AI ASSISTED CODING

LAB-11.1

CH.Vishnu vardhan

2303A51718

Batch-11

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:**

#Write a code to generate a stack class with push,pop,peek and isEmpty methods **CODE:**

lab6.py > ...

```python
#Write a code to generate a stack class with push,pop,peek and isEmpty methods
class Stack:
    def __init__(self):
        self.stack = []
    def push(self, item):
        self.stack.append(item)
    def pop(self):
        if not self.isEmpty():
            return self.stack.pop()
        else:
            raise IndexError("Stack is empty")
    def peek(self):
        if not self.isEmpty():
            return self.stack[-1]
        else:
            raise IndexError("Stack is empty")
    def isEmpty(self):
        return len(self.stack) == 0
# Example usage
if __name__ == "__main__":
    stack = Stack()
    stack.push(1)
    stack.push(2)
    stack.push(3)
    print(stack.peek())  # Output: 3
    print(stack.pop())   # Output: 3
    print(stack.isEmpty())  # Output: False
    print(stack.pop())   # Output: 2
    print(stack.pop())   # Output: 1
    print(stack.isEmpty())  # Output: True

```

OUTPUT:

```
False …
PS C:\Users\thota\OneDrive\Desktop\AIAC> & C:/Users/thota/AppData/Local/Programs/Python/Pytho
/AIAC/lab6.py
3
3
False
2
1
True
PS C:\Users\thota\OneDrive\Desktop\AIAC>
```

Task Description #2 – Queue Implementation

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

Sample Input Code: class

Queue:

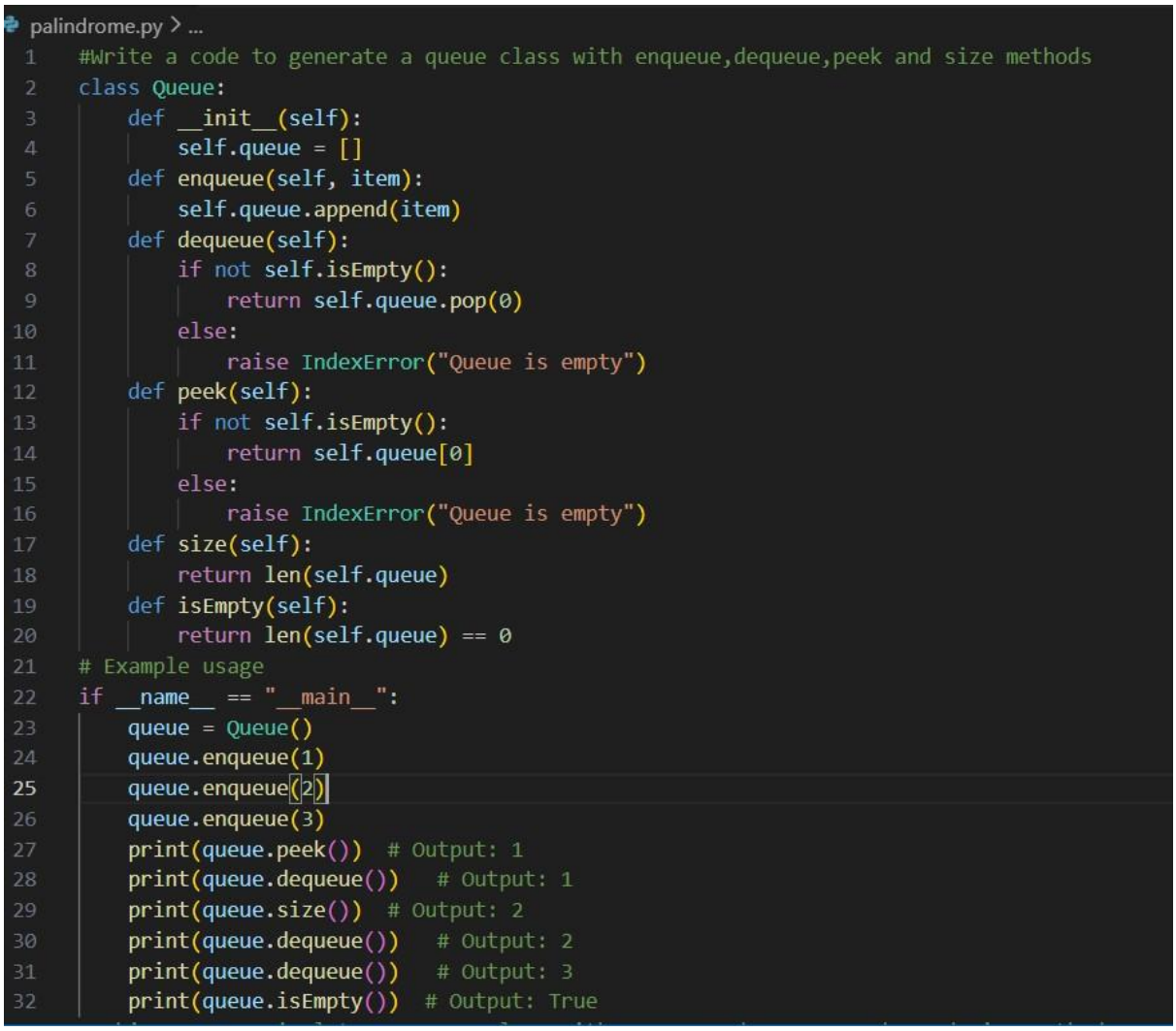
pass

Expected Output:

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

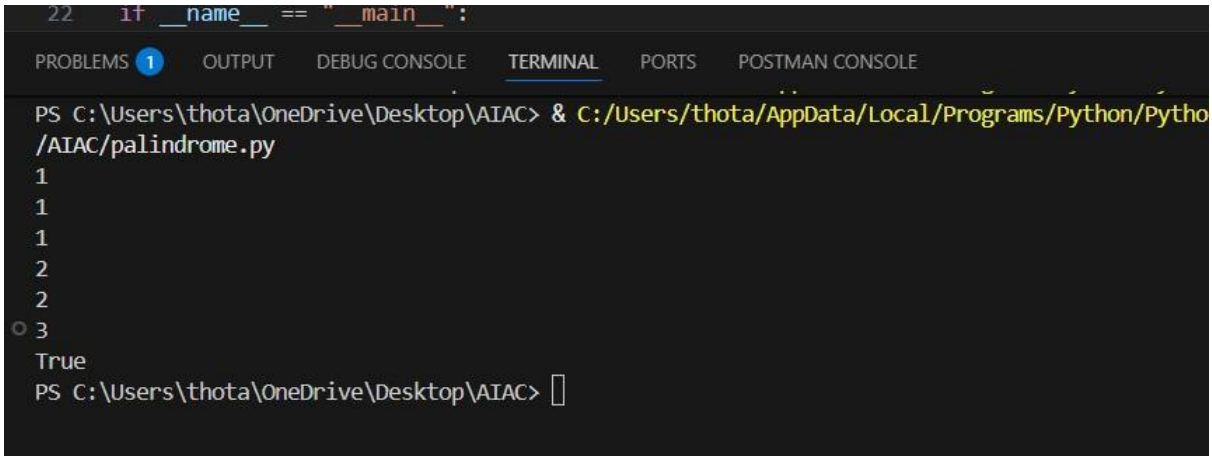
PROMPT:

#Write a code to generate a queue class with enqueue,dequeue,peek and size methods **CODE:**



```python
#Write a code to generate a queue class with enqueue,dequeue,peek and size methods
class Queue:
    def __init__(self):
        self.queue = []
    def enqueue(self, item):
        self.queue.append(item)
    def dequeue(self):
        if not self.isEmpty():
            return self.queue.pop(0)
        else:
            raise IndexError("Queue is empty")
    def peek(self):
        if not self.isEmpty():
            return self.queue[0]
        else:
            raise IndexError("Queue is empty")
    def size(self):
        return len(self.queue)
    def isEmpty(self):
        return len(self.queue) == 0
# Example usage
if __name__ == "__main__":
    queue = Queue()
    queue.enqueue(1)
    queue.enqueue(2)
    queue.enqueue(3)
    print(queue.peek())  # Output: 1
    print(queue.dequeue())   # Output: 1
    print(queue.size())  # Output: 2
    print(queue.dequeue())   # Output: 2
    print(queue.dequeue())   # Output: 3
    print(queue.isEmpty())  # Output: True
```

