ASSIGNMENT-11.1

AMGOTH VIKAS NAYAK

2403A51410

Lab 11-Data Structures with AI: Implementing Fundamental Structures.

Lab Objectives:

• Use AI to assist in designing and implementing fundamental data structures in Python.

• Learn how to prompt AI for structure creation, optimization, and documentation.

• Improve understanding of Lists, Stacks, Queues, Linked Lists, Trees, Graphs, and Hash Tables.

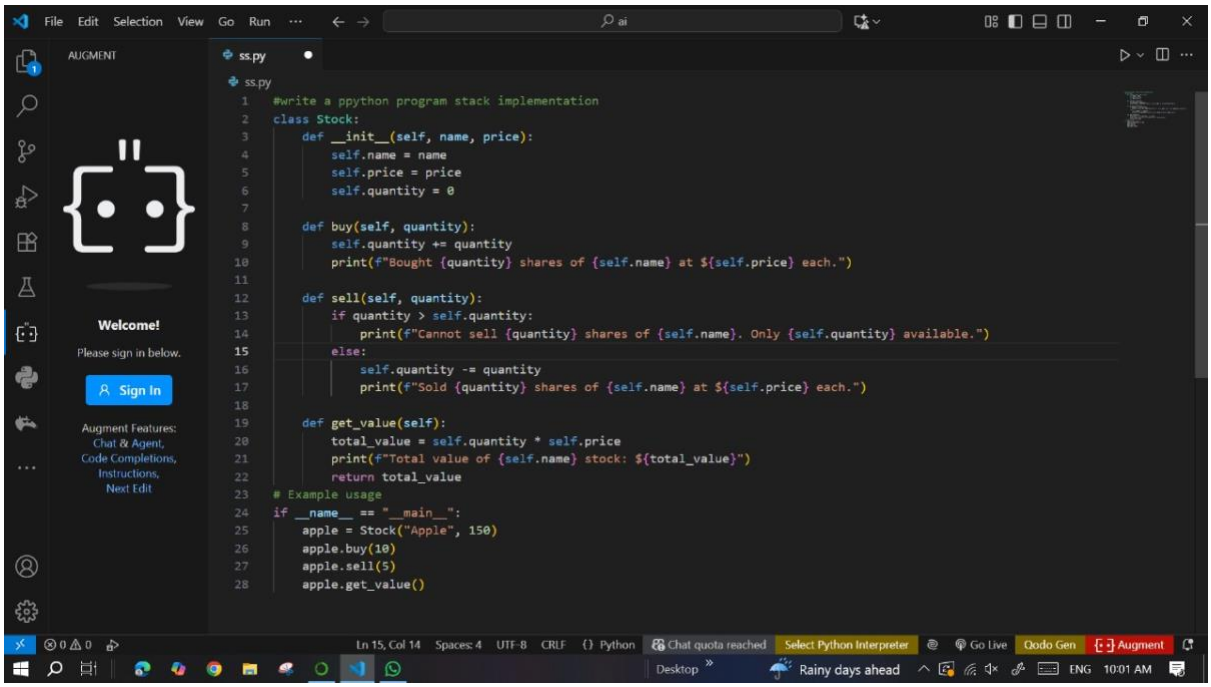
• Enhance code quality with AI-generated comments and performance suggestions.

Task Description #1 – Stack Implementation

Task: Use AI to generate a Stack class with push, pop, peek, and is_empty methods.

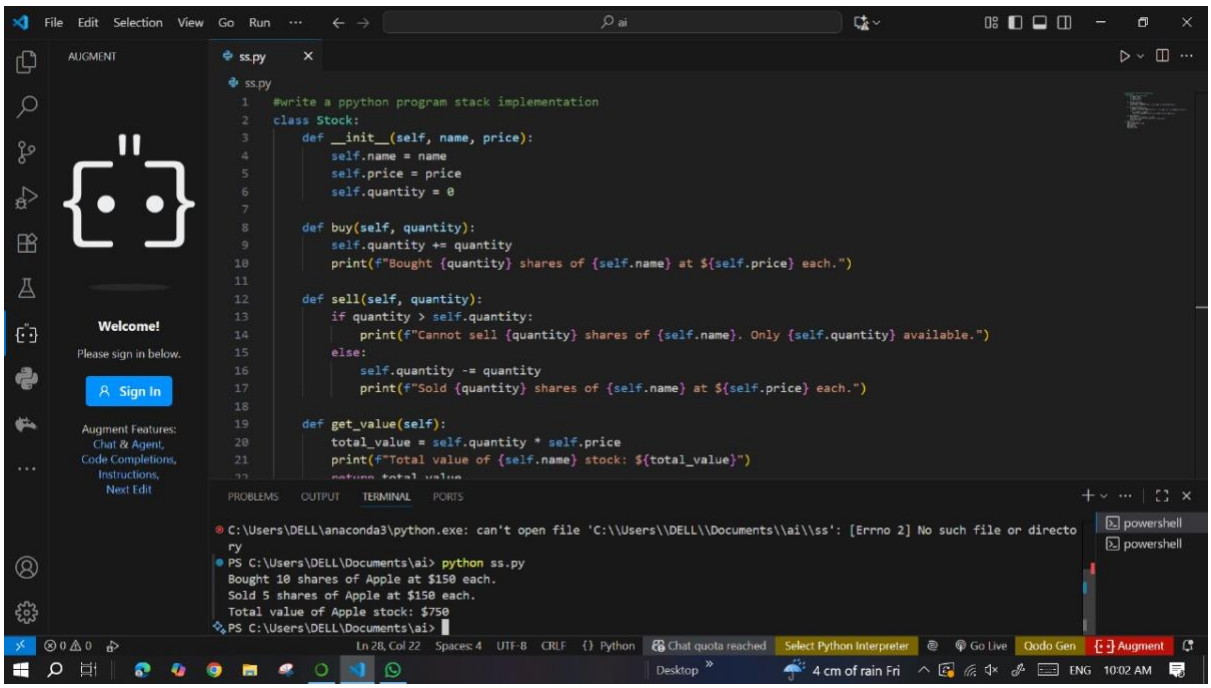
Sample Input Code:

```
class Stack:
    pass
```



Expected Output:

• A functional stack implementation with all required methods and docstrings.



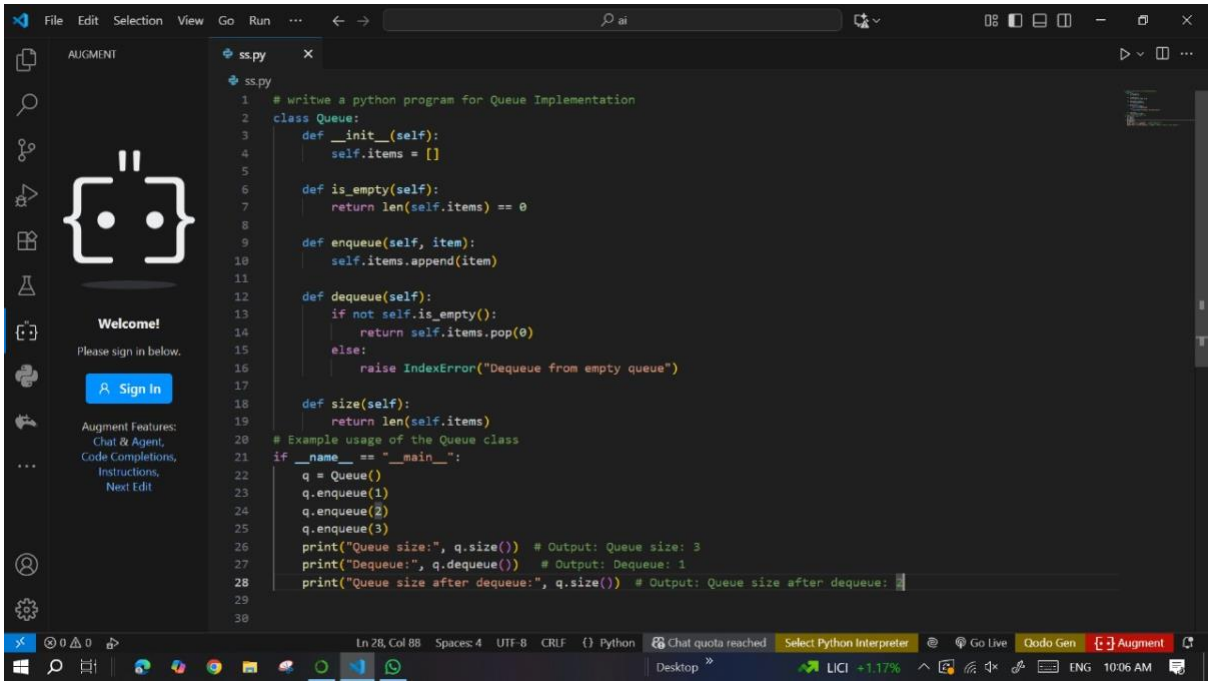
Task Description #2 – Queue Implementation

Task: Use AI to implement a Queue using Python lists.

Sample Input Code:

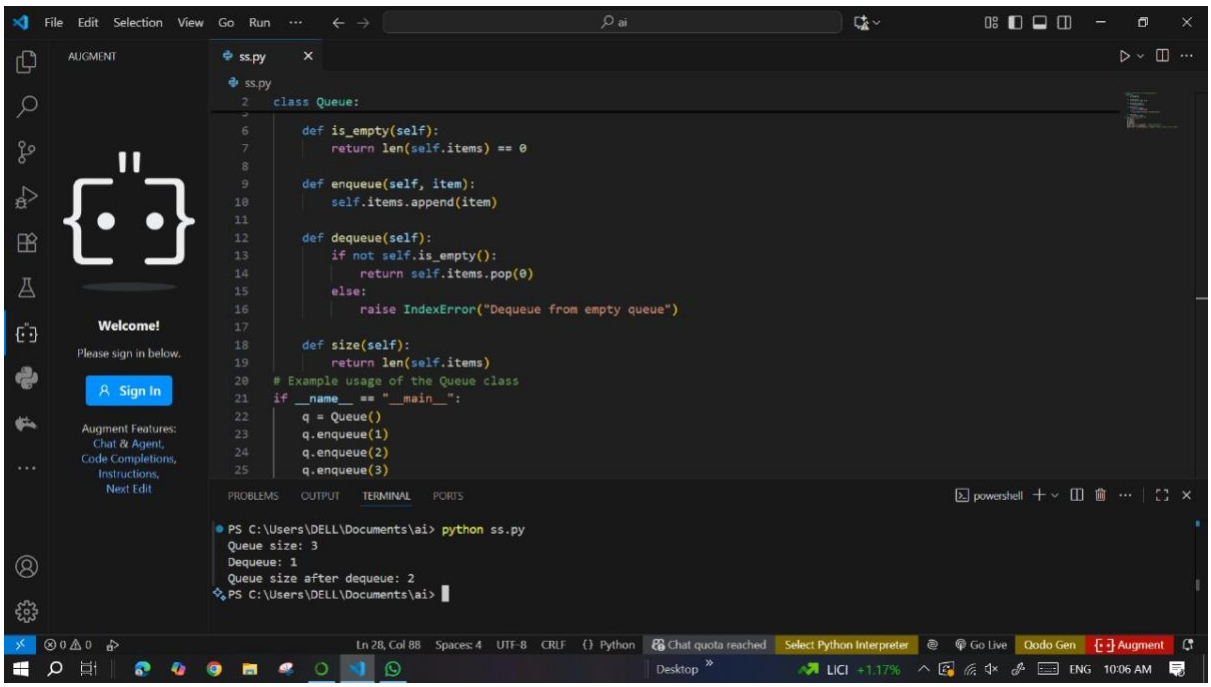
class Queue:

pass



Expected Output:

• FIFO-based queue class with enqueue, dequeue, peek, and size methods.



