import os,sys

from math import sqrt

def mean(lst): #Calculates the mean of a list

 sum = 0

 for i in range(len(lst)):

 sum+=lst[i]

 return(sum/len(lst))

def stddev(lst): #Calculates the Standard Deviation of a list

 sum=0

 mn=mean(lst)

 for i in range(len(lst)):

 sum+=pow((lst[i]-mn),2)

 return sqrt(sum/(len(lst)-1))

os.chdir('/Users/Arduino/Documents/Arduino') #points to directory containing the terminal log file

print("CWD = :",os.getcwd())

CanLogfile=open('TestlogMol.log','r') #opens the terminal log file

lines=CanLogfile.readlines() #reads data into variable

CanLogfile.close() # closes Cansat file

Tempfile=open('TempData.txt','w+') #opens the TempData file we will be using to work with the data

Tempfile.writelines(lines[3:-3]) # puts data from Cansat file into TempData but removes first 5 and last five lines (headers)

Tempfile.seek(0) # puts file pointer back to start of file

i=0

maxim=0

pressureI=[]

hours=[]

minutes=[]

seconds=[]

millisecs=[] #array that will hold the pressure data in order to calc and remove outliers

temprI=[] #array that will hold the temp data in order to calc and remove outliers

pressure=[] #array that will hold the tidied up pressure data

time=[] #array that will hold the tidied up time data data

tempr=[] #array that will hold the tidied up temp data

altMPX=[] #array that will hold the tidied up altitude data

velBMP=[]

x=[]

y=[]

z=[]

F=[]

tempr2=[]

pressure2=[]

altBMP=[]

humid=[]

lat=[]

long=[]

velG=[]

altG=[]

newsec=[]

spower=[]

velBMP.append(float(0))

#for Templine in Tempfile:

# TemplineSplit=Templine.split()

# print("len(TemplineSplit) = ",len(TemplineSplit))

# if len(TemplineSplit)==32:

# print("len(TemplineSplit) = ",len(TemplineSplit))

# checksum=0

# for count in range (1,31,2):

# checksum = checksum+float(TemplineSplit[count])

# print("TS[",count,"] = ",TemplineSplit[count])

# print("checksum = ",checksum)

# if checksum == float(TemplineSplit[31]):

# #print("checksum = ",checksum)

# #print(" float(TemplineSplit[31]) = ",float(TemplineSplit[31]))

# print("Data good")

# else:

# #print("checksum = ",checksum)

# #print(" float(TemplineSplit[31]) = ",float(TemplineSplit[31]))

# print("Data corrupt")

Tempfile.seek(0)

for Templine in Tempfile: # goes through tempfile line by line

 TemplineSplit=Templine.split() #splits each line of data separated by spaces into individual strings

 if len(TemplineSplit)==32: # should be 30 strings - if not ignore line

 pressureI.append(float(TemplineSplit[3])) #puts pressure data into array

 temprI.append(float(TemplineSplit[1])) #puts temp data into array

Tempfile.seek(0) #puts file pointer back to start of file

for Templine in Tempfile:

 TemplineSplit=Templine.split() # goes through lines one by one

 if (len(TemplineSplit)==32): # and ignores lines that are incomplete or too long

 pressureval=float(TemplineSplit[3]) #converts string to floating point and stores the Pressure data in an array

 temprval=float(TemplineSplit[1])

 currenttime=TemplineSplit[23];

 if (mean(pressureI)-2\*stddev(pressureI)<pressureval<mean(pressureI)+2\*stddev(pressureI))and(mean(temprI)-2\*stddev(temprI)<temprval<mean(temprI)+2\*stddev(temprI)and float(currenttime)>1000000):

 # filters out lines where the MPX4115 pressure and thermistor temp data are extreme and where the time data is not accurate

 tempr.append(round(temprval,2)) #puts Thermistor temp into array

 pressure.append(float(TemplineSplit[3])) #puts MPX4115 pressure into array

 altMPXv=44307.6\*(1- pow((float(TemplineSplit[3])/1013.25), 0.190284)) # Calc's Altitude from Pressure

 altMPX.append(round(altMPXv,2)) #puts Altitude (MPX4115) into array

 x.append(float(TemplineSplit[5])-0.16) #puts Accelerometer X gForce into array

 y.append(float(TemplineSplit[7])) #puts Accelerometer Y gForce into array

 z.append(float(TemplineSplit[9])) #puts Accelerometer Z gForce into array

 RF=sqrt((pow(float(TemplineSplit[5])-0.16,2)+pow(float(TemplineSplit[7]),2)+pow(float(TemplineSplit[9]),2))) #Calc's Resultant force