OUTPUT:



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:**

#Write a code to generate a singly linkedlist with insert and display methods

CODE AND OUTPUT:

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.

PROMPT:

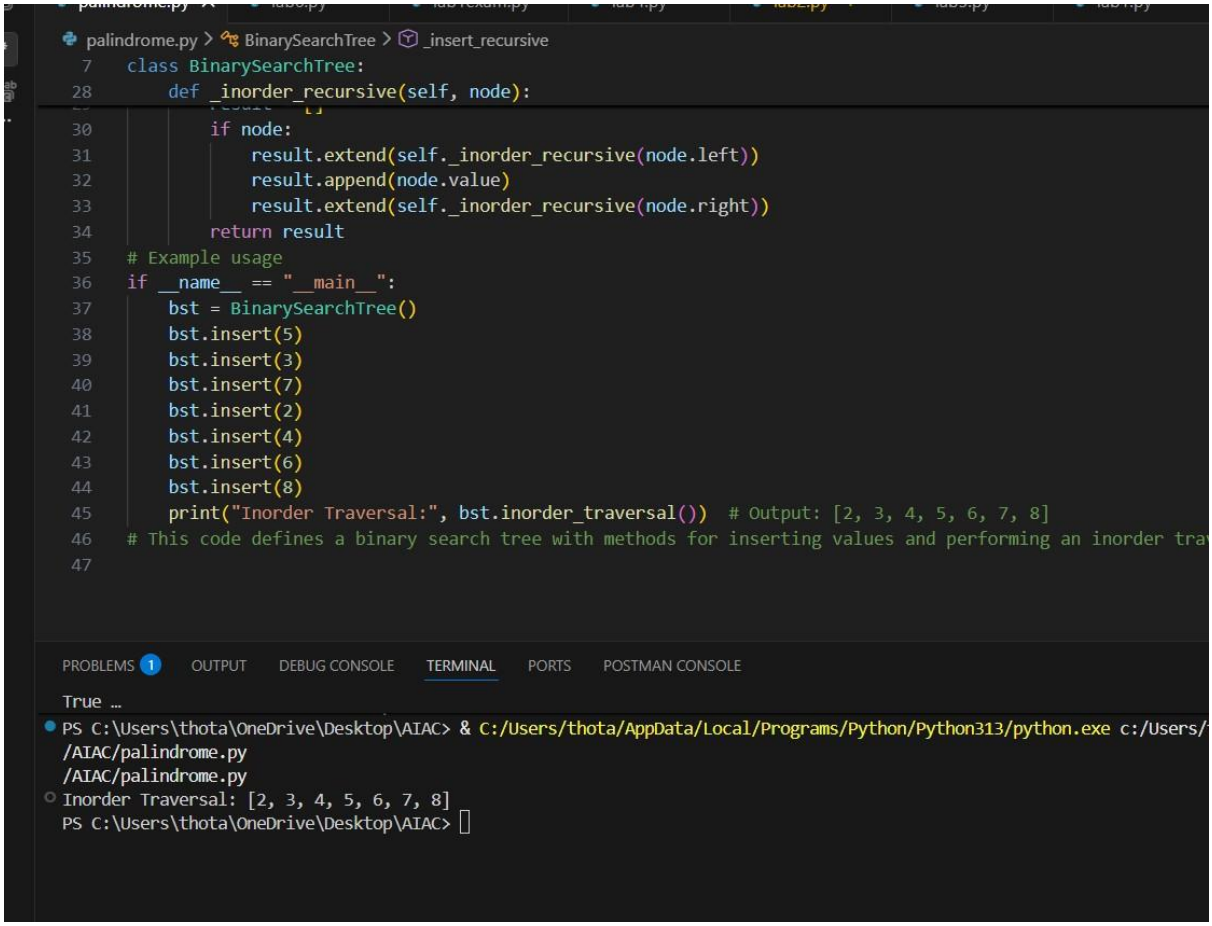
#Write a code to create a binary search tree and inorder traversal methods using recursive insert and travesal methods

