Code Analysis: Depth First Search (DFS) Traversal

# 1. Introduction

This code implements the Depth First Search (DFS) algorithm for traversing a graph. DFS is a graph traversal technique that explores as far as possible along each branch before backtracking. It is commonly used in pathfinding, solving puzzles, and analyzing graph structures.

# 2. Graph Representation

The graph is represented using a dictionary in Python, where each key is a node and its corresponding value is a list of neighboring nodes (adjacency list representation).

Example from the code:  
graph = {  
 'A': ['B', 'C'],  
 'B': ['D', 'E'],  
 'C': ['F'],  
 'D': [],  
 'E': ['F'],  
 'F': []  
}

# 3. Initialization of Variables

- start = 'A' → Starting node of the traversal.  
- stack = [start] → A stack used to keep track of nodes to visit.  
- visited = [] → A list to store nodes that have already been visited.

# 4. DFS Algorithm Explanation

The algorithm works as follows:  
1. While the stack is not empty, pop a node from the stack.  
2. If the node has not been visited, add it to the visited list.  
3. Retrieve all neighbors of the node from the graph.  
4. Push neighbors onto the stack in reverse order so that the leftmost neighbor is processed first.  
5. Continue until the stack becomes empty.

# 5. Why Use DFS?

DFS is useful because:  
- It explores deep paths before shallow ones.  
- It is memory efficient compared to Breadth First Search (BFS) when the graph has many branches.  
- It can be used to detect cycles, check connectivity, and solve problems such as maze traversal.

# Output

