Computer Engineering Department

BIM 309 – Artificial Intelligence

Homework 2
Report

Prepared By
Necip Şükrü
Aşık

Date: 29.12.2021

In this homework we have graph which contains countries and their neighbors. I tried to mapping this country and their neighbors and coloring them with at most 4 colours. When we coloring it we have a rule which is no two neighbor are colored with the same color. I track to follow the colors to ensure this rule with backtracking algorithm.

Here's the list:

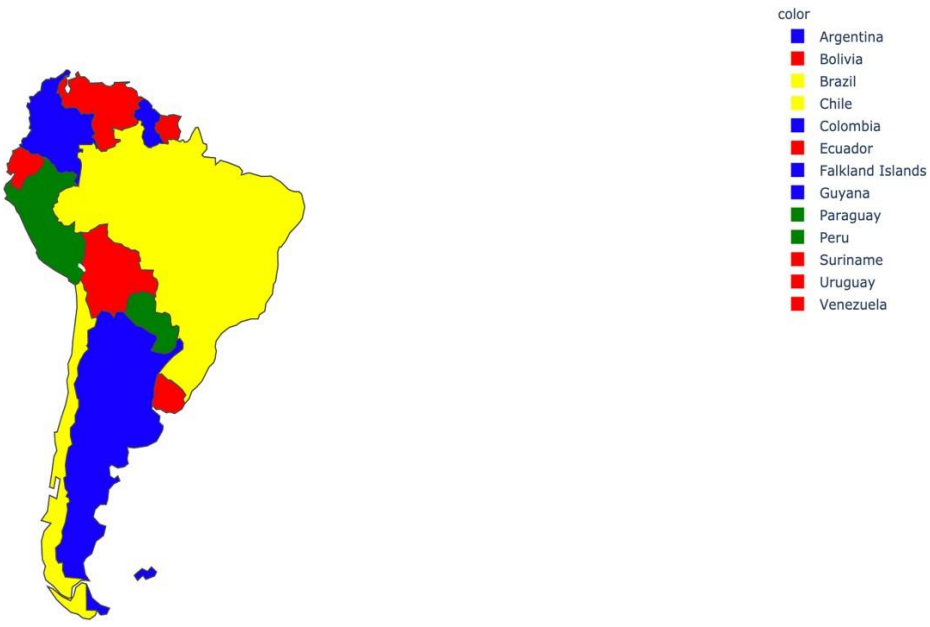
Country	Border Neighbors
Argentina	Bolivia, Chile, Paraguay, Uruguay
Bolivia	Argentina, Brazil, Chile, Paraguay, Peru
Brazil	Argentina, Bolivia, Colombia, Guyana, Paraguay, Peru, Suriname, Uruguay, Venezuela
Chile	Argentina, Bolivia, Peru
Colombia	Brazil, Ecuador, Peru, Venezuela
Ecuador	Colombia, Peru
Falkland Islands	NONE
Guyana	Brazil, Suriname, Venezuela
Paraguay	Argentina, Bolivia, Brazil
Peru	Bolivia, Brazil, Chile, Colombia, Ecuador
Suriname	Brazil, Guyana
Uruguay	Argentina, Brazil
Venezuela	Brazil, Colombia, Guyana

And I wrote the list in code like this:

```
graph = {"Argentina" : {'Bolivia','Chile','Paraguay',"Uruguay"},
         "Bolivia" : {'Argentina', 'Brazil', 'Chile', 'Paraguay', 'Peru'},
         "Brazil" : {'Argentina', 'Bolivia', 'Colombia', 'Guyana', 'Paraguay', 'Peru',
                     'Suriname', 'Uruguay', 'Venezuela'},
         "Chile" : {'Argentina', 'Bolivia', 'Peru'},
         "Colombia" : {'Brazil', 'Ecuador', 'Peru', 'Venezuela'},
         "Ecuador": {'Colombia', 'Peru'},
         "Falkland Islands" : {"Falkland Islands"},
         "Guyana" : {'Brazil', 'Suriname', 'Venezuela'},
         "Paraguay" : {'Argentina', 'Bolivia', 'Brazil'},
         "Peru" : {'Bolivia', 'Brazil', 'Chile', 'Colombia', 'Ecuador'},
         "Suriname" : {'Brazil', 'Guyana'},
         "Uruguay" : {'Argentina', 'Brazil'},
         "Venezuela" : {'Brazil', 'Colombia', 'Guyana'}}
```

