ASSIGNMENT-11.1

2303A510E3

BATCH-14

Task Description #1 – Stack Implementation

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

methods.

Sample Input Code:

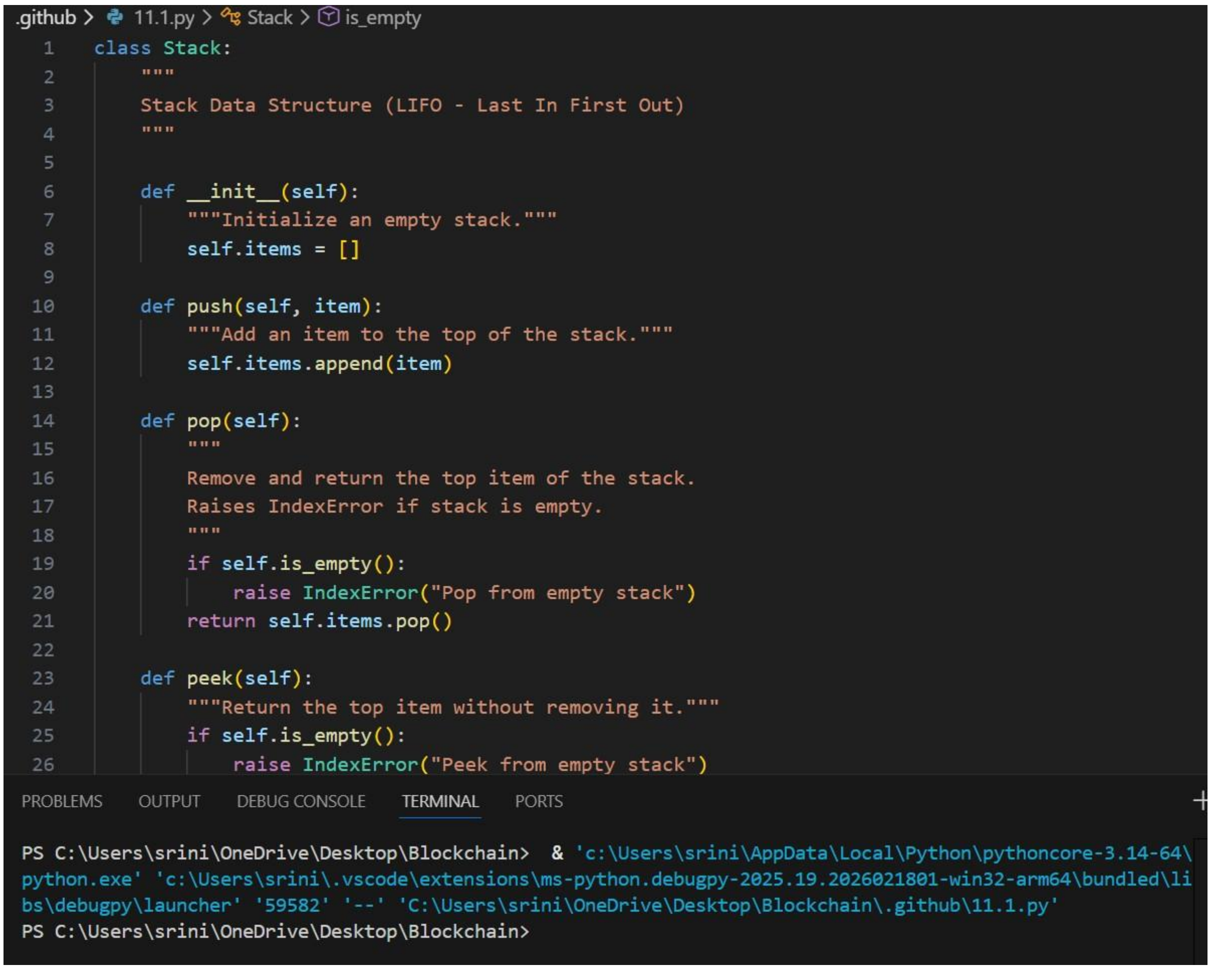
class Stack:

pass

Expected Output:

• A functional stack implementation with all required methods and

docstrings.



.github > 11.1.py > Stack > is_empty

```python
class Stack:
    """
    Stack Data Structure (LIFO - Last In First Out)
    """

    def __init__(self):
        """Initialize an empty stack."""
        self.items = []

    def push(self, item):
        """Add an item to the top of the stack."""
        self.items.append(item)

    def pop(self):
        """
        Remove and return the top item of the stack.
        Raises IndexError if stack is empty.
        """
        if self.is_empty():
            raise IndexError("Pop from empty stack")
        return self.items.pop()

    def peek(self):
        """Return the top item without removing it."""
        if self.is_empty():
            raise IndexError("Peek from empty stack")
```

```
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  & 'c:\Users\srini\AppData\Local\Python\pythoncore-3.14-64\
python.exe' 'c:\Users\srini\.vscode\extensions\ms-python.debugpy-2025.19.2026021801-win32-arm64\bundled\li
bs\debugpy\launcher' '59582' '--' 'C:\Users\srini\OneDrive\Desktop\Blockchain\.github\11.1.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
```

Task Description #2 – Queue Implementation

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

Sample Input Code:

