**PROGRAM :**

class Graph:

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

self.v = vertices

self.graph = [[0 for column in range(vertices)] for row in range(vertices)]

# A utility function to check if the current color assignment is safe for vertex v

def is\_safe(self, v, color, c):

for i in range(self.v):

if self.graph[v][i] == 1 and color[i] == c:

return False

return True

# A recursive utility function to solve m-coloring problem

def graph\_color\_util(self, m, color, v):

if v == self.v:

return True

for c in range(1, m+1):

if self.is\_safe(v, color, c):

color[v] = c

if self.graph\_color\_util(m, color, v+1):

return True

color[v] = 0

def graph\_coloring(self, m):

color = [0] \* self.v

if not self.graph\_color\_util(m, color, 0):

return False

# Print the solution

print("Solution exists and following are the assigned colors:")

for c in color:

print(c, end=" ")

# Driver Code

if \_\_name\_\_ == '\_\_main\_\_':

g = Graph(4)

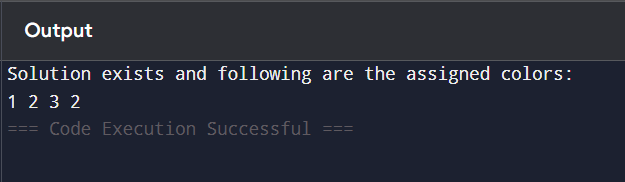
g.graph = [[0, 1, 1, 1], [1, 0, 1, 0], [1, 1, 0, 1], [1, 0, 1, 0]]

m = 3

# Function call

g.graph\_coloring(m)

**OUTPUT :**

****