Task Description #3 – Linked List

Task: Use AI to generate a Singly Linked List with insert and display methods.

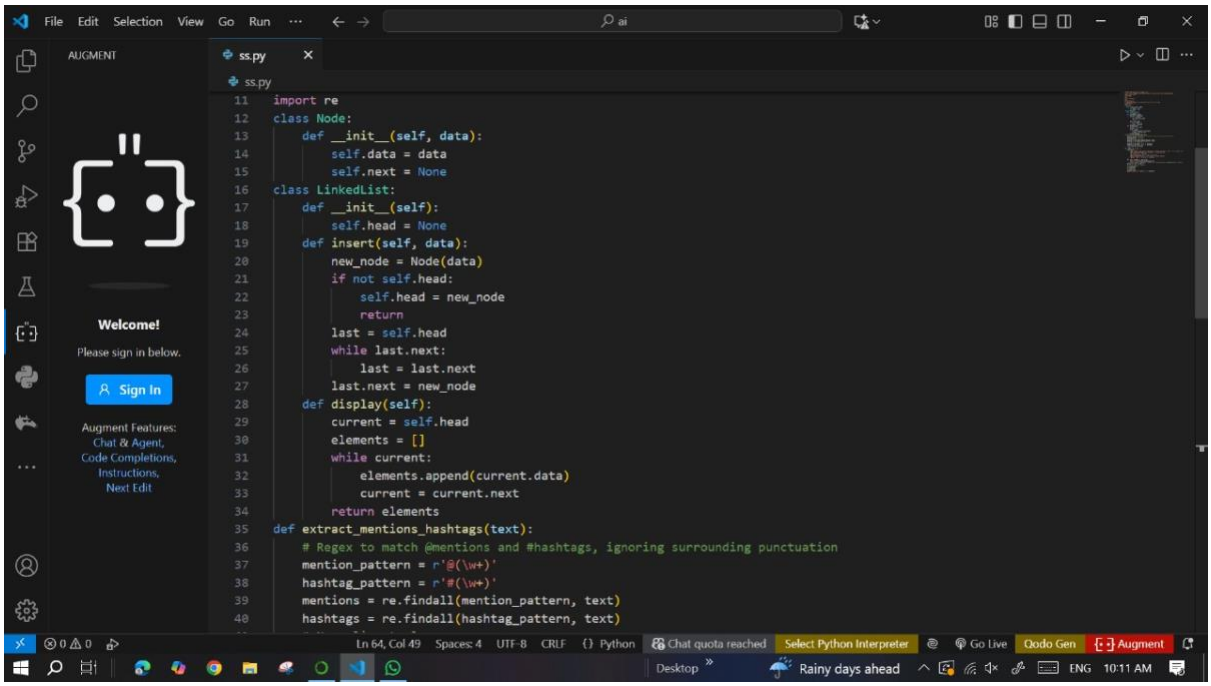
Sample Input Code:

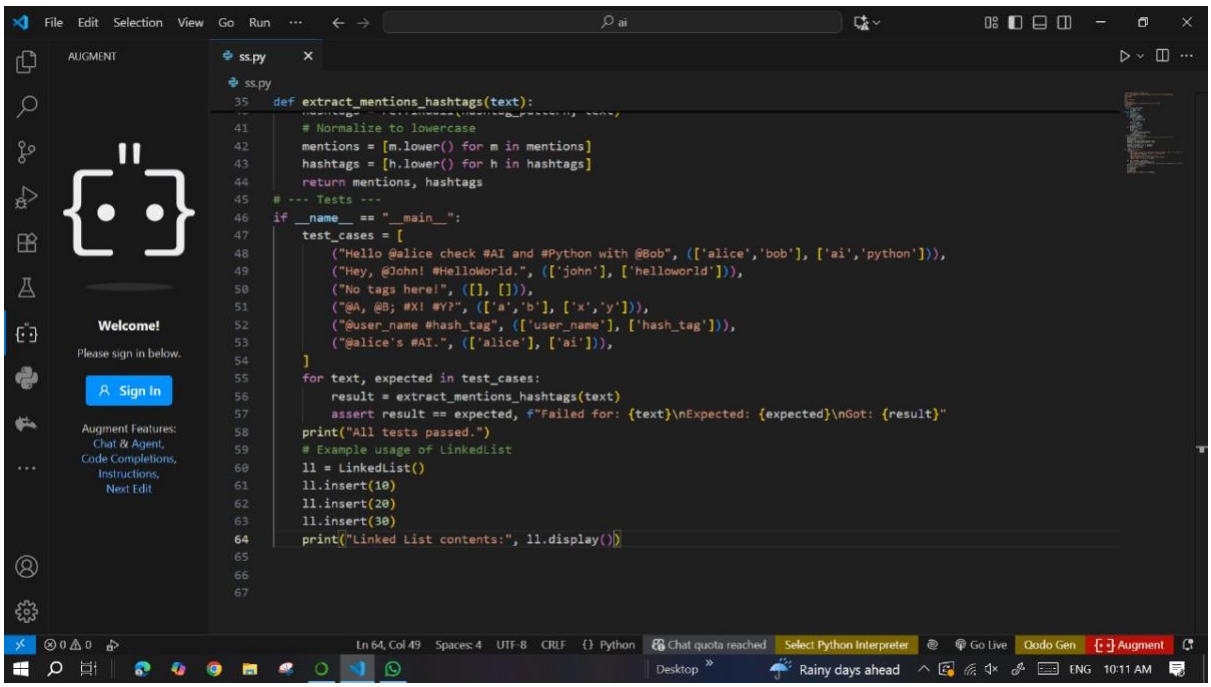
class Node:

pass

class LinkedList:

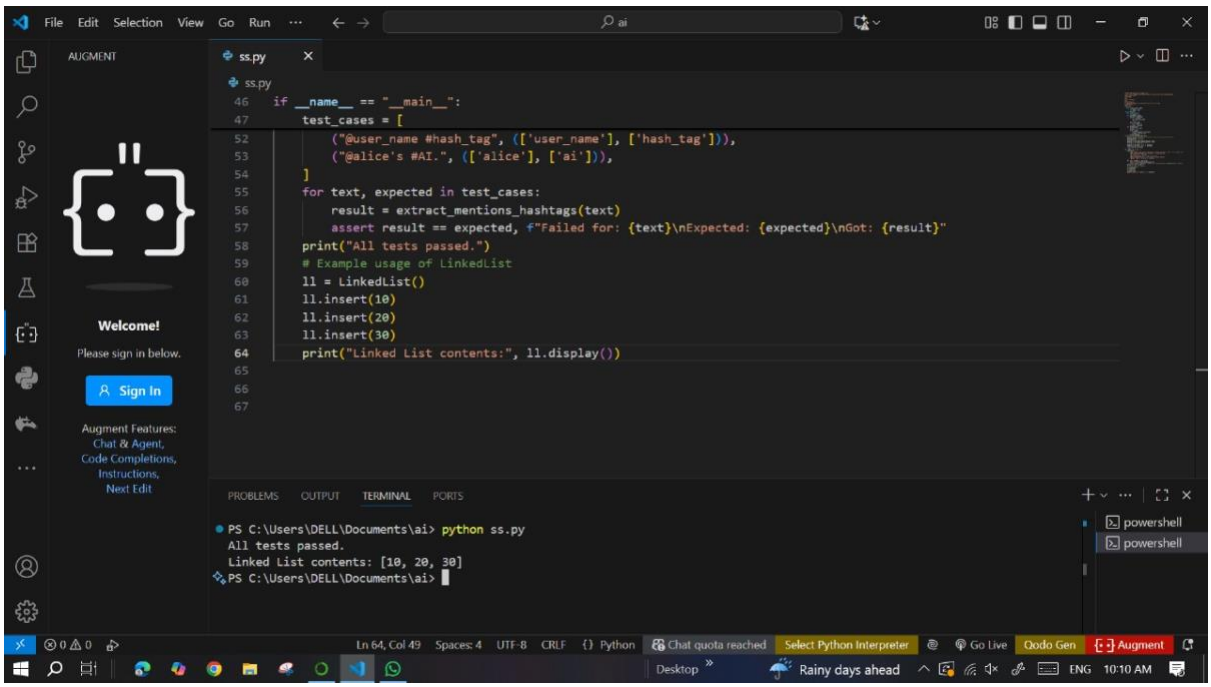
pass





Expected Output:

• A working linked list implementation with clear method documentation.



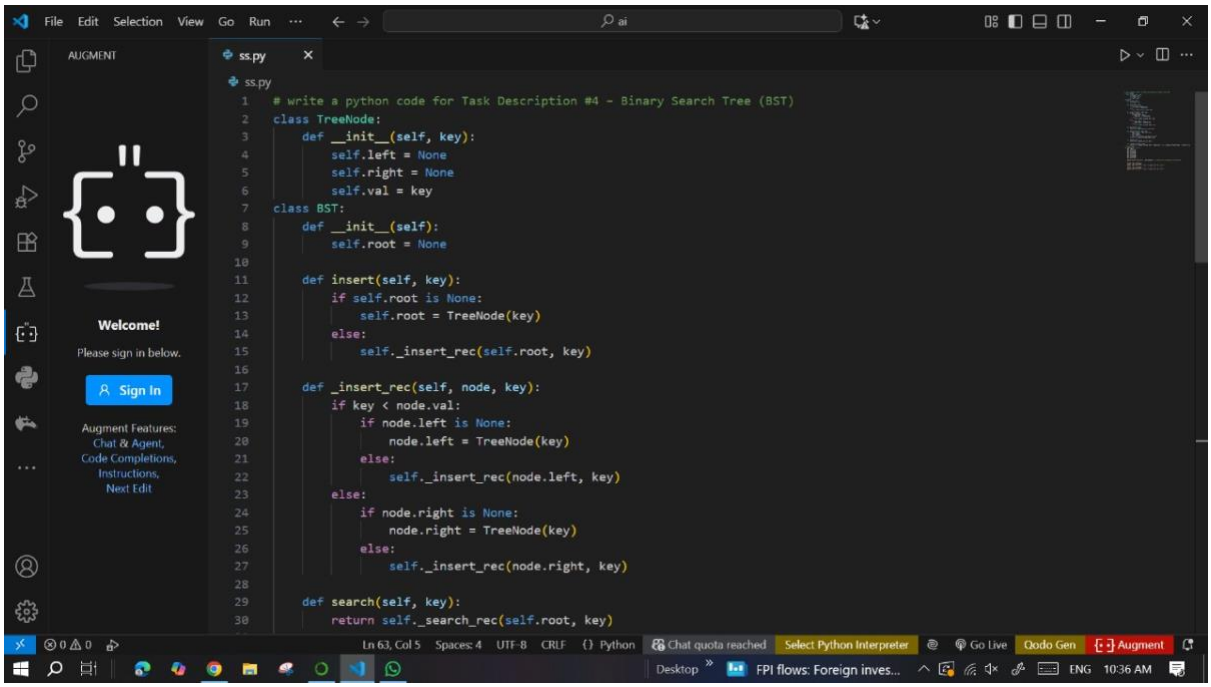
Task Description #4 – Binary Search Tree (BST)