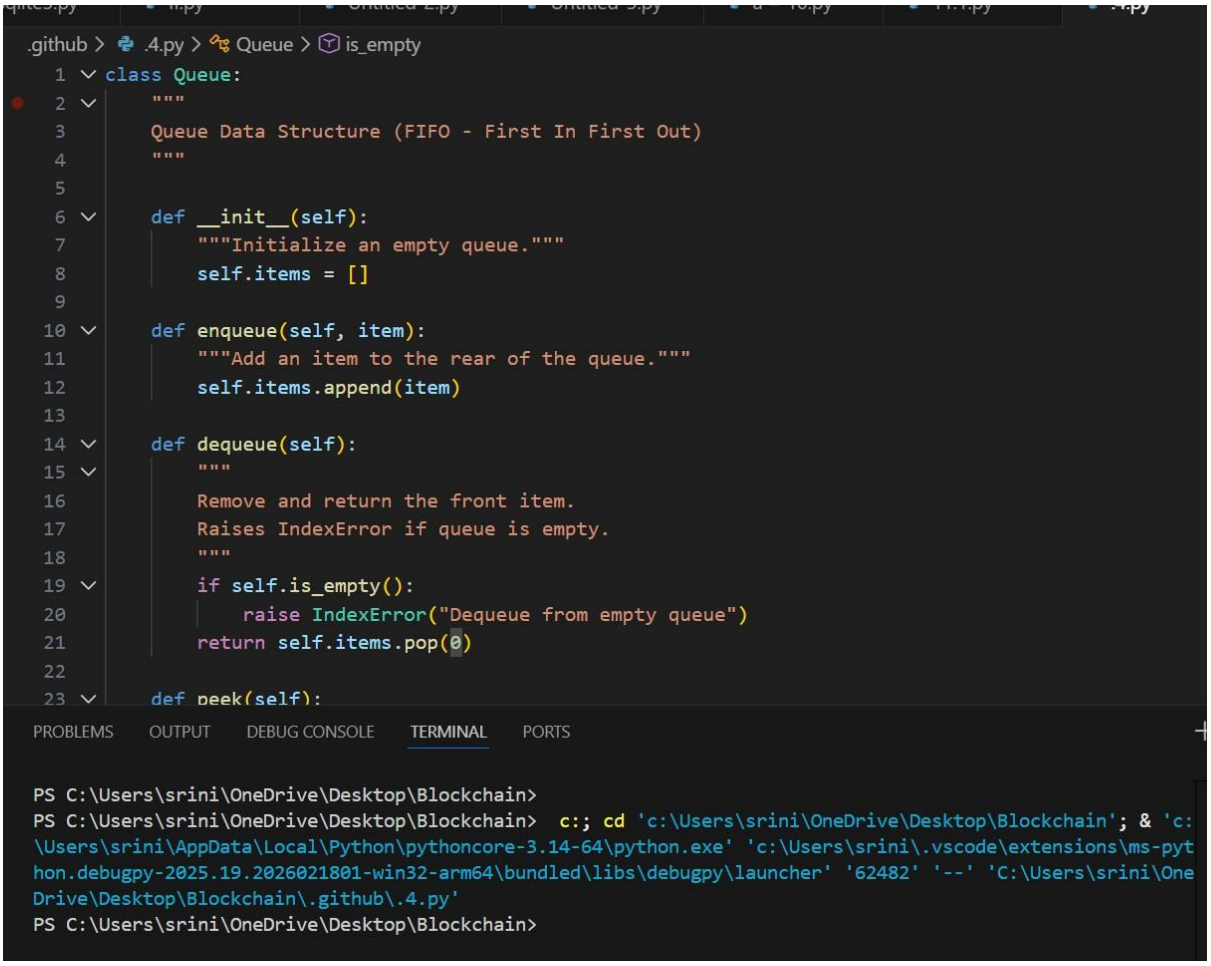
class Queue:

pass

Expected Output:

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

methods.



```python
class Queue:
    """
    Queue Data Structure (FIFO - First In First Out)
    """

    def __init__(self):
        """Initialize an empty queue."""
        self.items = []

    def enqueue(self, item):
        """Add an item to the rear of the queue."""
        self.items.append(item)

    def dequeue(self):
        """
        Remove and return the front item.
        Raises IndexError if queue is empty.
        """
        if self.is_empty():
            raise IndexError("Dequeue from empty queue")
        return self.items.pop(0)

    def peek(self):
```

```
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '62482' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\.4.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
```

