# Depth-First Search (DFS) with Stack & Node and Tree Traversals - Simple Guide

## Task 1: DFS with Stack & Node

What is DFS?  
Depth-First Search (DFS) is a way to explore a graph by going deep into one branch before backtracking. Here, we use a stack to keep track of nodes.

### Simple Code for DFS using Stack

class Node:  
 def \_\_init\_\_(self, value):  
 self.value = value  
 self.neighbors = []  
  
def dfs(start\_node):  
 stack = [start\_node]   
 visited = set()   
  
 while stack:  
 node = stack.pop()   
 if node not in visited:  
 print(node.value, end=" ")   
 visited.add(node)  
 stack.extend(reversed(node.neighbors))   
  
  
a = Node('A')  
b = Node('B')  
c = Node('C')  
d = Node('D')  
e = Node('E')  
  
a.neighbors = [b, c]  
b.neighbors = [d, e]  
c.neighbors = [e]  
d.neighbors = []  
e.neighbors = []  
  
  
dfs(a)

### Output

A B D E C

## Task 2: Tree Traversals - Inorder, Preorder, Postorder using DFS

What is Tree Traversal?  
When working with trees, DFS is used to visit nodes in different orders:  
1. Preorder (Root → Left → Right)  
2. Inorder (Left → Root → Right)  
3. Postorder (Left → Right → Root)

### Simple Code for Tree Traversals

class Node:  
 def \_\_init\_\_(self, value):  
 self.value = value  
 self.left = None  
 self.right = None  
  
# DFS Traversals  
def inorder(node):   
 if node:  
 inorder(node.left)  
 print(node.value, end=" ")  
 inorder(node.right)  
  
def preorder(node):   
 if node:  
 print(node.value, end=" ")  
 preorder(node.left)  
 preorder(node.right)  
  
def postorder(node):   
 if node:  
 postorder(node.left)  
 postorder(node.right)  
 print(node.value, end=" ")  
  
  
root = Node('A')  
root.left = Node('B')  
root.right = Node('C')  
root.left.left = Node('D')  
root.left.right = Node('E')  
root.right.left = Node('F')  
root.right.right = Node('G')  
  
  
print("Inorder Traversal: ", end="")   
inorder(root)  
print("\nPreorder Traversal: ", end="")  
preorder(root)  
print("\nPostorder Traversal: ", end="")  
postorder(root)

### Tree Structure

A  
 / \  
 B C  
 / \ / \  
 D E F G

### Output

Inorder Traversal: D B E A F C G  
Preorder Traversal: A B D E C F G  
Postorder Traversal: D E B F G C A