K.R. MANGALAM UNIVERSITY
THE COMPLETE WORLD OF EDUCATION

School of Engineering and Technology

Data Structure Report File

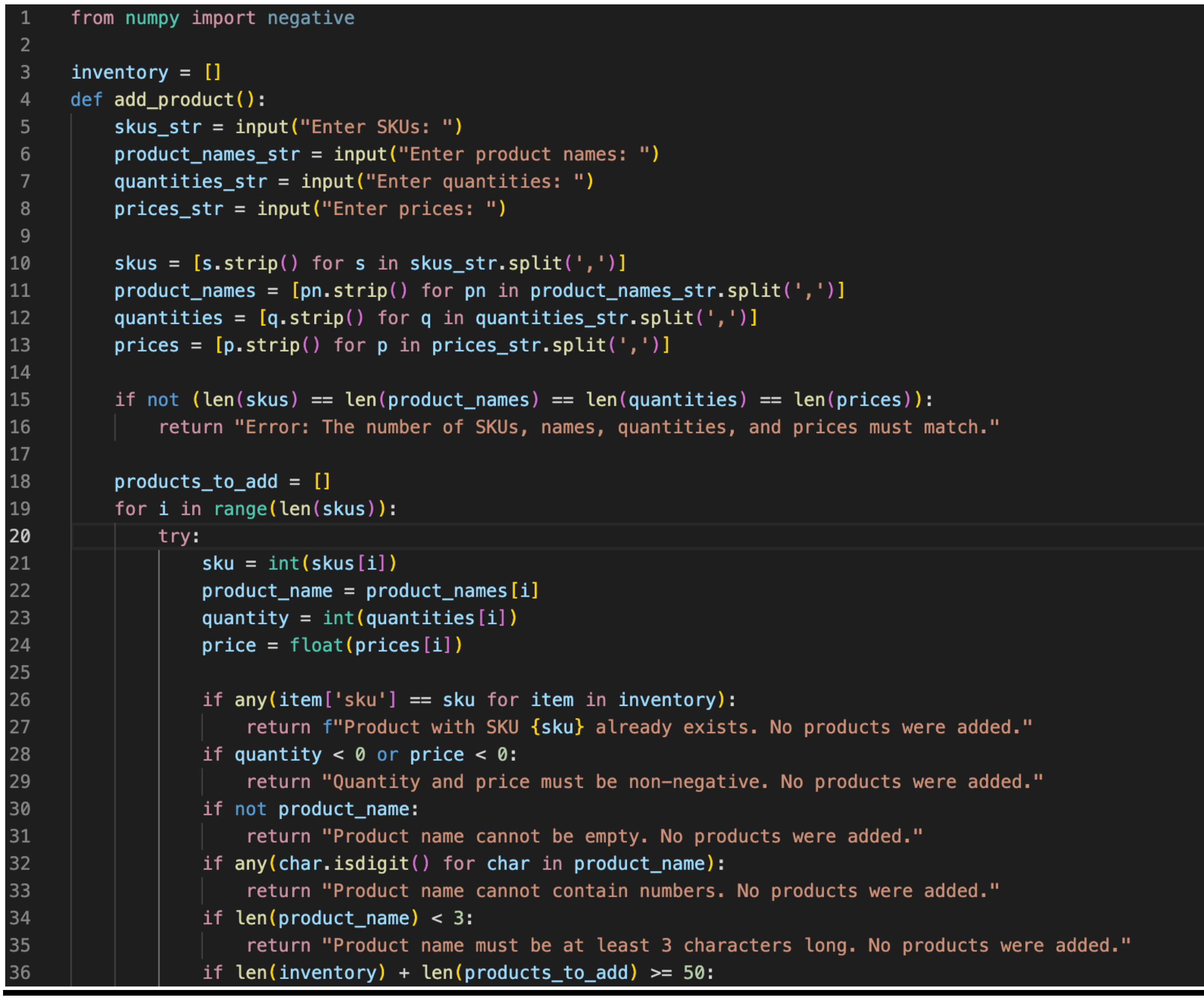
B.Tech CSE (Data Science)