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

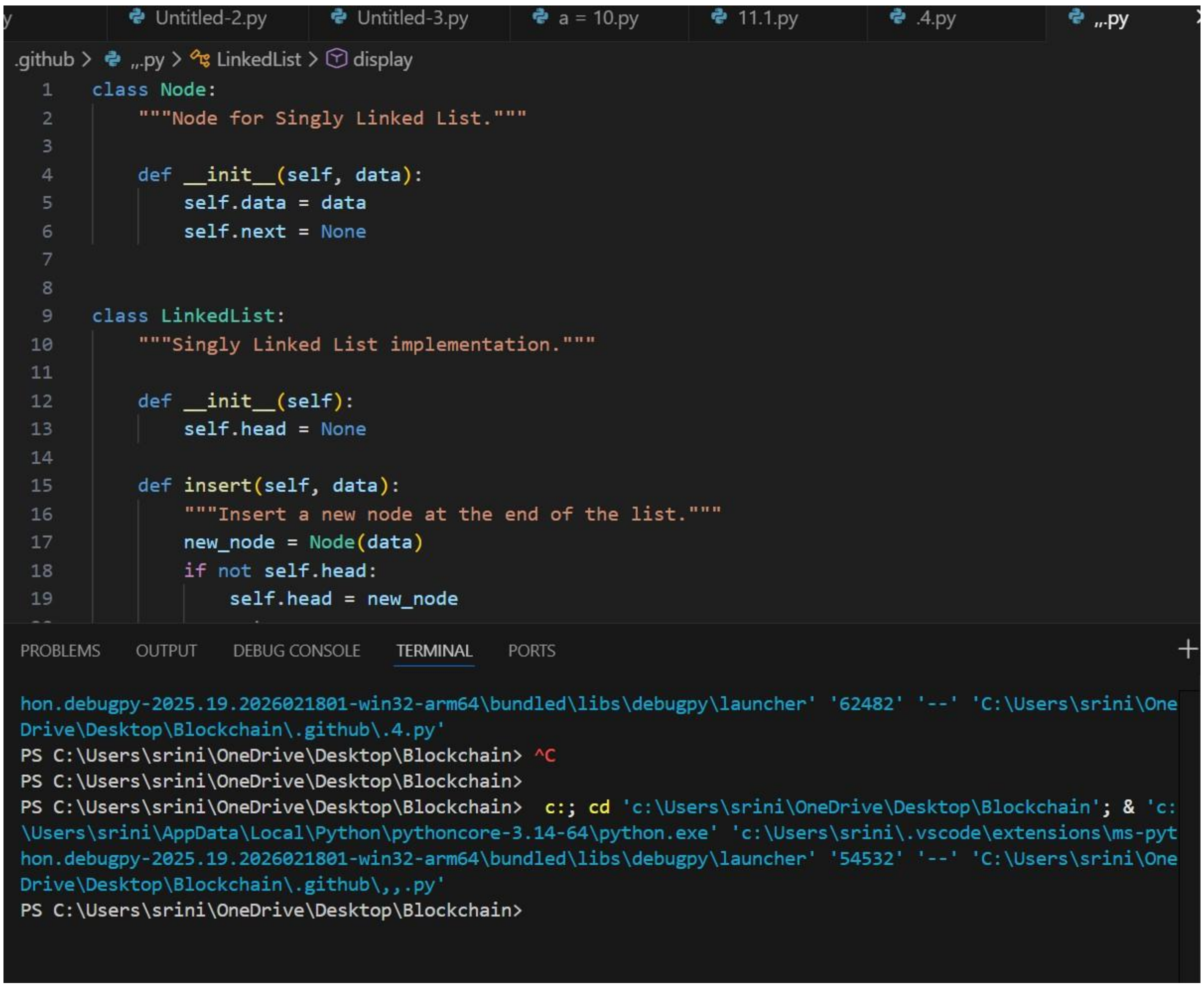
class LinkedList:

pass

Expected Output:

• A working linked list implementation with clear method

documentation.



```python
class Node:
    """Node for Singly Linked List."""

    def __init__(self, data):
        self.data = data
        self.next = None


class LinkedList:
    """Singly Linked List implementation."""

    def __init__(self):
        self.head = None

    def insert(self, data):
        """Insert a new node at the end of the list."""
        new_node = Node(data)
        if not self.head:
            self.head = new_node
```

```
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '62482' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\.4.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '54532' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\,,.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
```

Task Description #4 – Binary Search Tree (BST)

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

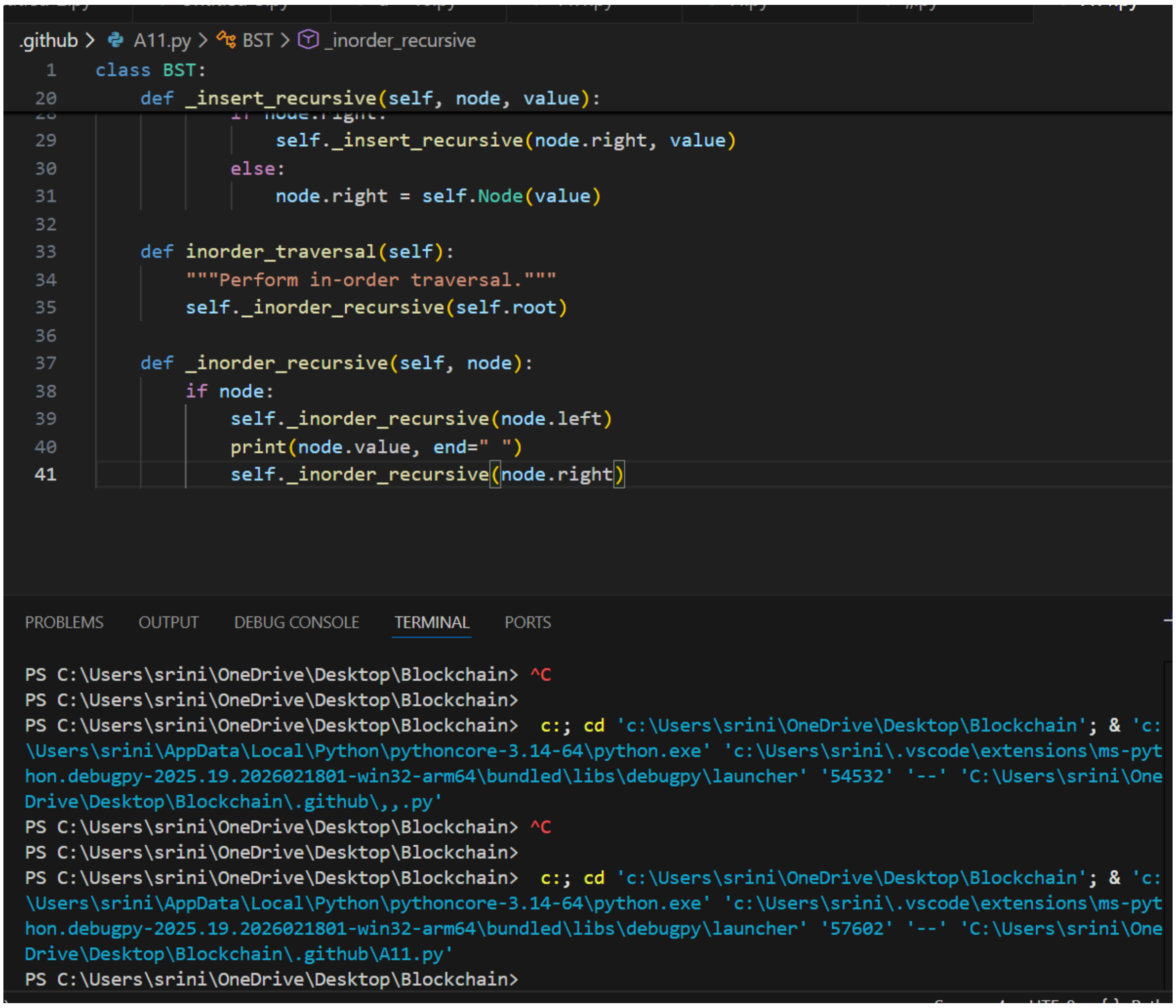
Sample Input Code:

class BST:

pass

Expected Output:

• BST implementation with recursive insert and traversal methods



```python
class BST:
    def _insert_recursive(self, node, value):
                self._insert_recursive(node.right, value)
            else:
                node.right = self.Node(value)

    def inorder_traversal(self):
        """Perform in-order traversal."""
        self._inorder_recursive(self.root)

    def _inorder_recursive(self, node):
        if node:
            self._inorder_recursive(node.left)
            print(node.value, end=" ")
            self._inorder_recursive(node.right)
```

```
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '54532' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\,,.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '57602' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\A11.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
```

Task Description #5 – Hash Table

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

delete methods.

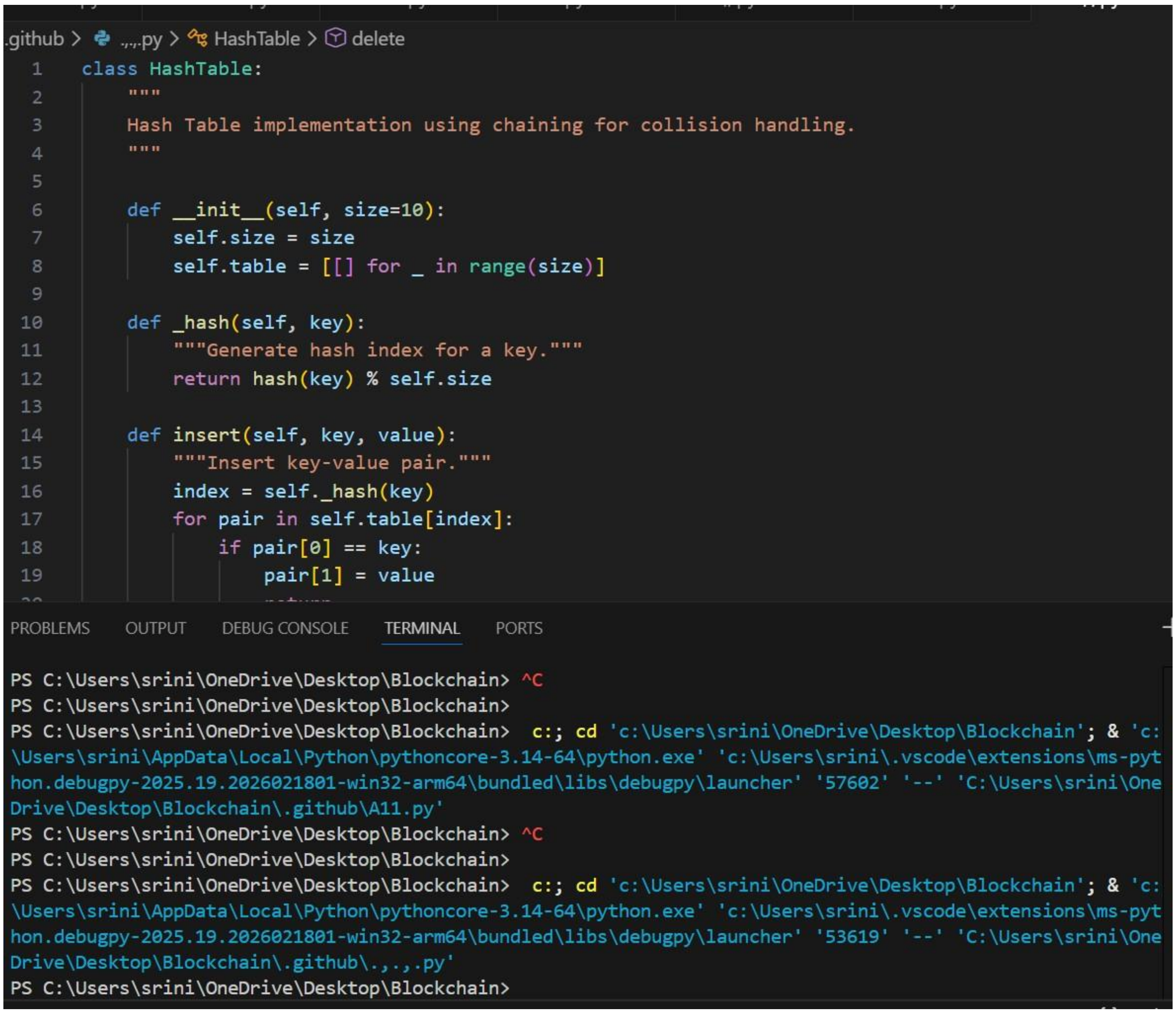
Sample Input Code:

class HashTable:

pass

Expected Output:

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



```python
class HashTable:
    """
    Hash Table implementation using chaining for collision handling.
    """

    def __init__(self, size=10):
        self.size = size
        self.table = [[] for _ in range(size)]

    def _hash(self, key):
        """Generate hash index for a key."""
        return hash(key) % self.size

    def insert(self, key, value):
        """Insert key-value pair."""
        index = self._hash(key)
        for pair in self.table[index]:
            if pair[0] == key:
                pair[1] = value
```

```
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '57602' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\A11.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '53619' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\.,.,.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
```

Task Description #6 – Graph Representation

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

Sample Input Code:

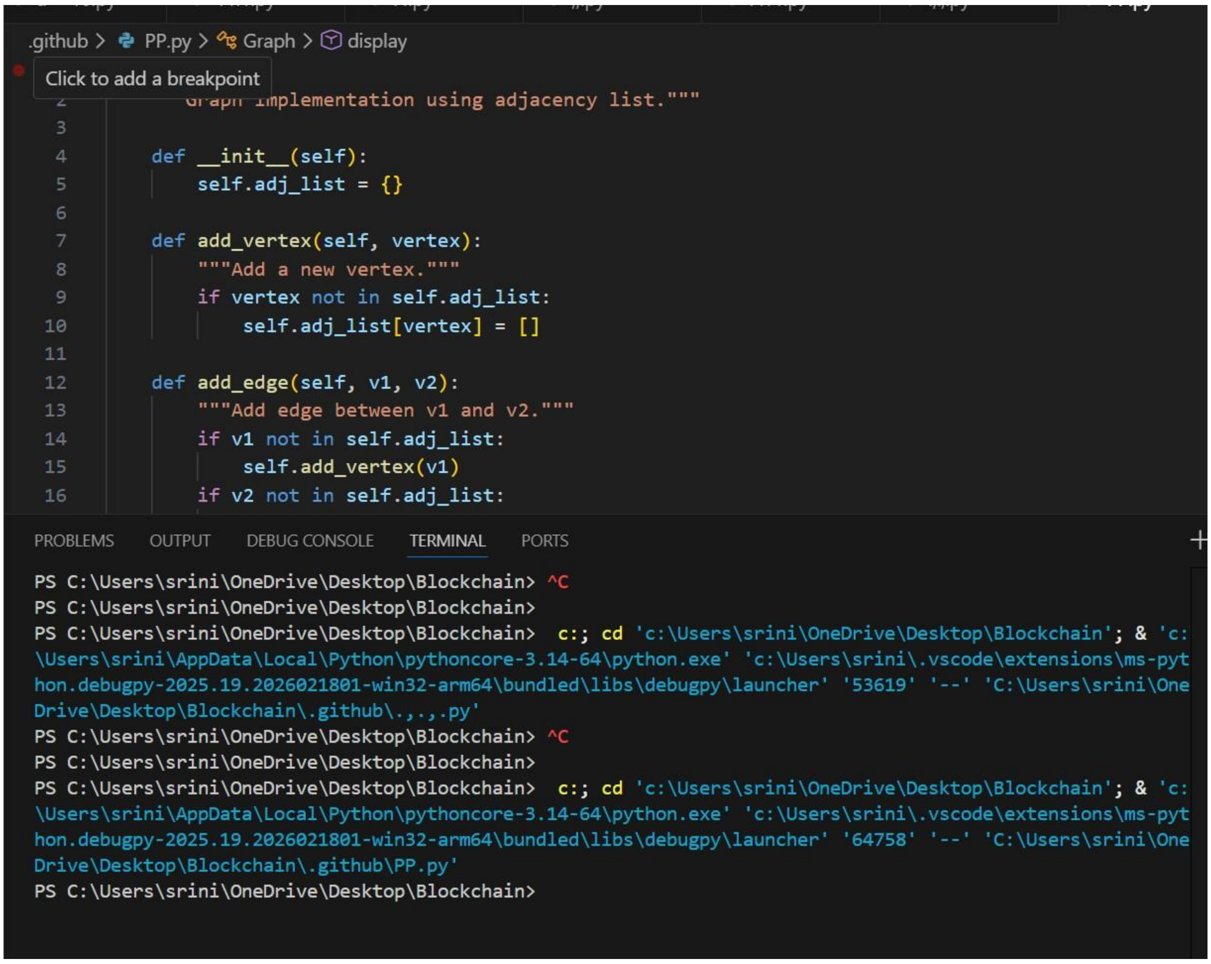
class Graph:

pass

Expected Output:

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

connections.



```python
    Graph implementation using adjacency list."""

    def __init__(self):
        self.adj_list = {}

    def add_vertex(self, vertex):
        """Add a new vertex."""
        if vertex not in self.adj_list:
            self.adj_list[vertex] = []

    def add_edge(self, v1, v2):
        """Add edge between v1 and v2."""
        if v1 not in self.adj_list:
            self.add_vertex(v1)
        if v2 not in self.adj_list:
```

```
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '53619' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\.,.,.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '64758' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\PP.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
```

Task Description #7 – Priority Queue

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

module.

Sample Input Code:

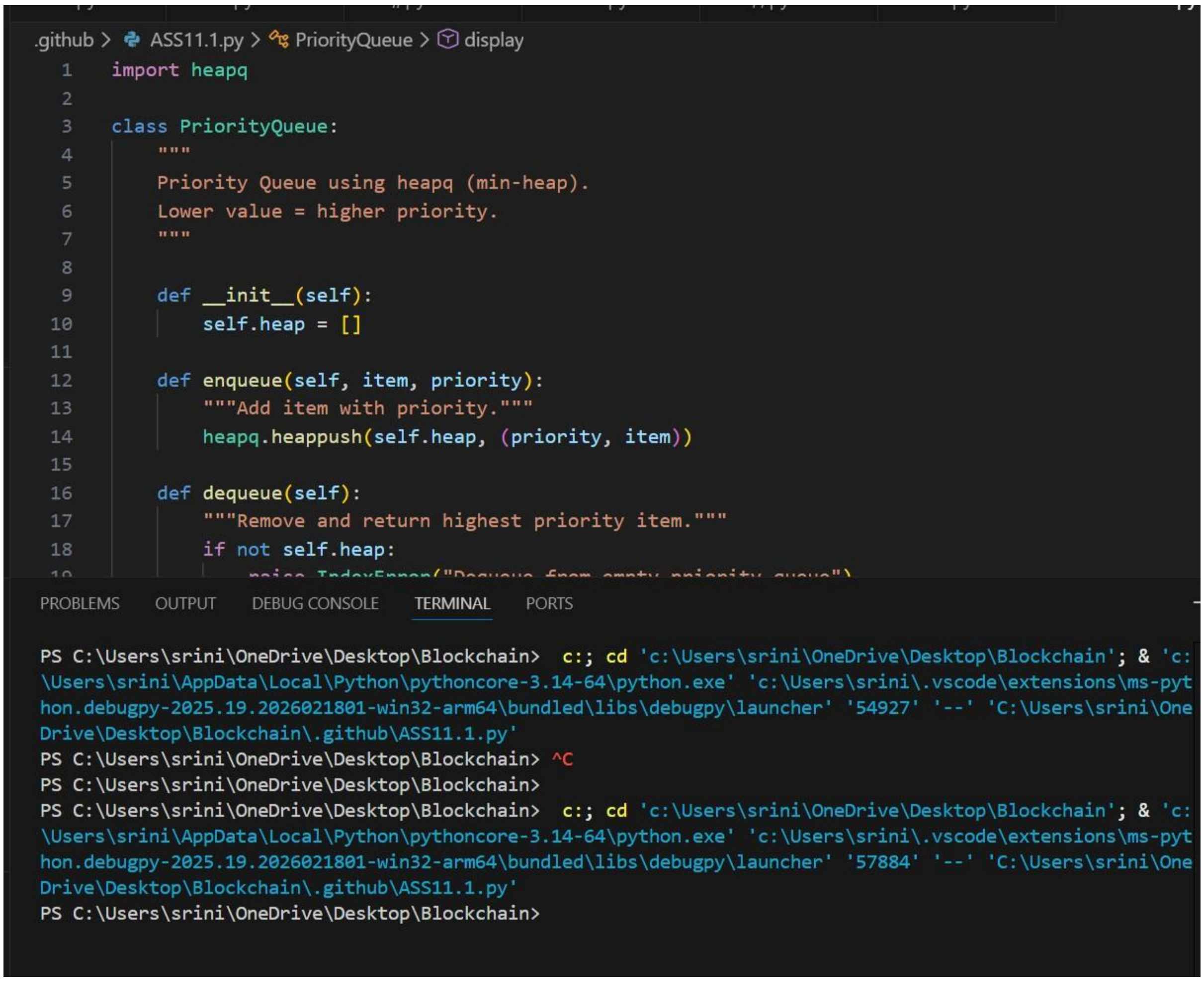
class PriorityQueue:

pass

Expected Output:

• Implementation with enqueue (priority), dequeue (highest priority),

and display methods.



.github > ASS11.1.py > PriorityQueue > display

```python
import heapq

class PriorityQueue:
    """
    Priority Queue using heapq (min-heap).
    Lower value = higher priority.
    """

    def __init__(self):
        self.heap = []

    def enqueue(self, item, priority):
        """Add item with priority."""
        heapq.heappush(self.heap, (priority, item))

    def dequeue(self):
        """Remove and return highest priority item."""
        if not self.heap:
            raise IndexError("Dequeue from empty priority queue")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '54927' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\ASS11.1.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '57884' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\ASS11.1.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
```

Task Description #8 – Deque

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

collections.deque.

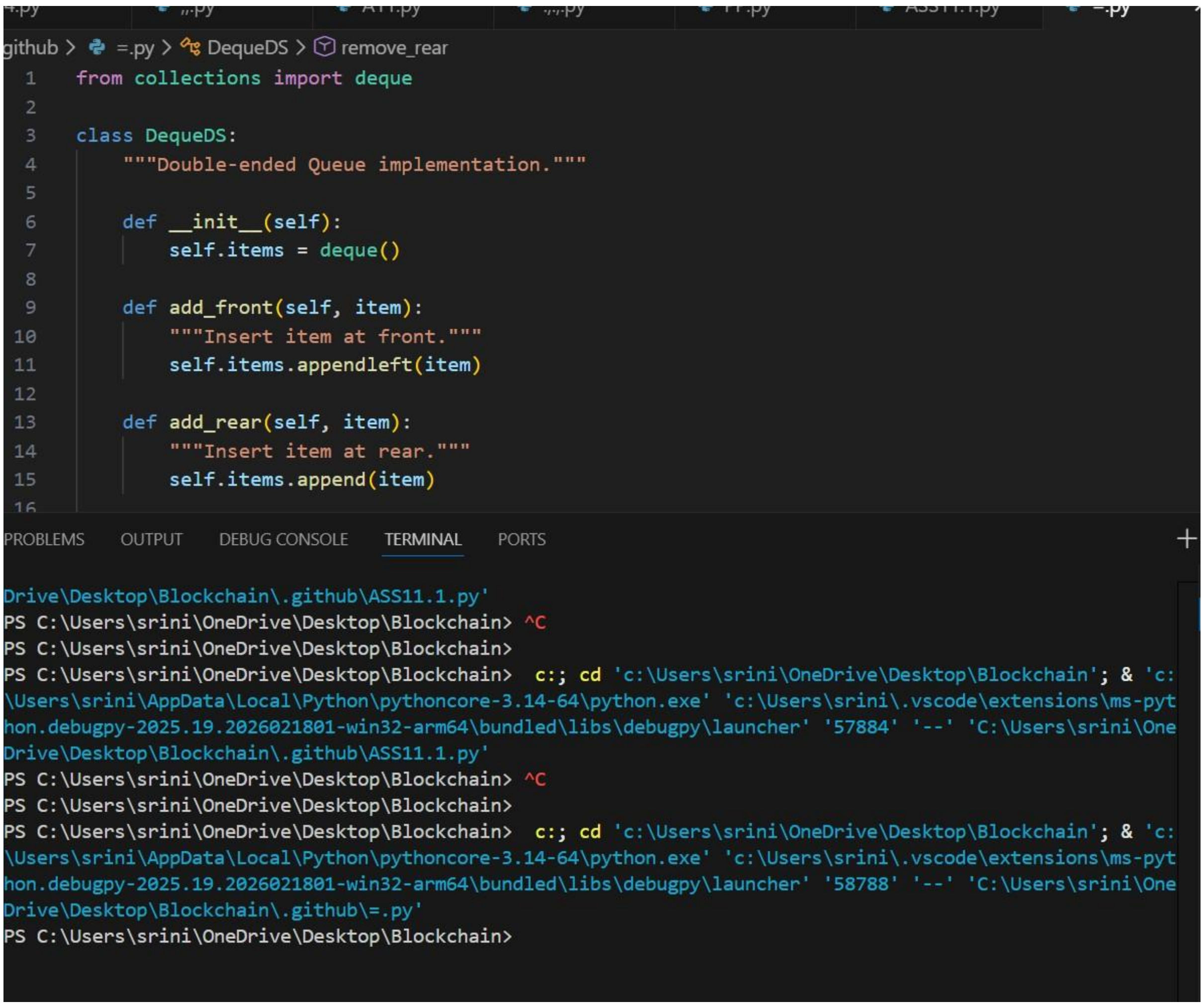
Sample Input Code:

class DequeDS:

pass

Expected Output:

• Insert and remove from both ends with docstrings.



```python
from collections import deque

class DequeDS:
    """Double-ended Queue implementation."""

    def __init__(self):
        self.items = deque()

    def add_front(self, item):
        """Insert item at front."""
        self.items.appendleft(item)

    def add_rear(self, item):
        """Insert item at rear."""
        self.items.append(item)
```

```
Drive\Desktop\Blockchain\.github\ASS11.1.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '57884' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\ASS11.1.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '58788' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\=.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
```

Task Description #9 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

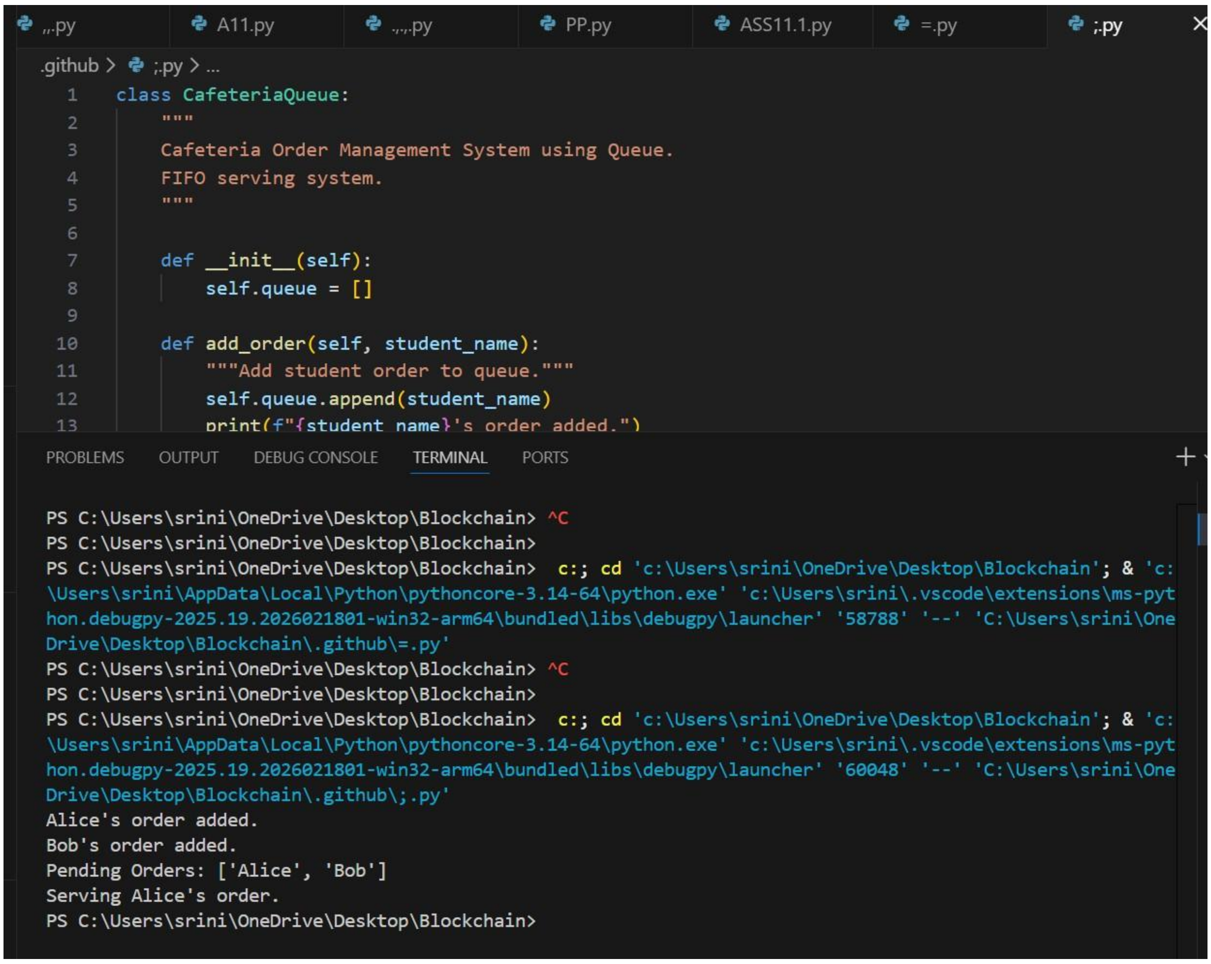
• Justify your choice in 2–3 sentences per feature.

