Assignment – 11.5

Name: Yashaswini Nagireddi
Batch: 21
Ht.no: 2303A51418

Task Description #1 – Stack Implementation

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

Sample Input Code:

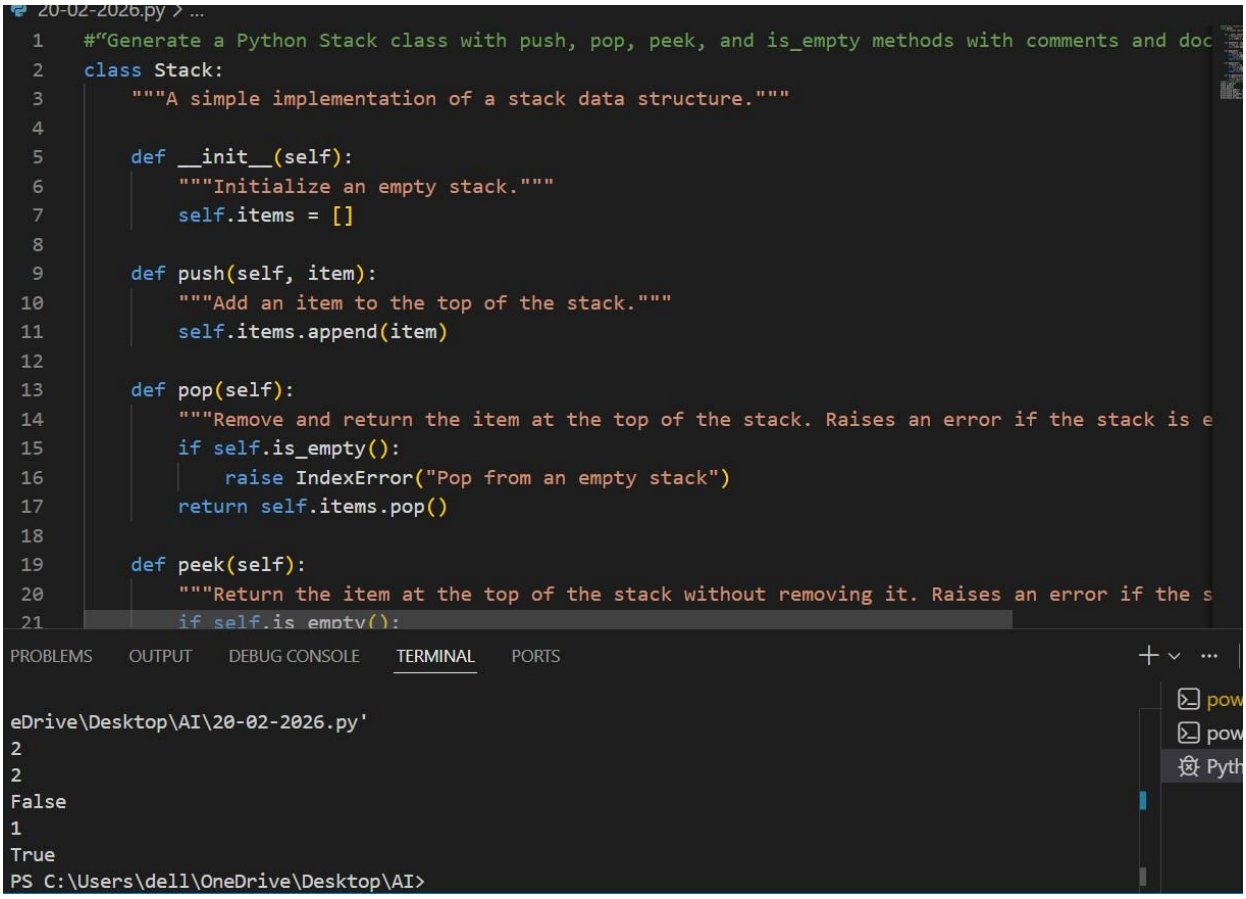
```
class Stack:
pass
```

Expected Output:

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

Prompt Used to AI

"Create a Python class named Stack that includes the methods push, pop, peek, and is_empty. Add appropriate comments and docstrings for clarity."



