2303A51739

Batch:25

Task 1: Smart Contact Manager (Arrays & Linked Lists)

Scenario

SR University's student club requires a simple Contact Manager Application to store members' names and phone numbers. The system should support efficient addition, searching, and deletion of contacts.

Tasks

1. Implement the contact manager using arrays (lists).

2. Implement the same functionality using a linked list for dynamic memory allocation.

3. Implement the following operations in both approaches:

o Add a contact

o Search for a contact

o Delete a contact

4. Use GitHub Copilot to assist in generating search and delete methods.

5. Compare array vs. linked list approaches with respect to:

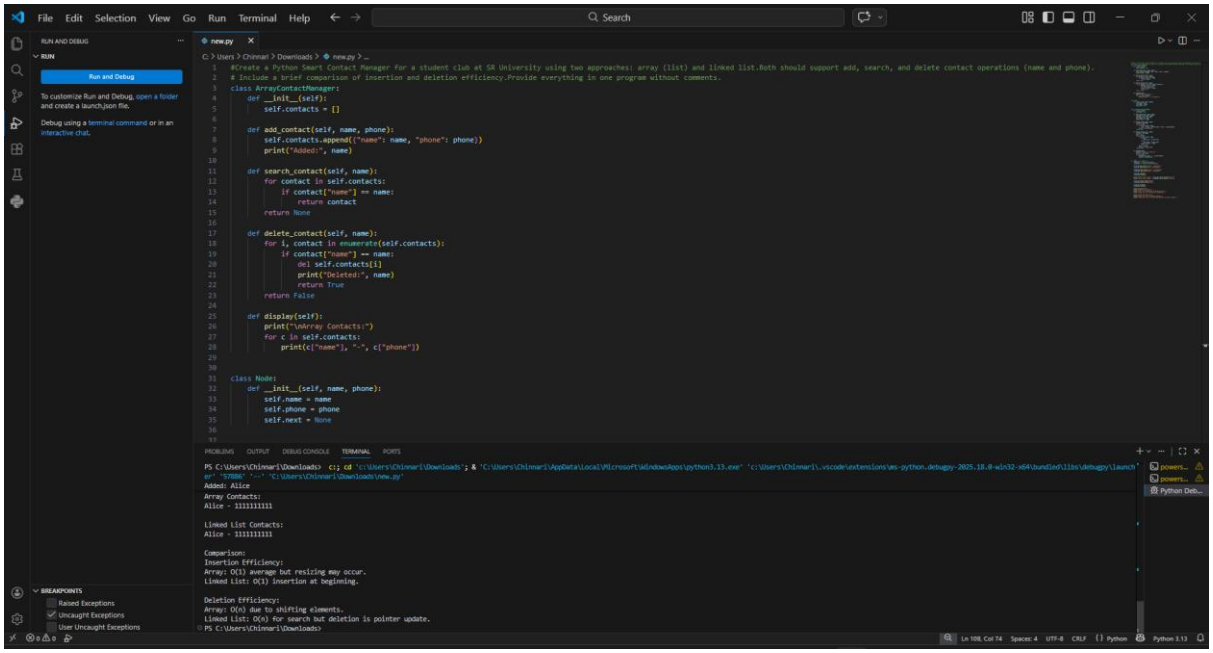
o Insertion efficiency

o Deletion efficiency

Expected Outcome

• Two working implementations (array-based and linked-list-based).

• A brief comparison explaining performance differences.



Task 2: Library Book Search System (Queues & Priority Queues)

Scenario

The SRU Library manages book borrow requests. Students and faculty submit requests, but faculty requests must be prioritized over student requests.

