Agent programs for real world problems

**Ex no.:** 2 **Reg No.:** RA1911003010917

**Date:** 01/02/2022 **Name:** Amrit Aanand

**Aim:** To develop agent programs for real world problem: Graph colouring problem

Description: Graph colouring is the procedure of assignment of colours to each vertex of a graph G such that no adjacent vertices get same colour. The objective is to minimise the number of colours while colouring a graph. The smallest number of colours required to colour a graph G is called its chromatic number of that graph.

# Method to Colour a Graph

The steps required to colour a graph G with n number of vertices are as follows –

* Arrange the vertices of the graph in some order.
* Choose the first vertex and colour it with the first colour. Choose the next vertex and colour it with the lowest numbered colour that has not been coloured on any vertices adjacent to it.
* If all the adjacent vertices are coloured with this colour, assign a new colour to it. Repeat this step until all the vertices are coloured.

**Code:**

# Edge Colouring:

import matplotlib.pyplot as plt

import networkx as nx

from matplotlib.patches import Polygon

import numpy as np

G = nx.Graph()

colors = {0:"red", 1:"green", 2:"blue", 3:"yellow"}

G.add\_nodes\_from([1,2,3,4,5])

G.add\_edges\_from([(1,2), (1,3), (2,4), (3,5), (4,5)])

nodes = list(G.nodes)

edges = list(G.edges)

color\_lists = []

color\_of\_edge = []

some\_colors = ['red','green','blue','yellow']

for i in range(len(nodes) + 1):

color\_lists.append([])

color\_of\_edge.append(-1)

def getSmallestColor(ls1,ls2):

i = 1

while(i in ls1 or i in ls2):

i = i + 1

return i

#iterate over edges

i = 0

for ed in edges:

newColor = getSmallestColor(color\_lists[ed[0]],color\_lists[ed[1]])

color\_lists[ed[0]].append(newColor)

color\_lists[ed[1]].append(newColor)

color\_of\_edge[i] = newColor

i = i + 1

# Makin graph again

G = nx.Graph()

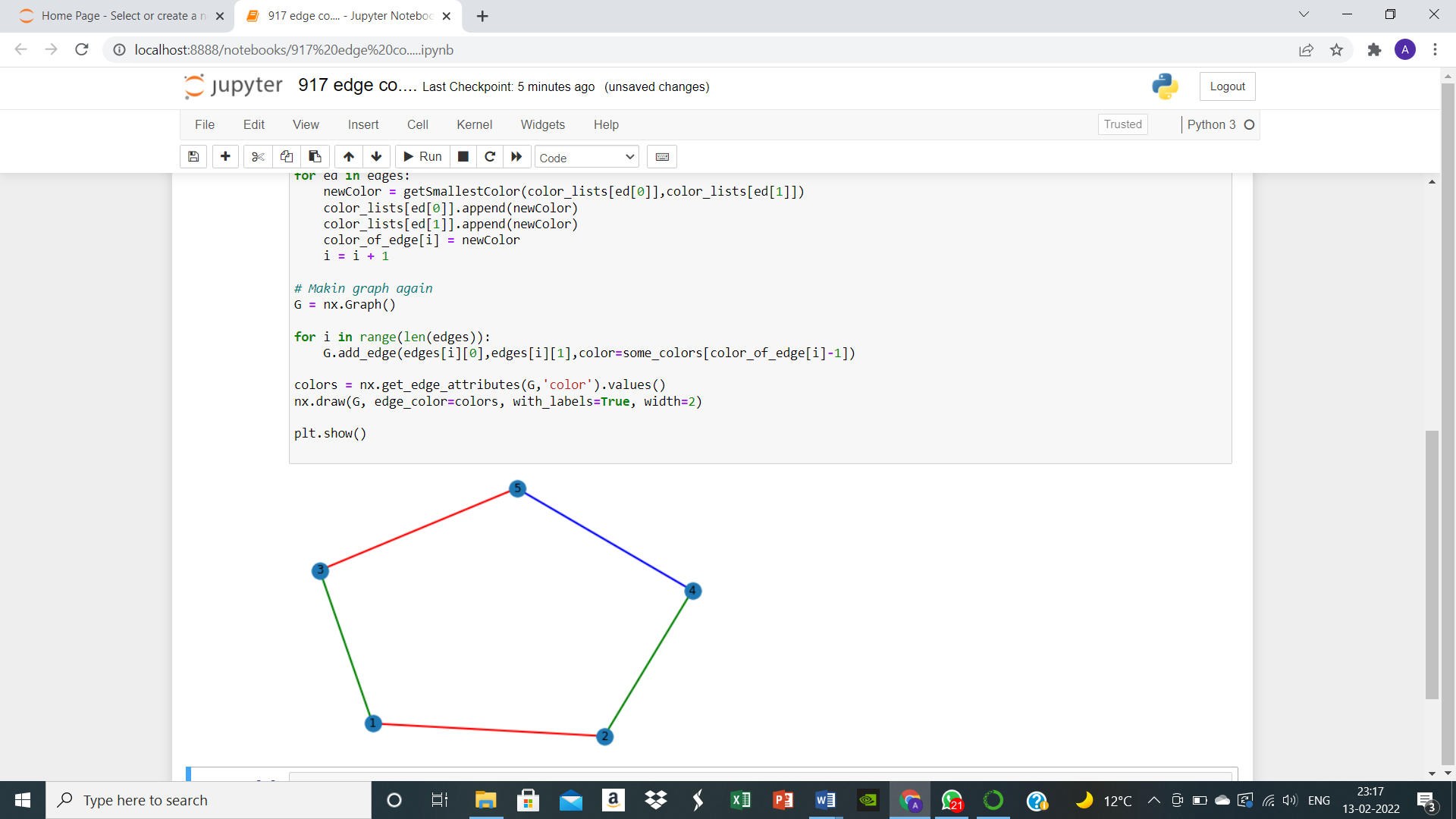
for i in range(len(edges)):

