```
# file for python3; ADI pack from CSV.                                  4/03/2024 JRJ
# input file loads data structures, thus program is changeable
import ADIsensaiMod20a as cSub
from dataDriver2b import zneS
# line 1 is a title and description, of CSV- ignored.
# lines 2,3 have the new ADI tags, comma separated- extra , at end
# line 4 has the data types, thus 2-4 need to be read into storage
# line 5 is blank for read-ability (standard). at h1 the data begins
# line 6-7 (ADI headers, userdef) transfer direct to the output file
# global constants:

tags,dtyp,ardyLd,wd,h1 = 3,4,False,49,8          #                              h1 n
hdr,year,fiNo,NoMat,ver = 6,2024,0,24,'33cF'     #default year if missing somewhere
#cols = [0,3,4,8,9,10,12,13,14,31,33,35,40,41]  #sm pack sequence -1, see library
QScol,Tm4Col,ZnCol = cSub.cols[0],cSub.cols[3],cSub.cols[9]
datyps,dtags,selL = [None]*49,[None]*49,cSub.selL   #load as blanks? copy selL in
library
line, n, m3, lend = None, 0, 0, '\r\n'           #m3 not used; can above be null
strings?
dtfile = 'ADI headings fin49o.csv'     #filenames and selector moved to library?
dtfile2,qc = 'miniMyL4OM.adi',len(cSub.cols)    #dept,corr,myJoin,dmpLen,spMe,addTy
(below flags)
fl4,fl0,fl3=(False,False,True,False,True,False),(True,True,False,True,True,False),
(False,True,False,True,True,False)
fl1,fl2,fl5=(False,True,True,False,True,False),(True,False,False,False,True,False),
(True,True,True,True,True,True)
flS,eor = [fl0,fl1,fl2,fl3,fl4,fl5],'<eor>'  #flag selLector moved to library

# maybe print can be diverted; g, is opened, or via a pipe. output records higher
(newer) than a #
# logging to the standard output device (tty)? convert this into a usable module for
both pak & unpak
```

```
26 def add_mini(s):             #user my_data, not in cSub
27     return s+frag1[0]+' '    #from file
28
29 def vald_znChg(z):
30     k1 = (z<=0) or (z>=NoMat)   #for 2 places below, next & next
31     return (not k1)
32
33 def roll_warn(qn,ts,zs):    #caller uses row list, subs above
34 #    qn = row[QScol]          #preserves an error reminder
35     th = ts[0:2]             #take the left 2 off, assume 4 wide
36     h = int(th)              #convert to an integer                          h local
37     dt = zneS.get(zs,NoMat) #dictionary
38     #try dt from letter zones; last chance
39     if dt == NoMat:  dt = cSub.getHr(zs)
40     if dt == NoMat:  print (40,'  flag in zone translate- value not in list')
41     wrn = False              #default, none
42     if vald_znChg(dt):
43         wrn = cSub.roll_ck(h,False,dt)  #non-0 forces false; next dt=4 & True
44         nt1 = cSub.adv_tm1(ts,dt)       #only adjusts time -100 possible
45     else:  nt1 = (int(ts),False)        #moved
46     if wrn:  print(46,'  adjustment forces a date roll2- QSO#',qn)     #log via
   wrt_1linA
47     print (47,'in warn- old & new',ts,nt1,'for QSO',qn,'modulo taken',lend) #output
   check
48     return (str(nt1[0]),wrn,dt,nt1[1])  #do not modulo the time, nt1 MUST be 4 ch?
49
50 def process_1row(m2):       #runs under col counter. read rec & split up
51
52     def trans_xlt(v,p,g):   #translation routine, value type tag- maybe to library?
53         #if true/false convert to Y N. convert case, maybe not?
54         if v == 'TRUE':
55             v = "Y"         #works ok now
```

```
56              return (v,p,g)
57          if v == 'FALSE':    #most of these are e(numerated) new p
58              v = "N"         #boolean comes from Sheets
59              return (v,p,g)
60          if dmpLen:          #add tag time_off
61              if g=='QSO_len':
62                  if v == "":  print(62,'  missing QSO length- cant convert')
  #log
63                  else:
64                      stt=cSub.tme_splt(row[Tm4Col])   #echo check stt
65                      oft=cSub.solv_tof(stt[0],stt[1],int(v))
66  #                   print(stt,v,oft)                 #look for null somewhere
67                      v=oft[0]            #check day flag- oft[1]
68                      if oft[1]:  print (68,'  flag in trans_xlt- day rolled over')
  #log
69                      p,g='t','time_off'               #add 8 to val, replace more input
70          # pair are exclusive
71          if ((g=='QSO_date') and rw[1]):
72              nwdat = cSub.roll_1day(v)                #process date roll, zone change
73              print(73,'  in date subs.
  old1',v,'new1',nwdat[0],'year',nwdat[1],'month',nwdat[2])  # +2 flags
74              v = nwdat[0]                        #wrn #1 flag or rw[1]?
75          return (v,p,g)                          #Value, tyPe, taG; p already t
76
77      row = cSub.line.split(',')   #this is a list now, starts at 0
78      rw = roll_warn(row[QScol],row[Tm4Col],row[ZnCol])  #make sure 0 is 4 characters (+
  leading 0?)
