**# 2** **Write a Program to implement Uninformed search technique: DFS**

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[v]= True

print (v)

# Recur for all the vertices adjacent to

# this vertex

for i in self.graph[v]:

if visited[i] == False:

self.DFSUtil(i, visited)

# The function to do DFS traversal. It uses

# recursive DFSUtil()

def DFS(self):

V = len(self.graph) #total vertices

# Mark all the vertices as not visited

visited =[False]\*(V)

# Call the recursive helper function to print

# DFS traversal starting from all vertices one

# by one

for i in range(V):

if visited[i] == False:

self.DFSUtil(i, visited)

# Driver code

# Create a graph given in the above diagram

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 Depth First Traversal")

g.DFS()

**Output:**

**Following is Depth First Traversal**

**0**

**3**

**2**

**1**