# Exercise 1: Implement the Constraint Satisfaction Problem of Map Colouring

**Tool**

Python

**Problem Statement**

We wish to color the 4 states in southern India, them being Andhra Pradesh, Karnataka, Tamil Nadu and Kerala such that no two neighbors should be sharing the same colors.

The following lists out the neighbors for a particular state.

neighbors['Andhra'] = ['Karnataka', 'TamilNadu']  
neighbors['Karnataka'] = ['Andhra', 'TamilNadu', 'Kerala']  
neighbors['TamilNadu'] = ['Andhra', 'Karnataka', 'Kerala']  
neighbors['Kerala'] = ['Karnataka', 'TamilNadu']  
  
We would represent the given information using an undirected 2D graph where the adjacency matrix graph[i][j] = 1 represents that i and j are neighbors and 0 represents otherwise.

**Input**

graph = [[0, 1, 1, 0],  
 [1, 0, 1, 1],  
 [1, 1, 0, 1],  
 [0, 1, 1, 0]]  
where graph[i][j] = 1 represents that i and j are neighbors and 0 represents otherwise. Row in order represent Andhra Pradesh, Karnataka, Tamil Nadu and Kerala.

**Logic / Search Technique**

Backtracking

**Implementation**

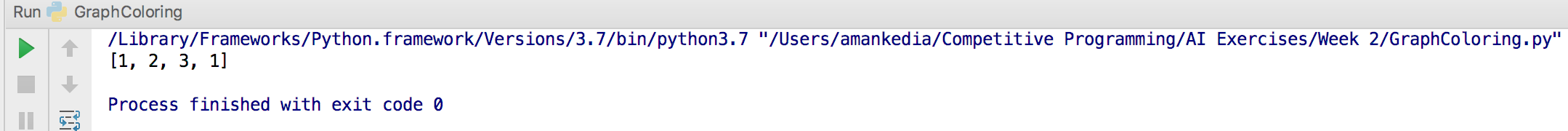
**def** is\_safe(n, graph, colors, c):  
 *# Iterate trough adjacent vertices  
 # and check if the vertex color is different from c* **for** i **in** range(n):  
 **if** graph[n][i] **and** c == colors[i]: **return False  
 return True***# n = vertex nb***def** graphColoringUtil(graph, color\_nb, colors, n):  
 *# Check if all vertices are assigned a color* **if** color\_nb+1 == n :  
 **return True** *# Trying differents color for the vertex n* **for** c **in** range(1, color\_nb+1):  
 *# Check if assignment of color c to n is possible* **if** is\_safe(n, graph, colors, c):  
 *# Assign color c to n* colors[n] = c  
 *# Recursively assign colors to the rest of the vertices* **if** graphColoringUtil(graph, color\_nb, colors, n+1): **return True** *# If there is no solution, remove color (BACKTRACK)* colors[n] = 0  
  
*#nb of vertex*vertex\_nb = 4  
*# nb of colors*color\_nb = 3  
*# Initiate vertex colors*colors = [0] \* vertex\_nb  
  
graph = [  
 [0, 1, 1, 0],  
 [1, 0, 1, 1],  
 [1, 1, 0, 1],  
 [0, 1, 1, 0]  
]  
  
*#beginning with vertex 0***if** graphColoringUtil(graph, color\_nb, colors, 0):  
 print(colors)  
**else**:  
 print(**"No solution"**)

**Output**

An array color[V] that should have numbers from 1 to m. color[i] should represent the color assigned to the ith state. The code should also return false if the graph cannot be colored with m colors.

[1, 2, 3, 1]

**Screenshot**

****

**TODO**

Extend this problem by modifying the input for all states of India and return the color palette for it.