79      if ((vald_znChg(rw[2]) and corr)):      #substitute replacements flag, old
80          row[ZnCol] = "UTC"                  #see corr flag, future
81          row[Tm4Col] = rw[0]                 #tuple out, #1 cks time roll
82      lnbuf = ''              #load empty line buffer
83      if myJoin:  lnbuf = add_mini(lnbuf)     #moved
```

```
    m = m2                  #start m while loop, m1 replaces msav         m local
    while m > 0:            #preset col loop and decrement last
        o = m2-m                                                          #o local
        if not dept:  o = cSub.cols[o]      #val,typ,tag = row[o],datyps[o],dtags[o]
#        print('debug1',o,row[o])
#        print('debug3',o,dtags[o])         #it was line!
#        print('debug2',o,datyps[o])

        xl = trans_xlt(row[o],datyps[o],dtags[o])   #a tuple back
        val,typ,tag=xl[0],xl[1],xl[2]               #reload
        if val == '':               #if null skip the field, or tag!
            #print('skipping col.',o,'null')     #too verbose
            if tag == 'country':    #warn if a blank country
                print (97,'            missing country skipping',n)
                    #log the record skipped!
                return ""
#            else:   pass
        else:       #exists- form group, opt to add data type
            thisCol = cSub.ele_pak(tag,val,typ,spMe,addTy)   #1st boolean +space
            lnbuf = lnbuf+thisCol   #inner2 sum group onto line
        m += -1      #countdown, 1 is last; data type above?              m local
        pass         #end while loop
#    if myJoin:  lnbuf = add_mini(lnbuf)     #moved to initialize
    lnbuf = lnbuf+eor       #config info line added here & global constant
    return lnbuf+lend

#def processLn1():                          gets tuple back, no longer used
#    return process_1row(qc)                partial a,d,e,i,j,k,m,n,o,af,ah,aj,ao,ap

print('  opening data files,','Prog1 version:',ver)   #id's the python version, log
too
```

```
dept,corr,myJoin,dmpLen,spMe,addTy = flS[selL][0],flS[selL][1],flS[selL][2],flS[selL][3],flS[selL][4],flS[selL][5]
nwMd = cSub.ovrrFlgs(1,selL)                #call new manual update subr
if crs < 0: pass        #add interm var!
else:                       #reset the flags listed above; copy sel from library
    crs = rtnPflgs(1)       #define flags
    dept,corr,myJoin,dmpLen,spMe,addTy = crs[0],crs[1],crs[2],crs[3],crs[4],crs[5]
    print(119,'command changes accepted, code:',nwMd[1])    #provide notice of change? conf code?
    selL = nwMd[0]          #copy local from global update for file open
print(121,'please- no commas in the data fields!')  #file handles for data to be transformed
e = open(dtfile2,'r')       #new file miniMy                                e f global
f = open(dtfile,'r')        #make the name a parameter
cfig = e.readline()         #read in my_data fragment                       e global
frag1 = cfig.split(',')     #remove trailing newline
print (frag1[0])            #echo check, below pgm adds the flag setting after X
ufiNo = input('output filenumber X?: ')   #can use default? maybe add flag set?
g = cSub.opn_outp(ufiNo,selL)             #open in append mode; h logging g global
if dept:  print('            full conversion,',selL,'other flags:',corr,myJoin,dmpLen,':my_data above:')
else:     print('            partial conversion,',selL,'other flags:',corr,myJoin,dmpLen,'shorter form:')

while True:
    cSub.lastline = cSub.line    #save for concat in lod_tags
    cSub.line = f.readline()     #is this a string? maybe...
# not sure how it returns a boolean & data value too?
    n += 1                  #pre-increment 0                                n global
    if not cSub.line: break #EOF, see book on boolean issue
    if n==1:                #line 1
```

```
        pass                #skips the title, no use here
    elif n==tags:           #line 3
        tagSrc = False      #concatinate file lines 2-3 @3 next; must be first!
        if tagSrc:  dtags = cSub.lod_tagS() #from file? warn if different lengths?
        else:               #eventually add data types to this
            dtags = cSub.tgStr  #subs QSO_len 4time_off
            dtags[12]='QSO_len' #exception CSV-adi; move 12 to library
        print (dtags)
    elif n==dtyp:
        datyps = cSub.lod_dtypes()  #at 4, or use tpStr w/ tagSrc flag
        print (datyps)              #echo check both d's
        ardyLd = True
    elif (n==hdr or n==(hdr+1)):
        print(152,"            a header line",n,'transferred')
        g.write(cSub.line)  #send the ADI headers, 2 lines              g global
    elif n==h1: print (154,"            start of new data- process @line")
    else:  pass             #skips line 2 and 5
# after line 7 (and the <eoh>) are data & maybe blank lines
    if n>=h1:               #in data section                            h1 global
        if cSub.line == '':      #also see break, above
            pass            #blank test python3; output a message?
        elif cSub.line == '\n':  #appended to a line, except EOF
            print(161,'            ignore blank',n)
# docs say new-line is appended to read if not EOF, which it is
        else:
#           print('cSub.line',n,'record',n-7)   #echo check before      n global
            print(cSub.rem_newln(cSub.line))    #works!
            if ardyLd:
                if dept:    cSub.line = process_1row(wd)    #no need to pass n?
                else:       cSub.line = process_1row(qc)
            else:  cSub.line = ''
        g.write(cSub.line)      #overwrites read-in line; skip if lower QSO#
```

```
172     else:  pass             #exit loop next (unreachable here)
173
174 e.close()                   #close datas                                  e,f,g global
175 f.close()
176 g.close()                       #close output
177 print(176,'closed all files')   #subroutine testing moved
```