```
# -*- coding: utf-8 -*-
"""
Calculator.py

This program has different mathematical computations and data plotting to excel
"""
import math
import matplotlib.pyplot as plt
import xlsxwriter
from datetime import datetime
import inspect
from numbers import Number

a = 6378137.0
e2 = (0.08181919104282)**2

LatMarco = -(22 + 49/60.0 + 08.76793/3600)
LonMarco = -(43 + 18/60.0 + 23.95193/3600)
hMarco = -1.461
print LatMarco
print LonMarco

Xonrj = 4283638.3579
Yonrj = -4026028.8217
Zonrj = -2466096.8361

Xriod = 4280294.8786
Yriod = -4034431.2247
Zriod = -2458141.3800

"""
360-(316+41/60.0+37.4/3600)
-43.3062777778

-22+49/60.0+4.2/3600
Out[3]: -21.182166666666667

22+49/60.0+4.2/3600
Out[4]: 22.817833333333333

Dados adquiridos do site: ftp://ftp.sirgas.org/pub/gps/SIRGAS/
para as semanas 2036 e 2040

(-1.2283013544738297, 0.13083121888820887)"""

Xriod2019 = 4280294.89778
Yriod2019 = -4034431.33971
Zriod2019 = -2458141.16186
""""
MATH FUNCTIONS:

deg2xyz
xyz2deg
radianError
"""
def deg2xyz(ddLat,ddLon,h): #VALIDADO
    radLat = math.radians(ddLat)
    radLon = math.radians(ddLon)
    N = ( a / ((1-(e2)*(math.sin(radLat))**2)**(0.5)))
    X = (N+h)*math.cos(radLat)*math.cos(radLon)
    Y = (N+h)*math.cos(radLat)*math.sin(radLon)
    Z = (N*(1-e2)+h)*math.sin(radLat)
    XYZ = (X,Y,Z)
    return XYZ

def xyz2deg(X,Y,Z): #VALIDADO
    b = a*((1-e2)**0.5)
    elinha2 = e2/(1-e2)
    tanU = (Z/((X**2+Y**2)**0.5))*(a/b)
    senU = tanU/((1+tanU**2)**0.5)
    cosU = 1/((1+tanU**2)**0.5)
    radLat = math.atan((Z+(elinha2*b*(senU**3)))/((X**2+Y**2)**0.5-\
                    e2*a*cosU**3))
    radLon = math.atan(Y/X)
    N = ( a / ((1-(e2)*(math.sin(radLat))**2)**(0.5)))
    h = (((X**2+Y**2)**0.5)/math.cos(radLat))-N
    ddLat = math.degrees(radLat)
    ddLon = math.degrees(radLon)
    LatLonH = (ddLat,ddLon,h)
    return LatLonH

def radianError(phi1,lamb1,phi2,lamb2): #VALIDADO
    phiAvg = math.radians((phi2-phi1)/2)
    phi1 = math.radians(phi1)
    lamb1 = math.radians(lamb1)
    phi2 = math.radians(phi2)
    lamb2 = math.radians(lamb2)
    A = 1 + 3.0/4*e2 + 45.0/64*e2**2 + 175.0/256*e2**3 + 11025.0/16384*e2**4 +\
    43659.0/65536*e2**5
    B = 3.0/4*e2 + 15.0/16*e2**2 + 525.0/512*e2**3 + 2205.0/2048*e2**4 + \
    72765.0/65536*e2**5
    C = 15.0/64*e2**2 + 105.0/256*e2**3 + 2205.0/4096*e2**4 +\
    10395.0/16384*e2**5
    D = 35.0/512*e2**3 + 315.0/2048*e2**4 + 31185.0/131072*e2**5
    E = 315.0/16384*e2**4 + 3465.0/65536*e2**5
    F = 693.0/131072*e2**5
    phiA = A*(phi2-phi1)
    phiB = B/2 * (math.sin(2*phi2) - math.sin(2*phi1))
    phiC = C/4 * (math.sin(4*phi2) - math.sin(4*phi1))
    phiD = D/6 * (math.sin(6*phi2) - math.sin (6*phi1))
    phiE = E/8 * (math.sin(8*phi2) - math.sin(8*phi1))
    phiF = F/10 * (math.sin(10*phi2) - math.sin(10*phi1))

    M = (a*(1-e2)) / (1-e2*math.sin(phiAvg)**2)**(1.5)
    N = ( a / ((1-(e2)*(math.sin(phiAvg))**2)**(0.5)))

    Ssimples = M * (phi2-phi1)

    S = a*(1-e2)*(phiA - phiB + phiC - phiD + phiE - phiF) #Comrpimento de arco
                                                            #de Meridiano
    L = N*math.cos(phiAvg)*(lamb1-lamb2)#Comprimento de arco de Paralelo

    result = (S,L)
```

```python
    return result

LatLonriod = xyz2deg(Xriod,Yriod,Zriod)
LatLonriod2019 = xyz2deg(Xriod2019,Yriod2019,Zriod2019)
XYZmarco = deg2xyz(LatMarco,LonMarco,hMarco)


deltaX = Xriod2019-Xriod
deltaY = Yriod2019-Yriod
deltaZ = Zriod2019-Zriod

Xmarco2019 = XYZmarco[0] + deltaX
Ymarco2019 = XYZmarco[1] + deltaY
Zmarco2019 = XYZmarco[2] + deltaZ

LatLonMarco2019 = xyz2deg(Xmarco2019,Ymarco2019,Zmarco2019)

"""
FILE READING FUNCTIONS:

readNMEA
"""

def readNMEA(fileName):
    result = []
    searchFix = True
    searchSat = False
    numberSat = 0
    satellites = []
    checkGPSsat = []
    hTime = 0
    mTime = 0
    sTime = 0
    nlines = 0
    with open(fileName, 'r') as nmeaFile:
        for line in nmeaFile:
            line = line.translate(None, '$')
            dataArray = line.split(',')
            if len(dataArray) == 0: # Skip blank lines
                continue
            sentenceId = dataArray[0]
            if searchFix and sentenceId == "GPGGA": # Searching for GNSS fixes
                fixQuality = dataArray[6]
                if fixQuality[0] == '1': # Check if solution exists
                    numberSat = int(dataArray[7]) # Number of satellites in fix
                    hTime = int(dataArray[1][0:2]) # Hour of fix
                    mTime = int(dataArray[1][2:4]) # Minute of fix
                    sTime = int(dataArray[1][4:6]) # Second of fix
                    searchSat = True # Start looking for fix satellites
                    searchFix = False
                    nlines = 0
                    satellites = []
            if searchSat and sentenceId == "GPGSA": # Searching for number of
                                                     #GPS satellites in fix
                fixType = int(dataArray[2]) # Checks if fix exists
                if fixType in [2, 3]:
                    checkGPSsat += [int(x) for x in dataArray[3:15]]
```

```
                                               if x is not ''] # Add observed
                                                            #satellite numbers
                    satGPS = len(checkGPSsat)

            if searchSat and sentenceId == "GNGSA": # Searching for
                                                    #GLONASS satellites in fix
                fixType = int(dataArray[2]) # Checks if fix exists
                if fixType in [2, 3]:
                    satellites += [int(x) for x in dataArray[3:15]
                                              if x is not ''] # Add observed
                                                             #satellite numbers
                    satGL = len(satellites) - satGPS
                    nlines += 1 # Looks at second line of observed satellites

            if searchSat and sentenceId == "BDGSA": # Searching for
                                                    #Beidou satellites in fix
                fixType = int(dataArray[2]) # Checks if fix exists
                if fixType in [2, 3]:
                    satellites += [int(x) for x in dataArray[3:15]
                                              if x is not ''] # Add observed
                                                             #satellite numbers
                    satBD = len(satellites) - satGPS - satGL
                    nlines += 1 # Looks at second line of observed satellites:


            if searchSat and sentenceId == "GAGSA": # Searching for Galileo
                                                    #satellites in fix
                fixType = int(dataArray[2]) # Checks if fix exists
                if fixType in [2, 3]:
                    satellites += [int(x) for x in dataArray[3:15]
                                              if x is not ''] # Add observed
                                                             #satellite numbers
                    satGA = len(satellites) - satGPS - satGL - satBD
                    nlines += 1 # Looks at second line of observed satellites
                    if nlines == 5:
                        epoch = (hTime, mTime, sTime, numberSat, satellites,
                                 satGPS,satGL,satBD,satGA)
                        result.append( epoch )
                        if numberSat != len(satellites):
                            print "Incoherent satellite count at the line ", \
                            len(result), "\n"
                        searchSat = False
                        searchFix = True # Starts looking for next fix
        nmeaFile.close()
    return result

#return result


#rinex = readRINEX('SmoothingDebug.19o')

"""
NMEA FILE PROCESSING

This step reads the nmea file and calculates the error from the smartphone
solution.