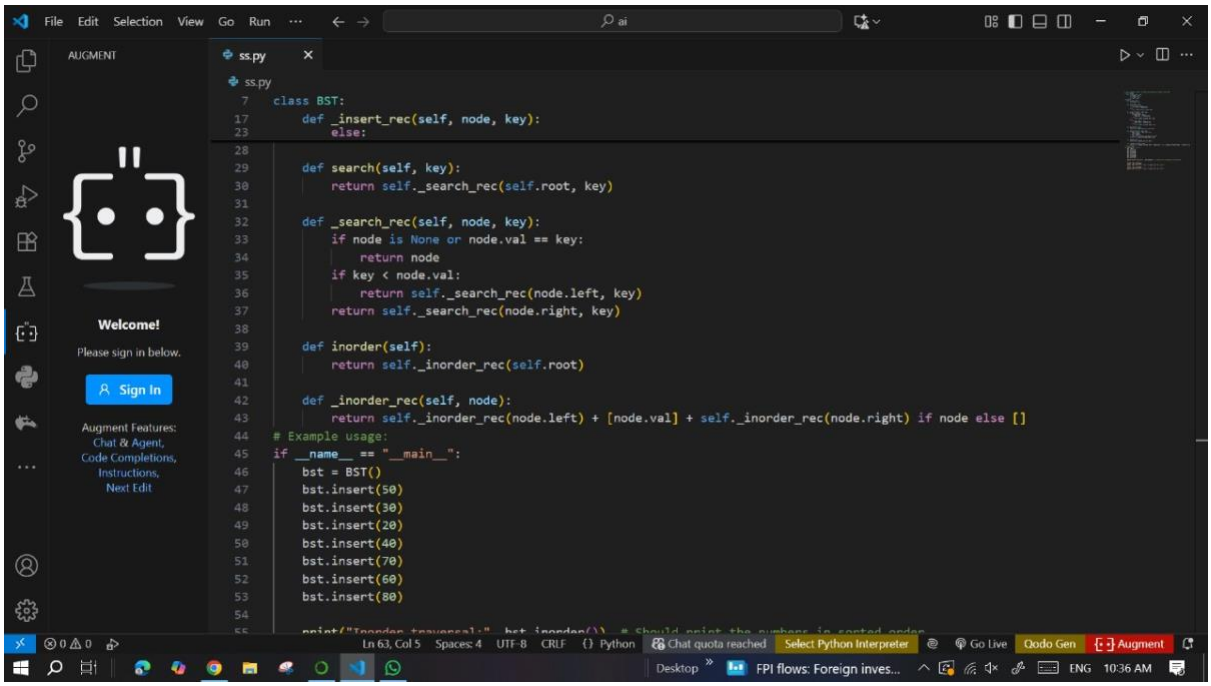
Task: Use AI to create a BST with insert and in-order traversal methods.

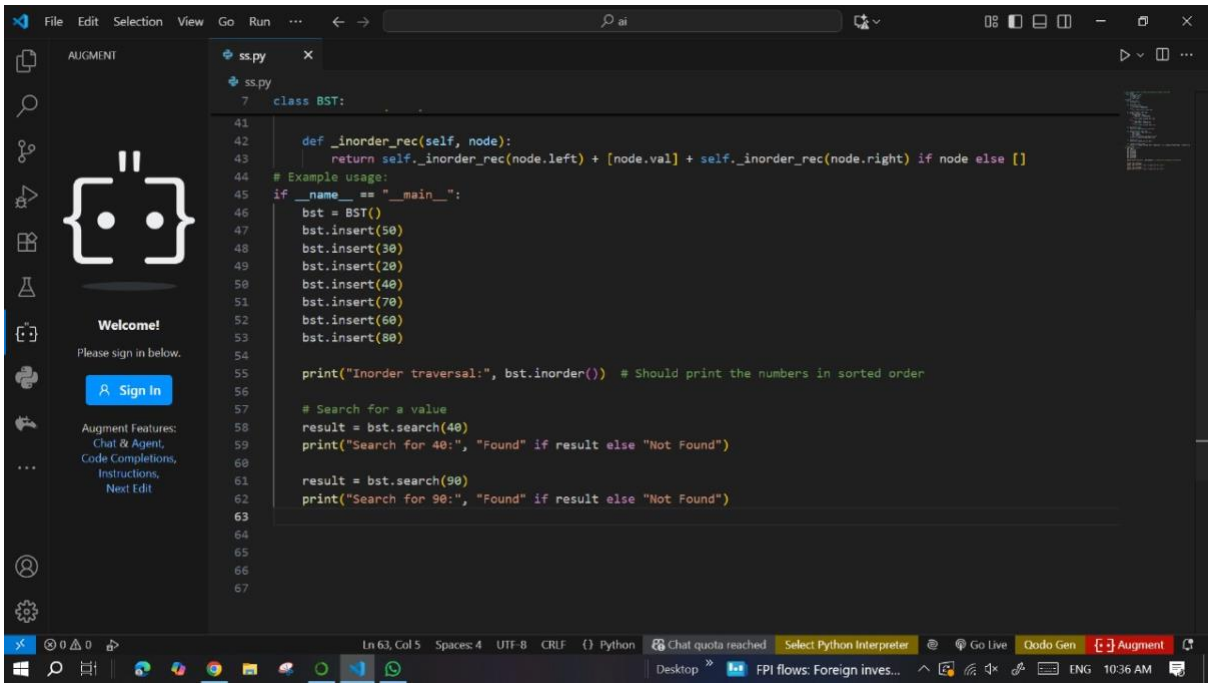
Sample Input Code:

class BST:

pass

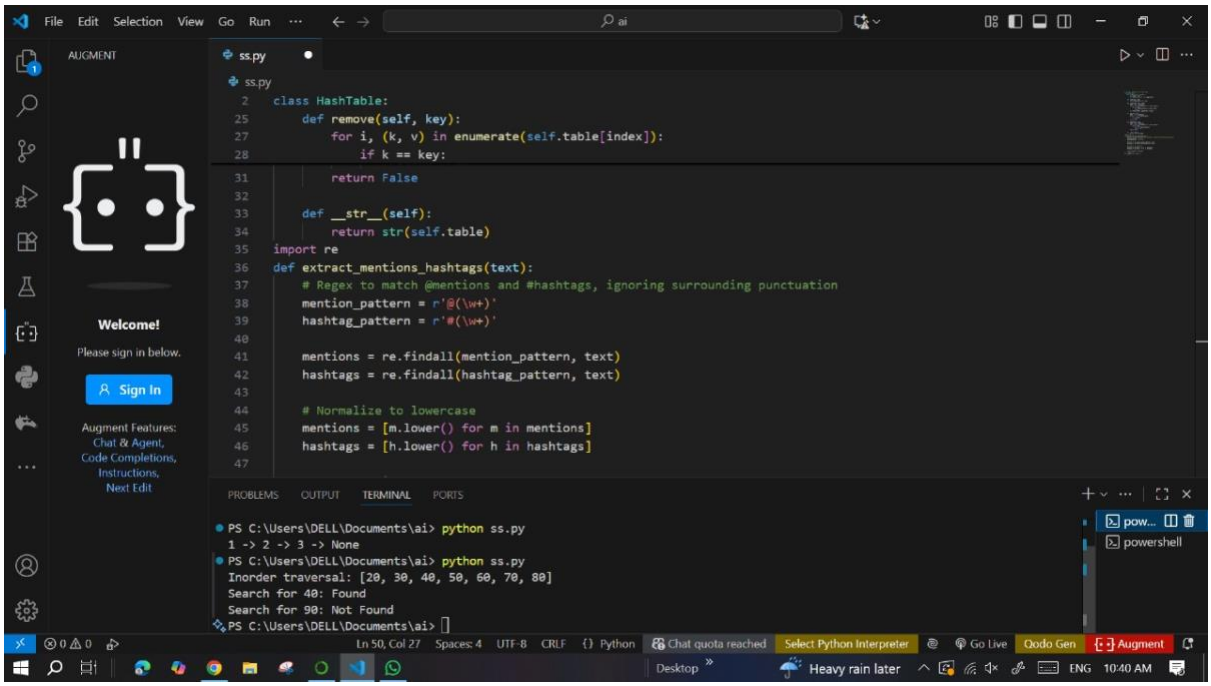

File Edit Selection View Go Run
AUGMENT
ss.py
Welcome!
Please sign in below.
Sign In
Augment Features:
Chat & Agent,
Code Completions,
Instructions,
Next Edit
# write a python code for Task Description #4 - Binary Search Tree (BST)
class TreeNode:
def __init__(self, key):
self.left = None
self.right = None
self.val = key
class BST:
def __init__(self):
self.root = None
def insert(self, key):
if self.root is None:
self.root = TreeNode(key)
else:
self._insert_rec(self.root, key)
def _insert_rec(self, node, key):
if key < node.val:
if node.left is None:
node.left = TreeNode(key)
else:
self._insert_rec(node.left, key)
else:
if node.right is None:
node.right = TreeNode(key)
else:
self._insert_rec(node.right, key)
def search(self, key):
return self._search_rec(self.root, key)
Ln 63, Col 5 Spaces: 4 UTF-8 CRLF {} Python
Chat quota reached
Select Python Interpreter
Go Live
Qodo Gen
Augment
Desktop
FPI flows: Foreign inves...
ENG
10:36 AM


File Edit Selection View Go Run
AUGMENT
ss.py
Welcome!
Please sign in below.
Sign In
Augment Features:
Chat & Agent,
Code Completions,
Instructions,
Next Edit
class BST:
def _insert_rec(self, node, key):
else:
def search(self, key):
return self._search_rec(self.root, key)
def _search_rec(self, node, key):
if node is None or node.val == key:
return node
if key < node.val:
return self._search_rec(node.left, key)
return self._search_rec(node.right, key)
def inorder(self):
return self._inorder_rec(self.root)
def _inorder_rec(self, node):
return self._inorder_rec(node.left) + [node.val] + self._inorder_rec(node.right) if node else []
# Example usage:
if __name__ == "__main__":
bst = BST()
bst.insert(50)
bst.insert(30)
bst.insert(20)
bst.insert(40)
bst.insert(70)
bst.insert(60)
bst.insert(80)
Ln 63, Col 5 Spaces: 4 UTF-8 CRLF {} Python
Chat quota reached
Select Python Interpreter
Go Live
Qodo Gen
Augment
Desktop
FPI flows: Foreign inves...
ENG
10:36 AM



Expected Output:

• BST implementation with recursive insert and traversal methods.