Tasks

1. Implement a Queue (FIFO) to manage book requests.

2. Extend the system to a Priority Queue, prioritizing faculty requests.

3. Use GitHub Copilot to assist in generating:

o enqueue() method

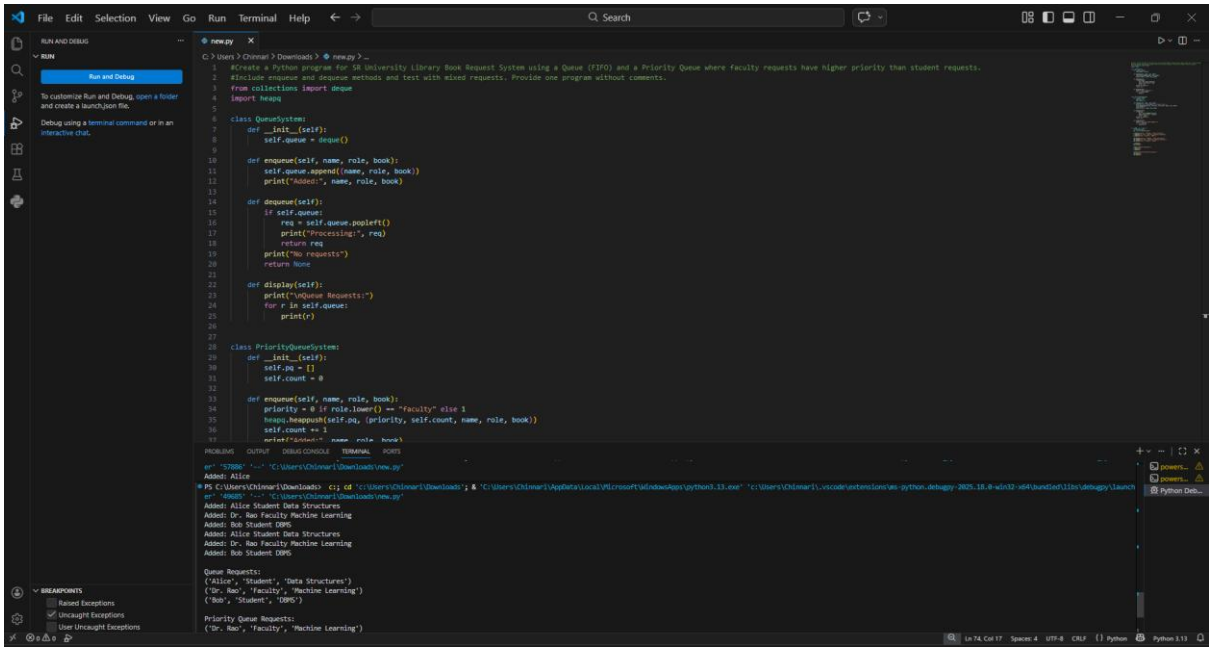
o dequeue() method

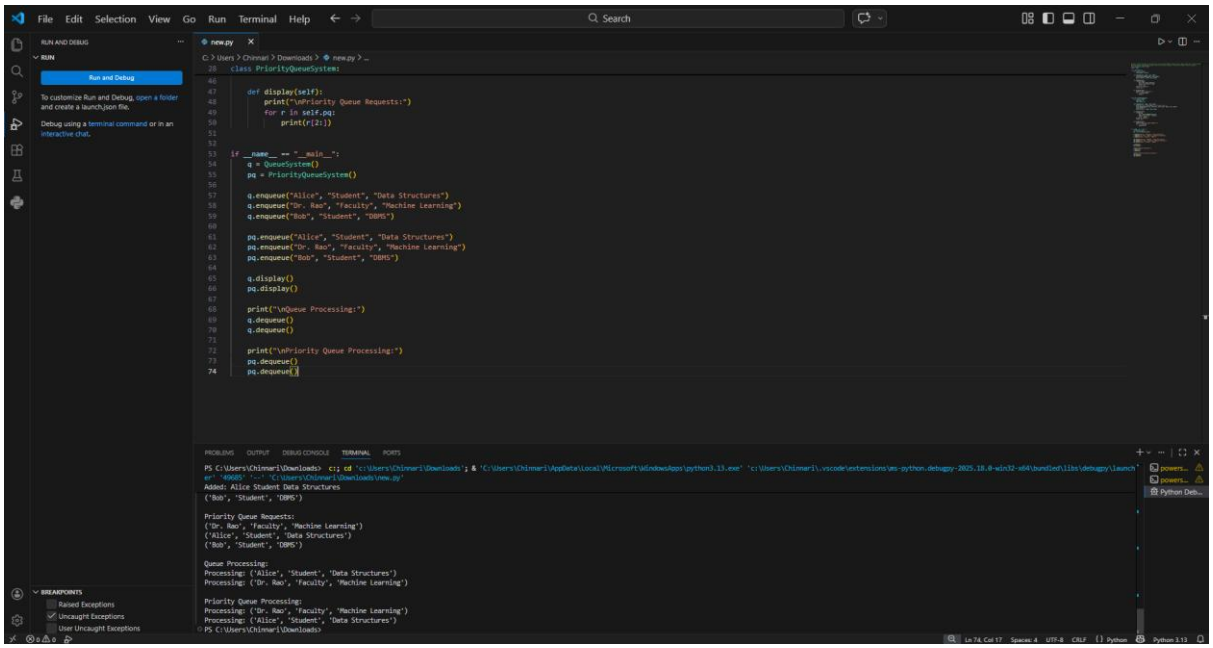
4. Test the system with a mix of student and faculty requests.