Prepared By:
Vidit Bansal
2401420018

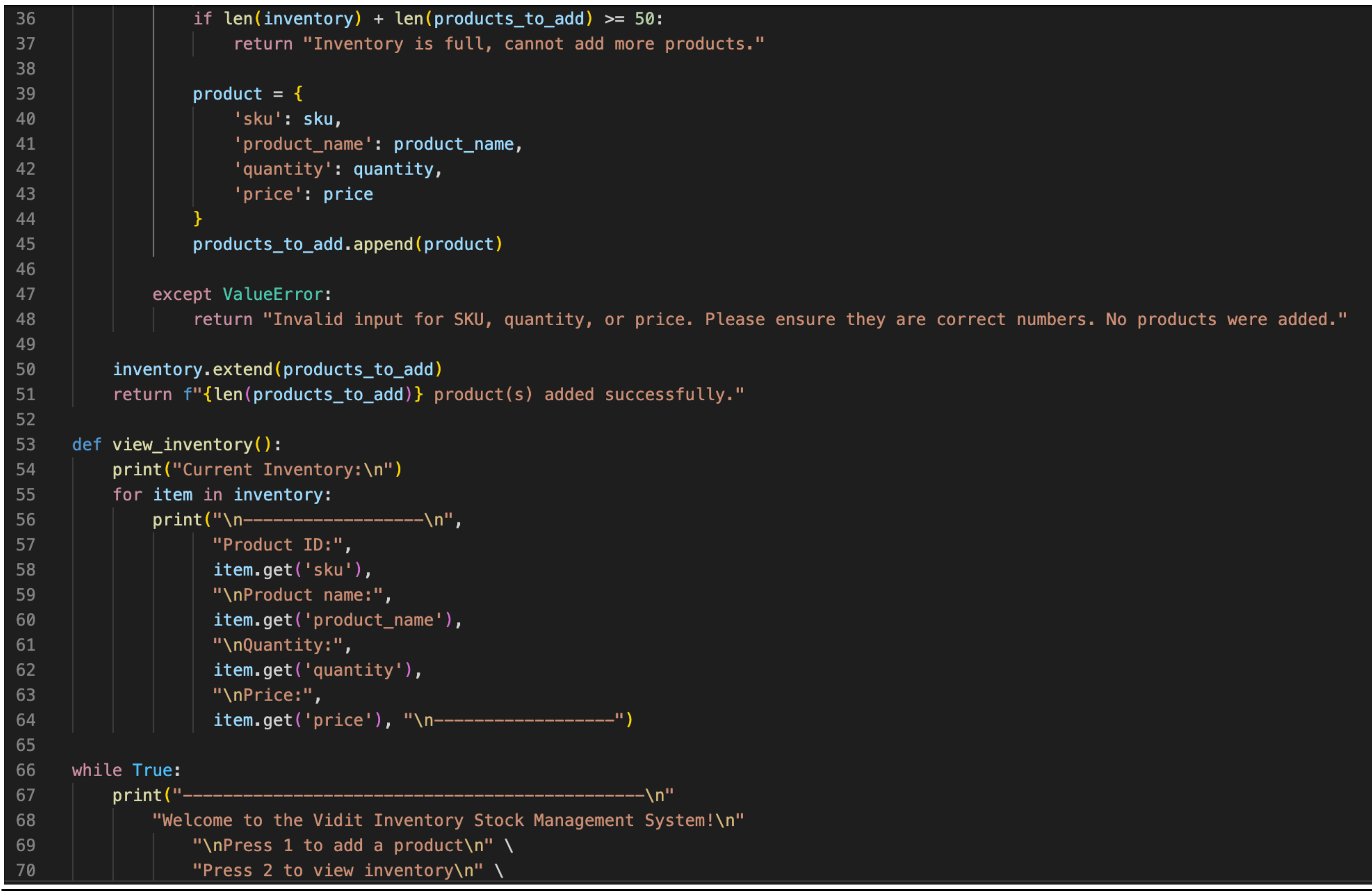
Submitted to:
Dr. Swati Gupta

1. Inventory Management System

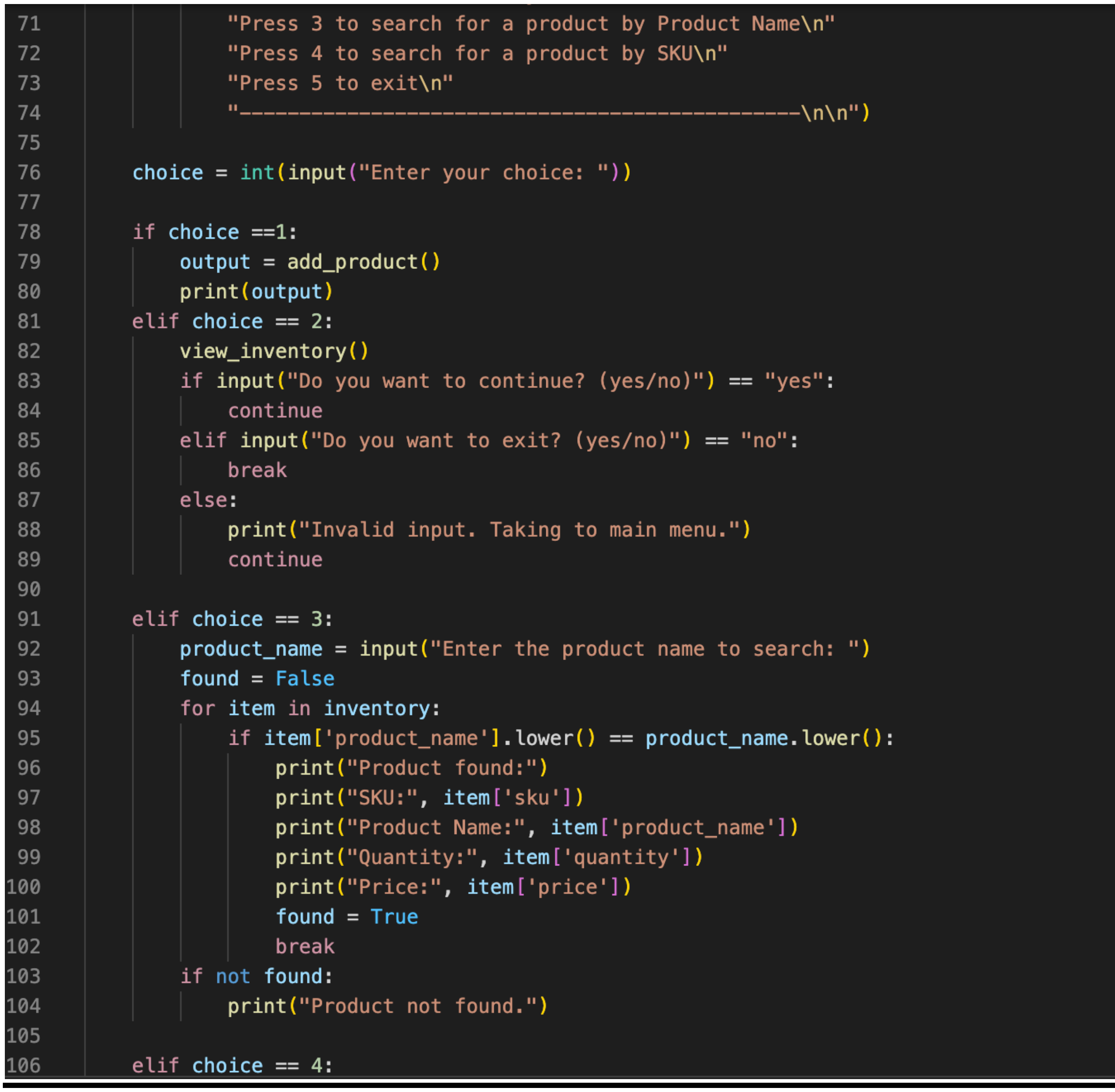
Input -



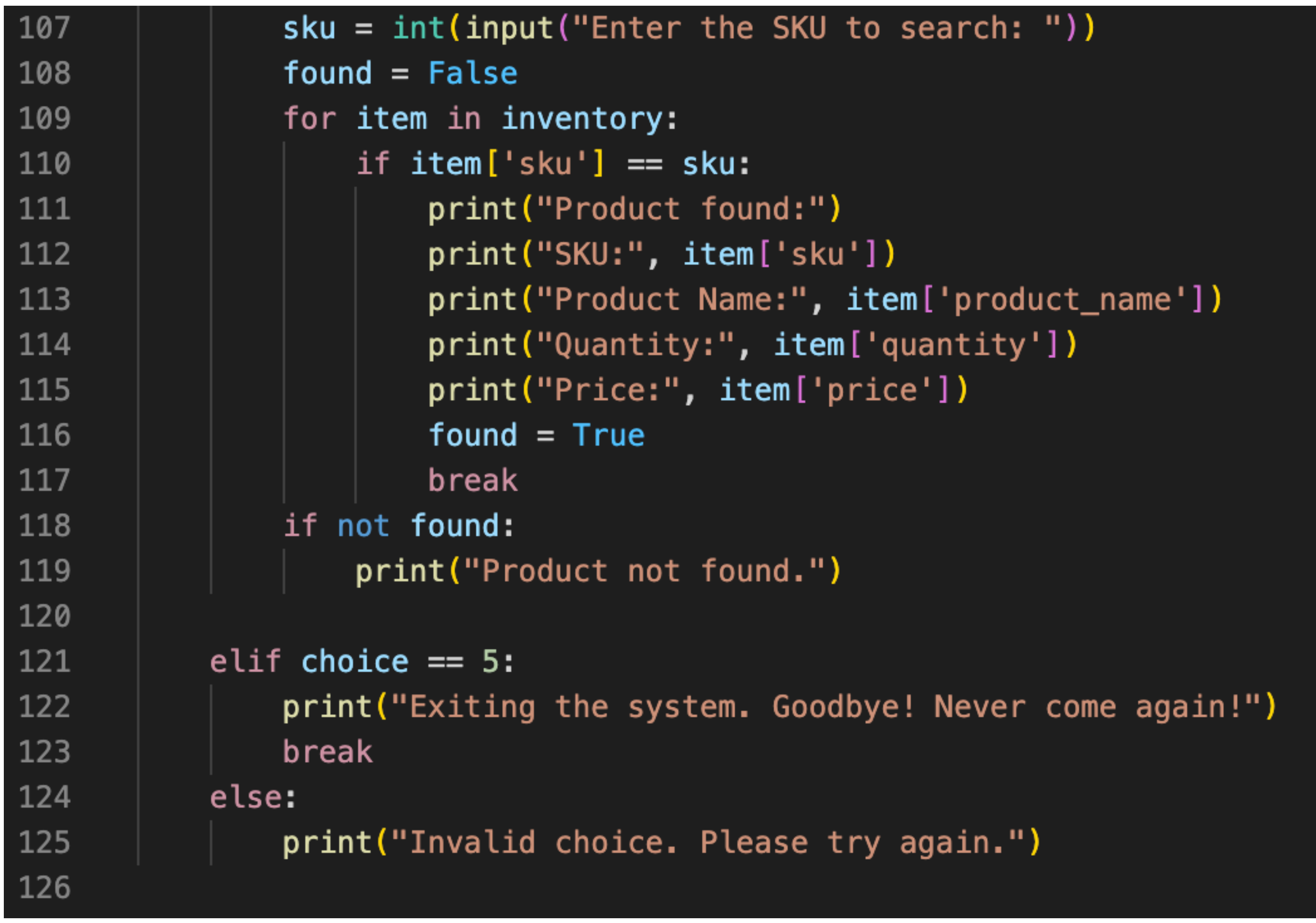
```python
from numpy import negative

inventory = []
def add_product():
    skus_str = input("Enter SKUs: ")
    product_names_str = input("Enter product names: ")
    quantities_str = input("Enter quantities: ")
    prices_str = input("Enter prices: ")

    skus = [s.strip() for s in skus_str.split(',')]
    product_names = [pn.strip() for pn in product_names_str.split(',')]
    quantities = [q.strip() for q in quantities_str.split(',')]
    prices = [p.strip() for p in prices_str.split(',')]

    if not (len(skus) == len(product_names) == len(quantities) == len(prices)):
        return "Error: The number of SKUs, names, quantities, and prices must match."

    products_to_add = []
    for i in range(len(skus)):
        try:
            sku = int(skus[i])
            product_name = product_names[i]
            quantity = int(quantities[i])
            price = float(prices[i])

            if any(item['sku'] == sku for item in inventory):
                return f"Product with SKU {sku} already exists. No products were added."
            if quantity < 0 or price < 0:
                return "Quantity and price must be non-negative. No products were added."
            if not product_name:
                return "Product name cannot be empty. No products were added."
            if any(char.isdigit() for char in product_name):
                return "Product name cannot contain numbers. No products were added."
            if len(product_name) < 3:
                return "Product name must be at least 3 characters long. No products were added."
            if len(inventory) + len(products_to_add) >= 50:
```



```
            if len(inventory) + len(products_to_add) >= 50:
                return "Inventory is full, cannot add more products."

            product = {
                'sku': sku,
                'product_name': product_name,
                'quantity': quantity,
                'price': price
            }
            products_to_add.append(product)

        except ValueError:
            return "Invalid input for SKU, quantity, or price. Please ensure they are correct numbers. No products were added."

    inventory.extend(products_to_add)
    return f"{len(products_to_add)} product(s) added successfully."

def view_inventory():
    print("Current Inventory:\n")
    for item in inventory:
        print("\n-------------------\n",
              "Product ID:",
              item.get('sku'),
              "\nProduct name:",
              item.get('product_name'),
              "\nQuantity:",
              item.get('quantity'),
              "\nPrice:",
              item.get('price'), "\n-------------------")

while True:
    print("----------------------------------------------\n"
        "Welcome to the Vidit Inventory Stock Management System!\n"
            "\nPress 1 to add a product\n" \
            "Press 2 to view inventory\n" \
```

