Read raw data (FFTS format)

```
import numpy as np
import os

def getRawData(filename,metainfo=None):
    """read raw data from FFTS into numpy record arrays
    metainfo ... if provided (e.g. if known from a previous file
    of the same type, this prevents parsing it again

    returns alldata,metainfo
    alldata contains the following fields (use alldata['FIELD'] to access)
    IEEE, DATFMT, DATALENGTH, BE_IDENT, TIMESTAMP, INTTIME, PHASENUM,
BENUMSEC,
    BLOCKINGFACTOR, DATABLOCK
    DATABLOCK is itself a record, containing:
    SECTION, NUMCHANNELS, DATA

    to access the raw data use 'alldata['DATABLOCK']['DATA']' which is a 3D
array,
    of shape (dump,baseband,specchannel), e.g., to access first dump, first
baseband
    spectrum 'alldata['DATABLOCK']['DATA'][0,0]', last dump, second
baseband:
    'alldata['DATABLOCK']['DATA'][-1,1]', etc.

    use:

        import dateutil.parser
        alltimestamps=[dateutil.parser.parse(a) for a in
alldata['TIMESTAMP']]
        allmjd=np.array([cal_mjd(dt.month,
dt.day+(dt.hour+dt.minute/60.+dt.second/3600.+dt.microsecond/3.6e9)/24.,
                                 dt.year) for dt in alltimestamps])

    to compute MJD values from the timestamps

    """
    totalsize=os.path.getsize(filename)
    if metainfo is None:
        # getting endian
        ieee=np.fromfile(filename , dtype = np.dtype([('IEEE','|S4')]) ,
count = 1 )
        endian='>' if ieee=="IEEE" else '<'
        metainfo=np.fromfile(filename , dtype = np.dtype([ ('IEEE','|S4'),
('DATFMT','|S4'),
        ('DATALENGTH','%si4'%endian), ('BE_IDENT','|S8'),
('TIMESTAMP','|S28'),
        ('INTTIME','%si4'%endian), ('PHASENUM','%si4'%endian),
('BENUMSEC','%si4'%endian),
```

```
        ('BLOCKINGFACTOR','%si4'%endian), ('SECTION','%si4'%endian),
        ('NUMCHANNELS','%si4'%endian) ]) , count = 1 )
    else:
        endian='>' if metainfo['IEEE'][0]=="IEEE" else '<'
    #
    intorfloat=np.int32 if metainfo['DATFMT'][0]=="I" else np.float32
    datfmt=('%si4'%endian) if metainfo['DATFMT'][0]=="I" else
('%sf4'%endian)
    datalength=metainfo['DATALENGTH'][0]
    numchannels=metainfo['NUMCHANNELS'][0]

    datablock=np.dtype([('SECTION','%si4'%endian),
('NUMCHANNELS','%si4'%endian),
    ('DATA',datfmt,numchannels)])
    record=np.dtype([ ('IEEE','|S4'), ('DATFMT','|S4'),
('DATALENGTH','%si4'%endian),
    ('BE_IDENT','|S8'), ('TIMESTAMP','|S28'), ('INTTIME','%si4'%endian),
    ('PHASENUM','%si4'%endian), ('BENUMSEC','%si4'%endian),
('BLOCKINGFACTOR','%si4'%endian),
('DATABLOCK',datablock,metainfo['BLOCKINGFACTOR'][0]*metainfo['BENUMSEC'][0]
)  ])
    record_length=record.itemsize
    num_records=totalsize/record_length
    alldata=np.fromfile(filename , dtype =record, count = num_records )
    return alldata,metainfo
#

fname="/be4-daten/AFFTS/2788.0001"
alldata,metainfo=getRawData(fname)

print alldata['DATABLOCK']['SECTION']
print alldata['DATABLOCK']['NUMCHANNELS']
print alldata['DATABLOCK']['DATA'].shape
```

If you want to convert the timestamps to MJD, you can do the following:

```
import dateutil.parser

def cal_mjd(mn,dy,yr):
    """calculate MJD from a data
    mn ... month (int)
    yr ... year (int)
    dy ... day (float, i.e. including fractions of a day)

    returns MJD

    (code snippet adapted from pyephem, which is using libastro)"""
    m=int(mn)
    y=int(yr)
    if y<0: y+=1
```

```
    if mn<3: mn+=12
    if mn<3: y-=1
    b=0
    if not ( (yr < 1582) | ((yr == 1582) & ((mn < 10) | ((mn == 10) & (dy <
15))))):
        a=y//100
        b=2 - a + a//4

    c=0
    if y<0:
        c=int((365.25*y) - 0.75) - 694025L
    else:
        c=int(365.25*y) - 694025L
    d=int(30.6001*(m+1))
    return b + c + d + dy - 0.5  +15020L -0.5
#

alltimestamps=[dateutil.parser.parse(a) for a in alldata['TIMESTAMP']]
allmjd=np.array([cal_mjd(dt.month,
dt.day+(dt.hour+dt.minute/60.+dt.second/3600.+dt.microsecond/3.6e9)/24.,
                         dt.year) for dt in alltimestamps])
```