

NAME : B.SRISHANTH

ROLL NO :2403A510G3

SUBJECT: AI ASSISTED CODING

ASSIGNMENT : 11.1

Task Description #1 – Stack Implementation

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

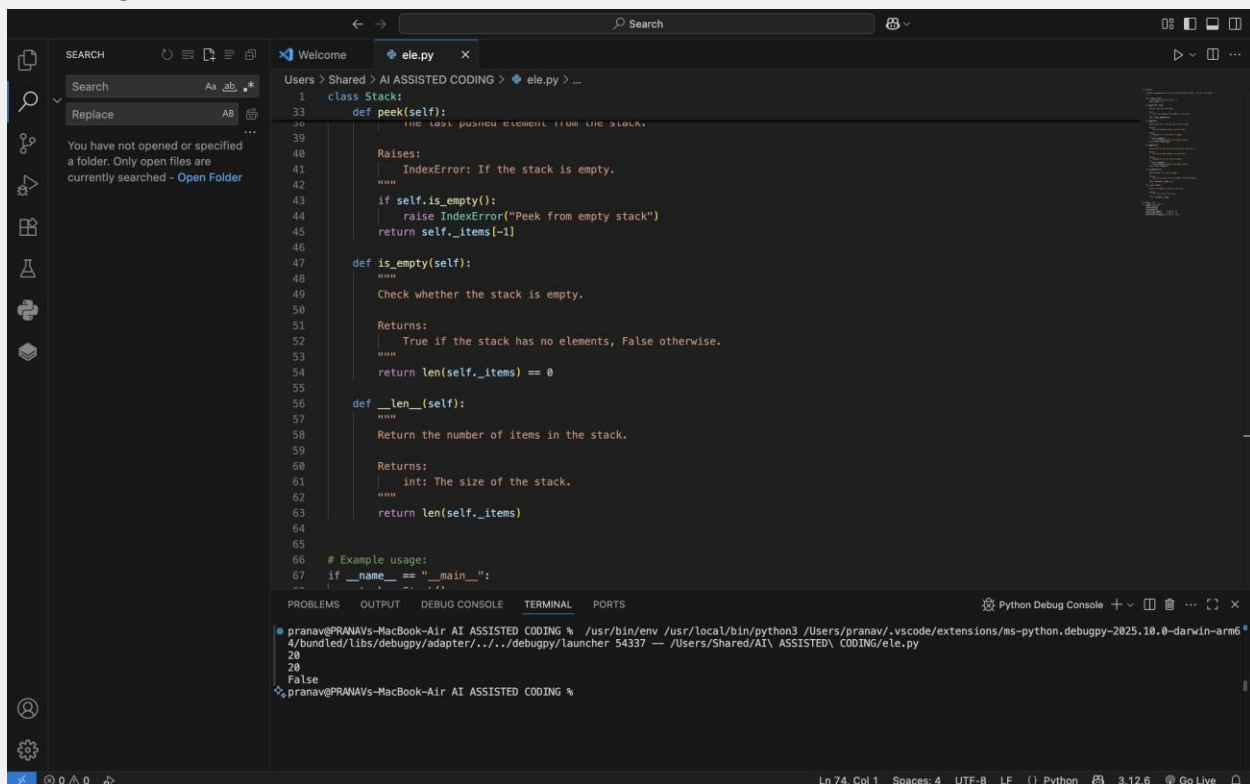
Sample Input Code:

class Stack:

pass

Expected Output:

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



```
1 class Stack:
33     def peek(self):
34         """
35         Return the last pushed element from the stack.
36
37         Raises:
38             IndexError: If the stack is empty.
39         """
40         if self.is_empty():
41             raise IndexError("Peek from empty stack")
42         return self._items[-1]
43
44     def is_empty(self):
45         """
46         Check whether the stack is empty.
47
48         Returns:
49             True if the stack has no elements, False otherwise.
50         """
51         return len(self._items) == 0
52
53     def __len__(self):
54         """
55         Return the number of items in the stack.
56
57         Returns:
58             int: The size of the stack.
59         """
60         return len(self._items)
61
62 # Example usage:
63 if __name__ == "__main__":
64     s = Stack()
65     s.push(10)
66     s.push(20)
67     print(s.is_empty())
68     print(s.peek())
69     print(len(s))
```

pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/.../debugpy/launcher 54337 -- /Users/Shared/AI/ ASSISTED CODING/ele.py

20
20
False
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %

OUTPUT :

```
56     def __len__(self):
57         """
58         Return the number of items in the stack.
59
60         Returns:
61             int: The size of the stack.
62         """
63         return len(self._items)
64
65
66 # Example usage:
67 if __name__ == "__main__":
    
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python Debug Console + - [] ... [] X

```
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54337 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
20
False
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %
```

Ln 74, Col 1 Spaces: 4 UTF-8 LF () Python 3.12.6 Go Live

PROMPT :

To generate a Stack class with push, pop, peek, and is_empty methods.

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

The screenshot shows a VS Code editor with a file named `ele.py` open. The code defines a `Queue` class with methods `size`, `is_empty`, `enqueue`, and `dequeue`. It also includes an example usage block. The terminal at the bottom shows the output of running the code, which is:

```
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54337 -- /Users/Shared/AI ASSISTED CODING/ele.py
20
20
False
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI ASSISTED CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54362 -- /Users/Shared/AI ASSISTED CODING/ele.py
10
10
2
False
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %
```

OUTPUT :

The screenshot shows a VS Code editor with a file named `ele.py` open. The code defines a `Stack` class with methods `size`, `is_empty`, `push`, and `pop`. It also includes an example usage block. The terminal at the bottom shows the output of running the code, which is:

```
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54337 -- /Users/Shared/AI ASSISTED CODING/ele.py
20
20
False
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI ASSISTED CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54362 -- /Users/Shared/AI ASSISTED CODING/ele.py
10
10
2
False
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %
```

PROMPT :

To implement a Queue using Python lists in a simple and basic way

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.

```
class Node:
    """
    A Node in a singly linked list.
    """
    Attributes:
        data: The value stored in the node.
        next: A reference to the next node in the list (default is None).
    """
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:
    """
    A simple implementation of a Singly Linked List.
    """
    def __init__(self):
        """Initialize an empty linked list."""
        self.head = None

    def insert(self, data):
        """
        Insert a new node with the given data at the end of the list.
        """
        Args:
            data: The value to be inserted into the list.
        """
        new_node = Node(data)

        # If the list is empty, new node becomes the head
```

```
/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54362 -- /Users/Shared/AI/ ASSISTED\ CODING/ele.py
10
10
2
False
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI/ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions
/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54398 -- /Users/Shared/AI/ ASSISTED\ CODING/ele.py
10 -> 20 => 30
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI/ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions
/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54410 -- /Users/Shared/AI/ ASSISTED\ CODING/ele.py
10 -> 20 => 30
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %
```

```
class LinkedList:
    def __init__(self):
        """Initialize an empty linked list."""
        self.head = None

    def insert(self, data):
        """
        Insert a new node with the given data at the end of the list.