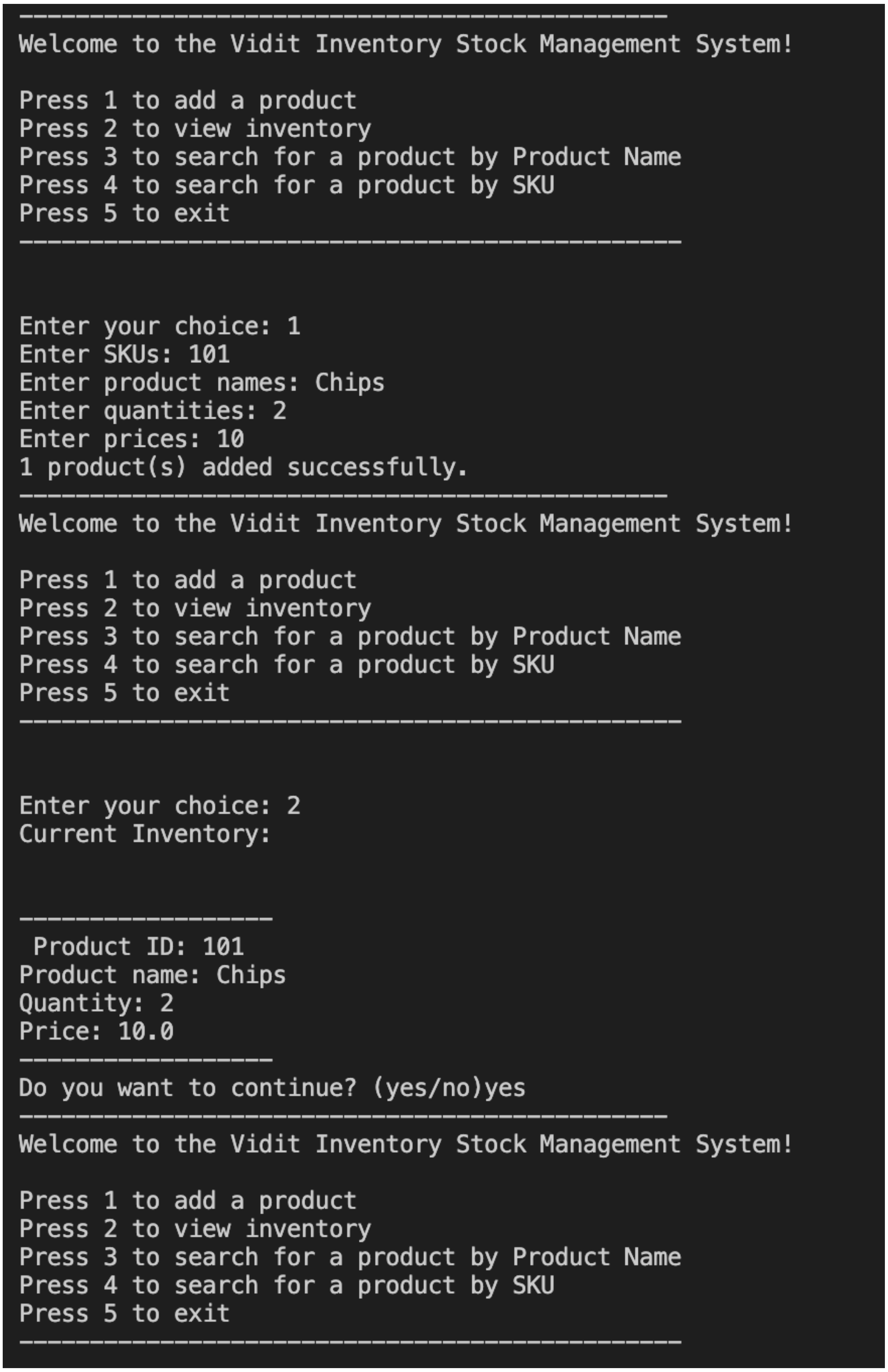


```python
            "Press 3 to search for a product by Product Name\n"
            "Press 4 to search for a product by SKU\n"
            "Press 5 to exit\n"
            "-----------------------------------------------\n\n")

    choice = int(input("Enter your choice: "))

    if choice ==1:
        output = add_product()
        print(output)
    elif choice == 2:
        view_inventory()
        if input("Do you want to continue? (yes/no)") == "yes":
            continue
        elif input("Do you want to exit? (yes/no)") == "no":
            break
        else:
            print("Invalid input. Taking to main menu.")
            continue

    elif choice == 3:
        product_name = input("Enter the product name to search: ")
        found = False
        for item in inventory:
            if item['product_name'].lower() == product_name.lower():
                print("Product found:")
                print("SKU:", item['sku'])
                print("Product Name:", item['product_name'])
                print("Quantity:", item['quantity'])
                print("Price:", item['price'])
                found = True
                break
        if not found:
            print("Product not found.")

    elif choice == 4:
```

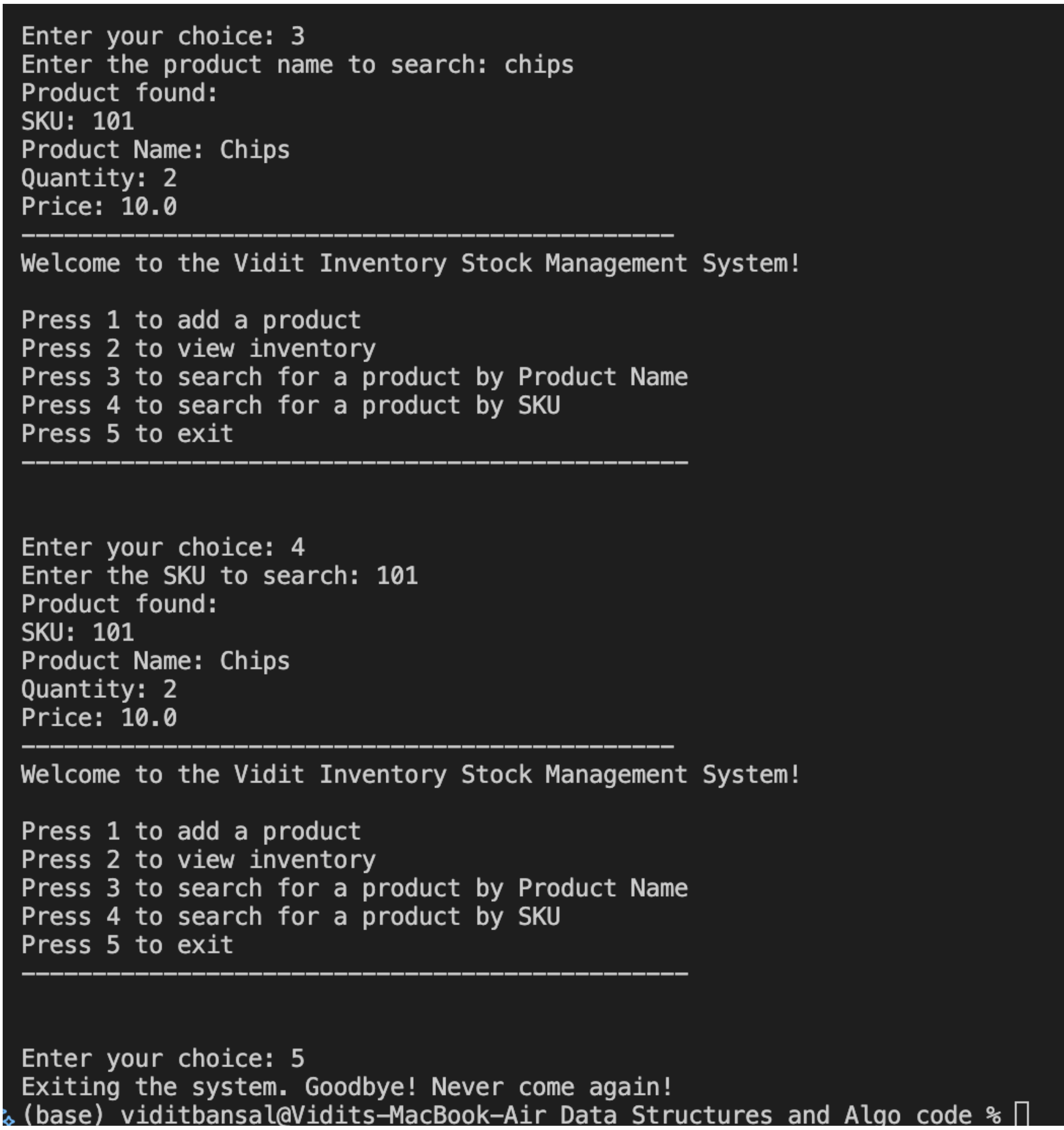


```python
        sku = int(input("Enter the SKU to search: "))
        found = False
        for item in inventory:
            if item['sku'] == sku:
                print("Product found:")
                print("SKU:", item['sku'])
                print("Product Name:", item['product_name'])
                print("Quantity:", item['quantity'])
                print("Price:", item['price'])
                found = True
                break
        if not found:
            print("Product not found.")

    elif choice == 5:
        print("Exiting the system. Goodbye! Never come again!")
        break
    else:
        print("Invalid choice. Please try again.")