Here’s to example of expected map:

And that’s the output of my code:

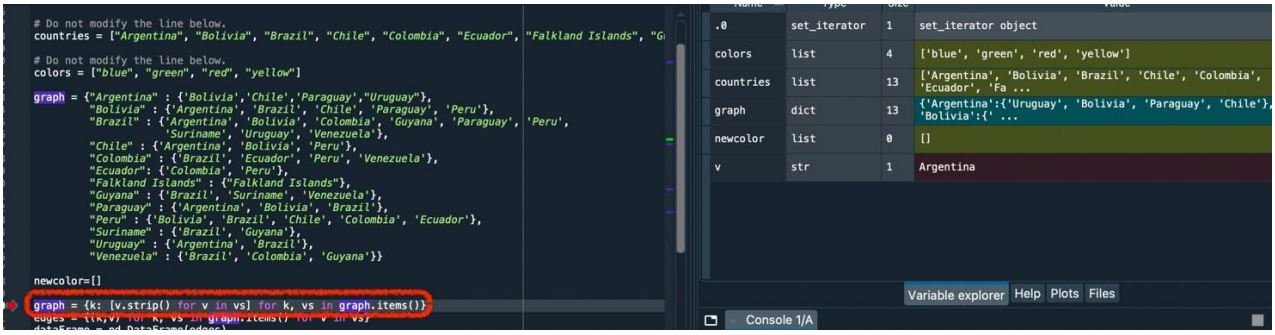


Now I will try to explain my code.

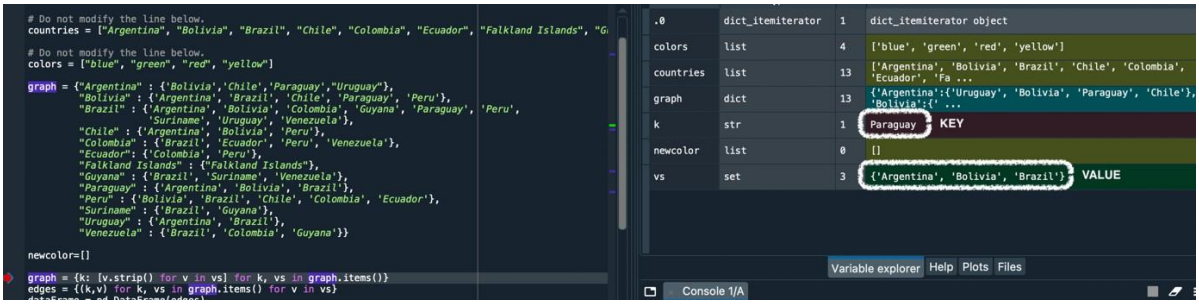
In first part I added the dictionary which named “graph” and “newcolor” list for saving the founded colors.

```
import plotly.express as px
import pandas as pd
import plotly.io as pio
import numpy as np
pio.renderers.default='browser'

# Do not modify the line below.
countries = ["Argentina", "Bolivia", "Brazil", "Chile", "Colombia", "Ecuador", "Falkland Islands", "Guy

# Do not modify the line below.
colors = ["blue", "green", "red", "yellow"]

graph = {"Argentina" : {'Bolivia','Chile','Paraguay',"Uruguay"},
         "Bolivia" : {'Argentina', 'Brazil', 'Chile', 'Paraguay', 'Peru'},
         "Brazil" : {'Argentina', 'Bolivia', 'Colombia', 'Guyana', 'Paraguay', 'Peru',
                     'Suriname', 'Uruguay', 'Venezuela'},
         "Chile" : {'Argentina', 'Bolivia', 'Peru'},
         "Colombia" : {'Brazil', 'Ecuador', 'Peru', 'Venezuela'},
         "Ecuador": {'Colombia', 'Peru'},
         "Falkland Islands" : {"Falkland Islands"},
         "Guyana" : {'Brazil', 'Suriname', 'Venezuela'},
         "Paraguay" : {'Argentina', 'Bolivia', 'Brazil'},
         "Peru" : {'Bolivia', 'Brazil', 'Chile', 'Colombia', 'Ecuador'},
         "Suriname" : {'Brazil', 'Guyana'},
         "Uruguay" : {'Argentina', 'Brazil'},
         "Venezuela" : {'Brazil', 'Colombia', 'Guyana'}}

newcolor=[]
```

