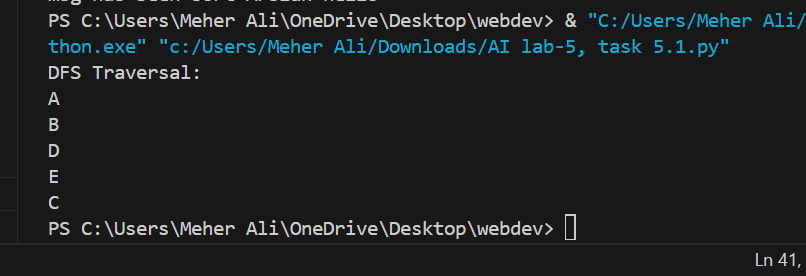
**LAB-5**

**TASK.1**

The Graph class represents an undirected graph using an adjacency list, where each node is an instance of the Node class. The Node class stores a value and a list of neighboring nodes. The Graph class provides methods to add nodes (add\_node()), create bidirectional edges (add\_edge()), and perform Depth-First Search (DFS) using a stack (dfs\_stack()). The DFS traversal starts from a given node and explores its neighbors iteratively, printing the traversal order. The program demonstrates these functionalities by constructing a graph with five nodes and performing DFS starting from node "A".



**TASK.2**

**In Depth-First Search (DFS) for trees, there are three primary traversal methods:**

**Preorder Traversal (Root → Left → Right)**

**Inorder Traversal (Left → Root → Right)**

**Postorder Traversal (Left → Right → Root)**

**These traversal methods are primarily used for binary trees.**

**Preorder Traversal (Root → Left → Right)**

**Visit the root node first, then the left subtree, and finally the right subtree.**

**Used for copying trees, prefix notation in expressions.**

**Example Traversal**

**Mathematica**

**A**

**/ \**

**B   C**

**/ \**

**D   E**

**Preorder: A → B → D → E → C**

**norder Traversal (Left → Root → Right)**

**Visit the left subtree first, then the root, and finally the right subtree.**

**Used for sorting and getting elements in ascending order.**

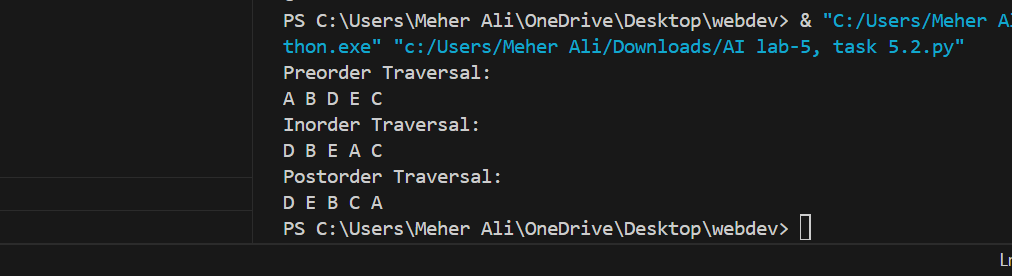
**Inorder: D → B → E → A → C**

**Postorder Traversal (Left → Right → Root)**

**Visit the left subtree first, then the right subtree, and finally the root.**

**Used for deleting trees, postfix notation in expressions.**

**Postorder: D → E → B → C → A**

****