"""
```

```
def nmea2deg(nmeafileName):
    result = []
    with open(nmeafileName, 'r') as nmeaFile:
        for line in nmeaFile:
            line = line.translate(None, '$')
            dataArray = line.split(',')
            if len(dataArray) == 0: # Pula linhas vazias
                continue
            sentenceId = dataArray[0]
            if sentenceId == "GPGGA":
                fixQuality = dataArray[6]
                #if fixQuality[0] != 'N' and fixQuality[1] != 'N':
                if fixQuality[0] != '0':
                    hTime = int(dataArray[1][0:2])
                    mTime = int(dataArray[1][2:4])
                    sTime = int(dataArray[1][4:6])
                    dLat = int(dataArray[2][0:2])
                    mLat = float(dataArray[2][2:])
                    dLon = int(dataArray[4][0:3])
                    mLon = float(dataArray[4][3:])
                    h = float(dataArray[9]) + float(dataArray[11])
                    sat = int(dataArray[7])
                    ddLat = dLat + mLat/60
                    ddLon = dLon + mLon/60
                    if dataArray[3] == "S":
                        ddLat = -ddLat
                    if dataArray[5] == "W":
                        ddLon = -ddLon
                    epoch = (hTime, mTime, sTime, ddLat, ddLon, h,sat)
                    result.append( epoch )
        nmeaFile.close()
        return result

def errorNMEA(nmeafileName):
    result = []
    nmea = nmea2deg(nmeafileName)
    sat = readNMEA(nmeafileName)
    for coord in nmea:
        error = radianError(LatLonMarco2019[0],LatLonMarco2019[1],
                            coord[3],coord[4])
        herror = coord[5] - hMarco
        epoch = (coord[0], coord[1], coord[2], error[0], error[1], herror)
        result.append( epoch )
    return result

"""
.POS FILE PROCESSING

This step reads a .POS file in XYZ format and calculates the error from
the survey
"""

def pos2xyz(fileName):
    result = []
    with open(fileName, 'r') as posFile:
        for line in posFile:
            if line[0] == '%':
                continue
            else:
                dataArray = line.split()
                if len(dataArray) == 0: # Pula linhas vazias
                    continue
                hTime = int(dataArray[1][0:2])
                mTime = int(dataArray[1][3:5])
                sTime = int(dataArray[1][6:8])
                Xpos = float(dataArray[2])
                Ypos = float(dataArray[3])
                Zpos = float(dataArray[4])
                epoch = (hTime, mTime, sTime, Xpos, Ypos, Zpos)
                result.append( epoch )
        posFile.close()
    return result

#latriod2019,lonriod2019
def errorPOSriod(fileName):
    result = []
    DDriod2019 = xyz2deg(Xriod2019,Yriod2019,Zriod2019)
    XYZpos = pos2xyz(fileName)
    for coord in XYZpos:
        DDpos = xyz2deg(coord[3],coord[4],coord[5])
        LatLonError = radianError(DDriod2019[0],DDriod2019[1],
                                  DDpos[0],DDpos[1])
        herror = DDpos[2] - DDriod2019[2]

        epoch = (coord[0], coord[1], coord[2],
                 LatLonError[0],LatLonError[1], herror)
        result.append( epoch )
    return result

def errorPOSsmartphone(fileName):
    result = []
    XYZpos = pos2xyz(fileName)
    for coord in XYZpos:
        DDpos = xyz2deg(coord[3],coord[4],coord[5])
        LatLonError = radianError(LatLonMarco2019[0],LatLonMarco2019[1],
                                  DDpos[0],DDpos[1])
        herror = DDpos[2] - hMarco

        epoch = (coord[0], coord[1], coord[2],
                 LatLonError[0], LatLonError[1], herror)
        result.append( epoch )
    return result

"""
ERROR ADJUSTMENT

This final step adjusts the NMEA coordinates to the POS file
"""
def nmea2xyz(nmeafileName):
    result = []
    nmeaDD = nmea2deg(nmeafileName)
    for coord in nmeaDD:
        XYZnmea = deg2xyz(coord[3],coord[4],coord[5])
        epoch = (coord[0], coord[1], coord[2],
                 XYZnmea[0], XYZnmea[1], XYZnmea[2])
        result.append( epoch )
    return result

def adjustmentNMEA(posFile,nmeafileName):
    result = []
    ref = pos2xyz(posFile)
    nmea = nmea2xyz(nmeafileName)
    for coord in nmea:
        for pos in ref:
            if pos[0] == coord[0] and pos[1] == coord[1] and \
            pos[2] == coord[2]:
                hTime = coord[0]
                mTime = coord[1]
                sTime = coord[2]
                Xerror = pos[3] - Xriod
                Yerror = pos[4] - Yriod
                Zerror = pos[5] - Zriod

                Xadjusted = coord[3] - Xerror
                Yadjusted = coord[4] - Yerror
                Zadjusted = coord[5] - Zerror
                LatLon = xyz2deg(Xadjusted,Yadjusted,Zadjusted)
                adjusted = (hTime, mTime, sTime, LatLon[0],LatLon[1],LatLon[2])
                result.append( adjusted )
    return result

def adjustmentErrorNMEA(posFile,nmeafileName):
    result = []
    nmea = adjustmentNMEA(posFile,nmeafileName)
    for coord in nmea:
        error = radianError(LatLonMarco2019[0],LatLonMarco2019[1],
                            coord[3],coord[4])
        herror = coord[5] - hMarco
        epoch = (coord[0], coord[1], coord[2],
                 error[0], error[1], herror)
        result.append( epoch )
    return result

"""
Statistical calculations
Calculates Averages, Root Mean Square errors and standard deviation.