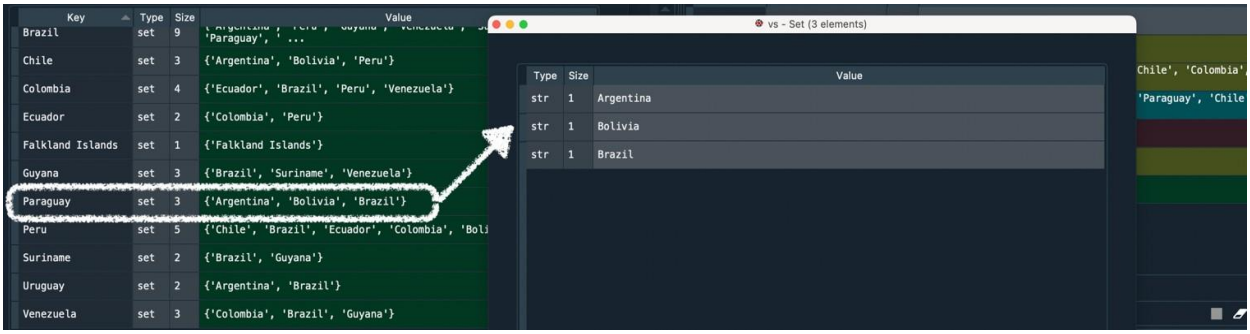
In this line, I traveled graph items with for:



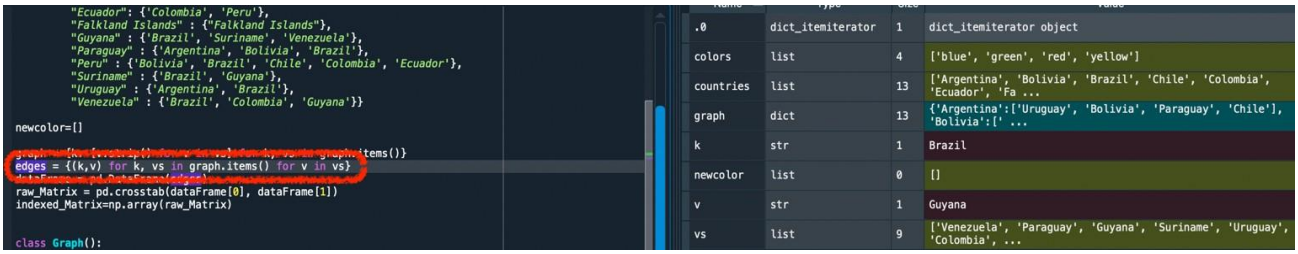
K variable holds the keys and vs variable holds the values:



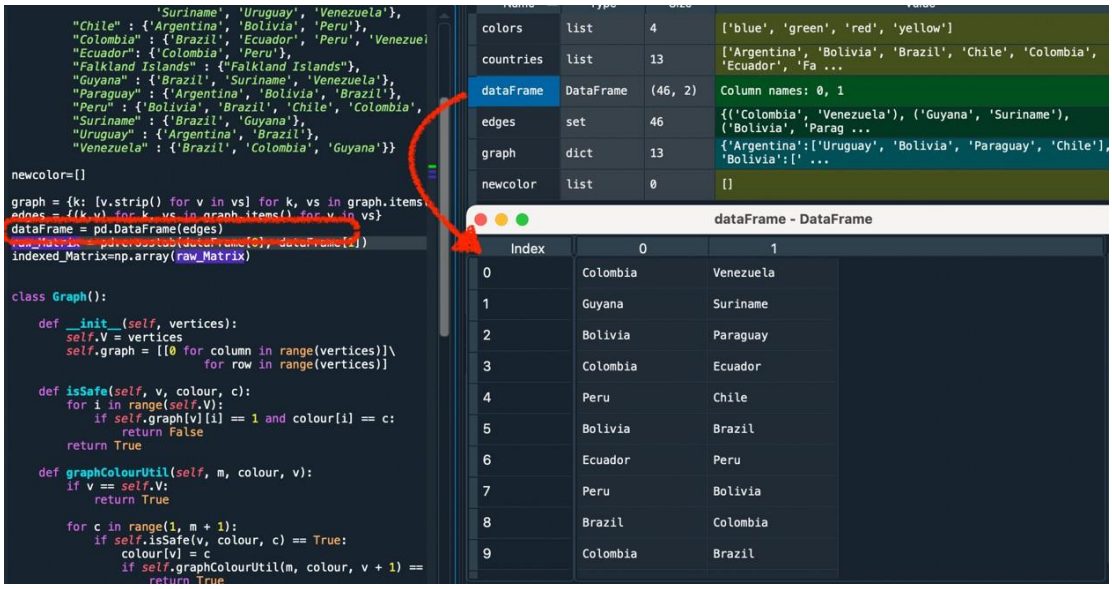
With this line I saved which country has which neighbors and I will used this variables with next line :



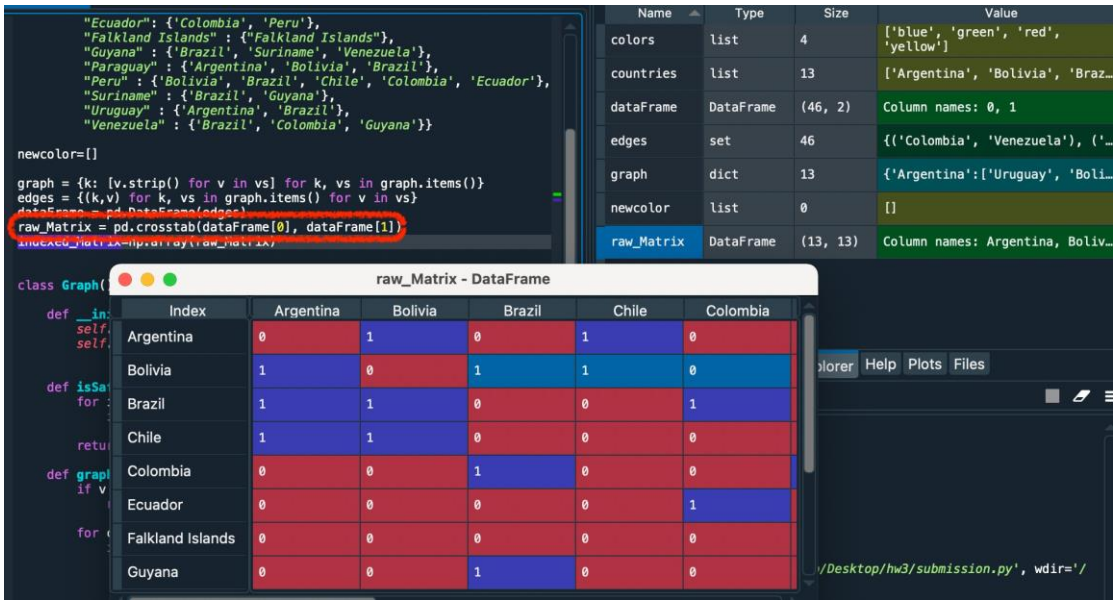
Then in “edges” line I had “vs” from previous code, with using it, I saved all country-neighbour pairs in the “edges”:



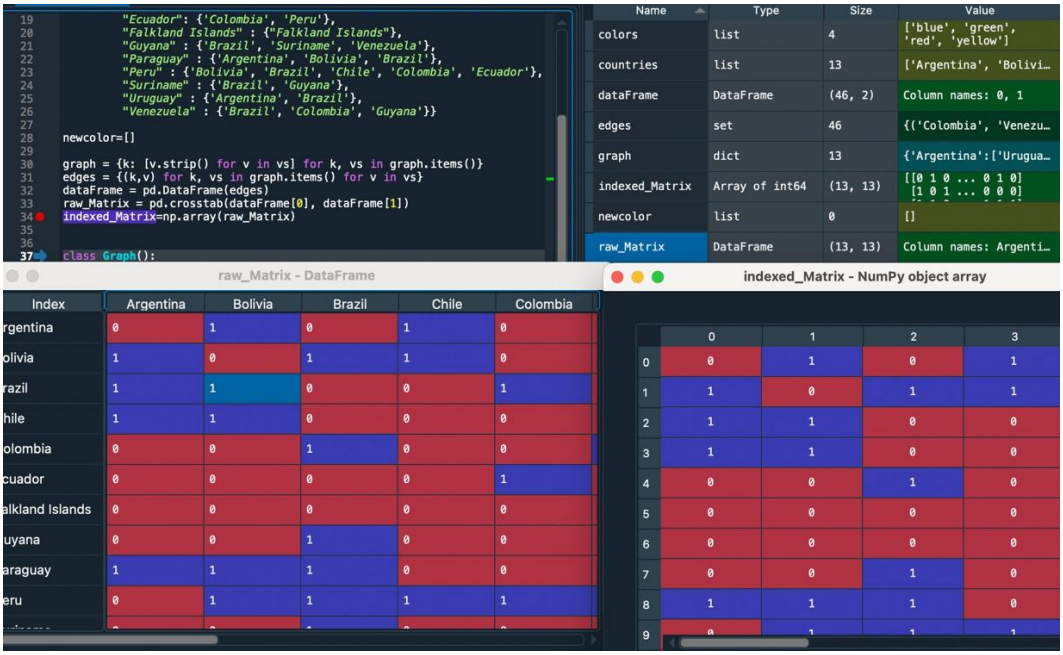
Then with “dataFrame” I gave indexes all this tuples for identification:



With crosstab I made a simple cross tabulation, in this way I made a matrix table of contry-neighbours. For example if Argentina’s neighbours are Bolivia Chile etc. their values are 1, if they are not their values are 0 :



Then we indexed all of this values again for identification :