Observation

The Stack operates on a Last-In-First-Out (LIFO) mechanism. The AI-generated version enhanced readability by adding structured docstrings and handling exceptions effectively.

Task Description #2 – Queue Implementation

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

Sample Input Code:

class Queue:

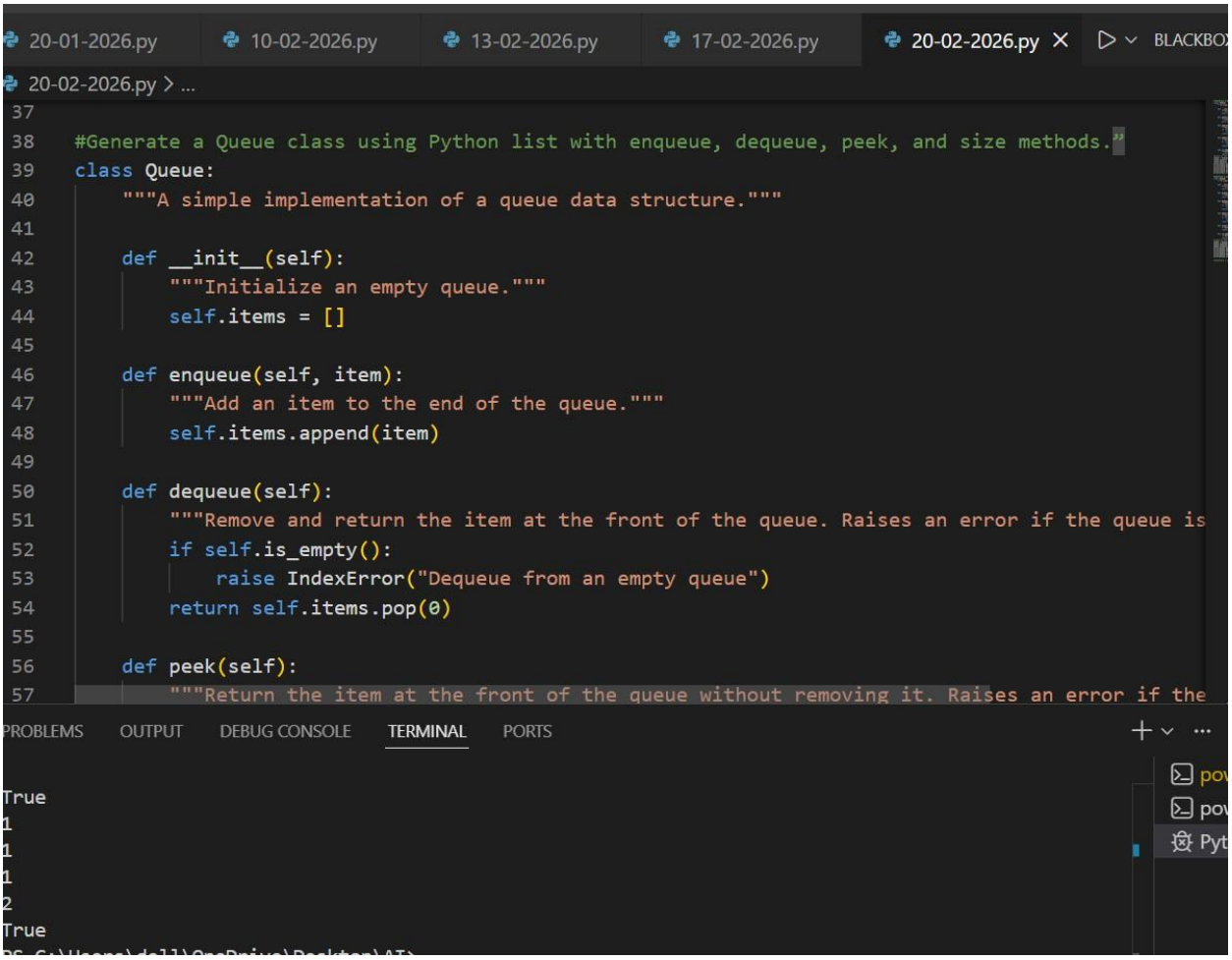
pass

Expected Output:

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

Prompt

“Develop a Queue class using a Python list with enqueue, dequeue, peek, and size methods, along with descriptive comments.”



