class MapColoring:

def \_\_init\_\_(self, graph, colors):

self.graph = graph

self.colors = colors

self.color\_assignment = {node: None for node in graph}

def is\_safe(self, node, color):

for neighbor in self.graph[node]:

if self.color\_assignment[neighbor] == color:

return False

return True

def backtrack(self, node):

if node == len(self.graph):

return True

for color in self.colors:

if self.is\_safe(node, color):

self.color\_assignment[node] = color

if self.backtrack(node + 1):

return True

self.color\_assignment[node] = None

return False

def solve(self):

if self.backtrack(0):

return self.color\_assignment

else:

return None

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

graph = {

0: [1, 2],

1: [0, 2, 3],

2: [0, 1, 3],

3: [1, 2, 4],

4: [3]

}

colors = ['Red', 'Green', 'Blue']

map\_coloring = MapColoring(graph, colors)

solution = map\_coloring.solve()

if solution:

print("Color assignment:", solution)

else:

print("No solution found.")