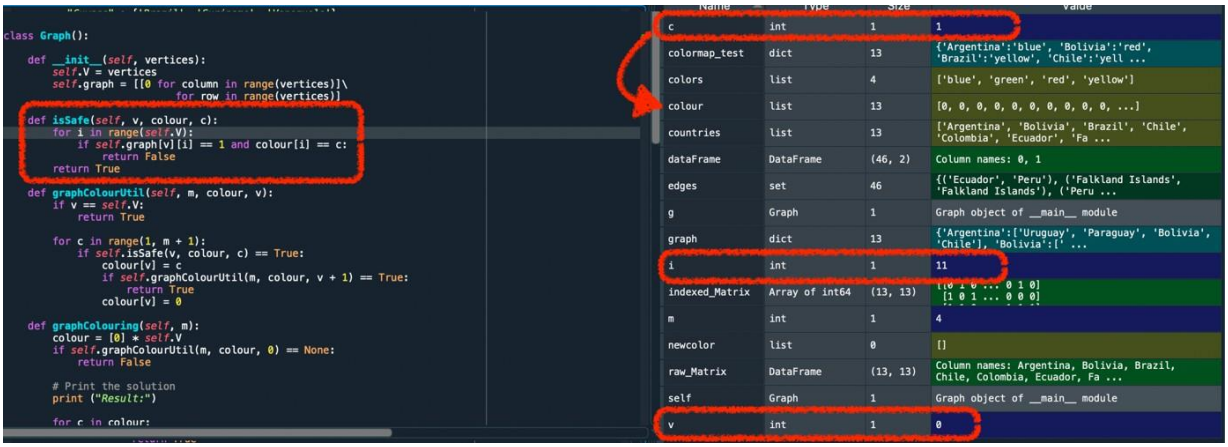


In Backtracking Algorithm, I try to assign colors one by one. Before assigning I checked if the color is already assigned to the neighbors, if this condition is supported I mark map with this color. If no color is possible for that region then backtrack and return false.

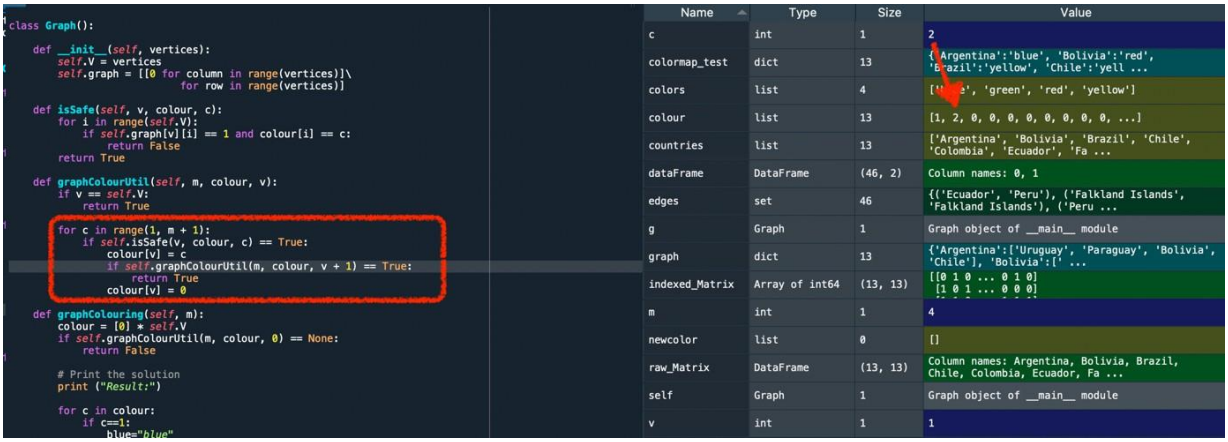
In main, I say that graph's size is 13 an this graph is equal to "indexed_Matrix". Then in top of the code we have "colors", I take the colors lenght and equal it to "m".

```
# Implement main to call necessary functions
if __name__ == "__main__":


    # coloring test
    colormap_test = {"Argentina": "blue",
                     "Bolivia": "red",
                     "Brazil": "yellow",
                     "Chile": "yellow",
                     "Colombia": "red",
                     "Ecuador": "yellow",
                     "Falkland Islands": "yellow",
                     "Guyana": "red",
                     "Paraguay": "green",
                     "Peru": "green",
                     "Suriname": "green",
                     "Uruguay": "red",
                     "Venezuela": "green"}

    g = Graph(13)
    g.graph =indexed_Matrix
    m = len(colors)
    g.graphColouring(m)

    my_dict = {k: v for k, v in zip(countries, newcolor)}
    plot_choropleth(my_dict)
```