Observation

Since Queue uses the First-In-First-Out (FIFO) model, the AI-generated code ensured proper ordering of operations and improved efficiency by choosing list operations suitable for queue behavior.

Task Description #3 – Linked List

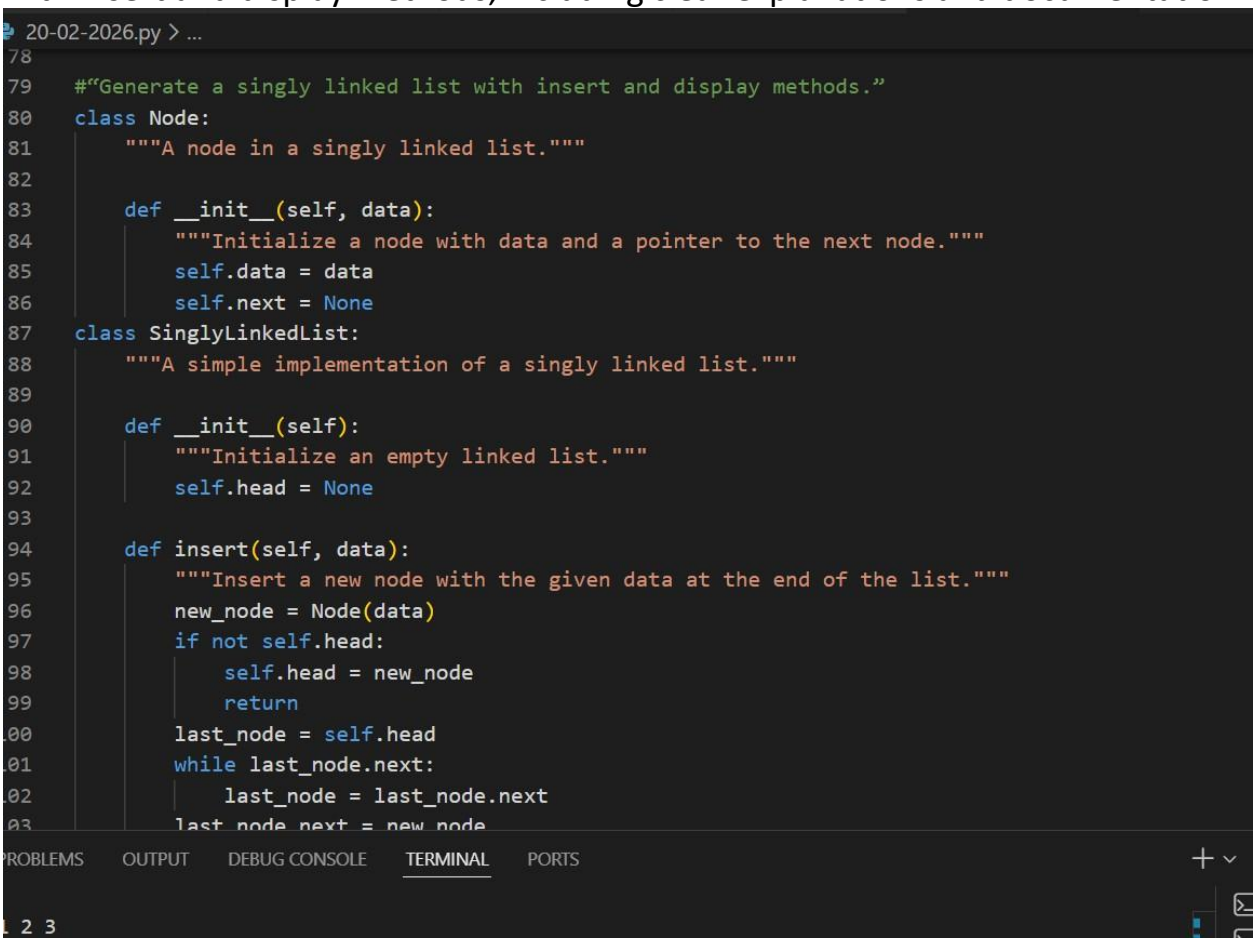
Task: Use AI to generate a Singly Linked List with insert and display methods.