CODE AND OUTPUT:

palindrome.py | lab6.py | lab1exam.py | lab4.py | lab2.py 1 | lab5.py

palindrome.py > BinarySearchTree > _insert_recursive

```python
#Write a code to create a binary search tree and inorder traversal methods using recursive
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None
class BinarySearchTree:
    def __init__(self):
        self.root = None
    def insert(self, value):
        if self.root is None:
            self.root = TreeNode(value)
        else:
            self._insert_recursive(self.root, value)
    def _insert_recursive(self, node, value):
        if value < node.value:
            if node.left is None:
                node.left = TreeNode(value)
            else:
                self._insert_recursive(node.left, value)
        else:
            if node.right is None:
                node.right = TreeNode(value)
            else:
                self._insert_recursive(node.right, value)
    def inorder_traversal(self):
        return self._inorder_recursive(self.root)
    def _inorder_recursive(self, node):
        result = []
        if node:
            result.extend(self._inorder_recursive(node.left))
            result.append(node.value)
            result.extend(self._inorder_recursive(node.right))
        return result
# Example usage
if __name__ == "__main__":
    bst = BinarySearchTree()
```



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

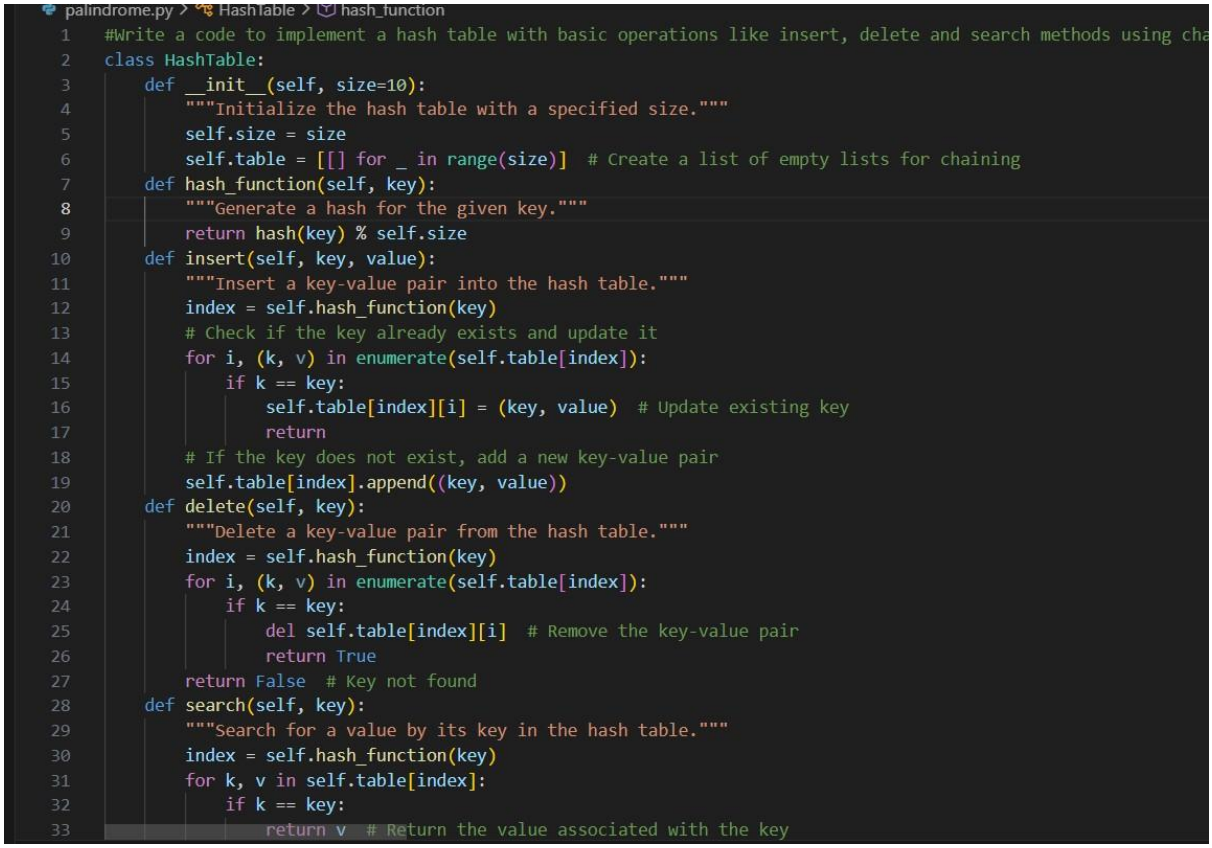
Expected Output:

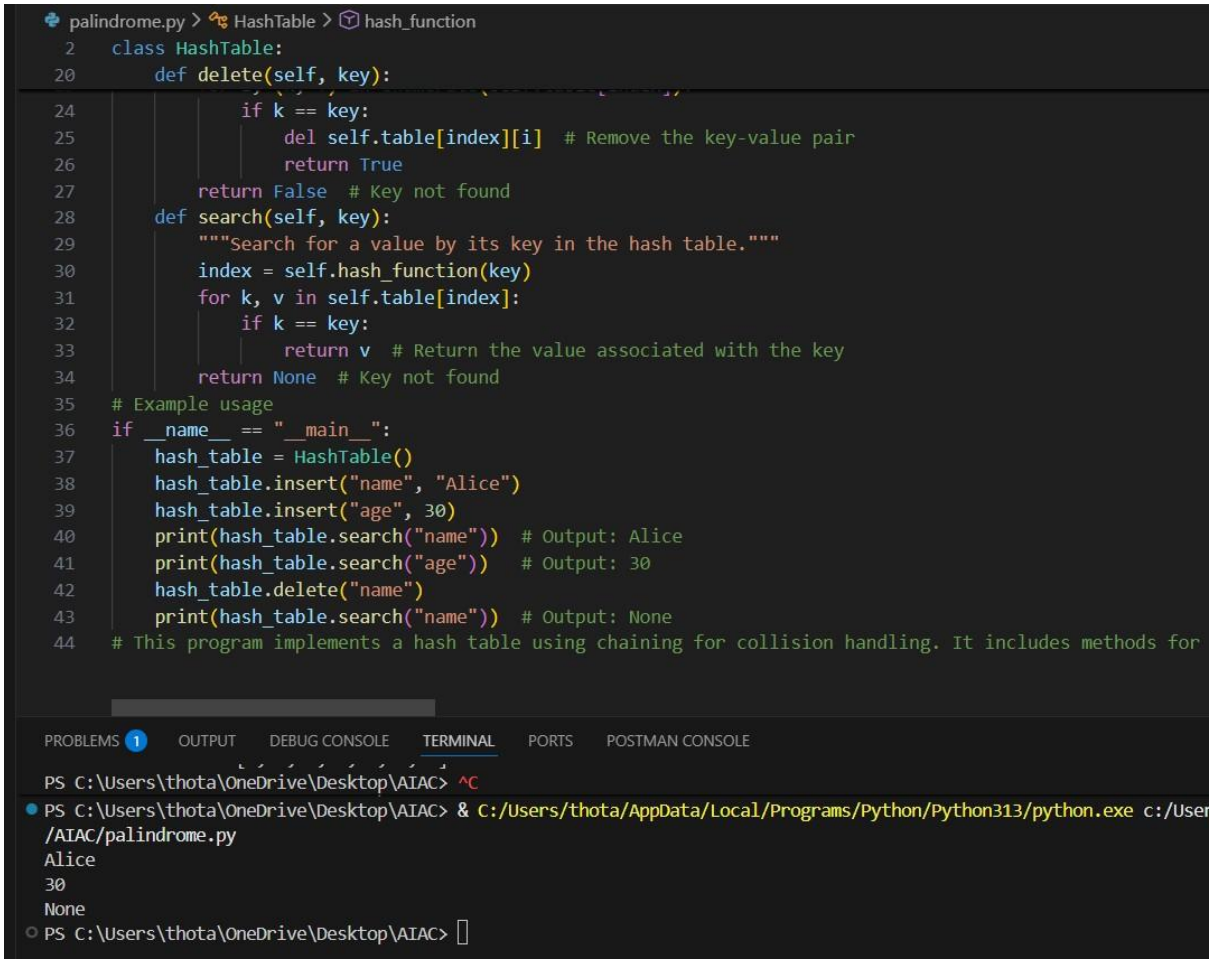
• Collision handling using chaining, with wellcommented methods.