Expected Outcome

• Working queue and priority queue implementations.

• Correct prioritization of faculty requests





Task 3: Emergency Help Desk (Stack Implementation)

Scenario

SR University's IT Help Desk receives technical support tickets from students and staff. While tickets are received sequentially, issue escalation follows a Last-In, First-Out (LIFO) approach.

Tasks

1. Implement a Stack to manage support tickets.

2. Provide the following operations:

o push(ticket)

o pop()

o peek()

3. Simulate at least five tickets being raised and resolved.

4. Use GitHub Copilot to suggest additional stack operations such as:

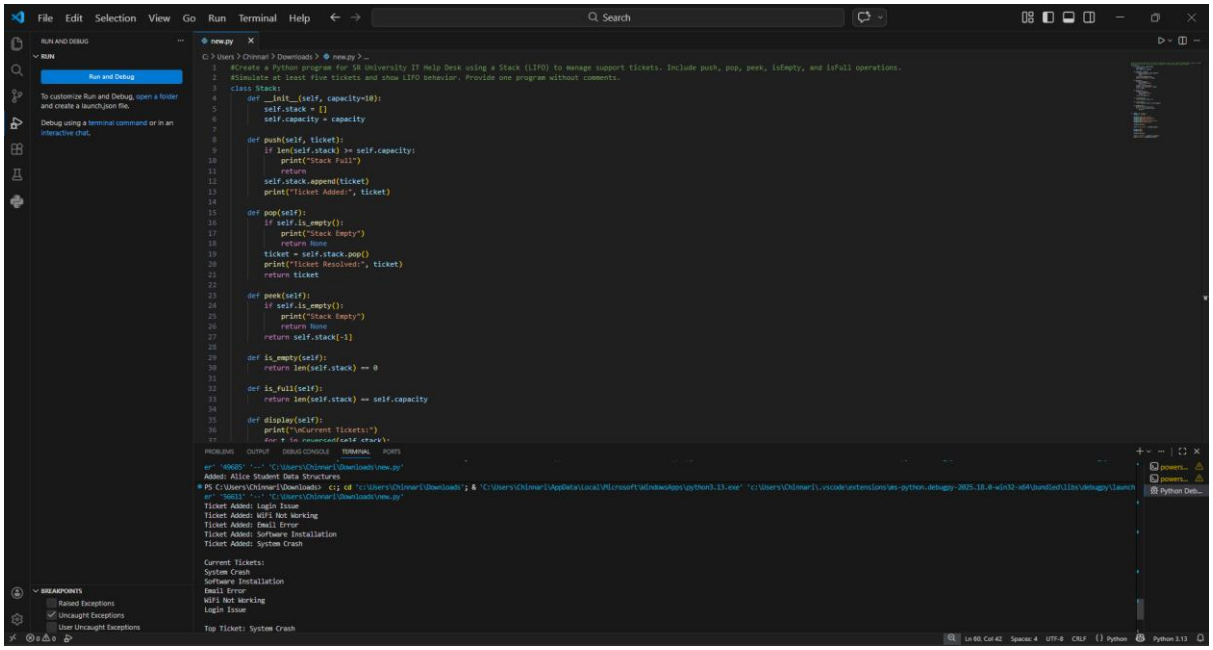
o Checking whether the stack is empty

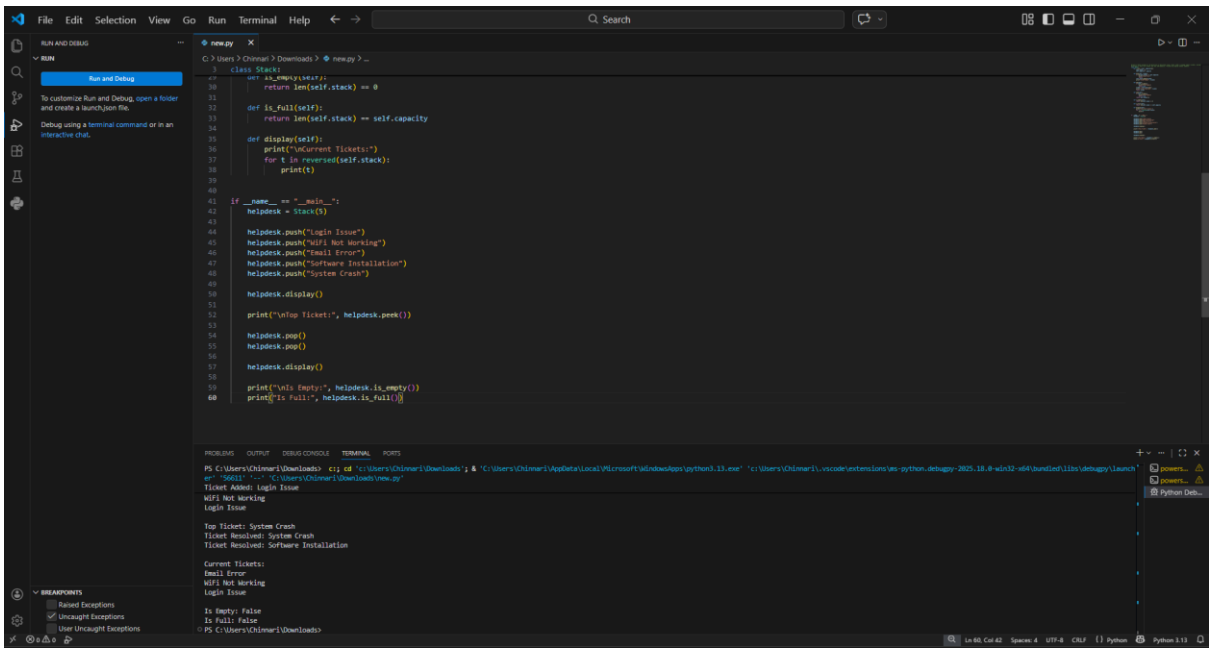
o Checking whether the stack is full (if applicable)

Expected Outcome

• Functional stack-based ticket management system.

• Clear demonstration of LIFO behavior.





Task 4: Hash Table

Objective

To implement a Hash Table and understand collision handling.

Task Description

Use AI to generate a hash table with:

• Insert

• Search

• Delete

Starter Code

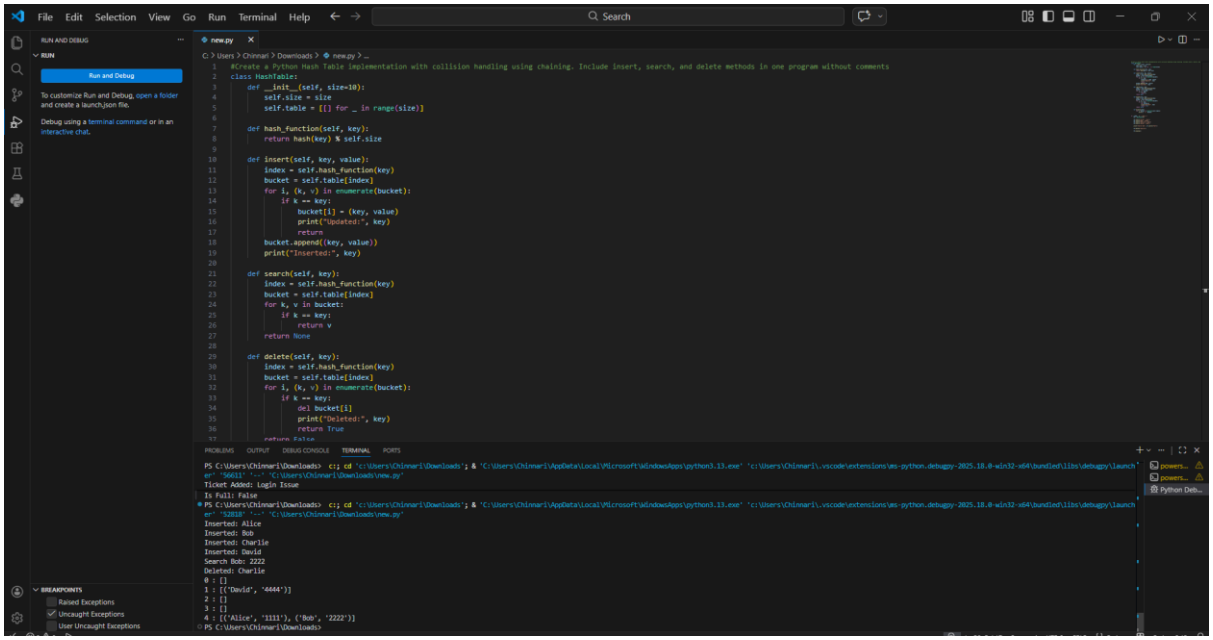
class HashTable:

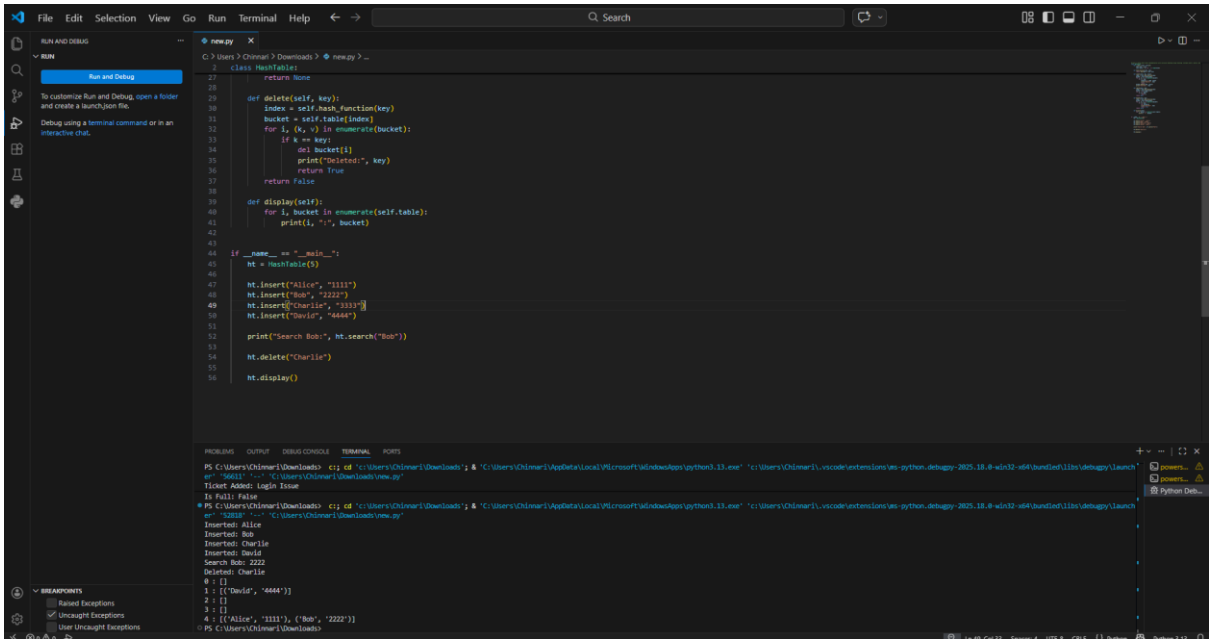
pass

Expected Outcome

• Collision handling using chaining

• Well-commented methods





Task 5: Real-Time Application Challenge

Scenario

Design a Campus Resource Management System with the following features:

- Student Attendance Tracking
- Event Registration System
- Library Book Borrowing
- Bus Scheduling System
- Cafeteria Order Queue