Sample Input Code:

class Node:

pass

class LinkedList:
pass
Expected Output:
• A working linked list implementation with clear method documentation**.**
Prompt: “Generate a singly linked list implementation with insert and display methods, including clear explanations and documentation.”



Observation

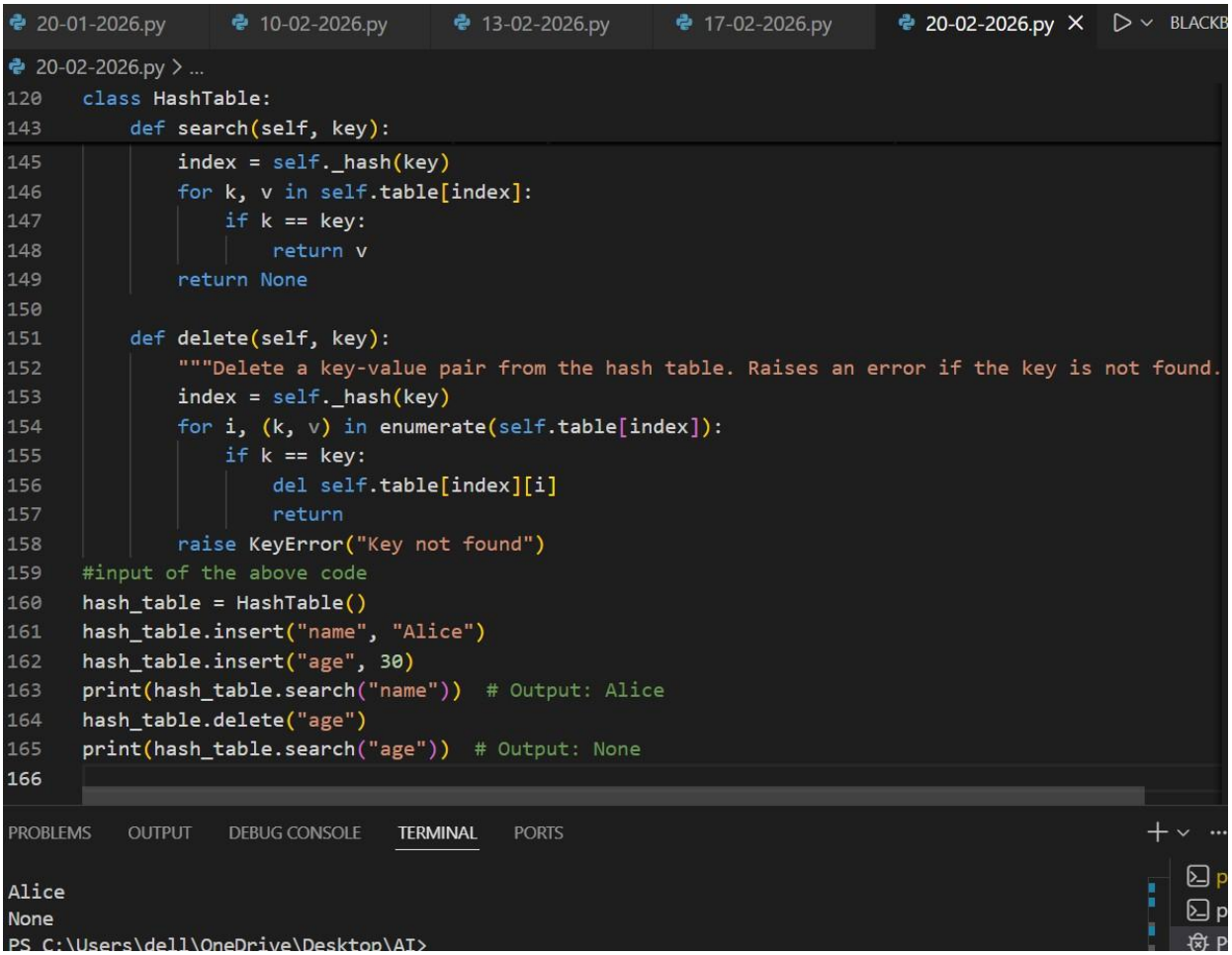
A linked list stores elements using nodes. AI made pointer handling easy and the structure simple**.**