 F.append(RF) #puts Accelerometer Resultant gForce into array

 tempr2.append(float(TemplineSplit[11])) #puts BMP180 Temp data into array

 pressure2.append(float(TemplineSplit[13])) #puts BMP180 Pressure data into array

 altBMP.append(float(TemplineSplit[15])) #puts BMP180 Altitude data into array

 humid.append(float(TemplineSplit[17])) #puts Relative Humidity data into array

 lat.append(TemplineSplit[19]) #puts GPS Latitude data into array

 long.append(TemplineSplit[21]) #puts GPS Longitude data into array

 time.append(TemplineSplit[23]) #puts GPS Time data into array

 velG.append(float(TemplineSplit[25])) #puts GPS Velocity data into array

 altG.append(float(TemplineSplit[27])) #puts GPS Altitude data into array

 spower.append(float(TemplineSplit[29])) #puts SolarPanel Power data into array

 if len(currenttime)==8: #GPS Time data will either be of length 8(Hr = single digit) or 9(Hr = double digit)

 hours.append(int(currenttime[0:1]))

 minutes.append(int(currenttime[1:3])) #splits up the GPS time data into Hrs, Mins, Secs, mSecs

 seconds.append(int(currenttime[3:5]))

 millisecs.append(int(currenttime[5:8]))

 else:

 if len(currenttime)==9: #GPS Time data will either be of length 8(Hr = single digit) or 9(Hr = double digit)

 hours.append(int(currenttime[0:2]))

 minutes.append(int(currenttime[2:4]))

 seconds.append(int(currenttime[4:6]))

 millisecs.append(int(currenttime[6:9]))

 newsec.append(round(hours[i]\*3600+minutes[i]\*60+seconds[i]+millisecs[i]/1000,2)) #newsec = GPS time in mS

 if i>0:

 deltatime=round(newsec[i]-newsec[i-1],2) #calculates the time between time readings (loop time)

 velBMPvalue=round((altBMP[i]-altBMP[i-1])/(deltatime),4) #calc's the Velocity using BMP180 Altiude data

 velBMP.append(round(velBMPvalue,2)) #puts BMP180 Velocity data into array

 i=i+1

filesize=len(pressure) #calc's number of lines of clean data

print("filesize = ",filesize);

cleandata=open('CleanData.txt','w') #opens Cleandata file for writing

cleandata.write("Time TmpEx PressMP Alt1 Vel1 Xg Yg Zg RgF TmpIn PresBMP Alt2 RelHum Lat Long TimeGPS Vel2 Alt(GPS) Hr Min Sec mS SolarPr\n")

for i in range(0,filesize-1):

 timediff=round(newsec[i]-newsec[0],2) #calcs the time between readings (loop time)

 cleandata.write(str(timediff))

 cleandata.write(" ") # all data is now written line by line into the Cleandata file

 cleandata.write(str(tempr[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(pressure[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(round(altMPX[i]-altMPX[0],2))) # |

 cleandata.write(" ") # |

 cleandata.write(str(velBMP[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(round(x[i],2))) # |

 cleandata.write(" ") # |

 cleandata.write(str(y[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(z[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(round(F[i],2))) # |

 cleandata.write(" ") # |

 cleandata.write(str(tempr2[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(pressure2[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(round(altBMP[i]-altBMP[0],2))) # |

 cleandata.write(" ") # |

 cleandata.write(str(humid[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(lat[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(long[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(time[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(round(velG[i],2))) # |

 cleandata.write(" ") # |

 cleandata.write(str(round(altG[i]-altG[0],2))) # |

 cleandata.write(" ") # |

 cleandata.write(str(hours[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(minutes[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(seconds[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(millisecs[i])) # |

 cleandata.write(" ") # |

 cleandata.write(str(spower[i])) # \/\/

 cleandata.write("\n")

cleandata.close()