```
/usr/bin/env python
# encoding: utf-8
"""
FieldEllipticals.py

Created by Sami-Matias Niemi on 2008-07-03.
Copyright (c) 2008 __Sami-Matias Niemi__. All rights reserved.
"""

#Optimizations:
# - maps possible field ellipticals
# - maps possible companions for these field ellipticals
# - tests if these mapped galaxies actually for a field elliptical system..

def cubicRealRoot(a, b, c, d):
    """
    Calculates the real roots of cubic equation. Note: returns only the _real_ roots!
    Code as found at http://www.josechu.com/ecuaciones_polinomicas/
    """
    twoonethird = 2.0**(1.0 / 3.0)
    delta = (-2.0 * b * b * b + 9.0 * a * b * c - 27.0 * a * a * d + ((4.0 * (-b * b + 3.0 * a * c)**3.0)**(1.0/2.0)) + ((-2.0 * b * b * b + 9.0 * a * b * c - 27.0 * a * a * d)**2.0))**(1.0 / 3.0)
    result = (-b / (3.0 * a) - (twoonethird * (-b * b + 3.0 * a * c)) / (3.0 * a * delta) + delta / (3.0 * twoonethird * a))
    return result

def MillenniumSimulationColumns():
    """
    A dictionary that contains columns of the Millennium Simulation Galaxy catalogue data.
    """
    MSdata = {
        'galaxyID' : 0, 'lastProg' : 1, 'descedantID' : 2, 'haloID' : 3, 'subHaloID' : 4,
        'fofID' : 5, 'treeId' : 6, 'firstProg' : 7, 'nextProg' : 8, 'typee' : 9,
        'snapnum' : 10, 'redshift' : 11, 'centralMvir' : 12, 'phkey' : 13, 'x' : 14, 'y' : 15,
        'z' : 16, 'zIndex' : 17, 'ix' : 18, 'iy' : 19, 'iz' : 20, 'velX' : 21, 'velY' : 22,
        'velZ' : 23, 'np' : 24, 'mvir' : 25, 'rvir' : 26, 'vvir' : 27, 'vmax' : 28, 'coldGas' : 29,
        'stellarMass' : 30, 'bulgeMass' : 31, 'hotGas' : 32, 'ejectedMass' : 33, 'blackholeMass' : 34,
        'metalsCG' : 35, 'metalsSM' : 36, 'metalsBM' : 37, 'metalsHG' : 38, 'metalsEM' : 39,
        'sfr' : 40, 'sfrBulge' : 41, 'xrayLum' : 42, 'diskRadius' : 43, 'coolingR' : 44,
        'mag_bc' : 45, 'mag_vc' : 46, 'mag_rc' : 47, 'mag_ic' : 48, 'mag_kc' : 49, 'mag_bB' : 50,
        'mag_vB' : 51, 'mag_rB' : 52, 'mag_iB' : 53, 'mag_kB' : 54, 'mag_bD' : 55, 'mag_vD' : 56,
        'mag_rD' : 57, 'mag_iD' : 58, 'mag_kD' : 59, 'massWAge' : 60, 'random' : 62}
    return MSdata

def MillenniumSimulationFormat():
    """
    Formation file for Millennium Simulation Galaxy catalogue data.
    """
    format = {'names': ('galaxyID','lastProg','descedantID','haloID','subHaloID',
                     'fofID','treeId','firstProg','nextProg','typee',
                     'snapnum','redshift','centralMvir','phkey','x','y',
                     'z','zIndex','ix','iy','iz','velX','velY',
                     'velZ','np','mvir','rvir','vvir','vmax','coldGas',
                     'stellarMass','bulgeMass','hotGas','ejectedMass','blackholeMass',
                     'metalsCG','metalsSM','metalsBM','metalsHG','metalsEM',
                     'sfr','sfrBulge','xrayLum','diskRadius','coolingR',
                     'mag_bc','mag_vc','mag_rc','mag_ic','mag_kc','mag_bB',
                     'mag_vB','mag_rB','mag_iB','mag_kB','mag_bD','mag_vD',
                     'mag_rD','mag_iD','mag_kD','massWAge','random'),
                 'formats': ('q','q','q','q','q','q','q','q','q',
                           'l','l','f','f','l','f','f','f','q',
                           'l','l','l','f','f','f','l','f','f',
                           'f','f','f','f','f','f','f','f','f',
                           'f','f','f','f','f','f','f','f','f',
                           'f','f','f','f','f','f','f','f','f',
                           'f','f','f','f','f','f','f','l')}
    return format

def tofile(fname, X, fmt='%4.1e', delimiter=' ', header = '#Header'):
    from numpy import savetxt
    #from pprint import pprint

    fmtToType = { 'd': int,
                  'f': float,
                  'e': float }
    if type(fmt) == tuple:
        fh = open(fname, 'w')
        fh.write(header + '\n')
        for row in X:
            #pprint(zip(fmt, row))
            fh.write(delimiter.join(
                             [(ft % fmtToType[ft[-1]](value)) for ft, value in zip(fmt, row)]
                             ) + '\n')
        fh.close()
    else:
        savetxt(fname, X, fmt, delimiter)

def basicStats(data):
    """Calculates basic statistics from a given array
    """
    if (len(data) > 1):
        import numpy as N
        median = N.median(data)
        mean = N.mean(data)
        std = N.std(data)
        max = N.max(data)
        min = N.min(data)
        var = N.var(data)
        return mean, median, std, var, max, min
    else:
        return (-99,)*6

def main():
    """Main program.