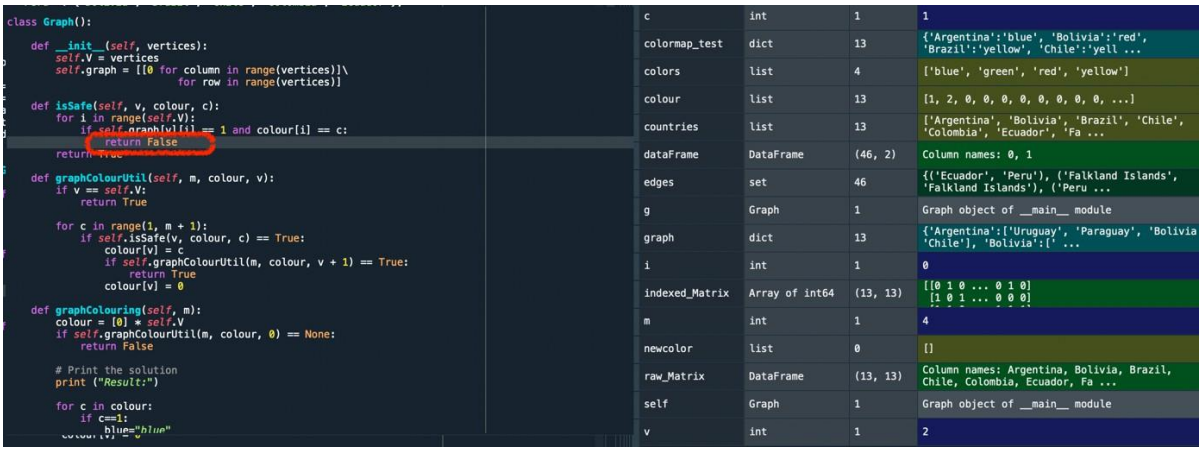
In “isSafe”, I controled the colors of neighbours, for ensure to their colors is not same. If graph[v][i]==1 and colour[i]==c is not true (v is 13 which comes from Graph(13) and c is number which indicates colors), continue with for loop from 0 to 12 and pass other function, if it is true I need to change the color:



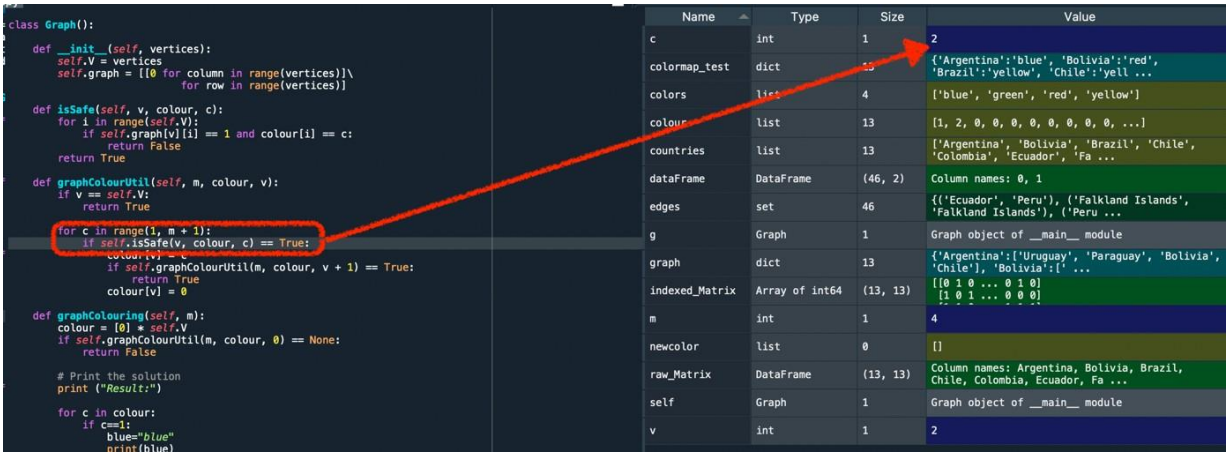
If for loop is done (for each operation) continue with this part. In this part c equaled to colour[v] and adding number of color to “colours” list then I increase the v and return the for loop again:



But if graph[v][i]==1 and colour[i]==c is not true:



We continue with this line. With this line c is increase and this means color changed :



```
class Graph():

    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]\
                      for row in range(vertices)]

    def isSafe(self, v, colour, c):
        for i in range(self.V):
            if self.graph[v][i] == 1 and colour[i] == c:
                return False
        return True

    def graphColourUtil(self, m, colour, v):
        if v == self.V:
            return True

        for c in range(1, m + 1):
            if self.isSafe(v, colour, c) == True:
                colour[v] = c
                if self.graphColourUtil(m, colour, v + 1) == True:
                    return True
                colour[v] = 0

    def graphColouring(self, m):
        colour = [0] * self.V
        if self.graphColourUtil(m, colour, 0) == None:
            return False

        # Print the solution
        print ("Result:")

        for c in colour:
            if c==1:
                blue="blue"
                print(blue)
```

Name	Type	Size	Value
c	int	1	2
colormap_test	dict	13	{'Argentina':'blue', 'Bolivia':'red', 'Brazil':'yellow', 'Chile':'yell ...
colors	list	4	['blue', 'green', 'red', 'yellow']
colour	list	13	[1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...]
countries	list	13	['Argentina', 'Bolivia', 'Brazil', 'Chile', 'Colombia', 'Ecuador', 'Fa ...
dataFrame	DataFrame	(46, 2)	Column names: 0, 1
edges	set	46	{('Ecuador', 'Peru'), ('Falkland Islands', 'Falkland Islands'), ('Peru ...
g	Graph	1	Graph object of __main__ module
graph	dict	13	{'Argentina':['Uruguay', 'Paraguay', 'Bolivia', 'Chile'], 'Bolivia':[' ...
indexed_Matrix	Array of int64	(13, 13)	[[0 1 0 ... 0 1 0] [1 0 1 ... 0 0 0]
m	int	1	4
newcolor	list	0	[]
raw_Matrix	DataFrame	(13, 13)	Column names: Argentina, Bolivia, Brazil, Chile, Colombia, Ecuador, Fa ...
self	Graph	1	Graph object of __main__ module
v	int	1	2

These loops occur one by one for 13 variables. And finally “colours” list going to be like this :

Indi	Type	Size	Value
0	int	1	1
1	int	1	2
2	int	1	3
3	int	1	3
4	int	1	1
5	int	1	2
6	int	1	1
7	int	1	1
8	int	1	4
9	int	1	4
10	int	1	2
11	int	1	2

Then I convert these numbers to colors with this code (If number is 1 convert it to blue, if 2 convert to red, if 3 convert to yellow and if 4 convert to green.) :

```
def graphColouring(self, m):
    colour = [0] * self.V
    if self.graphColourUtil(m, colour, 0) == None:
        return False

    # Print the solution
    print ("Result:")

    for c in colour:
        if c==1:
            blue="blue"
            print(blue)
            newcolor.append(blue)

        elif c==2:
            red="red"
            print("red")
            newcolor.append(red)

        elif c==3:
            yellow="yellow"
            print(yellow)
            newcolor.append(yellow)

        else:
            green="green"
            print(green)
            newcolor.append(green)

    return True
```

Then final “newcolor” list is :

Ind	Type	Size	Value
0	str	1	blue
1	str	1	red
2	str	1	yellow
3	str	1	yellow
4	str	1	blue
5	str	1	red
6	str	1	blue
7	str	1	blue
8	str	1	green
9	str	1	green
10	str	1	red
11	str	1	red
12	str	1	red

But I need to merge newcolors and countries so I merge them with this line and make it a dictionary :

```
g.graphColouring(m)

my_dict = {k: v for k, v in zip(countries, newcolor)}
plot_choropleth(my_dict)
```

In this code I used these libraries. I use pio.renderers.default=’browser’ for seeing map in browser :

```
import plotly.express as px
import pandas as pd
import plotly.io as pio
import numpy as np
pio.renderers.default='browser'
```