```

Output –



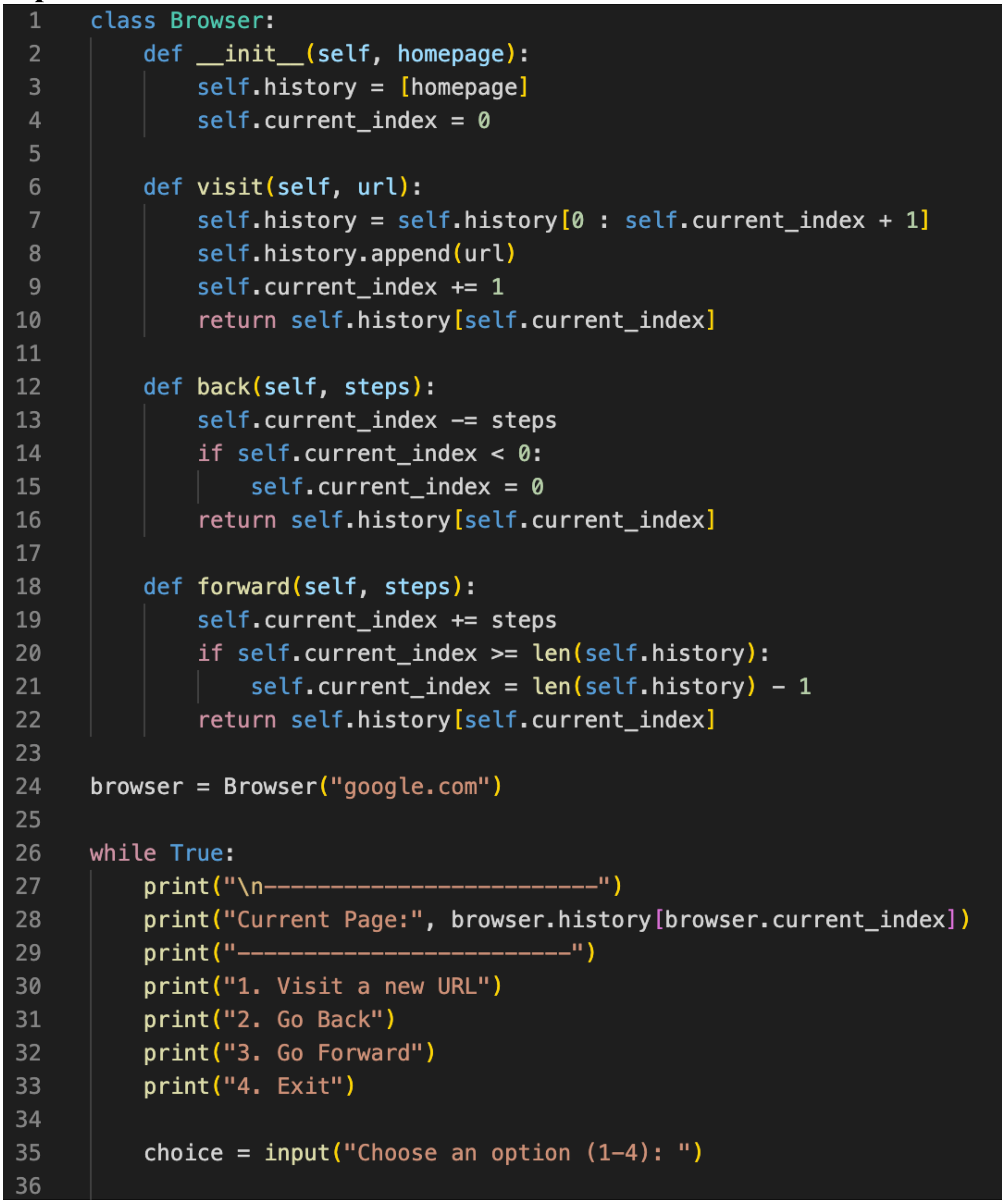
```
-----------------------------------------------
Welcome to the Vidit Inventory Stock Management System!

Press 1 to add a product
Press 2 to view inventory
Press 3 to search for a product by Product Name
Press 4 to search for a product by SKU
Press 5 to exit
------------------------------------------------



Enter your choice: 1
Enter SKUs: 101
Enter product names: Chips
Enter quantities: 2
Enter prices: 10
1 product(s) added successfully.
-----------------------------------------------
Welcome to the Vidit Inventory Stock Management System!

Press 1 to add a product
Press 2 to view inventory
Press 3 to search for a product by Product Name
Press 4 to search for a product by SKU
Press 5 to exit
------------------------------------------------



Enter your choice: 2
Current Inventory:



------------------
 Product ID: 101
Product name: Chips
Quantity: 2
Price: 10.0
------------------
Do you want to continue? (yes/no)yes
-----------------------------------------------
Welcome to the Vidit Inventory Stock Management System!

Press 1 to add a product
Press 2 to view inventory
Press 3 to search for a product by Product Name
Press 4 to search for a product by SKU
Press 5 to exit
------------------------------------------------
```



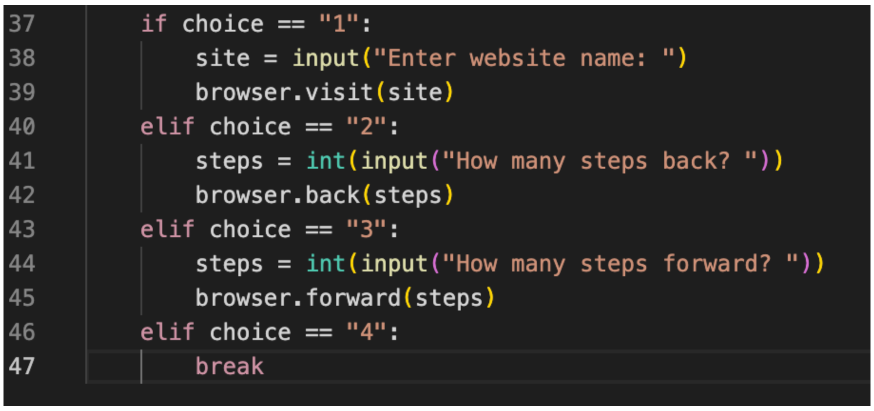
```
Enter your choice: 3
Enter the product name to search: chips
Product found:
SKU: 101
Product Name: Chips
Quantity: 2
Price: 10.0
---------------------------------------------
Welcome to the Vidit Inventory Stock Management System!

Press 1 to add a product
Press 2 to view inventory
Press 3 to search for a product by Product Name
Press 4 to search for a product by SKU
Press 5 to exit
----------------------------------------------


Enter your choice: 4
Enter the SKU to search: 101
Product found:
SKU: 101
Product Name: Chips
Quantity: 2
Price: 10.0
---------------------------------------------
Welcome to the Vidit Inventory Stock Management System!

Press 1 to add a product
Press 2 to view inventory
Press 3 to search for a product by Product Name
Press 4 to search for a product by SKU
Press 5 to exit
----------------------------------------------


Enter your choice: 5
Exiting the system. Goodbye! Never come again!
(base) viditbansal@Vidits-MacBook-Air Data Structures and Algo code %
```