    """
    import sys
    import os.path
    import ConfigParser
    from getopt import getopt
    import numpy
    import time
    import numpy.linalg

    starttime = time.time()
    verbose = False

    log = file("log.out", 'w')
    progress = file("progress.out", 'w')

    (opts, args) = getopt(sys.argv[1:], 'v')
    for opt, val in opts:
        if opt == '-v':   verbose = True

    welcome = "\nThis program searches Millennium Simulation Galaxy data for field ellipticals!\n"
    start = "The run was started at %s \n" % time.asctime(time.localtime())
    if verbose:
        print welcome
        print start

    log.write('This file contains the log of the FieldElliptical.py -program!')
    log.write(welcome)
    log.write(start)

    if len(args) < 1:
        print "Wrong number of commandline arguments! Give me the name of the parameter file!"
        sys.exit(-1)

    path, fname = os.path.split(args[0])

    #this is for parsing the config file
    config = ConfigParser.ConfigParser()
    config.read(fname)

    #parsing the parameters
    filename = config.get('default_run','filename')
    deltamag1 = config.getfloat('default_run','deltamag1')
    deltamag2 = config.getfloat('default_run','deltamag2')
    distance1 = config.getfloat('default_run','distance1')
    distance2 = config.getfloat('default_run','distance2')
    ellimit = config.getfloat('default_run','Ellimit')
    maglimit = config.getfloat('default_run','maglimit')
    xlow = config.getfloat('default_run','xlow')
    xup = config.getfloat('default_run','xup')
    ylow = config.getfloat('default_run','ylow')
    yup = config.getfloat('default_run','yup')
    zlow = config.getfloat('default_run','zlow')
    zup = config.getfloat('default_run','zup')
    safedist = config.getfloat('default_run','safedistance')

    log.write("You selected to use a following cube:\n")
    log.write("%6.2f <= x <= %6.2f\n" % (xlow+safedist, xup-safedist))
    log.write("%6.2f <= y <= %6.2f\n" % (ylow+safedist, yup-safedist))
    log.write("%6.2f <= z <= %6.2f\n" % (zlow+safedist, zup-safedist))

    #defines MS data column constants as a dictionary
    MS = MillenniumSimulationColumns()

    #reads the whole file
    #datafloat = numpy.loadtxt(filename, comments = '#', delimiter=',', skiprows=0)
    #firstof = data[0,MS['x']]
    form = MillenniumSimulationFormat()
    data = numpy.loadtxt(filename, comments = '#', delimiter=' ', skiprows=0, dtype=form)
    #firstof = data[0][MS['x']]

    galaxies = len(data)

    found = "\nFound %i galaxies from your data file!\n" % galaxies
    if verbose:
        print found
    log.write(found)
    log.flush()

    #results variables
    fieldEs = 0; companions = []; results  = []; FEllipticals = []; Ellipticals = []

    for line1, galaxy in enumerate(data):
        if (line1 % 500 == 0):
            progress.write("%10.6f per cent done...\n" % (float(line1)/float(galaxies)*100.))
