ASAP advanced tutorial

Main goal: To understand how to handle the data manually by building a quotient without using mx_quotient

TODO:

1. Load the data from RPFITS file
2. Play with various methods to get access to the data
3. Run the reduction script using the built-in quotient
4. Copy the template script doing the quotient
5. Modify makeQuotient method to replicate the functionality of the built-in mx_quotient

Prerequisite: Data file and a general reduction script:

Data file: 2009-04-02_2240_MMB-MX-11.9-0.13333.rpf

Script 1: tutorial.py general reduction script

Script 2: systemquotient.py script doing built-in quotient (to be copied and modified)

Script 3: userquotient.py the result (you're not supposed to look into it unless desperate)

Help: Use **help method** in asap to get information about parameters, etc

Some instructions

To run the script

```
ASAP> run tutorial.py
```

Copy systemquotient.py to a new name, change the import statement at the top of tutorial.py to load your file instead of the systemquotient.py.

```
from myquotient import *
```

Now your version of makeQuotient will be called from tutorial.py.
Use the following to load the data into a scan table called sc

```
ASAP> sc=scantable('2009-04-02_2240_MMB-MX-11.9-0.13333.rpf')
```

Inspect the content.

```
ASAP> sc.summary()
```

The file contains 7 scans. Each scan corresponds to observations of a source with a different beam of a 7-beam receiver. We want to use the median spectrum from all other scans as a reference when constructing the quotient using the formula

$$Result = T_{off} * (On/Off - 1),$$

where T_{off} is the system temperature measured during the reference scan, On is the signal spectrum, Off is the reference spectrum. We want to construct *Result* for each individual beam (out of 7 beams available) and average all these spectra together.

To do the selection

```
ASAP> sel=sc.get_selection()
ASAP> sel.set_beams(0)
ASAP> sel.set_scans(0)
ASAP> sel.set_polarisations(0)
ASAP> sc.set_selection(sel)
ASAP> selected_sc = sc.copy()
ASAP> selected_sc.summary()
```

To average or compute the median use one of

```
ASAP> ref.average_time(weight='median')
ASAP> scans.average_time()
```

To merge the scans together (i.e. individual quotients for each beam) build a python list first and then use merge and average_time

```
res=[]
for b in range(7):
    res.append(myFunctionReturningAScanTable(beam))
averaged_scantable = average_time(merge(res))
```

Note that merge will fail if the list has only 1 element. An if-statement may be necessary

To scale the scantable with the constant factor use

```
scan.scale(factor,tsys=False, insitu=True)
```

To add a constant use

```
scan.add(constant_to_add, insitu=True)
```

To get the tsys use

```
scan.get_tsys()
```

Note that some selection of data is usually necessary. Otherwise get_tsys() returns too many numbers.
To divide two spectra simply divide one scan table to another

```
quotient = signal / ref
```