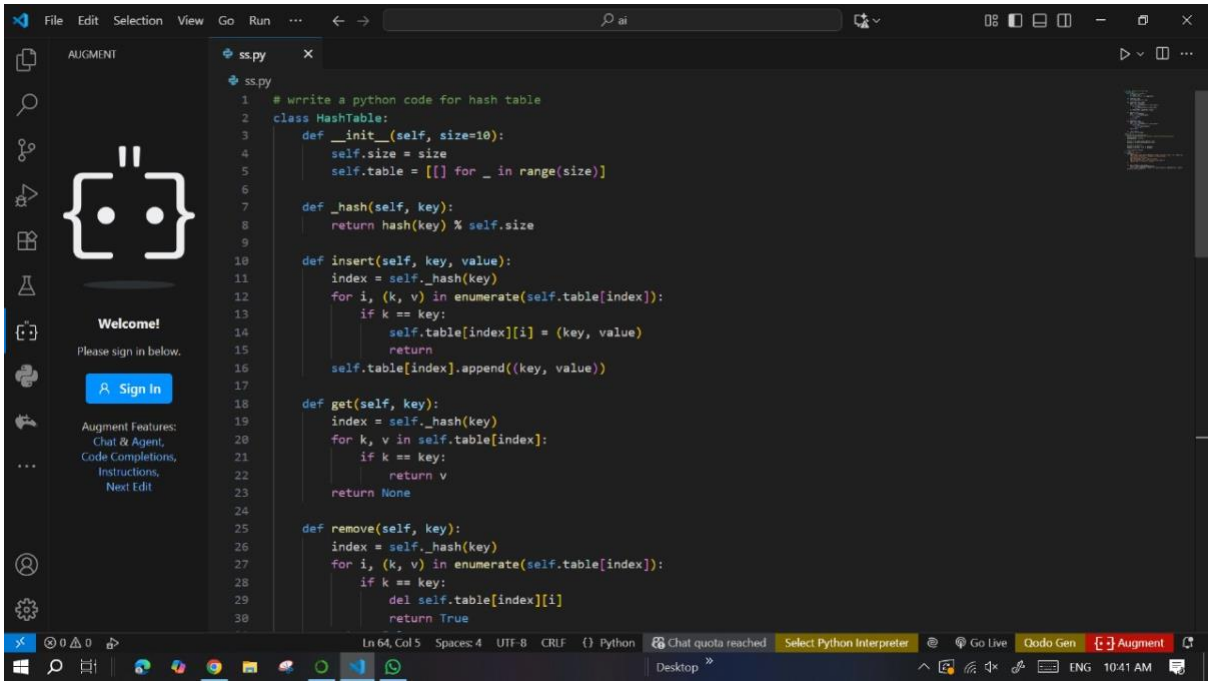
Task Description #5 – Hash Table

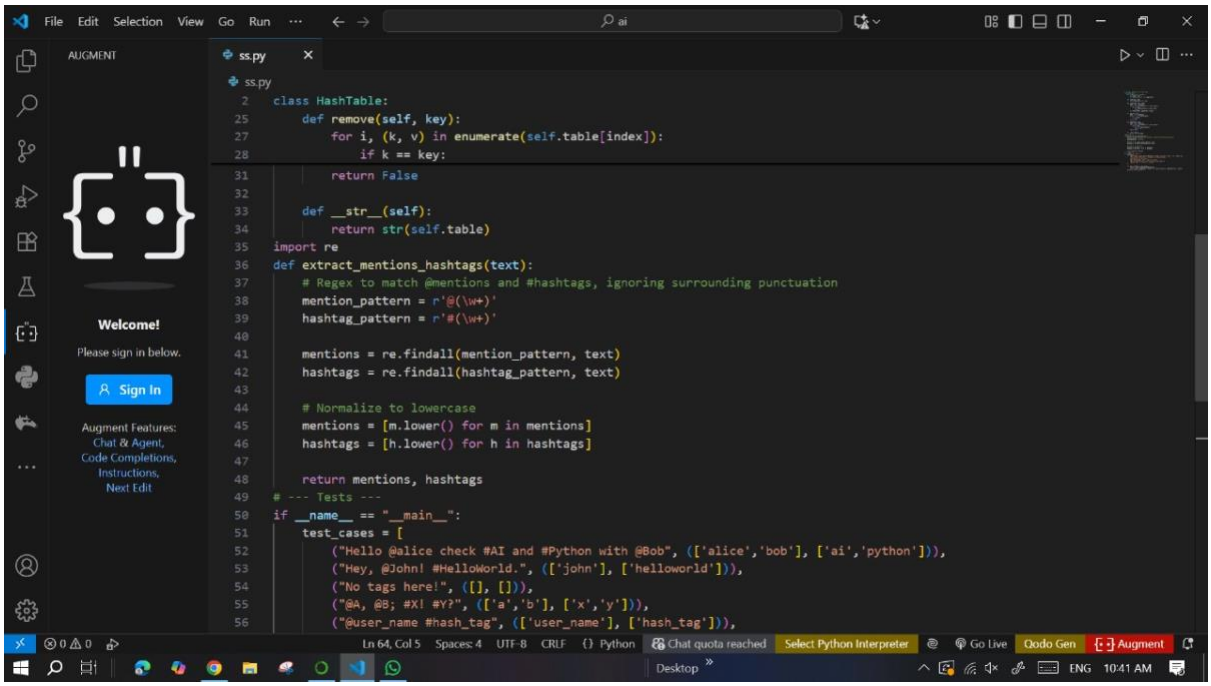
Task: Use AI to implement a hash table with basic insert, search, and delete methods.

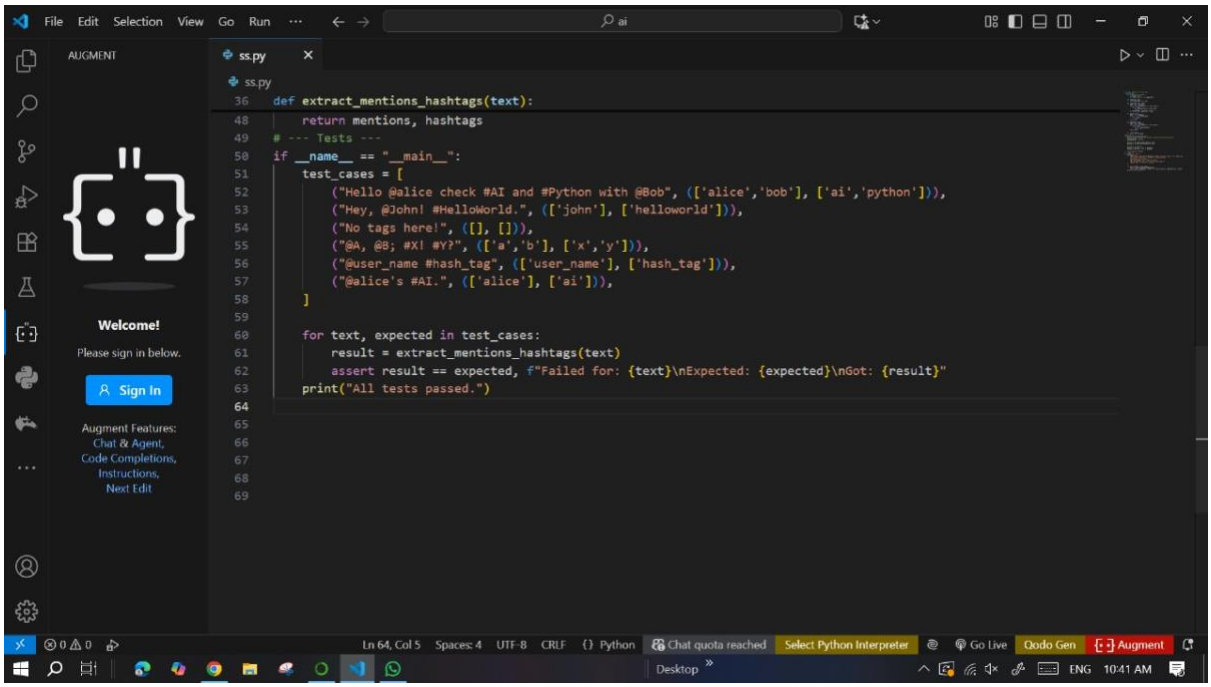
Sample Input Code:

class HashTable:

pass

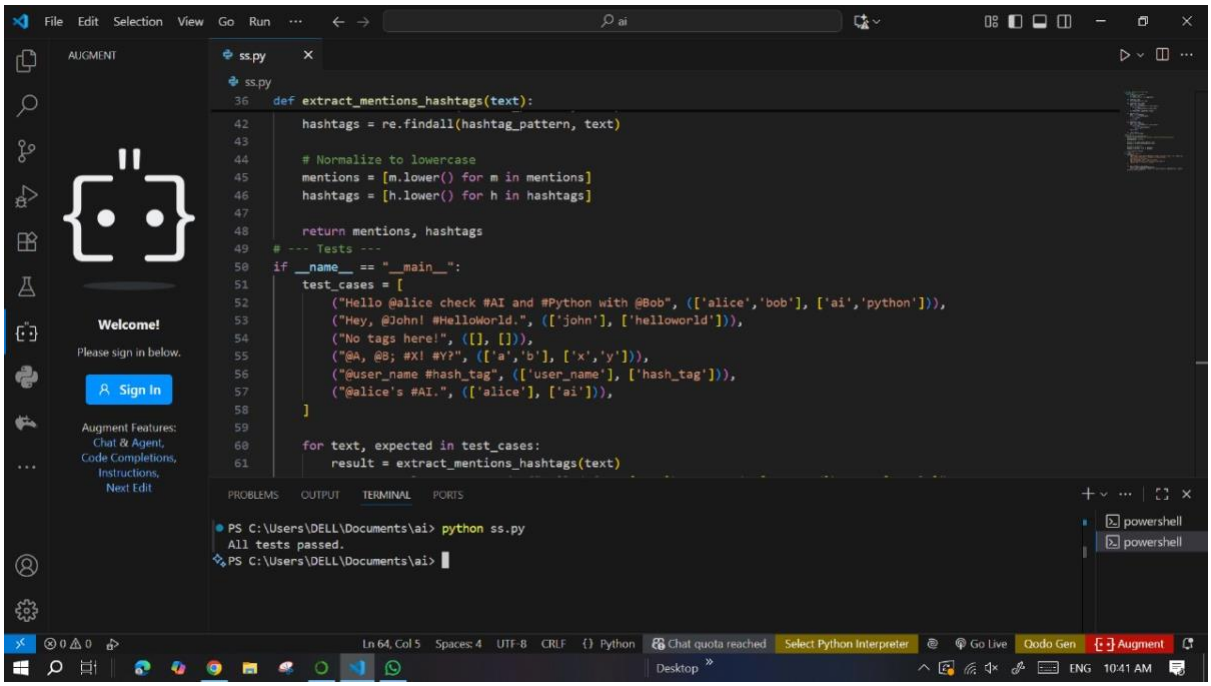

File Edit Selection View Go Run
AUGMENT
ss.py
Welcome!
Please sign in below.
Sign In
Augment Features:
Chat & Agent,
Code Completions,
Instructions,
Next Edit
# wrrite a python code for hash table
class HashTable:
def __init__(self, size=10):
self.size = size
self.table = [[] for _ in range(size)]
def _hash(self, key):
return hash(key) % self.size
def insert(self, key, value):
index = self._hash(key)
for i, (k, v) in enumerate(self.table[index]):
if k == key:
self.table[index][i] = (key, value)
return
self.table[index].append((key, value))
def get(self, key):
index = self._hash(key)
for k, v in self.table[index]:
if k == key:
return v
return None
def remove(self, key):
index = self._hash(key)
for i, (k, v) in enumerate(self.table[index]):
if k == key:
del self.table[index][i]
return True
Ln 64, Col 5 Spaces: 4 UTF-8 CRLF {} Python
Chat quota reached
Select Python Interpreter
Go Live
Qodo Gen
Augment
Desktop
ENG 10:41 AM