        Args:
            data: The value to be inserted into the list.
        """
        new_node = Node(data)

        # If the list is empty, new node becomes the head
        if self.head is None:
            self.head = new_node
        else:
            # Traverse to the last node
            current = self.head
            while current.next:
                current = current.next
            current.next = new_node

    def display(self):
        """
        Display the elements of the linked list in order.
        """
        elements = []
        current = self.head
```

```
/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54362 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
10
10
2
False
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54398 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
10 -> 20 -> 30
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54410 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
10 -> 20 -> 30
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %
```

OUTPUT :

```
empty_list = LinkedList()

/.../debugpy/launcher 54362 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
10
10
2
False
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54398 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
10 -> 20 -> 30
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54410 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
10 -> 20 -> 30
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %
```

PROMPT :

To generate a Singly Linked List with insert and display methods using nodes

Task Description #4 – Binary Search Tree (BST)

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

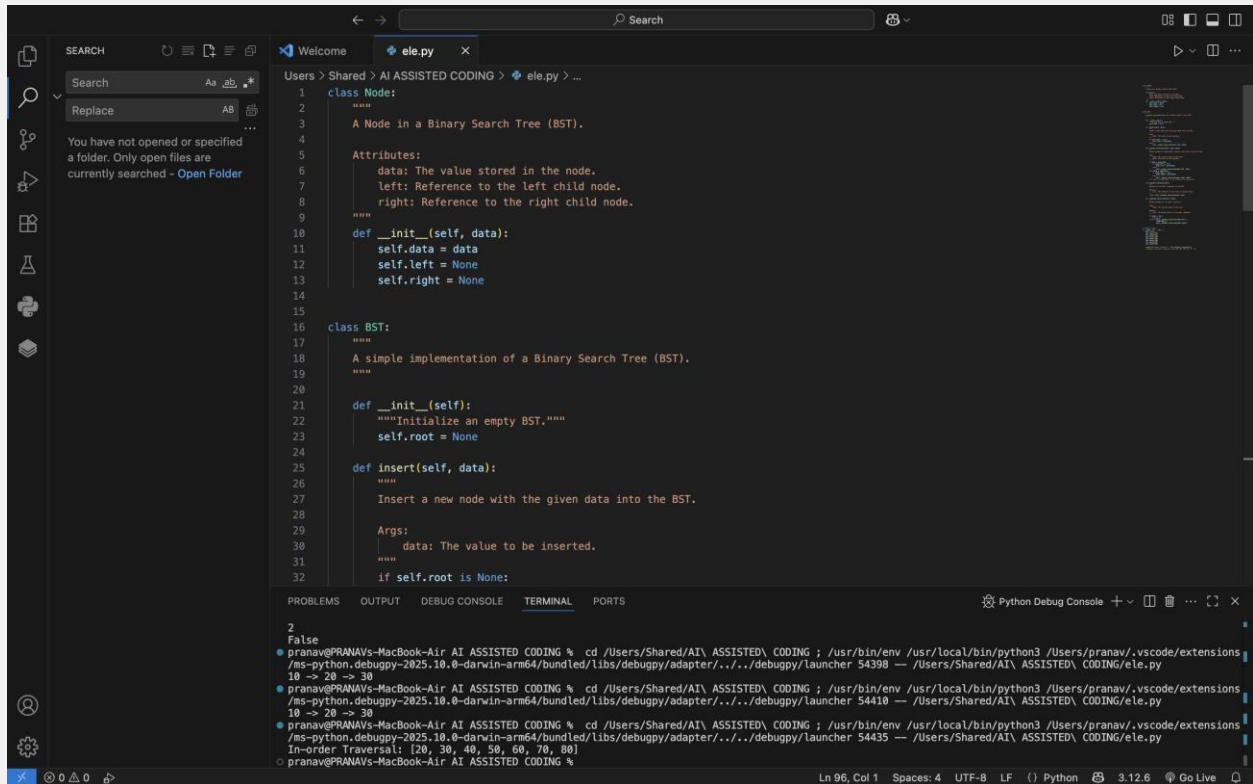
Sample Input Code:

```
class BST:
```

```
pass
```

Expected Output:

- BST implementation with recursive insert and traversal method



```
1 class Node:
2     """
3     A Node in a Binary Search Tree (BST).
4
5     Attributes:
6         data: The value stored in the node.
7         left: Reference to the left child node.
8         right: Reference to the right child node.
9     """
10    def __init__(self, data):
11        self.data = data
12        self.left = None
13        self.right = None
14
15    class BST:
16        """
17        A simple implementation of a Binary Search Tree (BST).
18        """
19
20        def __init__(self):
21            """Initialize an empty BST."""
22            self.root = None
23
24        def insert(self, data):
25            """
26            Insert a new node with the given data into the BST.
27
28            Args:
29                data: The value to be inserted.
30            """
31            if self.root is None:
```

```
2
False
pranav@PRANAVS-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions
/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/Libs/debugpy/adapter/../../debugpy/launcher 54398 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
10 -> 20 -> 30
pranav@PRANAVS-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions
/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/Libs/debugpy/adapter/../../debugpy/launcher 54410 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
10 -> 20 -> 30
pranav@PRANAVS-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions
/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/Libs/debugpy/adapter/../../debugpy/launcher 54435 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
In-order Traversal: [20, 30, 40, 50, 60, 70, 80]
pranav@PRANAVS-MacBook-Air AI ASSISTED CODING %
```

```
class BST:
    """
    A simple implementation of a Binary Search Tree (BST).
    """
    def __init__(self):
        """Initialize an empty BST."""
        self.root = None

    def insert(self, data):
        """
        Insert a new node with the given data into the BST.

        Args:
            data: The value to be inserted.
        """
        if self.root is None:
            self.root = Node(data)
        else:
            self._insert_recursive(self.root, data)

    def _insert_recursive(self, node, data):
        """
        Helper method to recursively insert a new value into the tree.