            progress.flush()
        #print line1
        #resets temp variables
        fieldElliptical  = False
        comp1 = []; comp2 = [];
        #Tests if in safe area
        if (((galaxy[MS['x']] - xlow) >= safedist) and
            ((galaxy[MS['y']] - ylow) >= safedist) and
            ((galaxy[MS['z']] - zlow) >= safedist) and
            ((xup - galaxy[MS['x']]) >= safedist) and
            ((yup - galaxy[MS['y']]) >= safedist) and
            ((zup - galaxy[MS['z']]) >= safedist)):
                #Just to be sure the bulge magnitude is not 99
                if (galaxy[MS['mag_bB']] < 30):
                    bulgemagdiff = galaxy[MS['mag_bB']] - galaxy[MS['mag_bc']]
                    T = cubicRealRoot(0.0047, -0.054, 0.342, - bulgemagdiff) - 5.0
                    galaxy = numpy.void.tolist(galaxy)
                    galaxy = galaxy + (T,)
                    if ((T <= ellimit) and (galaxy[MS['mag_bc']] <= maglimit)):
                        fieldElliptical  = True
                        for line2, companion in enumerate(data):
                            if (line1 != line2 and fieldElliptical):

                                #calculates the distance between the objects
                                coordsGal = numpy.array( (galaxy[MS['x']],
    galaxy[MS['y']], galaxy[MS['z']]) )
coordsCompanion = numpy.array( (companion[MS['x']], companion[MS['y']], companion[MS['z']]) )
distance = numpy.linalg.norm( coordsGal - coordsCompanion )

#calculates the magnitude difference
magdif = abs(galaxy[MS['mag_bc']] - companion[MS['mag_bc']])

#tests if companion fulfils field elliptical criteria
#breaks the loop if not
if (distance <= distance1):
    if (magdif <= deltamag1):
        fieldElliptical = False
        break
    if (galaxy[MS['mag_bc']] >= companion[MS['mag_bc']]):
        fieldElliptical = False
        break
if (distance <= distance2):
    if(magdif <= deltamag2):
        fieldElliptical = False
        break
    if (galaxy[MS['mag_bc']] >= companion[MS['mag_bc']]):
        fieldElliptical = False
        break

#saves the line number of companions and their morphology
if (distance <= distance1 and magdif > deltamag1):
    if (distance <= distance2 and magdif > deltamag2):
        T2 = 99
        if (companion[MS['mag_bB']] <= 0 and companion[MS['mag_bc']] <=0):
            bulgemagdiff2 = companion[MS['mag_bB']] - companion[MS['mag_bc']]
            T2 = cubicRealRoot(0.0047, -0.054, 0.342, -bulgemagdiff2) - 5.0
        else : T2 = 9
        comp = numpy.void.tolist(companion)
        comp += (T2,)
        comp2.append(comp)
    else:
        T1 = 99
        if (companion[MS['mag_bB']] <= 0 and companion[MS['mag_bc']] <=0):
            bulgemagdiff1 = companion[MS['mag_bB']] - companion[MS['mag_bc']]
            T1 = cubicRealRoot(0.0047, -0.054, 0.342, -bulgemagdiff1) - 5.0
```

```
                                        else: T1 = 9
                                        comp = numpy.void.tolist(companion)
                                        comp += (T1,)
                                        comp1.append(comp)

                        #saves non field ellipticals
                    if (fieldElliptical == False and T <= 0):
                        Ellipticals.append(galaxy)

        #saves output data
                if (fieldElliptical):
                    fieldEs +=1
                    galaxy += (len(comp1), len(comp2))
                    resultsgal = (0,) + galaxy

                    results.append(resultsgal)
                    FEllipticals.append(galaxy)

                    for line in comp1:
                    #for line, T in comp1:
                        #results.append(data[line])
                        res = (1,) + line + (len(comp1), len(comp2))
                        results.append(res)
                        companions.append(line)
                    #print results

                    for line in comp2:
                    #for line, T in comp2:
                        #results.append(data[line])
                        res = (2,) + line + (len(comp1), len(comp2))
                        results.append(res)
                        companions.append(line)

    progress.close()

    #print results
    #Formats the output
    outformFE = ('%d','%d','%d','%d','%d','%d','%d','%d','%d','%d','%d','%f','%f',
                '%d','%f','%f','%f','%d','%d','%d','%d','%f','%f','%f','%d','%f',
                '%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f',
                '%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f',
                '%f','%f','%f','%f','%f','%f','%f','%f','%f','%d','%f','%d','%d')

    outformre = ('%d', '%d','%d','%d','%d','%d','%d','%d','%d','%d','%d','%d','%f','%f',
                '%d','%f','%f','%f','%d','%d','%d','%d','%f','%f','%f','%d','%f',
                '%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f',
                '%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f',