Student Tasks

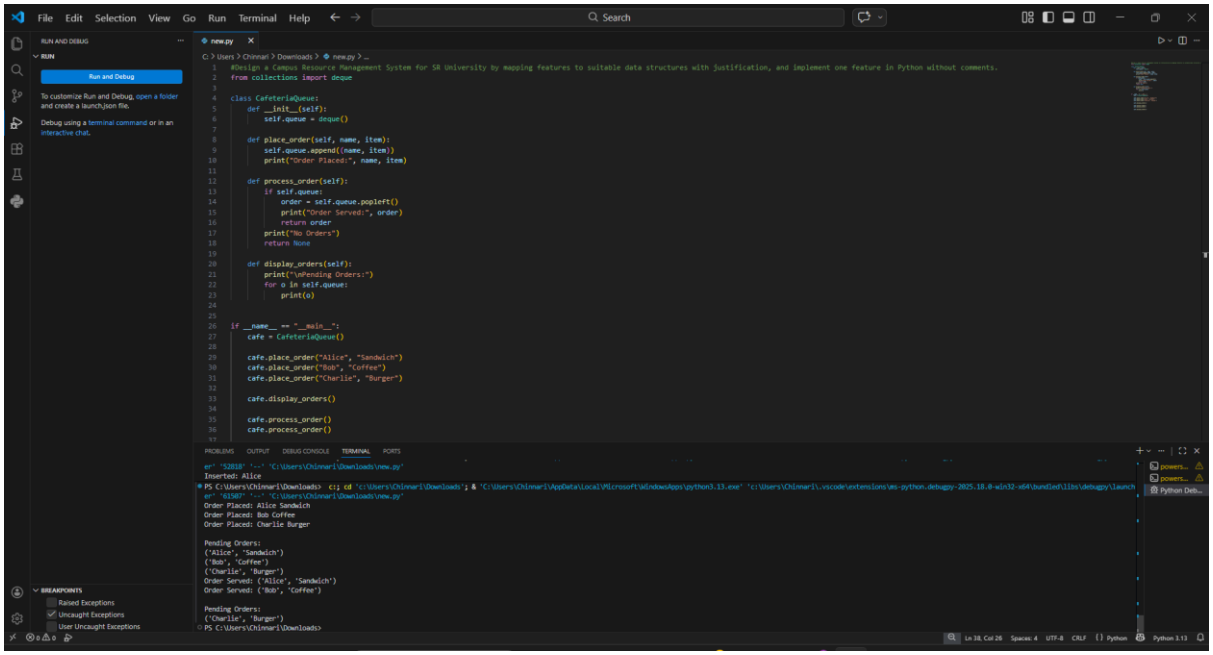
1. Choose the most appropriate data structure for each feature.

2. Justify your choice in 2–3 sentences.

3. Implement one selected feature using AI-assisted code generation.

Expected Outcome

- Mapping table: Feature → Data Structure → Justification
- One fully working Python implementation


File Edit Selection View Go Run Terminal Help
Search
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Debug using a terminal command or in an interactive chat.
new.py
C: > Users > Chinnari > Downloads > new.py > ...
#Design a Campus Resource Management System for SR University by mapping features to suitable data structures with justification, and implement one feature in Python without comments.
from collections import deque
class CafeteriaQueue:
def __init__(self):
self.queue = deque()
def place_order(self, name, item):
self.queue.append((name, item))
print("Order Placed:", name, item)
def process_order(self):
if self.queue:
order = self.queue.popleft()
print("Order Served:", order)
return order
print("No Orders")
return None
def display_orders(self):
print("\nPending Orders:")
for o in self.queue:
print(o)
if __name__ == "__main__":
cafe = CafeteriaQueue()
cafe.place_order("Alice", "Sandwich")
cafe.place_order("Bob", "Coffee")
cafe.place_order("Charlie", "Burger")
cafe.display_orders()
cafe.process_order()
cafe.process_order()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Inserted: Alice
PS C:\Users\Chinnari\Downloads> c:; cd 'c:\Users\Chinnari\Downloads'; & 'C:\Users\Chinnari\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\Chinnari\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launch
Order Placed: Alice Sandwich
Order Placed: Bob Coffee
Order Placed: Charlie Burger
Pending Orders:
('Alice', 'Sandwich')
('Bob', 'Coffee')
('Charlie', 'Burger')
Order Served: ('Alice', 'Sandwich')
Order Served: ('Bob', 'Coffee')
Pending Orders:
('Charlie', 'Burger')
PS C:\Users\Chinnari\Downloads>
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
Python Deb...
Ln 38, Col 26
Spaces: 4
UTF-8
CRLF
Python
Python 3.13