        Args:
            node: The current node in the tree.
            data: The value to be inserted.
        """
        if data < node.data:
            if node.left is None:
                node.left = Node(data)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python Debug Console

```
2
False
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.ms-python.debugpy-2025.10.0-darwin-arm64/bundled/Libs/debugpy/adapter/../../debugpy/launcher 54398 -- /Users/Shared/AI\ ASSISTED\ CODING
10 -> 20 -> 30
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.ms-python.debugpy-2025.10.0-darwin-arm64/bundled/Libs/debugpy/adapter/../../debugpy/launcher 54418 -- /Users/Shared/AI\ ASSISTED\ CODING
10 -> 20 -> 30
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/Libs/debugpy/adapter/../../debugpy/launcher 54435 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
In-order Traversal: [20, 30, 40, 50, 60, 70, 80]
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %
```

```
def _inorder_recursive(self, node):
    return (self._inorder_recursive(node.left) +
            (node.data) +
            self._inorder_recursive(node.right))

# Example usage:
if __name__ == "__main__":
    bst = BST()
    bst.insert(50)
    bst.insert(30)
    bst.insert(70)
    bst.insert(20)
    bst.insert(40)
    bst.insert(60)
    bst.insert(80)

    print("In-order Traversal:", bst.inorder_traversal())
    # Output: In-order Traversal: [20, 30, 40, 50, 60, 70, 80]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python Debug Console

```
2
False
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.ms-python.debugpy-2025.10.0-darwin-arm64/bundled/Libs/debugpy/adapter/../../debugpy/launcher 54398 -- /Users/Shared/AI\ ASSISTED\ CODING
10 -> 20 -> 30
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.ms-python.debugpy-2025.10.0-darwin-arm64/bundled/Libs/debugpy/adapter/../../debugpy/launcher 54418 -- /Users/Shared/AI\ ASSISTED\ CODING
10 -> 20 -> 30
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/Libs/debugpy/adapter/../../debugpy/launcher 54435 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
In-order Traversal: [20, 30, 40, 50, 60, 70, 80]
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %
```

OUTPUT :


```
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54435 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
In-order Traversal: [20, 30, 40, 50, 60, 70, 80]
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %
```

PROMPT :

o create a BST with insert and in-order traversal methods.

Task Description #5 – Hash Table

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

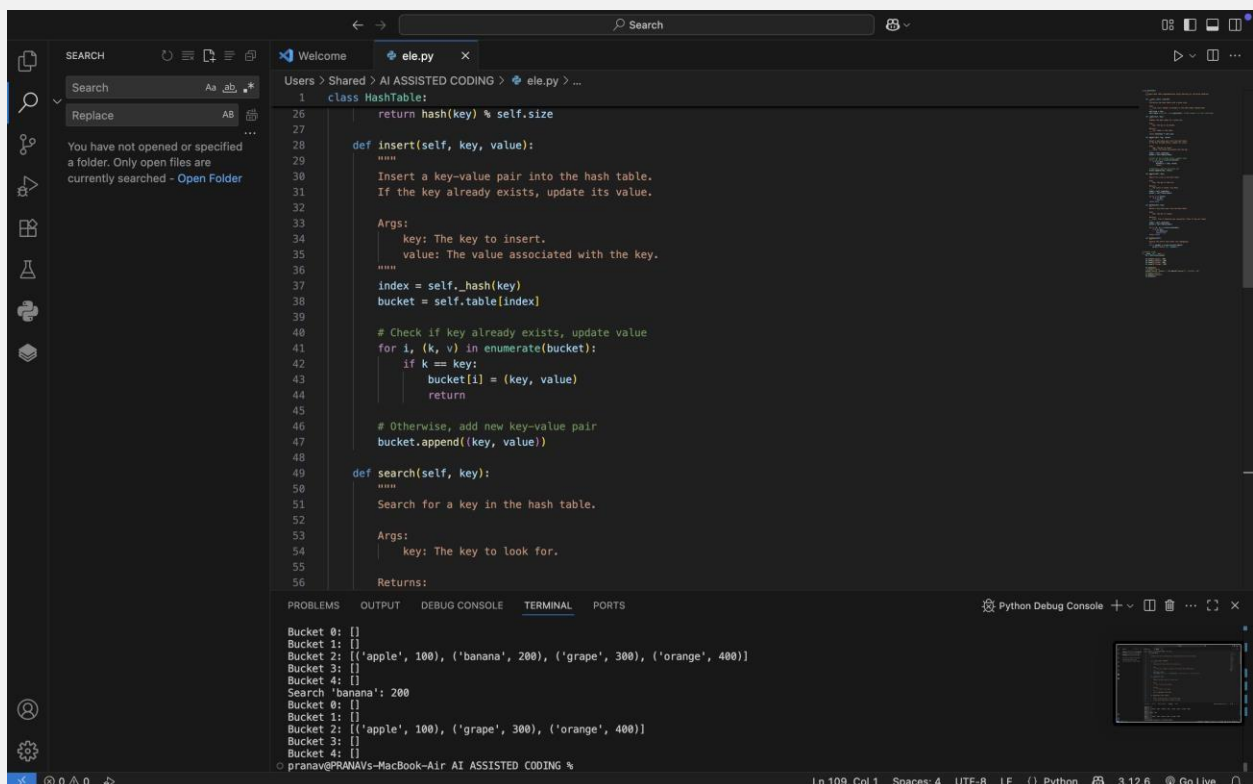
Sample Input Code:

class HashTable:

pass

Expected Output:

- Collision handling using chaining, with well-commented methods



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

    def insert(self, key, value):
        """
        Insert a key-value pair into the hash table.
        If the key already exists, update its value.

        Args:
            key: The key to insert.
            value: The value associated with the key.
        """
        index = self._hash(key)
        bucket = self.table[index]

        # Check if key already exists, update value
        for i, (k, v) in enumerate(bucket):
            if k == key:
                bucket[i] = (key, value)
                return

        # Otherwise, add new key-value pair
        bucket.append((key, value))

    def search(self, key):
        """
        Search for a key in the hash table.

        Args:
            key: The key to look for.

        Returns:
            The value associated with the key, or None if not found.
        """
        index = self._hash(key)
        bucket = self.table[index]

        for i, (k, v) in enumerate(bucket):
            if k == key:
                return v
        return None

    def delete(self, key):
        """
        Delete a key-value pair from the hash table.

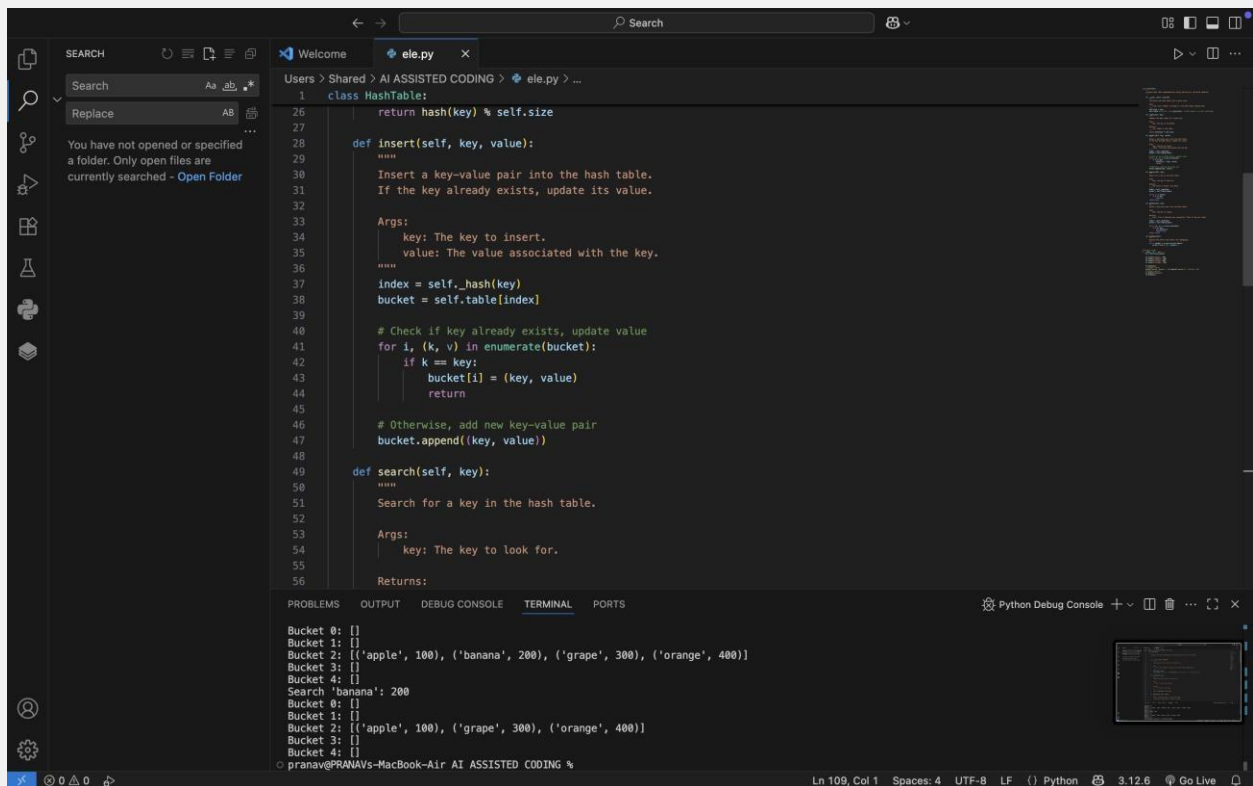
        Args:
            key: The key to delete.
        """
        index = self._hash(key)
        bucket = self.table[index]

        for i, (k, v) in enumerate(bucket):
            if k == key:
                del bucket[i]
                return

    def _hash(self, key):
        return hash(key) % self.size
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Python Debug Console
Bucket 0: []
Bucket 1: []
Bucket 2: [('apple', 100), ('banana', 200), ('grape', 300), ('orange', 400)]
Bucket 3: []
Bucket 4: []
Search 'banana': 200
Bucket 0: []
Bucket 1: []
Bucket 2: [('apple', 100), ('grape', 300), ('orange', 400)]
Bucket 3: []
Bucket 4: []
```

