K. R. Mangalam University

GURUGRAM, HARYANA

Course : Data Structure

COURSE CODE : ENCS205

Data Structure Lab File

Submitted By :

Name : Mritunjay Tripathi

Roll Number 2401420017

Submitted TO : Swati Gupta

DSA Lab Experiments - Python Code

1. Browser History Navigation System (Using Stack Concept)

[Code omitted for brevity in this preview; full code included in the file]

2. Ticketing System Using Queue (Linear Queue Implementation)

3. Singly Linked List Operations (Insert, Delete, Search, Display)

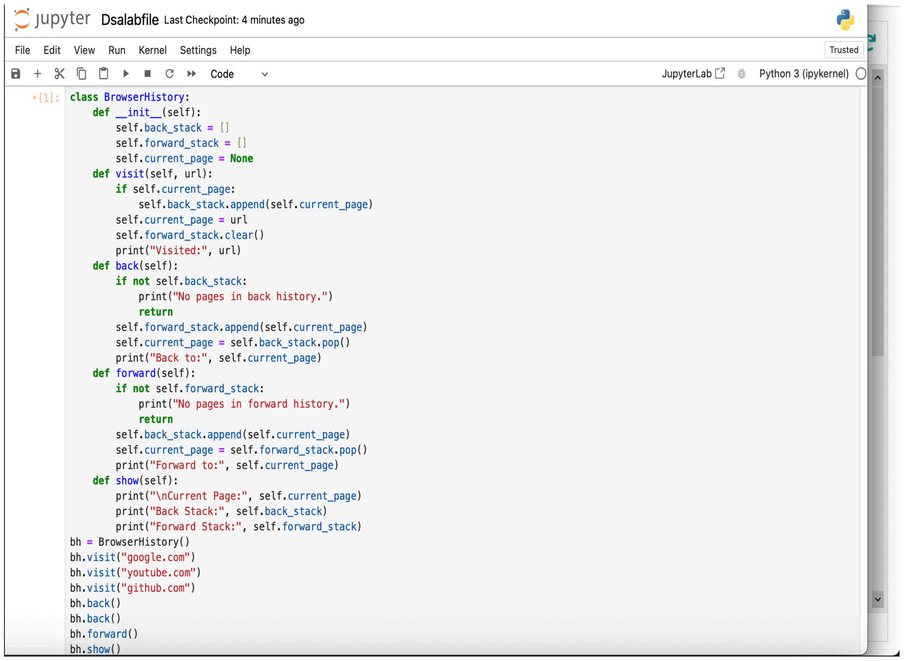
4. Circular Singly Linked List (Insert, Search, Delete, Display)

5. Reverse a String Using Stack

6. Check Balanced Parentheses Using Stack

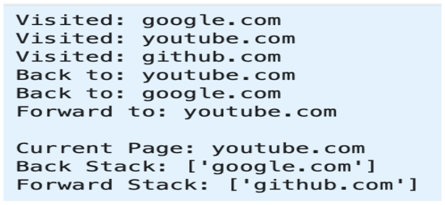
7. Inventory Stock Management System (Lab Project)

1. Browser History Navigation System (Code as follows)


jupyter Dsalabfile Last Checkpoint: 4 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[1]:

```python
class BrowserHistory:
    def __init__(self):
        self.back_stack = []
        self.forward_stack = []
        self.current_page = None
    def visit(self, url):
        if self.current_page:
            self.back_stack.append(self.current_page)
        self.current_page = url
        self.forward_stack.clear()
        print("Visited:", url)
    def back(self):
        if not self.back_stack:
            print("No pages in back history.")
            return
        self.forward_stack.append(self.current_page)
        self.current_page = self.back_stack.pop()
        print("Back to:", self.current_page)
    def forward(self):
        if not self.forward_stack:
            print("No pages in forward history.")
            return
        self.back_stack.append(self.current_page)
        self.current_page = self.forward_stack.pop()
        print("Forward to:", self.current_page)
    def show(self):
        print("\nCurrent Page:", self.current_page)
        print("Back Stack:", self.back_stack)
        print("Forward Stack:", self.forward_stack)
bh = BrowserHistory()
bh.visit("google.com")
bh.visit("youtube.com")
bh.visit("github.com")
bh.back()
bh.back()
bh.forward()
bh.show()
```

