**# 1. Python program to print BFS traversal for complete graph**

from collections import defaultdict

# This class represents a 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))

self.graph[v].append((u))

def dispg(self):

print(self.graph.items())

#Function to print a BFS of graph

def BFS(self, s,goal):

# Mark all the vertices as not visited

visited = [False] \* (len(self.graph))

# Create a queue for BFS

frontier\_q = []

# Mark the source node as visited and enqueue it

frontier\_q.append(s)

visited[s] = True

while frontier\_q:

# Dequeue a vertex from queue and print it

s = frontier\_q.pop(0)

print (s, end = " ")

if(s == goal):

print("\n Goal found")

return

# Get all adjacent vertices of the dequeued vertex s. # If adjacent hasnot been visited, then mark it visited and enqueue it

for i in self.graph[s]:

if visited[i] == False:

frontier\_q.append(i)

visited[i] = True

print("\n Goal not found")

# Driver code

# Create a graph given in the above diagram

g = Graph()

g.addEdge(0,1)

g.addEdge(0,3)

g.addEdge(1,2)

g.addEdge(2,3)

g.addEdge(3,4)

g.addEdge(4,5)

g.addEdge(3,5)

g.dispg()

print ("Following is Breadth First Traversal (starting from vertex 1)")

g.BFS(1,4)

**OUTPUT:**

dict\_items([(0, [1, 3]), (1, [0, 2]), (3, [0, 2, 4, 5]), (2, [1, 3]), (4, [3, 5]), (5, [4, 3])])

Following is Breadth First Traversal (starting from vertex 1)

1

Goal not found

Goal not found

0

Goal not found

2 3

Goal not found

Goal not found

4

Goal found