G.add\_edge(edges[i][0],edges[i][1],color=some\_colors[color\_of\_edge[i]-1])

colors = nx.get\_edge\_attributes(G,'color').values()

nx.draw(G, edge\_color=colors, with\_labels=True, width=2)

plt.show()



# Face Colouring:

import networkx as nx

G = nx.Graph()

colors = {0:"red", 1:"green", 2:"blue", 3:"yellow"}

G.add\_nodes\_from([1,2,3,4,5])

G.add\_edges\_from([(1,2), (1,3), (2,4), (3,4), (4,5) ])

nodes = list(G.nodes)

edges = list(G.edges)

some\_colors = ['red','green','blue','yellow']

no\_of\_faces = len(edges)+2-len(nodes)-1

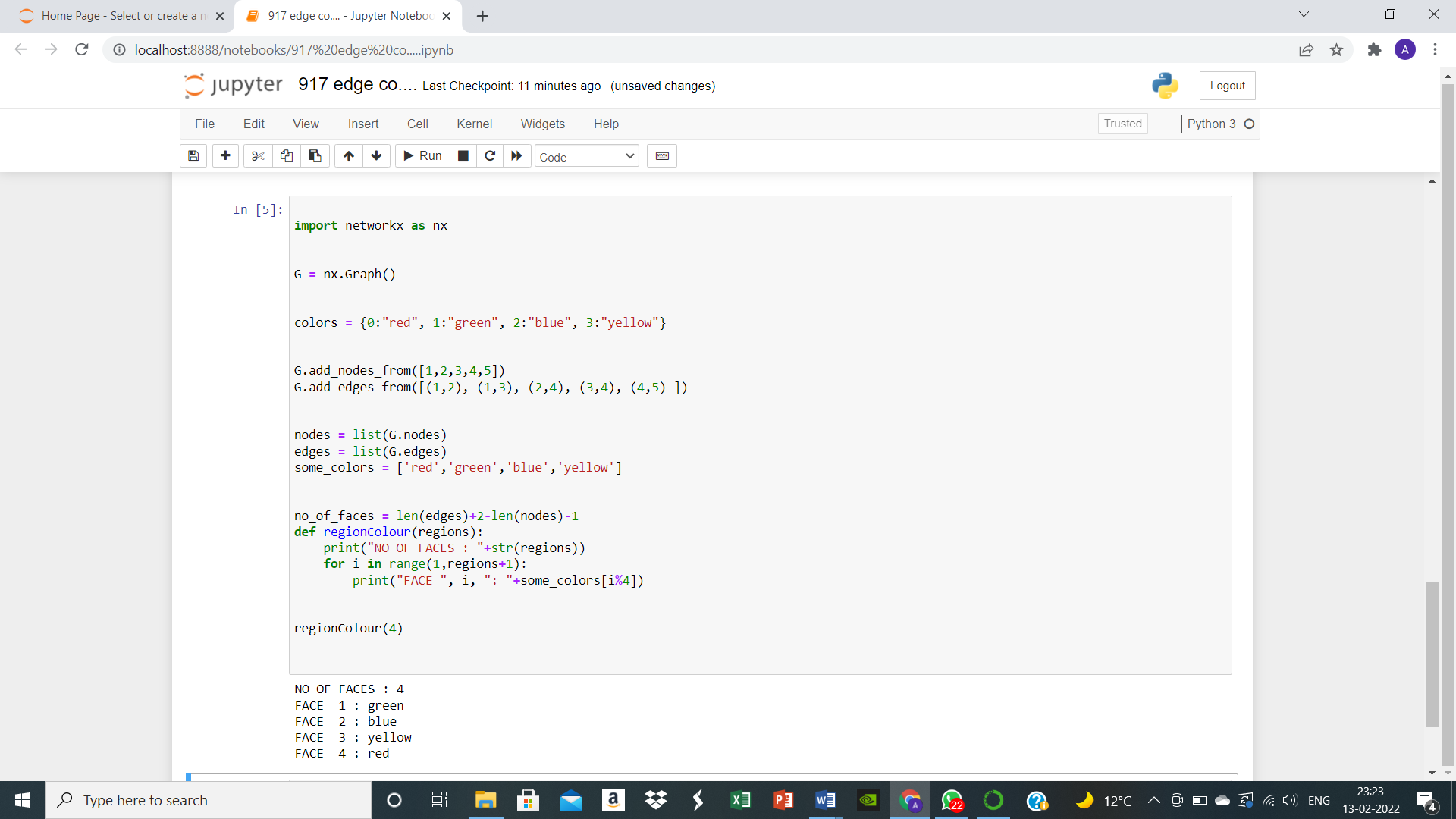
def regionColour(regions):