"""
def averageNMEA(fileName):
    nmeaCoords = nmea2deg(fileName)
    sumLat = 0
    sumLon = 0
    sumH = 0
    for coord in nmeaCoords:
        sumLat += coord[3]
        sumLon += coord[4]
        sumH += coord[5]
    averageLat = sumLat/len(nmeaCoords)
    averageLon = sumLon/len(nmeaCoords)
    averageH = sumH/len(nmeaCoords)
    avgError = radianError(LatLonMarco2019[0],LatLonMarco2019[1],
                             averageLat,averageLon)
    avgHerror = averageH - hMarco
    result = (avgError[0],avgError[1],avgHerror)
    print result
    return result



def averagePOS(fileName):
    XYZpos = pos2xyz(fileName)
    sumLat = 0
    sumLon = 0
    sumH = 0
    for coord in XYZpos:
        DDpos = xyz2deg(coord[3],coord[4],coord[5])
        sumLat += DDpos[0]
        sumLon += DDpos[1]
        sumH += DDpos[2]
    averageLat = sumLat/len(XYZpos)
    averageLon = sumLon/len(XYZpos)
    averageH = sumH/len(XYZpos)
    avgError = radianError(LatLonMarco2019[0],LatLonMarco2019[1],
                             averageLat,averageLon)
    avgHerror = averageH - hMarco
    result = (avgError[0],avgError[1],avgHerror)
    print result
    return result



def rmsNMEA(fileName):
    RMSvector = errorNMEA(fileName)
    squareSumLat = 0
    squareSumLon = 0
    squareSumH = 0
    for coord in RMSvector:
        squareSumLat += (coord[3])**2
        squareSumLon += (coord[4])**2
        squareSumH += (coord[5])**2
    RMSLat = math.sqrt((squareSumLat)/(len(RMSvector)-1))
    RMSLon = math.sqrt((squareSumLon)/(len(RMSvector)-1))
    RMSH = math.sqrt((squareSumH)/(len(RMSvector)-1))
    result = (RMSLat,RMSLon,RMSH)
    print result
    print len(RMSvector)
    return result

def rmsAdjNMEA(posFile,fileName):
    RMSvector = adjustmentErrorNMEA(posFile,fileName)
    squareSumLat = 0
    squareSumLon = 0
    squareSumH = 0
    for coord in RMSvector:
        squareSumLat += (coord[3])**2
        squareSumLon += (coord[4])**2
        squareSumH += (coord[5])**2
    RMSLat = math.sqrt((squareSumLat)/(len(RMSvector)-1))
    RMSLon = math.sqrt((squareSumLon)/(len(RMSvector)-1))
    RMSH = math.sqrt((squareSumH)/(len(RMSvector)-1))
    result = (RMSLat,RMSLon,RMSH)
    print result
    print len(RMSvector)
    return result

