Appendix

= Source code listing of colors.py =

```
import random
border = { }  # dictionary containing { region0 : [neighbor1, neighbor2, ...]
              #                         region1 : [neighbor1, neighbor2, ...]
              #                       }
state = { }   # dictionary containing { region0 : 0,  region1 : 1 , ...}
max = 0       # number of regions in the map
co = [ ]      # array of random color assignment for each region in state list

colors = ("red","green","blue","yellow")  # fixed names of 4 colors
tries = 1000           # limited number of tries in each search
mapname = "CANADA"     # name of file containing borders data

def randomColors(choices):    # choose a random color for each region
    for n in range(0,max):
        c = random.randint(0,choices-1)  # random number
        co[n] = colors[c]                # store random color in colors list

def listColors():             # print list of regions and colors
    for s in state:
        print s , " = " , co[state[s]]

def checkColors():  # check all pairs of regions and neighbors for same color
    okay = True                     # result (success so far)
    for x in border:                # each region in border dictionary
        for c in border[x]:         #   each in neighbors list
            a = state[x]            # get color of x
            b = state[c]            # get color of c
            if co[a] == co[b]:      # if colors match, then success = false
                okay = False
    return okay                     # return success or failure

def readborders():    # read data from text-file and store in border dictionary
    count = 0                       # counting regions
    infile = open(mapname,'rU')
    print "=== Map = ",mapname," ==="
    while(1):                       # loop through entire file
        indata = infile.readline()  # read data:  region,neighbor,neighbor...
        num = len(indata)
        if indata[num-1:num]=='\n':  # need to remove
            indata = indata[0:num-1]
        if len(indata)<1:           # stop at end of file
            break
        items = indata.split(",")   # change String to array
        num = len(items)
        border[items[0]] = items[1:num]         # add data to border dictionary
        print items[0] , " : " , border[items[0]]    # region : [neighbors]
        state[items[0]] = count                      # index of this region
        co.append("")               # makes co an array of matching lenght
        count = count + 1
    infile.close()
    print "==================="
    return count                    # number of regions
```

```
def go(choices):     # main loop - tries a set of random colors, then another
                     # repeating until success or quitting after 1000 tries
    global border, state, co, max, tries
    border = { }
    state = { }
    co = [ ]
    max = readborders()        # load data from file

    done=False                 # not yet successful
    for t in range(0,tries):   # try 1000 times
        if done == False:              # if still unsuccssful then
            randomColors(choices)      # choose colors
            if checkColors() == True:  # check whether successful
                listColors()           # if successful then print colors
                done = True
                return done            # quit early if successful

    if done==False:
        print "did not find a solution in ", tries, " tries"

    return done                        # return success or failure

def start():           # main logic - try 4 colors, then 3 if
                       #              it was successful
    global mapname

    mapname = raw_input("Map name (e.g. CANADA)?")
    result = False
    while(result == False):
        result = go(4)
        if result == True:
            result = False
            while(result == False):
                print "Trying with 3 colors"
                result = go(3)
                if result==False:
                    answer = raw_input("press Q to quit,[enter] to retry 3 colors")
                    if answer=="Q" or answer=="q":
                        result = True
        else:
            answer = raw_input("press Q to quit,[enter] to try 4 colors again")
            if answer == "Q" or answer == "q":
                return

start()

raw_input("== Finished ==")
```

- Sample Output -

The program ran successfully on many sets of sample data - here are a few:

FLAG a very simple map	GRID looks like this	CANADA map as shown above

```
IDLE 2.6.5
>>> ========================
>>>
Map name (e.g. CANADA)?flag
=== Map =  flag  ===
a  :  ['b', 'c', 'd']
b  :  ['a', 'c']
c  :  ['a', 'b', 'd']
d  :  ['a', 'c']
==================
a  =  yellow
c  =  green
b  =  blue
d  =  blue
Trying with 3 colors
=== Map =  flag  ===
a  :  ['b', 'c', 'd']
b  :  ['a', 'c']
c  :  ['a', 'b', 'd']
d  :  ['a', 'c']
==================
a  =  red
c  =  green
b  =  blue
d  =  blue
== Finished ==
```

Found 4-color solution, then automatically continued to find a 3-color solution.

```
IDLE 2.6.5
>>> ========================
>>>
Map name (e.g. CANADA)?grid
=== Map =  grid  ===
a  :  ['b', 'd']
b  :  ['a', 'c', 'e']
c  :  ['b', 'f']
d  :  ['a', 'e', 'g']
e  :  ['b', 'd', 'f', 'h']
f  :  ['c', 'e', 'i']
g  :  ['d', 'h']
h  :  ['e', 'g', 'i']
i  :  ['f', 'h']
==================
a  =  yellow
c  =  green
b  =  red
e  =  yellow
d  =  green
g  =  yellow
f  =  red
i  =  blue
h  =  red
Trying with 3 colors
=== Map =  grid  ===
a  :  ['b', 'd']
b  :  ['a', 'c', 'e']
c  :  ['b', 'f']
d  :  ['a', 'e', 'g']
e  :  ['b', 'd', 'f', 'h']
f  :  ['c', 'e', 'i']
g  :  ['d', 'h']
h  :  ['e', 'g', 'i']
i  :  ['f', 'h']
==================
a  =  blue
c  =  blue
b  =  green
e  =  blue
d  =  red
g  =  blue
f  =  green
i  =  blue
h  =  green
== Finished ==
```