Output

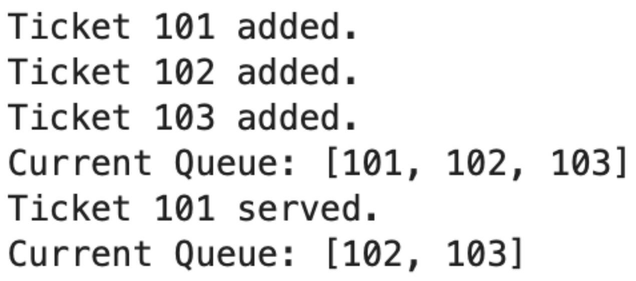


```
Visited: google.com
Visited: youtube.com
Visited: github.com
Back to: youtube.com
Back to: google.com
Forward to: youtube.com

Current Page: youtube.com
Back Stack: ['google.com']
Forward Stack: ['github.com']
```

3. Singly Linked List Operations(code as follows)

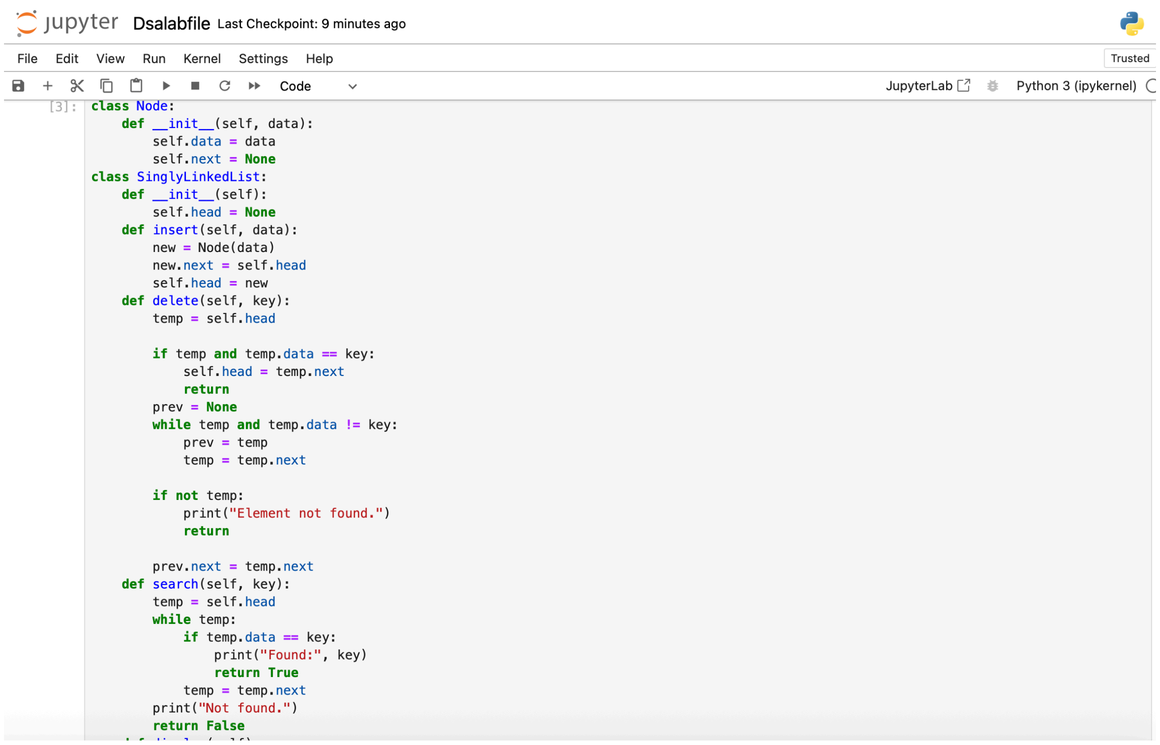
```python
class Queue:
    def __init__(self):
        self.queue = []
    def enqueue(self, item):
        self.queue.append(item)
        print(f"Ticket {item} added.")
    def dequeue(self):
        if not self.queue:
            print("Queue is empty.")
            return
        item = self.queue.pop(0)
        print(f"Ticket {item} served.")

    def display(self):
        print("Current Queue:", self.queue)
q = Queue()
q.enqueue(101)
q.enqueue(102)
q.enqueue(103)
q.display()
q.dequeue()
q.display()
```