def rmsPOS(fileName):
    RMSvector = errorPOSsmartphone(fileName)
    squareSumLat = 0
    squareSumLon = 0
    squareSumH = 0
    for coord in RMSvector:
        squareSumLat += (coord[3])**2
        squareSumLon += (coord[4])**2
        squareSumH += (coord[5])**2
    RMSLat = math.sqrt((squareSumLat)/(len(RMSvector)-1))
    RMSLon = math.sqrt((squareSumLon)/(len(RMSvector)-1))
    RMSH = math.sqrt((squareSumH)/(len(RMSvector)-1))
    result = (RMSLat,RMSLon,RMSH)
    print result
    print len(RMSvector)
    return result

def rmsRIOD(fileName):
    RMSvector = errorPOSriod(fileName)
    squareSumLat = 0
    squareSumLon = 0
    squareSumH = 0
    for coord in RMSvector:
        squareSumLat += (coord[3])**2
        squareSumLon += (coord[4])**2
        squareSumH += (coord[5])**2
    RMSLat = math.sqrt((squareSumLat)/(len(RMSvector)-1))
    RMSLon = math.sqrt((squareSumLon)/(len(RMSvector)-1))
    RMSH = math.sqrt((squareSumH)/(len(RMSvector)-1))
    result = (RMSLat,RMSLon,RMSH)
    print result
    print len(RMSvector)
    return result


def sdNMEA(fileName):
    avg = averageNMEA(fileName)
    nmeaError = errorNMEA(fileName)
    squareSumLat = 0
    squareSumLon = 0
    squareSumH = 0
    for coord in nmeaError:
        squareSumLat += (coord[3]-avg[0])**2
        squareSumLon += (coord[4]-avg[1])**2
        squareSumH += (coord[5]-avg[2])**2
    sdLat = math.sqrt(squareSumLat/len(nmeaError))
    sdLon = math.sqrt(squareSumLon/len(nmeaError))
    sdH = math.sqrt(squareSumH/len(nmeaError))
    result = (sdLat,sdLon,sdH)
    print result
    return result

def sdPOS(fileName):
    avg = averagePOS(fileName)
    nmeaError = errorPOSsmartphone(fileName)
    squareSumLat = 0
    squareSumLon = 0
    squareSumH = 0
    for coord in nmeaError:
        squareSumLat += (coord[3]-avg[0])**2
        squareSumLon += (coord[4]-avg[1])**2
        squareSumH += (coord[5]-avg[2])**2
    sdLat = math.sqrt(squareSumLat/len(nmeaError))
    sdLon = math.sqrt(squareSumLon/len(nmeaError))
    sdH = math.sqrt(squareSumH/len(nmeaError))
    result = (sdLat,sdLon,sdH)
    print result
    return result
"""
PLOTTING RESULTS

Outputs to excel spreadsheets
"""
def createExcel(fileName,data):
    cutfileName = fileName.split('.')