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

    def insert(self, key, value):
        """
        Insert a key-value pair into the hash table.
        If the key already exists, update its value.

        Args:
            key: The key to insert.
            value: The value associated with the key.
        """
        index = self._hash(key)
        bucket = self.table[index]

        # Check if key already exists, update value
        for i, (k, v) in enumerate(bucket):
            if k == key:
                bucket[i] = (key, value)
                return

        # Otherwise, add new key-value pair
        bucket.append((key, value))

    def search(self, key):
        """
        Search for a key in the hash table.

        Args:
            key: The key to look for.

        Returns:
            The value associated with the key, or None if not found.
        """
        index = self._hash(key)
        bucket = self.table[index]

        for i, (k, v) in enumerate(bucket):
            if k == key:
                return v
        return None

    def delete(self, key):
        """
        Delete a key-value pair from the hash table.

        Args:
            key: The key to delete.

        Returns:
            bool: True if deletion was successful, False if key not found.
        """
        index = self._hash(key)
        bucket = self.table[index]

        for i, (k, v) in enumerate(bucket):
            if k == key:
                del bucket[i]
                return True
        return False

    def display(self):
        """
        Display the entire hash table (for debugging).
        """
        for i, bucket in enumerate(self.table):
            print(f"Bucket {i}: {bucket}")

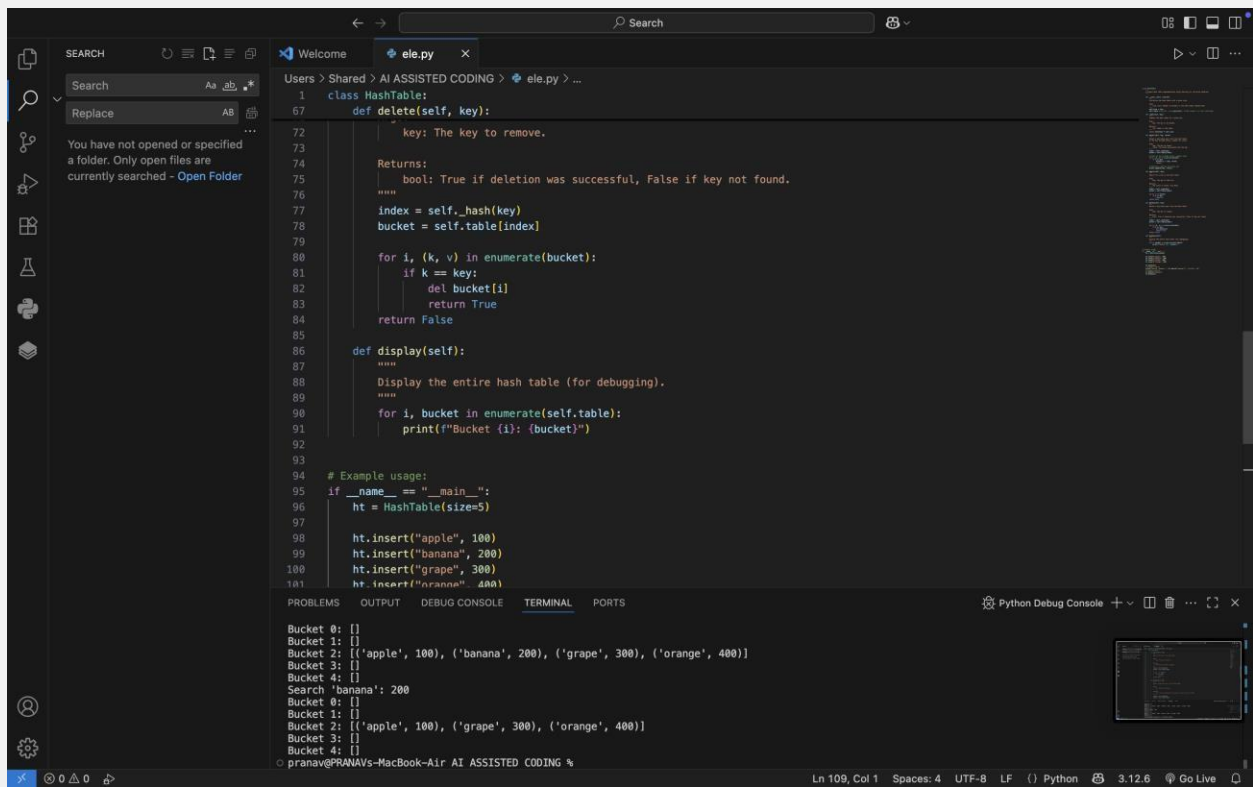
    def _hash(self, key):
        return hash(key) % self.size