2. Browsing History Navigation History

Input -

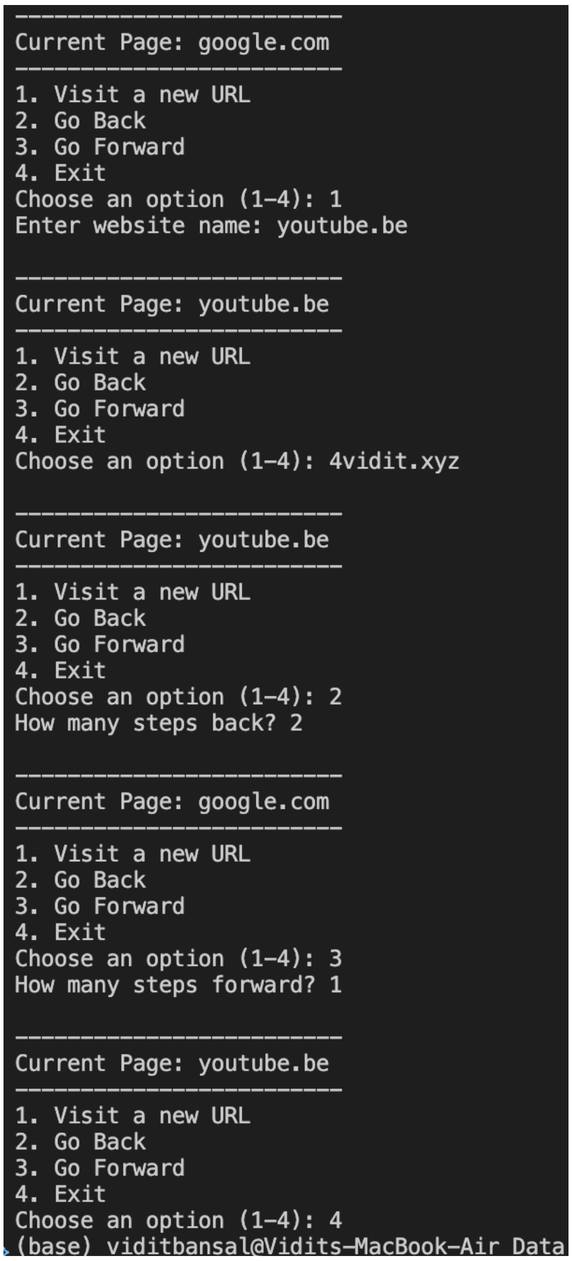


```python
class Browser:
    def __init__(self, homepage):
        self.history = [homepage]
        self.current_index = 0

    def visit(self, url):
        self.history = self.history[0 : self.current_index + 1]
        self.history.append(url)
        self.current_index += 1
        return self.history[self.current_index]

    def back(self, steps):
        self.current_index -= steps
        if self.current_index < 0:
            self.current_index = 0
        return self.history[self.current_index]

    def forward(self, steps):
        self.current_index += steps
        if self.current_index >= len(self.history):
            self.current_index = len(self.history) - 1
        return self.history[self.current_index]

browser = Browser("google.com")

while True:
    print("\n-------------------------")
    print("Current Page:", browser.history[browser.current_index])
    print("-------------------------")
    print("1. Visit a new URL")
    print("2. Go Back")
    print("3. Go Forward")
    print("4. Exit")

    choice = input("Choose an option (1-4): ")

```



```python
    if choice == "1":
        site = input("Enter website name: ")
        browser.visit(site)
    elif choice == "2":
        steps = int(input("How many steps back? "))
        browser.back(steps)
    elif choice == "3":
        steps = int(input("How many steps forward? "))
        browser.forward(steps)
    elif choice == "4":
        break
```

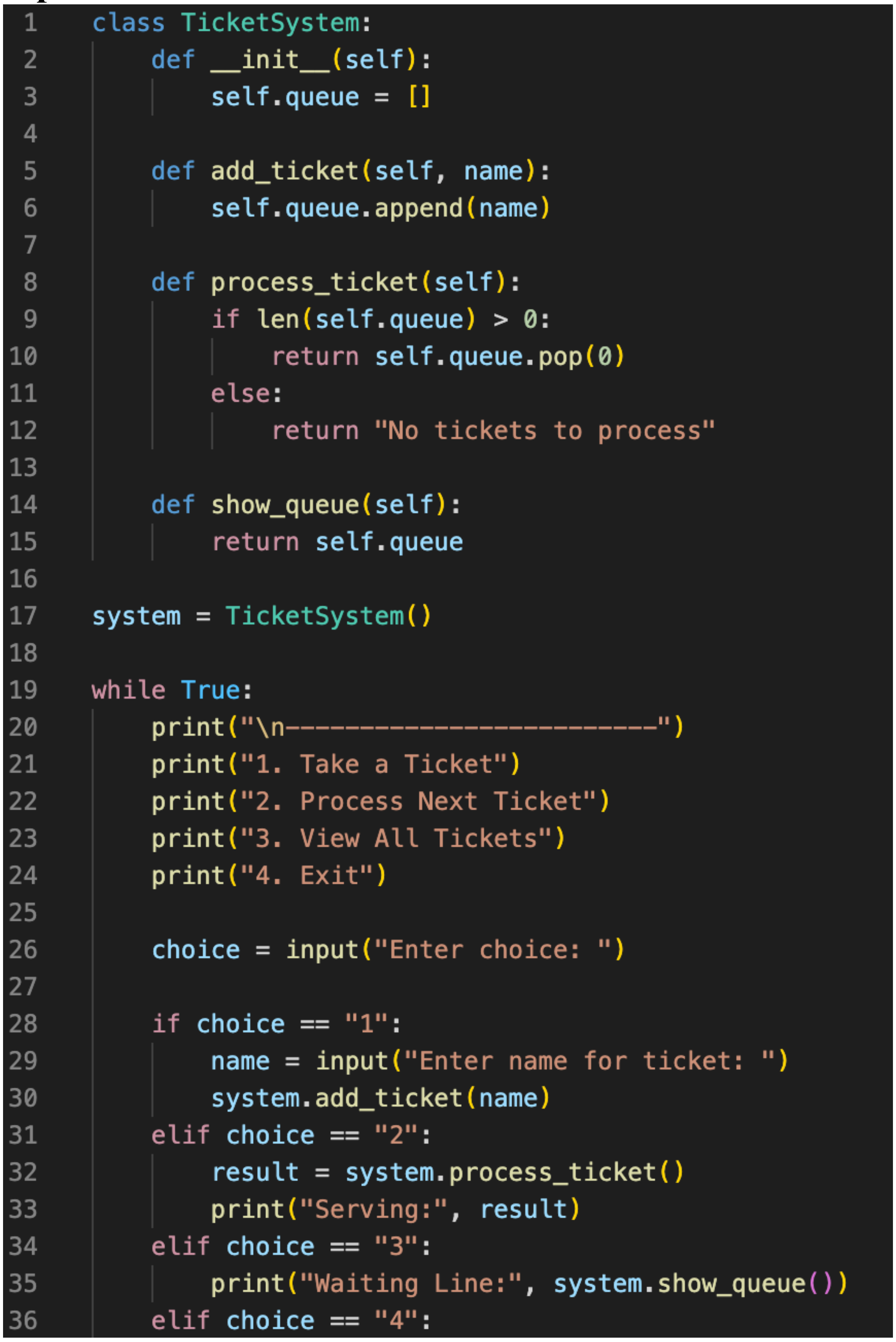
Output:



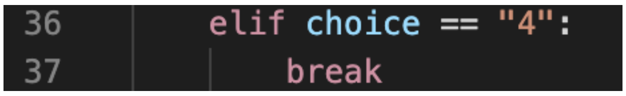
```
-------------------------
Current Page: google.com
-------------------------
1. Visit a new URL
2. Go Back
3. Go Forward
4. Exit
Choose an option (1-4): 1
Enter website name: youtube.be

-------------------------
Current Page: youtube.be
-------------------------
1. Visit a new URL
2. Go Back
3. Go Forward
4. Exit
Choose an option (1-4): 4vidit.xyz

-------------------------
Current Page: youtube.be
-------------------------
1. Visit a new URL
2. Go Back
3. Go Forward
4. Exit
Choose an option (1-4): 2
How many steps back? 2

-------------------------
Current Page: google.com
-------------------------
1. Visit a new URL
2. Go Back
3. Go Forward
4. Exit
Choose an option (1-4): 3
How many steps forward? 1

-------------------------
Current Page: youtube.be
-------------------------
1. Visit a new URL
2. Go Back
3. Go Forward
4. Exit
Choose an option (1-4): 4
(base) viditbansal@Vidits-MacBook-Air Data
```

3. Ticket Management System Using Linear Queue

Input -

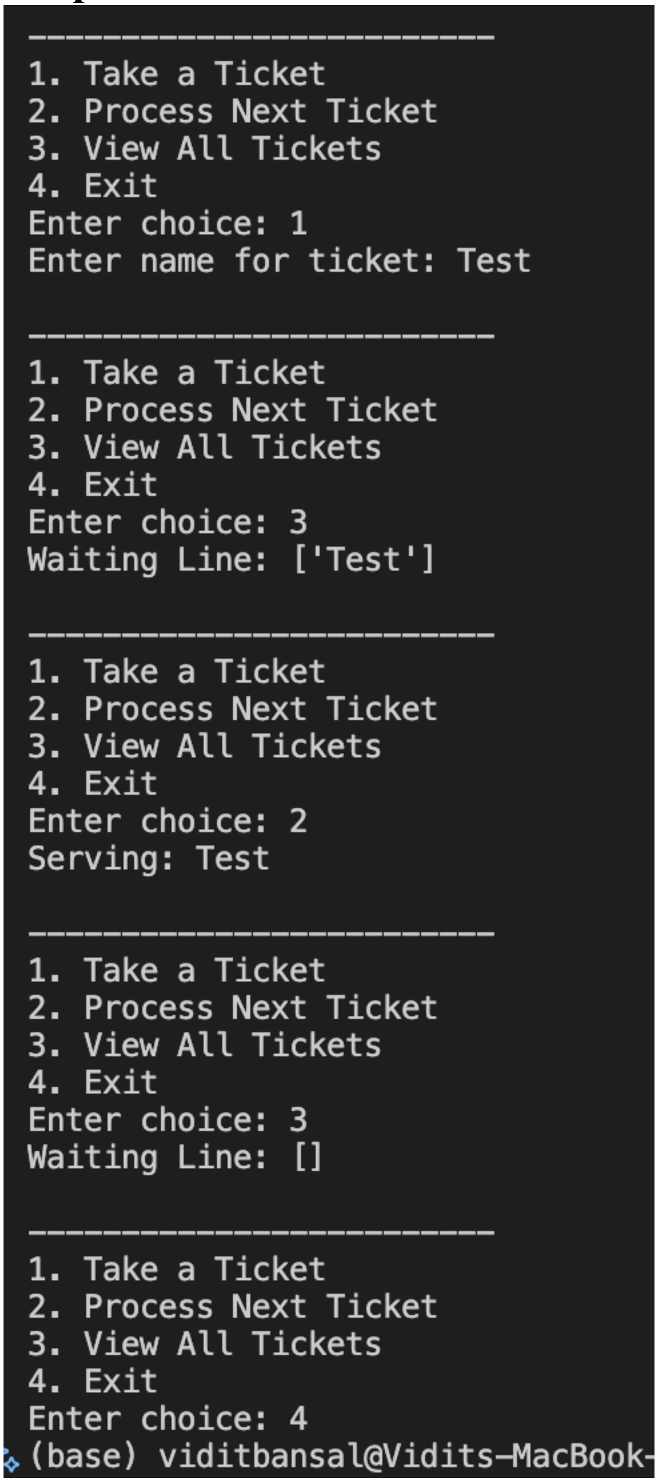


```python
class TicketSystem:
    def __init__(self):
        self.queue = []

    def add_ticket(self, name):
        self.queue.append(name)

    def process_ticket(self):
        if len(self.queue) > 0:
            return self.queue.pop(0)
        else:
            return "No tickets to process"

    def show_queue(self):
        return self.queue

system = TicketSystem()

while True:
    print("\n-------------------------")
    print("1. Take a Ticket")
    print("2. Process Next Ticket")
    print("3. View All Tickets")
    print("4. Exit")

    choice = input("Enter choice: ")

    if choice == "1":
        name = input("Enter name for ticket: ")
        system.add_ticket(name)
    elif choice == "2":
        result = system.process_ticket()
        print("Serving:", result)
    elif choice == "3":
        print("Waiting Line:", system.show_queue())
    elif choice == "4":
```



```python
        elif choice == "4":
            break
```

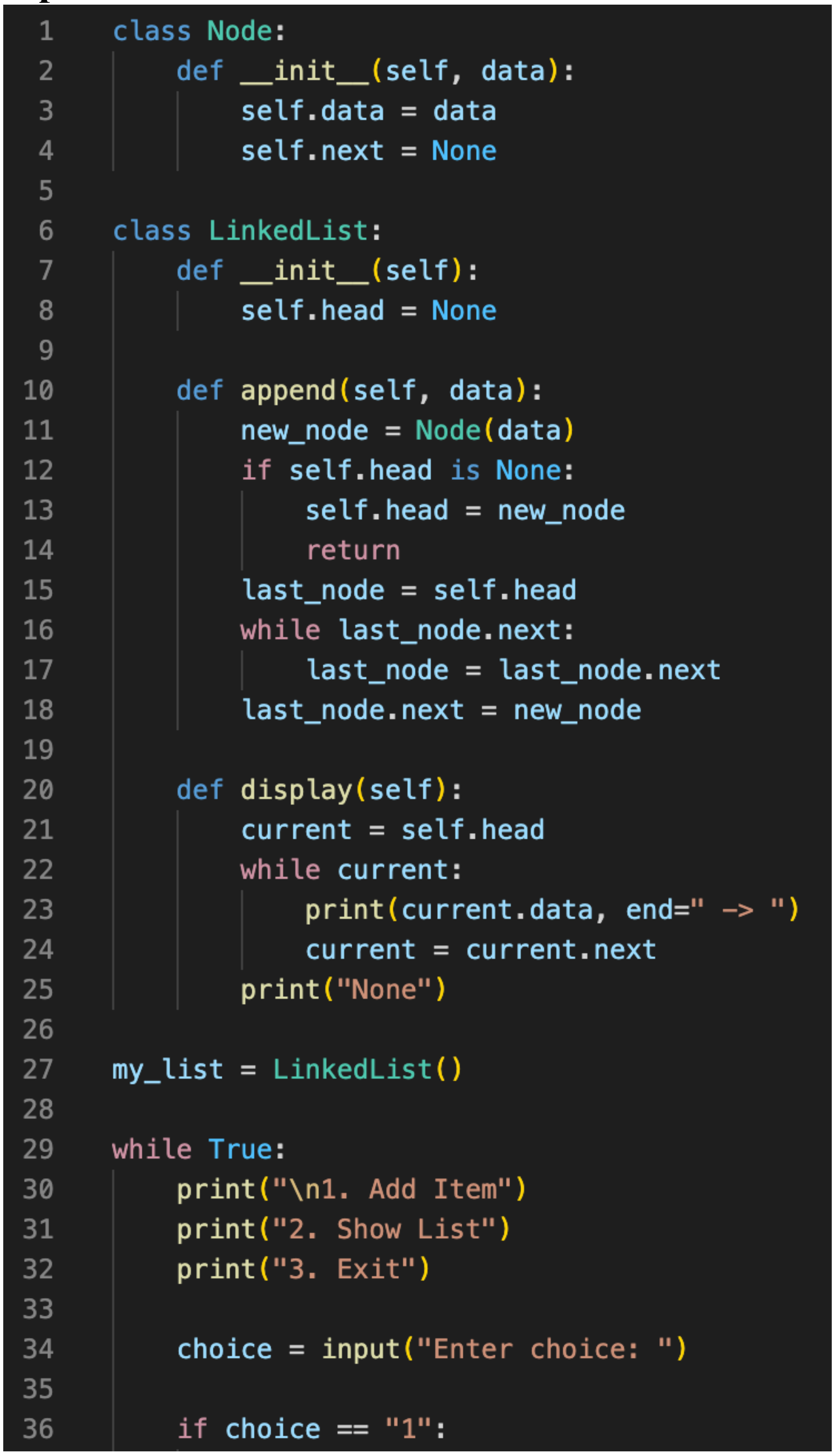
Output:



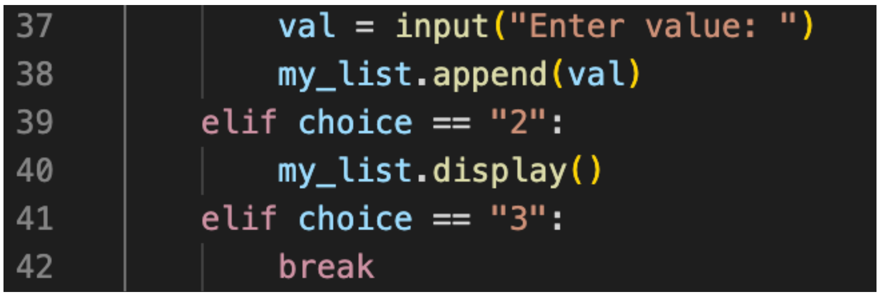
```
-------------------------
1. Take a Ticket
2. Process Next Ticket
3. View All Tickets
4. Exit
Enter choice: 1
Enter name for ticket: Test

-------------------------
1. Take a Ticket
2. Process Next Ticket
3. View All Tickets
4. Exit
Enter choice: 3
Waiting Line: ['Test']

-------------------------
1. Take a Ticket
2. Process Next Ticket
3. View All Tickets
4. Exit
Enter choice: 2
Serving: Test

-------------------------
1. Take a Ticket
2. Process Next Ticket
3. View All Tickets
4. Exit
Enter choice: 3
Waiting Line: []

-------------------------
1. Take a Ticket
2. Process Next Ticket
3. View All Tickets
4. Exit
Enter choice: 4
(base) viditbansal@Vidits-MacBook-
```

4. Singly Linked List

Input -

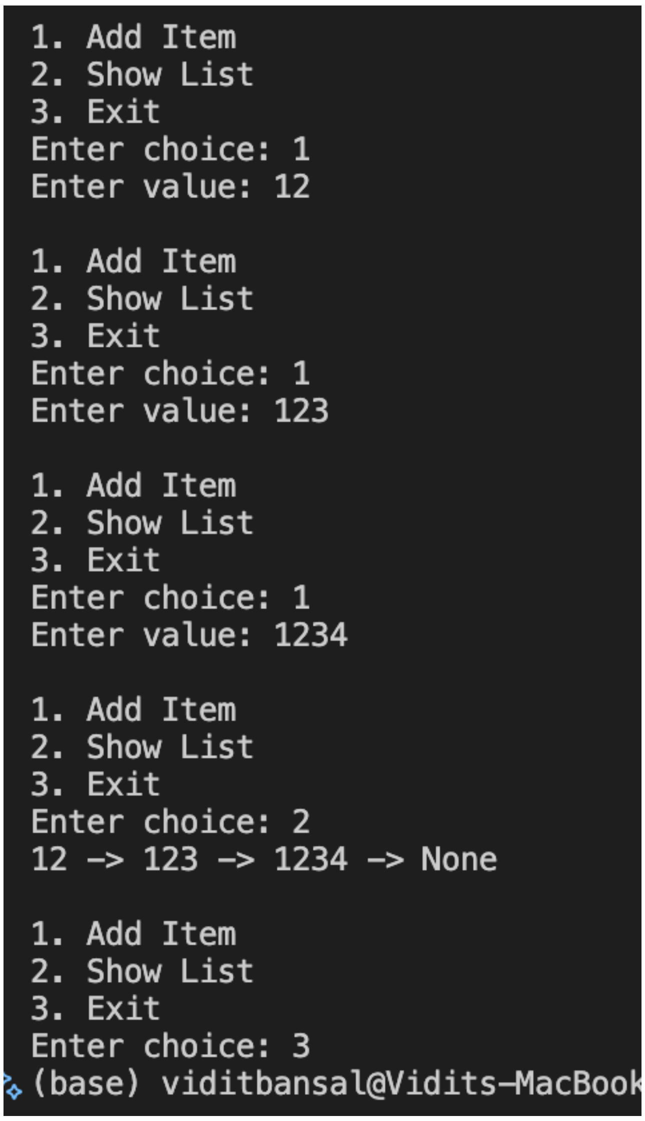


```python
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    def append(self, data):
        new_node = Node(data)
        if self.head is None:
            self.head = new_node
            return
        last_node = self.head
        while last_node.next:
            last_node = last_node.next
        last_node.next = new_node

    def display(self):
        current = self.head
        while current:
            print(current.data, end=" -> ")
            current = current.next
        print("None")

my_list = LinkedList()

while True:
    print("\n1. Add Item")
    print("2. Show List")
    print("3. Exit")

    choice = input("Enter choice: ")

    if choice == "1":
```



```python
            val = input("Enter value: ")
            my_list.append(val)
        elif choice == "2":
            my_list.display()
        elif choice == "3":
            break
```

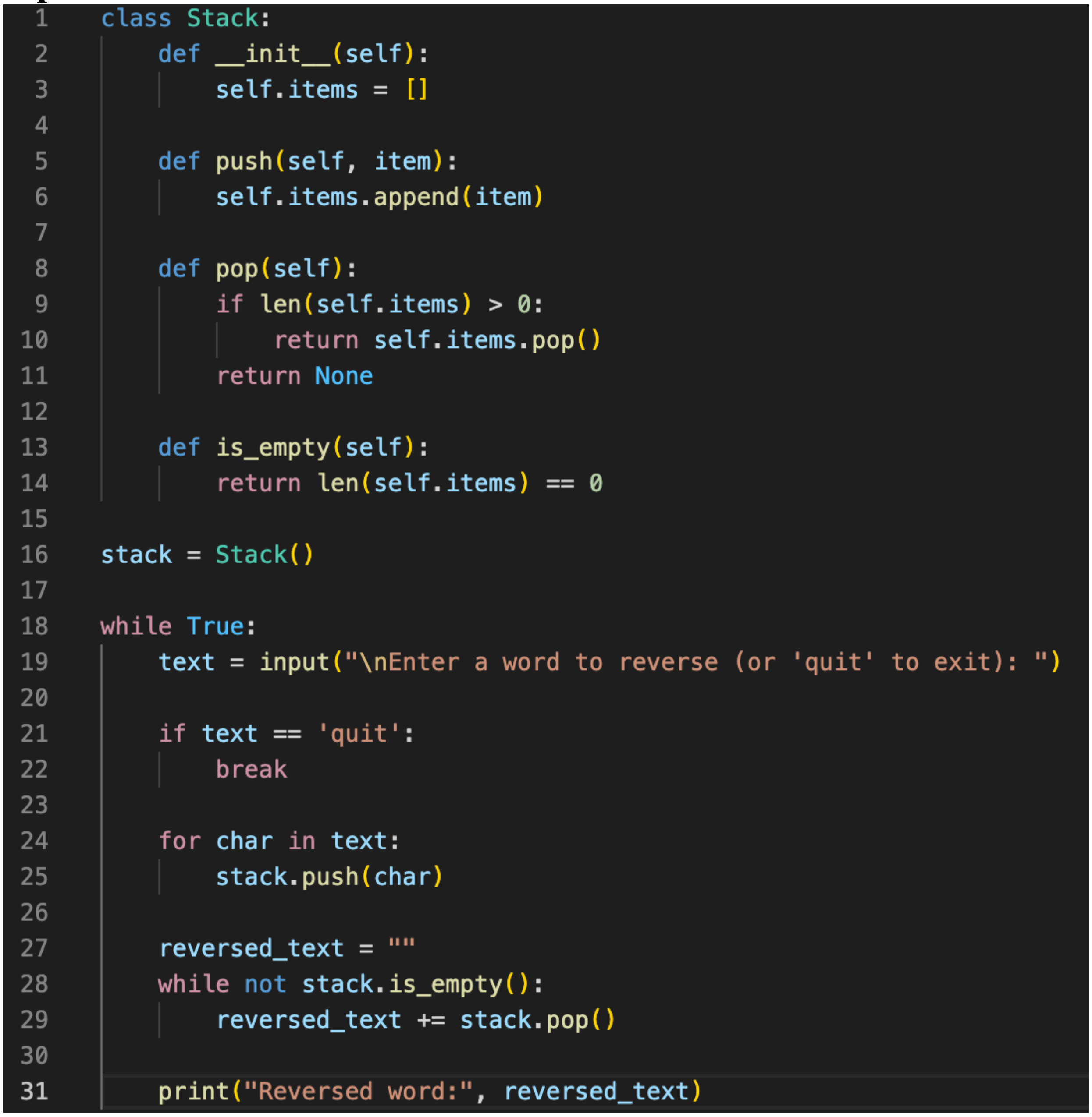
Output:



```
1. Add Item
2. Show List
3. Exit
Enter choice: 1
Enter value: 12

1. Add Item
2. Show List
3. Exit
Enter choice: 1
Enter value: 123

1. Add Item
2. Show List
3. Exit
Enter choice: 1
Enter value: 1234

1. Add Item
2. Show List
3. Exit
Enter choice: 2
12 -> 123 -> 1234 -> None

1. Add Item
2. Show List
3. Exit
Enter choice: 3
(base) viditbansal@Vidits-MacBook
```

