class Graph:

def \_\_init\_\_(self):

self.graph = {}

def add\_edge(self, u, v):

if u not in self.graph:

self.graph[u] = []

self.graph[u].append(v)

def dfs(graph, start):

visited = set()

stack = [start]

while stack:

node = stack.pop()

if node not in visited:

print(node, end=' ')

visited.add(node)

stack.extend(reversed(graph.get(node, [])))

g = Graph()

# Input graph edges

while True:

u = input("Enter start node of edge (or 'done' to finish): ")

if u.lower() == 'done':

break

v = input("Enter end node of edge: ")

g.add\_edge(u, v)

start\_node = input("Enter the starting node for DFS: ")

print("DFS Traversal:")

dfs(g.graph, start\_node)

Output:

Enter start node of edge (or 'done' to finish): A

Enter end node of edge: B

Enter start node of edge (or 'done' to finish): A

Enter end node of edge: C

Enter start node of edge (or 'done' to finish): B

Enter end node of edge: D

Enter start node of edge (or 'done' to finish): C

Enter end node of edge: D

Enter start node of edge (or 'done' to finish): D

Enter end node of edge: E

Enter start node of edge (or 'done' to finish): done

Enter the starting node for DFS: A

DFS Traversal:

A B D E C