# Example usage:
if __name__ == "__main__":
    ht = HashTable(size=5)

    ht.insert("apple", 100)
    ht.insert("banana", 200)
    ht.insert("grape", 300)
    ht.insert("orange", 400)

    print("Search 'banana':", ht.search("banana"))
    print("Search 'apple':", ht.search("apple"))
    print("Search 'grape':", ht.search("grape"))
    print("Search 'orange':", ht.search("orange"))
    print("Search 'banana' (deleted):", ht.search("banana"))
    ht.delete("banana")
    ht.display()
```

Bucket 0: []
Bucket 1: []
Bucket 2: [('apple', 100), ('banana', 200), ('grape', 300), ('orange', 400)]
Bucket 3: []
Bucket 4: []
Search 'banana': 200
Bucket 0: []
Bucket 1: []
Bucket 2: [('apple', 100), ('grape', 300), ('orange', 400)]
Bucket 3: []
Bucket 4: []



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

    def insert(self, key, value):
        """
        Insert a key-value pair into the hash table.
        If the key already exists, update its value.

        Args:
            key: The key to insert.
            value: The value associated with the key.
        """
        index = self._hash(key)
        bucket = self.table[index]

        # Check if key already exists, update value
        for i, (k, v) in enumerate(bucket):
            if k == key:
                bucket[i] = (key, value)
                return

        # Otherwise, add new key-value pair
        bucket.append((key, value))

    def search(self, key):
        """
        Search for a key in the hash table.

        Args:
            key: The key to look for.

        Returns:
            The value associated with the key, or None if not found.
        """
        index = self._hash(key)
        bucket = self.table[index]

        for i, (k, v) in enumerate(bucket):
            if k == key:
                return v
        return None

    def delete(self, key):
        """
        Delete a key-value pair from the hash table.

        Args:
            key: The key to delete.

        Returns:
            bool: True if deletion was successful, False if key not found.
        """
        index = self._hash(key)
        bucket = self.table[index]

        for i, (k, v) in enumerate(bucket):
            if k == key:
                del bucket[i]
                return True
        return False

    def display(self):
        """
        Display the entire hash table (for debugging).
        """
        for i, bucket in enumerate(self.table):
            print(f"Bucket {i}: {bucket}")

    def _hash(self, key):
        return hash(key) % self.size

# Example usage:
if __name__ == "__main__":
    ht = HashTable(size=5)

    ht.insert("apple", 100)
    ht.insert("banana", 200)
    ht.insert("grape", 300)
    ht.insert("orange", 400)

    print("Search 'banana':", ht.search("banana"))
    print("Search 'apple':", ht.search("apple"))
    print("Search 'grape':", ht.search("grape"))
    print("Search 'orange':", ht.search("orange"))
    print("Search 'banana' (deleted):", ht.search("banana"))
    ht.delete("banana")
    ht.display()
```

Bucket 0: []
Bucket 1: []
Bucket 2: [('apple', 100), ('banana', 200), ('grape', 300), ('orange', 400)]
Bucket 3: []
Bucket 4: []
Search 'banana': 200
Bucket 0: []
Bucket 1: []
Bucket 2: [('apple', 100), ('grape', 300), ('orange', 400)]
Bucket 3: []
Bucket 4: []

OUTPUT :



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python Debug Console + - [ ] ... [ ] x

Bucket 0: []
Bucket 1: []
Bucket 2: [('apple', 100), ('banana', 200), ('grape', 300), ('orange', 400)]
Bucket 3: []
Bucket 4: []
Search 'banana': 200
Bucket 0: []
Bucket 1: []
Bucket 2: [('apple', 100), ('grape', 300), ('orange', 400)]
Bucket 3: []
Bucket 4: []
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %
```

Ln 109, Col 1 Spaces: 4 UTF-8 LF () Python 3.12.6 Go Live

PROMPT :

To implement a hash table with basic insert, search, and delete

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.

```
class Graph:
    """
    A simple graph implementation using an adjacency list.
    """
    def __init__(self):
        """Initialize an empty graph."""
        self.graph = {}

    def add_vertex(self, vertex):
        """
        Add a vertex to the graph.
        """
        Args:
            vertex: The vertex to be added.
        """
        if vertex not in self.graph:
            self.graph[vertex] = []

    def add_edge(self, v1, v2):
        """
        Add an edge between two vertices (undirected graph).
        """
        Args:
            v1: The first vertex.
            v2: The second vertex.
        """
        if v1 not in self.graph:
            self.add_vertex(v1)
        if v2 not in self.graph:
            self.add_vertex(v2)
```

Search 'banana': 200
Bucket 0: []
Bucket 1: []
Bucket 2: [['apple', 100], ('grape', 300), ('orange', 400)]
Bucket 3: []
Bucket 4: []

pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54677 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
A -> B, C
B -> A, C
C -> A, B

