more ticket requests.")
            return

        if self.front == -1:
            self.front = 0

        self.rear += 1
        self.queue[self.rear] = ticket_id
        print(f"Ticket Request Added: {ticket_id}")

    # Process a ticket request (Dequeue)
    def dequeue(self):
        if self.isEmpty():
            print("Queue is Empty! No ticket to process.")
            return None

        ticket = self.queue[self.front]
        print(f"Ticket Processed: {ticket}")
        self.front += 1
        return ticket

    # Display all pending tickets
    def display(self):
        if self.isEmpty():
            print("No pending ticket requests.")
            return

```



```python
        print("Pending Tickets:", end=" ")
        for i in range(self.front, self.rear + 1):
            print(self.queue[i], end=" ")
        print()


# --------------------------------------------------------
# Main Program (Menu Driven)
# --------------------------------------------------------

def main():
    q = TicketQueue(size=5)   # queue can hold 5 ticket requests

    while True:
        print("\n======= Ticketing System =======")
        print("1. Add Ticket Request (Enqueue)")
        print("2. Process Ticket (Dequeue)")
        print("3. Show Pending Tickets")
        print("4. Exit")

        choice = input("Enter your choice (1-4): ")

        if choice == "1":
            ticket_id = input("Enter Ticket ID: ")
            q.enqueue(ticket_id)

        elif choice == "2":
            q.dequeue()

        elif choice == "3":
            q.display()

        elif choice == "4":
            print("Exiting Ticketing System.")
            break

        else:
            print("Invalid choice! Please enter a number between 1-4.")


# Run the program
if __name__ == "__main__":
    main()
```

Output



```
======= Ticketing System =======
1. Add Ticket Request (Enqueue)
2. Process Ticket (Dequeue)
3. Show Pending Tickets
4. Exit
Enter your choice (1-4): 1
Enter Ticket ID: 1212
Ticket Request Added: 1212

======= Ticketing System =======
1. Add Ticket Request (Enqueue)
2. Process Ticket (Dequeue)
3. Show Pending Tickets
4. Exit
Enter your choice (1-4): 1
Enter Ticket ID: 21222
Ticket Request Added: 21222

======= Ticketing System =======
1. Add Ticket Request (Enqueue)
2. Process Ticket (Dequeue)
3. Show Pending Tickets
4. Exit
Enter your choice (1-4): 2
Ticket Processed: 1212

======= Ticketing System =======
1. Add Ticket Request (Enqueue)
2. Process Ticket (Dequeue)
3. Show Pending Tickets
4. Exit
Enter your choice (1-4): 3
Pending Tickets: 21222

======= Ticketing System =======
1. Add Ticket Request (Enqueue)
2. Process Ticket (Dequeue)
3. Show Pending Tickets
4. Exit
Enter your choice (1-4): 4
Exiting Ticketing System.
sourabhsuman@Sourabhs-MacBook-Air DSA %
```