File Edit Selection View Go Run
AUGMENT
ss.py
Welcome!
Please sign in below.
Sign In
Augment Features:
Chat & Agent,
Code Completions,
Instructions,
Next Edit
class HashTable:
def remove(self, key):
for i, (k, v) in enumerate(self.table[index]):
if k == key:
return False
def __str__(self):
return str(self.table)
import re
def extract_mentions_hashtags(text):
# Regex to match @mentions and #hashtags, ignoring surrounding punctuation
mention_pattern = r'@(\w+)'
hashtag_pattern = r'#(\w+)'
mentions = re.findall(mention_pattern, text)
hashtags = re.findall(hashtag_pattern, text)
# Normalize to lowercase
mentions = [m.lower() for m in mentions]
hashtags = [h.lower() for h in hashtags]
return mentions, hashtags
# --- Tests ---
if __name__ == "__main__":
test_cases = [
("Hello @alice check #AI and #Python with @Bob", (['alice','bob'], ['ai','python'])),
("Hey, @John! #HelloWorld.", (['john'], ['helloworld'])),
("No tags here!", ([], [])),
("@A, @B; #X! #Y?", (['a','b'], ['x','y'])),
("@user_name #hash_tag", (['user_name'], ['hash_tag'])),
Ln 64, Col 5 Spaces: 4 UTF-8 CRLF {} Python
Chat quota reached
Select Python Interpreter
Go Live
Qodo Gen
Augment
Desktop
ENG 10:41 AM



Expected Output:

• Collision handling using chaining, with well-commented methods.



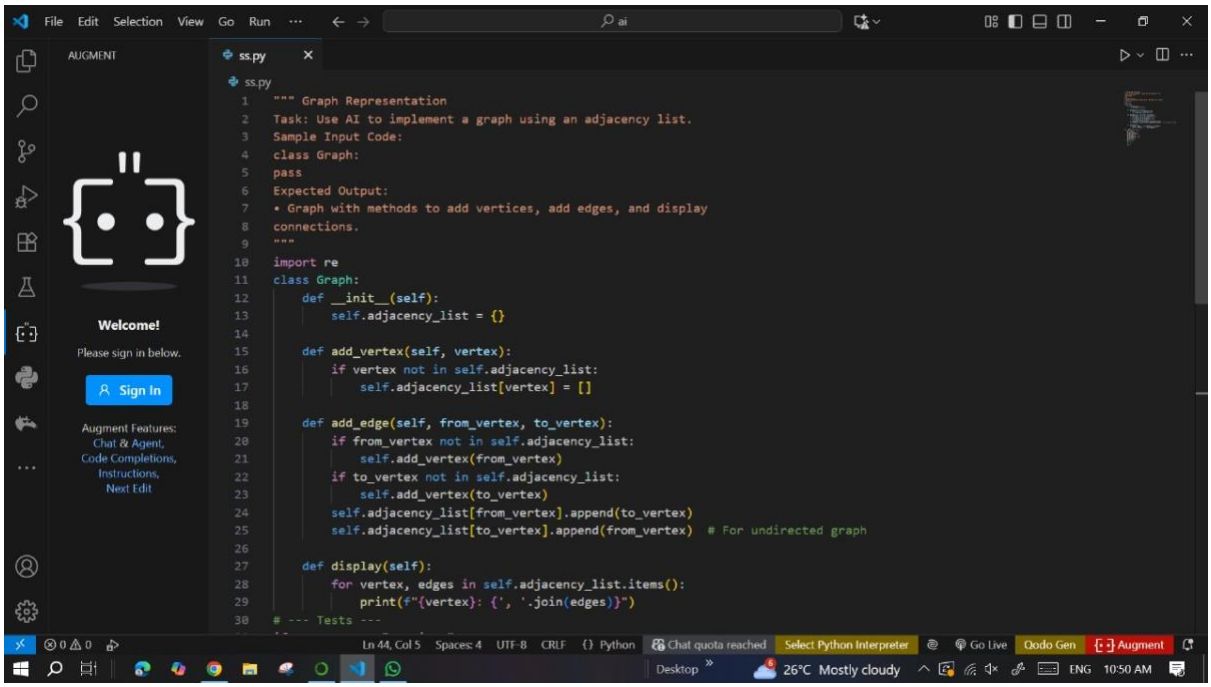
Task Description #6 – Graph Representation

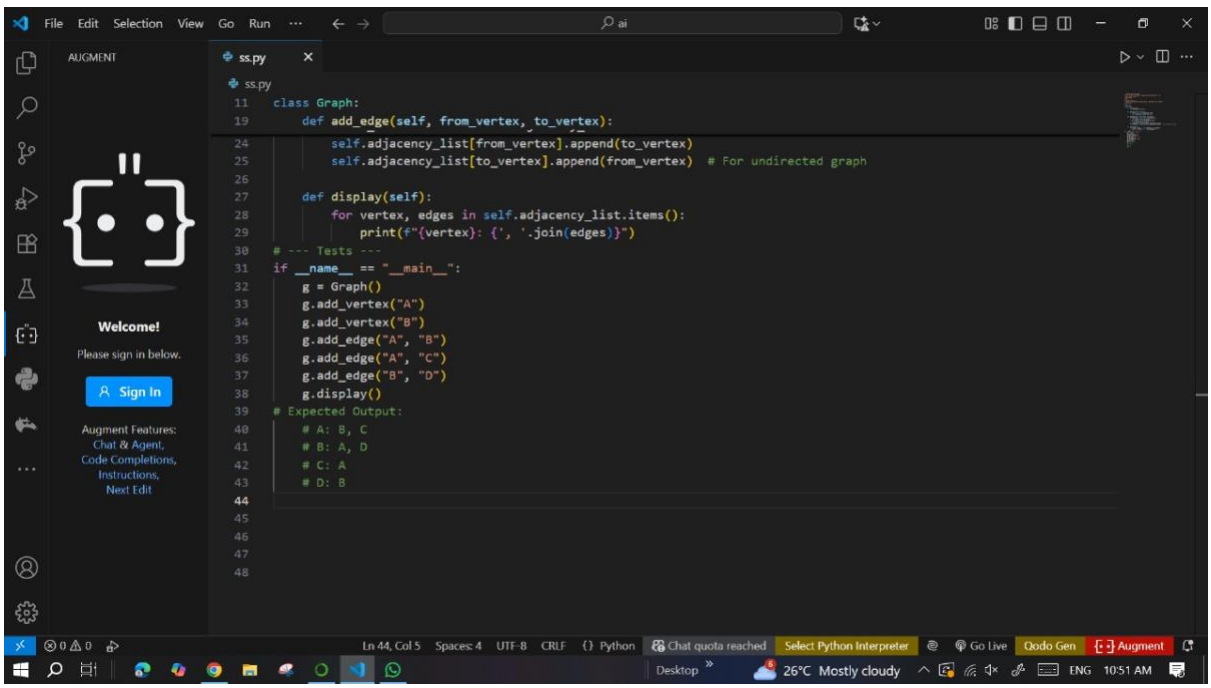
Task: Use AI to implement a graph using an adjacency list.

Sample Input Code:

class Graph:

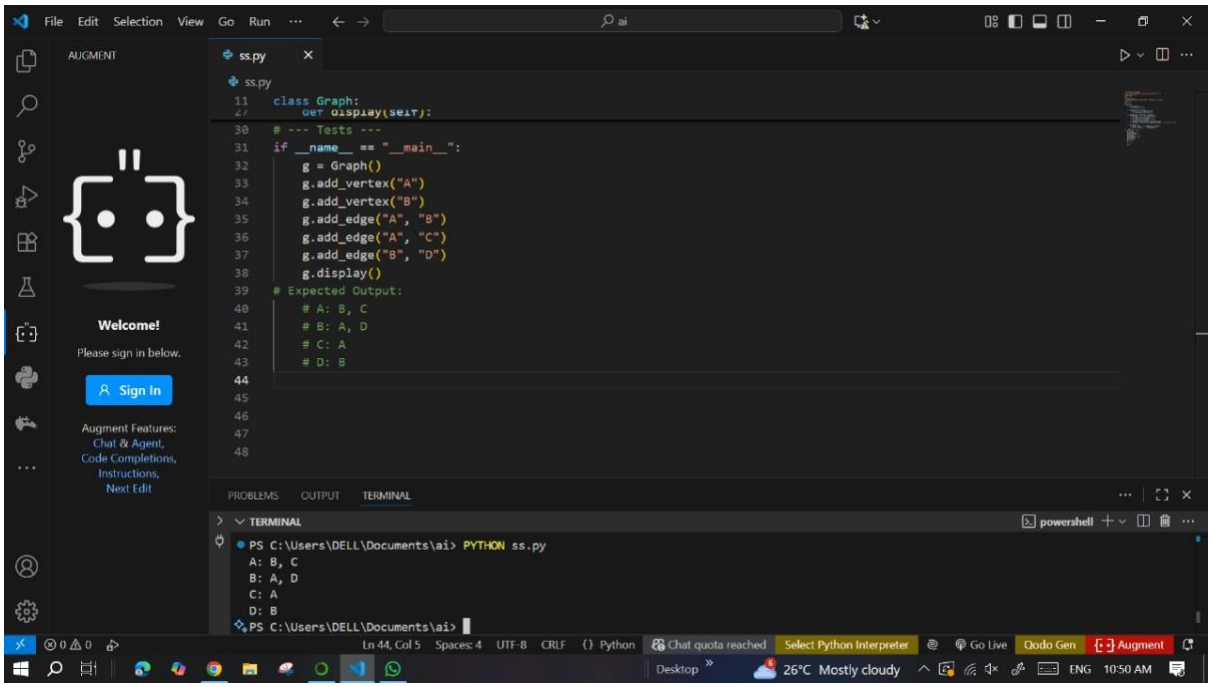
pass





Expected Output:

• Graph with methods to add vertices, add edges, and display connections.