```
def add_edge(self, v1, v2):
    """
    v2: the second vertex.
    """
    if v1 not in self.graph:
        self.add_vertex(v1)
    if v2 not in self.graph:
        self.add_vertex(v2)

    self.graph[v1].append(v2)
    self.graph[v2].append(v1) # For undirected graph

def display(self):
    """
    Display the adjacency list representation of the graph.
    """
    for vertex, neighbors in self.graph.items():
        print(f"{vertex} -> {', '.join(map(str, neighbors))}")

# Example usage:
if __name__ == "__main__":
    g = Graph()
    g.add_vertex("A")
    g.add_vertex("B")
    g.add_vertex("C")
    g.add_edge("A", "B")
    g.add_edge("A", "C")
    g.add_edge("B", "C")

    g.display()
    # Example Output:
```

Search 'banana': 200
Bucket 0: []
Bucket 1: []
Bucket 2: [['apple', 100], ('grape', 300), ('orange', 400)]
Bucket 3: []
Bucket 4: []

pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54677 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
A -> B, C
B -> A, C
C -> A, B

OUTPUT:


```
class PriorityQueue:
    def dequeue(self):
        """
        Raises:
            IndexError: If the priority queue is empty.
        """
        if self.is_empty():
            raise IndexError("Dequeue from empty priority queue")
        return heapq.heappop(self._queue)[1]

    def display(self):
        """
        Display the contents of the priority queue (unsorted).
        """
        print("Priority Queue contents:", self._queue)

    def is_empty(self):
        """
        Check whether the priority queue is empty.
        Returns:
            True if empty, False otherwise.
        """
        return len(self._queue) == 0

# Example usage:
if __name__ == "__main__":
    pq = PriorityQueue()
    pq.enqueue("Task A", 2)
    pq.enqueue("Task B", 1)
    pq.enqueue("Task C", 3)
```

Bucket 4: []

pranav@PRANAVS-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/ms-python.debugpy-2825.10.0-darwin-arm64/bundled/Libs/debugpy/adapter/../../debugpy/launcher 54677 -- /Users/Shared/AI\ ASSISTED\ CODING

A -> B, C
B -> A, C
C -> A, B

pranav@PRANAVS-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/ms-python.debugpy-2825.10.0-darwin-arm64/bundled/Libs/debugpy/adapter/../../debugpy/launcher 54765 -- /Users/Shared/AI\ ASSISTED\ CODING

Priority Queue contents: [(1, 'Task B'), (2, 'Task A'), (3, 'Task C')]
Dequeued: Task B
Dequeued: Task A

pranav@PRANAVS-MacBook-Air AI ASSISTED CODING %

OUTPUT :

```
Priority Queue contents: [(1, 'Task B'), (2, 'Task A'), (3, 'Task C')]  
Dequeued: Task B  
Dequeued: Task A  
pranav@PRANAVS-MacBook-Air AI ASSISTED CODING %
```

PROMPT :

To implement a priority queue using Python's heapq module.

Task Description #8 – Deque

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

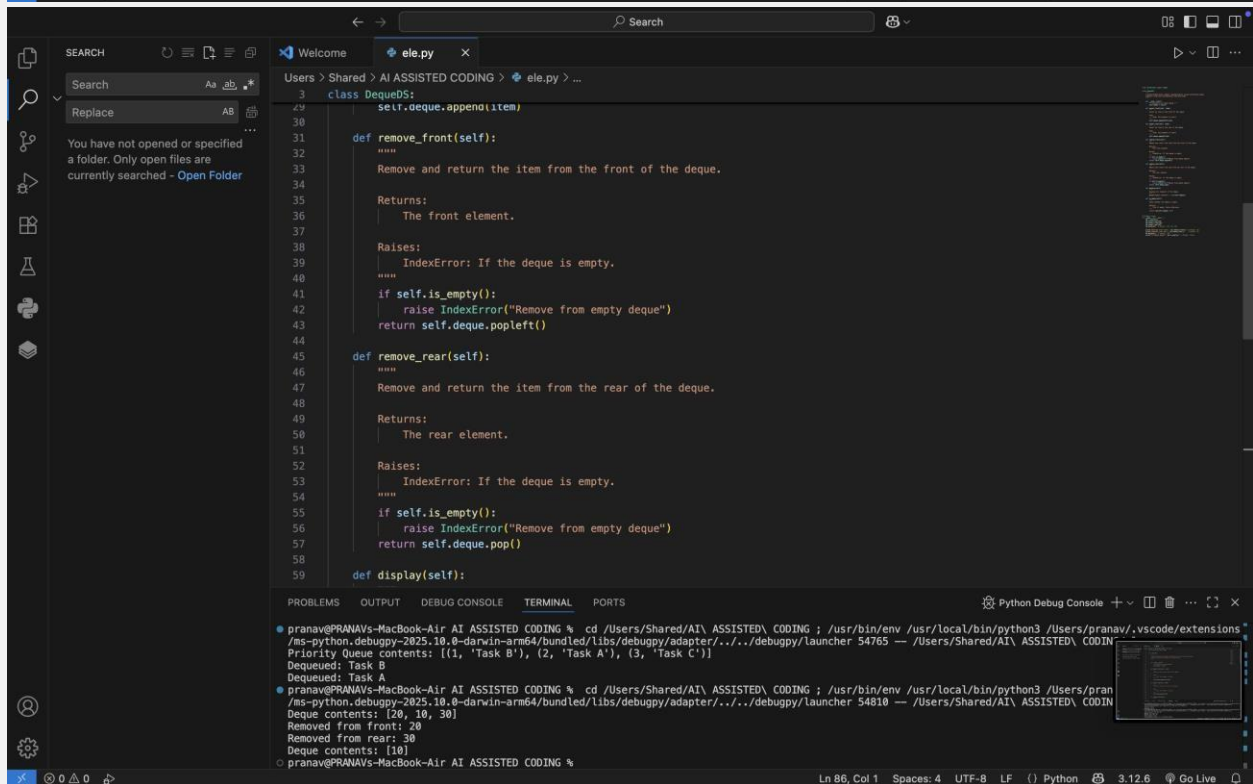
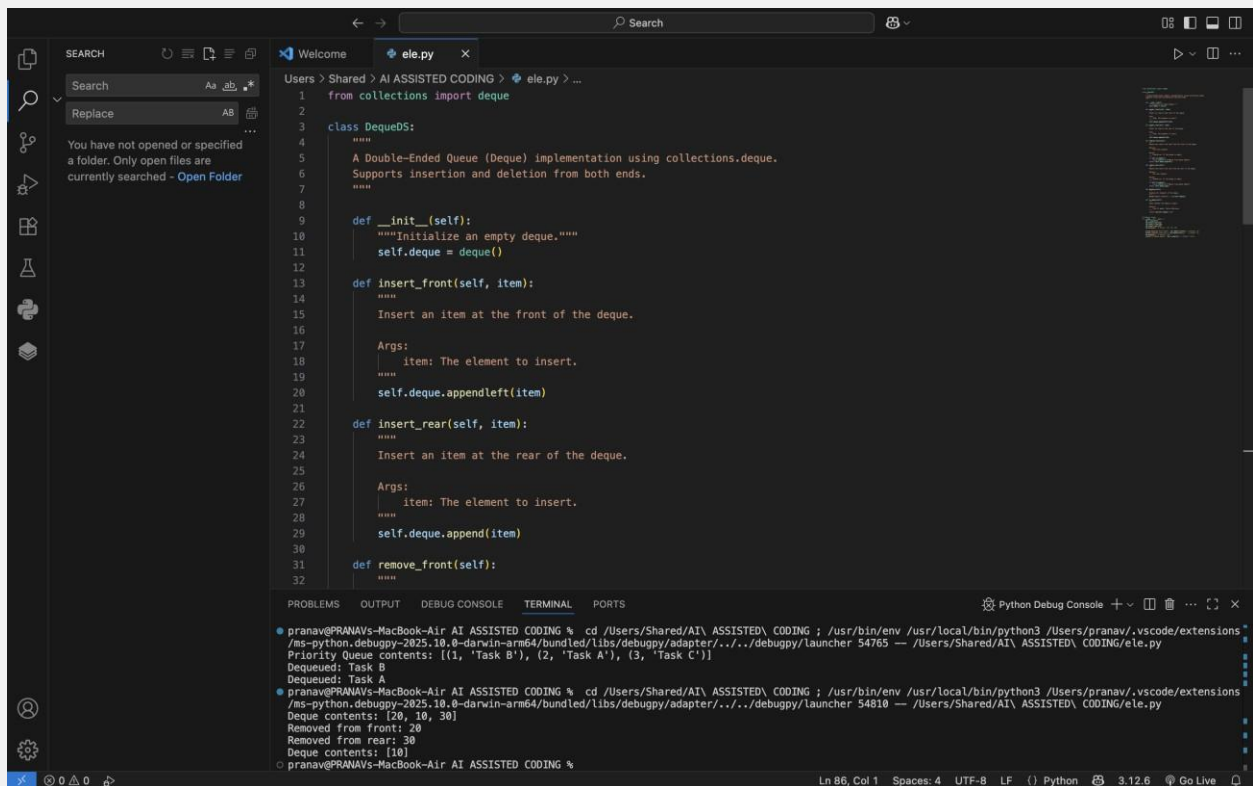
Sample Input Code:

```
class DequeDS:
```

```
pass
```

Expected Output:

- Insert and remove from both ends with docstrings




```

class DequeDS:
    def remove_rear(self):
        return self.deque.pop()

    def display(self):
        """
        Display all elements of the deque.
        """
        print("Deque contents:", list(self.deque))

    def is_empty(self):
        """
        Check whether the deque is empty.

        Returns:
            True if empty, False otherwise.
        """
        return len(self.deque) == 0

# Example usage:
if __name__ == "__main__":
    dq = DequeDS()
    dq.insert_rear(10)
    dq.insert_front(20)
    dq.insert_rear(30)
    dq.display() # Output: [20, 10, 30]

    print("Removed from front:", dq.remove_front()) # Output: 20
    print("Removed from rear:", dq.remove_rear()) # Output: 30
    print("Is deque empty?", dq.is_empty()) # Output: False
  
```

OUTPUT:

```

pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI ASSISTED CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions
/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54765 -- /Users/Shared/AI ASSISTED CODING
Priority Queue contents: [(1, 'Task B'), (2, 'Task A'), (3, 'Task C')]
Dequeued: Task B
Dequeued: Task A
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI ASSISTED CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pran
Deque contents: [20, 10, 30]
Removed from front: 20
Removed from rear: 30
Deque contents: [10]
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING %
  
```

OUTPUT:

```

print("Removed from front:", dq.remove_front()) # Output: 20
print("Removed from rear:", dq.remove_rear()) # Output: 30
print("Is deque empty?", dq.is_empty()) # Output: False
  
```

PROMPT :

To implement a double-ended queue using collections.deque

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

PROMPT :

To implement a double-ended queue using collections.deque

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

SEARCH

Search

Replace

You have not opened or specified a folder. Only open files are currently searched - Open Folder

Welcome ele.py 9+

Users > Shared > AI ASSISTED CODING > ele.py

```
1 # Data Structure Comparison Table
2
3 | Data Structure | Insertion (Push/Enqueue/Insert) | Deletion (Pop/Dequeue/Delete) | Access (Indexing/Peek) | Search | Notes |
4 |-----|-----|-----|-----|-----|-----|
5 | **Stack (List-based)** | O(1) (push) | O(1) (pop) | O(1) (peek) | O(n) | LIFO (Last-In, First-Out) |
6 | **Queue (List-based)** | O(1) (enqueue at end) | O(n) (dequeue from front) | O(1) (peek front) | O(n) | FIFO (First-In, First-Out); 'deque' is faster |
7 | **Queue ('deque')** | O(1) | O(1) | O(1) | O(n) | Efficient with 'collections.deque' |
8 | **Singly Linked List** | O(1) (insert at head) / O(n) (end) | O(1) (delete at head) / O(n) (search position) | O(n) | O(n) | Sequential traversal only |
9 | **Doubly Linked List** | O(1) (insert at head/tail if tracked) | O(1) (delete at head/tail if tracked) | O(n) | O(n) | Supports both directions |
10 | **Binary Search Tree (BST)** | O(log n) (avg), O(n) (worst) | O(log n) (avg), O(n) (worst) | O(log n) (avg) | O(log n) (avg) | Worst case is skewed tree |
11 | **Balanced BST (AVL/Red-Black)** | O(log n) | O(log n) | O(log n) | O(log n) | Always balanced |
12 | **Hash Table** | O(1) avg, O(n) worst | O(1) avg, O(n) worst | O(1) avg | O(1) avg | Uses hashing + chaining/probing |
13 | | | | O(1) (remove edge, if known) | O(V+E) traversal | O(V+E) | Space-efficient for sparse graphs |
14 | | | | O(1) (remove edge) | O(V) (check neighbors) | O(V^2) | Space O(V^2), better for dense graphs |
15 | **Tries** | O(m) (m = length of key) | O(m) | O(m) | O(m) | Efficient for prefix searches |
```

PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python Debug Console