PROMPT:

#Write a code to implement a hash table with basic operations like insert, delete and search methods using chaining for collision handling with well commented methods

CODE AND OUTPUT:





Task Description #6 – Graph Representation

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

Sample Input Code: class

Graph:

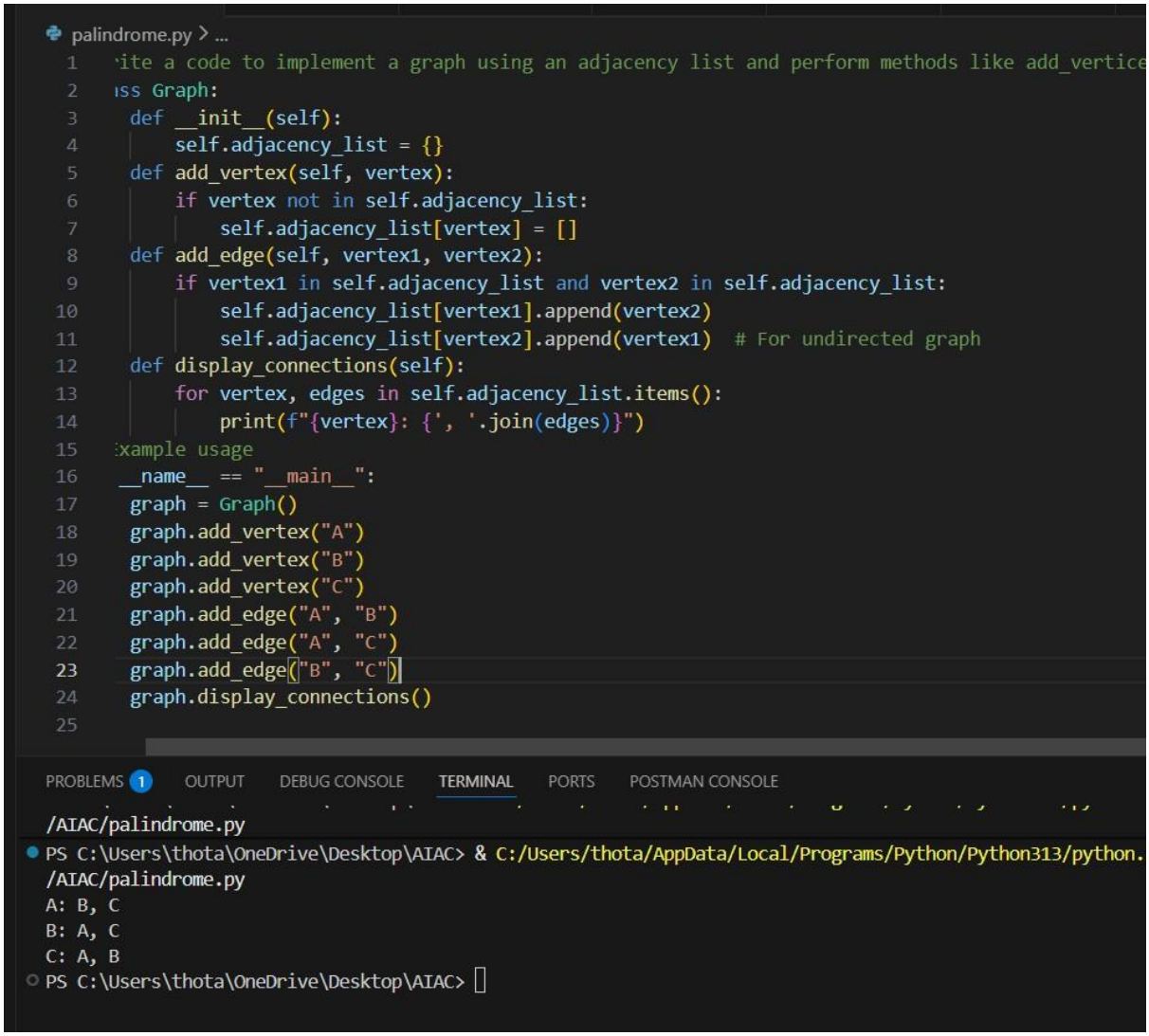
pass

Expected Output:

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

PROMPT:

#Write a code to implement a graph using an adjacency list and perform methods like add_vertices,add_edges and display connections **CODE AND OUTPUT:**



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.

PROMPT:

#Write a code to implement a priority queue using python's heapq module and implement with methods for enqueue,dequeue and display methods

CODE AND OUTPUT:

palindrome.py | lab6.py | lab1exam.py | lab4.py | lab2.py 1 | lab5.py

palindrome.py > PriorityQueue > is_empty

```python
#Write a code to implement a priority queue using python's heapq module and implemen
import heapq
class PriorityQueue:
    def __init__(self):
        self.elements = []
    def enqueue(self, item, priority):
        heapq.heappush(self.elements, (priority, item))
    def dequeue(self):
        if not self.is_empty():
            return heapq.heappop(self.elements)[1]
        else:
            raise IndexError("Priority Queue is empty")
    def display(self):
        print("Priority Queue:")
        for priority, item in sorted(self.elements):
            print(f"Item: {item}, Priority: {priority}")
    def is_empty(self):
        return len(self.elements) == 0
# Example usage
if __name__ == "__main__":
    pq = PriorityQueue()
    pq.enqueue("Task 1", priority=3)
    pq.enqueue("Task 2", priority=1)
    pq.enqueue("Task 3", priority=2)
    pq.display()
```

palindrome.py | lab6.py | lab1exam.py | lab4.py | lab2.py 1 | lab5.py

palindrome.py > PriorityQueue > is_empty

```python
#Write a code to implement a priority queue using python's heapq module and implemen
import heapq
class PriorityQueue:
    def __init__(self):
        self.elements = []
    def enqueue(self, item, priority):
        heapq.heappush(self.elements, (priority, item))
    def dequeue(self):
        if not self.is_empty():
            return heapq.heappop(self.elements)[1]
        else:
            raise IndexError("Priority Queue is empty")
    def display(self):
        print("Priority Queue:")
        for priority, item in sorted(self.elements):
            print(f"Item: {item}, Priority: {priority}")
    def is_empty(self):
        return len(self.elements) == 0
# Example usage
if __name__ == "__main__":
    pq = PriorityQueue()
    pq.enqueue("Task 1", priority=3)
    pq.enqueue("Task 2", priority=1)
    pq.enqueue("Task 3", priority=2)
    pq.display()
```