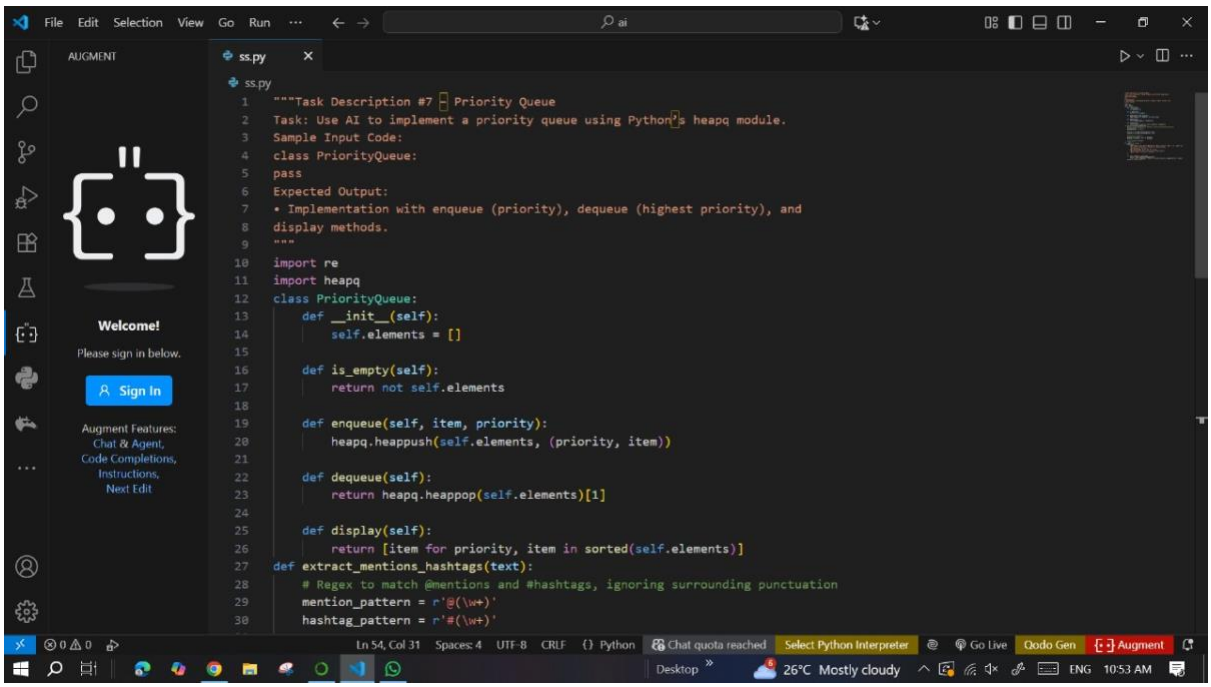
Task Description #7 – Priority Queue

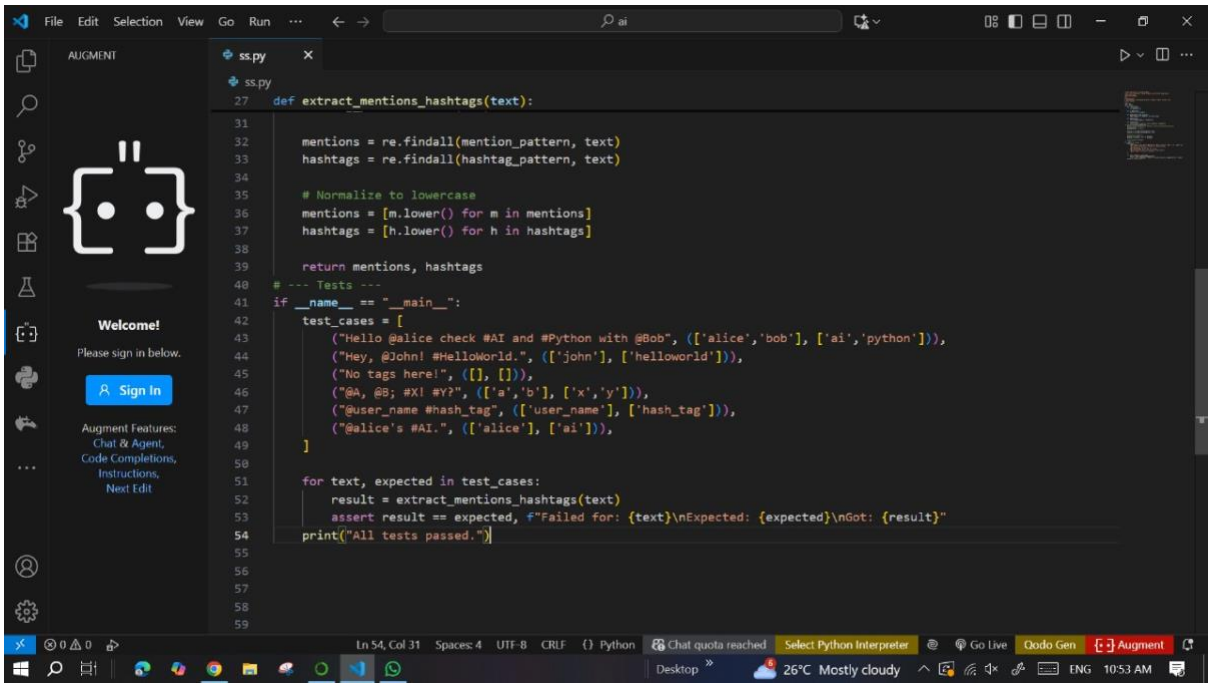
Task: Use AI to implement a priority queue using Python's heapq module.

Sample Input Code:

class PriorityQueue:

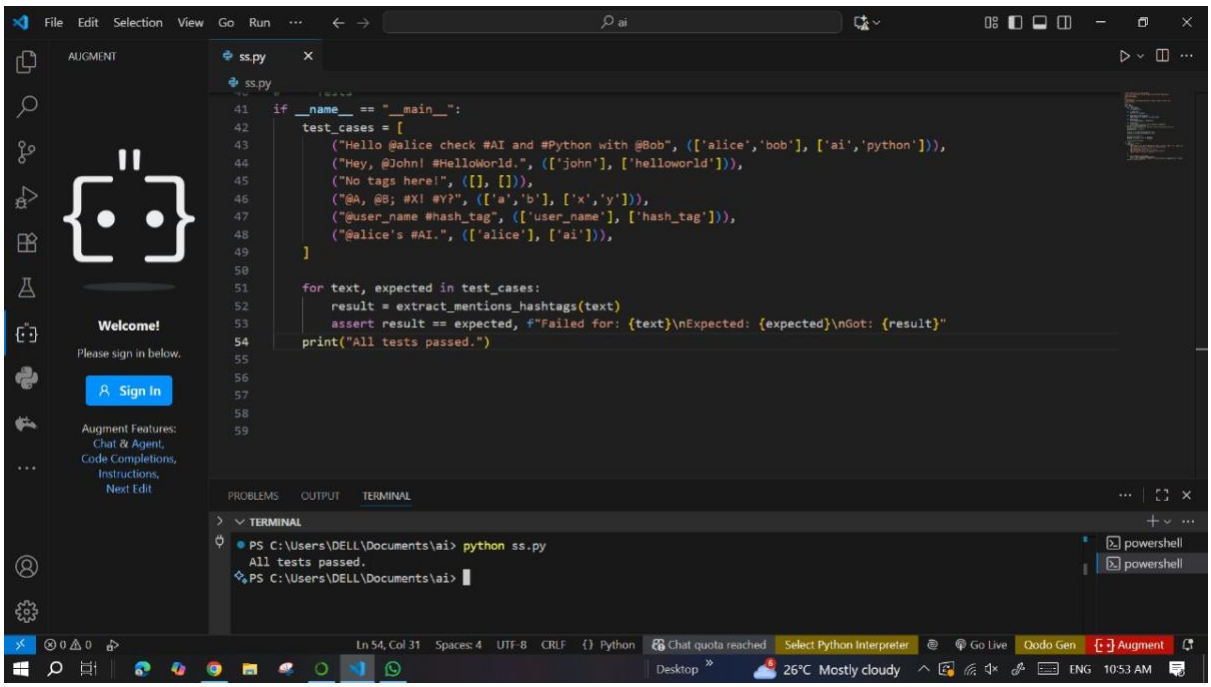
pass





Expected Output:

• Implementation with enqueue (priority), dequeue (highest priority), and display methods.