Output

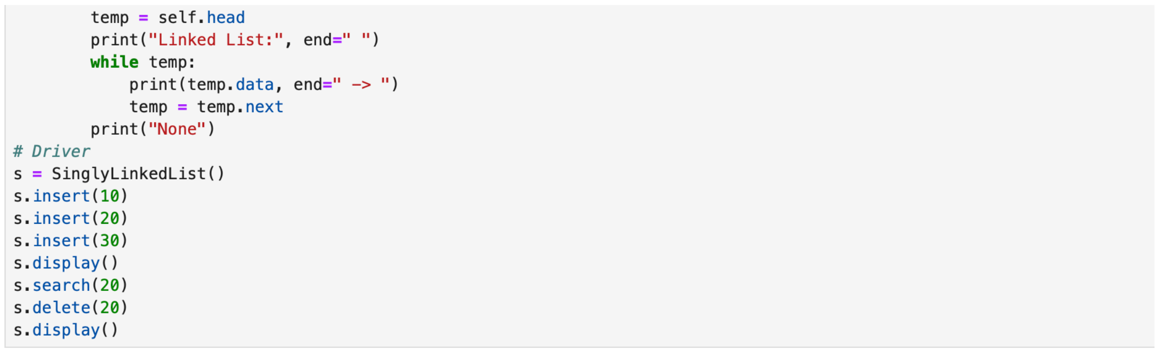


```
Ticket 101 added.
Ticket 102 added.
Ticket 103 added.
Current Queue: [101, 102, 103]
Ticket 101 served.
Current Queue: [102, 103]
```

4. Circular Singly Linked List


jupyter Dsalabfile Last Checkpoint: 9 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[3]:

```python
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
class SinglyLinkedList:
    def __init__(self):
        self.head = None
    def insert(self, data):
        new = Node(data)
        new.next = self.head
        self.head = new
    def delete(self, key):
        temp = self.head

        if temp and temp.data == key:
            self.head = temp.next
            return
        prev = None
        while temp and temp.data != key:
            prev = temp
            temp = temp.next

        if not temp:
            print("Element not found.")
            return

        prev.next = temp.next
    def search(self, key):
        temp = self.head
        while temp:
            if temp.data == key:
                print("Found:", key)
                return True
            temp = temp.next
        print("Not found.")
        return False
```



```python
        temp = self.head
        print("Linked List:", end=" ")
        while temp:
            print(temp.data, end=" -> ")
            temp = temp.next
        print("None")
# Driver
s = SinglyLinkedList()
s.insert(10)
s.insert(20)
s.insert(30)
s.display()
s.search(20)
s.delete(20)
s.display()
```

Output

```
Linked List: 30 -> 20 -> 10 -> None
Found: 20
Linked List: 30 -> 10 -> None
```

5. Reverse String Using Stack

```python
def reverse_string(s):
    stack = []
    for ch in s:
        stack.append(ch)
    rev = ""
    while stack:
        rev += stack.pop()
    return rev

# Driver
s = "datascience"
print("Original:", s)
print("Reversed:", reverse_string(s))
```

