# Python3 program to print DFS traversal

# from a given graph

from collections import defaultdict

# This class represents a directed graph using

# adjacency list representation

class Graph:

# Constructor

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

# default dictionary to store graph

self.graph = defaultdict(list)

# function to add an edge to graph

def addEdge(self, u, v):

self.graph[u].append(v)

# A function used by DFS

def DFSUtil(self, v, visited):

# Mark the current node as visited

# and print it

visited.add(v)

print(v, end=' ')

# Recur for all the vertices

# adjacent to this vertex

for neighbour in self.graph[v]:

if neighbour not in visited:

self.DFSUtil(neighbour, visited)

# The function to do DFS traversal. It uses

# recursive DFSUtil()

def DFS(self, v):

# Create a set to store visited vertices

visited = set()

# Call the recursive helper function

# to print DFS traversal

self.DFSUtil(v, visited)

# Driver's code

# Create a graph given

# in the above diagram

if \_\_name\_\_ == "\_\_main\_\_":

g = Graph()

g.addEdge(0, 1)

g.addEdge(0, 2)

g.addEdge(1, 2)

g.addEdge(2, 0)

g.addEdge(2, 3)

g.addEdge(3, 3)

print("Following is DFS from (starting from vertex 2)")

# Function call

g.DFS(2)