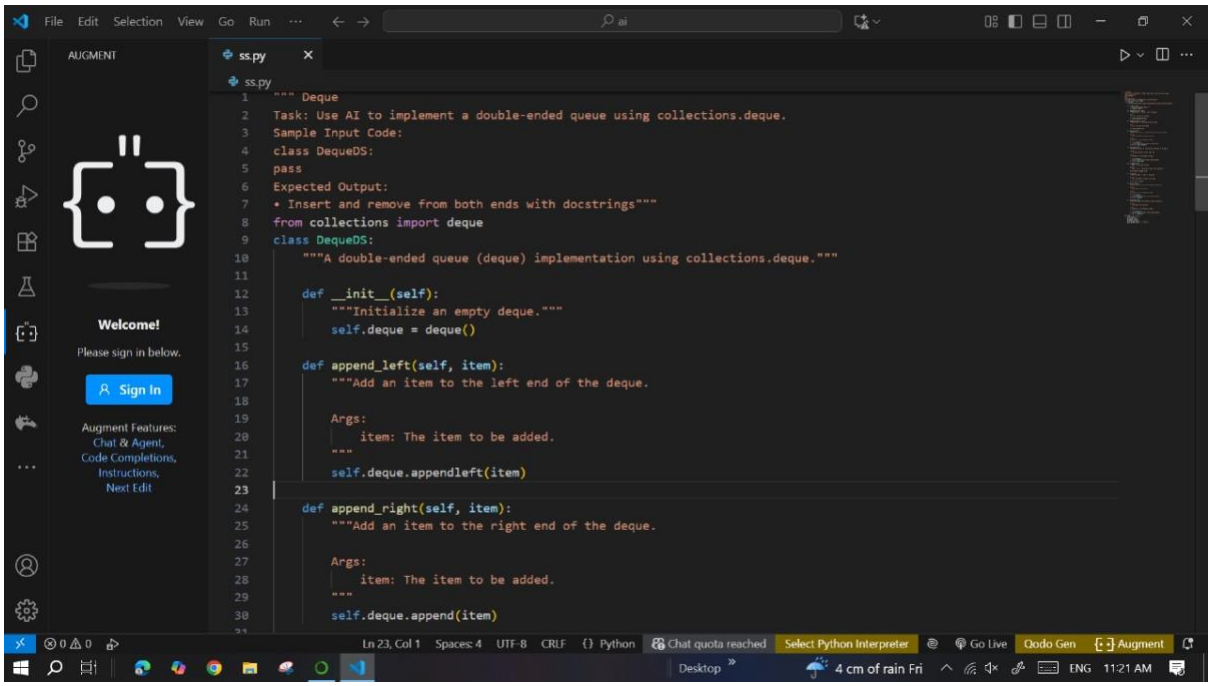
Task Description #8 – Deque

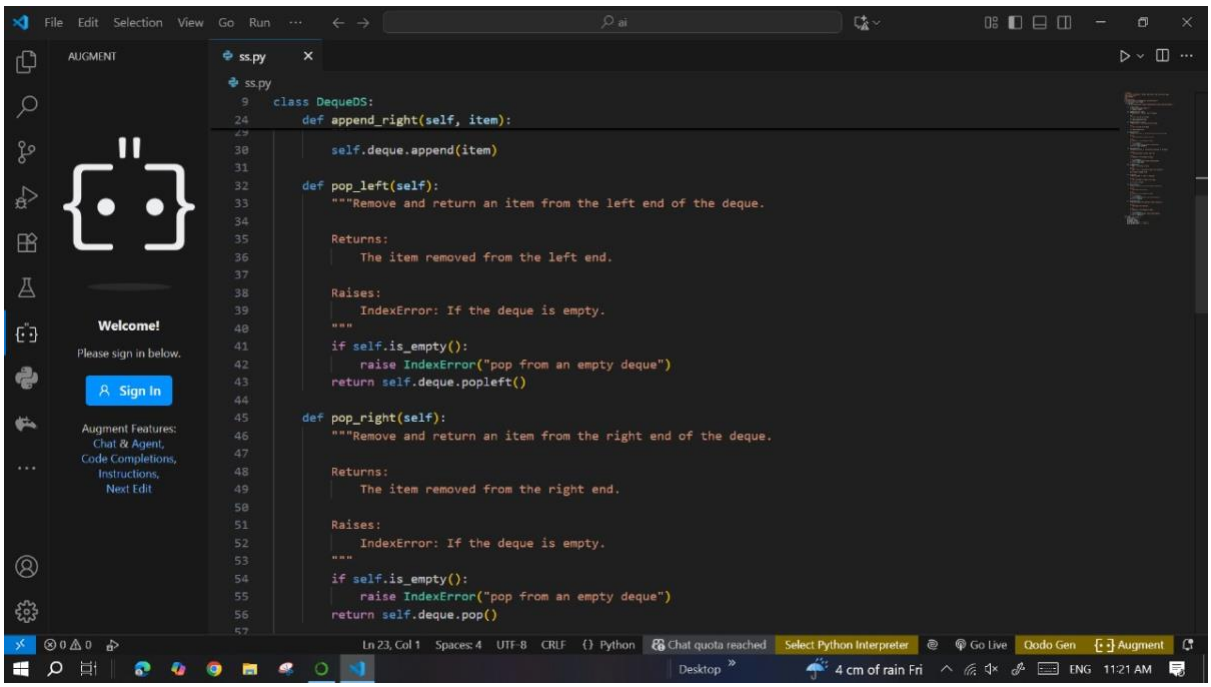
Task: Use AI to implement a double-ended queue using collections.deque.

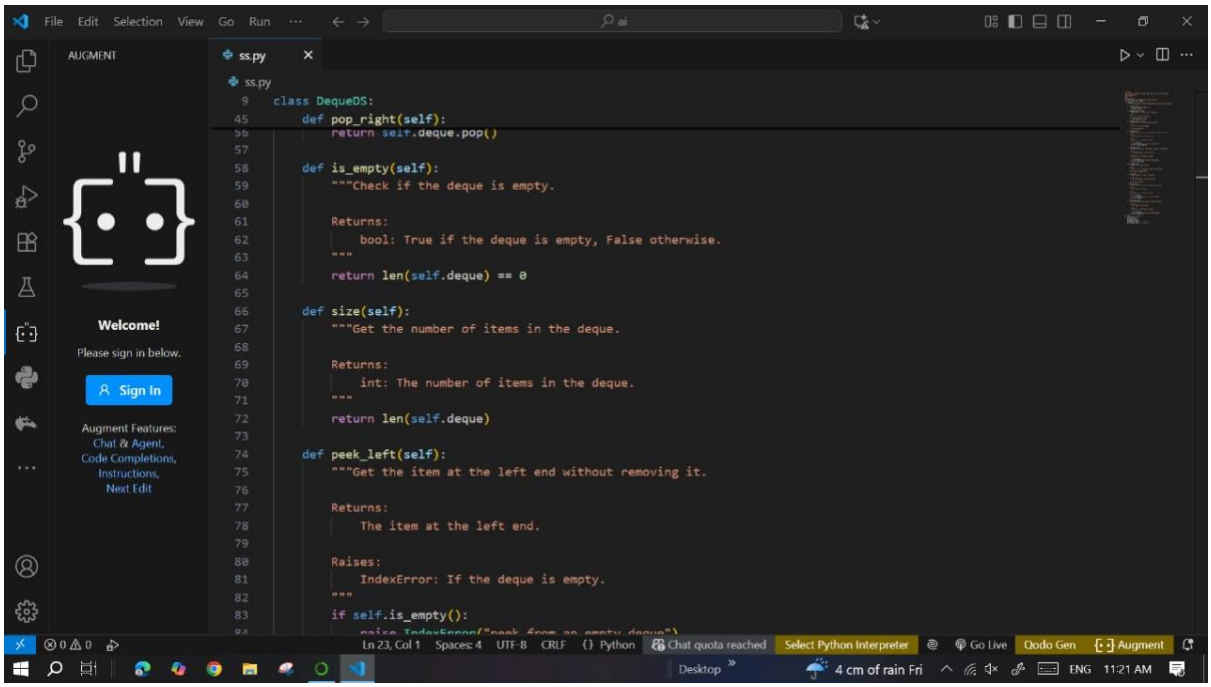
Sample Input Code:

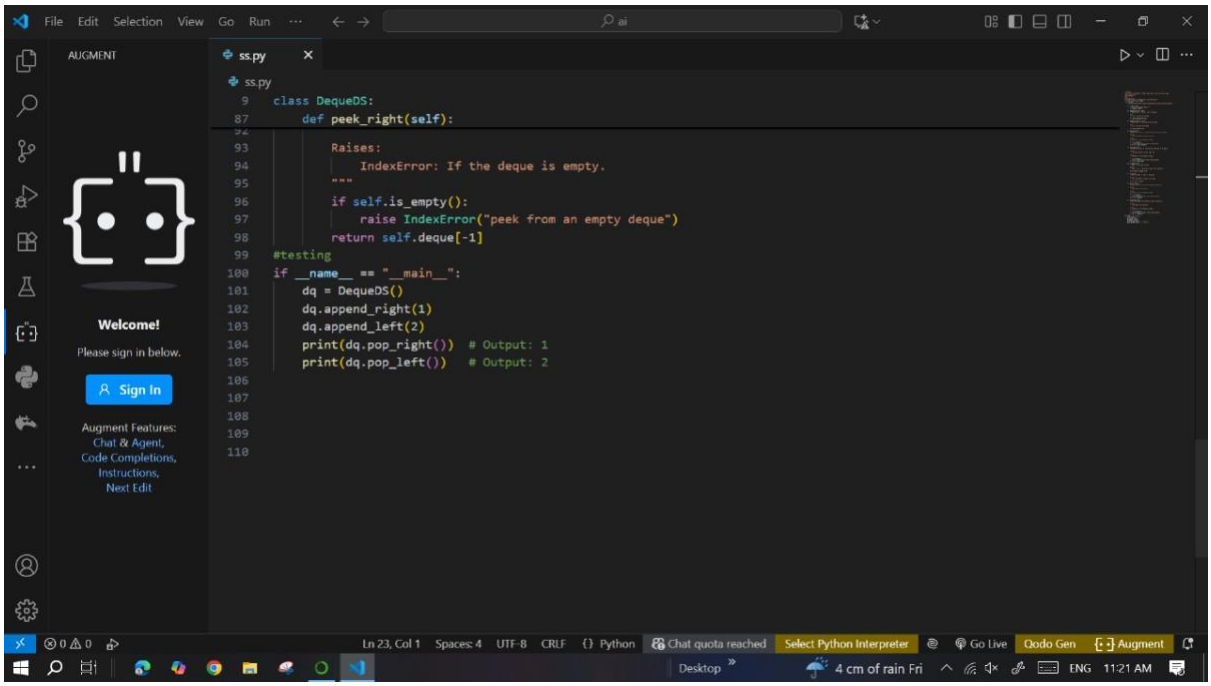
class DequeDS:

pass


File Edit Selection View Go Run
AUGMENT
ss.py
Welcome!
Please sign in below.
Sign In
Augment Features:
Chat & Agent,
Code Completions,
Instructions,
Next Edit
"""Deque
Task: Use AI to implement a double-ended queue using collections.deque.
Sample Input Code:
class DequeDS:
pass
Expected Output:
• Insert and remove from both ends with docstrings"""
from collections import deque
class DequeDS:
"""A double-ended queue (deque) implementation using collections.deque."""
def __init__(self):
"""Initialize an empty deque."""
self.deque = deque()
def append_left(self, item):
"""Add an item to the left end of the deque.
Args:
item: The item to be added.
"""
self.deque.appendleft(item)
def append_right(self, item):
"""Add an item to the right end of the deque.
Args:
item: The item to be added.
"""
self.deque.append(item)
Ln 23, Col 1 Spaces: 4 UTF-8 CRLF {} Python Chat quota reached Select Python Interpreter Go Live Qodo Gen Augment
Desktop
4 cm of rain Fri
ENG 11:21 AM

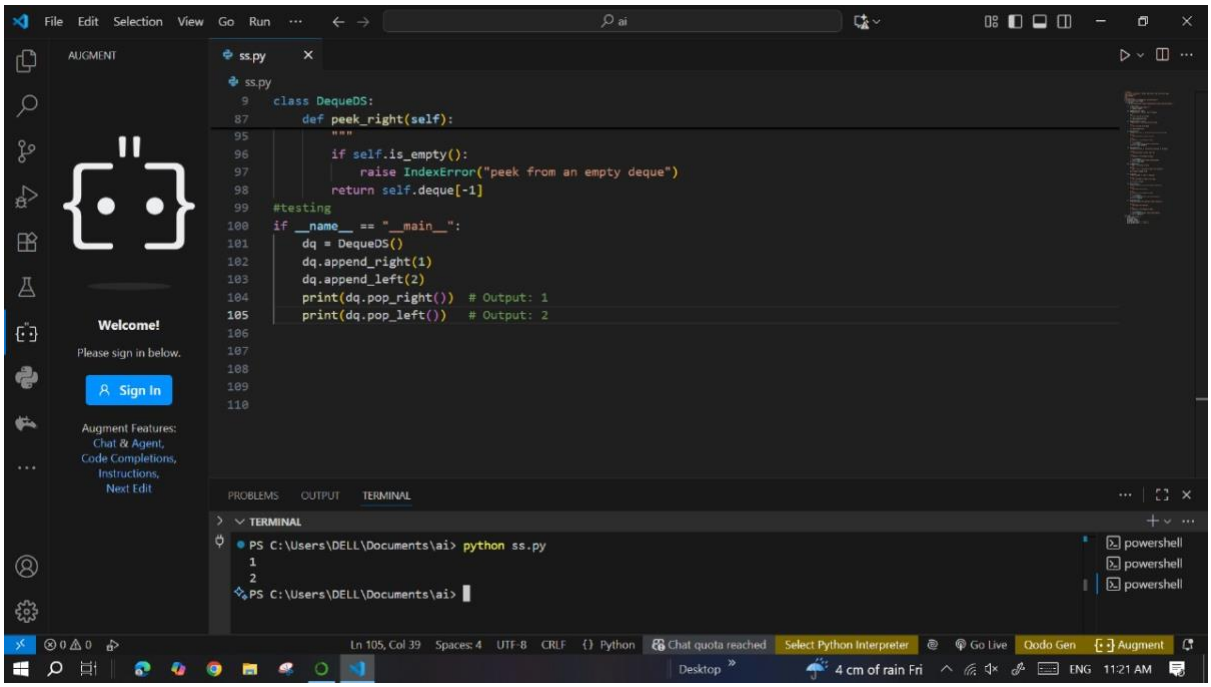

File Edit Selection View Go Run
AUGMENT
ss.py
Welcome!
Please sign in below.
Sign In
Augment Features:
Chat & Agent,
Code Completions,
Instructions,
Next Edit
class DequeDS:
def append_right(self, item):
self.deque.append(item)
def pop_left(self):
"""Remove and return an item from the left end of the deque.
Returns:
The item removed from the left end.
Raises:
IndexError: If the deque is empty.
"""
if self.is_empty():
raise IndexError("pop from an empty deque")
return self.deque.popleft()
def pop_right(self):
"""Remove and return an item from the right end of the deque.
Returns:
The item removed from the right end.
Raises:
IndexError: If the deque is empty.
"""
if self.is_empty():
raise IndexError("pop from an empty deque")
return self.deque.pop()
Ln 23, Col 1 Spaces: 4 UTF-8 CRLF {} Python Chat quota reached Select Python Interpreter Go Live Qodo Gen Augment
Desktop
4 cm of rain Fri
ENG 11:21 AM





Expected Output:

• Insert and remove from both ends with docstrings.