Found a 4-color solution and a 3-color solution.

```
IDLE 2.6.5
>>> ================================ REST
>>>
Map name (e.g. CANADA)?canada
=== Map =  canada  ===
YU  :  ['BC', 'NW']
BC  :  ['YU', 'NW', 'AL']
NW  :  ['YU', 'NU', 'BC', 'AL', 'SA']
AL  :  ['BC', 'NW', 'SA']
SA  :  ['AL', 'NW', 'MA']
NU  :  ['NW', 'MA', 'QU']
MA  :  ['SA', 'NU', 'ON']
ON  :  ['MA', 'QU']
QU  :  ['ON', 'NU', 'NF', 'NB']
NF  :  ['QU']
NB  :  ['QU', 'NS', 'PE']
NS  :  ['NB', 'PE']
PE  :  ['NB', 'NS']
==================
ON  =  yellow
MA  =  green
BC  =  blue
NB  =  red
AL  =  green
NF  =  yellow
PE  =  green
QU  =  green
SA  =  yellow
NS  =  yellow
YU  =  yellow
NU  =  blue
NW  =  red
Trying with 3 colors
=== Map =  canada  ===
YU  :  ['BC', 'NW']
BC  :  ['YU', 'NW', 'AL']
NW  :  ['YU', 'NU', 'BC', 'AL', 'SA']
AL  :  ['BC', 'NW', 'SA']
SA  :  ['AL', 'NW', 'MA']
NU  :  ['NW', 'MA', 'QU']
MA  :  ['SA', 'NU', 'ON']
ON  :  ['MA', 'QU']
QU  :  ['ON', 'NU', 'NF', 'NB']
NF  :  ['QU']
NB  :  ['QU', 'NS', 'PE']
NS  :  ['NB', 'PE']
PE  :  ['NB', 'NS']
==================
did not find a solution in  1000  tries
press Q to quit,[enter] to retry 3 colors
```

Found a 4-color solution, but attempt at a 3-color solution failed after 1000 tries.

- Further Tests -

Succeeding with 3-color scheme for Canada - after several tries	USFLAG (no stars) 

``
==========
did not find a solution in 1000 tries
press Q to quit,[enter] to retry 3 colors
Trying with 3 colors
=== Map = canada ===
YU : ['BC', 'NW']
BC : ['YU', 'NW', 'AL']
NW : ['YU', 'NU', 'BC', 'AL', 'SA']
AL : ['BC', 'NW', 'SA']
SA : ['AL', 'NW', 'MA']
NU : ['NW', 'MA', 'QU']
MA : ['SA', 'NU', 'ON']
ON : ['MA', 'QU']
QU : ['ON', 'NU', 'NF', 'NB']
NF : ['QU']
NB : ['QU', 'NS', 'PE']
NS : ['NB', 'PE']
PE : ['NB', 'NS']
==========
did not find a solution in 1000 tries
press Q to quit,[enter] to retry 3 colors
Trying with 3 colors
=== Map = canada ===
YU : ['BC', 'NW']
BC : ['YU', 'NW', 'AL']
NW : ['YU', 'NU', 'BC', 'AL', 'SA']
AL : ['BC', 'NW', 'SA']
SA : ['AL', 'NW', 'MA']
NU : ['NW', 'MA', 'QU']
MA : ['SA', 'NU', 'ON']
ON : ['MA', 'QU']
QU : ['ON', 'NU', 'NF', 'NB']
NF : ['QU']
NB : ['QU', 'NS', 'PE']
NS : ['NB', 'PE']
PE : ['NB', 'NS']
==========
ON = blue
MA = green
BC = red
NB = red
AL = blue
NF = blue
PE = green
QU = green
SA = red
NS = blue
YU = blue
NU = blue
NW = green
== Finished ==
``
	``
	IDLE 2.6.5
	>>> ========== RESTART ==========
	>>>
	Map name (e.g. CANADA)?usflag
	=== Map = usflag ===
	r1 : ['w1', 'bb']
	w1 : ['r1', 'r2', 'bb']
	r2 : ['w1', 'w2', 'bb']
	w2 : ['r2', 'r3', 'bb']
	r3 : ['w2', 'w3', 'bb']
	w3 : ['r3', 'r4', 'bb']
	r4 : ['w3', 'w4', 'bb']
	w4 : ['r4', 'r5', 'bb']
	r5 : ['w4', 'w5']
	w5 : ['r5', 'r6']
	r6 : ['w5', 'w6']
	w6 : ['r6', 'r7']
	r7 : ['w6']
	bb : ['r1', 'w1', 'r2', 'w2', 'r3', 'w3', 'r4', 'w4']
	==========
	did not find a solution in 1000 tries
	press Q to quit,[enter] to try 4 colors again
	=== Map = usflag ===
	r1 : ['w1', 'bb']
	w1 : ['r1', 'r2', 'bb']
	r2 : ['w1', 'w2', 'bb']
	w2 : ['r2', 'r3', 'bb']
	r3 : ['w2', 'w3', 'bb']
	w3 : ['r3', 'r4', 'bb']
	r4 : ['w3', 'w4', 'bb']
	w4 : ['r4', 'r5', 'bb']
	r5 : ['w4', 'w5']
	w5 : ['r5', 'r6']
	r6 : ['w5', 'w6']
	w6 : ['r6', 'r7']
	r7 : ['w6']
	bb : ['r1', 'w1', 'r2', 'w2', 'r3', 'w3', 'r4', 'w4']
	==========
	r4 = blue
	r5 = yellow
	r6 = green
	r7 = blue
	r1 = blue
	r2 = blue
	r3 = yellow
	bb = red
	w6 = red
	w5 = red
	w4 = green
	w3 = green
	w2 = green
	w1 = yellow
	``
	
	Needed 2x1000 tries to find a 4-color solution. Also found a 3-color solution, but required 20x1000 tries (not shown)

Further sample runs are shown in the accompanying videos.