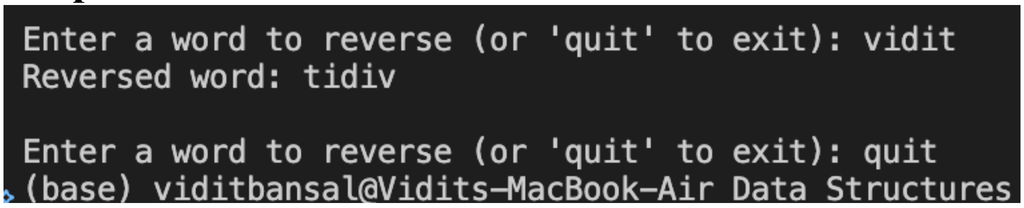
5. Reverse of String Using Stack

Input -



```python
class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if len(self.items) > 0:
            return self.items.pop()
        return None

    def is_empty(self):
        return len(self.items) == 0

stack = Stack()

while True:
    text = input("\nEnter a word to reverse (or 'quit' to exit): ")

    if text == 'quit':
        break

    for char in text:
        stack.push(char)

    reversed_text = ""
    while not stack.is_empty():
        reversed_text += stack.pop()

    print("Reversed word:", reversed_text)
```

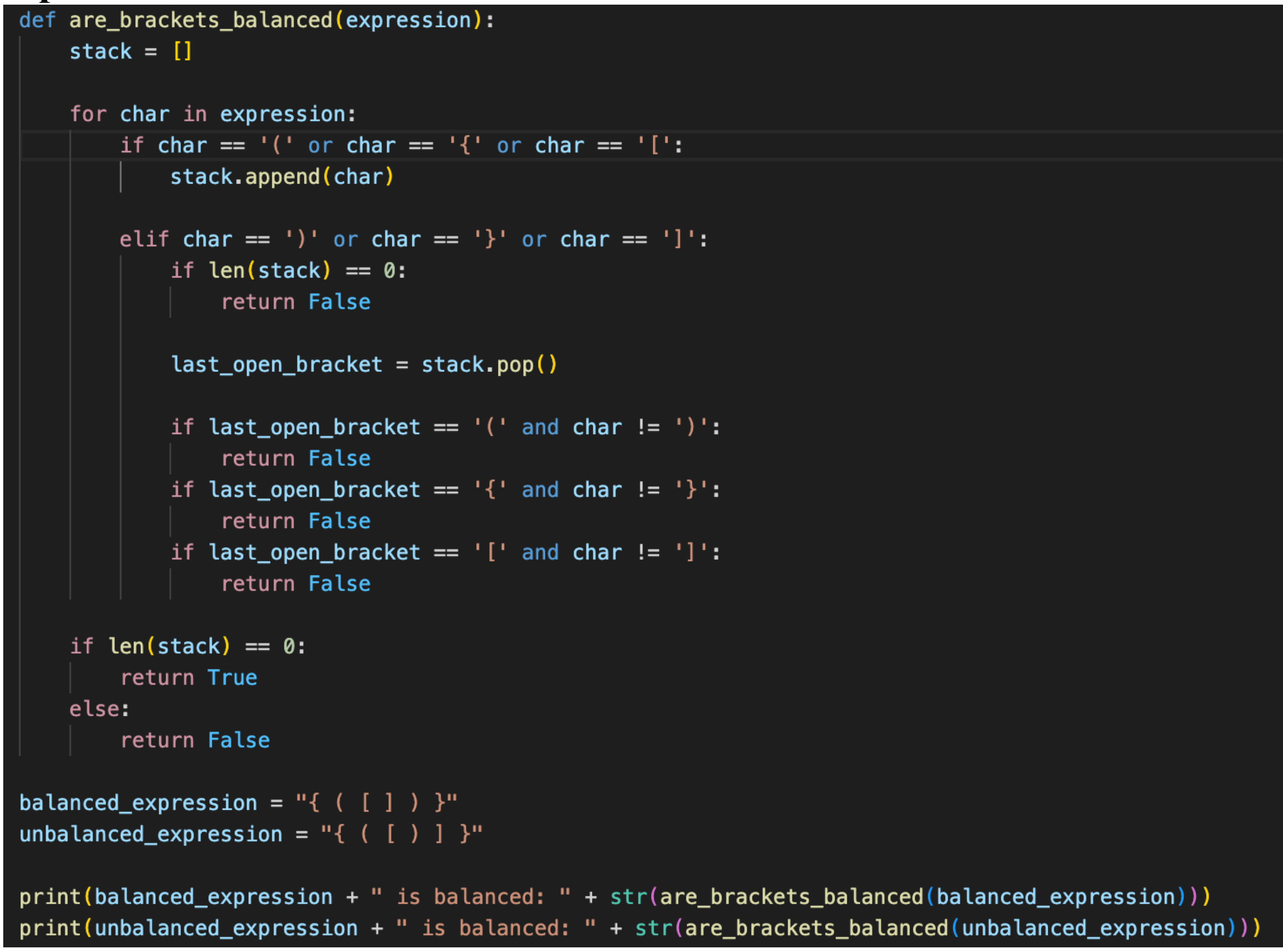
Output:



```
Enter a word to reverse (or 'quit' to exit): vidit
Reversed word: tidiv

Enter a word to reverse (or 'quit' to exit): quit
(base) viditbansal@Vidits-MacBook-Air Data Structures
```

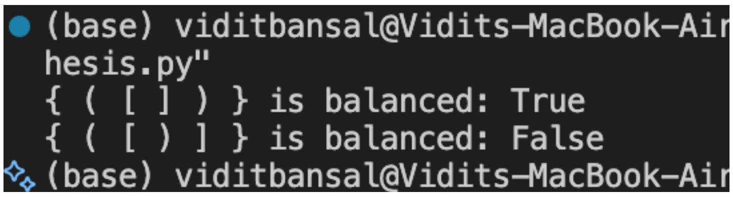
6. Balanced Parenthesis Using Stack

Input -



```python
def are_brackets_balanced(expression):
    stack = []

    for char in expression:
        if char == '(' or char == '{' or char == '[':
            stack.append(char)

        elif char == ')' or char == '}' or char == ']':
            if len(stack) == 0:
                return False

            last_open_bracket = stack.pop()

            if last_open_bracket == '(' and char != ')':
                return False
            if last_open_bracket == '{' and char != '}':
                return False
            if last_open_bracket == '[' and char != ']':
                return False

    if len(stack) == 0:
        return True
    else:
        return False

balanced_expression = "{ ( [ ] ) }"
unbalanced_expression = "{ ( [ ) ] }"

print(balanced_expression + " is balanced: " + str(are_brackets_balanced(balanced_expression)))
print(unbalanced_expression + " is balanced: " + str(are_brackets_balanced(unbalanced_expression)))
```

Output:



```
(base) viditbansal@Vidits-MacBook-Air
hesis.py"
{ ( [ ] ) } is balanced: True
{ ( [ ) ] } is balanced: False
(base) viditbansal@Vidits-MacBook-Air
```