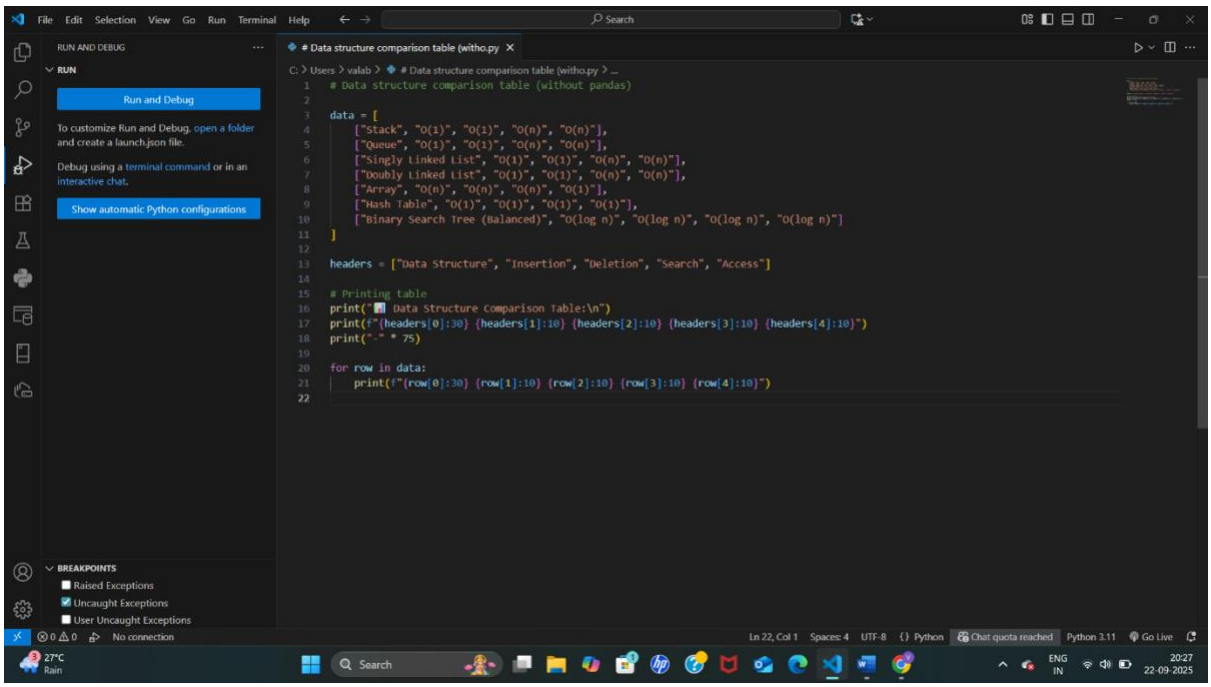


Task Description #9 – AI-Generated Data Structure Comparisons

Task: Use AI to generate a comparison table of different data structures (stack, queue, linked list, etc.) including time complexities.

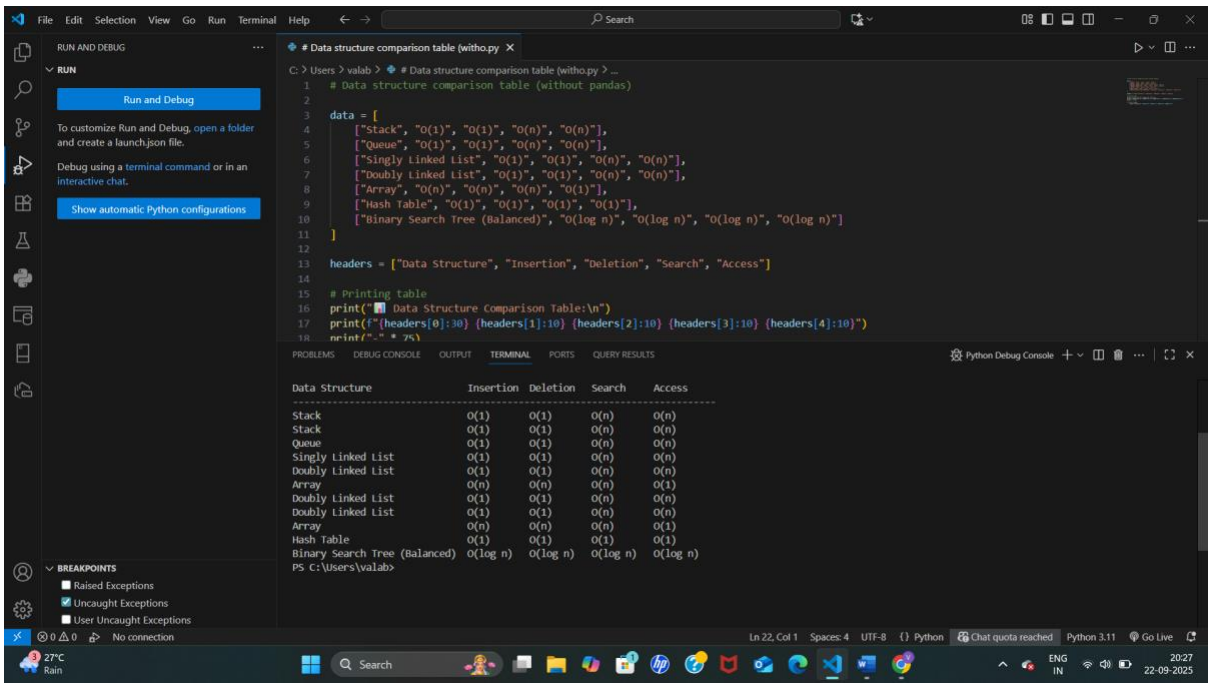
Sample Input Code:

 # No code, prompt AI for a data structure comparison table.



Expected Output:

• A markdown table with structure names, operations, and complexities.



Task Description #10 Real-Time Application Challenge – Choose the Right Data Structure

Scenario:

Your college wants to develop a Campus Resource Management System that handles:

1. Student Attendance Tracking – Daily log of students entering/exiting the campus.

2. Event Registration System – Manage participants in events with quick search and removal.

3. Library Book Borrowing – Keep track of available books and their due dates.

4. Bus Scheduling System – Maintain bus routes and stop connections.

5. Cafeteria Order Queue – Serve students in the order they arrive.

Student Task:

• For each feature, select the most appropriate data structure from the list below:

o Stack

o Queue

o Priority Queue

o Linked List

o Binary Search Tree (BST)

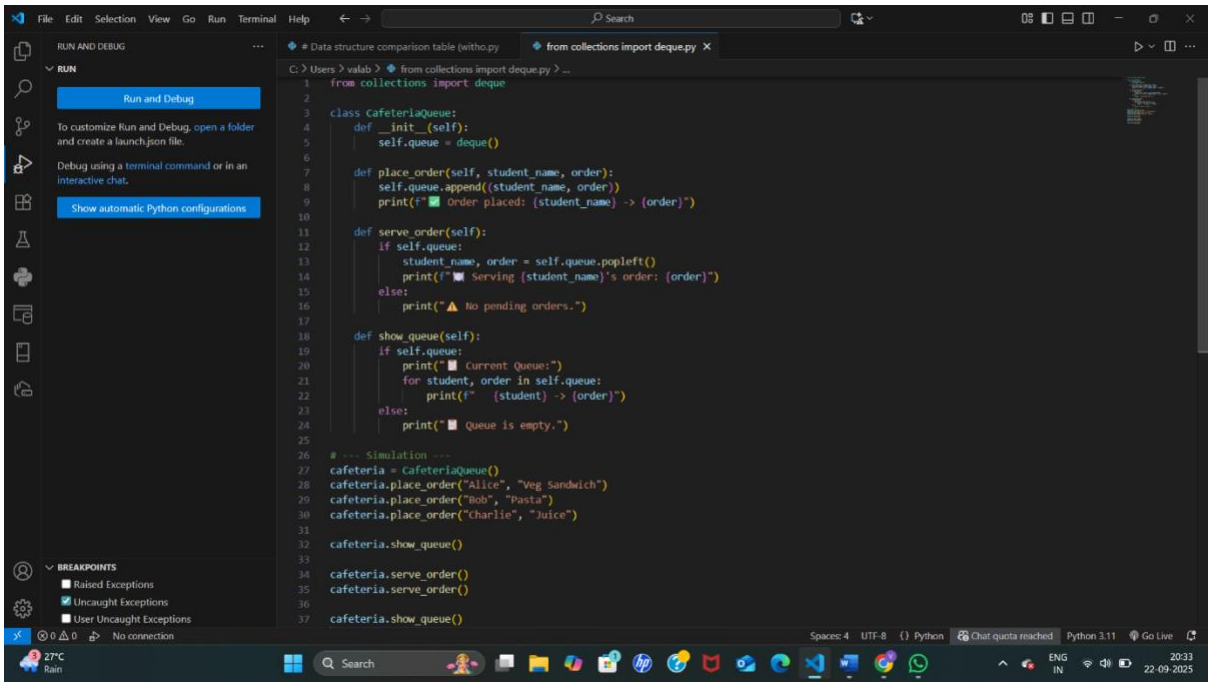
o Graph

o Hash Table

o Deque

• Justify your choice in 2–3 sentences per feature.

• Implement one selected feature as a working Python program with AI-assisted code generation.



Expected Output:

• A table mapping feature → chosen data structure → justification.

• A functional Python program implementing the chosen feature with

comments and docstrings.

Deliverables (For All Tasks)

1. AI-generated prompts for code and test case generation.

2. At least 3 assert test cases for each task.

3. AI-generated initial code and execution screenshots.

4. Analysis of whether code passes all tests.

5. Improved final version with inline comments and explanation.

6. Compiled report (Word/PDF) with prompts, test cases, assertions, code, and output.