Basic exercises

February 13, 2006

The portal HOWTO page

Throughout these and the other questions references are made to HOWTOs on our webpages. The HOWTO section can be reached from the main portal page:

http://portal.astro-wise.org/

by clicking on "Howto's" in the bar.

Using CVS to get code anonymously

Question 1: Use CVS to get a personal copy of the astro-wise pipeline software.

(See the HOWTO "General -> Getting started"). The password for anonymous CVS is 'astrowize'.

```
Answer:
cvs -d :pserver:anoncvs@cvs.astro-wise.org:/cvsroot login
cvs -d :pserver:anoncvs@cvs.astro-wise.org:/cvsroot checkout opipe
```

Question 2: Set an environment variable "AWEPIPE" to point to the code.

Note that by default you will be using the "csh" shell and that is what this question assumes.

```
Answer:
Edit your ~/.cshrc file and add the following line:
setenv AWEPIPE /path/to/your/opipe
And if not there yet, also add the following line
(check with your local AWE representative if pathname is correct):
source ~aworacle/bin/coraenv

Now "source ~/.cshrc" to update your environment.
```

Setting up your environment (username/password)

Question 3: Setup your ∼/.awe/Environment.cfg file.

It is necessary to obtain a database account (username/password) in order to get write access ('commit') to the database. This information should be stored as follows:

In your home directory make a directory '.awe' (starting with a dot) and in this subdirectory make a file called 'Environment.cfg' which contains the following lines:

```
[global]
database_user: <your database username>
database_password: <your database password>
```

Then make this file readable only for yourself:

```
chmod a-rwx,u+rw Environment.cfg
```

Test by opening the astro-wise prompt that your username/password are used:

```
awe
```

If the welcome message ends with a message such as this:

```
Current profile:
- username : awehelmich
- database : aw01.omegarac.nova.aw
- project  : WORKSHOP2005

awe>
```

you now have access.

```
Answer: see above
```

The astro-wise/python documentation

The procedures above are also explained on the web Howto's. There are several places where one can find help and documentation:

- The portal HOWTO section
- The Astro-Wise User Manual 'General User and Development Manual', available at the portal DOCUMENTS section
- docstrings throughout the code, readable with the "help" built-in function

Question 4: Display the help file (docstring) of BiasFrame.

```
Answer:
awe> help(BiasFrame)

Hit "q" to exit the help page.
```

At the Astro-Wise prompt; looking around

Question 5: Start awe and print all classes/variables known in the main so-called namespace

Most of these classes are representations of real-world FITS files and their meta data, or objects containing the solution of astrometry, photometry etc.

```
Answer:
awe> dir()
```

Question 6: Look through the list and then display the namespace for builtins (builtin functions and classes for Errors, Warnings)

The built-in functions of Python are functions such as 'range', 'len', 'int', 'float', 'dir' and 'help' that are used throughout our code, and can be useful in an interactive awe session.

```
Answer:
awe> dir(__builtins__)
```

Question 7: Display the namespace of Chip.

This may give you some insight in methods and properties of Chip, which is a representation of a single CCD in an instrument. In the case of WFI there are 8 Chip objects in the database; one for each CCD.

```
Answer:
awe> dir(Chip)
```

Querying the database from Python

Now that we have an open awe session and are connected to the database with our database username and password, we can query the database from here. See the HOWTO on the portal webpage (Python -> Queries) for more information. For these execises it can be helpful to use "tab" a lot for completion of names. You may need to use commands shown in the first questions to figure out the names of properties that you need to query on later.

Question 8: Which properties of RawTwilightFlatFrames are stored in the database?

```
Answer:
awe> RawTwilightFlatFrame.get_persistent_properties()
['DATE', 'DATE_OBS', 'EXPTIME', 'LST', 'MJD_OBS', 'OBJECT', 'OBSERVER', 'UTC',
'chip', 'extension', 'filename', 'filter', 'globalname', 'imstat',
'instrument', 'is_valid', 'object_id', 'observing_block', 'overscan_x_stat',
'overscan_y_stat', 'prescan_x_stat', 'prescan_y_stat', 'process_status',
'quality_flags', 'raw_fits_data', 'template']
```

Question 9: In this list 'chip' points to another class. Which properties of this class are stored in the database?

```
Answer (1):
awe> RawTwilightFlatFrame.chip.get_persistent_properties()
['NAXIS1', 'NAXIS2', 'OVSCX', 'OVSCY', 'PRSCX', 'PRSCY', 'name', 'object_id',
'orientation']
```

```
Answer (2):
awe> Chip.get_persistent_properties()
['NAXIS1', 'NAXIS2', 'OVSCX', 'OVSCY', 'PRSCX', 'PRSCY', 'name', 'object_id',
'orientation']
```

Question 10: Which instruments are known in the database?

This can be done in several ways. One way is to query for all instruments of which the name is not equal to an empty string. Alternatively you could use the 'like' functionality to ask for all Instruments for which the name is equal to any string.

```
Answer (1):
awe> q = Instrument.name != ''
awe> for i in q: print i.name
Answer (2):
awe> q = Instrument.name.like('*')
awe> for i in q: print i.name
```

Question 11: Execute the following line in awe to select a particular ReducedScienceFrame from the database.

```
awe> sci = ReducedScienceFrame.filename.like('Sci*ccd52*51360.fits')[0]
```

Question 12: For this ReducedScienceFrame, retrieve the image itself and the bias image that was used. The FITS images will be saved to the directory where awe was started.

```
Answer:
awe> sci.retrieve()
awe> sci.bias.retrieve()
```

Question 13: For the ReducedScienceFrame of the previous exercise print the observing date, the object, instrument, filter and chip (CCD) name.

```
Answer:
awe> print sci.DATE_OBS, sci.OBJECT, sci.instrument.name, sci.filter.name,
sci.chip.name
# Or
awe> print sci.DATE_OBS
awe> print sci.OBJECT
# etc.
```

Question 14: For the ReducedScienceFrame selected, print the filenames of the RawBiasFrames used in the creation of the master bias that was used in debiassing the ReducedScienceFrame.

```
Answer:
awe> for frame in sci.bias.raw_bias_frames: print frame.filename
```

Question 15: What about the exposure levels of the raw twilight flats that were used?
Note that the flat field used in the reduction is a MasterFlatFrame which is a combination of a master dome (DomeFlatFrame) and master twilight (TwilightFlatFrame) flat.

```
Answer:
awe> for frame in sci.flat.twilightflat.raw_twilightflat_frames:
print frame.imstat.median
```

System calls from awe

It is possible to do system calls from awe. The "os" module is imported by default and can be used to run commands as follows:

```
awe> os.system('ls')
awe> os.system('pwd')
```