    excelfileName = cutfileName[0] + '.xlsx'
    workbook = xlsxwriter.Workbook(excelfileName)
    worksheet = workbook.add_worksheet()
    date_format_type = 'hh:mm:ss'
    date_format = workbook.add_format({'num_format': date_format_type,
                                       'align': 'left'})
    row = 0
    col = 0
    worksheet.write(row, col, 'HH')
    worksheet.write(row, col+1, 'MM')
    worksheet.write(row, col+2, 'SS')
    worksheet.write(row, col+3, 'Erro Latitude (m)')
    worksheet.write(row, col+4, 'Erro Longitude (m)')
    worksheet.write(row, col+5, 'Erro Altitude (m)')
    worksheet.write(row, col+6, 'HHMMSS')
    worksheet.write(row, col+7, 'Total de Satelites')
    row += 1
    for epoch in data:
        col = 0
        for item in epoch:
            worksheet.write(row, col, item)
            col += 1
#        time = str(epoch[0]) + ':' + str(epoch[1]) + ':' + str(epoch[2])
#        ymdtime = '0001-01-01 ' + time
#        date_time = datetime.strptime(ymdtime, '%Y-%m-%d %H:%M:%S')
        time = datetime(2019, 1, 1, epoch[0], epoch[1], epoch[2])
        worksheet.write_datetime(row, col, time, date_format)
        row += 1

    workbook.close()

"""
```

```
#1) DGNSS posição
rmsRIOD('riod0151_semifiltrado.pos')
rmsNMEA('20190115.txt')
rmsRIOD('riod0421_semifiltrado.pos')
rmsNMEA('20190211.txt')
rmsRIOD('riod0431_semifiltrado.pos')
rmsNMEA('20190212.txt')

#2) DGNSS observação
rmsPOS('20190115_Sta91500.pos')
rmsPOS('20190115_Sta91500_DGNSS.pos')
rmsPOS('20190115_Sta91500_Smoothed.pos')
rmsPOS('20190115_Sta91500_Smoothed_DGNSS.pos')
print
rmsPOS('20190211_Sta91500.pos')
rmsPOS('20190211_Sta91500_DGNSS.pos')
rmsPOS('20190211_Sta91500_Smoothed.pos')
rmsPOS('20190211_Sta91500_Smoothed_DGNSS.pos')
print
rmsPOS('20190212_Sta91500.pos')
rmsPOS('20190212_Sta91500_DGNSS.pos')
rmsPOS('20190212_Sta91500_Smoothed.pos')
rmsPOS('20190212_Sta91500_Smoothed_DGNSS.pos')


#3.2) SSP Com ajuste Sirgas2019
rmsNMEA('20190115.txt')
rmsPOS('20190115_Sta91500.pos')
rmsPOS('20190115_Sta91500_Smoothed.pos')
rmsPOS('20190115_Sta91500_Smoothed5sec.pos')
rmsPOS('20190115_Sta91500doppleravg_Smoothed.pos')
print
rmsNMEA('20190211.txt')
rmsPOS('20190211_Sta91500.pos')
rmsPOS('20190211_Sta91500_Smoothed.pos')
rmsPOS('20190211_Sta91500_Smoothed5sec.pos')
rmsPOS('20190211_Sta91500doppleravg_Smoothed.pos')
print
rmsNMEA('20190212.txt')
rmsPOS('20190212_Sta91500.pos')
rmsPOS('20190212_Sta91500_Smoothed.pos')
rmsPOS('20190212_Sta91500_Smoothed5sec.pos')
rmsPOS('20190212_Sta91500doppleravg_Smoothed.pos')

#4) Static
rmsPOS('20190115_Sta91500.pos')