• 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.



```python
class CafeteriaQueue:
    """
    Cafeteria Order Management System using Queue.
    FIFO serving system.
    """

    def __init__(self):
        self.queue = []

    def add_order(self, student_name):
        """Add student order to queue."""
        self.queue.append(student_name)
        print(f"{student_name}'s order added.")
```

```
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '58788' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\=.py'
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '60048' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\;.py'
Alice's order added.
Bob's order added.
Pending Orders: ['Alice', 'Bob']
Serving Alice's order.
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
```

Task Description #10: 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

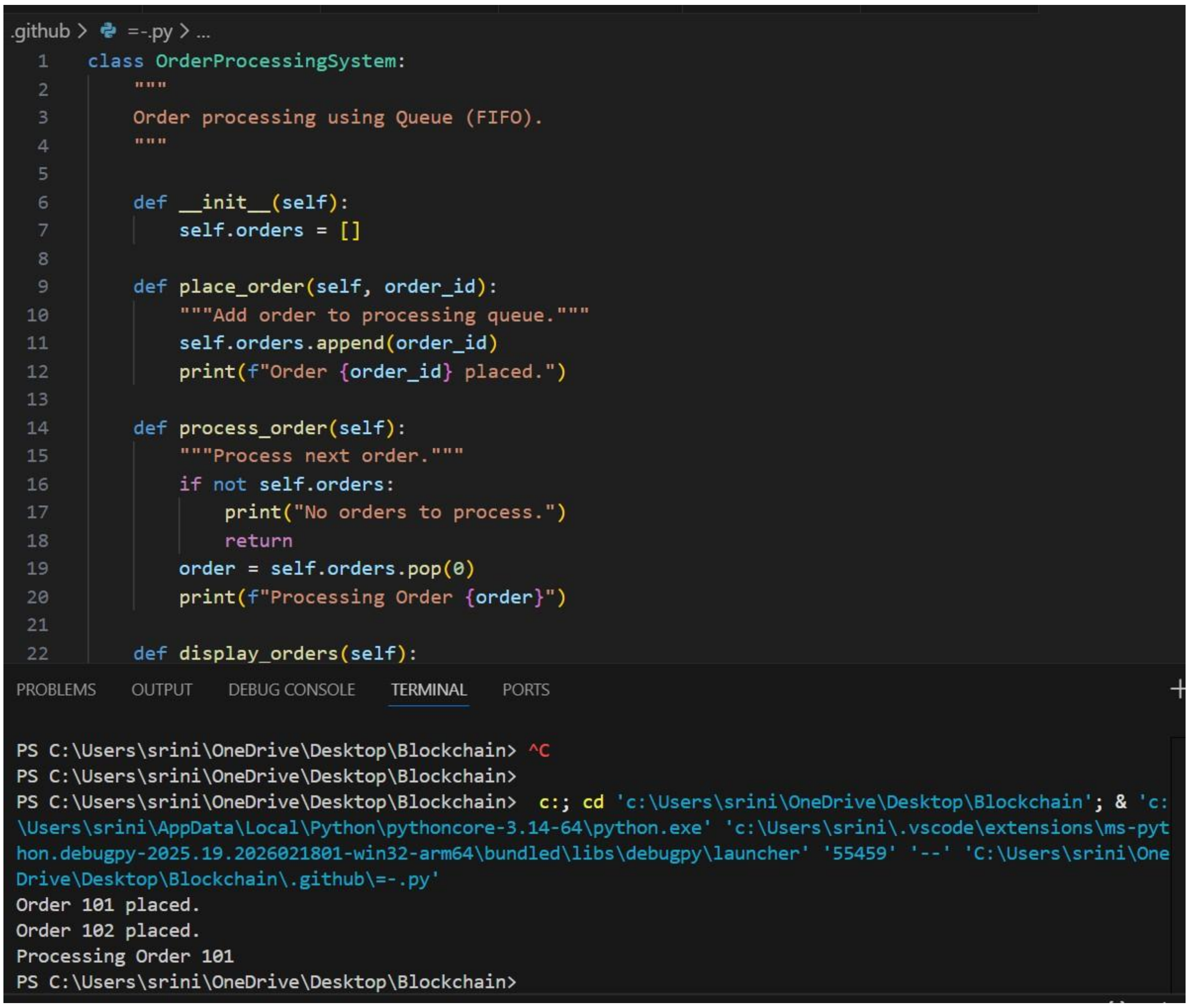
• Justify your choice in 2–3 sentences per feature.

• 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.



```python
class OrderProcessingSystem:
    """
    Order processing using Queue (FIFO).
    """

    def __init__(self):
        self.orders = []

    def place_order(self, order_id):
        """Add order to processing queue."""
        self.orders.append(order_id)
        print(f"Order {order_id} placed.")

    def process_order(self):
        """Process next order."""
        if not self.orders:
            print("No orders to process.")
            return
        order = self.orders.pop(0)
        print(f"Processing Order {order}")

    def display_orders(self):
```

```
PS C:\Users\srini\OneDrive\Desktop\Blockchain> ^C
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
PS C:\Users\srini\OneDrive\Desktop\Blockchain>  c:; cd 'c:\Users\srini\OneDrive\Desktop\Blockchain'; & 'c:
\Users\srini\AppData\Local\Python\pythoncore-3.14-64\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.19.2026021801-win32-arm64\bundled\libs\debugpy\launcher' '55459' '--' 'C:\Users\srini\One
Drive\Desktop\Blockchain\.github\=-.py'
Order 101 placed.
Order 102 placed.
Processing Order 101
PS C:\Users\srini\OneDrive\Desktop\Blockchain>
```