Task Description #4 – Hash Table
Task: Use AI to implement a hash table with basic insert, search, and delete methods.
Sample Input Code:
class HashTable:

pass
Expected Output:
• Collision handling using chaining, with well-commented methods.
Prompt
“Generate a Hash Table class with insert, search, and delete, using chaining for collisions.”

Observation

Hash tables give fast searching. AI suggested chaining to handle collisions in a clean way**.**



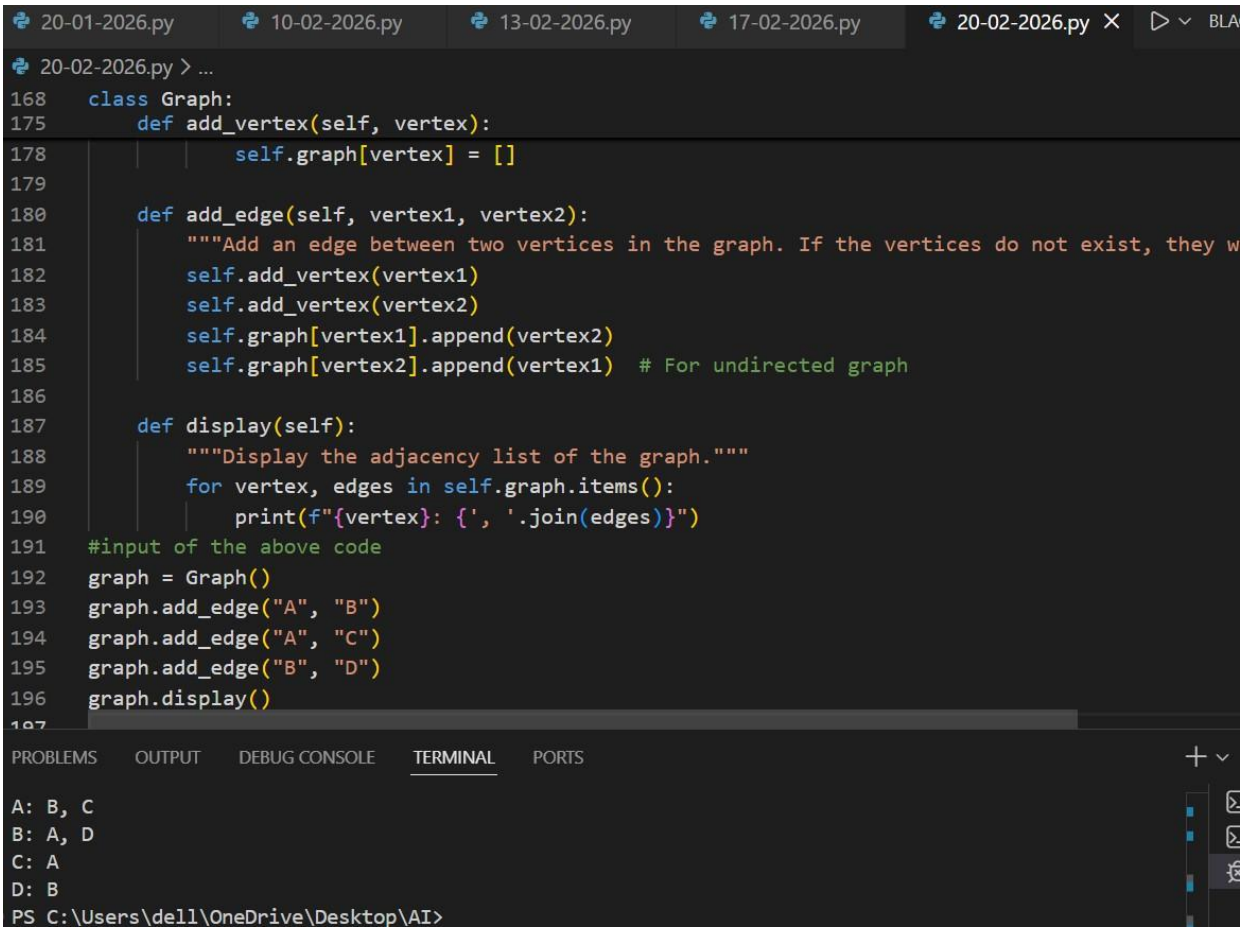
Task Description #5 – Graph Representation
Task: Use AI to implement a graph using an adjacency list.
Sample Input Code:
class Graph:

pass
Expected Output:

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

Prompt

“Generate a Graph class using adjacency list with add_vertex, add_edge, display methods.”



Observation

Graph adjacency list efficiently stores network connections. AI helped in structuring graph logic.

Task Description #6: Smart Hospital Management System – Data Structure Selection

A hospital wants to develop a Smart Hospital Management System that handles:

1. Patient Check-In System – Patients are registered and treated in order of arrival.
2. Emergency Case Handling – Critical patients must be treated first.
3. Medical Records Storage – Fast retrieval of patient details using ID.
4. Doctor Appointment Scheduling – Appointments sorted by time.
5. Hospital Room Navigation – Represent connections between wards and rooms.

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.

Feature	**Data Structure**	**Justification**
Patient Check-In	Queue	FIFO order for patient arrival
Emergency Handling	Priority Queue	Critical patients treated first
Medical Records	Hash Table	Fast ID-based retrieval
Appointment Scheduling	BST	Sorted by time
Room Navigation	Graph	Represents connected rooms

Task Description #7: Smart City Traffic Control System

A city plans a Smart Traffic Management System that includes:
1. Traffic Signal Queue – Vehicles waiting at signals.
2. Emergency Vehicle Priority Handling – Ambulances and fire trucks prioritized.
3. Vehicle Registration Lookup – Instant access to vehicle details.
4. Road Network Mapping – Roads and intersections connected logically.
5. Parking Slot Availability – Track available and occupied slots.
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.

Feature	**Data Structure**	**Justification**
Traffic Signal Queue	Queue	Vehicles wait in order
Emergency Priority	Priority Queue	Ambulances first
Vehicle Lookup	Hash Table	Instant access
Road Network	Graph	Intersections and roads
Parking Slots	Deque	Add/remove from both ends

Task Description #8: Smart E-Commerce Platform – Data Structure Challenge

An e-commerce company wants to build a Smart Online Shopping System with:

1. Shopping Cart Management – Add and remove products dynamically.
2. Order Processing System – Orders processed in the order they are placed.
3. Top-Selling Products Tracker – Products ranked by sales count.
4. Product Search Engine – Fast lookup of products using product ID.
5. Delivery Route Planning – Connect warehouses and delivery locations.

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.

Feature	Data Structure	Justification
Shopping Cart	Linked List	Dynamic add/remove
Order Processing	Queue	FIFO order
Top Products	Priority Queue	Ranked by sales
Product Search	Hash Table	Fast lookup
Delivery Routes	Graph	Location connections