print("NO OF FACES : "+str(regions))

for i in range(1,regions+1):

print("FACE ", i, ": "+some\_colors[i%4])

regionColour(4)



# Vertex Colouring:

# import matplotlib.pyplot as plt

# import networkx as nx

# G = nx.Graph()

# colors = {0:"red", 1:"green", 2:"blue"}

# G.add\_nodes\_from([1,2,3,4,5])

# G.add\_edges\_from([(1,2), (1,3), (2,4), (3,5), (4,5)])

# d = nx.coloring.greedy\_color(G, strategy = "largest\_first")

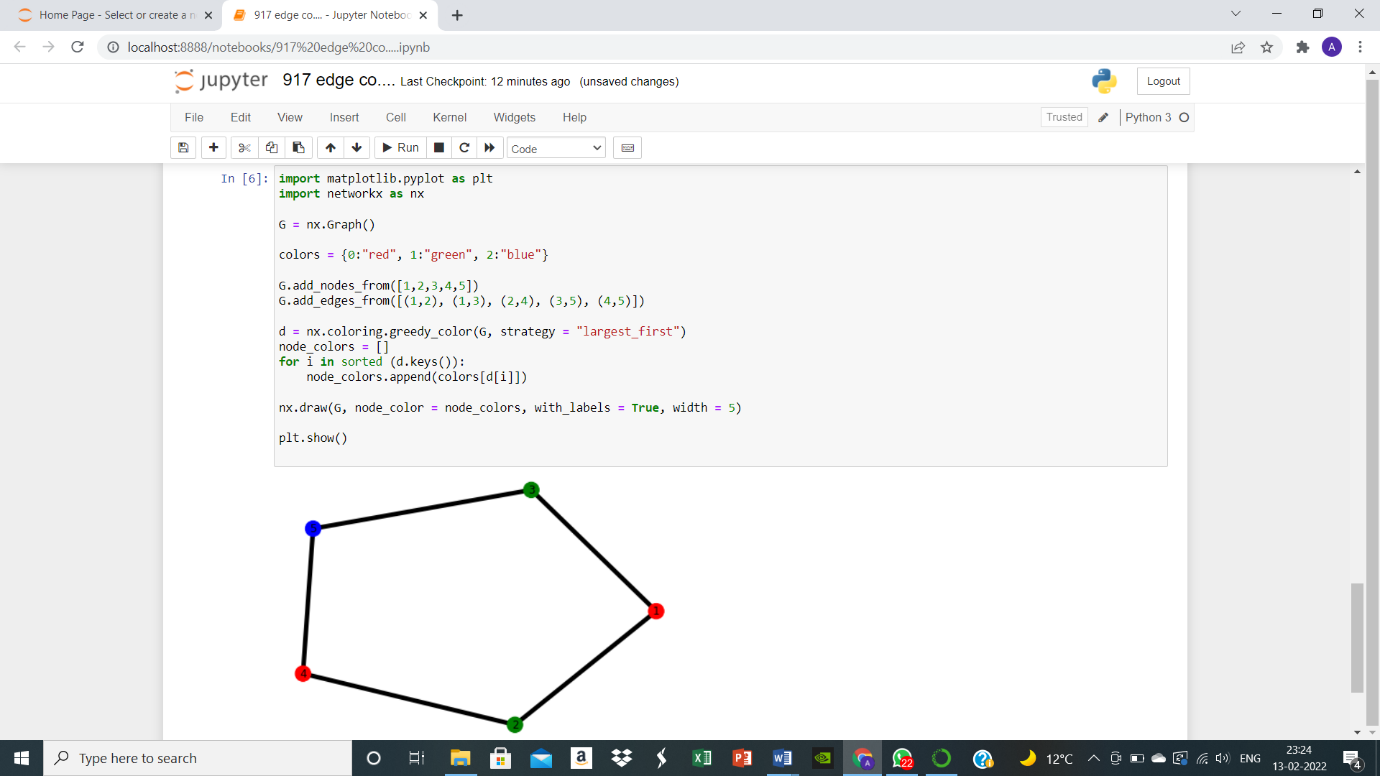
# node\_colors = []

# for i in sorted (d.keys()):

# node\_colors.append(colors[d[i]])

# nx.draw(G, node\_color = node\_colors, with\_labels = True, width = 5)

# plt.show()



**Result:** Graph colouring problem (agent program for real world problem) was discussed and implemented using Python.