rmsPOS('20190115_Sta91500_Static.pos')
rmsPOS('20190115_Sta91500_Smoothed_Static.pos')
print
rmsPOS('20190211_Sta91500.pos')
rmsPOS('20190211_Sta91500_Static.pos')
rmsPOS('20190211_Sta91500_Smoothed_Static.pos')
print
rmsPOS('20190212_Sta91500.pos')
rmsPOS('20190212_Sta91500_Static.pos')
rmsPOS('20190212_Sta91500_Smoothed_Static.pos')
```

```
Excel Plots
20190115 - January 15th 2019
#createExcel('20190115.txt',errorNMEA('20190115.txt'))
#createExcel('riod0151_semifiltrado.pos',errorPOSriod('riod0151_semifiltrado.pos'))
#createExcel('20190115_Sta91500.pos',errorPOSsmartphone('20190115_Sta91500.pos'))
createExcel('20190115_Sta91500_DGNSS.pos',errorPOSsmartphone('20190115_Sta91500_DGNSS.pos'))
#createExcel('20190115_Sta91500_Smoothed.pos',errorPOSsmartphone('20190115_Sta91500_Smoothed.pos'))
createExcel('20190115_Sta91500_Smoothed_DGNSS.pos',errorPOSsmartphone('20190115_Sta91500_Smoothed_D
#createExcel('20190115_Sta91500_Smoothed5sec.pos',errorPOSsmartphone('20190115_Sta91500_Smoothed5se
#createExcel('20190115_Sta91500dopplеravg_Smoothed.pos',errorPOSsmartphone('20190115_Sta91500dopple
#print

201902111 - February 11th 2019
#createExcel('20190211.txt',errorNMEA('20190211.txt'))
#createExcel('riod0421_semifiltrado.pos',errorPOSriod('riod0421_semifiltrado.pos'))
#createExcel('20190211_Sta91500.pos',errorPOSsmartphone('20190211_Sta91500.pos'))
createExcel('20190211_Sta91500_DGNSS.pos',errorPOSsmartphone('20190211_Sta91500_DGNSS.pos'))
#createExcel('20190211_Sta91500_Smoothed.pos',errorPOSsmartphone('20190211_Sta91500_Smoothed.pos'))
createExcel('20190211_Sta91500_Smoothed_DGNSS.pos',errorPOSsmartphone('20190211_Sta91500_Smoothed_D
#createExcel('20190211_Sta91500_Smoothed5sec.pos',errorPOSsmartphone('20190211_Sta91500_Smoothed5se
#createExcel('20190211_Sta91500doppleravg_Smoothed.pos',errorPOSsmartphone('20190211_Sta91500dopple
#print
201902112 - February 12th 2019
#createExcel('20190212.txt',errorNMEA('20190212.txt'))
#createExcel('riod0431_semifiltrado.pos',errorPOSriod('riod0431_semifiltrado.pos'))
#createExcel('20190212_Sta91500.pos',errorPOSsmartphone('20190212_Sta91500.pos'))
createExcel('20190212_Sta91500_DGNSS.pos',errorPOSsmartphone('20190212_Sta91500_DGNSS.pos'))
#createExcel('20190212_Sta91500_Smoothed.pos',errorPOSsmartphone('20190212_Sta91500_Smoothed.pos'))
createExcel('20190212_Sta91500_Smoothed_DGNSS.pos',errorPOSsmartphone('20190212_Sta91500_Smoothed_D
#createExcel('20190212_Sta91500_Smoothed5sec.pos',errorPOSsmartphone('20190212_Sta91500_Smoothed5se
#createExcel('20190212_Sta91500doppleravg_Smoothed.pos',errorPOSsmartphone('20190212_Sta91500dopple
"""
```