```
# -*- coding: utf-8 -*-
"""
Created on Fri Oct 21 09:25:09 2016

@author: James Raines

Purpose: Vegetation Recovery after Wildfire
"""

from time import clock
t0 = clock()
import gdal, gdalnumeric
import numpy as np
import os
from lab5functions import *

path = r'pathHere'
bpath = path + 'L5_big_elk/'

#set working directory
os.chdir(path)

#calcualtes NDVI using landsat bands 3 and 4, red and infrared
def cal_ndvi(band3, band4):
    ndvi = (band4 - band3) / (band4 + band3)
    return ndvi

def zonalStatistics (zone,values):
    '''
    calculates the mean, standard deviation, min and max for the values in the zone

    :param zone: a numpy array that defines the mask for the statisical calulcations of the values
    :param values: a numpy array in which you want to calcualte zonal statistics for
    :return: a numpy array with the zone values in the first column followed by the four statistics
    '''
    zonalStats = []
    zone_values = np.unique(zone)

    for value in zone_values:
        valbool = np.where(zone == value)
        subset = values[valbool]
        mean = np.mean(subset)
        std = np.std(subset)
        mi = np.min(subset)
        ma = np.max(subset)
        zonalStats.append([value, mean, std, mi, ma])

    zonalStats = np.asarray(zonalStats)
    for line in zonalStats:
        print line
    return zonalStats

#creates variables and parses the bands for the ndvi
data = os.listdir(path)
bdata = os.listdir(bpath)
tifs = [x for x in data if x.endswith('.tif')]
bands = [x for x in bdata if x.endswith('.tif')]
```

```
bandPairs = [bands[x:x + 2] for x in range(0, len(bands), 2)]

#Part 1
#reads dem as numpy array and gets the raster dimensions
demObj = gdal.Open(tifs[0])
demGeo = demObj.GetGeoTransform()
dem = demObj.GetRasterBand(1).ReadAsArray()

#calculates slope and aspect for dem
slp,asp = slopeAspect(dem, demGeo[1])
print 'Slope and aspect have been calculated for the DEM'

#reclassifies the aspect into 8 cardinal directions (N,NE,E,SE,S,SW,W,NW)
aspRe = reclassAspect(asp)
print 'Aspect has been reclassified into 8 cardinal directions (N,NE,E,SE,S,SW,W,NW)'

#reclassifies slope into 10 classes
slpRe = reclassByHisto(slp, 10)
print 'Slope has been reclassified into 10 bins by histogram'

#calculates NDVI for all years and stores in dictionary ndvis
ndvis = {}
for band in bandPairs:
    b3 = gdalnumeric.LoadFile(bpath + band[0])
    b4 = gdalnumeric.LoadFile(bpath + band[1])
    ndvis[band[0][9:13]] = cal_ndvi(b3, b4)
print 'NDVI has been calculated for years',sorted((ndvis.keys())),'and stored with those keys in th

#creates an array for the fire area raster
fire = gdalnumeric.LoadFile(tifs[1])
#seperates the healthy forest and burned area into its their own arrays
healthyForest = fire == 2
burnedArea = fire == 1

#calcualtes the Recovery Ratio for all years and stores in dictionary rr and list rrlist
rr = {}
rrlist = []
for k, v in np.sort(ndvis.items()):
    healthy_subset = v[healthyForest]
    burned_array = np.where(burnedArea == True, v, np.nan)
    mean = np.mean(healthy_subset)
    ratio = burned_array/float(mean)
    rrlist.append(ratio)
    rr[k] = ratio

print 'The Recovery Ratio has been calculated for years',sorted((rr.keys())),'and stored with those

#stacks all the recovery ratio arrays into a 280, 459, 10 array for the polyfit
dStack = np.dstack(rrlist)
#replaces all np.nan with 0 to prevent errors in the polyfit
rrStack = np.where(np.isfinite(dStack), dStack, 0)

#calculates the trend in Recovery Ratio with a first-degree polynomial
coefAr = np.zeros(ndvis['2002'].shape) #zeros array to store the polyfit outputs
for row in range(0, rrStack.shape[0]):
    for col in range(0, rrStack.shape[1]):
            fi = rrStack[row,col,:]
```

```
            coef, inter = np.polyfit(range(0, rrStack.shape[2]),fi,1)
            coefAr[row,col] = coef

for k, v in sorted(rr.items()):
    print 'The mean recovery ratio for',k,'is:',np.nanmean(v)
print 'The mean coefficient of recovery across all years is:',np.mean(coefAr[burnedArea])

#Part 2
print 'Slope zonal statistics:'
coefArSubset = coefAr[burnedArea]
slpZS = zonalStatistics(slpRe,coefArSubset)
print 'Aspect zonal statistics:'
aspZS = zonalStatistics(aspRe,coefArSubset)

#creates the nodata values as -99 for the output
coef_output = np.where(burnedArea == True,coefAr,-99)

#gets and registers the driver
driver = demObj.GetDriver()
driver.Register()

#creates the Coefficient_of_recovery.tif file
outFile = driver.Create('Coefficient_of_recovery.tif',dem.shape[1],dem.shape[0],1,gdal.GDT_Float32)
outFile.GetRasterBand(1).WriteArray(coef_output)
outFile.GetRasterBand(1).ComputeStatistics(False)
outFile.SetGeoTransform(demGeo)
outFile.SetProjection(demObj.GetProjection())
outFile.FlushCache()
print 'Coefficient_of_recovery.tif has been created'
print 'Elapsed time: ',round(clock()-t0,4),' seconds'
```