7. Write the python program to implement BFS.

**AIM :** program to implement BFS.

**ALGORITHM :**

1. Start from the source vertex and enqueue it into the queue.
2. Mark the source vertex as visited.
3. While the queue is not empty:

* Dequeue a vertex from the queue.
* Process the vertex (print it or perform some operation).
* Enqueue all adjacent vertices of the dequeued vertex that are not visited and mark them as visited.

4. Repeat step 3 until the queue is empty.

**PROGRAM :**

from collections import defaultdict, deque

class Graph:

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

self.graph = defaultdict(list)

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

self.graph[u].append(v)

def bfs(self, start\_vertex):

visited = set()

queue = deque([start\_vertex])

visited.add(start\_vertex)

while queue:

vertex = queue.popleft()

print(vertex, end=" ")

for neighbor in self.graph[vertex]:

if neighbor not in visited:

queue.append(neighbor)

visited.add(neighbor)

graph = Graph()

graph.add\_edge(0, 1)

graph.add\_edge(0, 2)

graph.add\_edge(1, 2)

graph.add\_edge(2, 0)

graph.add\_edge(2, 3)

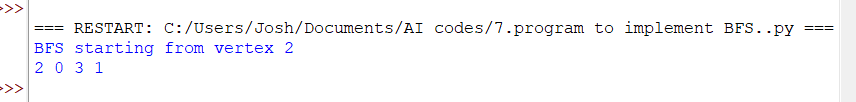
graph.add\_edge(3, 3)

start\_vertex = 2

print("BFS starting from vertex", start\_vertex)

graph.bfs(start\_vertex)

**OUT PUT :**

****