Task Description #8 – Deque

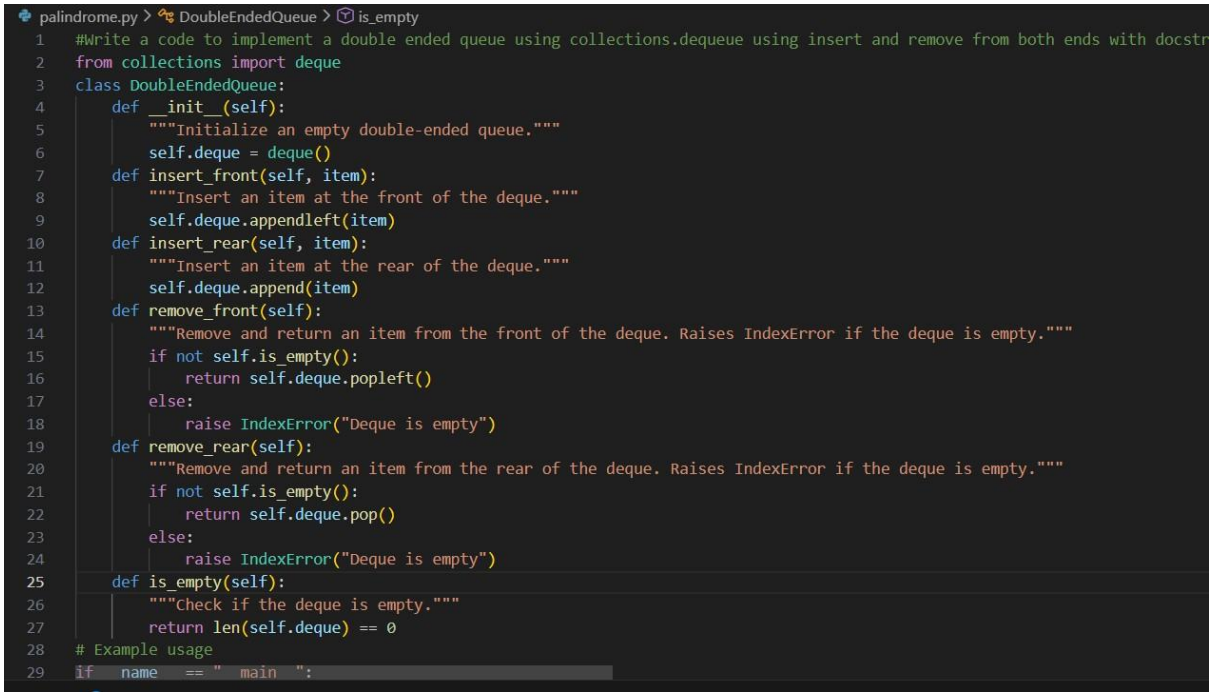
Task: Use AI to implement a double-ended queue using collections.deque. Sample Input Code: class DequeDS: pass

Expected Output:

- Insert and remove from both ends with docstrings.

PROMPT:

#Write a code to implement a double ended queue using collections.dequeue using insert and remove from both ends with docstring **CODE AND OUTPUT:**



```
 3  class DoubleEndedQueue:
19      def remove_rear(self):
22              return self.deque.pop()
23          else:
24              raise IndexError("Deque is empty")
25      def is_empty(self):
26          """Check if the deque is empty."""
27          return len(self.deque) == 0
28  # Example usage
29  if __name__ == "__main__":
30      deq = DoubleEndedQueue()
31      deq.insert_rear(1)
32      deq.insert_rear(2)
33      deq.insert_front(0)
34      print(deq.deque)  # Output: deque([0, 1, 2])
35      print(deq.remove_front())  # Output: 0
36      print(deq.remove_rear())   # Output: 2
37      print(deq.is_empty())      # Output: False
38      print(deq.remove_front())  # Output: 1
39      print(deq.is_empty())      # Output: True
40  # This code implements a double-ended queue (deque) using the collections.d
41
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

```
/AIAC/palindrome.py
1
True
PS C:\Users\thota\OneDrive\Desktop\AIAC> ^C
PS C:\Users\thota\OneDrive\Desktop\AIAC> & C:/Users/thota/AppData/Local/Programs/Python
/AIAC/palindrome.py
deque([0, 1, 2])
0
2
False
1
True
PS C:\Users\thota\OneDrive\Desktop\AIAC>
```