                '%f','%f','%f','%f','%f','%f','%f','%f','%f','%d','%f','%d','%d')

    outformco = ('%d','%d','%d','%d','%d','%d','%d','%d','%d','%d','%d','%f'
        ,'%f',
                      '%d','%f','%f','%f','%d','%d','%d','%d','%f','%f','%f','%d','%f',
                      '%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f',
                      '%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f',
                      '%f','%f','%f','%f','%f','%f','%f','%f','%f','%d', '%f')

    outformel = ('%d','%d','%d','%d','%d','%d','%d','%d','%d','%d','%d','%f','%f',
                      '%d','%f','%f','%f','%d','%d','%d','%d','%f','%f','%f','%d','%f',
                      '%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f',
                      '%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f','%f',
                      '%f','%f','%f','%f','%f','%f','%f','%f','%f','%d', '%f')

    #prints to files
    fehed = '#MS columns + T #comp1 #comp2'
    fched = '#ID + MS columns + T #comp1 #comp2'
    cohed = '#MS columns + T'
    elhed = '#MS columns + T'
    tofile('FieldEllipticals.out', FEllipticals, fmt=outformFE, delimiter=' ', header=fehed)
    tofile('FieldEsandCompanions.out', results, fmt=outformre, delimiter =' ', header=fched)
    tofile('Companions.out', companions, fmt=outformco, delimiter =' ', header=cohed)
    tofile('Ellipticals.out', Ellipticals, fmt=outformel, delimiter =' ', header=elhed)

    #reads files again to different type of array
    FieldE = numpy.loadtxt('FieldEllipticals.out', comments = '#', delimiter =' ', skiprows=0)
    Comps = numpy.loadtxt('Companions.out', comments = '#', delimiter=' ', skiprows=0)
    Ell = numpy.loadtxt('Ellipticals.out', comments = '#', delimiter=' ', skiprows=0)

    #calculates some statistics
    Compsmass = basicStats(Comps[:,MS['mvir']])
    CompsMagb = basicStats(Comps[:,MS['mag_bc']])
    CompsColdGas = basicStats(Comps[:,MS['coldGas']])
    CompsStellarMass = basicStats(Comps[:,MS['stellarMass']])
    CompsBHMass = basicStats(Comps[:,MS['blackholeMass']])
    CompsBulgeMass = basicStats(Comps[:,MS['bulgeMass']])
    Ellmass = basicStats(Ell[:,MS['mvir']])
    EllMagb = basicStats(Ell[:,MS['mag_bc']])
    EllColdGas = basicStats(Ell[:,MS['coldGas']])
    EllStellarMass = basicStats(Ell[:,MS['stellarMass']])
    EllBHMass = basicStats(Ell[:,MS['blackholeMass']])
    EllBulgeMass = basicStats(Ell[:,MS['bulgeMass']])
    FieldEmass = basicStats(FieldE[:,MS['mvir']])
    FieldEMagb = basicStats(FieldE[:,MS['mag_bc']])
    FieldEColdGas = basicStats(FieldE[:,MS['coldGas']])
    FieldEStellarMass = basicStats(FieldE[:,MS['stellarMass']])
    FieldEBHMass = basicStats(FieldE[:,MS['blackholeMass']])
    FieldEBulgeMass = basicStats(FieldE[:,MS['bulgeMass']])

    #writes statistics to a file
    fmtt = "%16s"*7 +"\n"
    fmts = "%16s" + "%16.5f"*6 + "\n"
    statfile = open('Stats.out','w')
    statfile.write("#This file contains some statistics.\n")
    statfile.write("#For field ellipticals:\n")
    statfile.write(fmtt % ("#name", "mean", "median", "std", "var", "max", "min"))
    statfile.write(fmts % (("Mvir",) + FieldEmass))
    statfile.write(fmts % (("Mag_B",) + FieldEMagb))
    statfile.write(fmts % (("ColdGas",) + FieldEColdGas))
    statfile.write(fmts % (("StellarMass",) + FieldEStellarMass))
    statfile.write(fmts % (("BlackHoleMass",) + FieldEBHMass))
    statfile.write(fmts % (("BulgeMass",) + FieldEBulgeMass))
    statfile.write("#For ellipticals (excluding field ellipticals):\n")
    statfile.write(fmtt % ("#name", "mean", "median", "std", "var", "max", "min"))
    statfile.write(fmts % (("Mvir",) + Ellmass))
    statfile.write(fmts % (("Mag_B",) + EllMagb))
    statfile.write(fmts % (("ColdGas",) + EllColdGas))
    statfile.write(fmts % (("StellarMass",) + EllStellarMass))
    statfile.write(fmts % (("BlackHoleMass",) + EllBHMass))
    statfile.write(fmts % (("BulgeMass",) + EllBulgeMass))
    statfile.write("#For companion galaxies:\n")
    statfile.write(fmtt % ("#name", "mean", "median", "std", "var", "max", "min"))
    statfile.write(fmts % (("Mvir",) + Compsmass))
    statfile.write(fmts % (("Mag_B",) + CompsMagb))
    statfile.write(fmts % (("ColdGas",) + CompsColdGas))
    statfile.write(fmts % (("StellarMass",) + CompsStellarMass))
    statfile.write(fmts % (("BlackHoleMass",) + CompsBHMass))
    statfile.write(fmts % (("BulgeMass",) + CompsBulgeMass))
    statfile.close()

    #end of loops
    foundFE= "Found %d Field Ellipticals from your data!\n" % fieldEs
    if verbose:
        print foundFE
    log.write(foundFE)

    stoptime = time.time()
    stopstr = "Running time of the program was %.2f minutes.\n" % ((stoptime-starttime)/60.)
    succ = "The program terminated successfully!\n"
    if verbose:
        print  stopstr
        print succ
    log.write(stopstr)
    log.write(succ)

    log.close()

if __name__ == '__main__':
    main()
```