```
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54765 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
Priority Queue contents: [(1, 'Task B'), (2, 'Task A'), (3, 'Task C')]
Dequeued: Task B
Dequeued: Task A
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54818 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
Deque contents: [20, 10, 30]
Removed from front: 20
Removed from rear: 30
Deque contents: [10]
```

SEARCH

Search

Replace

You have not opened or specified a folder. Only open files are currently searched - Open Folder

Welcome ele.py 9+

Users > Shared > AI ASSISTED CODING > ele.py

```
1
2
3 | Deletion (Pop/Dequeue/Delete) | Access (Indexing/Peek) | Search | Notes |
4 |-----|-----|-----|-----|
5 | O(1) (pop) | O(1) (peek) | O(n) | LIFO (Last-In, First-Out) |
6 | O(n) (dequeue from front) | O(1) (peek front) | O(n) | FIFO (First-In, First-Out); 'deque' is faster |
7 | O(1) | O(1) | O(n) | Efficient with 'collections.deque' |
8 | O(n) (end) | O(1) (delete at head) / O(n) (search position) | O(n) | O(n) | Sequential traversal only |
9 | O(1) (delete at head/tail if tracked) | O(n) | O(n) | Supports both directions |
10 | O(log n) (avg), O(n) (worst) | O(log n) (avg) | O(log n) (avg) | Worst case is skewed tree |
11 | O(log n) | O(log n) | O(log n) | Always balanced |
12 | O(1) avg, O(n) worst | O(1) avg | O(1) avg | Uses hashing + chaining/probing |
13 | O(1) (remove edge, if known) | O(V+E) traversal | O(V+E) | Space-efficient for sparse graphs |
14 | O(1) (remove edge) | O(V) (check neighbors) | O(V^2) | Space O(V^2), better for dense graphs |
15 | O(m) | O(m) | O(m) | Efficient for prefix searches |
```

PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python Debug Console

```
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54765 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
Priority Queue contents: [(1, 'Task B'), (2, 'Task A'), (3, 'Task C')]
Dequeued: Task B
Dequeued: Task A
pranav@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54818 -- /Users/Shared/AI\ ASSISTED\ CODING/ele.py
Deque contents: [20, 10, 30]
Removed from front: 20
Removed from rear: 30
Deque contents: [10]
```

PROMPT :

To generate a comparison table of different data structures (stack, queue, linked list, etc.) including time 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

```
class Queue:
    """
    A simple Queue implementation (FIFO - First In, First Out).
    Used here for cafeteria order management.
    """

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

    def enqueue(self, item):
        """
        Add an item (order) to the queue.

        Args:
            item (str): The order to add.
        """
        self._items.append(item)

    def dequeue(self):
        """
        Remove and return the next order in the queue.

        Returns:
            str: The order served next.

        Raises:
            IndexError: If the queue is empty.
        """
        if self.is_empty():
            raise IndexError("No orders to serve.")
        return self._items.pop(0)

    def is_empty(self):
        return len(self._items) == 0

    def peek(self):
        """View the next order without removing it.

        Returns:
            str: The next order in line.
        """
        if self.is_empty():
            raise IndexError("No orders to serve.")
        return self._items[0]
```

```
/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54810 -- /Users/Shared/AI/ ASSISTED\ CODING/ele.py
Deque contents: [20, 10, 30]
Removed from front: 20
Removed from rear: 30
Deque contents: [10]
pranave@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI/ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions
/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54847 -- /Users/Shared/AI/ ASSISTED\ CODING/ele.py
Cafeteria Order Queue: ['Order 1: Veg Sandwich', 'Order 2: Chicken Burger', 'Order 3: Cold Coffee']
Next order to serve: Order 1: Veg Sandwich
Serving: Order 1: Veg Sandwich
Cafeteria Order Queue: ['Order 2: Chicken Burger', 'Order 3: Cold Coffee']
pranave@PRANAVs-MacBook-Air AI ASSISTED CODING %
```

```
class Queue:
    """
    A simple Queue implementation (FIFO - First In, First Out).
    Used here for cafeteria order management.
    """

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

    def enqueue(self, item):
        """
        Add an item (order) to the queue.

        Args:
            item (str): The order to add.
        """
        self._items.append(item)

    def dequeue(self):
        """
        Remove and return the next order in the queue.

        Returns:
            str: The order served next.

        Raises:
            IndexError: If the queue is empty.
        """
        if self.is_empty():
            raise IndexError("No orders to serve.")
        return self._items.pop(0)

    def is_empty(self):
        return len(self._items) == 0

    def peek(self):
        """View the next order without removing it.

        Returns:
            str: The next order in line.
        """
        if self.is_empty():
            raise IndexError("No orders to serve.")
        return self._items[0]
```

```
/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54810 -- /Users/Shared/AI/ ASSISTED\ CODING/ele.py
Deque contents: [20, 10, 30]
Removed from front: 20
Removed from rear: 30
Deque contents: [10]
pranave@PRANAVs-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI/ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/.vscode/extensions
/ms-python.debugpy-2025.10.0-darwin-arm64/bundled/libs/debugpy/adapter/../../debugpy/launcher 54847 -- /Users/Shared/AI/ ASSISTED\ CODING/ele.py
Cafeteria Order Queue: ['Order 1: Veg Sandwich', 'Order 2: Chicken Burger', 'Order 3: Cold Coffee']
Next order to serve: Order 1: Veg Sandwich
Serving: Order 1: Veg Sandwich
Cafeteria Order Queue: ['Order 2: Chicken Burger', 'Order 3: Cold Coffee']
pranave@PRANAVs-MacBook-Air AI ASSISTED CODING %
```

OUTPUT:

```
Deque contents: [20, 10, 30]
Removed from front: 20
Removed from rear: 30
Deque contents: [10]
pranav@PRANAVS-MacBook-Air AI ASSISTED CODING % cd /Users/Shared/AI\ ASSISTED\ CODING ; /usr/bin/env /usr/local/bin/python3 /Users/pranav/anaconda3/bin/python3 -- /Users/Shared/AI\ ASSISTED\ CODING
Cafeteria Order Queue: ['Order 1: Veg Sandwich', 'Order 2: Chicken Burger', 'Order 3: Cold Coffee']
Next order to serve: Order 1: Veg Sandwich
Serving: Order 1: Veg Sandwich
Cafeteria Order Queue: ['Order 2: Chicken Burger', 'Order 3: Cold Coffee']
pranav@PRANAVS-MacBook-Air AI ASSISTED CODING %
```

PROMPT :

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.