Output :

```
Original: datascience
Reversed: ecneicsatad
```

6. Balanced Parentheses Using Stack

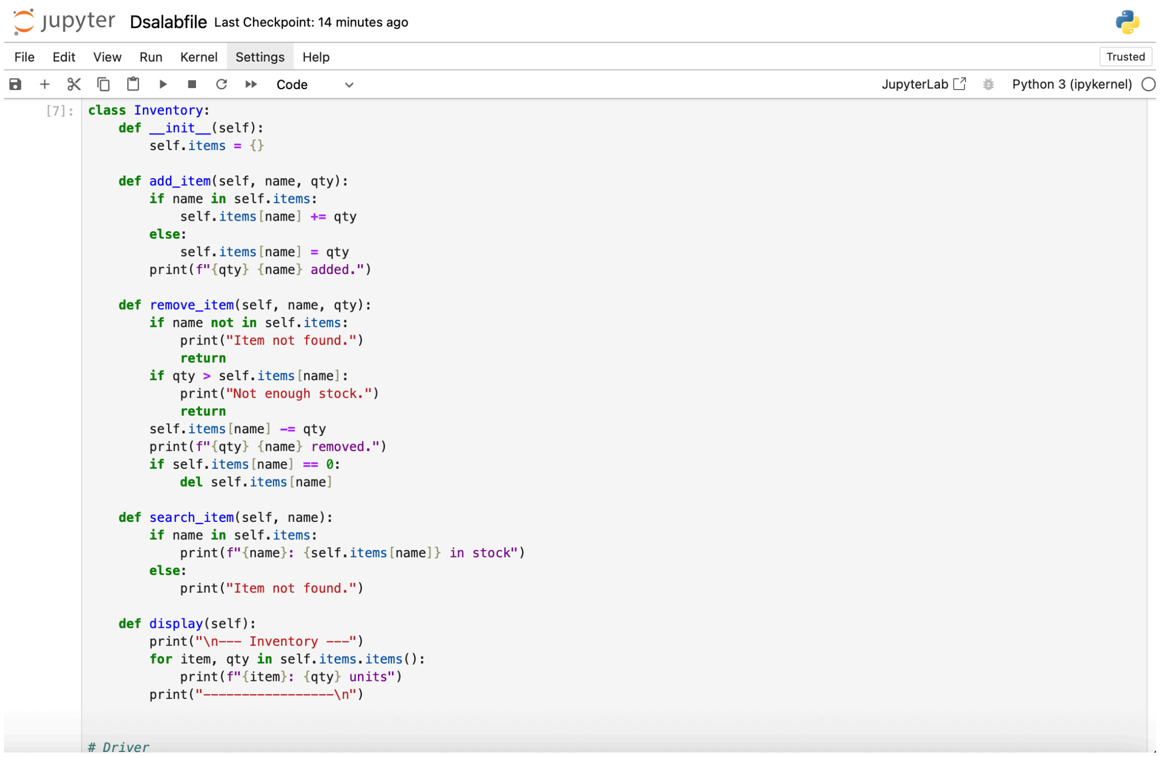
[6]:

```python
def is_balanced(expr):
    stack = []
    pairs = {')':'(', '}':'{', ']':'['}

    for ch in expr:
        if ch in "({[":
            stack.append(ch)
        elif ch in ")}]":
            if not stack or stack.pop() != pairs[ch]:
                return False
    return len(stack) == 0
expression = "{[()()]}"
print("Balanced" if is_balanced(expression) else "Not Balanced")
```

Output

```
Balanced
```

7. Inventory Stock Management System


Jupyter Dsalabfile Last Checkpoint: 14 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[7]:

```
class Inventory:
    def __init__(self):
        self.items = {}

    def add_item(self, name, qty):
        if name in self.items:
            self.items[name] += qty
        else:
            self.items[name] = qty
        print(f"{qty} {name} added.")

    def remove_item(self, name, qty):
        if name not in self.items:
            print("Item not found.")
            return
        if qty > self.items[name]:
            print("Not enough stock.")
            return
        self.items[name] -= qty
        print(f"{qty} {name} removed.")
        if self.items[name] == 0:
            del self.items[name]

    def search_item(self, name):
        if name in self.items:
            print(f"{name}: {self.items[name]} in stock")
        else:
            print("Item not found.")

    def display(self):
        print("\n--- Inventory ---")
        for item, qty in self.items.items():
            print(f"{item}: {qty} units")
        print("-----------------\n")


# Driver
```

```
# Driver
inv = Inventory()
inv.add_item("Rice", 50)
inv.add_item("Sugar", 20)
inv.remove_item("Rice", 10)
inv.search_item("Rice")
inv.display()
```

Output:

```
50 Rice added.
20 Sugar added.
10 Rice removed.
Rice: 40 in stock

--- Inventory ---
Rice: 40 units
Sugar: 20 units
-----------------
```